Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Interfaz gráfica para la manipulación de robots Scorbot Er IX y V+: Consola ACL y módulo de conexión

Graphic interface for the manipulation of Scorbot Er robots IX and V +: ACL console and connection module

Jorge Manuel Gutiérrez Reyes

D. **Santiago Torres Álvarez**, con N.I.F. 43.369.478-B profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Rafael Arnay del Arco**, con N.I.F. 78.599.591-G profesor Ayudante Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Interfaz gráfica para la manipulación de robots Scorbot Er IX y V+ Consola ACL y módulo de conexión”

ha sido realizada bajo su dirección por D. **Jorge Manuel Gutiérrez Reyes**,con N.I.F. 54.063.662-T.

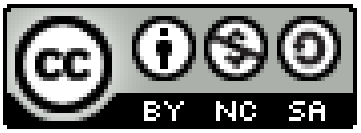
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de septiembre de 2019.

Agradecimientos

Dar las gracias tanto a Santiago como a Rafa por introducirme en esta parte del mundo de la robótica, y que me ha ayudado tanto de forma personal como en mi ámbito laboral.

Agradecer sobretodo a mis padres y mi hermana.

Licencia



Resumen

Desde hace casi 20 años, el Área de Ingeniería de Sistemas y Automática de la Universidad de La Laguna dispone de robots manipuladores tipo PUMA, de los modelos Scorbot Er IX y V+ de Intelitek que se encuentran en el laboratorio de Robótica.

Debido al paso del tiempo los equipos se van actualizando, pero los programas de control de los brazos robóticos, proporcionados por el fabricante, se quedan obsoletos.

A causa de esto, el objetivo de este trabajo ha sido desarrollar un nueva terminal de control y un módulo que permita la conexión entre la terminal y la nueva interfaz con el controlador de los brazos robóticos, y que permita la manipulación de estos.

Con ello, podemos actualizar su software de control y poder ejecutar la aplicación en equipos modernos. Además, se pretende introducir mejoras que no se han implementado antes y hacer más amena la interacción con el robot, facilitando el uso a los alumnos.

Palabras clave: Robots, Consola, Manipulación

Abstract

For almost 20 years, the Systems and Automation Engineering Area of the Universidad de La Laguna has had PUMA-type manipulator robots of the Intelitek Scorbot ER IX and V models in the Robotics labs.

With the pass of the time, the equipment has been updated, but the control programs for the robotic arms, provided by the manufacturer, have become obsolete.

Due to this, the objective of this work is to develop a new control terminal and a module that allows the connection between the terminal and the new interface with the controller of the robotic arms, enabling the manipulation of the robots.

With this, we can update the control software and run the designded application on modern equipment. In addition, the aim is to introduce improvements that have not been implemented before, enabling the use for the students and making interaction with the robots more enjoyable.

Keywords: Robotic arms, Terminal, Manipulation

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Descripción

En el laboratorio de Robótica del Departamento de Ingeniería Informática y de Sistemas se encuentran los robots manipuladores Scorbot Er IX y Scorbot Er V+. Desde hace bastante tiempo el control de estos robots se hace a través de terminal, ya que el programa de control que se utilizaba está sin licencias, y su estado es obsoleto para las nuevas versiones del sistema operativo Windows.

1.1.1. Scorbot Er V+

El Scorbot Er V+ es un robot de articulación vertical, con cinco juntas de revolución. Cuando se le agrega la pinza, el robot posee seis grados de libertad. Este diseño permite a la herramienta final ser colocada y orientada arbitrariamente en un gran espacio de trabajo [1].

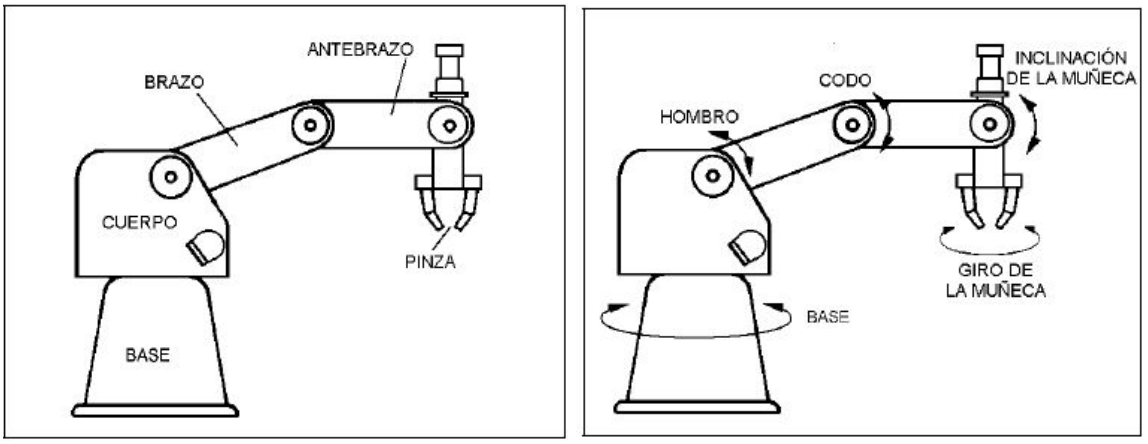


Figura 1.1: Bielas y juntas del Scorbot Er V+

1.1.2. Scorbot Er IX

El Scorbot Er IX es un robot de articulación vertical, con cinco articulaciones de revolución. Con la pinza, el robot tiene cinco grados de libertad. Este diseño, al igual que el modelo anterior, permite que el efector final pueda ser posicionado y orientado de manera arbitraria dentro de un amplio espacio de trabajo [2].

