UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# Laboratorio virtual de tarjeta de adquisición de datos en Java

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Juan José Freire Barea

**Tutor**: Antonio Martí Campoy

Curso 2016 - 2017

# Resumen

El proyecto aquí presentado consiste en simular un dispositivo *hardware* utilizado en la asignatura de informatización industrial, impartida en el Grado en ingeniería en Tecnologías Industriales de la Escuela Técnica Superior de Ingenieros Industriales de la Universitat Politècnica de València para la adquisición de datos en una computadora, dicho *hardware* se trata de la tarjeta NuDAQ 9112 Series de la empresa ADLink.

El principal problema del uso de este *hardware* es su alto coste de adquisición, por ello se emprende este proyecto con la finalidad de ofrecer al alumnado una herramienta multiplataforma que simule el funcionamiento de esa tarjeta para que este pueda usarla en su aprendizaje.

Se puede observar que para el uso de la tarjeta original intervienen tres componentes esenciales: la propia tarjeta física, el proceso de *hardware* controlado por la tarjeta y la librería necesaria para su uso que nos suministra el fabricante.

Por ello para la simulación de estos componentes se diseñan tres aplicaciones capaces de replicar con exactitud su funcionamiento:

La tarjeta de adquisición de datos virtual, aplicación que simula el comportamiento de la tarjeta original, manteniendo toda su funcionalidad y valores internos.

El proceso de *hardware* virtual que simulará y representará el proceso de *hardware* original controlado por la tarjeta.

La librería virtual la cual ofrecerá la misma funcionalidad que la librería original pero adaptada a las nuevas necesidades.

**Palabras clave:** nudaq 9112, adlink, simulación, adquisición, datos, java, multiplataforma, hardware.

# Resum

El projecte ací presentat consistix a simular un dispositiu maquinari utilitzat en l'assignatura d'informatització industrial, impartida en el Grau en enginyeria en Tecnologies Industrials de l'Escola Tècnica Superior d'Enginyers Industrials de la Universitat Politècnica de València per a l'adquisició de dades en una computadora, el dit maquinari es tracta de la targeta NuDAQ 9112 Sèries de l'empresa ADLink,

El principal problema de l'ús d'este maquinari és el seu alt cost d'adquisició, per això s'emprén este projecte amb la finalitat d'oferir una ferramenta multiplataforma a l'alumnat que simule el funcionament d'eixa targeta perquè l'alumnat puga usar-la en el seu aprenentatge.

Es pot observar que per a l'ús de la targeta original intervenen tres components essencials: la pròpia targeta física, el procés de maquinari que la targeta controla i la llibreria necessària per al seu ús que ens subministra el fabricant.

Per això per a la simulació d'estos components es dissenyen tres aplicacions capaçes de simular amb exactitud el funcionament dels compoments anomenats:

La targeta d'adquisició de dades virtual, aplicació que simula el comportament de la targeta original, mantenint tota la seua funcionalitat i valors interns.

El procés de maquinari virtual que simularà i representarà el procés de maquinari original controlat per la targeta.

La llibreria virtual la qual oferirà la mateixa funcionalitat que la llibreria original però adaptada a les noves necessitats.

**Paraules clau:** nudaq 9112, adlink, simulació, adquisició, dades, java, multiplataforma, maquinari.

# Abstract

The project presented here consist in simulate a hardware device used in the subject industrial computerization. This subject is taught at the Universitat Politècnica de València in the degree of computer scince, this hardware is the NuDAQ 9112 Series card manufactured by ADLink that is used for data acquisition in a computer.

The main problem using this hardware is its high purchase price, that is why this project is undertaken with the purpose of offering a multiplatform tool to the students that simulate the operation of that card so the students can use it in their learning.

It is possible to observe that in the use of the original card exist three essential components: the own physical card, the hardware process and the library for its use supplied by the manufacturer.

Three applications are developed in order to simulate the components students need to use:

The virtual acquisition card, this card simulate the original card functionality maintaining their original internal values.

The virtual hardware process that simulate the original hardware process controlled by the card.

The virtual library that will offer the same functionality of the original library but adapted to the new specifications

**Keywords** nudaq 9112, adlink, simulate, acquisition, date, java, multiplataform, hardware.

# Tabla de contenidos

# Índice de ilustraciones

# 1. Introducción

En esta memoria se describe el trabajo realizado en la elaboración del Trabajo Fin de Grado presentado.

El objeto de este trabajo es solventar un problema dado en la asignatura de informatización industrial, impartida en el Grado en ingeniería en Tecnologías Industriales de la Escuela Técnica Superior de Ingenieros Industriales de la Universitat Politècnica de València y cuya docencia pertenece al director de este proyecto entre otros. El problema radica en el alto coste de adquisición junto con los altos costes de mantenimiento de uno de los recursos empleados en el laboratorio de la asignatura, se trata de una tarjeta de adquisición de datos, concretamente la NuDAQ / NuIPC 9112 Series.

Esta tarjeta conectada en un computador personal (PC) se emplea junto con un simulador *hardware* de procesos conectado a dicha tarjeta que permite que el alumno desarrolle aplicaciones para controlar el proceso simulado, leyendo los sensores y controlando los actuadores del sistema.

El alumno utiliza una librería proporcionada por el fabricante para acceder a las entradas y salidas de la tarjeta desde su algoritmo, implementado en lenguaje C.

Para solventar este problema el alumno Miguel Carro Pellicer presento un proyecto de final de carrera (“Simulador De Tarjeta De Adquisición De Datos Nudaq/Nuipc 9112 Series”) con fecha de 17/07/07 que consistía en el desarrollo de una herramienta informática que fuera equivalente a disponer del *hardware* de manera física y fuera capaz de simular todo el proceso de uso de forma fiel.

Este proyecto fue desarrollado en C y haciendo uso de PIPES por lo que su uso está limitado por estas tecnologías, por ello la finalidad de este proyecto no es otra que la de actualizar esta herramienta anteriormente descrita realizando una aplicación multiplataforma conservando todas las características de la tarjeta original.

## 1.1 Motivación

La motivación para el desarrollo de este proyecto de final de carrera principalmente es la de realizar una herramienta solida que sea capaz de ofrecer a los alumnos la posibilidad de trabajar de manera clara y les permita entender todo el sistema simulado sin coste alguno para el alumno.

Junto con la motivación anterior surge otra motivación más personal, que, no es otra que la de aprender y formarme más en campos específicos de la informática, todo informático debe sentirse motivado por aprender materia nueva que desconocía.

## 1.2 Tarjetas de adquisición de datos

La adquisición de datos consiste en la toma de muestras analógicas del mundo real para generar una serie de datos que puedan ser tratados por un ordenador, estas muestras son fenómenos eléctricos o físicos, algunos ejemplos podrían ser el voltaje, temperatura o sonido.

Para que estas muestras analógicas sean adecuadas para su transformación a señal digital es necesaria una etapa de acondicionamiento la cual es realizada por una tarjeta de adquisición de datos (DAQ).

Las tarjetas de adquisición de datos son periféricos que se conectan al PC mediante algunos de los medios disponibles del ordenador personal. Estos medios pueden ser tanto internos al ordenador como externos, dependiendo del medio empleado podríamos clasificar las tarjetas en:

### Tarjetas de adquisición de datos internas

Son aquellas tarjetas que se conectan al PC mediante puertos internos alojados dentro del propio ordenador tales como el PCI y PCI Express, puertos que se encuentran ubicados en la placa base del ordenador.

Estos puertos varían su forma y conexión dependiendo de la revisión de puerto que se trate, en el caso del PCI Express tenemos hasta 4 revisiones siendo el PCI Express 4.0 el más rápido de ellos ofreciendo velocidades de conexión de hasta 252,1 Gbit/s (31,5 GB/s).

El precio de este tipo de tarjetas varía dependiendo el fabricante y sus características, como ejemplo ponemos el modelo de tarjeta empleado en el laboratorio, la tarjeta PCI-9112 fabricada por el ADLINK Technology que tiene un precio aproximado de 500€.

### Tarjetas de adquisición de datos externas

Tarjetas que se conectan al PC mediante puertos externos, los más comúnmente utilizados son el USB (Universal Serial Bus) y el puerto *Firewire*, la ventaja de estas frente a las internas es que permiten una mayor movilidad y reutilización en diferentes ordenadores personales.

El precio de este tipo de tarjetas por el contrario suele ser superior al de las tarjetas internas a igualdad de características, como ejemplo tenemos la tarjeta LOG-0003-016G-1GB-PC del fabricante Mide que tiene un precio aproximado de 1050€.

### Tarjetas de adquisición de datos en red

Este tipo de conexión es más actual que los otros dos tipos, existen muchos fabricantes que venden este tipo de tarjetas en la actualidad, son tarjetas que se conectan a través de un puerto Ethernet o a través de WiFi.

Son el tipo de tarjetas más caro, donde tarjetas como la 976-PXI-3950 del fabricante ADLINK Technology cuesta 3250€ aproximadamente.

## 1.3 Propuesta de solución

Para la solución del problema se propone la elaboración de tres aplicaciones independientes que se comunicarán entre ellas para la transmisión de datos.

Se decide que sean tres aplicaciones independientes para que la solución ofrecida sea lo más fiel al comportamiento real del sistema que se quiere simular, así pues, las aplicaciones que se realizaran serán:

- Librería de la tarjeta virtual.
- Tarjeta de adquisición de datos virtual.
- Proceso de *hardware* virtual.

### Librería de la tarjeta virtual

La librería de la tarjeta virtual será la implementación de una librería que simulará la original ofrecida por el fabricante y que será adaptada a las nuevas necesidades del proyecto. Mientras que la librería original permite la comunicación de la aplicación del usuario con la tarjeta física, esta nueva librería permitirá la comunicación de la aplicación del usuario con la tarjeta virtual.

### Tarjeta de adquisición de datos virtual

Sera una representación virtual de la tarjeta de *hardware* físico utilizada en los laboratorios. Realizará la mayor parte de las funciones que realiza la tarjeta real, pero en un entorno software.

### Proceso de hardware virtual

Será la simulación de un conjunto de sensores y actuadores, tanto analógicos como digitales, los cuales sustituyen a los que forman el proceso *hardware* conectado a la tarjeta física. Este proceso virtual enviará información y la recibirá de la aplicación del usuario a través de la librería virtual y de la tarjeta virtual.

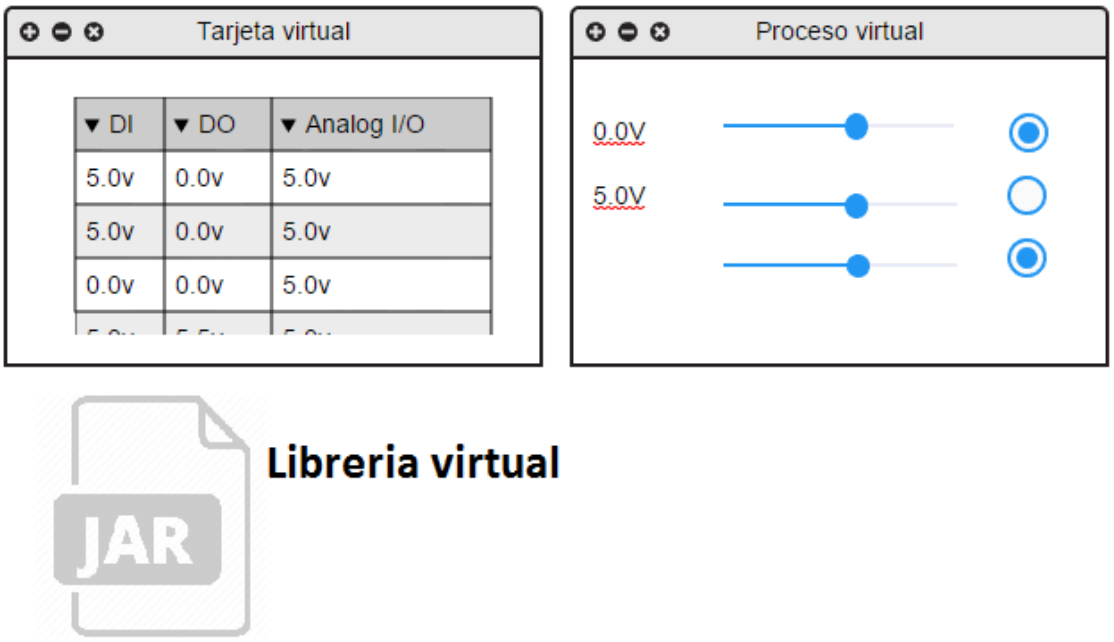


*Ilustración 1 Elementos que forman el simulador*

# 2. Requisitos

El objetivo es realizar una arquitectura fiel a la realidad con los componentes comentados anteriormente. Se realiza un estudio de requisitos por componente.

## 2.1 Librería de la tarjeta virtual

La librería es un subconjunto de funciones y variables adaptadas a las necesidades actuales del proyecto a partir de las proporcionadas por el fabricante. El alumno podrá hacer uso de esta librería virtual incluyéndola en su proyecto. Esta librería le permitirá interactuar con la tarjeta virtual, solicitando o modificando el estado de sus entradas y salidas a través de las funciones que posee, como si de la librería real se tratara.

El principal requisito de la librería es que el alumno pueda hacer uso de sus funciones como si de la librería original se tratara. Estas funciones deben ser implementadas y modificadas para que cubran nuestras necesidades.

- Conexión con la tarjeta

El usuario debe disponer de un método en la librería que le permita conectarse con la tarjeta virtual que se encuentre en ejecución.

Esta función se encarga de localizar la tarjeta virtual en ejecución y establecer el un canal de comunicación por el cual se realizarán las comunicaciones con esta.

- Recibir el estado de una entrada analógica

El usuario puede obtener los valores analógicos de entrada de la tarjeta virtual mediante esta función.

Haciendo uso de la conexión descrita en la función anterior se manda una petición que atiende la tarjeta virtual.

Posteriormente la tarjeta virtual devolverá con otro mensaje a la librería virtual el valor solicitado.

- Enviar el estado de una salida analógica

El usuario puede escribir valores analógicos en las salidas de la tarjeta virtual mediante esta función.

Dicho cambio será realizado mediante el envió de un mensaje a la tarjeta virtual haciendo uso de la conexión establecida anteriormente con esta.

- Recibir el estado de una entrada digital

El usuario puede obtener los valores de entrada digitales de la tarjeta virtual mediante esta función.

Para ello se hace uso de la conexión establecida con la tarjeta por la cual será enviado el mensaje con la petición del valor.

Posteriormente obtenemos una réplica a nuestra petición por parte de la tarjeta con el valor solicitado.

- Enviar el estado de una salida digital

El usuario tiene la capacidad de escribir el valor deseado en las salidas digitales de la tarjeta virtual.

Se envía un mensaje a la tarjeta virtual haciendo uso de la conexión establecida y se manda el mensaje codificado para remplazar el valor anterior.

- Desconectarse de la tarjeta

Función que termina las comunicaciones con la tarjeta virtual.

## 2.2 Tarjeta de adquisición virtual

La tarjeta virtual tiene la función de simular con exactitud el comportamiento que tiene la tarjeta real física, para ello dispone de tres listas que contendrán cada uno de los posibles estados de la tarjeta. Se encuentran clasificados en:

- Valores analógicos
- Entradas digitales
- Salidas digitales

Estos estados estarán esperando cambios continuamente, además la tarjeta virtual tiene la capacidad de atender peticiones al mismo tiempo tanto del proceso de *hardware* virtual como de la librería, procesando las peticiones cuando sea necesario y dando respuestas en las lecturas de valores.

Las funciones principales que cumple la tarjeta de adquisición son las siguientes:

- Establecer comunicación

Se tienen que establecer los medios de comunicación empleados para la comunicación entre aplicaciones.

- Recibir mensajes de la librería

La librería tiene la capacidad de recibir mensajes de la librería, solicitando o dando el estado de una entrada o salida.

- Enviar mensajes a la librería.

Se envían mensajes a la librería virtual en el momento que se solicita la lectura de un valor de la tarjeta de adquisición virtual.

- Decodificar los mensajes

La librería tiene que decodificar todos los mensajes entrantes entendiendo las necesidades de cada mensaje recibido.

- Codificar los mensajes

Al igual que tiene la capacidad de decodificar los mensajes es necesaria la capacidad de codificarlos para su envió.

- Cambio de valores

La tarjeta debe cambiar sus valores internos en el momento que sea necesario por una petición entrante o saliente a esta.

- Lectura de valores

La tarjeta lee u obtiene sus valores internos cuando sean solicitados a través de la librería o proceso de *hardware* virtual.

## 2.3 Proceso de hardware virtual

El proceso de *hardware* virtual es el encargado de simular un conjunto de dispositivos de entrada y salida, tanto analógicas y digitales, simulando el proceso de *hardware* sobre el que trabaja la tarjeta de adquisición de datos. El usuario puede actuar sobre los dispositivos de entrada modificando sus características, y, por tanto, al variar las entradas del proceso virtual se producirá un cambio que tiene que verse reflejado en la tarjeta de adquisición virtual. Esto es posible ya que se trata de un proceso virtual gráfico que posee una GUI, esta permite interactuar al usuario con una serie de controles gráficos que representan a los dispositivos de entrada mencionados anteriormente sobre los que se modificara sus características.

