UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Simulador gráfico de un proceso de empaquetado de naranjas para tarjetas de adquisición de datos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jaume Masiá Izquierdo

Tutor: Antonio Martí Campoy

2016/2017

Resumen

Este proyecto consiste en desarrollar una aplicación Java que simule gráficamente el completo funcionamiento de un proceso de empaquetado de naranjas. Además, el proceso de empaquetado se comunicará con una tarjeta virtual con la que intercambiará datos de sensores y actuadores. Finalmente, mediante un programa en C se realizará el control de su funcionamiento.

Palabras clave: proceso, tarjeta de adquisición, virtual, naranjas, simulador, sensores, actuadores, java.

Resum

Aquest projecte consisteix en desenvolupar una aplicació Java que simule gràficament el complet funcionament d'un procés d'empaquetat de taronges. A més, el procés d'empaquetat es comunicarà amb una targeta virtual en la que intercanviarà dades de sensors i actuadors. Finalment, mitjançant un programa en C es realitzarà el control del seu funcionament.

Paraules clau: procés, targeta d'adquisició, virtual, taronges, simulador, sensors, actuadors, java.

Abstract

This Project consist in developing a Java application that in a graphic way simulate the complete work of an orange packaging process. Moreover, this packaging process will be communicated with a virtual card which will exchange sensors and actuators data. Finally, the programme C will be used to make the control of the functioning.

Keywords: process, acquisition card, virtual, oranges, simulator, sensor, actuator, java.

Índice

Índice de figuras

Índice de tablas

1. Introducción

La presente memoria describe el trabajo realizado en el actual trabajo final de grado.

En el laboratorio de prácticas de la asignatura “Informatización Industrial”, del grado en ingeniería en tecnologías industriales de la ETSII de la UPV se utiliza la tarjeta “NuDAQ/NuIPC 9112 Series”. Junto a la tarjeta se utiliza un simulador hardware de procesos conectado a dicha tarjeta. Los valores de las entradas de la tarjeta eran modificados por los alumnos actuando manualmente sobre el proceso. Un programa de ordenador desarrollado por los alumnos reaccionaba a estas entradas y modificaba las salidas de la tarjeta, controlando de este modo el proceso hardware.

El principal problema residía en que, tanto la tarjeta como el simulador hardware tienen un coste muy elevado para los alumnos, por lo que impedía que estos pudieran disponer de estos para practicar en sus respectivas casas. Así pues, se decidió desarrollar, en el marco de un Proyecto Final de Carrera, unas herramientas software equivalentes al hardware descrito anteriormente.

Miguel Carro Pellicer en su proyecto final de carrera “Simulador de tarjeta de adquisición de Datos ‘NuDAQ / NuIPC 9112 Series’ ” presentado en 2007 implementó dichas herramientas software con el objetivo de simular el completo funcionamiento del hardware utilizado en las prácticas de la asignatura.

Las herramientas implementadas son las siguientes:

- Tarjeta virtual de adquisición de datos simulando la tarjeta “NuDAQ / NuIPC 9112 Series”
- Simulador software de procesos que permite modificar las entradas de la tarjeta y visualizar las salidas de la tarjeta virtual.
- Librería de la tarjeta modificada para comunicarse con procesos virtuales.

Miguel Carro (2007, p.41) incluía como futuras implementaciones: “mejorar el proceso virtual existente añadiendo funcionalidades, o basar nuevos procesos virtuales en otras ideas”. Es aquí donde se enmarca el actual TFG.

2. Objetivos

El objetivo de este trabajo final de grado consiste en desarrollar un simulador gráfico de un proceso industrial con sensores y actuadores, que se conectará a la Tarjeta virtual y su librería. De esta manera los alumnos podrán realizar prácticas con un simulador de proceso industrial sin necesidad del hardware.

El proceso virtual se comunicará únicamente con la tarjeta virtual de adquisición de datos mediante el envío y recepción de datos expresados en voltios, estos datos nos permitirán simular sensores y actuadores en el proceso. Por otro lado, el alumno podrá interactuar con parte de la interfaz gráfica del proceso virtual.

También deberemos proporcionar un programa que utilice la librería de la tarjeta y con el que se probará el correcto funcionamiento del simulador gráfico. Este programa será el que deberá hacer el alumnado de prácticas en un futuro.

El simulador a desarrollar consistirá en un proceso de encajonamiento de naranjas, en el cual se simulará el correcto funcionamiento de carga y desplazamiento de las cajas en un sistema continuo. El alumnado deberá programar correctamente dicha carga con las cantidades adecuadas y los actuadores para el movimiento de las cintas que transportan las cajas.

3. Especificación de requisitos

En este apartado expondremos la especificación de requisitos software de la aplicación que vamos a implementar. En primer lugar, realizaremos una pequeña introducción en la que comentaremos el propósito de la aplicación y su contexto.

Seguidamente, se hará referencia a la perspectiva del producto, las funciones principales, las características de los usuarios y las restricciones y suposiciones más importantes.

Posteriormente, trataremos los requisitos funcionales y no funcionales.

3.1. Introducción

En esta pequeña introducción hablaremos del propósito y del ámbito del sistema. Para ello utilizaremos el estándar IEEE 830/1998 de ERS en el que se describen todos los pasos para realizar un buen documento de especificación de requisitos. Como se trata de un proyecto de pequeña envergadura y con fines académicos, algunos de los puntos del estándar no se pueden aplicar y por ese motivo no aparecerán en el documento.

3.1.1. Propósito

El propósito principal de este punto es definir de forma clara todas las funcionalidades del sistema, así como sus restricciones principales.

3.1.2. Ámbito del sistema

En este proyecto implementaremos una aplicación que simule un proceso virtual que nombraremos como Proceso de empaquetado de naranjas. Este consistirá en la simulación de llenado de diferentes cajas con determinadas cantidades de naranjas. Todo el sistema estará dotado de diversos sensores y actuadores que se comunicarán con la tarjeta virtual mediante unos protocolos específicos.

El usuario del sistema mantendrá una interacción con la interfaz de la aplicación, encargándose de colocar en la simulación gráfica una caja de madera, una de plástico o una de cartón de forma aleatoria.

El principal beneficio que nos proporciona esta aplicación es la de proporcionar al usuario una herramienta para poder hacer uso de la tarjeta de adquisición de datos ‘NuDAQ / NuIPC 9112 Series’ sin necesidad de utilizar un proceso industrial real. Además, el usuario podrá mediante la implementación de un pequeño programa hacer uso de los diferentes sensores y actuadores para verificar el correcto funcionamiento del proceso.

3.1.3. Definiciones, Acrónimos y Abreviaturas

- ERS (Especificación de requisitos del sistema): Es el estándar utilizado en nuestro proyecto.

- JNA (*Java Native Acces*): Nos permite incorporar librerías de otros sistemas en nuestro programa Java.
- DO: (*Digital output*): Representa las salidas digitales de la tarjeta virtual.
- AO: (*Analog output*): Representa las salidas analógicas de la tarjeta virtual.

3.1.4. Referencias

- IEEE Recommended Practice for Software Requeriments Specification. ANSI/IEEE 830/1998

3.1.5. Visión general del documento

El documento de ERS está dividido en tres secciones. En la primera que es en la que nos encontramos ahora, nos proporciona una visión general del documento ERS. En la segunda sección, se describen, de forma general, las principales funciones que vamos a implementar. Por último, en la tercera sección, se definen detalladamente los requisitos que debe cumplir el sistema.

3.2. Descripción general

3.2.1. Perspectiva del producto

En este proyecto se pretende desarrollar una aplicación que simule un proceso industrial que estará conectado a una tarjeta de adquisición de datos virtual respetando sus protocolos de intercomunicación. Esta tarjeta estará a su vez conectada con su librería y con el programa de prácticas del alumno, como se puede ver en la Figura 1.

