UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA BIOMÉDICA

SIMULADOR 3D EN TIEMPO REAL DE UN BRAZO ROBÓTICO

SERGIO VÁZQUEZ ESPINOSA

2017

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Dpto. de Tecnología Fotónica y Bioingeniería

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA BIOMÉDICA

SIMULADOR 3D EN TIEMPO REAL DE UN BRAZO ROBÓTICO

SERGIO VÁZQUEZ ESPINOSA

2017

1. Resumen

En este trabajo se presenta un simulador virtual en 3D en tiempo real del brazo robótico PhantomX Reactor de 5 grados de libertad. El simulador tiene la finalidad de apoyar el complejo aprendizaje de la ingeniería robótica, así como de controlar de manera más intuitiva y remota un brazo robótico. Para la consecución del proyecto se utilizó como herramientas la interfaz gráfica de Matlab (GUI), el propio Matlab, el modelo en 3D del brazo robótico que fue elaborado mediante el toolbox de realidad virtual de Matlab, *V-Realm Builder* y, por último, se utilizó el Arduino IDE para programar la tarjeta *Arbotix-M robocontroller,* conectada a nuestro brazo robótico físico.

Palabras clave: brazo robot, simulador 3D, Matlab, realidad virtual.

2. Abstract

This project presents a real time 3D virtual simulator of a PhantomX Reactor Robotic Arm with five DOFs (Degree of Freedom). The aim of this simulator is to support the complex learning process of robotic engineering and to remote control the robotic arm in a more intuitive way.

In order to carry out this project, the tools used are MATLAB and its graphical user interface (GUI). The 3D robotic arm model was created by means of MATLAB's virtual reality toolbox, V-Realm Builder. Last but not least, Arduino IDE was used to program the *Arbotix-M robocontroller's* microcontroller, which was connected to our physical robotic arm.

Index terms: robotic arm, 3D simulator, Matlab, Virtual reality

Índice

1. Resumen iii
2. Abstract iv
3. Objetivos del proyecto. 1
4. Motivación 2
5. Estado del Arte 3
6. Introducción 4
   6.1 Entorno de desarrollo. 4
      6.1.1 Software de simulación. 4
      6.1.2 Software de modelado. 4
      6.1.3 Software de diseño de interfaz gráfica. 5
   6.2 Elección de placa. 5
   6.3 Robot PhantomX Reactor 6
   6.4 Cinemática directa. 8
   6.5 Cinemática inversa 10
7. Características principales 13
8. Hardware 14
   8.1 Cable USB 14
   8.2 PhantomX Reactor 14
   8.3 Dynamixel 12A Actuator. 14
9. Software 15
   9.1 Arduino IDE. 15
   9.2 Matlab 15
   9.3 V-Realm Builder 15
   9.4 GUI de Matlab 16
10. Implementación 18
   10.1 V-Realm Builder 18
   10.2 GUI de Matlab 21
   10.3 Matlab 24
   10.4 Arduino IDE 30
      10.4.1 Envío de datos 30
      10.4.2 Recepción de datos 30
11. Conclusiones 33
12. Líneas futuras 34
13. Bibliografía 35

14. ANEXO I. MANUAL DE USUARIO ARDUINO IDE. 37

Índice de Tablas

Tabla 1: Medidas de cada una de las extremidades del PhantomX Reactor 6
Tabla 2: Limitaciones mecánicas de cada una de las articulaciones 6
Tabla 3: Especificaciones del servomotor Dynamixel AX-12A 14
Tabla 4: Descripción botones del GUI Matlab 17

Índice de Figuras

Figura 1. Placa Arbotix-M Robocontroller.. 5
Figura 2. PhantomX Reactor Robot Arm. 6
Figura 3. Representación de los grados de libertad y sistemas de coordenadas locales. s 7
Figura 4. Representación de los grados de libertad con el sistema de referencia seleccionado. 7
Figura 5. Sistemas de referencia locales y par cinemático (2,3). 9
Figura 6. Arquitectura del proyecto 13
Figura 7. Interfaz V-Realm Builder 16
Figura 8. Interfaz de usuario GUI 17
Figura 9. Características del robot 18
Figura 10. Ventana de configuración para la rotación de la base del robot 18
Figura 11. Ventana de configuración del centro de rotación del brazo 19
Figura 12. Visión general del modelo sin terminar 20
Figura 13. Ventana de configuración de la extrusión insertada 20
Figura 14. Modelo 3D del brazo robótico en dos vistas. Frontal y Superior 21
Figura 15. Creación de la interfaz de Matlab 21
Figura 16. Interfaz GUI Matlab sin ejecutar 22
Figura 17. GUI de Matlab ejecutado 23
Figura 18. Esquema de recepción de datos con Matlab 24
Figura 19. Lista de las funciones más importantes para la manipulación del modelo 3D 25
Figura 20. Configuración de la rotación de cada una de las articulaciones de nuestro modelo 3D. 26
Figura 21. Funcionamiento de la lectura y asignación de los estados de los servos 27
Figura 22. Proceso al ejecutar la cinemática inversa 28
Figura 23. Esquema de funcionamiento de recepción de datos en Arbotix Std 31
Figura 24. Interfaz de IDE de Arduino 37
Figura 25. Elección de placa en IDE 38
Figura 26. Selección de puerto COM 39

