Utilización de software libre para adquisición de datos en la asignatura de Instrumentación Electrónica

En este documento se recopilan brevemente las **lecciones aprendidas** a lo largo de este proyecto de innovación docente:

1. Como conclusión general, es posible realizar las prácticas de adquisición de datos utilizando software libre. Los alumnos manejan el entorno con suficiente soltura y pueden aprender los conceptos básicos de adquisición de datos, que es el obejtivo docente de esta parte del curso.

2. Por su bajo precio, antes del inicio de este proyecto se seleccionó la tarjeta DAQ LabJack U3-HV por su precio muy asequible. Si bien existe documentación para controlar esta tarjeta desde Python en Linux, dicha documentación es claramente mejorable. En las prácticas de la asignatura se recogen los comandos más importantes, que alivian este hecho. Tampoco se puede considerar muy elegante la interfaz. Así, se mezclan algunos comandos de alto nivel con los de bajo nivel. Entre los primeros, podemos encontrar la lectura de una entrada analógica (*getAIN*) o el establecimiento de un valor en una salida digital (*setFIOState*). Pero en ocasiones, sólo podemos recurrir a un método de propósito general (*getFeedback*). Este hecho no impide trabajar con la tarjeta, pero ofrece una interfaz menos clara.

3. La operación en modo streaming ha presentado algunos problemas, si bien se han podido solventar. Para ver el tipo de problemas que aparecen, supongamos que tenemos el siguiente código (supuesto que se ejecuta en una función por ejemplo):

```
# Abrimos la tarjeta
import u3
d=u3.U3()
# Configuramos los pines y el modo streaming
# (2 canales, el 0 analógico, y los digitales) a 5 Hz
d.configIO(FIOAnalog = 0x0f)
d.streamConfig(NumChannels=2,PChannels=[0,193],  NChannels=[31,31],
ScanFrequency=5)
# Comenzamos la adquisicion
d.streamStart()
# Vamos recogiendo los datos en una lista hasta 100 valores
res=d.streamData()
ain=[ ]
for nn in range(0,100):
    data=res.next()
    if(data is not None): ain.extend(data['AIN0'])
# Cerramos el flujo y la tarjeta
d.streamStop()
d.close()
```

En principio, el código anterior debería funcionar. Sin embargo, según la frecuencia de muestreo o el número de datos, a veces se producen errores. Creemos que se trata de errores internos porque el driver del fabricante no gestiona bien la toma de datos automática y su paso al PC.
Además al producirse un error, la tarjeta se queda con el modo streaming activado, y sólo desconectándola del puerto USB se puede volver a controlar.
Al final, se desarrolló una plantilla para los alumnos (ver capturaAlumnos.py). En ella, utilizamos directamente un iterador para ir obteniendo los datos, método que sí funciona:

```
for dic in d.streamData():
  ...
```

Y ahora la variable *dic* contiene un campo ['AIN0'] con los valores del voltaje del canal 0.
Además, en la plantilla se gestionan excepciones, de forma que cualquier error en el código es gestionado cerrando el acceso a la tarjeta de forma correcta antes de volver de la función, evitando el engorroso proceso de desconectar y volver a conectar la tarjeta.

Si bien la experiencia ha sido positiva, para **futuros cursos** se puede mejorar en estos aspectos:

- Encontrar un entorno para programar en Python más amigable, en especial con la posibilidad de realizar depuraciones en modo gráfico.
- Realizar un software intermedio entre el driver y el código de usuario, haciendo un acceso más elegante y evitando los errores en modo streaming.