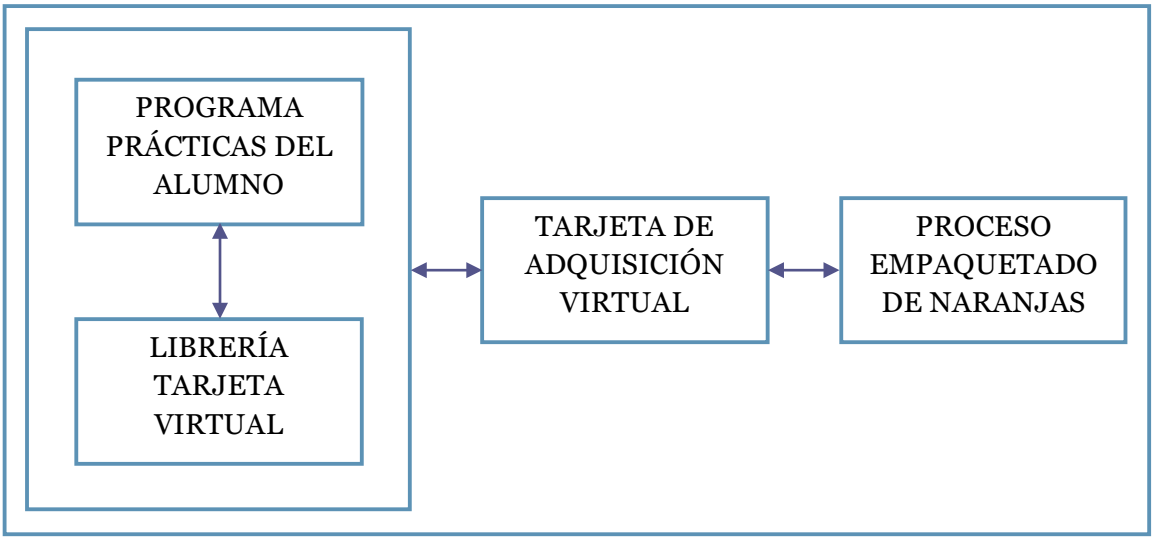


Figura 1: Esquema del sistema

Este programa del alumno podrá mediante la utilización de las diferentes entradas y salidas de la tarjeta implementar un algoritmo que realice el control para obtener el correcto funcionamiento del proceso mostrado en la aplicación.

3.2.2. Funciones del producto

A continuación, se describen las funciones que tendrá la aplicación que deseamos implementar:

- Comunicación con la tarjeta: El proceso podrá visualizar y/o actualizar los datos de las entradas y salidas de la tarjeta virtual.
- Simulación de sensores y actuadores: El proceso simulará sensores y actuadores conectados a la tarjeta.
- Simulación gráfica del funcionamiento físico del proceso: Simulará visualmente el proceso de empaquetado de naranjas.
- Validación del proceso: El usuario implementará un programa para el correcto control del proceso que, a su vez, dicho proceso simulará el resultado ya sea correcto o incorrecto.
- Interacción del usuario: El usuario se encargará de colocar en la simulación una caja de forma aleatoria entre los tres tipos existentes.

3.2.3. Características de los usuarios

El tipo de usuario que interactuará con la aplicación serán los alumnos de la asignatura. Estos serán los encargados de implementar el programa del alumno que utiliza la librería de la tarjeta virtual, enviando o recibiendo los datos de los actuadores y sensores de la aplicación a través de la tarjeta para simular el correcto funcionamiento del proceso. Por lo que, los alumnos que hayan implementado el programa de control serán los que interactúen con la aplicación para poner las cajas en el simulador y de este modo, visualizar el correcto funcionamiento o, en el caso de tener un error en la implementación de su programa, el incorrecto funcionamiento del proceso.

3.2.4. Restricciones

A continuación, enumeraremos las restricciones más importantes que debe respetar nuestra aplicación:

- Los mecanismos de intercomunicación entre procesos deben respetar los protocolos de la tarjeta virtual.
- La aplicación debe poderse visualizar correctamente en pantallas con resolución superior a 1440 x 1080.
- La aplicación debe funcionar perfectamente en el sistema operativo Windows 8.
- La aplicación debe ser la encargada de ejecutar la tarjeta virtual y también de cerrarla al mismo tiempo que se cierra la aplicación.

3.2.5. Suposiciones y dependencias

La aplicación está diseñada suponiendo que el ordenador que el alumno utilice tiene Windows 8, la máquina virtual de Java instalada, y algún entorno de desarrollo en lenguaje C para realizar el programa del alumno. Además, el ordenador se caracterizará por tener disco duro y memoria suficiente para almacenar y ejecutar simultáneamente las tres aplicaciones (5Gb y 2Gb respectivamente). Y por último, se utilizarán monitores que permitan una resolución igual o superior a 1440 x 1080.

3.3. Requisitos específicos

En este apartado nos centraremos en los requisitos principales del sistema, describiendo los requisitos funcionales y los de rendimiento.

3.3.1. Requisitos funcionales

A continuación, enumeraremos todos los requisitos funcionales de la aplicación describiendo de forma más detallada cada uno de ellos.

Número de requisito	**1**
Nombre del requisito	Ejecutar la tarjeta virtual desde la aplicación
Descripción del requisito	La aplicación deberá crear un proceso para poder ejecutar paralelamente la tarjeta virtual al abrirla.

Número de requisito	**2**
Nombre del requisito	Comunicar el proceso con la tarjeta de manera correcta
Descripción del requisito	Se crearán *pipes* en la aplicación de Java que utilicen los mismos protocolos que la tarjeta virtual para poder tener una conexión estable y correcta con ésta.

Número de requisito	**3**
Nombre del requisito	Simular los sensores y actuadores
Descripción del requisito	La aplicación simulará todos los sensores y actuadores existentes en el proceso virtual.

Número de requisito	**4**
Nombre del requisito	Simular gráficamente el funcionamiento físico del proceso
Descripción del requisito	La aplicación deberá simular gráficamente lo más aproximado a la realidad todo el funcionamiento del proceso industrial.

Número de requisito	**5**
Nombre del requisito	Validar el correcto control del simulador
Descripción del requisito	El simulador tendrá diferentes escenas gráficas para simular diferentes soluciones al control del proceso, tanto correctas como incorrectas.

Número de requisito	**6**
Nombre del requisito	Interacción de los alumnos con la aplicación
Descripción del requisito	El alumno interactuará con la aplicación mediante un botón que pondrá cajas de forma aleatoria en la cinta.

3.3.2. Requisitos de rendimiento.

La aplicación desarrollada deberá comunicar y transmitir datos con la tarjeta virtual de forma instantánea. Solo tendrá un tiempo de espera a la hora de ejecutar la tarjeta, que se le mostrará al usuario mediante una ventana de carga.

3.4. Requisitos no funcionales

En este apartado vamos a exponer los requisitos no funcionales o atributos de calidad que deberá cumplir nuestro producto, para ello utilizaremos el modelo de calidad del producto software definido por la ISO/IEC 25010

- **Adecuación funcional**

  El producto proporcionará todas las funciones para simular el correcto funcionamiento del proceso virtual cuando el alumno proporciona un programa de control adecuado.

- **Eficiencia de desempeño**

  Los tiempos de comunicación con la tarjeta virtual deben ser instantáneos, además debe permitir a la tarjeta utilizar parte de los recursos del sistema para su ejecución.

- **Compatibilidad**

  El intercambio de datos entre el proceso virtual y la tarjeta virtual debe ser estable y completo, sin errores ni estados bloqueantes.

- **Fiabilidad**

  No pueden producirse errores de cálculos proporcionando a los resultados un número infinito de decimales. Además, debe estar disponible en todo momento.

- **Mantenibilidad**

  Debe permitir cambiar datos relativos a los elementos del proceso, como por ejemplo el peso de las cajas, sin alterar la simulación final.

- **Portabilidad**

  La aplicación se podrá incorporar y utilizar en cualquier ordenador de forma rápida y efectiva.

4. Diseño de la aplicación

A continuación, vamos a centrarnos en la parte de diseño de la aplicación. Este apartado lo hemos dividido en cuatro apartados. En el primero de ellos se explicarán los mecanismos de intercomunicación haciendo referencia al trabajo de Miguel Carro.

En el segundo, se comentará el conexionado de entradas y salidas y en el último, los elementos y el control del proceso.

4.1. Mecanismos de intercomunicación

Los mecanismos de intercomunicación que utilizó Miguel Carro en su proyecto estaban basados en *pipes* del sistema operativo, es decir, en *pipes* de Windows, y tanto la tarjeta virtual como la librería, los utilizan para comunicarse entre ellas mismas y con el proceso. Debido a que en nuestro proyecto no se pretende cambiar ni la tarjeta virtual, ni la biblioteca de la tarjeta se ha mantenido este tipo de comunicación.

A continuación, detallaremos las tramas utilizadas para la transmisión de datos. Esta estructura es la utilizada por Miguel Carro.

[CÓDIGO_OPERACIÓN # DIRECCIÓN o CANAL # VALOR]

La trama está formada por 3 elementos separados por el carácter "#" y tiene un máximo de 20 caracteres en total.