De igual manera si un cambio es realizado desde la aplicación del usuario a través de la librería, afectando a alguna de las entradas o salidas del proceso, el usuario deberá ver el cambio en el dispositivo de salida.

Es necesario que el proceso de *hardware* virtual tengas las siguientes funcionalidades:

- Establecer comunicación

Se tienen que establecer los medios de comunicación empleados para la comunicación entre aplicaciones.

- Recibir mensajes de la tarjeta

El proceso de *hardware* tiene la capacidad de recibir mensajes de la tarjeta, solicitando el cambio de alguno de sus componentes.

- Enviar mensajes a la tarjeta

Se envían mensajes a la tarjeta virtual para solicitar el cambio de alguno de sus valores.

- Decodificar los mensajes

El proceso de *hardware* tiene que decodificar todos los mensajes entrantes entendiendo las necesidades de cada mensaje recibido.

- Codificar los mensajes

Al igual que tiene la capacidad de decodificar los mensajes es necesaria la capacidad de codificarlos para su envió.

- Cambio de valores gráficamente

El proceso debe modificar sus valores gráficamente en el momento que sea necesario por una petición entrante o saliente a esta.

- Interfaz gráfica

Ofrece una interfaz gráfica capaz de mostrar todos los elementos y sus cambios de manera clara.

# 3. Arquitectura del simulador

En este apartado se describirá la arquitectura que presenta el proyecto realizado, describiendo su estructura, conexión y protocolos.

Como se ha comentado anteriormente la arquitectura del proyecto está compuesta de tres aplicaciones de la siguiente manera:

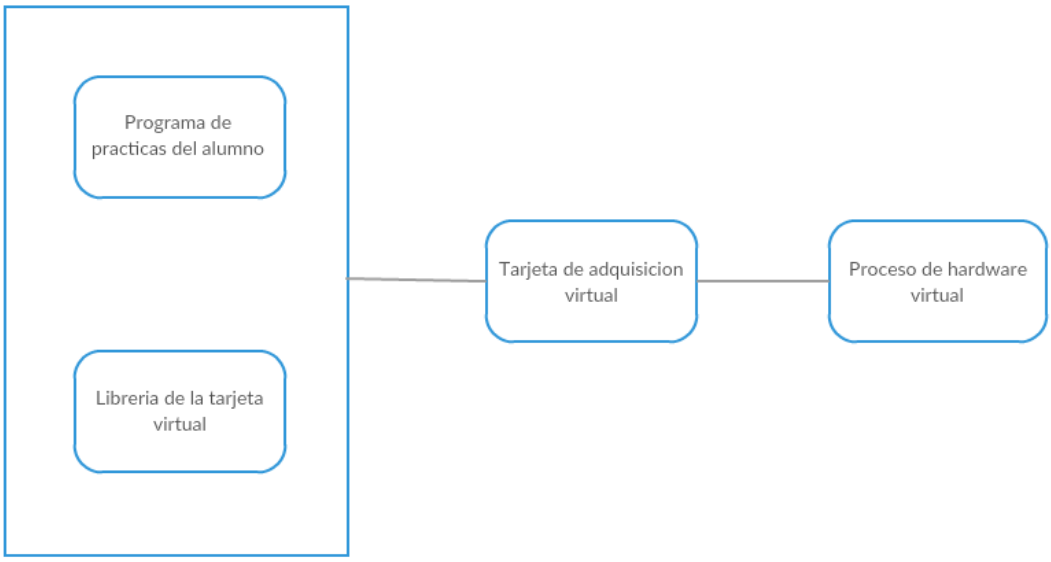


*Ilustración 2 Arquitectura del simulador*

Se elige esta arquitectura por ser lo más fiel a la realidad. En la realidad nos encontramos con una tarjeta de *hardware* que es un dispositivo físico que se encuentra conectado al ordenador. La representación de este dispositivo será nuestra aplicación tarjeta de adquisición virtual.

La librería del fabricante escribe en, o lee de, los puertos físicos de entrada/salida en los que está conectada la tarjeta a través de sus registros. Para las salidas, las funciones de la librería escriben en determinados puertos, es decir, en determinados registros de la tarjeta valores decimales que se convertirán en voltaje. Para las entradas, las funciones de las librerías leen de determinados puertos, es decir, registros de la tarjeta donde se encuentras los valores digitales que representan los voltajes de las entradas. También existen funciones para escribir o leer determinados parámetros de operación o configuración de la tarjeta. En los computadores actuales, esto se gestiona a través de un driver propio de la tarjeta y los servicios del sistema operativo.

En este TFG, la librería no accederá a los puertos físicos de la tarjeta, ya que esta no existe, sino que enviará o recibirá los valores digitales que representan los voltajes en las entradas y salidas mediante mensajes a y desde la tarjeta virtual. De esta forma el usuario, es decir el programador, verá las mismas funciones que en la librería física, haciendo transparente el uso de una tarjeta real o la virtual.

En la tarjeta física, sus entradas y salidas estarán conectadas a los sensores y actuadores del proceso que se quiere controlar, permitiendo leer su estado y actuar sobre él. En este TFG se desarrollará una aplicación gráfica que simulará dispositivos de entrada y salida que se conectarán a la tarjeta virtual, permitiendo al alumno controlar un proceso virtual.

## 3.1 Aplicaciones

### Librería de la tarjeta virtual

Esta librería nos dará toda la funcionalidad necesaria para interactuar con la tarjeta de adquisición de datos virtual, dispondrá de la misma funcionalidad que la librería original pero adaptada a las nuevas necesidades, para ello es necesario que exista conexión entre la librería y la tarjeta.

### Tarjeta de adquisición de datos

La tarjeta de adquisición virtual queda situada en el centro del esquema, esta debe mostrar en todo momento los valores que contiene, como si de la tarjeta física se tratara.

Para ello debe estar atendiendo peticiones de cambios tanto de la librería como del proceso de *hardware* virtual, de igual manera si recibe un cambio por parte de la librería virtual esta tendrá que informar del cambio realizado al proceso si el valor que se desea modificar es accesible desde el proceso de *hardware* virtual.

### Proceso de hardware virtual

El proceso de *hardware* virtual que es el encargado de simular la interacción entre la tarjeta y el mundo real. Cuando una magnitud física cambia, debe poder comunicarse con la tarjeta virtual, de igual manera tendrá la capacidad de representar los valores de salida de la tarjeta virtual.

## 3.2 Conexión entre aplicaciones

La conexión entre aplicaciones se realizará con *sockets*. Se decide hacer uso de esta tecnología puesto que es sencilla, nos la ofrece el propio sistema operativo, es conocida por los alumnos de la universidad ya que es impartida en la asignatura de redes que se imparte en el grado de ingeniería informática y que está incluida dentro de la formación básica, además permite modificaciones o ampliaciones futuras de manera sencilla.

Al hacer uso de esta tecnología cualquier alumno que lo desee podrá realizar modificaciones sobre el proyecto de manera sencilla, ya que el fin del proyecto es ofrecer una herramienta a la universidad y a sus alumnos.

La estructura de comunicaciones que decide montarse por simplicidad y comodidad es la siguiente:

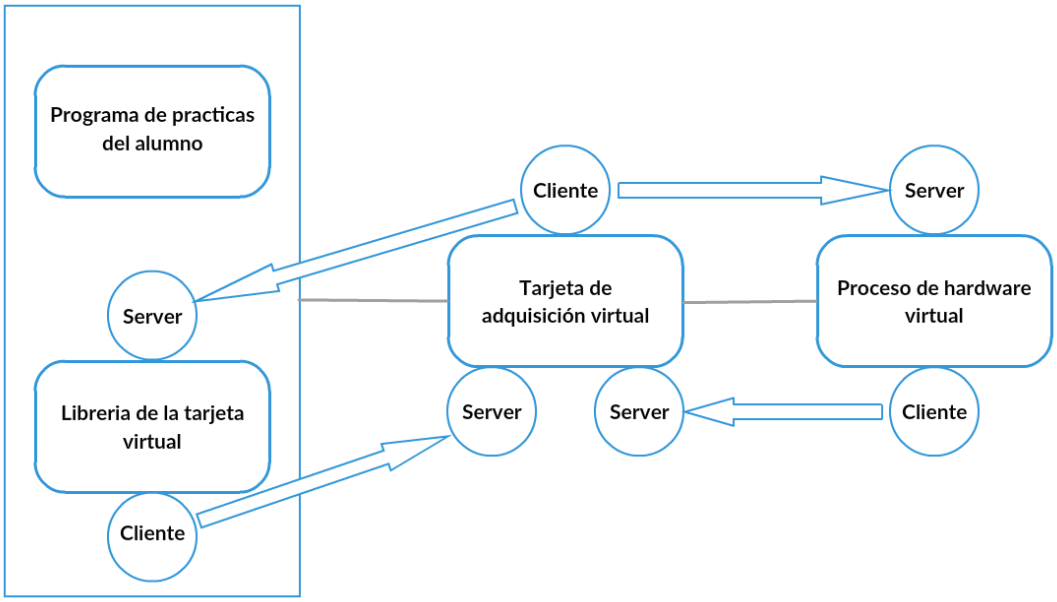


*Ilustración 3 Conexión entre aplicaciones*

## Conexiones entre aplicación

Librería de la tarjeta virtual(Cliente) > Tarjeta de adquisición virtual(Servidor)

- Conexión con la tarjeta.
- Solicitud de modificación de valor.
- Solicitud de lectura de valor.

Tarjeta de adquisición virtual(Cliente) > Librería de la tarjeta virtual(Servidor)

- Respuesta a la acción de conexión.
- Envío de valor solicitado.

Tarjeta de adquisición virtual(Cliente) > Proceso de *hardware* virtual(Servidor)

- Replica de modificación de valor por parte de la librería.
- Solicitud de cierre de la aplicación.

Proceso de *hardware* virtual(Cliente) > Tarjeta de adquisición virtual(Servidor)

- Conexión con la tarjeta.
- Solicitud de modificación de valor.

Se procede a analizar la estructura de conexiones componente a componente, de manera que quede detallado el proceso de conexión que sigue cada aplicación en el momento de interacción.

### Librería de la tarjeta virtual

En primer lugar, la librería tiene un proceso de conexión con la tarjeta de adquisición virtual el cual consiste en la apertura de un *socket* servidor que esperará por la confirmación de la tarjeta y un *socket* cliente que por cada puerto que se encuentre en uso hará envío de una consulta, esta consulta consistirá en el envío del identificador de la librería "LIB" y la recepción del mismo por parte de la tarjeta, consulta que se realizará siempre dentro del rango de puertos prefijados. Cuando esta consulta sea procesada por la tarjeta, devolverá una respuesta de confirmación, sabiendo así donde se encuentra alojado el servidor de la tarjeta.

Cuando se quiera leer el estado de una entrada de la tarjeta primeramente se abrirá un *socket* servidor para esperar a la respuesta de la tarjeta virtual, seguidamente con un *socket* de tipo cliente se mandará la petición junto con el puerto donde debe devolver el resultado, siendo así atendido por el servidor que abrimos anteriormente. Cuando se finalice esta operación se devolverá el valor.

Para modificar el estado de una salida de la tarjeta se procede a abrir un *socket* cliente que enviará el mensaje para que se realice esta modificación sin esperar respuesta de la tarjeta.

### Tarjeta de adquisición de datos virtual

La tarjeta de adquisición tendrá en todo momento dos servidores que serán abiertos sobre dos puertos específicos dentro de un rango prefijado, evitando así que pueda ocurrir un fallo en el intento de apertura sobre un puerto que se encuentre ya en uso en la maquina donde se ejecute la aplicación.

Se decide hacer uso de dos servidores para casos en los que se tenga que atender peticiones simultáneas de ambas aplicaciones (la librería y el proceso) y no puedan ocurrir bloqueos en el puerto, por lo que cada servidor estará ejecutándose en puertos diferentes.

Estos servidores se encontrarán escuchando mensajes para leer entradas o modificar salidas continuamente y cada uno de los servidores atenderá las posibles peticiones de la tarjeta o del proceso, quedando así destinado cada servidor a atender las peticiones específicas de una de las aplicaciones en concreto. En el momento de recepción de alguna petición cada uno de estos servidores hará uso de una función específica para la decodificación del mensaje, realizando así las acciones necesarias, en caso de que una petición entrante de la librería modifique alguno de los valores mostrados por el proceso de *hardware* la tarjeta informaría al proceso de *hardware* virtual de estos cambios.

Junto con los *sockets* servidores que atenderán todas las peticiones entrantes se hará uso de otro tipo de *sockets*, *sockets* cliente que serán usados cada vez que se desee realizar el envió de algún mensaje a alguna de las otras aplicaciones, este cliente será abierto en el momento del envió del mensaje y finalizara su vida en el momento que el mensaje sea recibido por el destinatario.

### Proceso de hardware virtual

De igual manera que la librería virtual, el proceso de *hardware* virtual también tendrá que encontrar en qué puerto se encuentra escuchando el servidor de la tarjeta virtual que ha de atender sus peticiones, por lo que el proceso iniciará un proceso de búsqueda como el anteriormente descrito en la librería con la peculiaridad que este mandará junto con su identificador el número de puerto donde deberá enviar sus peticiones la tarjeta sobre el proceso, que serán atendidas por un servidor abierto de forma permanente. Este servidor será abierto realizando una búsqueda de un puerto libre donde ubicarlo, para lo cual se buscará un puerto libre entre un rango prefijado de puertos para agilizar la búsqueda de un puerto libre de uso.

Cada vez que este servidor reciba una petición de la tarjeta de adquisición virtual será decodificado, realizando así la acción pertinente. De igual manera cada vez que se quiera enviar un mensaje a la tarjeta virtual se abrirá un *socket* cliente que se encargara de mandar un mensaje a la tarjeta y será cerrado posteriormente.

## 3.3 Protocolos

### Protocolo de conexión

Sobre los *sockets* pueden emplearse dos protocolos de conexión diferentes, estos son el TCP (*Transmission Control Protocol*) y el UDP (*User Datagram Protocol*).

En este proyecto se decide hacer uso del protocolo de conexión TCP, esta elección es debida a varias razones que se comentan a continuación.

El protocolo TCP junto con el protocolo UDP son estudiados en la asignatura de redes impartida en el grado en ingeniería informática en la formación obligatoria, por lo que los alumnos deberían conocer ambos protocolos y más concretamente su uso sobre el lenguaje java utilizado para este proyecto. Son protocolos relativamente sencillos de entender, de manera que el usuario tendrá facilidad para entender el funcionamiento de las conexiones realizadas entre aplicaciones.

Existen diferencias entre ambos protocolos las cuales hacen que cada uno sea mejor o peor dependiendo de las necesidades del proyecto, el protocolo UDP es sin conexión y no garantiza la entrega de datos mientras que el protocolo TCP es con conexión y garantiza la entrega de datos, esta es la diferencia que hace el TCP más útil en nuestro proyecto, esta es la garantía con la que se transmiten los datos, permitiéndonos así la correcta recepción de las peticiones entre las aplicaciones y pudiendo disponer de la información de que el envió se ha realizado con éxito.

## Protocolo de aplicación

Para los mensajes se decide utilizar texto plano para su codificación, por su simplicidad y la facilidad con la que se puede entender si se hace uso de herramientas que permitan mostrar la traza de conexiones y el envío de mensajes que existe entre las distintas aplicaciones.

El formato utilizado es el mismo para las tres aplicaciones y para todos los sentidos de comunicación. El formato es el siguiente:

ID. Emisor#Acción#Tipo dato#Valor a modificar#Valor

Ahora procederemos a analizar los campos de la cadena del mensaje uno a uno y a ver sus posibles valores y rangos.

Campo ID. Emisor:

Se trata del campo que dará identidad al remitente del mensaje y a la acción a realizar, esto resulta imprescindible para saber en todo momento desde cualquiera de las aplicaciones de quien procede un mensaje y como se debe de actuar.

Sus posibles valores son:

- LIB, valor que hace referencia a un mensaje que proviene de la librería.
- PHV, valor que hace referencia a un mensaje que proviene del proceso de *hardware* virtual.
- TAR, valor que hace referencia a un mensaje que proviene de la tarjeta de adquisición virtual.

Campo Acción:

Este campo nos indicará la acción que pretende llevarse a cabo con el mensaje enviado.

Sus posibles valores son:

- C, letra utilizada para realizar la acción de la conexión.
- M, letra utilizada para la modificación de alguno de los valores.
- R, letra utilizada cuando se espera una respuesta, como en la lectura de un valor.

Campo Tipo dato:

Este campo nos indicará a la hora de realizar una modificación o lectura donde se quieren modificar los valores o de donde se quieren obtener.

Sus posibles valores son:

- AI, iniciales que hacen referencia a *analogic input.*
- AO, iniciales que hacen referencia a *analogic output*.
- DI, iniciales que hacen referencia a *digital input*.
- DO, iniciales que hacen referencia a *digital output*.

Campo Valor a modificar:

Con este campo nos queremos referir al pin correspondiente al conector indicado con el tipo de dato donde se encuentra el valor a modificar o leer.