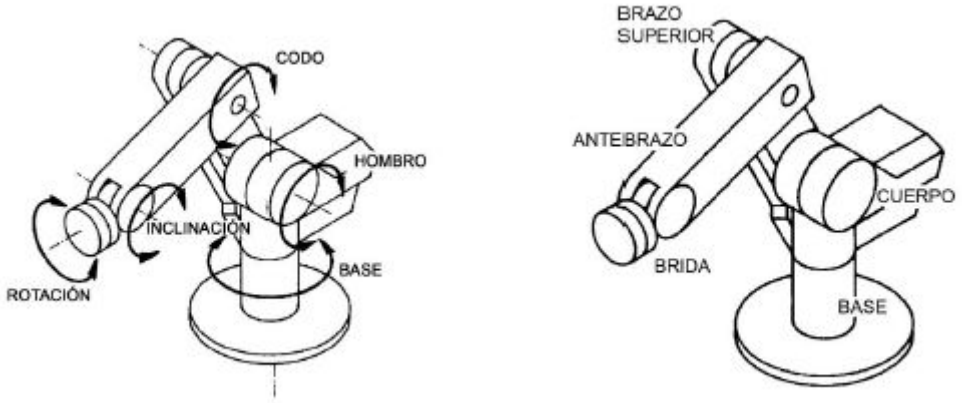


Figura 1.2: Bielas y juntas del Scorbot Er IX

1.1.3. Controlador-A y Controlador-B

Ambos brazos robóticos están conectados a un controlador. En este caso, el Controlador-A está conectado al Scorbot Er V+ y el Controlador-B al Scorbot Er IX. Cada controlador tiene su particularidad, especificaciones de diseño y de comunicación con el robot y dispositivos externos distintas, así como comandos exclusivos.

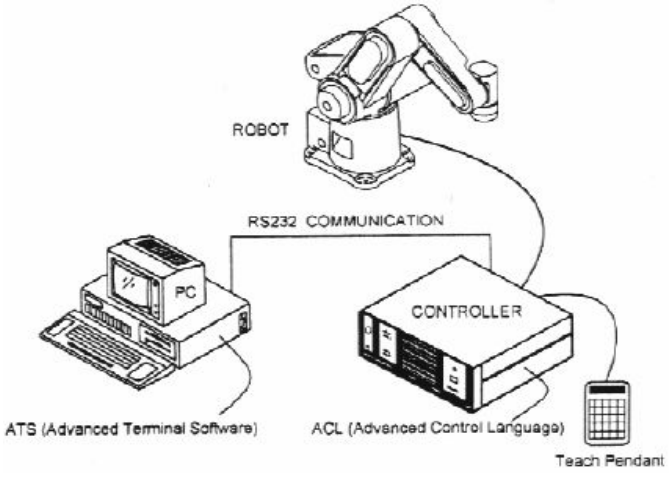


Figura 1.3: Componentes del sistema robótico

1.1.4. Lenguaje ACL

El ACL, Lenguaje de Control Avanzado, es un lenguaje y entorno de programación de robótica avanzado y multitarea.

El ACL está almacenado en memorias EPROM [3] dentro del Controlador-B y del Controlador-A y se puede acceder a él con cualquier PC estándar o terminal, por medio de una línea RS232 [4] de comunicaciones.

Las características del ACL incluyen:

- Ejecución directa de comandos del robot.

- Control de entrada y salida de datos.
- Programación de usuario del robot.
- Ejecución simultánea de programas (total soporte multitarea).
- Ejecución sincronizada de programas.
- Sencilla gestión de ficheros.

1.1.5. Conexión Serial

El controlador y el PC se comunican a través de un canal RS232 con la siguiente configuración:

- 9600 baudios
- 8 bits
- 1 bit de parada
- Sin bit de paridad
- Protocolo XON/XOFF

1.2. Objetivos

El Objetivo de este trabajo Fin de Grado es la creación de un módulo que sirva para gestionar la conexión serial y una consola de comandos ACL para la comunicación visual entre el usuario y el robot. Estos módulos serán añadidos a la interfaz para la manipulación de los robots Scorbot Er IX y Scorbot Er V+ y, con ello, crear una base para que en próximos proyectos se pueda seguir ampliando las funcionalidades de esta interfaz.

Además, conseguimos un objetivo extra. Con la utilización de Unity podremos importar a diferentes sistemas operativos sin la necesidad de muchos cambios en el código. Por lo tanto, conseguimos tener una aplicación que funcione en las versiones más recientes de los sistemas operativos.

También nos marcamos, a parte de que la terminal tenga las mismas funcionalidades que la actual, la inclusión de nuevas características tales como un historial de comandos, autocompletar comandos y un aspecto visual no tan monótono como el que ofrece la terminal de la que disponen los robots, ATS, la cual provee el fabricante para la comunicación serial con el controlador y la edición de programas en el lenguaje ACL.