- **Código_operación**: Indica el código de la función de la librería que ha enviado dicha trama. Existen las siguientes opciones:
    - **"DO"**: La trama con este código de operación la recibe el proceso virtual desde la tarjeta virtual y contendrá el estado de las salidas digitales de la tarjeta.
    - **"AO"**: La trama con este código la recibe el proceso virtual desde la tarjeta virtual y contendrá el estado de las salidas analógicas de la tarjeta.
    - **"XX"**: Las tramas con este código de operación las utiliza el proceso virtual para enviar a la tarjeta virtual que debe actualizar en sus registros el valor digital.
    - **"XZ"**: Las tramas con este código de operación las utiliza el proceso virtual para enviar a la tarjeta virtual que debe actualizar en sus registros el valor analógico.

- **Dirección o Canal:** En las operaciones digitales este campo indicará el número de puerto al que se hace referencia pudiendo ser este desde 0 a 16. En las analógicas indicará el canal, existen 16 canales, aunque según el protocolo establecido por la tarjeta en las salidas analógicas solo se utilizará 0 y 1.

- **Valor:** Hace referencia al campo de datos de la trama. Cuando la trama contiene un valor digital este campo contendrá un número entero, los bits del cual representan cada una de las salidas o entradas digitales. En cuanto a las tramas

que contienen valores analógicos, el campo “Valor” contendrá un número en formato decimal. En todos los casos el valor tendrá un tamaño de 4 bytes.

Para decodificar este tipo de tramas solo hay que tener en cuenta que el carácter separador determina cada elemento y sus datos.

Seguidamente, se especificarán los elementos que forman parte del protocolo de comunicación en el proceso.

- **Tubotest**
  Este tubo será el medio de comunicación entre la tarjeta virtual y el proceso virtual y se encargará de recibir las tramas identificadas como operaciones de salida. Se utilizará de modo unidireccional, de la tarjeta virtual al proceso virtual.
- **Tubotest2**
  Este tubo será el medio de comunicación entre el proceso virtual y el de la tarjeta virtual, y se encargará de enviar los datos actualizados a la tarjeta mediante tramas identificadas con esta operación.
- **EventoEnviar**
  Este evento se utiliza por motivos de sincronización y lo utilizará el proceso virtual para notificar a la tarjeta que se ha enviado una actualización de los datos a través del tubo “tubotest2”.

La importancia de tener dos tubos, uno para recibir los datos de la tarjeta y otro para su actualización, es principalmente para evitar que la aplicación quede bloqueada por un exceso en el tamaño máximo del buffer debido a que las llamadas no están perfectamente sincronizadas. Este problema no existe en la comunicación entre la tarjeta y la biblioteca ya que éstas están completamente sincronizadas.

Además, como he mencionado anteriormente la tarjeta utiliza *pipes* de Windows para la comunicación. Por lo que existe un gran problema al comunicar la aplicación Java con protocolos de comunicación de Windows. No obstante, en el apartado de implementación explicaremos como se ha resuelto.

El esquema actual queda como se muestra en la Figura 2:

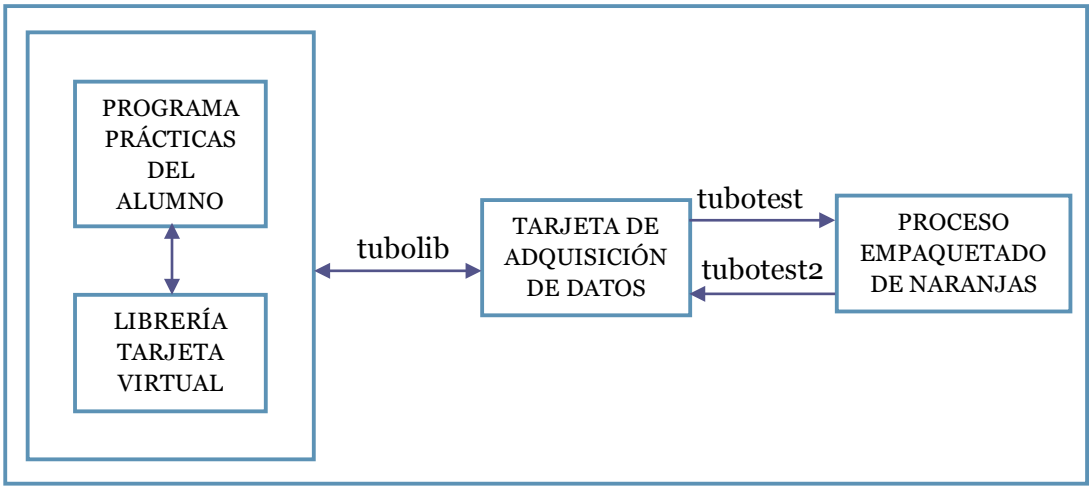


Figura 2: Esquema de comunicación de procesos

4.2. Tablas de entradas y salidas

Al igual que en un proceso físico real, donde cada sensor y cada actuador deberá conectarse a un canal de entrada o salida de la tarjeta de adquisición, será necesario establecer los canales en los que se conectan los diferentes sensores y actuadores del proceso virtual. Esta información es necesaria por dos razones: Una, para que el proceso virtual reaccione correctamente a los cambios de las salidas de la tarjeta virtual y envíe adecuadamente la información de sus sensores; y dos, para que el alumno tenga la información necesaria para implementar su algoritmo de control, leyendo el estado de los sensores a través de los canales correctos y actuando sobre las salidas apropiadas.

Todos estos datos se pueden apreciar en la Tabla 1, 2, 3 y 4 mostradas a continuación.

Entradas digitales		
Bit	Descripción	Estado
3	Caja sobre la báscula	1 - Si / 0 - No
2	Tolva abierta	1 - Si / 0 - No

Tabla 1: Entradas digitales

Salidas digitales		
Bit	Descripción	Estado
6	Abrir tolva	1 – Si / 0 – No
3	Empujar caja	1 – No / 0 – Si

Tabla 2: Salidas digitales

Entradas analógicas		
Canal	Descripción	Rango
3	Peso báscula	-5 v .. 5 v

Tabla 3: Entradas analógicas

Salidas analógicas		
Canal	Descripción	Rango
1	Carga tolva	0 v .. 5 v

Tabla 4: Salidas analógicas

4.3. Elementos del proceso

La aplicación que vamos a diseñar representa un proceso industrial. El proceso en cuestión consiste en la simulación gráfica del empaquetado de naranjas y está compuesto principalmente por cinco elementos que formaran todo el conjunto: las cajas, la báscula, la tolva de llenado, el sensor de presencia y las cintas de transporte. Estos elementos simulan los sensores y actuadores además de simular gráficamente el comportamiento dependiendo de los datos que se les proporcione desde la tarjeta virtual.

A continuación, explicaremos cada uno de estos elementos y al mismo tiempo, proporcionaremos los cálculos específicos de los valores de cada elemento.

- **Las cajas y su transporte.**

Existen tres tipos de cajas: de madera, de plástico y de cartón. Cada una de ellas tiene un peso en kilogramos determinado y diferentes gráficos para mostrar la cantidad de naranjas que contiene. Además, las cajas se desplazan sobre unas cintas que las transportan hasta la posición estimada.

A continuación, en la Tabla 5 podemos observar los diferentes tipos de caja y su peso en kg además del gráfico de cada una de ellas.

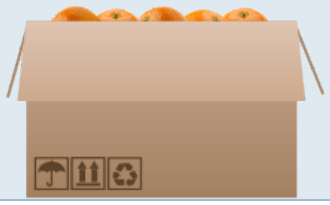
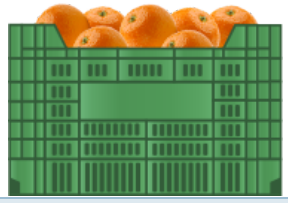
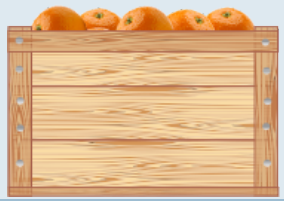
Tipo de caja	Peso caja vacía (kg)	Gráfico caja
Caja cartón	1 kg	
Caja plástico	2 kg	
Caja madera	3,5 kg	

Tabla 5: Tipos de cajas

- **La báscula y el sensor de presencia.**

La báscula contendrá en la parte inferior un monitor que mostrará el peso situado encima de ella. Además, dicha báscula contendrá un sensor para detectar que acaba de situarse una caja encima de ella y de este modo enviar a la tarjeta virtual el peso de la caja vacía ya que aún no se ha procedido a su llenado.