Se trata de un campo numérico con un único significado donde su rango puede variar entre 0 y 36 que es el número de pin máximo del conector para salidas y entradas analógicas.

Campo Valor:

Este campo será usado para indicar el valor con el cual se quiera realizar la acción indicada.

Es un campo de tipo numérico que variará su valor y su rango dependiendo de si se trata de un tipo de dato analógico o digital sobre el que se quiera realizar la acción.

Posibles valores para datos digitales:

- 0.
- 1.

Posibles valores para datos analógicos:

- Valores > -5.
- Valores < 5.

Por ejemplo, un mensaje enviado de la librería de la tarjeta virtual a la tarjeta de adquisición de datos virtual podría ser este:

LIB#M#DO#1#1

Como primer valor tenemos el identificador del emisor, en este caso se trata de la librería de la tarjeta virtual del cual cogemos sus iniciales para hacer el identificador “LIB”.

El siguiente valor en la cadena del mensaje será la acción a realizar en la tarjeta virtual que como se ha explicado anteriormente será la acción de modificar un valor.

Como siguiente valor tendremos el tipo de campo sobre el cual se va a querer modificar su valor, en este caso las siglas corresponden a *digital output* (salida digital).

Seguidamente el número de pin sobre el conector que se quiere modificar, en este caso estaríamos modificando el pin número 1 que se corresponde con la salida digital número 2.

Por último, tendríamos como último valor en la cadena del mensaje el valor por el cual se va a proceder a cambiar el valor actual, en este caso tendríamos un 1.

A continuación, veremos una ilustración de como seria el envío de este mismo mensaje a lo largo de las tres capas representadas, aplicación, *sockets* y transporte.

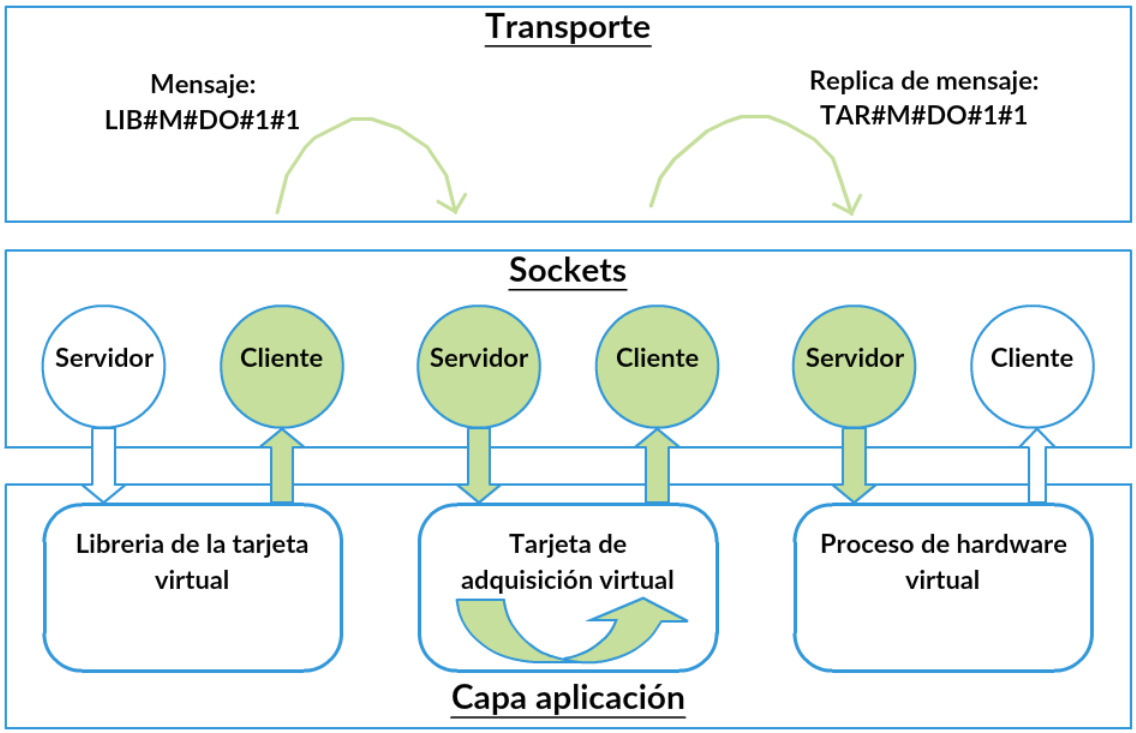


*Ilustración 4 Ejemplo de uso del protocolo de la aplicación*

El camino que seguirá será el siguiente:

1. El primer paso será el envío del mensaje por parte de la librería a través de su *socket* cliente.
2. Seguidamente el servidor que se encuentra escuchando de la librería en la tarjeta virtual recibiría la petición.
3. La tarjeta virtual, puesto que se trata de una modificación sobre un valor que se encuentra representado en el proceso de *hardware*, deberá replicar este mensaje al proceso de *hardware* para que este lo modifique de igual manera. La única variación que existirá con el mensaje original será el cambio de identificador, esto es debido a que el mensaje cambiara su procedencia siendo ahora el emisor la tarjeta virtual.
4. Por último, el proceso de *hardware* virtual recibirá el mensaje replicado y actualizará la información de la salida en la pantalla.

# 4. Detalles de la implementación

En este apartado se describirá todo lo referente a los detales de la implementación, tales como: tecnologías empleadas, herramientas empleadas, detalles de implementación de la librería, tarjeta y proceso de *hardware*.

## 4.1 Tecnologías

### Java

Es el lenguaje empleado en el desarrollo del proyecto. Se trata de un lenguaje de programación orientado a objetos y concurrente, fue diseñado específicamente para no tener dependencia alguna de ninguna plataforma, de manera que permite realizar el desarrollo de un programa y su ejecución de igual manera en cualquier plataforma, por lo que se le llama lenguaje multiplataforma.

Java fue desarrollado por James Gosling de la empresa Sun Microsystems y publicado en 1995, empresa que después fue adquirida por la compañía Oracle.

La sintaxis de java principalmente deriva del lenguaje C y C++ con la característica que posee menos funcionalidad a bajo nivel de la que brindan estos lenguajes. Las aplicaciones programadas en java son compiladas a *bytecode*, estas compilaciones necesitan de una máquina virtual de java para su ejecución, de ahí que se trate de un lenguaje multiplataforma, ya que en cada plataforma hardware y sistema operativo hay una máquina virtual java, por lo que funciona en todas las plataformas de igual manera.

La versión utilizada para el desarrollo del proyecto es la más reciente conocida, la versión 1.8 del JDK de java.

### JavaFX

JavaFx es un conjunto de paquetes de gráficos y multimedia que facilita a los desarrolladores la elaboración de aplicaciones gráficas, está escrita como un API (Application Programming Interface) de java y puede encontrarse integrada en el propio lenguaje desde su versión 1.7.

Sus principales características son las siguientes:

- Permite la utilización de FXML que se trata de un lenguaje de marcado declarativo basado en XML para el desarrollo de las interfaces de la aplicación.
- Interoperabilidad, permitiendo portar de manera sencillas las aplicaciones desarrolladas con Swing.

- Proporciona los principales controles de interfaz necesarios para el desarrollo de aplicaciones con interfaz gráfico de usuario (GUI), así como el uso de tecnologías web para la personalización de controles como CSS (*Cascade Style Sheed*).
- Proporciona control sobre el API *canvas* de java que permite dibujar de manera directa sobre la ventana.
- Soporte multi táctil para las aplicaciones desarrolladas.
- Aceleración por hardware para gráficos.

### Sockets

Un *socket* es el punto final o inicial de una interconexión que realizamos entre varios programas a través de la red.

Existen dos tipos de *socket*:

- Orientado a la conexión, donde se establece un canal entre servidor y cliente fiable y sin perdidas de información. TCP
- No orientado a la conexión, donde se establece un canal entre servidor y cliente no fiable donde pueden ocurrir perdidas de información. UDP

Existen dos tipos de finalidad para un *socket*:

- Servidor, se encarga de permanecer a la escucha en un puerto de la red a la espera de mensajes.
- Cliente, debe saber la dirección IP y el puerto a la cual mandará el mensaje.

### Microsoft Windows

Sistema operativo utilizado para el desarrollo del proyecto, concretamente su versión Windows 10.

Se trata de un sistema operativo de pago, distribuido por la empresa Microsoft y que tuvo su primera versión de sistema operativo el 20 de noviembre de 1985 y que ha sufrido diversos cambios hasta ser el sistema operativo que hoy en día conocemos.

## 4.2 Herramientas

### Eclipse

Se trata de un entorno de desarrollo integrado o entorno de desarrollo interactivo, aplicación informática que proporciona funcionalidades para facilitar el desarrollo de software. Fue creado por IBM (International Business Machines) inicialmente y desarrollado actualmente por Eclipse Fundation.

Usualmente un IDE consiste en un editor de código fuente junto con herramientas de ayuda a la construcción de código automáticamente, y a su vez suelen disponer también de depuradores y compiladores.

Eclipse posee un gran abanico de *plugins* y funcionalidades añadidas que permite utilizarlo para casi cualquier lenguaje existente, El lenguaje Java es soportado de forma nativa por lo que no es necesario instalar ningún *plugin.*

La versión utilizada para el desarrollo del proyecto es eclipse neon que fue publicada el 22 de junio de 2016, más concretamente su revisión 3.

### Gluon Scene builder

Se trata de una herramienta que facilita el desarrollo de las vistas o *layouts* de la aplicación de manera visual ofreciendo la posibilidad de modificar todo lo referente a estas de manera sencilla.

Ofrece también la posibilidad de realizar una visualización previa del diseño que se está realizando sin tener nada de funcionalidad desarrollada en la aplicación.

Esta herramienta es ofrecida por la empresa Gluon.

## 4.3 Librería de la tarjeta virtual

### Estructura

La estructura que sigue la librería de la tarjeta virtual ha sido decidida por propia voluntad, es una aplicación que carece de interfaz, debido a esto no se ha optado por usar la estructura habitualmente usada en la programación orientada a objetos, la estructura de modelo vista y controlador.

Aquí se ha optado por separar el paquete de clases relacionado con la comunicación entre aplicaciones del paquete de clases principal de la aplicación.

## Paquetes

Como se ha comentado anteriormente se ha optado por separar la aplicación en dos paquetes:

- ***Paquete communication***

Este paquete de clases está dedicado a contener todos los recursos necesarios para establecer comunicación con la tarjeta de adquisición virtual.

Contiene dos clases de java, una primera clase llamada Cliente que es la encargada de generar un *socket* cliente y transmitir un mensaje a la tarjeta virtual cuando sea necesario.

También contiene una segunda clase java llamada ServerQA, se ha decidido llamar a esta clase de esta manera por su funcionalidad interna, esta clase java instancia un *socket* servidor el cual es el encargado de recibir la respuesta por parte de la tarjeta virtual al mensaje que le lance la librería, entonces las iniciales QA se corresponden con las palabras inglesas *question* & *answer* que traducidas a nuestro idioma son pregunta y respuesta.

- ***Paquete library***

Este paquete contiene la clase principal para el uso de la librería, la clase *library*, clase que contiene todos los métodos necesarios para realizar la comunicación con la tarjeta de adquisición virtual, haciendo uso de las clases del paquete de comunicación.

## Ficheros

Se procede a detallar la funcionalidad de cada uno de los ficheros ubicados en los distintos paquetes.

- ***Paquete communication***

Clase Client

Esta clase está compuesta por una variable estática que no cambiará su valor, será el host sobre el que se va a lanzar el *socket* cliente y de un constructor de objeto público que recibirá como parámetros el puerto sobre al que se procede a hacer él envió y el mensaje a enviar.

Primeramente, se instancia el *socket* cliente con el host y el puerto como valores de entrada, posteriormente se abre un flujo de datos de salida sobre el *socket* cliente por el cual se enviará el mensaje y por último se cierran tanto el flujo de datos como el *socket* al finalizar su transmisión.

Clase ServerQA

Esta clase extiende mediante herencia de java de la clase Thread, esto significa que en su interior tendrá un método *run* que ejecutará todas las instrucciones de código en segundo plano sobre un hilo de ejecución sin bloquear la aplicación principal. Se ha decidido programarla así para evitar que penalice la experiencia de usuario con bloqueos molestos.

La principal funcionalidad de esta clase es la de enviar mensajes a la tarjeta virtual por los cuales la librería deberá esperar una respuesta por parte de la tarjeta virtual.

Para ello cuenta en su interior con los siguientes métodos:

- Método run

Método utilizado para la ejecución asíncrona de la conexión con la tarjeta virtual y el envió del mensaje junto con la espera de la respuesta por parte de la tarjeta virtual.

- Métodos modificadores get y set

Métodos para la modificación del valor de las variables destinadas a la conexión con la tarjeta virtual y el contenido del mensaje, variables como por ejemplo el puerto de conexión donde se encuentra escuchando la tarjeta virtual y que ha sido almacenado con anterioridad al envío del mensaje en la conexión.

- Método connectServer

Método que realiza la búsqueda de un puerto libre y conexión de un *socket* servidor para la recepción de la respuesta al envío del mensaje por parte de la tarjeta virtual.

- Método searchCard

Destinado a la búsqueda y localización del puerto donde se encuentra escuchando la tarjeta virtual, para hacer uso de este en los envíos y generar así una conexión con la tarjeta virtual.

- Método sendQA

Este es el método llamando desde el método asíncrono run que gestiona las operaciones necesarias para él envió del mensaje y la recepción de la respuesta generada por la tarjeta virtual.

- Método showAlert

Método que se encargara de la generación de ventanas modales de alerta cuando se realice la conexión o desconexión de manera exitosa.

- ***Paquete Library***

Clase Library

Los métodos que componen esta clase son los siguientes:

- Register_Card

Esta función se encarga de localizar la tarjeta virtual en ejecución y establecer el *socket* por el cual se realizarán las comunicaciones con esta.

Para ello inicia la localización ser *socket* servidor alojado en la tarjeta virtual a la espera de recibir confirmación por parte de la tarjeta.

- AI_ReadChannel

El usuario puede obtener los valores analógicos de entrada de la tarjeta virtual mediante esta función.

La función recibe como parámetro de entrada el pin del cual se quiere obtener su valor, se enviará a la tarjeta un mensaje especificando el tipo de valor y el pin del cual se quiere obtener.

Posteriormente la tarjeta virtual devolverá con otro mensaje a la librería virtual el valor solicitado.

- AO_WriteChannel

El usuario puede escribir valores analógicos en las salidas de la tarjeta virtual mediante esta función.

Para ello la función recibe como parámetros de entrada el pin sobre el cual se quiere escribir el valor y el valor a escribir, después se procede a enviar el mensaje a la tarjeta virtual con los valores de entrada junto con la acción que se pretende realizar, una vez recibido el mensaje la tarjeta virtual establecerá el valor de la salida.

- DI_ReadPort

Se pueden obtener los valores de entrada digitales de la tarjeta virtual mediante esta función.

La función recibe como argumento el pin del que se desea obtener el valor, con este valor envía un mensaje a la tarjeta virtual con el tipo de acción a realizar.

Posteriormente la tarjeta virtual devuelve el valor del pin solicitado.

- DO_WritePort

El usuario tiene la capacidad de escribir el valor deseado en las salidas digitales de la tarjeta virtual.

Para ello la función recibe como parámetros de entrada el pin sobre el que se desea realizar la escritura y el valor a escribir.

- Disconnect_Card

Función que termina las comunicaciones con la tarjeta virtual.

## 4.4 Tarjeta de adquisición de datos virtual

### Estructura

En la tarjeta de adquisición de datos virtual sí que se ha adoptado la estructura modelo, vista y controlador, puesto que se trata de una aplicación la cual hemos dotado de interfaz gráfica para representar sus valores.

Además de la estructura citada en el párrafo anterior se cuenta también con un paquete dedicado a la comunicación con las otras dos aplicaciones desarrolladas en este trabajo (la librería virtual y el proceso virtual) que se incluye en la capa de servicios. También cuenta con otro paquete dedicado a clases que dotan de utilidad a la aplicación, son utilidades las cuales se usan de manera concurrente por varias clases de la aplicación, por este motivo se decide aislar toda esta funcionalidad dentro de una o varias clases en un paquete llamado útil.

### Paquetes

La aplicación se compone de los siguientes paquetes:

- ***Paquete application***

Este paquete de clases contiene la clase Main, clase principal de la aplicación que tiene el código necesario para lanzar su ejecución de forma visual cargando la vista principal.

Además, esta clase posee el método que decodifica los mensajes entrantes, tanto de la librería virtual como del proceso de *hardware* virtual.

- ***Paquete communication***

Este paquete al igual que el mencionado anteriormente en la librería virtual contiene todas las clases necesarias para realizar las conexiones entre aplicaciones.

Dispone de una clase Cliente la cual se encarga de enviar los mensajes a las otras dos aplicaciones, una clase EchoThread la cual atenderá una petición concreta por parte de alguna de las dos aplicaciones y una clase Server que escuchará en todo momento las peticiones de las otras dos aplicaciones de manera no bloqueante.