2. Antecedentes y estado actual

2.1. Contexto

Actualmente, el Departamento de Ingeniería Informática y de Sistemas dispone de unos brazos robóticos con casi 20 años de antigüedad, por lo que sus programas de control son bastantes antiguos. Se está accediendo a los brazos a través de un emulador de terminal que permite el acceso al ACL desde un PC. Esto hace que la interacción con el robot sea tediosa y poco motivadora.

A parte, se utilizaba SCORBASE para windows, que es un software de control de robots que brinda un entorno para la operación y la programación del robot. Sin embargo, este software quedó obsoleto, su instalación se ha vuelto un problema, ya que no existen licencias, y en los equipos nuevos no se puede ejecutar correctamente.

En el presente, solo dos equipos poseen estos dos programas de control y no se pueden ejecutar en paralelo.

2.2. SCORBASE para Windows

El SCORBASE para windows es la aplicación que nos provee el fabricante. Nos permite:

- Controlar en tiempo real los cinco ejes del robot
- Definir el movimiento del robot
- Almacenar y cargar programas y posiciones

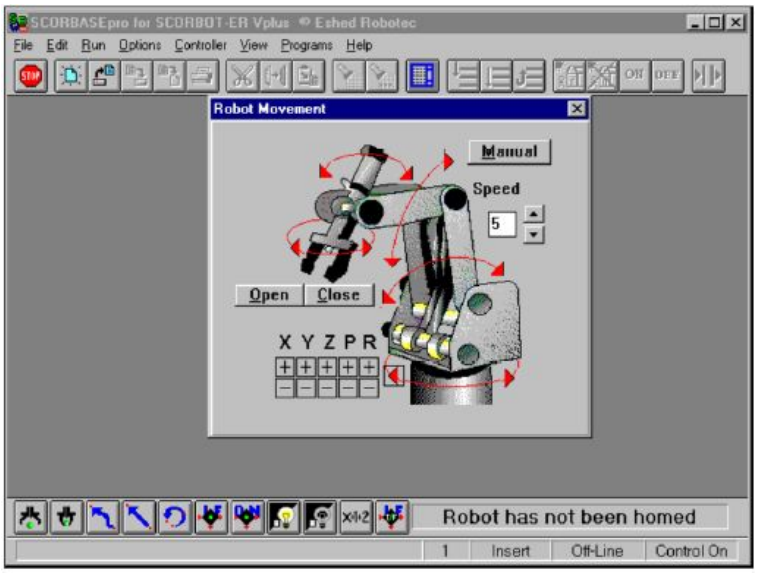


Figura 2.1: Interfaz de SCORBASE para windows

2.3. Emulador de Terminal ACL

Se utiliza el programa DosBox [5] que es un emulador, donde se ejecutará el programa ACL.exe. Este programa es un terminal “tonto”, puesto que no agrega ninguna funcionalidad. Todo lo que recibe desde el controlador, lo imprimirá por pantalla.

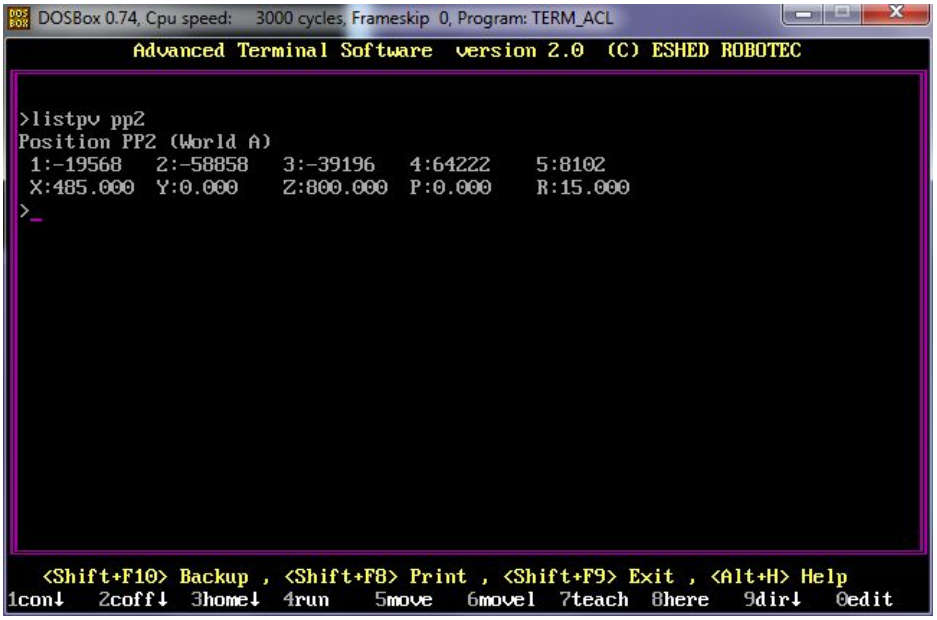


Figura 2.2: Interfaz del emulador DosBox

3. Herramientas y metodología

3.1. Entornos de desarrollo

Durante el proyecto se ha utilizado, como entorno de desarrollo principal, Unity3D [6]. Este es un motor gráfico para el desarrollo de videojuegos, creado por Unity Technologies [7], que nos permite desde un mismo entorno exportar nuestros resultados a cualquier plataforma como Linux [8], Windows [9], Android [10], IOS [11], consolas, etc.