Como se ha podido observar en la Tabla 3 el peso de la báscula se envía a la entrada analógica de la tarjeta virtual y su rango es de -5 a 5 voltios. Por tanto, al peso calculado en la báscula en kg se le deberá aplicar una conversión para convertirlo en voltios. Esta conversión se muestra en la siguiente recta de sensibilidad:

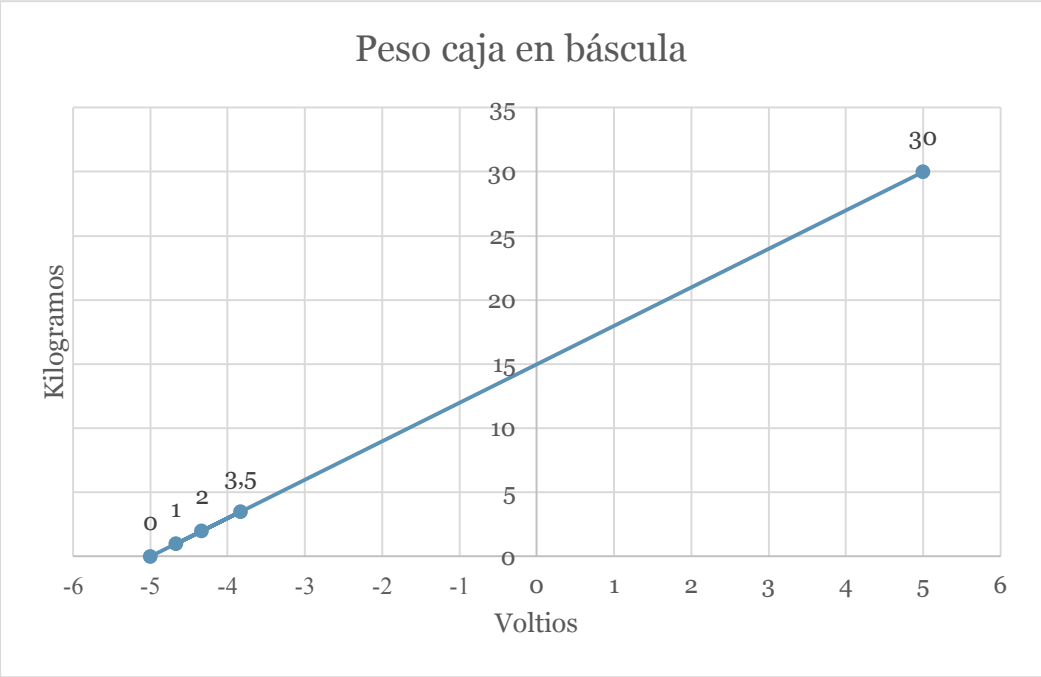


Figura 3: Recta de sensibilidad de la báscula.

- **La tolva de llenado**

La tolva se encarga de llenar las cajas situadas en la báscula. Para ello, posee diferentes sensores y actuadores para poder realizar la simulación. Para que la tolva pueda realizar la simulación gráfica de llenado debe en primer lugar, confirmar que la caja se encuentra en la báscula y, en segundo lugar, debe cargar el peso adecuado para poder soltar la carga de naranjas. El peso de carga se recibe desde la tarjeta en un rango de 0 a 5 voltios como se ha podido observar en la Tabla 4 por lo que se deberá convertir a kg antes de realizar la carga de la tolva. Para ello, se utiliza la siguiente recta de sensibilidad:

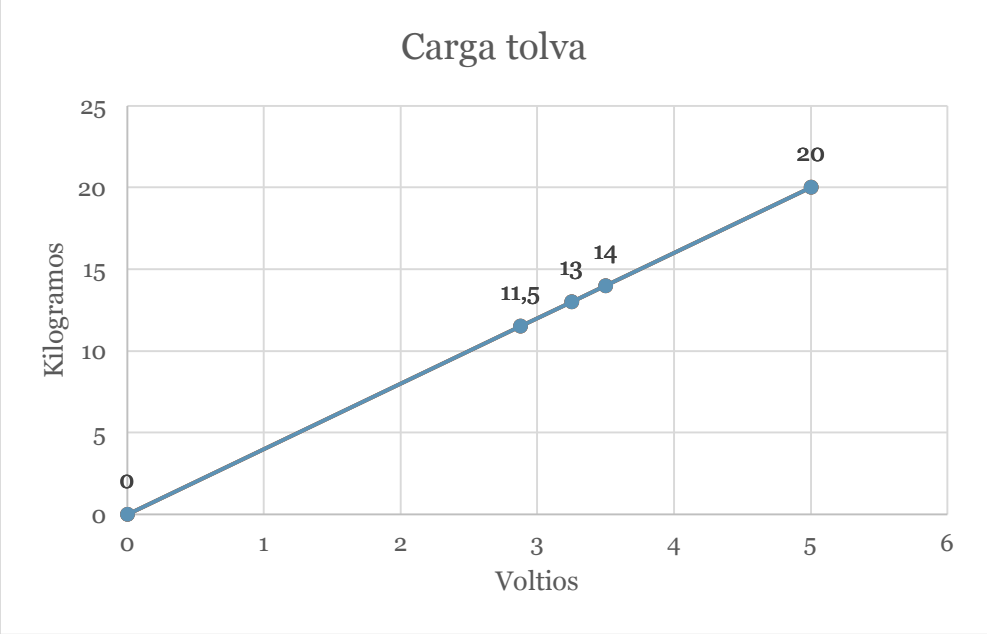


Figura 4: Recta de sensibilidad de la tolva

Además, una vez cargada la tolva, ésta debe simular el actuador de abrir tolva recibiendo desde la salida digital de la tolva el valor correcto. Una vez la tolva esté abierta, ésta simulará gráficamente el llenado de la caja con las naranjas necesarias.

4.4. Control del proceso

Como hemos mencionado anteriormente, este proyecto va unido a otro que simula una tarjeta de adquisición de datos, y que ambos formarían la práctica de una asignatura del grado de ingeniería de tecnologías industriales. El enunciado que se presentaría en dichas prácticas sería el siguiente:

"Una cooperativa agrícola de cítricos tiene un contrato con una empresa de transporte donde se estipula que el peso de los paquetes debe ser de 15 kg cada uno independientemente del contenido. Para ello, se debe indicar a la tolva de llenado el peso de naranjas que debe dejar caer en la caja situada en la báscula. Además, se disponen de tres tipos de caja: de cartón, de plástico y de madera con un peso de 1 kg, 2 kg y 3,5 kg, respectivamente, que se colocan en la cinta de manera aleatoria."

Para resolver el problema, los alumnos mediante la implementación de un programa en C deberán hacer las conversiones, cálculos y transmisiones de datos correctas. En primer lugar, se le proporciona el peso de la báscula que será el correspondiente al peso de la caja situada encima de ella en vacío, y que deberá leer desde la tarjeta virtual. Este valor se encuentra en formato digital, por lo que deberá, basándose en la recta del convertidor analógico a digital de la tarjeta y en la recta de sensibilidad mostrada en el Figura 3, convertir este valor digital a kilogramos.

Una vez tenga el peso de la caja en la báscula debe calcular el peso restante para alcanzar los 15 kg establecidos. Este peso restante lo deberá convertir nuevamente a un valor digital haciendo uso esta vez de la recta de sensibilidad de la Figura 4 y de la recta del convertidor digital a analógico de la tarjeta, y seguidamente, actualizarlo en la tarjeta virtual.

En la Tabla 6 se puede observar la relación de todos los valores con el cálculo correcto.

	Peso caja en báscula		Carga tolva		Total
	Kg	**Voltios**	**Kg**	**Voltios**	**Kg**
Caja cartón	1	-4,66	14	3,5	15
Caja plástico	2	-4,33	13	3,25	15
Caja madera	3,5	-3,83	11,5	2,875	15

Tabla 6: Relación kg-voltios de las cajas

Si los cálculos realizados por el alumno son correctos entonces la simulación del proceso será completa consiguiendo que se visualice gráficamente el llenado de la caja y observándose al final una disposición uniforme de naranjas en ella. Después de mostrar en el monitor de la báscula que el peso es de 15 kg, es decir, correcto, la caja se desplazará en dirección a la derecha en la línea de paquetes para transporte hasta el final de la ventana. Una vez desaparezca del simulador se podrá colocar una nueva caja en la cinta, y así sucesivamente mientras los valores calculados sean correctos.