- ***Paquete model***

Este paquete contiene los modelos que se ha considerado como necesarios para el mejor uso y funcionamiento de la aplicación y su principal finalidad es administrar la lógica de la aplicación.

Un modelo en la programación orientada a objetos se encarga de abstraer toda la información relacionada con un objeto de una misma clase, en nuestro paquete contamos con dos modelos:

Modelo Card

Es la clase que representa la tarjeta de adquisición de datos virtual y contiene los mismos valores y registros que la tarjeta física.

Modelo InputOutput

Se ha decidido crear este modelo para la representación de una entrada o salida de la tarjeta de adquisición virtual, donde cada objeto de este tipo contendrá un atributo pin indicando el pin de la tarjeta donde se aloja y un valor.

- ***Paquete útil***

Paquete destinado a contener todas las clases que ofrecen funcionalidades a la aplicación que sean de uso común en múltiples clases.

Contiene una clase Utils la cual inicializa las listas de valores de la tarjeta virtual con valores predeterminados.

- ***Paquete view***

En este paquete se encuentra todo el contenido relacionado con la GUI de la aplicación, desde las vistas hasta los controladores de estas, incluyendo también los recursos necesarios como podrían ser iconos e imágenes.

En su raíz contiene el controlador de la vista, además contiene en su interior otros dos paquetes que se mencionan a continuación.

- ***Paquete view.fxml***

Este paquete se encuentra dentro del paquete *view*, se crea este paquete para contener los *layouts* de las vistas, archivos que darán aspecto visual a la ventana de la aplicación.

- ***Paquete view.resources***

Este paquete contiene todos los recursos necesarios para emplear en la vista como por ejemplo el icono empleado en la ventana de la aplicación.

## Ficheros

Se procede a detallar la funcionalidad de cada uno de los ficheros ubicados en los distintos paquetes.

- ***Paquete application***

Clase Main

Esta clase es la clase principal de la aplicación y la que se encarga de la ejecución de la aplicación.

Los métodos que componen esta clase son los siguientes:

- start

Método propio del ciclo de vida de una clase que extiende del tipo de objeto *aplication* en JavaFx, se ejecuta en el momento de arranque de la aplicación y aquí se instancian e inicializan todos los *sockets* servers, además se realizar la llamada al método que inicializa la vista de la aplicación.

- stop

Dentro de este método se realiza la parada de los servidores que utiliza la tarjeta de adquisición de datos para la escucha de peticiones de las otras dos aplicaciones.

Es un método junto con *start* del ciclo de vida de una aplicación en JavaFx y este es llamado en el momento de cierre de la aplicación.

- initRootLayout

Este método se encarga de realizar la carga del *layout* o vista de la aplicación sobre la ventana de la aplicación.

Para ello primeramente se encarga de realizar una instancia de la vista FXMLLoader para posteriormente pasarle la vista cargada al *stage* que es un objeto que contiene la instancia de la ventana de la aplicación.

- main

Método principal de la aplicación, toda aplicación java debe tener un método main que será el método encargado de lanzar la aplicación junto con los argumentos de entrada de esta.

- processMessage

Es el método llamado por los *sockets server* en el momento de recepción de un mensaje por parte de las otras dos aplicaciones, este recibe como parámetro el mensaje codificado.

Lo primero que realiza el método es la descomposición del mensaje en cadenas utilizando el símbolo “#” como delimitador, una vez obtenidos los valores de la cadena se encarga de realizar la acción adecuada mediante una serie de condiciones.

- ***Paquete communication***

Clase Client

Se trata de la misma clase utilizada en la librería de la tarjeta virtual por lo que su descripción coincide con la descripción de esta en la sección de la librería virtual.

Clase EchoThread

Se trata de una clase que extiende de Thread por lo que realiza su ejecución de forma asíncrona, esta es utilizada en el momento de recepción de petición por parte de alguna de las otras aplicaciones. Se instancia esta clase para que se encargue de la recepción del mensaje y no se realice de manera bloqueante en el servidor de la tarjeta de adquisición virtual.

Sus métodos son:

- EchoThread

Constructor de la clase que recibirá como argumentos de entrada la instancia del *socket* cliente del cual debe atender la petición y una instancia de la clase Main para poder hacer uso de su método *processMessage* y procesar el mensaje entrante.

- run

Método heredado de la clase Thread, es el método que se realiza en el momento del lanzado de la tarea asíncrona.

Este método ejecuta el código necesario para la recepción del mensaje y posterior envió al método *processMessage* de la clase Main.

Clase Server

Se trata de una clase que extiende de Thread por lo que realiza su ejecución de forma asíncrona, esta es instanciada desde la clase principal de la aplicación y su escucha es alojada en un puerto con la finalidad de atender las peticiones entrantes de alguna de las otras aplicaciones.

Posee los siguientes métodos:

- Server

Método constructor de clase que recibe como parámetros de entrada un identificador de servidor y una instancia de la clase Main.

Dentro del constructor se realiza la inicialización de las variables necesarias en la clase junto con la llamada de método connectServer.

- run

Método heredado de la clase Thread, es el método que se realiza en el momento del lanzado de la tarea asíncrona.

Este método ejecuta cada vez que recibe una petición de alguna de las aplicaciones y la acepta instancia un nuevo objeto del tipo EchoThread que obtiene el mensaje entrante y hace una llamada al método processMessage de la clase Main.

- close

Método que cierra las conexiones abiertas utilizadas para la recepción de mensajes y detiene la ejecución asíncrona lanzada para tal fin.

- getServerSocket

Método generado para la consulta del objeto instanciado serverSocket sobre el que se realizan las recepciones de mensajes.

- connectServer

Método que se encarga de encontrar un puerto disponible donde alojar el servidor de escucha para posteriormente instanciar el servidor con el puerto encontrado.

- ***Paquete model***

Clase Card

Como se ha comentado anteriormente esta clase es el modelo encargado de simular a la tarjeta de adquisición física, sus valores internos se encontrarán todos alojados en tres listas de valores que representan cada uno de los conectores de entrada y salida de los que dispone la tarjeta de adquisición de datos.

Existirán métodos genéricos para la consulta y modificación de los valores de cada una de las listas, además de métodos que inicialicen los valores de estas listan tomando unos valores por defecto que se encuentran almacenados en la clase Utils alojada en el paquete útil.

Además de estos métodos se dispone de los siguientes métodos:

- dac

Método que transforma un valor digital recibido como parámetro en un valor analógico de salida que devolverá en tipo flotante.

- adc

Método que transforma un valor analógico recibido como parámetro en un valor digital de salida que devolverá en tipo flotante.

Clase InputOutput

Esta clase contiene dos atributos que son clave y valor, representan el pin donde se aloja como entrada o salida y su valor actual,

Se decide generar este modelo para facilitar la lectura y escritura de los valores sobre las listas de valores, de manera que únicamente conociendo la clave se pueda modificar su valor.

Dentro de esta clase existen métodos genéricos de lectura y escritura de los dos atributos que posee.

- ***Paquete útil***

Clase Utils

Como se comenta anteriormente en el apartado de paquetes esta clase posee todos los métodos que dotan de funcionalidad a la aplicación y sean usados de manera reiterada en diferentes clases la aplicación.

En la clase Utils empleada en la tarjeta de adquisición virtual los métodos que posee son métodos para la inicialización de los valores por defecto de la tarjeta de adquisición virtual y la consulta o escritura de estos.

- ***Paquete view***

Clase CardInformationController

Clase que se encarga de controlar todo lo referente a la vista de la aplicación, desde la inicialización de todos los componentes visuales que se ven en la aplicación hasta el control de cada una de las acciones realizada sobre ellos.

Para ello posee los siguientes métodos:

- initialize

Primeramente, este método se encarga de la inicialización de todos los componentes gráficos que se aprecian en la vista de la aplicación.

Seguidamente se encarga añadir escuchadores a las tablas de la vista donde se cargarán los datos de las tres listas de valores de la clase Card, esto se realiza para que en el momento que se modifique un valor de la tarjeta virtual y cambie su valor en la tabla de valores que corresponda, se añada al final del registro una “V” indicando la unidad de medida.

Por último, se cargan sobre las tablas las listas de valores de la tarjeta virtual las cuales estarán continuamente sometidas a cambios y lecturas.

- changeValue

Método que recibe como parámetros de entrada el tipo de valor, el pin sobre el que realizar la acción y el valor a modificar.

Con estos valores y mediante unas condiciones se encarga de modificar el valor correspondiente sobre la lista de valores de la tarjeta de adquisición virtual, quedando reflejado el cambio sobre la tabla de valores de la vista.

- getValue

Método que recibe como parámetros de entrada el tipo de valor del que se desea realizar la lectura y el pin donde se encuentra alojado.

Mediante el uso de una serie de condiciones se procede a la lectura del valor de la lista de la tarjeta virtual que corresponda.

### *Paquete view.fxml*

Vista cardInformation

Este fichero se trata de la vista de la aplicación y se encuentra escrito en XML (Extensible Markup Language), dentro encontraremos todos los componentes visuales que componen la aplicación.

### *Paquete view.resources*

Recurso icon

Este archivo se trata de un recurso usado en la aplicación, concretamente se trata del icono utilizado para la ventana e icono de archivo de la aplicación que se encuentra en formato .png.

## Interfaz

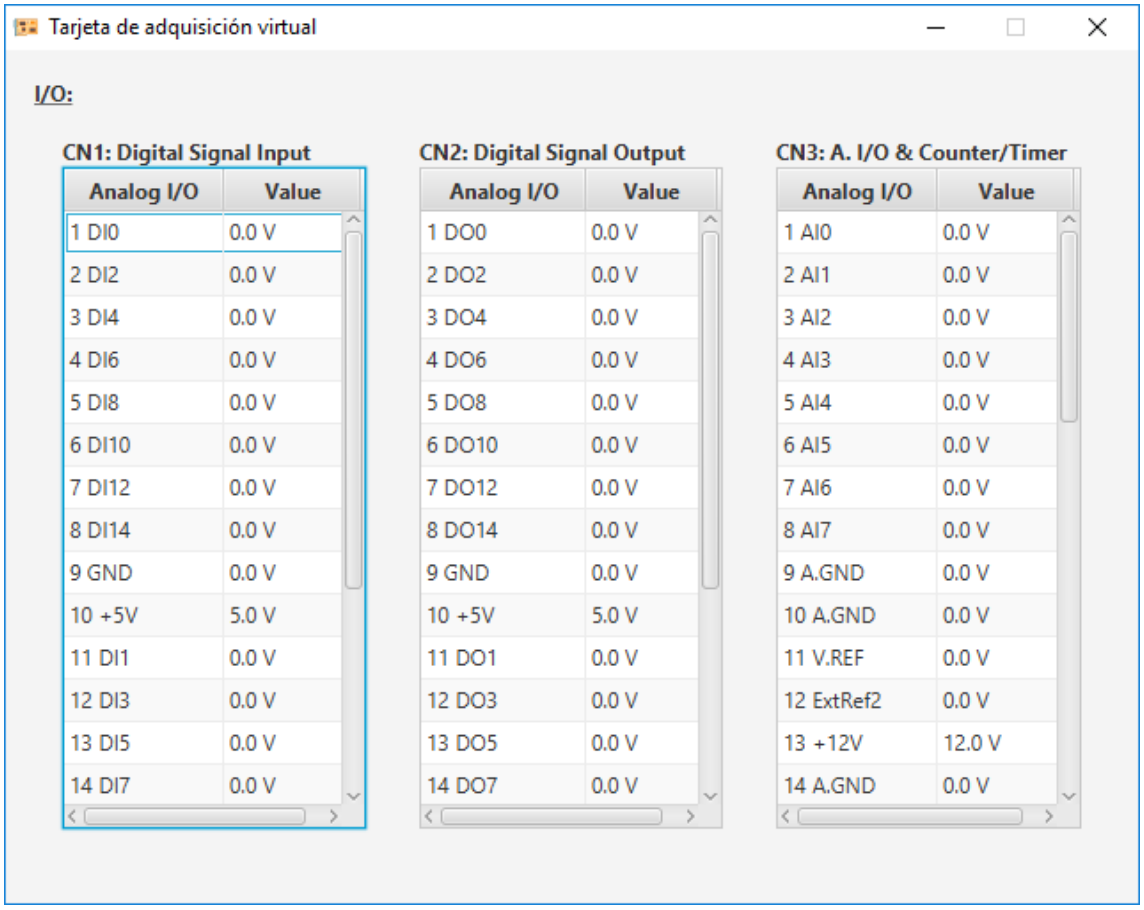


*Ilustración 5 Interfaz de usuario de la tarjeta de adquisición virtual*

Como se aprecia en la imagen anterior la interfaz de la tarjeta de adquisición virtual consiste en tres tablas de valores las cuales representan cada uno de los conectores que posee la tarjeta de adquisición de datos física original.

Para cada tabla tenemos dos columnas donde la primera nos indica el número de pin junto con el valor al que corresponde y en la segunda columna el valor con el que se encuentra en tiempo real ese pin.

En la primera tabla tenemos el conector número 1 que corresponde con el conector de entradas digitales.

En la segunda tabla que corresponde con el conector numero 2 tendremos todos los valores de la salida digital de la tarjeta.

Por último, en la última tabla tenemos los valores correspondientes a la entrada y salida analógica de la tarjeta de adquisición virtual.

## 4.5 Proceso de hardware virtual

### Estructura

La estructura de la aplicación del proceso de *hardware* virtual es la misma estructura que sigue la tarjeta de adquisición virtual puesto que se trata de una aplicación de similares características a excepción de que el proceso de *hardware* virtual no dispone de modelos puesto que no necesita el uso de ningún tipo de objeto representativo.

### Paquetes

La estructura de paquetes que se ha seguido es la misma que la utilizada en la tarjeta de adquisición virtual que es la siguiente:

- ***Paquete application***

Este paquete de clases contiene la clase Main, clase principal de la aplicación que tiene el código necesario para lanzar su ejecución de forma visual cargando la vista principal.

Además, esta clase posee el método que decodifica los mensajes entrantes, tanto de la librería virtual como del proceso de *hardware* virtual.

- ***Paquete communication***

Este paquete al igual que el mencionado anteriormente en la librería virtual contiene todas las clases necesarias para realizar las conexiones entre aplicaciones.

Dispone de una clase de una Cliente la cual se encarga de enviar los mensajes a las otras dos aplicaciones, una clase EchoThread la cual atenderá una petición concreta por parte de alguna de las dos aplicaciones, una clase Server que escucha en todo momento las peticiones de las otras dos aplicaciones de manera no bloqueante y una clase ServerQA que se encarga de realizar las peticiones con espera de respuesta a la tarjeta virtual como podría ser la conexión a esta con espera de confirmación.

- ***Paquete útil***

Paquete destinado a contener todas las clases que ofrezcan funcionalidades a la aplicación que sean de uso común en múltiples clases.

Contiene una clase Utils la cual contiene inicializara las listas de valores de la tarjeta virtual con valores predeterminados.

- ***Paquete view***

En este paquete se encuentra todo el contenido relacionado con la GUI de la aplicación, desde las vistas hasta los controladores de estas, incluyendo también los recursos necesarios en estas como podrían ser iconos e imágenes.

En su raíz contiene el controlador de la vista, además contiene en su interior otros dos paquetes que se mencionan a continuación.

- ***Paquete view.fxml***

Este paquete se encuentra dentro del paquete *view*, se crea este paquete para contener los *layouts* de las vistas, archivos que darán aspecto visual a la ventana de la aplicación.

- ***Paquete view.resources***

Este paquete contiene todos los recursos necesarios para emplear en la vista como por ejemplo podría ser el icono empleado en la ventana de la aplicación.

## Ficheros

Se procede a detallar la funcionalidad de cada uno de los ficheros ubicados en los distintos paquetes.

- ***Paquete application***

Clase Main

La clase Main del proceso de *hardware* virtual será la misma que la utilizado en la tarjeta virtual excepto dos cambios, el primero será el diferente funcionamiento en el método processMessage donde la funcionalidad será la misma a rasgos generales pero la estructura de condiciones que seguirá el método para la decodificación será diferente puesto que como se comenta en el apartado de protocolos el proceso de *hardware* virtual únicamente recibirá mensajes de la tarjeta virtual y la estructura de los mensajes que reciba vendrán sin identificador.

El segundo cambio mencionado será la inclusión de un nuevo método:

- searchCard

Método que se encarga de realizar la búsqueda de la tarjeta de adquisición virtual para realizar la conexión y establecer las comunicaciones.

### *Paquete communication*

Clase Client

Esta clase tiene el mismo funcionamiento y la misma implementación que la descrita en la clase Client de la tarjeta de adquisición virtual.

Clase EchoThread

Esta clase tiene el mismo funcionamiento y la misma implementación que la descrita en la clase EchoThread de la tarjeta de adquisición virtual.

Clase Server

Esta clase tiene el mismo funcionamiento y la misma implementación que la descrita en la clase Server de la tarjeta de adquisición virtual.

Clase ServerQA

Esta clase tiene el mismo funcionamiento y la misma implementación que la descrita en la clase ServerQA de la librería de la tarjeta virtual.