3. Objetivos del proyecto.

El objetivo general del proyecto es implementar un software que controle el brazo robótico PhantomX Reactor, por medio de una interfaz gráfica intuitiva y que realice el control de manera eficiente. De modo que sea útil para el aprendizaje, futuras investigaciones o implementaciones.

Los objetivos específicos son los siguientes:

- Estudio de diferentes softwares para hacer frente las diferentes partes del proyecto.
- Implementación del modelo 3D del brazo robótico con sus correctas medidas y con el respectivo estudio del movimiento.
- Implementación de diferentes funciones en Matlab, que serán útiles para la consecución de diferentes tareas a llevar a cabo durante el proyecto, como, por ejemplo, lectura de datos, conexión y desconexión del puerto serie, movimiento de servos, etc.…
- Estudio, diseño e implementación de una interfaz gráfica para el manejo del robot de forma remota.
- Implementación de diferentes funciones en el microcontrolador para la lectura de los datos que recibe por el puerto serie, así como la invocación a las diferentes funciones de interés implementados con anterioridad.

4. Motivación

Los alumnos de la titulación de Máster Universitario en Ingeniería Biomédica poseen ciertas carencias en lo relativo a diseño hardware, creación de objetos o entornos interactivos y desarrollo de un sistema de control, debido a las pocas asignaturas dedicadas durante el grado y máster a esta rama, en concreto a la de la bioingeniería. Con este proyecto se ha pretendido ampliar los conocimientos en estos campos y subsanar dichas carencias.

La principal motivación del proyecto es adquirir la capacidad de analizar, evaluar, seleccionar, configurar y vincular diferentes entornos que permitan el desarrollo y la ejecución de la aplicación que se desee.

Además, es interesante extrapolar el proyecto que se desarrolla en este trabajo a diferentes áreas, como la médica o la enseñanza, con el fin de que la curva de aprendizaje tanto del cirujano como del alumno sea la más rápida posible.

5. Estado del Arte

Desde que en 1947 se desarrollara el primer telemanipulador, con el objetivo de manipular elementos radiactivos sin riesgos para el operador, se ha investigado y desarrollado diferentes tipos de robots para aplicaciones específicas, ya sea para el ámbito militar, médico o aplicaciones de seguridad residencial. [1]

De manera paralela al desarrollo de telemanipuladores, aparecieron nuevas compañías desarrolladoras de software de simulación. En la actualidad, no se contempla que haya un robot sin antes haber simulado su comportamiento con un software de simulación virtual y verificado su funcionalidad, ya sea para ingenieros de diseño de productos o investigadores que verifican y validan sus teorías. La simulación se ha convertido en el puente de comunicación entre los equipos de diseño y los consumidores. Los programas de simulación se han convertido en una herramienta fundamental en el diseño y análisis de todo tipo de procesos, más aún, en procesos que involucren sistemas mecánicos en movimiento que implican un profundo análisis físico y matemático [2]. De esta manera, en los últimos años han aparecido numerosos proyectos de investigación relacionados con disciplinas de simulación virtual. Una correcta simulación es necesaria como primer paso para la obtención de herramientas capaces de ser utilizadas para el análisis y diseño de robots. Así, la detección de errores en el prototipado es más sencillo. [3]

Los robots manipuladores en el área médica se llevan diseñando desde hace varias décadas para especialidades como la neurocirugía y la traumatología, especialidades que requieren una intervención muy precisa. Estos robots presentan una mejor visualización del campo operatorio y ofrecen una mayor precisión en las operaciones gracias a los algoritmos implementados en él. [4]

El sistema Da Vinci proporciona al paciente y cirujano la modalidad de un tratamiento más efectivo y menos invasivo, incluso en los procedimientos más complejos. Aunque el periodo de aprendizaje de la técnica sea largo, los costes de los equipos y procedimiento sean más altos y la falta de sensación háptica para el cirujano sea un hecho, cosas que se pueden mejorar en el futuro, los beneficios más destacados del uso de la robótica respecto a la práctica manual de la cirugía están: [6][7]