Este entorno de desarrollo tiene una versión gratuita y contiene todas las características necesarias para la creación de interfaces de una manera sencilla. Además posee una extensa documentación, manuales y tutoriales donde nos explican detalladamente los elementos de esta herramienta. Así, también posee una comunidad de usuarios, que a pesar que no es muy extensa, podremos consultar para resolver dudas.

Microsoft Visual Studio [12] es el otro entorno de desarrollo que utilizamos. Creado por Microsoft [13], se puede usar para editar, depurar y compilar código. Tiene diferentes características tales como:

- Limpieza de código,
- Refactorización,
- IntelliSense,
- CodeLens.

3.2. Características de Unity

Microsoft Visual Studio viene integrado en Unity y se utiliza para el desarrollo de los scripts. Estos son escritos en C# [14], lenguaje que Unity soporta, y que fue una de las razones por la cual se eligió Unity como entorno de desarrollo.

- Escena: Contendrá la pantalla principal. Se utiliza para mostrar al simulador y los diferentes desplegables, así como una terminal.
- Gameobject: Son los elementos que se añaden a la escena. Cada Gameobject tiene un propósito específico. Se le pueden añadir más funcionalidades mediante componentes.
- Componentes: Son las piezas funcionales de cada GameObject. Cada componente añade propiedades y controla el comportamientos de los Gameobjects. Hay muchos tipos de componentes integrados diferentes en Unity, pero también se pueden crear componentes utilizando la Unity Scripting API.
- Script: Son escritos en c# o Java [15] y permiten ampliar características que con los componentes integrados no podemos realizar.

4. Desarrollo

Este proyecto se integra con el proyecto de mi compañero Óscar Josué Catari Gutiérrez, creando una interfaz funcional de usuario que será ampliamente utilizada por los estudiantes que hagan sus prácticas con estos dos modelos de robot en el futuro. Con lo expuesto en el capítulo anterior y las necesidades en el proyecto de mi compañero, hemos decidido utilizar Unity para el desarrollo de ambos proyectos.

4.1. Conexión

La gestión de la conexión la realiza el script SerialController.cs, que será el encargado de gestionar la conexión entre el programa y el controlador. Realizará tanto la conexión como desconexión, así como escribir y leer por el puerto. También se han implementado funciones específicas vinculadas a los comandos que se pueden hacer desde la interfaz.

Para la implementación de la conexión, se ha creado un Gameobject llamado Connection. Este GameObject contendrá un componente, que será un script en C#, llamado SerialController. A este script hay que pasarle como parámetros, el Gameobject que contiene a la Terminal.

La clase SerialController nos proporciona los mecanismos para el inicio, gestión y finalización de la conexión al controlador. Las siguientes propiedades de la configuración del puerto serie no se podrán modificar externamente:

- Baudios
- ReadTimeout
- WriteTimeout
- Bit de paridad
- Bit de parada
- Bits de datos

4.1.1. Atributos de SerialController.cs

Estos son los atributos más importantes:

Gameobjects asociados

```
// Gameobjects vinculado a la Terminal
public GameObject T;
private Terminal Terminal;
```

Figura 4.1: Atributos de SerialController.cs

Estos atributos nos permiten conectarnos a la Terminal y enviar la información recibida y las señales necesarias para la correcta visualización en ella.

Puerto Serie

```
// Objeto usado para la comunicacón serial
private SerialPort serialPort;
private Thread portReadingThread;
```

Figura 4.2: Atributos de SerialController.cs

El atributo serialPort nos permite crear una conexión serial y gestionarla. En cuanto a portReadingThread nos ayudará a crear un hilo para la lectura del puerto sin bloquear la interfaz.

4.1.2. Métodos conexión/desconexión

Nos permite abrir o cerrar el puerto. Además se ha añadido la opción de listar los puertos serie que hay disponibles y poder cambiar el nombre del puerto en caso de que ya esté ocupado.

4.1.3. Método escritura

Para realizar la escritura se debe de convertir el string en un array de char ya que el controlador solo acepta char. Además hay que hacer el envío char a char, puesto que el controlador devuelve el carácter que hayamos enviado para así verificar que la escritura fue correcta. Seguidamente, se procede a leer del puerto.

4.1.4. Método lectura

Para realización de la lectura se creará un hilo para que la interfaz no se bloquee. Entrará en un bucle donde su condición de parada será el carácter “>”. Dependiendo del contenido o del último carácter que se haya enviado o del tipo de comando enviado, se realizará una acción u otra. Entre las que se encuentran:

- Introducción de un comando.
- Envío de datos.
- Continuación de la respuesta de un comando.

4.2. Terminal

La única parte visual del proyecto. Contiene las mismas funciones que la terminal que se encuentra en los ordenadores del laboratorio, aunque agregando nuevas funcionalidades y un aspecto visual diferente. Contiene atajos desde teclado.

4.2.1. Atributos

Estos son los atributos más importantes:

Gameobjects asociados

```
//Gameobjects vinculado al Controlador, Conexión y el Panel
public GameObject panel;
public GameObject Control;
private Controller Controller;
```

Figura 4.3: Atributos de CommandTerminal.cs

Con estos atributos seremos capaces de poder acceder al Controlador y al Panel que contendrá a la Terminal. Desde el panel extraemos las medidas y la posición de nuestra Terminal.