- ***Paquete útil***

Clase Utils

Esta clase tiene dos métodos que ofrecen funcionalidad de uso reiterado en las diferentes clases de la aplicación:

- round

Es un método desarrollado para el redondeo de cualquier cifra numérica dejándola únicamente con el número de decimales especificado como parámetro de entrada.

Para ello recibe como parámetros de entrada el número a redondear y el número de decimales que se quiere conseguir, mediante el uso de funciones matemáticas de la librería Math implementada en el lenguaje de programación java se deja el número con los decimales especificados y se devuelve como resultado de la ejecución del método.

- showAlert

Método que genera alertas modales de forma visual desde cualquier clase de la aplicación donde sea llamada, dichas alertas sirven para informar al usuario si fuera necesario del correcto envió de peticiones o la correcta conexión con la tarjeta virtual.

- ***Paquete view***

Clase HardwareProcessController

Esta clase es la encargada de controlar todos los componentes visuales que aparecen en la vista de la aplicación, para ello implementa los siguientes métodos:

- ***Paquete view.fxml***

Vista hardwareProcess

Este fichero se trata de la vista de la aplicación y se encuentra escrito en XML (Extensible Markup Language), dentro encontraremos todos los componentes visuales que componen la aplicación.

- ***Paquete view.resources***

Recurso green

Se trata de una imagen en formato .png que es utilizada para indicar de forma visual la correcta activación de una de las salidas digitales de la tarjeta de adquisición virtual.

Recurso measure

Imagen en formato .png que es utilizada como icono de la aplicación y la ventana de la interfaz de esta.

Recurso red

Imagen en formato .png utilizada para indicar que alguna de las salidas digitales de la tarjeta de adquisición virtual se encuentra desactivada.

Recurso voltage

Imagen utilizada al lado de los componentes que informan del estado de las salidas analógicas de la tarjeta de adquisición virtual para indicar el tipo de medida empleado.

## Interfaz

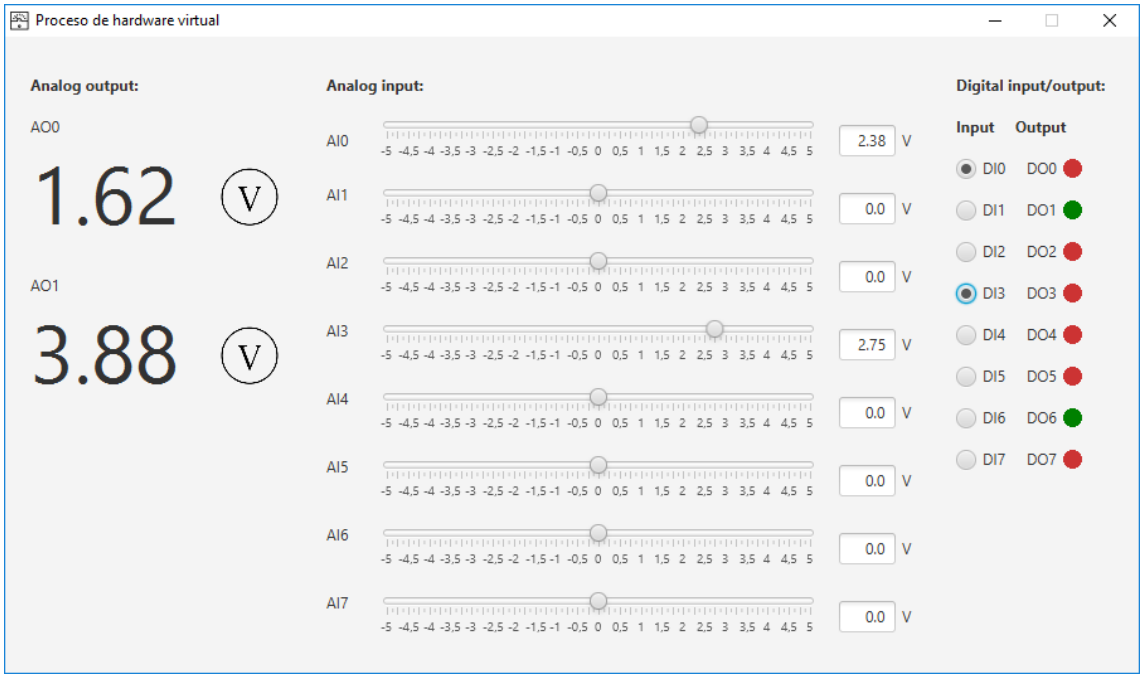


*Ilustración 6 Interfaz de usuario del proceso de hardware virtual*

En la imagen anterior se observa la interfaz del proceso de *hardware* virtual, en ella podemos observar diversos componentes:

- Los indicadores de salidas analógicas localizados en la parta izquierda de la ventana, estos nos indican el valor de salida que está dando la tarjeta de adquisición virtual para los canales analógicos.
- En la parte central tenemos *sliders* donde podemos modificar hasta 8 de las entradas analógicas que posee la tarjeta virtual.

- Al lado de los *sliders* tenemos cajas de texto donde saldrá reflejado el valor introducido en el *slider*, donde además podemos introducir un valor de manera manual que se verá reflejado en el *slider* y modificaran las entradas analógicas de la tarjeta.
- En la parte derecha de la ventana tendremos unos *radio button* que nos permiten modificar las entradas digitales de la tarjeta virtual.
- Junto a las entradas digitales en la parte derecha podemos observar el valor de las salidas digitales que posee la tarjeta en tiempo real.

# 5. Conclusiones

Se ha conseguido desarrollar con éxito el simulador de la tarjeta virtual, herramienta multiplataforma de la que dispondrán los alumnos de la asignatura Informatización Industrial para la realización de prácticas o labores de aprendizaje.

Se ha conseguido alcanzar con éxito las metas propuestas al inicio de la realización del proyecto. Primeramente, se pretendía que fuera una aplicación multiplataforma para que el alumnado no tuviera problemas en su uso y disfrute, además se ha conseguido desarrollar una herramienta que brinda las mismas características que el *hardware* real al que simula siendo además fiel a la manera en que funciona dicho *hardware*.

Además de todo esto se ha logrado una buena comunicación y entendimiento con el director de este proyecto, por lo que el software desarrollado estará dentro de las expectativas.

Por último, concluir diciendo que este proyecto me ha ayudado a desarrollar mis conocimientos de manera personal en una rama de la informática de la cual mis conocimientos eran limitados y me ha ayudado a comprender los problemas que surgen en el aprendizaje para el alumnado de estos conocimientos debido a los altos costes de los materiales necesarios para tal fin.

# Bibliografía


1. PCI-9112 - Data Acquisition - General-Purpose DAQ – ADLINK. Disponible en: http://www.adlinktech.com

2. ADLINK. (2001). NuDAQ / NuIPC 9112 Series Multi-function DAS Cards for PCI / 3U CompactPCI User's Guide [Manual de usuario]. Disponible en: http://www.adlinktech.com/publications/manual/NuIPC3UIO/CP9112_50-11111-201.pdf

3. CARRO PELLICER, MIGUEL. (17/07/07). PFC: "Simulador De Tarjeta De Adquisición De Datos Nudaq/Nuipc 9112 (Director Académico: Martí Campoy, Antonio)

4. A. Martí Campoy, J.C, Campelo, Serrano Martín, Juan José Alonso Díaz, Marina Coll Arnau, Salvador. (2011). Practical student teaching through integrated true, virtual and remote laboratories (Páginas 382 - 385). Editorial: SciTePress - Science and Technology Publications

5. A. Martí Campoy, J.C. Campelo Rivadulla, R. Ors Carot. (2001). Simulador de tarjeta de adquisición de datos (Páginas 1751 - 1759). IX Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas (CUIEET)

6. Martí Campoy, Antonio, Ors Carot, Rafael Pérez Jiménez, Alberto. (2001). Laboratorio de Informatización Industrial. Universitat Politècnica de València

7. Universitat Politècnica de València. (2015 - 2016). Apuntes de la asignatura Diseño de software.

8. Universitat Politècnica de València. (2014 - 2015). Apuntes de la asignatura Interfaces persona computados.

9. Universitat Politècnica de València. (2014 - 2015). Apuntes de la asignatura Redes de computadores.

10. Oracle. (2017). Información acerca de JavaFX. Disponible en: http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

11. Oracle. (2017). Información acerca de Java. Disponible en: https://www.java.com/en/download/faq/whatis_java.xml

12. Gluon. (2017). Información acerca de Gluon Scene Builder. Disponible en: http://gluonhq.com/products/scene-builder/

13. Mccdaq. (2004 - 2012). Data Acquisition Handbook. Disponible en: http://www.mccdaq.com/pdfs/anpdf/Data-Acquisition-Handbook.pdf

14. Joan Vila. (2008). Sockets en Java. Disponible en: https://poliformat.upv.es/access/content/group/OCW_6069_2008/T2.-Comunicaci%C3%B3n%20I%3A%20del%20C_S%20al%20modelo%20de%20objetos/Tecnolog%C3%ADa%20JAVA/Java-sockets.pdf

# Anexo 1: Manual de usuario

En este apartado se detallará de manera precisa tanto los pasos a seguir para la utilización de la librería en nuestros proyectos, así como el uso de cada una de las otras dos aplicaciones que componen este trabajo de final de grado, la tarjeta de adquisición virtual y el proceso de *hardware* virtual.

## 1. Manual de uso de la librería de la tarjeta virtual

Para usar la librería de la tarjeta virtual lo primero que tendremos que tener es la librería en formato .jar, preferiblemente en un sitio fácil de encontrar. Si fuera necesario y la distribución de esta se realizara comprimida primeramente sería necesario descomprimir él .jar.

En el ejemplo que aquí se relata se procede a dejar el fichero .jar de la librería en el mismo escritorio.

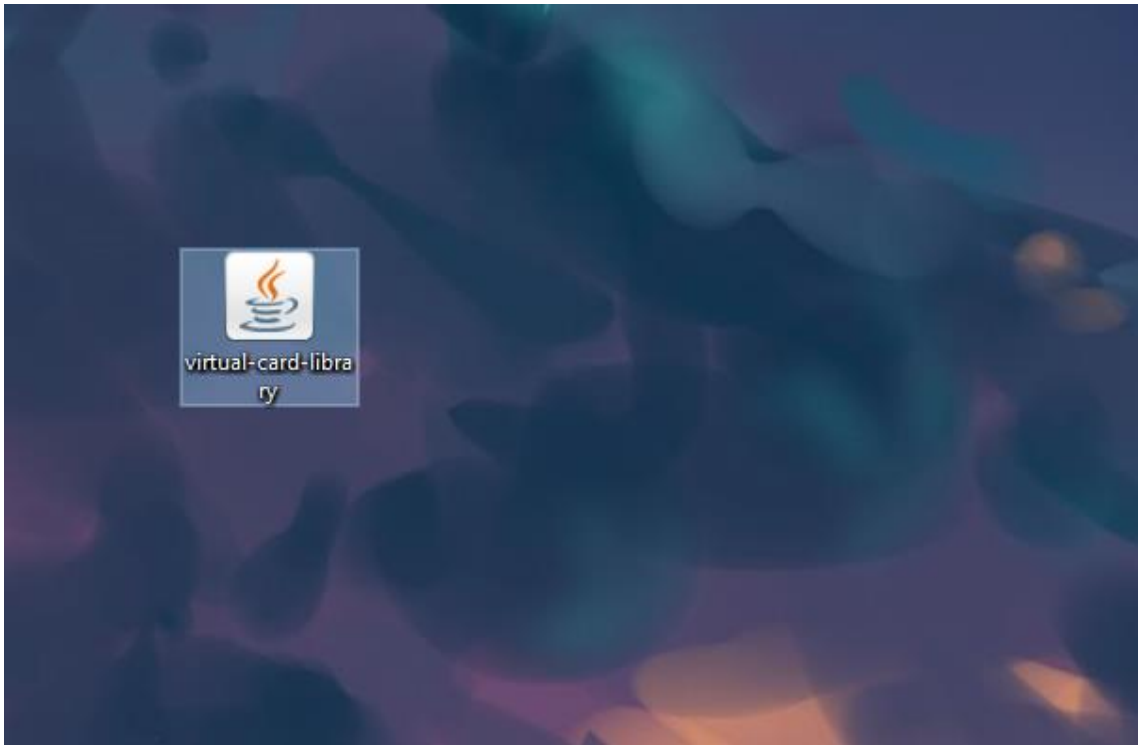


*Ilustración 7 Manual de uso de librería, virtual-card-library .jar*

El siguiente paso será crear el directorio donde alojaremos la librería para su uso, este directorio debe tener un nombre especifico y se ubicará dentro de la carpeta del proyecto donde se vaya a utilizar la librería, esto se hace para que la carga de la librería en la aplicación a desarrollar se haga desde una ruta relativa y no de problemas posteriormente en la distribución de la aplicación.

Este directorio recibirá el nombre de lib.

Como ejemplo de creación del directorio desde el IDE eclipse en su versión neon lo que tendremos que hacer es clic derecho sobre el proyecto donde se desea crear el directorio, en el submenú *new* seleccionar la opción *folder*.

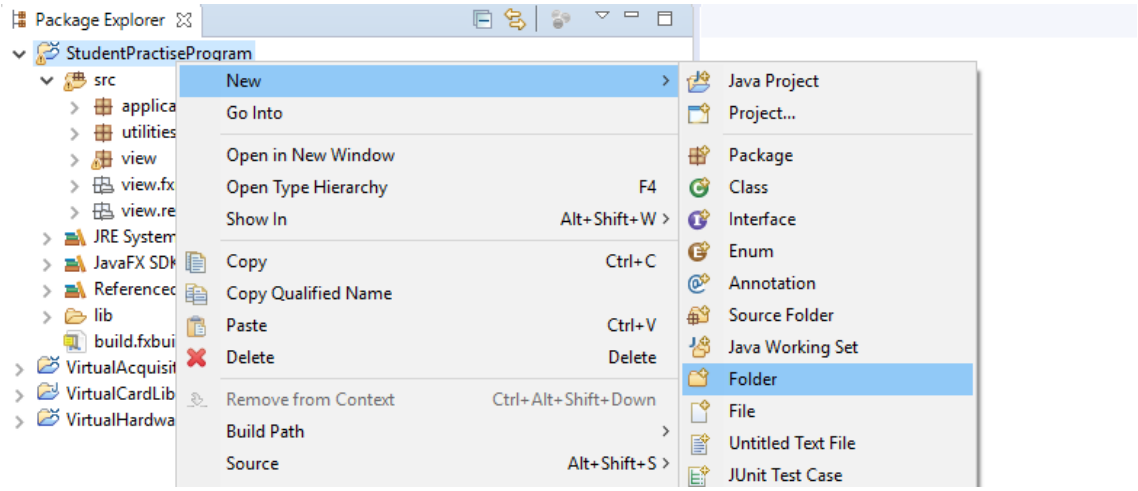


*Ilustración 8 Manual de uso de librería, nuevo directorio*

Posteriormente saldrá una ventana donde introduciremos el nombre del directorio y haremos clic en *finish*.

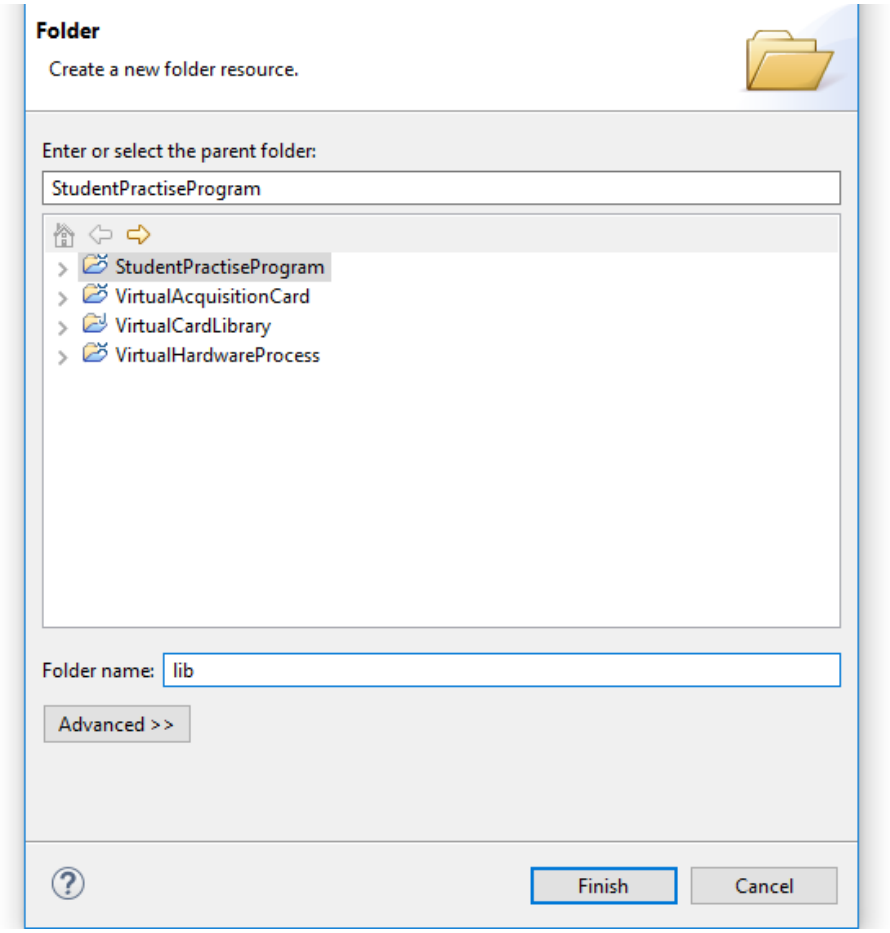


*Ilustración 9 Manual de usuario de librería, crear directorio*

Ahora con la carpeta creada solo tendremos que desplazarnos a donde se encuentra nuestra librería en formato .jar y hacer clic derecho sobre ella y seleccionar la opción del menú copiar para posteriormente pegarla sobre la carpeta en el IDE.