- Riesgo de infección.
- Escasa perdida sanguínea
- Menor dolor posoperatorio
- Corta estancia hospitalaria

6. Introducción

En este trabajo se detalla la simulación de un brazo robótico de 5 grados de libertad (Phantom X Reactor), proporcionándole al robot el código necesario por medio del microcontrolador *Arbotix Std* para definir su comportamiento mediante las ecuaciones cinemáticas que rigen su comportamiento e implementando un simulador que se encargue de reproducir el movimiento que se llevaría a cabo en el robot físico. Dentro de la interfaz de usuario creada se puede manejar el robot mediante diferentes botones del sistema.

Actualmente, se están implementando muchos proyectos de robótica de control en diferentes ámbitos profesionales, ya sea para mejorar el aprendizaje sobre el manejo de diferentes máquinas (simuladores), o bien, mejorar la eficacia del trabajo hecho manualmente, a través de la implementación de un controlador a un robot manipulador.

Para este trabajo se hace uso de diferentes softwares para la implementación de diferentes módulos en los que se divide el proyecto, de modo que, conforme se realiza el TFM, se va testeando la comunicación entre los diferentes módulos para comprobar el correcto funcionamiento del sistema. Es tan importante la implementación de los diferentes módulos, como la comunicación existente entre ellos para que cada módulo aporte la información pertinente a los demás.

El programa de simulación que se utiliza es Matlab, incluyendo sus herramientas. Una de las herramientas que se utiliza es GUI de Matlab, una herramienta para implementar la interfaz gráfica para el manejo del robot a través del ordenador. La otra herramienta que se utiliza es el V-Realm Builder, que permite crear modelos 3D y conectar el robot real al modelo creado con este mismo software. Junto a estas herramientas citadas, también se utiliza Arduino IDE como software de programación y grabación del microcontrolador.

De manera introductoria se describirá el proceso de selección de los distintos programas utilizados para llevar a cabo la aplicación, al igual que se detallará las características del robot y nociones sobre la cinemática necesaria para entender la comunicación con éste y su manera de actuar. Posteriormente, junto al proceso de diseño de las distintas partes, se explicará más detenidamente los softwares seleccionados.

6.1 Entorno de desarrollo.

6.1.1 Software de simulación.

En este aspecto, no hay ninguna duda que el software a utilizar iba a ser Matlab, aparte de estar familiarizados con él, es una potentísima herramienta de trabajo, sobre todo en análisis matemático y análisis de datos, con muchas aplicaciones y herramientas que nos podrían ayudar en la realización del proyecto. [8]

6.1.2 Software de modelado.

La herramienta Virtual-Realm Builder no es el software más potente para realizar modelados 3D, pero gracias al módulo Simulink 3D Animation permite interactuar con diferentes escenas virtuales mediante Matlab. Por una parte, su sencillez de uso y su compatibilidad con Matlab es

uno de los grandes factores a tener en cuenta a la hora de elegir este software de modelado, pero, por otra parte, no tiene tanta resolución ni es tan completo como puede ser SolidWorks que es una herramienta mucho más potente, pero que a la hora de trabajar con Matlab presenta más carencias como, por ejemplo, la compatibilidad.

6.1.3 Software de diseño de interfaz gráfica.

Siguiendo la misma tónica que en la elección del software de modelado, GUI de Matlab, no es la herramienta más potente para la creación de interfaces gráficas como puede ser JAVA 3D o LabView, pero la familiaridad con Matlab y su compatibilidad fueron factores claves para la elección de este software. GUI de Matlab permite un control sencillo de las aplicaciones de software, lo cual elimina la necesidad de aprender otra herramienta de desarrollo a parte de Matlab y escribir comandos a fin de ejecutar una aplicación.

6.2 Elección de la tarjeta.

La tarjeta que se utilizó para monitorizar y controlar los servos del robot fue la Arbotix-M Robocontroller (*Figura 1)*. La principal diferencia que se distingue en esta placa es que utiliza dos puertos USART (pueden recibir y transmitir datos simultáneamente), mientras que placas como la Arduino tienen solamente uno. [9]

Para la realización de este proyecto se necesitan dos puertos USART, uno de ellos para la comunicación de los servos con la tarjeta y el otro para la comunicación de la tarjeta con el ordenador.