En el caso que el alumno no realice los cálculos correctamente se pueden producir dos situaciones. La primera, que la carga de la tolva sea inferior a la adecuada, con lo que el peso de la caja no alcanzará los 15 kg. En este caso, la simulación gráfica de llenado será la misma, aunque en la caja no se apreciaran naranjas en su interior.

La segunda, que la carga de la tolva sea excesiva, con lo que el peso de la caja superará los 15 kg. En esta situación se simulará gráficamente un llenado excesivo de naranjas con lo que acabaran saliéndose fuera y en la caja se apreciará gráficamente una pila de naranjas.

En ambos casos, después del llenado, la caja se elevará mediante un brazo hidráulico llevando la caja a la línea de paquetes con fallo.

En la siguiente Figura 5 se puede observar un boceto inicial de la distribución de los elementos en la ventana de la aplicación.

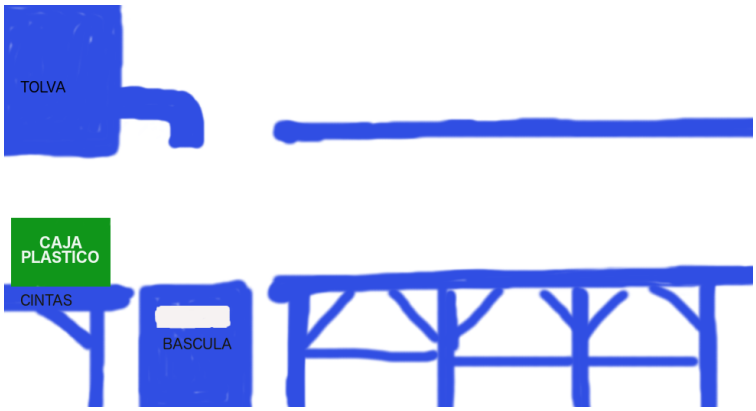


Figura 5: Boceto del proceso virtual

5. Implementación

En este apartado de la memoria vamos a explicar todos los detalles pertinentes de cada parte después de haber realizado la implementación de la aplicación. También comentaremos las tecnologías utilizadas para su realización, además de las librerías utilizadas en cada punto.

Como es habitual, cuando se programa una aplicación en Java hay que tomar como base un diagrama de clases para visualizar mejor la estructura. En la Figura 6 se puede observar el diagrama completo de la aplicación en la que se aprecian todos los elementos del proceso, además de los mecanismos de comunicación y de la interfaz de usuario.

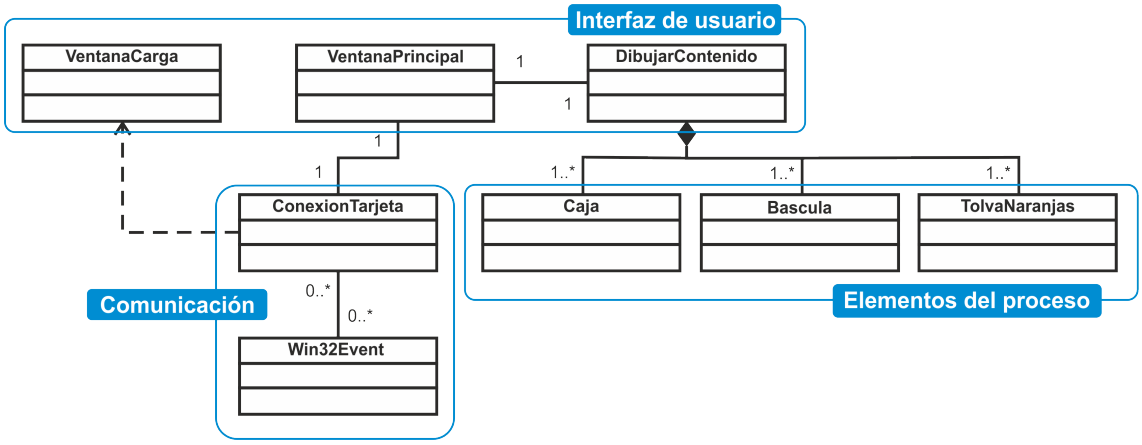


Figura 6: Diagrama de clases

5.1. Mecanismos de intercomunicación

Como ya hemos explicado, nuestra aplicación está compuesta por clases. Por tanto, en este punto entraremos en detalle en las funciones que intervienen o forman parte de la comunicación entre procesos y en las clases en las que se utilizan, pero solo dentro de la aplicación gráfica, ya que las funciones que utiliza la tarjeta y la biblioteca quedan claramente expuestas en el proyecto de Miguel Carro.

Para realizar la comunicación lo primero que se deberá hacer es conectar la tarjeta virtual. Para ello, en la clase principal de la aplicación llamada VentanaPrincipal.java crearemos un proceso con la clase Runtime de Java de la tarjeta y lo ejecutaremos de forma que la tarjeta virtual quedará activa.

A continuación, describiremos la clase “conexiónTarjeta” y todos los métodos que posee. Para poder realizar conexiones con la tarjeta al mismo tiempo que el simulador realiza animaciones gráficas, necesitamos utilizar multitareas para ello, dicha clase debe heredar de la clase Thread. La cabecera quedara de la siguiente manera:

```
public class ConexionTarjeta extends Thread {}
```

Así pues, esta clase se podrá ejecutar como un hilo de procesamiento paralelo al proceso principal. La clase Thread implementa la interface Runnable, por lo que en nuestra clase necesitamos implementar el método run(). Este método es uno de los más importantes de toda la aplicación ya que se encarga de conectarse a la tarjeta y de crear los *pipes* tubotest y tubotest2 para recibir o actualizar las salidas y entradas de la tarjeta virtual. Para ello, hemos utilizado la clase RandomAccesFile disponible en Java.

Seguidamente, mostraremos los métodos que se encargan de recibir y actualizaciones datos tanto analógicos como digitales.

```
public static Double recibirValorAnalogico(String canal) {}
```

Para recibir los datos de las salidas analógicas debemos llamar a la función pasando por parámetros el canal como *String* y ésta nos devolverá un *Double* con el valor exacto.

```
public static int recibirValorDigital() {}
```

En el caso de las salidas digitales simplemente se llama a la función y ésta nos devuelve un *Integer* con el valor.

```
public static void enviarValorAnalogico(double n , int canal){}
```

Pasando a las funciones de actualización tenemos en primer lugar la que se encarga de las entradas analógicas. En esta función le pasaremos el valor deseado y el canal en el cual se actualizará el dato.

```
public static void enviarValorDigital(int n){}
```

Por último, para actualizar las entradas digitales se le pasará a la función simplemente el valor entero que corresponde a los bits que deseamos tener activados o no.

En los métodos cuya función es actualizar las entradas de la tarjeta, es obligatorio debido al diseño de la tarjeta virtual, enviar una señal que informe a la tarjeta de que se le ha enviado un dato. Para ello, hemos creado una nueva clase llamada “Win32Event” que utiliza la librería JNA disponible en Java que permite crear eventos como los existentes en el kernel32 de Windows creando instancias de la misma. En nuestro caso, solo hemos creado un evento para el envío y no para la recepción, ya que nuestro proceso está constantemente leyendo desde la tarjeta.

```
static Win32Event signal = new Win32Event("EventoEnviar");
```

5.2. Elementos del proceso

En este apartado nos centraremos en explicar cómo hemos implementado las clases de los elementos del proceso.

La aplicación contiene tres elementos que interactúan con las entradas y salidas de la tarjeta, de manera que proporcionan al usuario una visión del funcionamiento gracias a las animaciones de dichos elementos. Para estas clases solo mostraremos su cabecera y los atributos que contiene, explicando brevemente para que sirve cada uno.