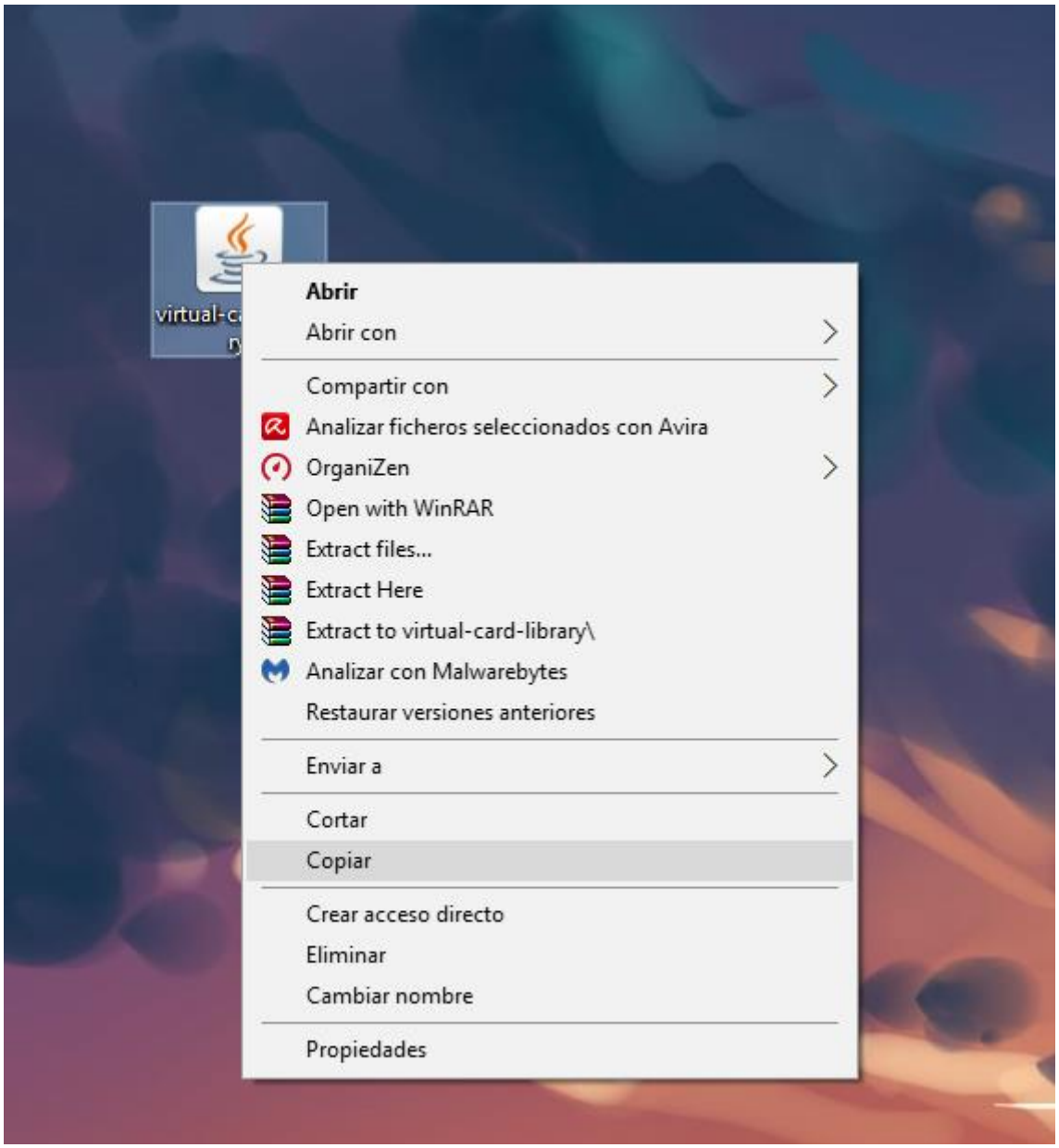


*Ilustración 10 Manual de usuario de librería, copiar .jar*

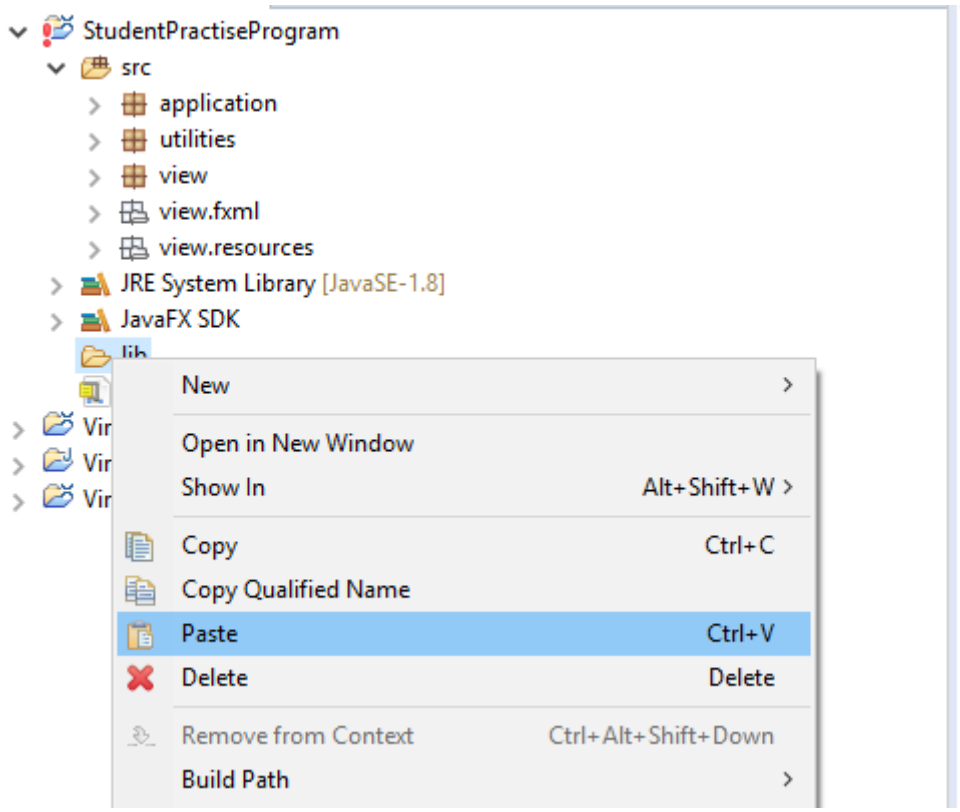


*Ilustración 11 Manual de usuario de librería, pegar .jar*

Quizá después de este paso sea necesario hacer clic derecho sobre el proyecto y seleccionar la opción *refresh* para que se vean reflejados los cambios sobre el proyecto.

Ahora con nuestra librería ya ubicada dentro del directorio procedemos a añadirla en la configuración del proyecto.

Para ello se tendrá que acceder a las opciones de compilación, la manera más sencilla de acceder a ellas es haciendo clic derecho sobre el proyecto y elegir la opción *propierties* ubicada al final del menú emergente.

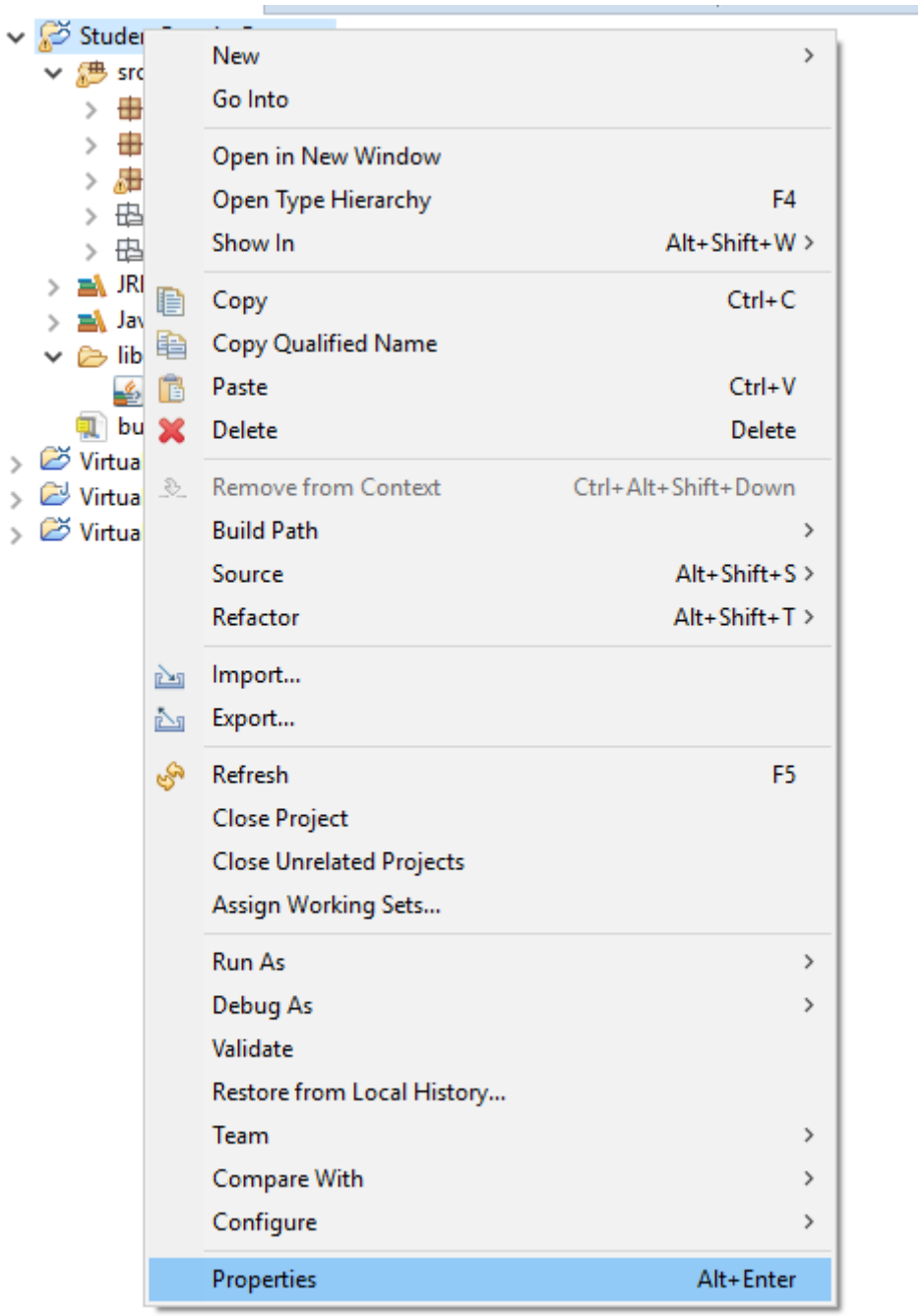


*Ilustración 12 Manual de usuario de librería, propiedades del proyecto*

Una vez dentro de las propiedades entraremos en la sección *Java Build Path*, donde dentro de esta veremos una pestaña llamada Libraries

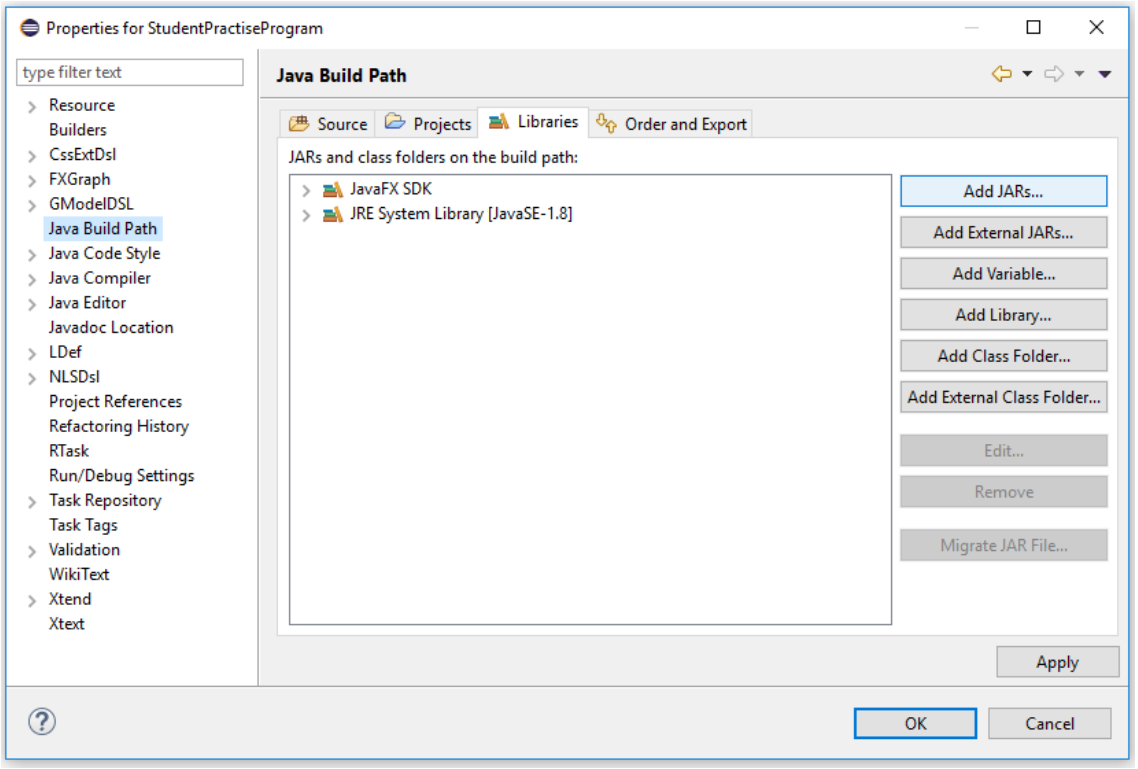


*Ilustración 13 Manual de usuario de librería, añadir .jar*

Aquí haremos clic sobre el botón *Add JARs* situado a la derecha junto a los demás botones.

Posteriormente solo tendremos que seleccionar la ubicación dentro del proyecto donde hemos alojado él .jar y ya podríamos hacer uso de la librería dentro de nuestro proyecto.

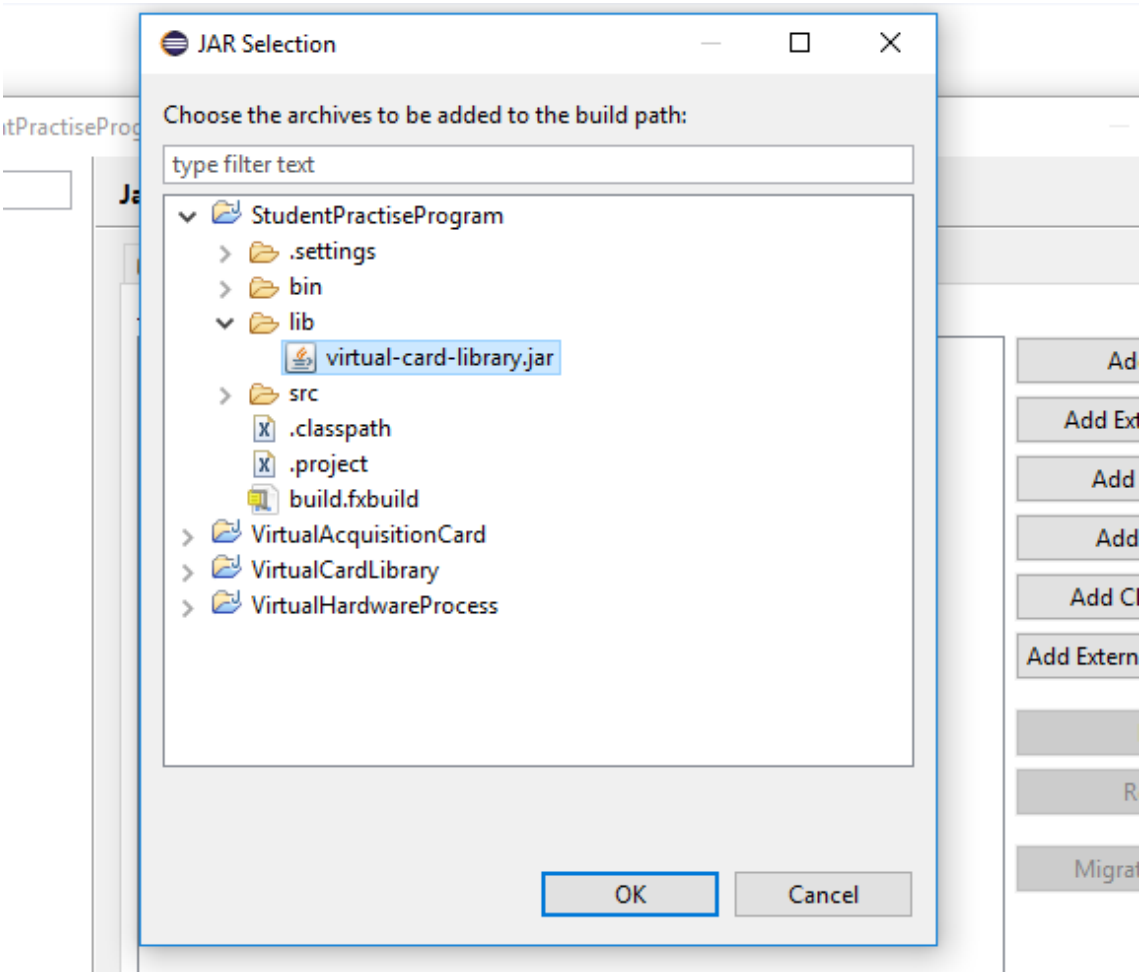


*Ilustración 14 Manual de usuario de librería, seleccionar .jar*

Por último, tendremos que importar la librería dentro de la clase donde vayamos a hacer uso de esta.

```
11 import javafx.scene.control.RadioButton;
12 import javafx.scene.control.Slider;
13 import javafx.scene.control.TextField;
14 import javafx.scene.input.MouseEvent;
15 import library.Library;
16 import utilities.Utils;
17
18 public class StudentProgramController {
19
```

*Ilustración 15 Manual de usuario de librería, importar librería*

Lo que nos permitirá hacer usos de sus clases, métodos o variables que posea.

```
    public void handle(MouseEvent event) {
        Library.AO_WriteChannel(31, (float) Utils.round(ao1.getValue(), 2));
        event.consume();
    }
});
```

*Ilustración 16 Manual de usuario de librería, ejemplo de uso*

## 2. Manual de uso de la tarjeta virtual y proceso de hardware virtual

En este apartado se detallará el cómo hacer uso de la tarjeta virtual y del proceso de *hardware* virtual de manera detallada.