Dibujo

```
//Dibujo de la terminal
Rect window;
Rect history;
TextEditor editor_state;
Vector2 scroll_position;
Vector2 scroll_position_history;
GUIStyle verticalScrollbar;
GUIStyle window_style;
GUIStyle label_style;
GUIStyle input_style;
Texture2D background_texture;
Texture2D background_texture_border;
Texture2D input_background_texture;
```

Figura 4.4: Atributos de CommandTerminal.cs

Estos atributos son los necesarios para dibujar la terminal, así como para configurar su aspecto visual.

Log e Historial

```
//Log y historial
public CommandLog Buffer { get; private set; }
public CommandHistory History { get; private set; }
```

Figura 4.5: Atributos de CommandTerminal.cs

Con estos atributos generamos un log, donde almacenaremos la respuestas de los comandos introducidos, y un historial, que almacenará todos los comandos que han sido introducidos. Estos son sus scripts asociados:

- CommandLog.cs: Contiene los métodos necesarios para la gestión del log, como la limpieza del mismo, si hay desbordamiento o añadir nuevos elementos.

- CommandHistory.cs: Tiene casi el mismo comportamiento que CommandLog.cs. Contiene los métodos necesarios para la gestión del historial, como añadir nuevos elementos, recorrer el historial o limpiarlo.

4.2.2. Método de dibujado de la terminal

La terminal consta de dos rectángulos que contendrá a la terminal en sí y un historial de comandos. A su vez, se le creará un marco alrededor. Todos los comandos ejecutados y la información que ellos devuelvan, se almacena en un log, llamado Buffer que es impreso en la pantalla. Además los comandos se almacenan en otro log, llamado History, que nos servirá para listar todos los comandos que hayamos introducido.

Terminal

Desde que añadimos datos a el Buffer, se dibuja en la pantalla, así como el input que nos servirá para introducir los comandos. Cuando el comando introducido necesite la introducción de datos, se modificará la vista para que permita la introducción de datos y se asemeje a la disposición que tenía en la antigua terminal.

Historial

En el historial se visualiza todos los comandos que hayamos escritos, y acceder a ellos de forma rápida haciendo doble click sobre el comando. Automáticamente, en el input de la terminal aparecerá el comando.

4.2.3. Método de captura de teclas

Las capturas de teclas se harán desde la función OnGUI(). Según el tipo de tecla que pulsemos se ejecutará alguna acción u otra:

Enter

Al presionar esta tecla ejecutaremos el comando escrito en la terminal. Se mostrará a continuación la respuesta del controlador.

Flecha Arriba

Cambiará el texto que se encuentra en el prompt. Mostrará el comando que hayamos puesto antes del actual. Si no existe, no cambia el prompt.

Flecha Abajo

Cambiará el texto que se encuentra en el prompt. Mostrará el comando que hayamos puesto después del actual. Si no existe, no cambia el prompt.

Tabulador

Cambiará el texto que se encuentra en el prompt. Mostrará el comando que coincida que el texto que hayamos escrito en el prompt. Para hacer los match buscará en el historial y en la lista de comandos que se han cargado.

4.2.4. Método de envío de comandos

Cuando introducimos un comando, se actualizará la vista añadiendo el nombre del comando, así como su acción. También se añadirá el nombre del comando al historial.

Dependiendo del tipo de comando que se ejecute, hay dos tipos de caso. Un comando que devuelva una respuesta o un comando que necesite la introducción de datos. El primer caso se subdivide en dos, dependiendo de la cantidad de información que devuelva el controlador. Hay que modificar la vista para tratar estos casos y que se adecue al formato que tenía en la anterior terminal.

4.3. Controlador

Es el que enlaza todos los componentes. El software realizado por el compañero se conectará a este controlador, el cual actúa como puente entre las dos partes. Será el encargado de ejecutar los comandos en forma online o offline. También tiene métodos específicos para ejecutar comandos de forma online desde la interfaz.

4.3.1. Atributos

Estos son los atributos más importantes:

Gameobjects asociados

```
//Gameobjects vinculado a la Conexión, Terminal y GameController
public GameObject Terminal;
public GameObject Connection;
public GameObject GameC;
private Terminal Term;
private SerialController Conexion;
private TargetControl TargetControl;
private CommandControl CommandControl;
private IK Robot;
```

Figura 4.6: Atributos de Controller.cs

Cada uno de estos atributos nos permitirán acceder a los métodos y propiedades públicas de la Terminal, la Conexión y GameController. Con ello seremos capaces de comunicarnos con el simulador, la terminal y la conexión serial. Como hemos dicho anteriormente, este componente es el que enlaza con los demás componentes de este proyecto y la parte del simulador.

4.3.2. Método carga de comandos

Este método nos permite cargar información sobre los comandos que podemos ejecutar en el simulador. Esta información se carga desde un XML. Toda esta información la podemos ver desde el comando Help.

4.3.3. Métodos para ejecución de comandos online

Permite ejecutar comandos desde la terminal o la interfaz hacia el controlador del robot. A su vez, cuando se haya terminado el envío del comando, se procede a leer la información enviada por el robot. Aquí hay que diferenciar dos casos.

