E/S simples con la tarjeta DAQ LabJack U3-HV

Carlos Tomás Medrano Sánchez, Juan Carlos García López, Guillermo Palacios Navarro, Inmaculada Plaza García.

Objetivos: El objetivo de la práctica es conocer y utilizar las E/S simples que tenemos en la tarjeta DAQ. Para acceder a ella utilizaremos Python, que nos permite acceder a alto nivel al driver de la tarjeta. Utilizaremos un simulador de la tarjeta en una máquina virtual.

Trabajo previo: Repasar la práctica 0. Buscar información en internet para conocer las características básicas de la tarjeta (o echar un vistazo al Manual de Usuario en moodle). Se recomienda leer antes la práctica y pensar con antelación las soluciones.

1. Acceso general al simulador de la tarjeta.

Como **paso previo** nos conectaremos a la máquina virtual (ver práctica 0).

El acceso general al simulador de la tarjeta se realiza como se indica en el vídeo adjunto de la práctica 1 (ver moodle).

Nota: A partir de ahora, siempre que quieras usar la tarjeta deberás antes abrirla como hemos hecho en el vídeo.

2. E/S digitales

La tarjeta tiene una serie de pines que pueden utilizarse como entradas o salidas digitales. Para poder utilizarlos, los pasos que hay que hacer son:

- Configurar los pines como digitales.
- Configurar los pines como entrada o salida.
- Establecer o leer el valor de un pin.

Vamos a ver cómo se hace esto. Trabajaremos con los pines FIO7-0, numeración que puedes ver en la propia tarjeta. Los 4 altos pueden ser analógicos o digitales. Los 4 bajos son entradas analógicas siempre y se denominan también AIN3-0. Para configurar los pines como analógicos basta con usar *configIO* con el parámetro *FIOAnalog*. Este es un número de 8 bits, cada uno asociado a un FIO. Si el bit se pone a '0' tengo un pin digital (entrada o salida), si es '1' analógico (de entrada). Por ejemplo, para configurar los 4 pines superiores como digitales haremos (Nota: como en C, en Python los números hexadecimales van con '0x'. Así, 0x0f es 00001111 en binario):

In[]: *d.configIO(FIOAnalog = 0x0f)*

Observa que el comando nos devuelve la configuración de las salidas.

Para configurar la dirección del bit, usaremos el comando *getFeedback()*. Este es un comando de propósito general que permite: i) Enviar una orden y ii) Interpretar la respuesta de la tarjeta. En nuestro caso, la orden viene dada por *u3.BitDirWrite(IONumber, direction)*. Si escribimos un 1 en *direction*, configuraremos el pin como salida, si no, como entrada. Por ejemplo, para colocar los bits 7 y 6 como salida, y los 5 y 4 como entrada haremos esto:

In[]: *d.getFeedback(u3.BitDirWrite(u3.FIO7, 1))*
In[]: *d.getFeedback(u3.BitDirWrite(u3.FIO6, 1))*
In[]: *d.getFeedback(u3.BitDirWrite(u3.FIO5, 0))*
In[]: *d.getFeedback(u3.BitDirWrite(u3.FIO4, 0))*

La dirección de un pin se puede saber obteniendo la respuesta a la orden *u3.BitDirRead(pin)*. Por ejemplo, la siguiente orden debería dar un '1' si todo ha ido correctamente:

In[]: d.getFeedback(u3.BitDirRead(u3.FIO6))

Una vez que hemos configurado un pin como salida, podemos establecer su valor con *setFIOState(fioNum, state)*. El siguiente comando establece un '1' lógico en la salida FIO7.

In[]: d.setFIOState(u3.FIO7, 1)

En el simulador puedes comprobar que se activa la salida (en el laboratorio se haría con un polímetro).
Nota: en realidad con al usar *setFIOState* se configura el pin como digital de salida de forma automática.

Para leer el valor de un pin de entrada, hay que usar *getFIOState(fioNum)*. El siguiente comando lee el valor del pin 5:

In[]: d.getFIOState(u3.FIO5)

Nota previa a los ejercicios: Dentro de un módulo que escribas tú mismo (un fichero .py con funciones), puedes importar otros módulos y usarlos. Lo normal es que se importen al **principio del fichero**, para que sean accesibles en cualquier punto de él. Debes tener en cuenta, que el espacio de variables de la consola y del módulo (fichero) son distintos. Sólo se comunican a través de los argumentos de las funciones. A la inversa, dentro del fichero tampoco podrás usar un módulo si no lo has importado en el fichero. Hay que distinguir entre las variables y funciones de un módulo, y los métodos de asociados a una variable. A los primeros no se puede acceder salvo que se haya importado el módulo. A los segundos podemos acceder si hemos pasado la variable, ya que es un objeto que lleva con ellos los métodos.

Ejemplos:

- ¿Puedo usar el valor u3.FIO7 en un fichero? No, si no has importado el módulo u3 en el mismo fichero.

- ¿Puedo usar el método *d.configIO* dentro de una función si he pasado la variable *d* (que representa la tarjeta) como argumento? Sí, ya que es un método asociado a él, lo "lleva consigo".

Ejercicio 2.1: Ondas digitales por software.

- Prepara un fichero en *python* que defina una función con tres argumentos: uno el tiempo en alto (T1); otro el tiempo en bajo (T2), ambos en segundos y un tercer argumento que represente la tarjeta abierta (*d*). La función debe generar de forma permanente una onda periódica con esos parámetros T1 y T2. Para ello, puedes ayudarte del módulo de python llamado *time*. Si lo importas (*import time*) dentro del fichero, dispondrás de la función *time.sleep(x)*, que hace 'dormir' la ejecución de ese programa durante *x* segundos (*x* es un número real, puedes poner fracciones de segundo).