Puesto que se trata de una aplicación desarrollada en java para hacer uso de la aplicación únicamente será necesario disponer de los archivos .jar de esta.

Por ello el primer paso a realizar será ubicar los archivos .jar en un sitio accesible y si fuera necesario descomprimirlo en caso de que lo obtuviéramos comprimido.

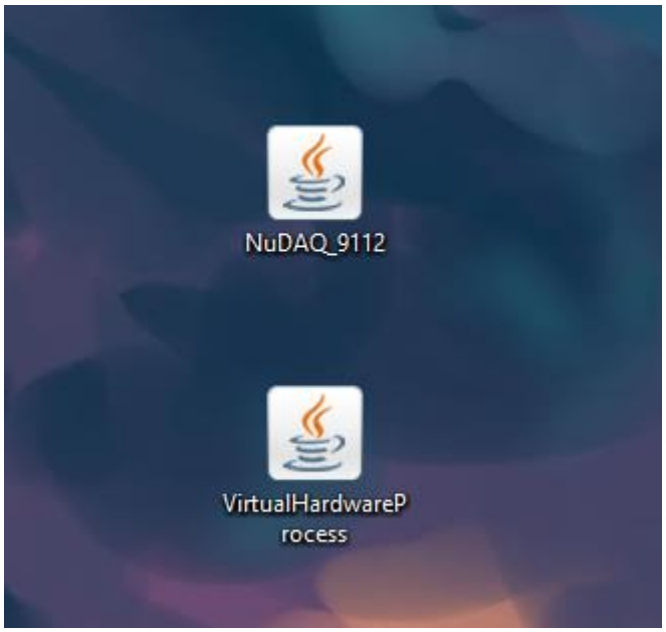


*Ilustración 17 Manual usuario de la tarjeta y proceso, aplicaciones*

El primero paso será lanzar la tarjeta de adquisición virtual llamada NuDAQ_9112, para ello bastará con hacer doble clic sobre el .jar.

El estado de la aplicación que se ejecutará será el siguiente:

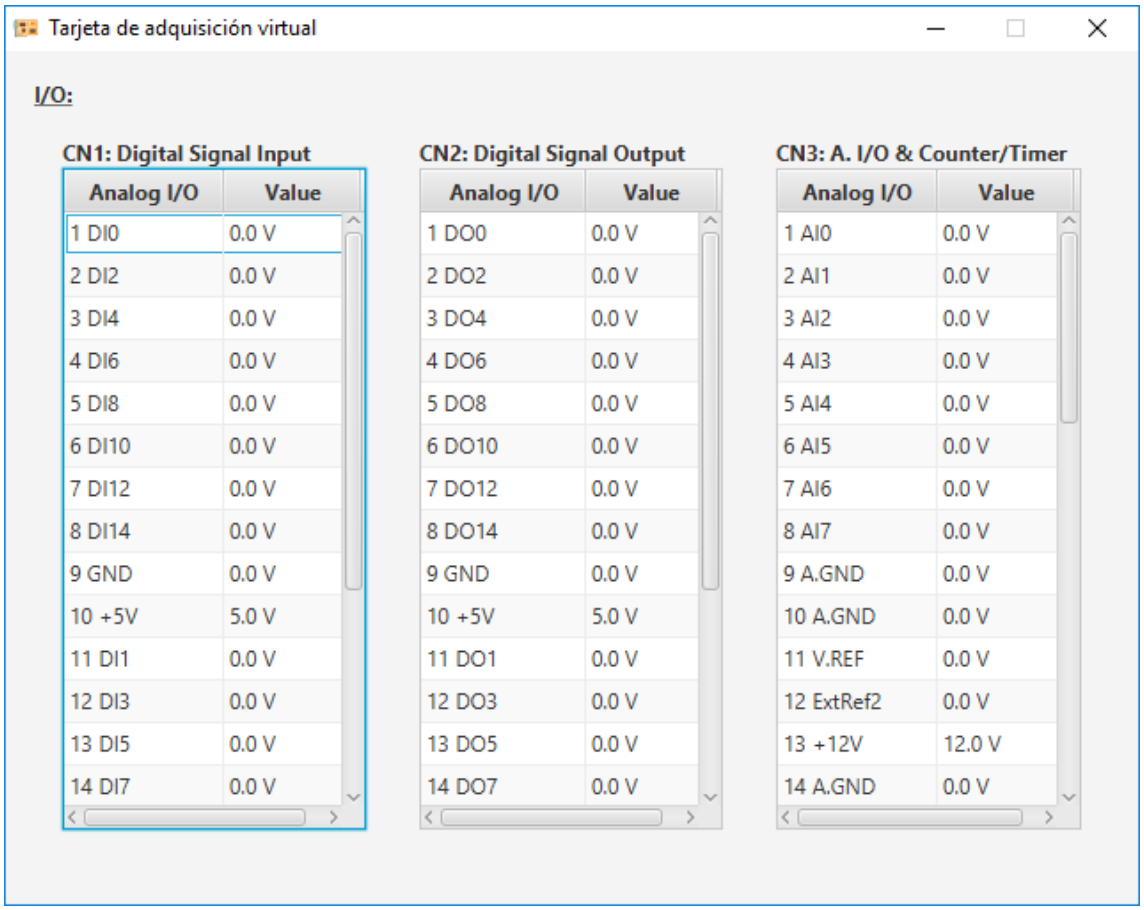


*Ilustración 18 Manual de usuario de la tarjeta y proceso, tarjeta virtual*

Esta aplicación dispone de tres tablas donde veremos reflejados todos los datos de la tarjeta de adquisición virtual.

Como siguiente paso tendremos que lanzar el proceso de *hardware* virtual llamada VirtualHardwareProcess, bastara con hacer doble clic sobre el .jar.

En el momento de ejecución del proceso de *hardware* virtual este nos indicara mediante una ventana informativa que se ha conectado de manera automática a la tarjeta virtual.

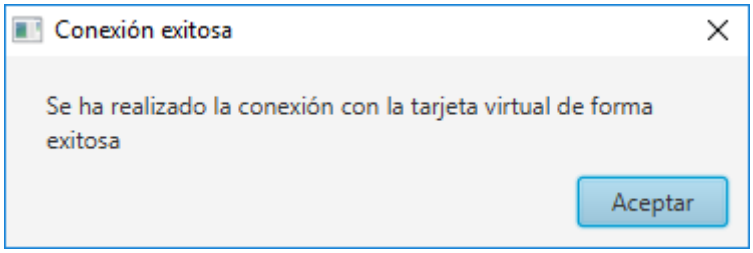


*Ilustración 19 Manual de usuario de la tarjeta y proceso, confirmación de conexión de proceso*

Si en el momento de ejecución no se encontrara en ejecución la tarjeta de adquisición virtual lanzada la aplicación nos informaría que es imposible conectar y podremos reconectar con la tarjeta mediante una opción localizada en el menú superior de la aplicación llamado opciones

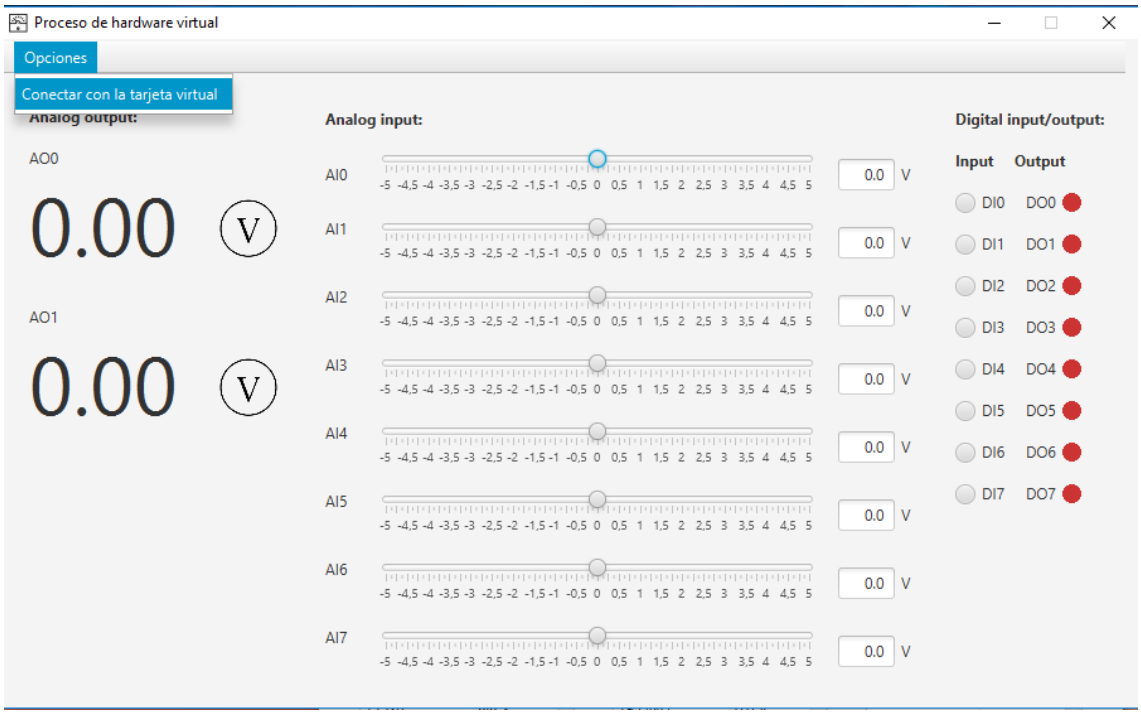


*Ilustración 20 Manual de usuario de la tarjeta y proceso, volver a conectar proceso*

Ahora simplemente consistiría en interactuar con el proceso de *hardware* virtual para ver reflejado algún cambio sobre la tarjeta virtual.

Si quisiéramos realizar un cambio sobre las entradas analógicas es posible deslizar alguno de los *sliders* o introducir el valor de manera manual en las cajas de texto del centro de la aplicación de la siguiente manera.

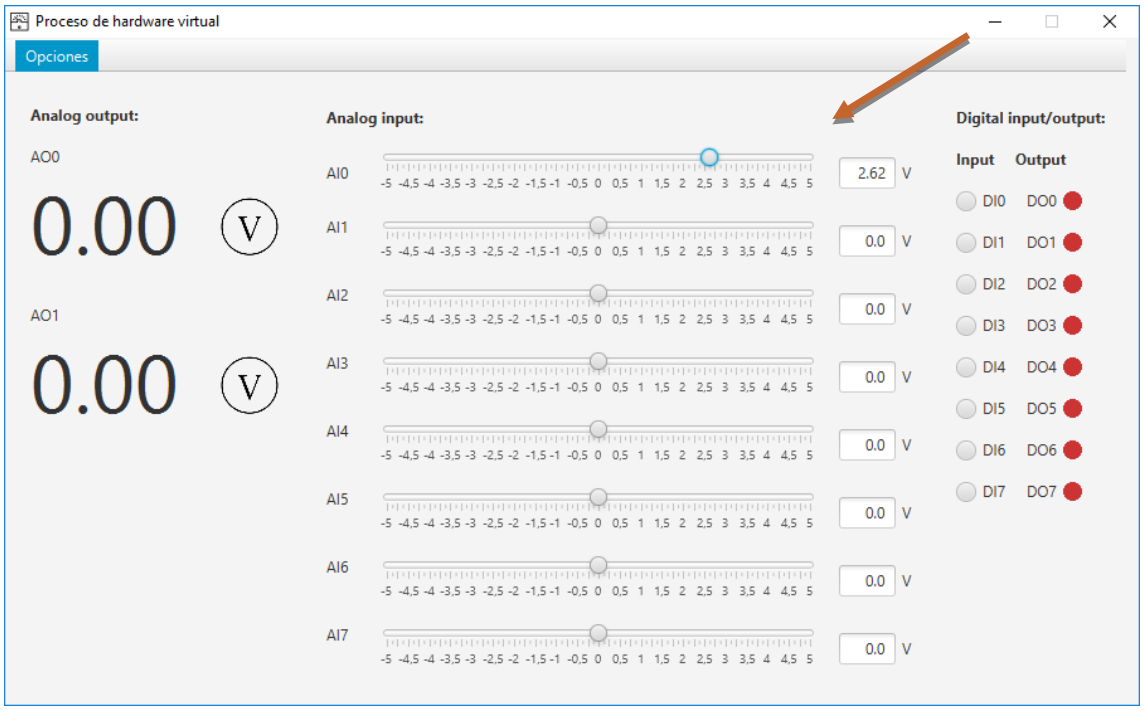


*Ilustración 21 Manual de usuario de la tarjeta y proceso, modificar valor de entrada analógica*

Automáticamente después de realizar la acción veríamos reflejado el cambio sobre la tarjeta virtual.

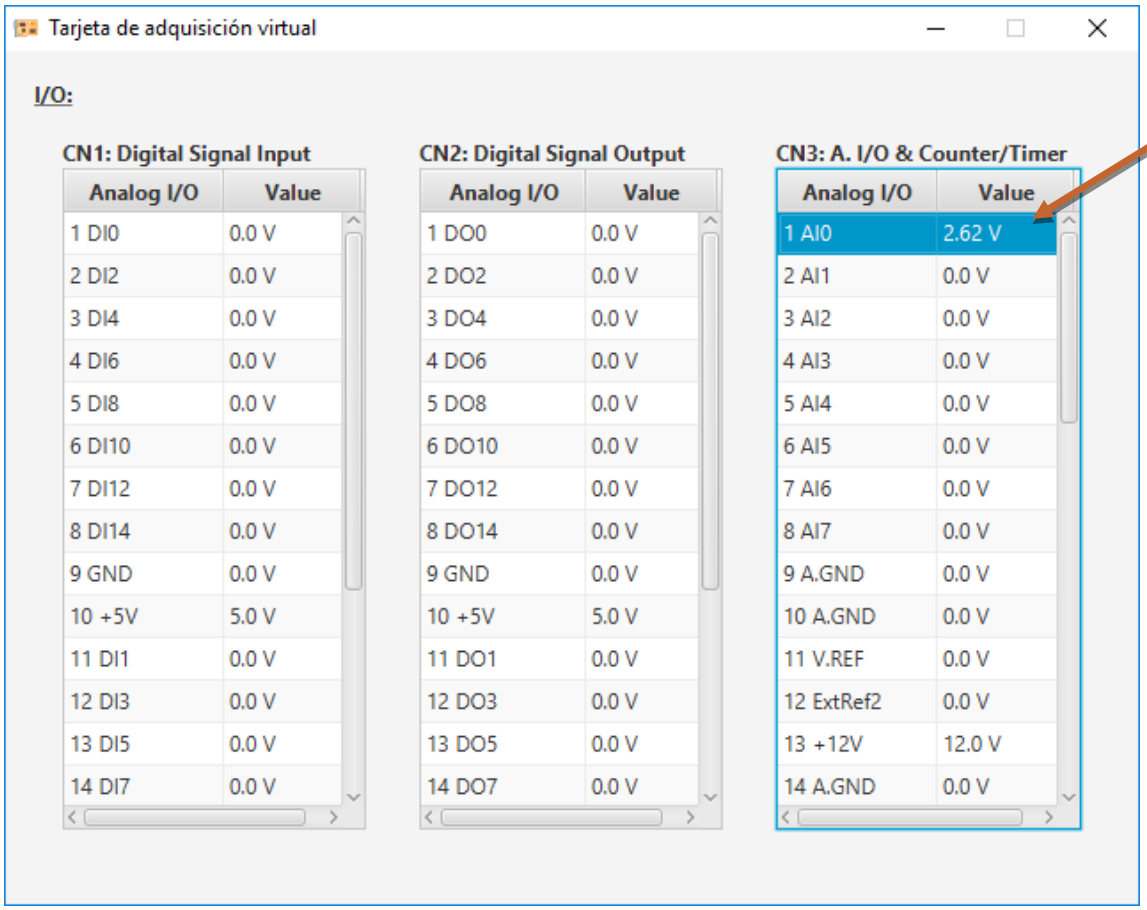


*Ilustración 22 Manual de usuario de la tarjeta y proceso, cambio de entrada analógica en tarjeta*

De igual manera si realizáramos un cambio sobre las entradas digitales localizadas a la derecha lo veríamos reflejado en la tarjeta virtual.