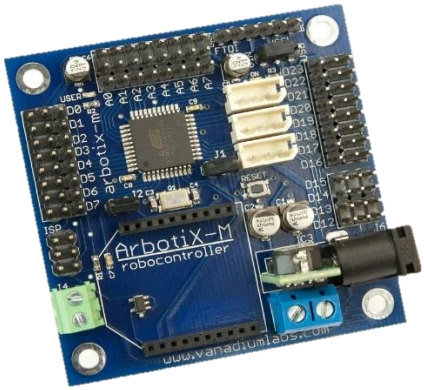

***Figura 1**. Placa Arbotix-M Robocontroller. Robocontroller, A. and IL-ARBOTIXM, I. (2017). ArbotiX-M Robocontroller. [online] Trossenrobotics.com. Available at: http://www.trossenrobotics.com/p/arbotix-robot-controller.aspx.*

6.3 Robot PhantomX Reactor

El robot a utilizar en este TFM es el brazo robótico PhantomX Reactor de la empresa Trossen Robotics que se muestra en la *Figura 2* ya ensamblado. Para su control, se utiliza el microcontrolador Arbotix-M robocontroller, como se explicó con anterioridad.

***Figura 2**. PhantomX Reactor Robot Arm. Kit, P. and KIT-RK-REACTOR, I. (2017). PhantomX AX-12 Reactor Robot Arm. [online] Trossenrobotics.com. Available at: http://www.trossenrobotics.com/p/phantomx-ax-12-reactor-robot-arm.aspx.*

Con el fin de conocer las propiedades físicas del robot para su posterior manejo, se representarán los grados de libertad, rotaciones y ejes de referencia del robot tal y como se muestra en la *Figura 3*. A su vez, es importante tener en cuenta las dimensiones del robot que se definen en la *Tabla 1* y las limitaciones mecánicas de los ángulos de rotación que se definen en la *Tabla 2.* De este modo, estos datos son las condiciones iniciales para empezar a desarrollar la cinemática del robot. Las articulaciones están formadas por los servomotores Dynamixel AX-12 que operan con tensiones de 9 a 12 V. [11]

Rotación	Mínimo (rad)	Máximo (rad)
q_1	-2.62	2.62
q_2	-0.33	2.97
q_3	-2.89	0.26
q_4	1.83	1.86
q_5	**-2.62**	**2.62**

***Tabla 2.** Limitaciones mecánicas de cada una de las articulaciones. Gutiérrez, Á. and Monasterio, F. (2017). Cinemática. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es*

Segmento	Medida (mm)
l_0	86.8
l_1	31.0
l_2	150.2
l_3	146.3
l_4	70.0
l_5	66.3

***Tabla 1**. Medidas de cada una de las extremidades del PhantomX Reactor. Gutiérrez, Á. and Monasterio, F. (2017). Cinemática. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es*

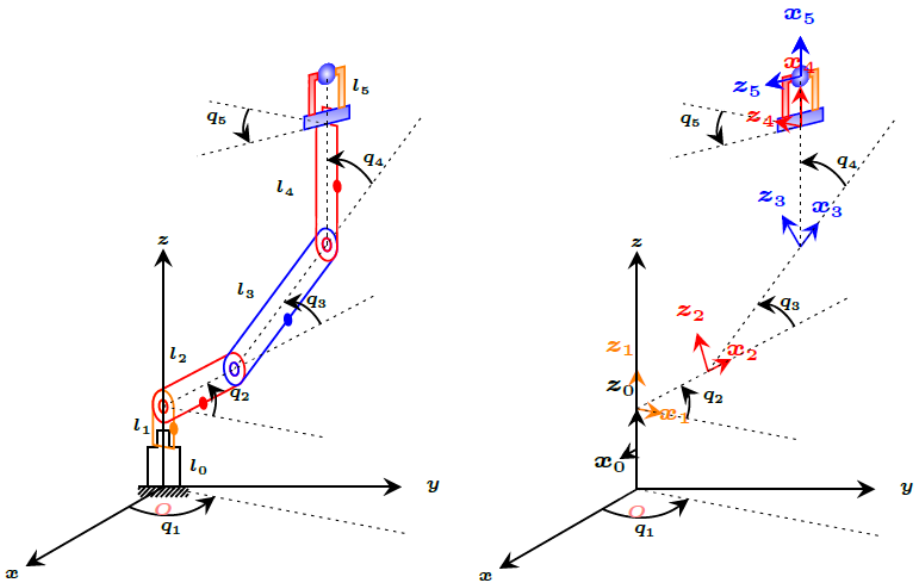

Figura 3. *Representación de los grados de libertad y sistemas de coordenadas locales. Gutiérrez, Á. and Monasterio, F. (2017). Enunciado Práctica 1. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es*

En la *Figura 3* se representan los sistemas de referencia locales de cada una de las piezas del robot. Debido a que las articulaciones a estudiar son todas rotacionales, se definen los sistemas de referencia locales como una rotación del sistema de referencia de su par cinemático. Donde l_i es la distancia entre articulaciones del par cinemático y l_0 la distancia entre el anclaje del suelo y la primera articulación del primer par cinemático. Por otro lado, se denominará punto Q, un punto significativo en el extremo del brazo, en este caso, el punto azul que se visualiza en medio de las pinzas del robot. A diferencia de cómo se representa en la *Figura 3* se elige un sistema de referencia propio, que es más intuitivo (*Figura 4*).