En primer lugar, tenemos la clase *Caja.java*:

```
public class Caja {
	Image imagen;		//Gráfico que representa la caja.
	int dx = 10;		//Desplazamiento horizontal de la caja por cada repintado.
	int dy = 10;		//Desplazamiento vertical de la caja por cada repintado.
	int id;			//Identificador del tipo de caja.
	int x;			//Posición horizontal de la caja en la ventana
	int y;			//Posición vertical de la caja en la ventana
	int ytop = 364;		//Límite superior para el movimiento vertical de la caja.
	int xdown = 430;	//Límite inferior para el movimiento horizontal de la caja.
	double pesoCaja;	//Kilogramos que pesa cada caja vacía.
	boolean enBascula = false;	//Indica si la caja se encuentra sobre la báscula.
	boolean Fin = false;	//Indica que la caja ha realizado todo el recorrido.
```

Seguidamente, la clase *Bascula.java*:

```
public class Bascula {
	double peso;		//Indica el peso que hay sobre la báscula.
	Image imagen;		//Gráfico que representa la báscula.
	int pos = 750;		//Posición vertical de la báscula.
	int indiceNOK = 4;	//Índice del inicio de la animación con carga errónea.
```

Por último, la clase *TolvaNaranjas.java*:

```
public class TolvaNaranjas {
	boolean vacia;		//Indica si la tolva está llena o vacía.
	double cantidad;	//Cantidad de carga de naranjas en kg.
	int countImages;	//Contador de imágenes utilizadas para la animación.
	int numImages;		//Número de imágenes necesarias para la animación.
	Image imagen;		//Gráfico que representa la báscula.
	int f, faux;		//Identificador para cada imagen de la animación.
	int bucle1;		//Bucle para carga de naranjas correcta.
	int bucle2;		//Bucle para carga de naranjas superior a la establecida.
	int op;			//Indica la opción de animación resultante.
```

5.3. Interfaz de usuario

Para finalizar, en cuanto a implementación, nos centraremos en la parte de interfaz de usuario del proceso virtual. Lo primero que encontramos como interfaz de usuario si observamos la Figura 6 es la clase *VentanaPrincipal.java* que hereda de *JFrame* creando así una ventana para la aplicación. Dentro de esta ventana añadimos un *JPanel* que se implementa aparte en la clase DibujarContenido.java. Esta clase, como su propio nombre indica, se encarga de dibujar todos los elementos del simulador de empaquetado de naranjas con la intención de proporcionarle al usuario una visión lo más fiel posible a la realidad.

Para ello, los gráficos empleados están diseñados expresamente para esta aplicación. Todos ellos son imágenes vectoriales realizadas con el programa Corel Draw X7 con medidas en píxeles y exportadas en formato PNG.

A continuación, se van a describir todos los elementos gráficos que aparecerán en nuestro simulador. En primer lugar, se podrá observar una cinta situada a la izquierda de la aplicación que se encargará de transportar la caja hasta la báscula. Esta cinta se sitúa sobre una estructura colocada en el suelo y de manera recta, sin ninguna pendiente.

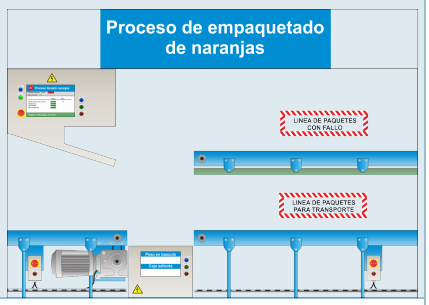
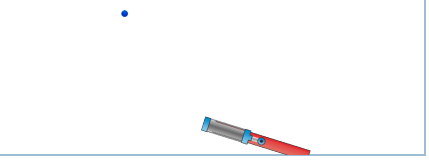
Tendrá una longitud inferior a la mitad de la pantalla, ya que en el centro de ésta se situará la báscula.

La báscula de pesado se sitúa, como hemos mencionado anteriormente, aproximadamente en el centro de la pantalla. Ésta poseerá una parte superior móvil señalizada, además de una pantalla donde se mostrará el peso de la caja situada en la parte mencionada anteriormente y un histórico del peso anterior y la calificación obtenida del mismo. También contendrá diferentes leds lumínicos que indicaran la situación de la báscula. Seguidamente, a su derecha encontraremos dos cintas paralelas, una a la misma altura que ésta y otra, a una altura más elevada. Por tanto, la báscula dispondrá de un sistema que elevará la parte señalada hasta colocarse a la altura exacta de la cinta superior.

Por otro lado, en la parte superior izquierda encontraremos la tolva de llenado de naranjas. Ésta contendrá un monitor con datos informativos y otros leds lumínicos que indicaran diferentes estados de la tolva o alertas. Al activarse el llenado ésta desplazará la rampa de expulsión hasta la posición idónea para dejar caer las naranjas en la caja, una vez situada correctamente se abrirá y dejará caer las naranjas.

Uno de los elementos más significativos del proceso, aunque no esté presente en todo momento en la ventana son las cajas. Se tendrán tres tipos de cajas para los empaquetados: de madera, de plástico y de cartón, cada una con un peso determinado. Estas cajas se desplazarán por la primera cinta hasta llegar a la báscula, una vez en ella, la tolva dejará caer las naranjas en su interior y dependiendo de si la carga ha sido correcta o no, la báscula depositará la caja en la cinta correspondiente.

Para realizar la implementación de la simulación de estos gráficos hemos optado por utilizar la clase Graphics de Java. Esta clase nos permite pintar dentro de un panel todos los componentes que deseemos incluir ya sean imágenes o texto. Además, la característica más importante y por la cual se optó por utilizar esta herramienta, es por el hecho de poder crear un objeto Timer de un determinado intervalo y gracias al cual con la función *actionPerformed()* repintar los componentes infinitamente como nosotros deseemos. En nuestro caso, como se puede observar en la Tabla 7, hemos incorporado los siguientes elementos:

Componente	Código	Imagen
Imagen del fondo base del simulador	`g.drawImage( new ImageIcon( getClass().getResource( "/imagenes/fondo.png")).getImage() ,0,0,this);`	
Imagen de la tolva de llenado de naranjas.	`g.drawImage(tolva.getImagen(), 40, 267, this);`	

Componente	Código	Imagen
Imagen de la caja de naranjas.	`g.drawImage(cajaG.getImagen(), cajaG.getX(), cajaG.getY(), this);`	
Imagen de la báscula de pesado.	`g.drawImage(bascula.getImagen(), 413, bascula.getPos(), this);`	
Texto de la pantalla de la báscula.	`g.drawString(bascula.toString(),515,855);` `g.drawString(history,485,897);`	

Tabla 7: Componentes gráficos

Otro elemento importante que se encuentra dentro de la clase *DibujarContenido.java* aunque no esté integrado en la simulación gráfica es el botón "PONER CAJA EN CINTA" insertado directamente en el *JPanel* como un *JButton* de Java.

A continuación, en la Figura 7, podemos observar el resultado final de la aplicación que muestra el simulador gráfico de empaquetado de naranjas.

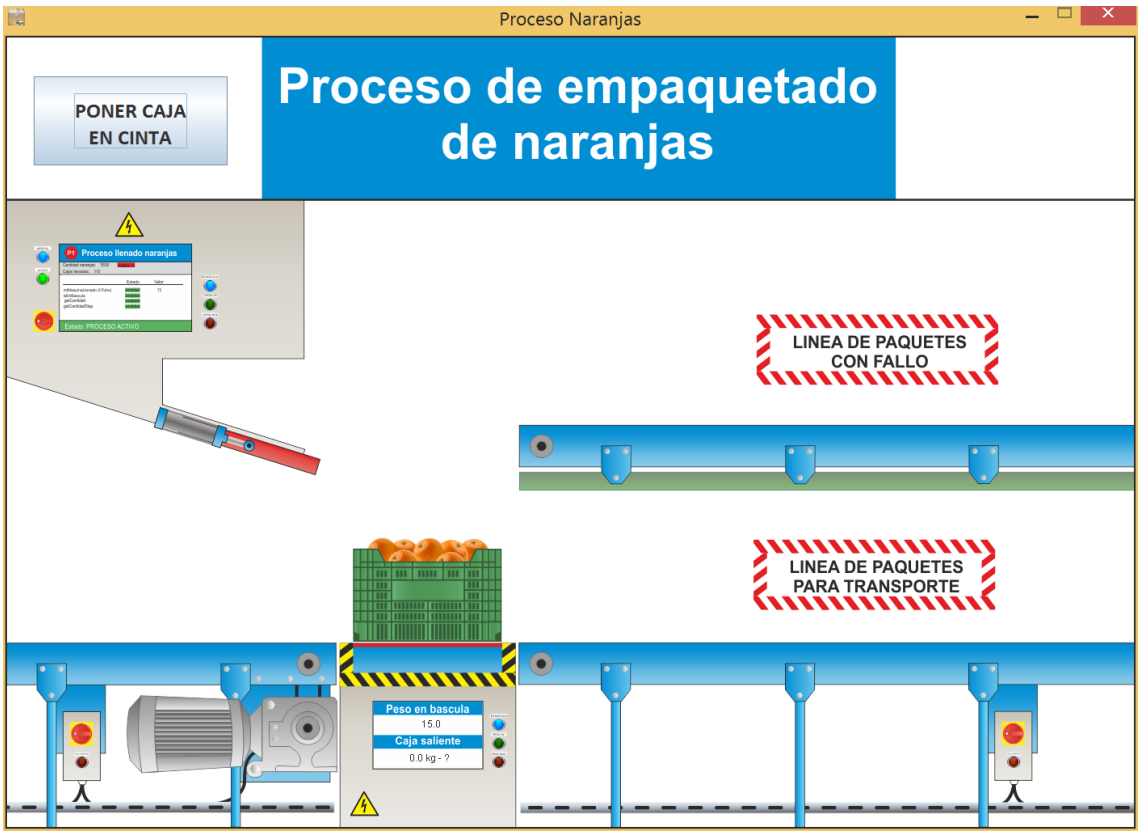


Figura 7: Gráfico completo del simulador

Por otro lado, otra clase que encontramos en la parte de presentación, aunque no tan importante es la de VentanaCarga.java. En esta ventana, como se puede ver en la Figura 8, se muestran los datos de la aplicación además de un mensaje del estado de la ejecución de la tarjeta virtual. De esta manera se le proporciona a la tarjeta virtual el tiempo necesario para realizar la conexión de forma adecuada.