En un primer caso, la información enviada será plasmada en la terminal, tal y como viene desde el robot.

En el segundo caso entran varios métodos. Son aquellos que se ejecutan desde la interfaz. Por cada serie de comandos, se han hecho unos métodos específicos dependiendo si el tipo de comando necesita inserción de datos, recuperar información del controlador del robot o el resto de comandos.

4.3.4. Método para ejecución de comandos offline

Este método se encarga de ejecutar los comandos disponibles del simulador. Su aspecto visual será idéntico a la de la terminal en modo online, pero sin las limitaciones de tiempos de espera que existen en el mismo.

4.3.5. Integración

En la Figura 4.7 podemos ver las diferentes partes del presente proyecto, así como su integración con el proyecto del compañero y cómo se realiza su flujo de datos.

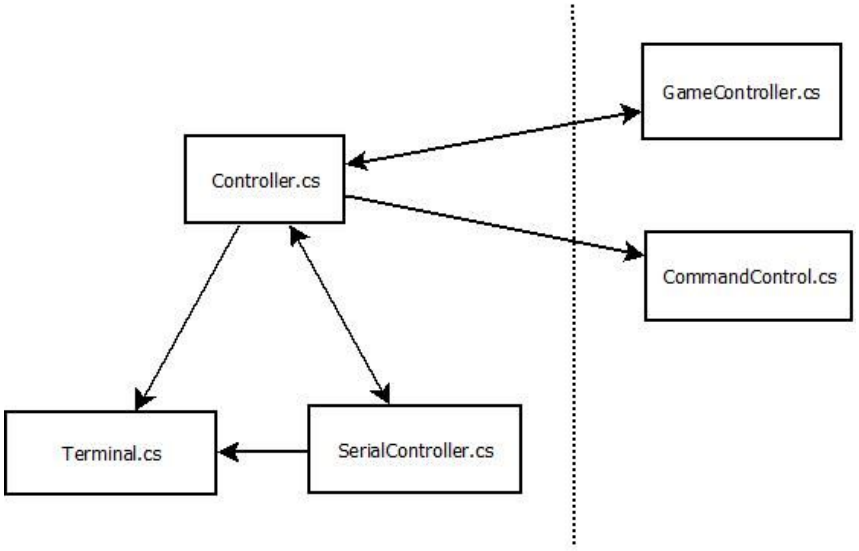


Figura 4.7: Diagrama de bloques

Controller será el encargado de hacer de puente entre los dos proyectos. Aquí cabe destacar que hay una dependencia entre la conexión Serial y la Terminal ya que al utilizar hilos para la lectura del puerto, es necesario una conexión directa para la impresión por pantalla.

Desde la interfaz, se pueden ejecutar los siguientes comandos hacia el robot real:

- Home
- Teach
- Teachr
- Move
- Movec
- Movel
- Shiftc
- Here
- Speed
- Speedl

Para la ejecución de dichos comandos, se provee al compañero de una serie de funciones que están implementadas en el Controller. Estas funciones tienen su peculiaridad, tales como si es necesario la introducción de datos, si devuelven datos o si devuelven si todo ha salido correcto. Hay un comando, que no se ha nombrado, pero que está implementado. Este comando es Listpv. Con este comando y con el comando Teach podremos sincronizar los puntos entre el robot real y el simulador y viceversa. Igualmente, se han implementado las funciones necesarias para la gestión de la conexión serial, como abrir o cerrar la conexión hacia el robot real o modificar el puerto al cual nos conectamos desde la interfaz.

También el compañero implementó las funciones necesarias para poder ejecutar comandos en el robot simulado desde la consola. Estas se encuentran implementadas en CommandControl.cs. Desde Controller, se hacen llamadas a estas funciones, según el comando que se haya escrito en la consola.

4.3.6. Funcionalidades de la consola

A continuación se muestra como funciona la consola y las utilidades que posee. La consola permite la escritura, pero no copiar desde ella.

Cuando se hayan escrito uno o mas comandos, estos se irán almacenando en un historial, y se podrán visualizar en la lista que hay en la derecha.

Figura 4.8: Consola

Si queremos recuperar comandos que ya hayamos escrito, tenemos diferentes formas. Con las flechas arriba y abajo podemos recorrer el historial. En la figura, podemos observar como hemos pulsado la tecla arriba dos veces.