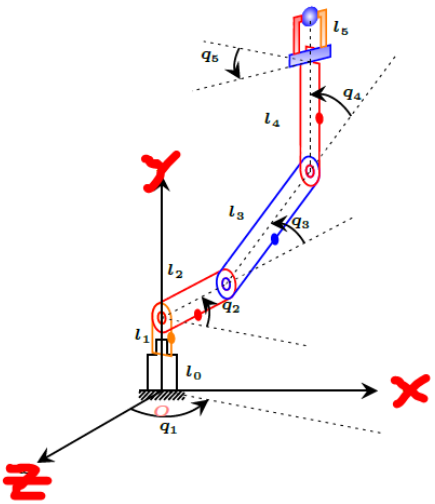

Figura 4. *Representación de los grados de libertad con el sistema de referencia seleccionado. Gutiérrez, Á. and Monasterio, F. (2017). Enunciado Práctica 1. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es*

En la *Figura 4* se visualiza cómo la primera articulación del robot gira sobre el eje Y, las tres siguientes sobre el eje Z y, por último, el giro de las pinzas del robot se realiza sobre el eje X.

Respecto a su instalación, se hace uso del software de Arduino, Arduino IDE y, también, de las diferentes herramientas de *Arbotix Std* disponibles para la asignatura de Control y Robótica en Medicina del Máster Universitario en Ingeniería Biomédica de la UPM, formado por diferentes *templates* que se encargan de la gestión de los servomotores y de las diferentes articulaciones y otra que almacenaba las diferentes coordenadas generalizadas. [12]

6.4 Cinemática directa.

Para poder desarrollar la interfaz gráfica del simulador 3D del brazo robótico, primero se debe saber cómo se comporta éste. Para ello, se debe realizar la cinemática directa e inversa y comprobar que, una vez implementado en el brazo, funciona correctamente.

A continuación, se presentará el análisis cinemático directo del robot, mediante el cual, al introducir los valores de los ángulos de cada una de las articulaciones, se obtiene la posición espacial del extremo final del brazo robótico.

Si se conoce las coordenadas de un punto *P* referido a un sistema S_1, es posible obtener las coordenadas en el otro sistema S_0, si se conoce la rotación R_0^1 y traslación t_0^1 entre ambos sistemas,

$$p_0 = R_0^1 p_1 + t_0^1 \tag{1.1}$$

donde p_0 son las coordenadas del punto *P* referido a S_0, p_1 son las mismas coordenadas del mismo punto *P* referido a S_1 y t_0^1 representa el vector de traslación expresado en coordenadas de S_0. Esta transformación dado por la Ecuación *1.1*, se denomina movimiento rígido.

Una vez obtenido el sistema de referencia, se define el conjunto de ejes de rotación de *U* que viene dado por la posición de cada pieza respecto a su par cinemático:

$$U = \{z_o, -y_1, -y_2, -y_3, x_4\} \tag{1.2}$$

El origen o_i de cada sistema de referencia local S_i se sitúa en la articulación del par cinemático con la excepción de o_n que se sitúa en el punto Q definido anteriormente

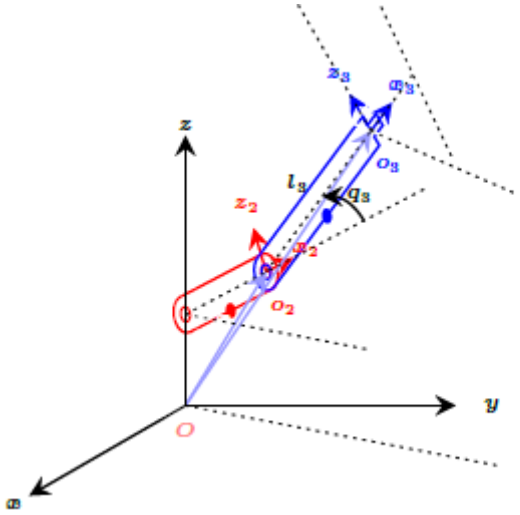

Figura 5. *Sistemas de referencia locales y par cinemático (2,3). Gutiérrez, Á. and Monasterio, F. (2017). Enunciado Práctica 1. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es*

En la *Figura 3* se muestra el par cinemático (2,3) que permite definir la coordenada generalizada q_3. El eje de rotación de la pieza 3 es -y_2.