Figura 8: Ventanas de carga de la aplicación

6. Tecnologías utilizadas

A continuación, vamos a exponer todas las tecnologías utilizadas para el desarrollo del trabajo final de grado.

- **Eclipse Neón**

Como sabemos la aplicación está escrita en lenguaje Java y para su implementación hemos utilizado Eclipse. Además, hemos incluido una serie de librerías para su realización que enumeraremos a continuación:

1. Librería JNA (Java Native Access): Java nos permite gracias a JNA la posibilidad de incorporar en nuestras implementaciones java, librerías nativas (DLLs) desarrolladas completamente en C.
2. Paquete AWT: Este paquete de Java nos proporciona la funcionalidad para crear los gráficos de todo el simulador gracias a su clase Graphics.
3. Paquete IO: Este paquete de Java proporciona al sistema entradas y salidas a flujos de datos y acceso al sistema de ficheros. Una de las funciones más importantes que nos proporciona este paquete para nuestra aplicación es la clase RandomAccessFile que nos permite conectarnos a los *pipes* de Windows.

- **CorelDraw**

Para la realización de los gráficos hemos utilizado la versión de prueba gratuita de CorelDraw Graphics Suite X7. Se trata de un programa de diseño gráfico de imágenes vectoriales.

- **Borland C++ compiler**

Se trata del compilador de C utilizado para la realización del programa del alumno. Es un compilador rápido y de alta calidad, muy bueno para utilizarlo con software de Windows en aplicaciones de consola.

7. Pruebas del sistema

En este apartado vamos a analizar las pruebas realizadas para evaluar el funcionamiento gráfico del proceso. Antes de pasar a explicar las pruebas, en la Figura 9 se muestra un pequeño diagrama de flujo de los pasos que realiza la caja.

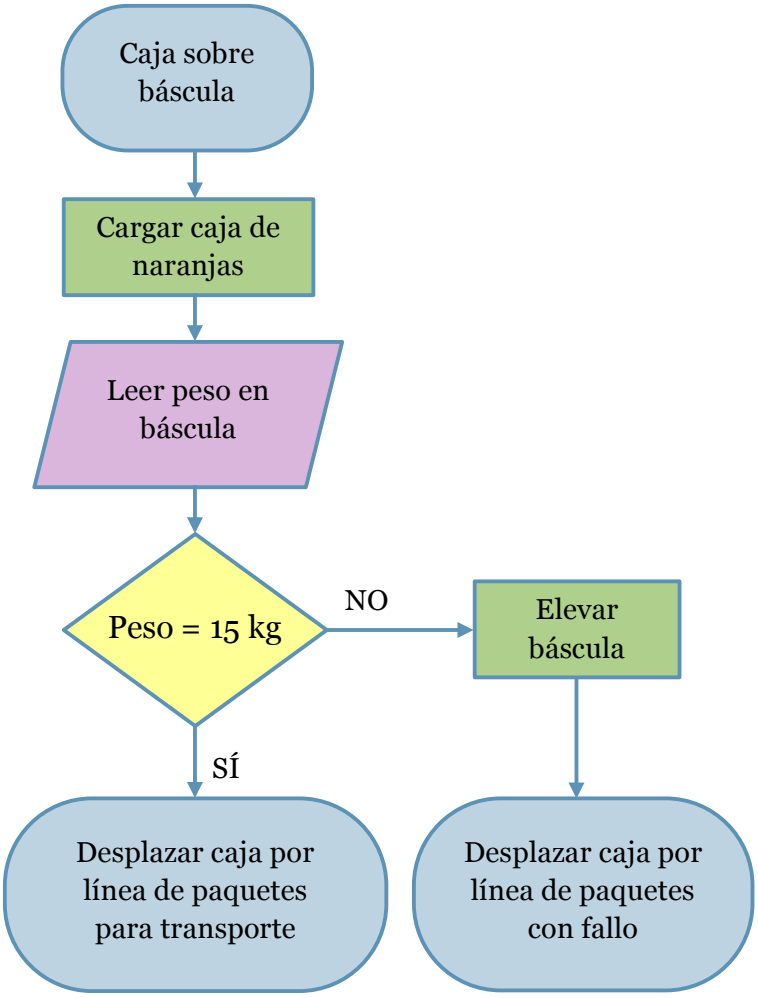


Figura 9: Diagrama de flujo para cajas

Para ello, hemos realizado la implementación de tres programas de prácticas del alumno. En primer lugar, hemos realizado pruebas con un programa de prácticas completamente correcto, donde se ha podido comprobar el correcto funcionamiento de la aplicación. Realizando una simulación de llenado de la caja adecuado y desplazando está por la línea de paquetes para transporte como se puede observar en la Figura 10. Para confirmar completamente el correcto funcionamiento, hemos realizado la simulación un número determinado de veces para poder comprobar todos los tipos de cajas.

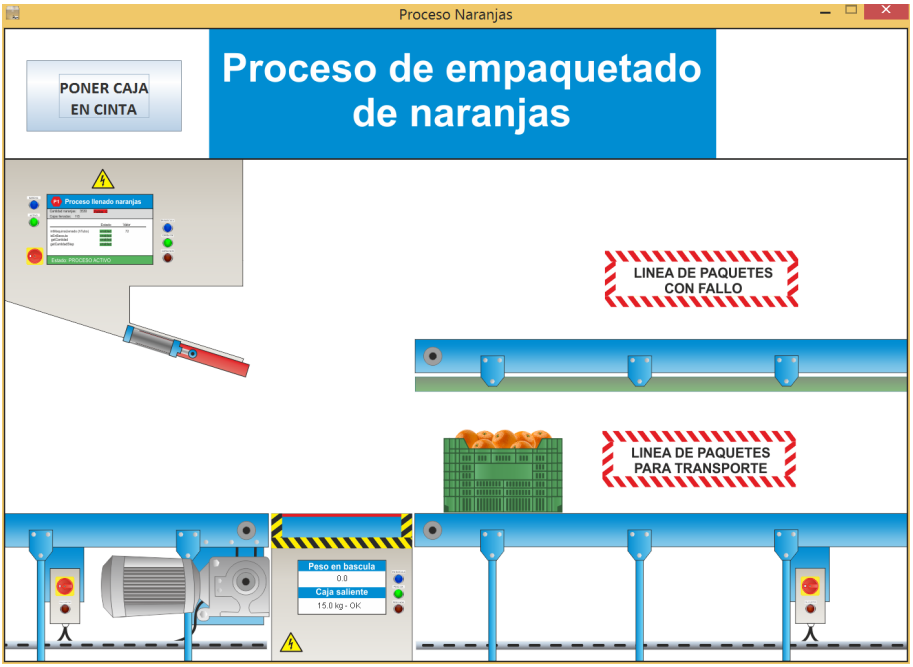


Figura 10: Gráfico de caja con carga correcta

Seguidamente, hemos probado dos nuevos programas de prácticas erróneos. Uno en el que el peso de carga de la tolva es inferior al correcto, por lo que se simulará un llenado escaso de naranjas dejando la caja vacía y el desplazamiento de está por la línea de paquetes con fallo. Para llegar a dicha línea la báscula se elevará con un brazo hidráulico. En la siguiente Figura 11 se puede observar el resultado.