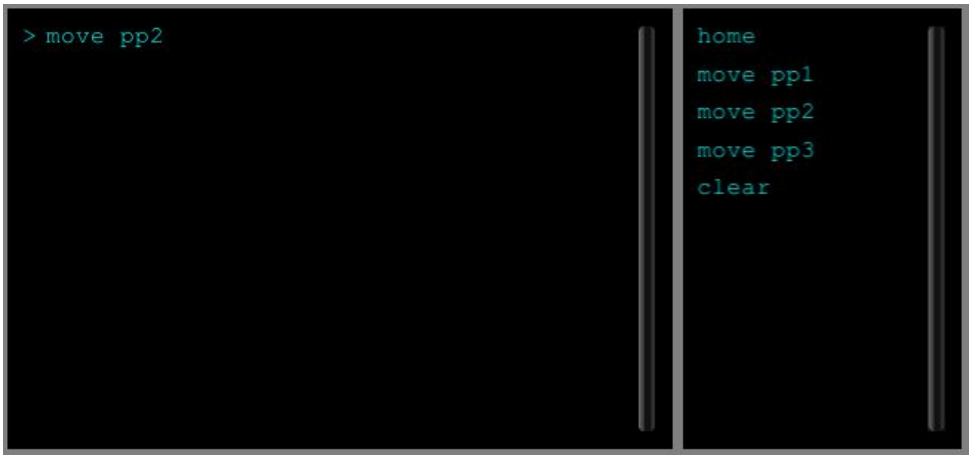


Figura 4.9: Presión tecla arriba

También se puede recuperar comandos, utilizando el tabulador. Para ello, debemos escribir parte del nombre del comando (Figura 4.10) y presionando el tabulador nos recorrerá tanto los comandos disponibles (Figura 4.11) como el historial (Figura 4.12).

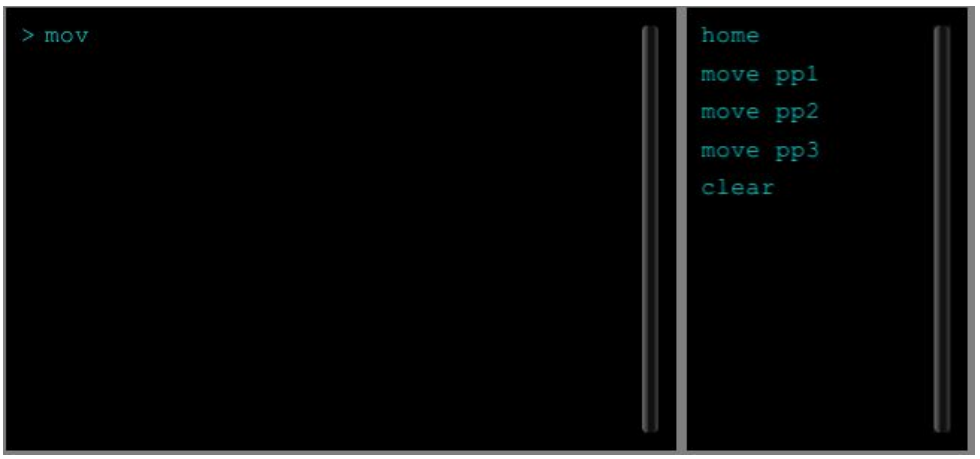


Figura 4.10: Consola con parte del comando

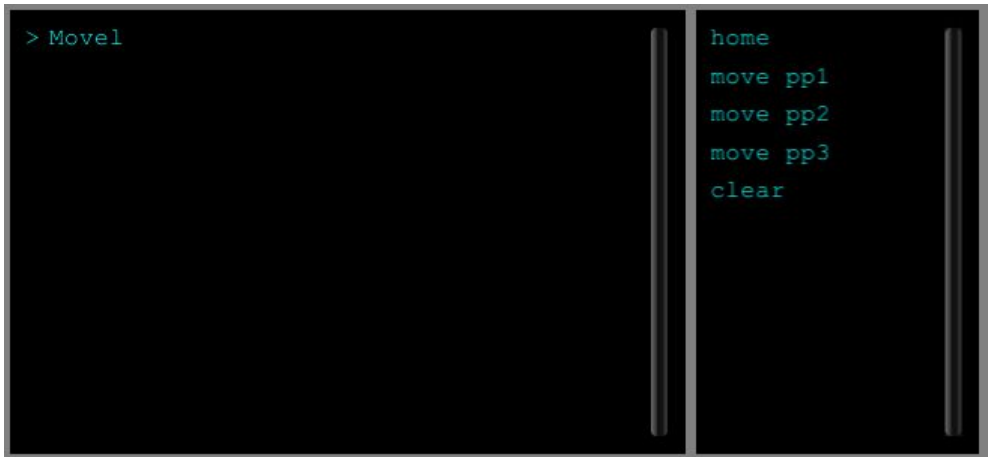


Figura 4.11: Presión tecla tabulador

Figura 4.12: Presión tecla tabulador

Y otra de las formas de recuperar comandos es haciendo doble click en uno de los comandos que se encuentran en el listado que aparece a la derecha de la consola. Automáticamente en el prompt aparecerá el comando. (Figura 4.13)

Figura 4.13: Doble click en el historial

5. Conclusiones y líneas futuras

5.1. Conclusiones

La interfaz que proporciona el fabricante para la conexión, manipulación y ejecución de programas en los robots, tiene un diseño poco amigable y crea poco “feeling” con el usuario. Debido a esto, el interés del alumno decae y en algunos casos propicia a que su rendimiento sea menor, incluso llegando a dejar de lado las actividades que hacen sobre estos robots.

Con este proyecto, mejoramos ese “feeling” con el usuario. Con esta nueva terminal, introducida en la interfaz, se proporciona una visión diferente a lo visto hasta ahora. Aunque se mantienen algunas características, el conjunto de toda la aplicación va a proporcionar a los alumnos nuevas herramientas y sacar el máximo provecho a los robots del laboratorio.

De igual forma, poder ejecutar de forma visual algunos de los comandos más utilizados propicia un mayor interés en el alumno, así como poder realizar estas acciones de una manera más cómoda y sencilla.