Se define el vector d_{i-1}^{i} expresado en coordenadas del sistema de referencia S_{i-1}, como el vector

$$d_{i-1}^{i} = \overrightarrow{o_{i-1} - o_i} \tag{1.3}$$

Dando por supuesto que l_i es la distancia entre articulaciones del par cinemático

$$l_i = |\overline{o_{i-1} - o_i}|$$

Se define el conjunto $\tilde{O} = \{\widetilde{o_0}, \widetilde{o_1}, \widetilde{o_2}, \widetilde{o_3}, \widetilde{o_4}, \widetilde{o_5}, \widetilde{o_6}\}$ como

$$\tilde{O} = \{l_0 k, l_1 k, l_2 i, l_3 i, l_4 i, l_5 i\} \tag{1.4}$$

donde o_i es el punto expresado en coordenadas del sistema S_{i-1}, se refiere a la orientación del cuerpo antes de realizar la rotación. Con este convenio:

$$d_{i-1}^{i} = R_{i-1}^{i} \widetilde{o_i} \tag{1.5}$$

Por lo que con esta ecuación se obtiene una relación recursiva de este vector por medio del cálculo de R de la siguiente manera, teniendo en cuenta el par cinemático de la *Figura 3*. Las matrices de rotación quedan definidas de la siguiente forma:

$$R_0^1 = R_{z,q1} = \begin{bmatrix} cosq_1 & sinq_1 & 0 \\ -sinq_1 & cosq_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.6a}$$

$$R_1^2 = R_{-y,q2} = \begin{bmatrix} cosq_2 & 0 & -sinq_2 \\ 0 & 1 & 0 \\ sinq_2 & 0 & cosq_2 \end{bmatrix} \quad (1.6b)$$

$$R_2^3 = R_{-y,q3} = \begin{bmatrix} cosq_3 & 0 & -sinq_3 \\ 0 & 1 & 0 \\ sinq_3 & 0 & cosq_3 \end{bmatrix} \quad (1.6c)$$

$$R_3^4 = R_{-y,q4} = \begin{bmatrix} cosq_4 & 0 & -sinq_4 \\ 0 & 1 & 0 \\ sinq_4 & 0 & cosq_4 \end{bmatrix} \quad (1.6d)$$

$$R_4^5 = R_{x,q5} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosq_5 & sinq_5 \\ 0 & -sinq_5 & cosq_5 \end{bmatrix} \quad (1.6e)$$

Con estos datos se calculan los vectores d_{i-1}^i mediante la regla de la cadena se puede calcular de forma recursiva, a partir de la ecuación fundamental de la cinemática directa de posición, se obtiene los diferentes vectores hasta d_0^5:

$$d_0^i = d_0^{i-1} + R_0^{i-1} d_{i-1}^i \quad (1.7)$$

La relación puede expresarse en forma recursiva, a partir de la ecuación fundamental de la cinemática directa de orientación, se calculan todas las rotaciones hasta R_0^5:

$$R_0^i = R_0^{i-1} R_{i-1}^i \quad (1.8)$$

Con estas dos últimas ecuaciones se obtiene de manera recursiva la posición del punto Q y la orientación del extremo final, expresados en las coordenadas del sistema S_0.

Implementando estas ecuaciones en Matlab se obtiene la función de cinemática directa para para calcular la posición y orientación del extremo de la articulación, pasándole como parámetro de entrada cada uno de los ángulos en los que se encuentran las articulaciones. [9]

6.5 Cinemática inversa

Esta técnica permite obtener los ángulos de rotación de los elementos del brazo robot a partir de las coordenadas generalizadas (X_O, Y_0, Z_0) y la orientación. El robot con el que se trabaja tiene cinco grados de libertad. Las tres últimas articulaciones intersecan en un punto y permiten el desacoplo cinemático de posición y orientación. Al punto de intersección de las tres últimas articulaciones se le denomina centro de la muñeca.