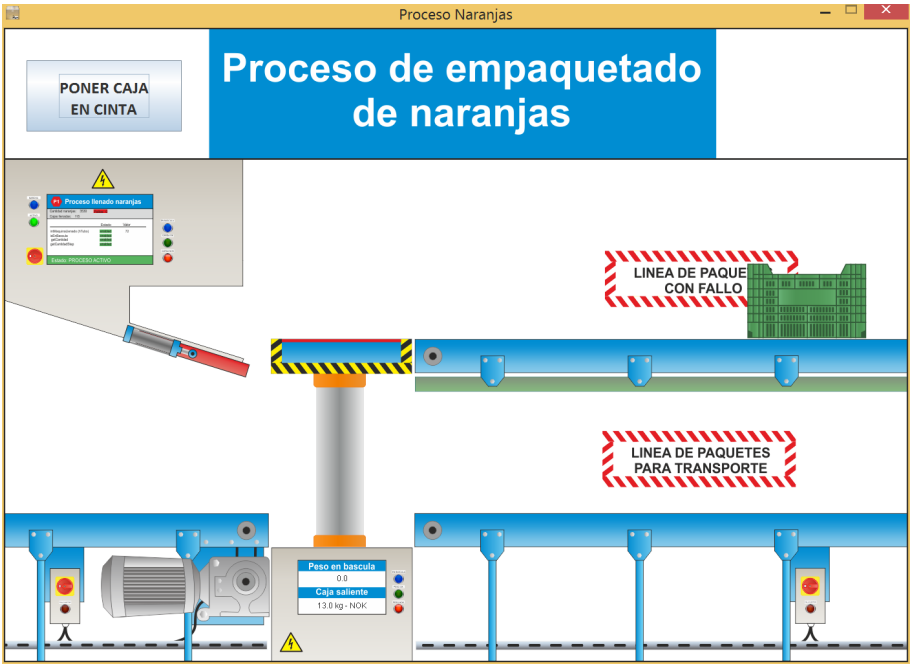


Figura 11: Gráfico de caja con carga insuficiente

Por último, hemos realizado pruebas con un programa en el que el peso de carga es excesivo de manera que al llenar la caja las naranjas acaban desbordándose dejando la caja con una pila de éstas. Finalmente, la báscula eleva la caja hasta dejarla en la línea de paquetes con fallo. En la Figura 12 se puede observar más detalladamente.

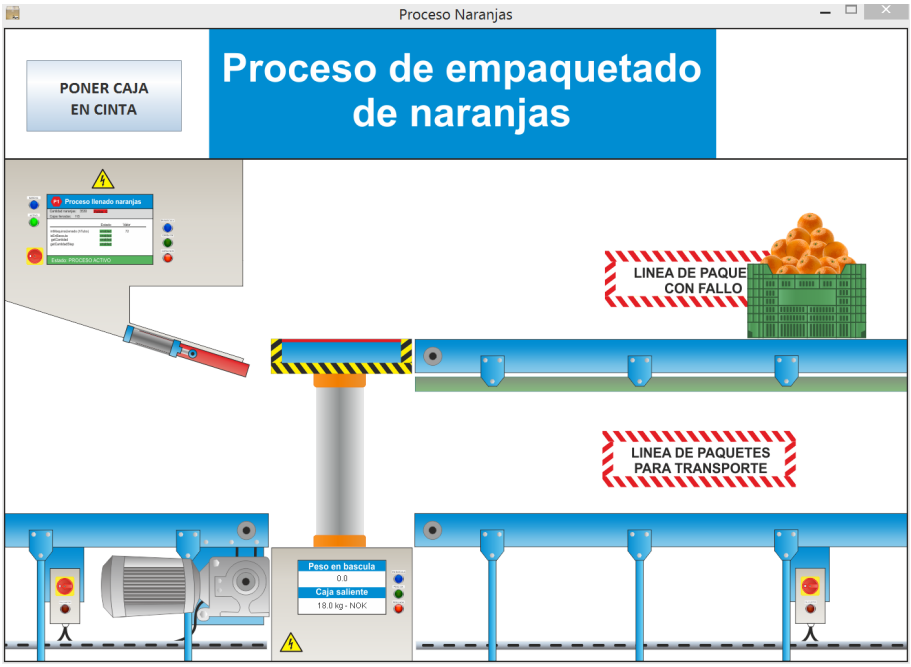


Figura 12: Gráfico de caja con carga excesiva

En ambos casos erróneos la caja se queda a medio camino de la línea de paquetes con fallo bloqueando así el programa, avisando al alumno de que su programa tiene algún error.

8. Futuras implementaciones

Considero que el trabajo realizado y comentado en esta memoria puede mejorarse en el futuro en los siguientes aspectos. El primero, y el más importante, sería el de incorporar en la aplicación un compilador para el programa del alumno. Para ello, se insertaría en la ventana principal una barra de herramientas que nos daría acceso al compilador. Este sería un editor de texto en el que se le permitiría al alumno, sin necesidad de salir de la aplicación; escribir, compilar y ejecutar su programa de prácticas.

En segundo lugar y último, considero que se podría incorporar en la aplicación un *JTextArea* con la intención de que este realizara la función de consola y mostrara los estados del proceso en cada instante de tiempo mientras va transcurriendo el proceso virtual de empaquetado de naranjas. Por ejemplo, cuando la caja de madera estuviera en la báscula podría salir la siguiente línea de texto en la consola: “17:22:10 CajaMadera en báscula” y así, conseguiríamos mostrar al usuario todos los datos de los diferentes estados realizados por el proceso.

9. Conclusión

Para finalizar el trabajo final de grado explicaremos nuestras conclusiones tanto técnicas como personales.

9.1. Conclusiones técnicas

En este proyecto uno de los retos más importantes que se nos planteo fue el de comunicar un proceso en Java, el simulador de empaquetado, con un proceso en C, la tarjeta virtual.

Al principio, al ver las dificultades que comportaba la comunicación entre diferentes lenguajes además de utilizar protocolos específicos de un sistema operativo, se planteó la posibilidad de crear un programa "puente" en C que conectara con los dos procesos y se comunicara con la aplicación java a través de "sockets" y con la tarjeta mediante *pipes*. Pero finalmente, después de muchas pruebas y de horas de investigación, conseguimos alcanzar nuestro objetivo y realizar una conexión entre procesos perfectamente estable.

Gracias a este trabajo he podido descubrir la gran variedad de librerías que nos ofrece Java y hasta donde se puede llegar si se profundiza en cada una de ellas.

9.2. Conclusiones personales

Durante la realización de este trabajo de final de grado me he sentido muy satisfecho.

Al principio todo era un reto que poco a poco he ido consiguiendo alcanzar todos los objetivos que me había propuesto y en el tiempo estimado. Por este motivo, considero que realizar este trabajo es una forma de ponerme a prueba y ver hasta dónde podía llegar.

Después de conseguir este trabajo me considero una persona capaz de realizar y conseguir cualquier reto en lo que se refiere al campo de la informática.

10. Bibliografía

Carro, M. (2007). *Simulador de tarjeta de adquisición de datos "NuDAQ/NulPC 9112 Series*. (Proyecto final de carrera). València: Universidad Politécnica de València.

Martí, A., Campelo, J.C., Serrano, J.J, Alonso, M., Coll, S. (2011). Practical student teaching through integrated true, virtual remote laboratorios. En A. Verbraeck, M. Helfert, J. Codeiro, B. Shishkov (Eds.) *Proceedings of the 3rd International Conference on Computer Supported Education* (vol. 1, pp. 382-385). Noordwijkerhout, Netherlands: SciTePress.

Martí, A., Ors, R., Pérez, A. (2001). *Laboratorio de informatización industrial*. València: Universidad Politécnica de Valencia.

Martí, A., Campelo, J.C., Ors, R. (2001). Simulador de tarjeta de adquisición de datos. En *IX Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas* (CUIEET) (pp.1751-1759). Vigo: Gráficas de Vigo.

PCI-9112 - Data Acquisition - General-Purpose DAQ – ADLINK. Recuperado de: http://www.adlinktech.com/

Chua Hock-Chuan (2016). Java Programming Tutorial Custom Graphics. *Programing notes*. Recuperado de: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4b_CustomGraphics.html

García E. (2013). Ficheros de Acceso Aleatorio en Java. Clase RandomAccessFile. *Programación Java*. Recuperado de: http://puntocomnoesunlenguaje.blogspot.com.es/2013/06/java-ficheros-acceso-aleatorio.html

Oracle and/or its affiliates (1993-2016). Class RandomAccessFile. *Java Platform SE 7*. Recuperado de: https://docs.oracle.com/javase/7/docs/api/java/io/RandomAccessFile.html

Timothy Wall (2007-2015). Java Native Access (JNA). *JNA API 4.2.1*.. Recuperado de: https://java-native-access.github.io/jna/4.2.1/