- Importa ese fichero desde *ipython*, y utiliza la función con distintos valores de T1 y T2. **OJO:** usa valores de T1 y T2 de varios segundos, ya que en el simulador sólo lo podemos comprobar visualmente. En el laboratorio lo haríamos con el osciloscopio y podríamos usar T1, T2 mayores de 10 ms. También se comprobaría en el laboratorio que para valores bajos de T1 y T2, la onda no es perfecta, especialmente si haces que el PC realice otras tareas (por ejemplo, abrir otros programas, ver un vídeo en youtube, etc).

3. Entradas analógicas

En este apartado vamos a manejar las entradas analógicas simples. En la tarjeta LabJack U3-HV, el papel de los pines FIO3-0 está destinado siempre a entradas analógicas de 'alto voltaje' (en la tarjeta son AIN3-0), mientras que FIO7-4 se pueden configurar como entradas analógicas o digitales. Por seguridad, en ninguna entrada FIO7-4 puede haber más de 3.6 V. A la hora de trabajar con entradas analógicas, debemos tener en cuenta:

- La configuración de los pines, si es necesaria.
- El rango en el que queremos trabajar, dentro de los permitidos por la tarjeta.

Para configurar FIO7-4 como entradas analógicas usaremos *configIO* y el parámetro *FIOAnalog*, pero esta vez escribiremos un 1 en el bit correspondiente. Por ejemplo, si queremos usar FIO7-6 como entradas analógicas y FIO5-4 como pines digitales escribiremos:

In[]: d.configIO(FIOAnalog = 0xCF)

La lectura se realiza a través de *getAIN(posChannel, negChannel)*. La elección de los canales positivos y negativos permite realizar varios tipos de medida con rangos distintos (ver sección 2.6.1 del manual de usuario):

Medida	Rango	posChannel	negChannel
Simple (Single ended)	0 a 2.4 V en FIO7-4 -10.3 a 10.3 V en FIO3-0	7-0 (según el canal de lectura FIO7-0)	31
Medida diferencial, entre los dos canales	-2.44 a 2.44 V, sólo en FIO7-4	FIO7-4	FIO7-4
Simple (Single ended)	0 a 3.6 V para FIO7-4 -10 a 20 V para FIO3-0	7-0 (según el canal FIO7-0)	32

Por ejemplo, esto lee el valor del voltaje en el pin 6 en modo simple:

In[]: d.getAIN(posChannel=6, negChannel=31)

Ejercicio 3.1: Lectura de entradas analógicas**.**

- Establece un valor con la barra deslizante del simulador en AIN0 (=FI0). Realiza la lectura y comprueba que el valor es el esperado. Nota: en el laboratorio lo haríamos con un divisor de tensión por ejemplo.

- Prueba también a realizar una medida diferencial entre FIO7 y FIO6.

4. Salidas analógicas

Las salidas analógicas de esta tarjeta están situadas en los pines DAC1-0. Las salidas van de 0.04 a 4.95 V (aunque para hacer las cuentas tomaremos de 0 a 5 V), a través de un conversor de 10 bits. La acción sobre estos pines se realiza con el comando de propósito general *getFeedback*. En este caso, la orden que se da para establecer el valor en DAC0 es *u3.DAC0_16(value)* (idem para DAC1 cambiando DAC0 por DAC1). *Value* es un entero de 16 bits (0 a 65535), pero en realidad sólo los 10 bits altos son los efectivos para establecer un valor. En este caso establecemos un valor entero; para saber el valor de voltaje al que corresponde hay que realizar la conversión a mano. Por ejemplo, vamos a establecer el valor de 512 (la mitad del rango con 10 bits 0-1023), que multiplicamos por 2^6=64 para llevarlo a los bits altos (equivale a desplazar 6 bits).

*In[]: d.getFeedback(u3.DAC0_16(512*64))*

Comprueba en el simulador la salida en la barra deslizante que corresponde a DAC0. ¿Cuánto vale 1 LSB en este sistema? ¿Qué valor hay que poner si queremos tener un valor de salida de 1.1 V? Nota: en el laboratorio usaríamos el

OJO: el valor que colocamos en *u3.DAC0_16(value)*, *value*, tiene que ser entero. Si fuese un flotante, la parte entera se obtiene simplemente como *int(value)*. Por ejemplo, *int(3.2)* es *3*.

Ejercicio 4.1: Ondas analógicas por software

- Prepara un fichero en *python* que defina una función para generar una onda analógica triangular (aproximádamente) positiva. La función debe tener como entradas: 1) el periodo en s (*T*); 2) la amplitud en V (*A*), que puede ir hasta 5

V; 3) La variable *d* que representa la tarjeta.
Notas: si te es más fácil, puedes generar la onda con unos parámetros fijos y luego ir añadiendo la opción de modificarla. También puede ser más fácil hacer una onda sinusoidal primero. Para controlar el tiempo puedes usar *time.sleep()*, pero en este caso puede ser más fácil utilizar *time.time()*. Esta función devuelve el tiempo en segundos desde el 1 de enero de 1970. De esta manera si al inicio de un programa hacemos:

t1 = time.time()

Ahora en cualquier punto del código si hacemos:

t = time.time()-t1

tenemos acceso al tiempo desde el inicio del código y podemos escribir cualquier función del tiempo, por ejemplo una sinusoidal:

*math.sin(2*math.pi*f*t)*

OJO que a la tarjeta al final siempre hay que pasarle un número entero, recuerdad que *int(x)* obtiene la parte entera de x.

- Comprueba visualmente de forma aproximada la onda en el simulador (tendrás que usar un T de varios segundos, si no no se aprecia).