El procedimiento que se sigue es el siguiente:

1. Obtener el centro de la muñeca C.

$$\overrightarrow{OC} = \overrightarrow{OQ} - d_M \vec{a} \quad (2.1)$$

en donde Q corresponde con el extremo de la mano y $\vec{a}$ corresponde con el vector avance y d_M con la dimensión de la mano que viene dada por

$$d_M = l_4 + l_5 \quad (2.2)$$

2. Resolver el problema cinemático inverso de la posición C para obtener $\{q_1, q_2, q_3\}$.

El eje de rotación de la primera pieza es z_0 por lo que q_1 se obtiene de forma inmediata

$$q_1 = atan2(C_y, C_x) \quad (2.3)$$

Para obtener q_2 y q_3 es necesario resolver un sistema de dos ecuaciones trigonométricas

$$C_{x'} = l_2 cosq_2 + l_3 \cos(q_2 + q_3) \quad (2.4a)$$

$$C_{z'} = l_2 sinq_2 + l_3 \sin(q_2 + q_3) \quad (2.4b)$$

donde:

$$C_x = C_{x'} cosq_1 \quad (2.5a)$$

$$C_y = C_{x'} sinq_1 \quad (2.5b)$$

$$C_z = C_{x'} sinq_1 \quad (2.5c)$$

Al resolver el sistema de ecuaciones dado por las ecuaciones (2.4a) y (2.4b) se obtiene q_2, la raíz se toma negativa debido a las limitaciones del ángulo:

$$q_2 = -\beta + atan2(L, -\sqrt{(1 - L^2)}) \quad (2.6)$$

donde:

$$\beta = atan2(C_{x'}, C_{z'}) \quad (2.7)$$

$$L = \frac{l_2^2 + c_{x'}^2 + c_{z'}^2 - l_3^2}{2l_3\sqrt{(c_{x'}^2 + c_{z'}^2)}} \quad (2.8)$$

$$C_{x'} = C_x cosq_1 + C_y \sin q_1 \quad (2.9)$$

$$C_{z'} = C_z - (l_0 + l_1) \quad (2.10)$$

Para finalizar con este punto, se obtiene q_3 a partir del sistema de ecuaciones dado por (2.4), conocida q_2

$$q_3 = -q_2 + atan2(C_{z'} - l_2 sinq_2, C_{x'} - l_2 cosq_2 \quad (2.11)$$

3. Se resuelve el problema cinemático inverso de orientación. Cálculo de la matriz de rotación R_0^3, llamada R_{SM}.

$$R_0^3 = R_0^1 R_1^2 R_2^3 \quad (2.12)$$

Las matrices R_0^1, R_1^2 y R_2^3 se calculan tal y como se ha explicado en la formulación de la cinemática directa.

4. Se calcula la matriz R_{SM} que se corresponde con la matriz U, a partir de las matrices R_{SB} y R_{S0}

La matriz de rotación R_{S0} esta referida al sistema inercial y depende de cómo se haya definido el eje de avance fijo de la mano.

Se calcula el vector ***n*** de la siguiente forma:

$$n = s\ x\ a \tag{2.13}$$

Como el eje de avance es *el eje X*

$$R_{SO} = \begin{pmatrix} a_x & -n_x & -s_x \\ a_y & -n_y & -s_y \\ a_z & -n_z & -s_z \end{pmatrix} \tag{2.14}$$

Se calcula la matriz R_{SM} que se define como U a partir de ahora, de este modo se resuelve el problema cinemático inverso de orientación

$$q_4 = atan2(U_{31}, U_{11}) \tag{2.15a}$$

$$q_5 = atan2(-U_{23}, U_{22}) \tag{2.15b}$$

De esta forma, se calcula todos los ángulos sabiendo las coordenadas generalizadas y la orientación del robot, por lo que nuestra función en Matlab tendrá nueve parámetros de entradas (coordenadas generalizadas donde se sitúa la punta de la pinza, vector de desplazamiento y vector de aproximación) y como salida un vector con 5 elementos, uno por ángulo de cada articulación. [9]

7. Características principales

En este apartado, se describen los diferentes módulos o hitos de los que se compone el proyecto y se especifican las principales características de cada uno de ellos. Para la realización del Trabajo Fin de Máster se han implementado los siguientes módulos.