La mayor parte de los objetivos que nos planteamos al principio de este proyecto se han cumplido. Aunque ha habido dificultades, que hemos ido resolviendo sobre la marcha, hemos dejado una buena base para que en un futuro esta aplicación vaya creciendo y mejorando.

5.2. Líneas Futuras

En relación a lo antes expuesto, hemos dejado una base para que en próximos TFGs se vayan ampliando y mejorando las diferentes características y funcionalidades que hemos implementado. Además con la utilización de Unity, tenemos un entorno que durará muchos años, y podemos seguir desarrollando este proyecto durante un largo periodo de tiempo.

Una de las próximas tareas que se pueden realizar, es la unificación de las dos partes en las que se compone este proyecto. También, otras de las tareas sería seguir ampliando el conjunto de comandos que se ejecutan en el robot.

En cuanto a la terminal, se le pueden añadir opciones de personalización y herramientas de edición, tales como copiar el texto que imprime en la interfaz.

6. Summary and Conclusions

6.1. Deductions

The current interface provided by the manufacturer for the connection, control and programs execution in the robots, has an unfriendly environment and brings a non-satisfactory feeling to the user. Due to this, the interest of the student declines and, in some cases, leads to a lower efficiency, even leaving aside the activities on these robots.

With this project, we improve that issue about the feeling with the user. Introducing this new terminal in the interface provides a different point of view to that seen so far. Although some features remain, the whole application will provide students with new tools and make the most of the lab robots.

In the same way, being able to execute in a visual way allows a greater interest in the student to use some of the most important commands , as well as being able to carry out these actions in a more comfortable and simple way.

Most of the objectives we considered at the beginning of this project have been reached. Even though there have been difficulties, which we have been resolved on the fly, we have left a good basis for this application to grow and improve in the future.

6.2. Future Lines

In relation to the above, we have left a basis for future final degree projects to expand and upgrade the different features and functionalities we have implemented. Also, with the use of Unity, we have an environment that will remain many years, and the project can be developed over a long period of time.

One of the next tasks that can be carried out is the unification of the two parts that this project is composed. Furthermore, other tasks would be to continue expanding the set of commands that run on the robot.

As to the terminal, we can add customization options and editing tools, such as copying the text the interface prints.

7. Presupuesto

A continuación se muestra un desglose del presupuesto de este proyecto:

Tipos	Descripción	Horas	Precio/Hora	Precio
Herramientas	Unity3D y Visual Studio 2017	-	-	0€
Herramientas	Brazos Robóticos			0€
Desarrollo	Programador	340h	20€/h	6800€
Total		340h		6800€

Tabla 7.1: Resumen presupuesto

Bibliografía

[1] *SCORBOT-ER 9Pro Manual del usuario (Catálogo #200034-ES Rev. B).* (2011). Manchester:Intelitek Inc.

[2] *SCORBOT-ER V Plus Manual del usuario (Catálogo #100265 Rev.A).* (2000). Manchester:Eshed Robotec

[3] Colaboradores de Wikipedia. (2019a, 29 julio). Memoria EPROM - Wikipedia, la enciclopedia libre. Recuperado 23 mayo, 2019, de https://es.wikipedia.org/wiki/Memoria_EPROM

[4] Colaboradores de Wikipedia. (2019b, 24 julio). RS-232 - Wikipedia, la enciclopedia libre. Recuperado 23 mayo, 2019, de https://es.wikipedia.org/wiki/RS-232

[5] Colaboradores de Wikipedia. (2019c, 14 julio). DOSBox - Wikipedia, la enciclopedia libre. Recuperado 21 julio, 2019, de https://es.wikipedia.org/wiki/DOSBox

[6] Unity Technologies. (s.f.-a). Unity - Unity. Recuperado 14 junio, 2019, de https://unity.com/es

[7] Unity Technologies. (s.f.-b). Company - Unity. Recuperado 14 junio, 2019, de https://unity3d.com/es/company

[8] Linux. (2019, 24 julio). Linux.org. Recuperado 14 junio, 2019, de https://www.linux.org/

[9] Microsoft. (s.f.-a). Sistema operativo Windows. Recuperado 14 junio, 2019, de https://www.microsoft.com/es-es/windows

[10] Alphabet. (s.f.). Android. Recuperado 14 junio, 2019, de https://www.android.com/intl/es_es/

[11] Colaboradores de Wikipedia. (2019d, 23 julio). ios. Recuperado 14 junio, 2019, de https://es.wikipedia.org/wiki/IOS

[12] Microsoft. (2019, 19 julio). IDE de Visual Studio, editor de código, Azure DevOps y App Center - Visual Studio. Recuperado 14 junio, 2019, de https://visualstudio.microsoft.com/es/

[13] Microsoft. (s.f.-b). Microsoft - Official Home Page. Recuperado 14 junio, 2019, de https://www.microsoft.com/es-es

[14] Colaboradores de Wikipedia. (2019e, 29 julio). Lenguaje de programación C#. Recuperado 14 junio, 2019, de https://es.wikipedia.org/wiki/C_Sharp

[15] Colaboradores de Wikipedia. (2019f, 1 agosto). Lenguaje de programación JAVA. Recuperado 14 junio, 2019, de https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)