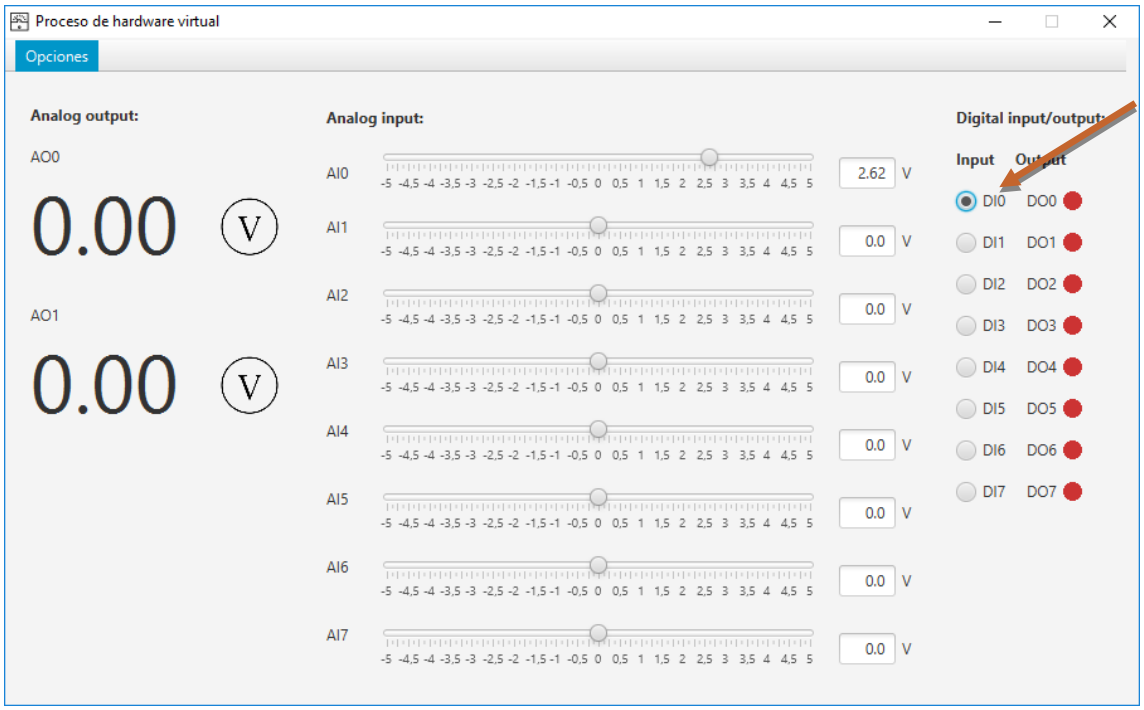


*Ilustración 23 Manual de usuario de la tarjeta y proceso, cambio de entrada digital*

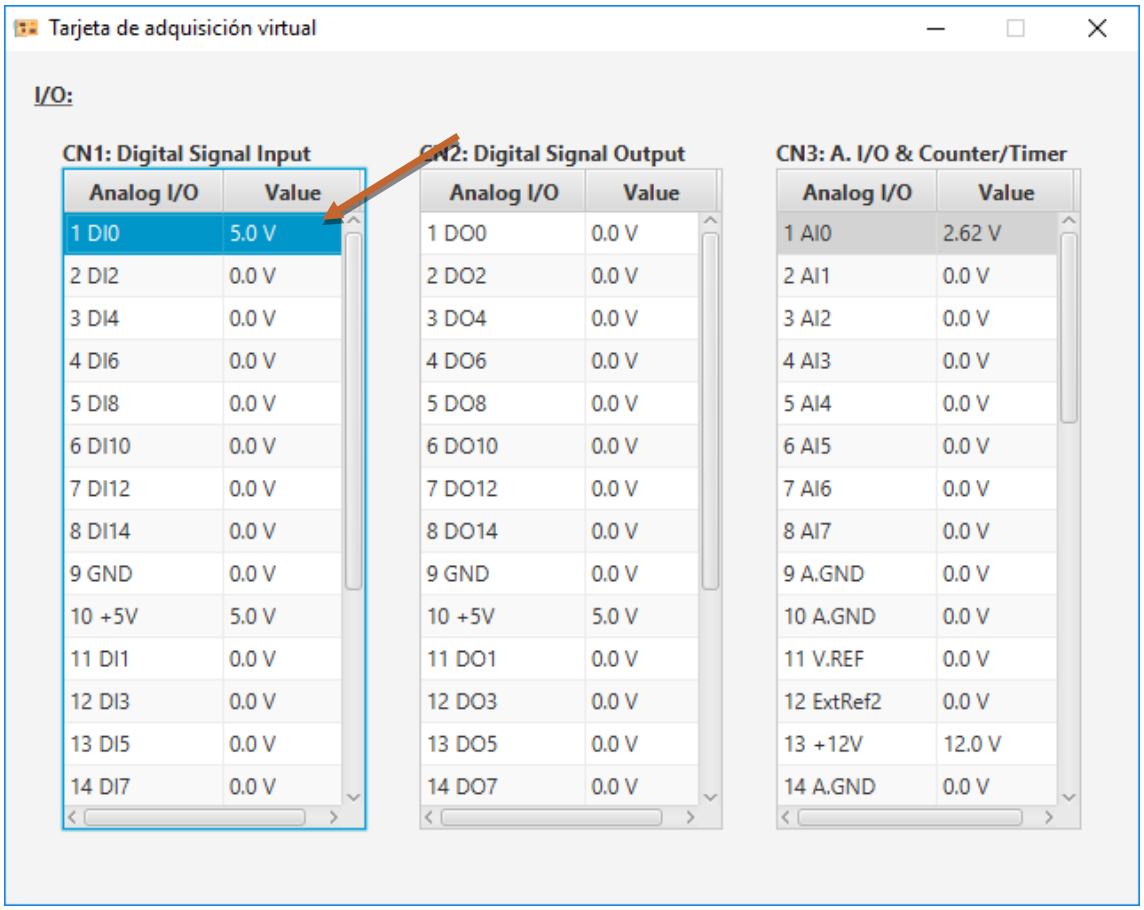


*Ilustración 24 Manual de usuario de la tarjeta y proceso, cambio de entrada digital en tarjeta*

Si se hiciera uso de la librería virtual realizando cambios sobre alguna de las salidas analógicas o digitales de la tarjeta estas quedarían reflejadas tanto en la tarjeta virtual como en el proceso de *hardware* virtual.

Aquí se muestra una aplicación de prueba realizada para comprobar el correcto funcionamiento de la librería virtual.

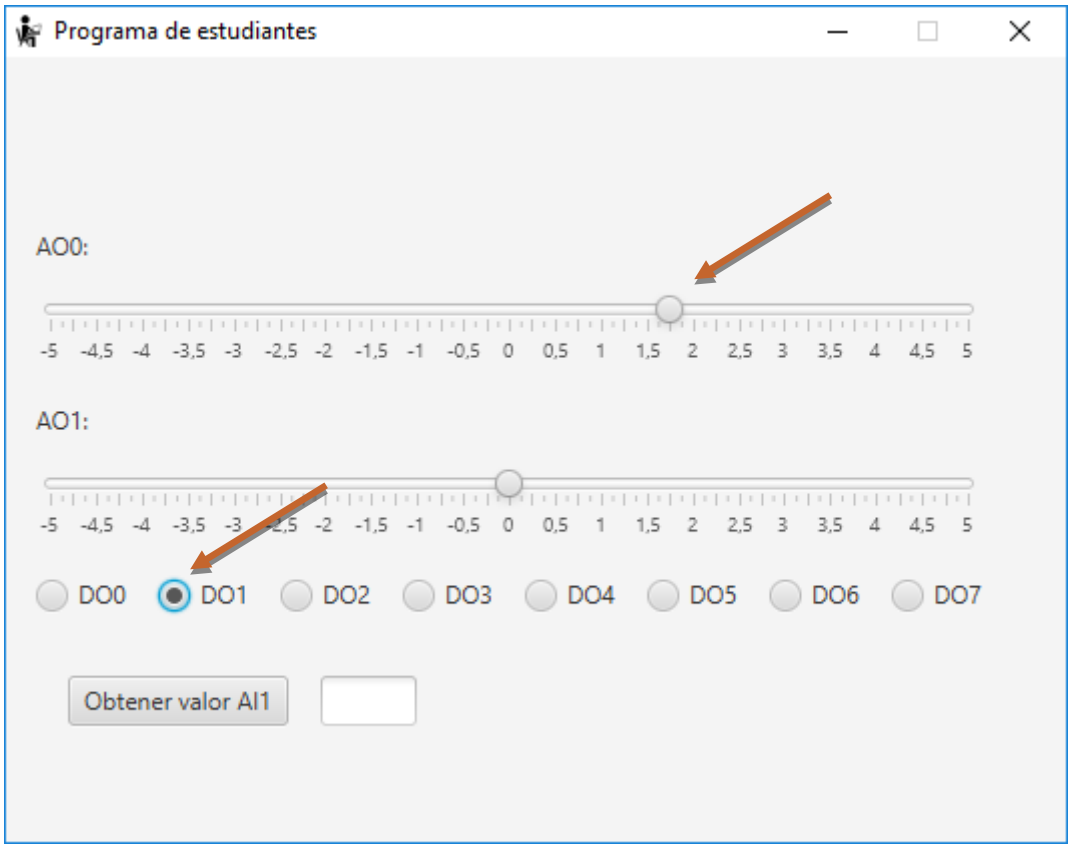


*Ilustración 25 Manual de usuario de la tarjeta y proceso, aplicación de prueba de la librería*

Vemos que tras realizar un cambio sobre la salida analógica número 0 y la salida digital numero 1 los cambios se ven reflejados tanto en la tarjeta como en el proceso.

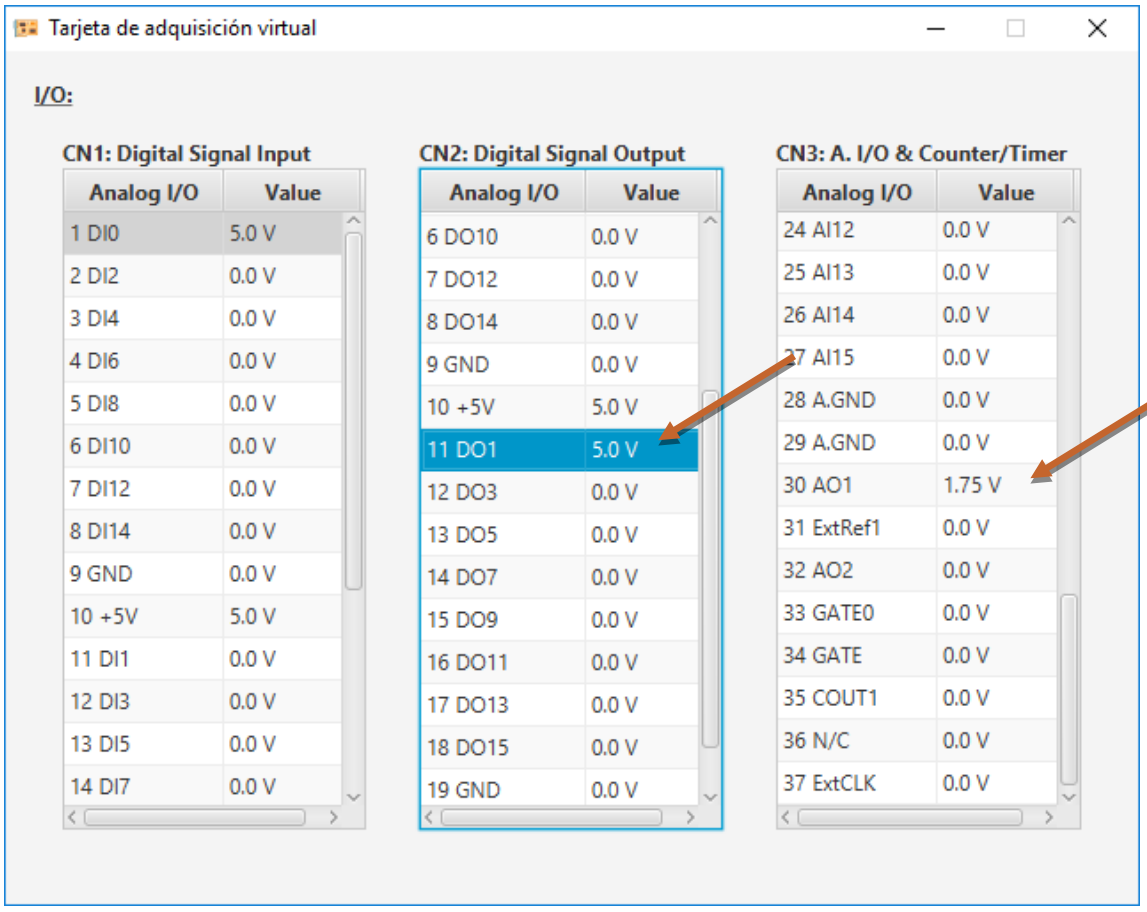


*Ilustración 26 Manual de usuario de la tarjeta y proceso, cambios de la librería reflejados en la tarjeta*

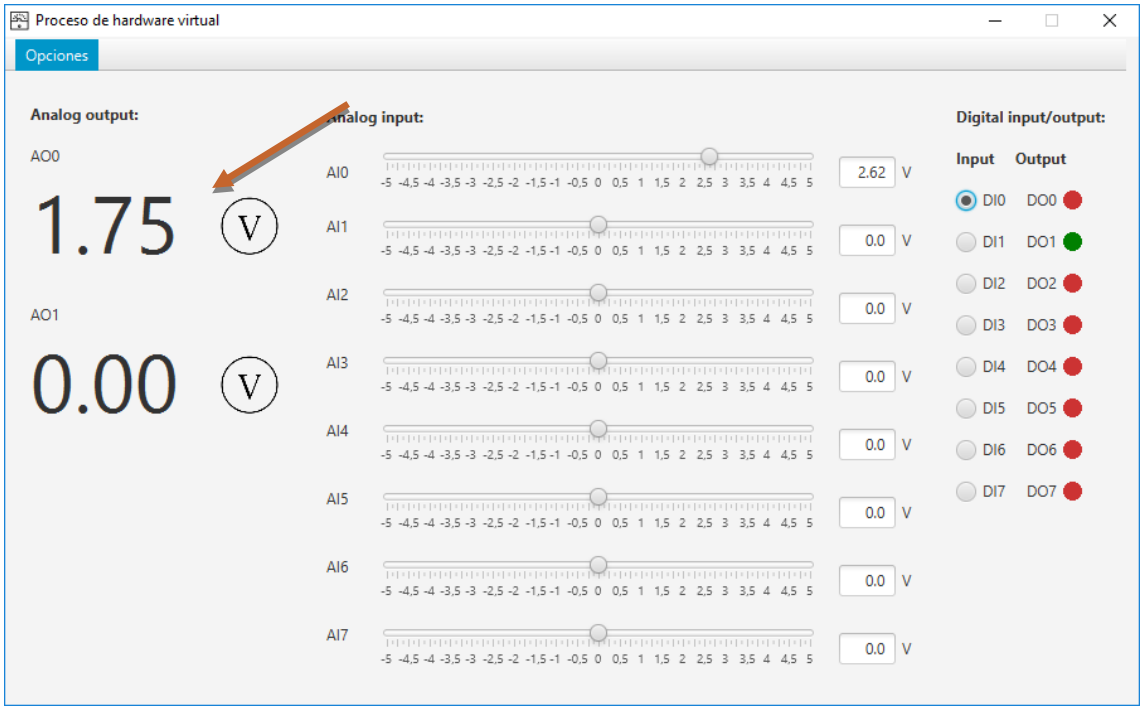


*Ilustración 27 Manual de usuario de la tarjeta y proceso, cambio de la librería reflejados en el proceso*

Para finalizar con el trabajo solo tendríamos que cerrar las aplicaciones.