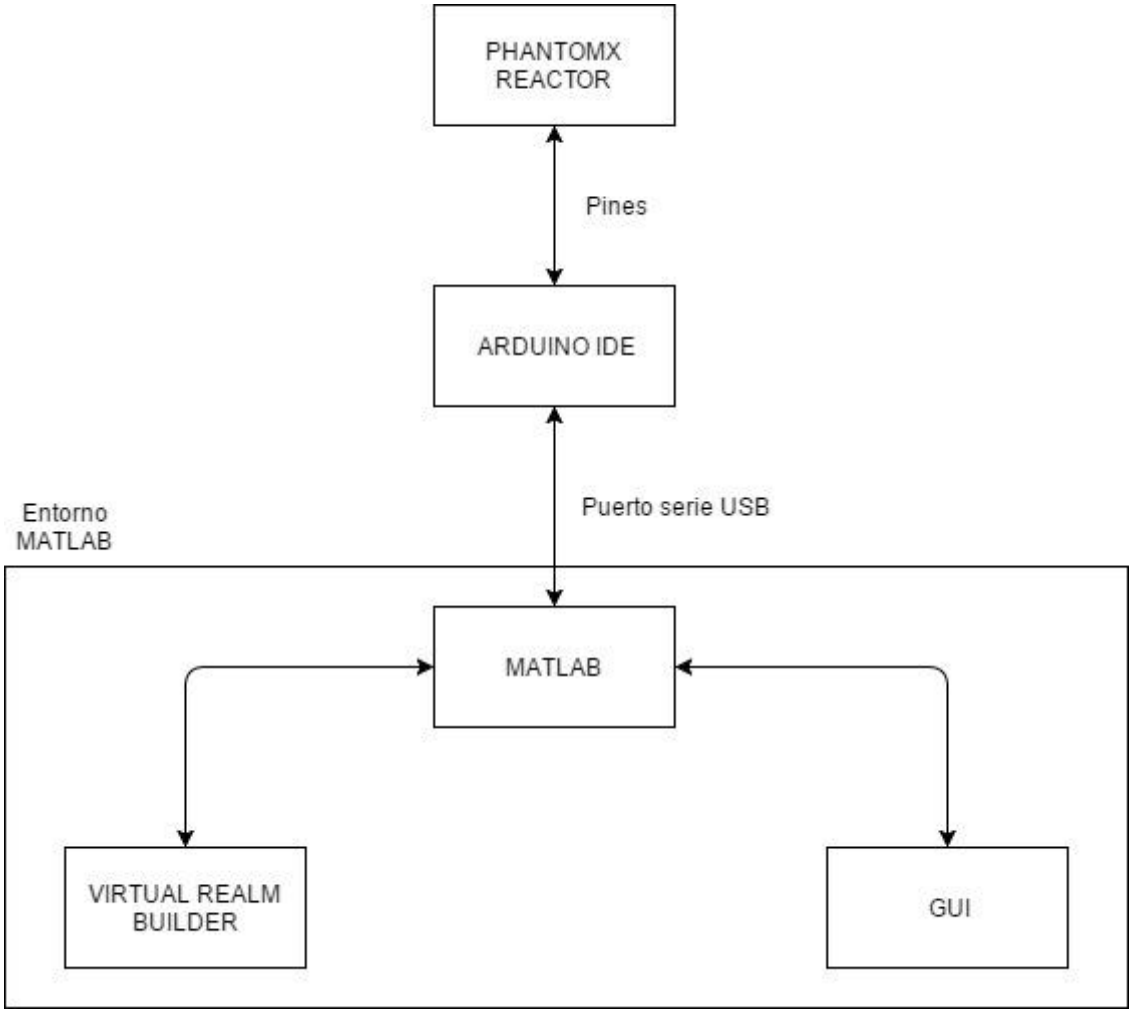

Figura 6. *Arquitectura del proyecto*

Por un lado, el robot que a través de los pines del Arbotix se comunica con la tarjeta, recibiendo la placa la información referente a los servos. El Arbotix-M robocontroller, mediante el software Arduino IDE es el módulo con contacto directo con el robot, toda la información procesada en Matlab debe pasar a través del puerto serie USB a la tarjeta Arbotix. Por lo tanto, es muy importante la correcta programación de la lectura y escritura de datos a partir del puerto serie, de manera que ésta sea óptima y haga exactamente lo que se quiera.

Por otro lado, Matlab y sus herramientas, GUI y Virtual Realm Builder, en parte, ambas herramientas son programadas a través de Matlab, aunque la parte de diseño se hace de manera independiente como se explicará más tarde.

8. Hardware

Lo más importante en la parte de hardware son las especificaciones del robot, aunque también hay que hacer mención que la comunicación entre Matlab y la placa Arbotix se hace a través de un cable USB.

8.1 Cable USB

El puerto USB está incorporado en todos los ordenadores de sobremesa y portátiles, tiene una velocidad de transmisión de más de 12 Mbps, más que suficiente para nuestro proyecto y provee de alimentación al dispositivo que se conecta. Además, una de sus principales características es que permite el envío y la recepción de datos de manera simultánea con el Arbotix Std. [10]

8.2 PhantomX Reactor

Como se comentó en la introducción, el PhantomX Reactor tiene 5 grados de libertad. La estructura y la base sólida está hecha en ABS, que es un material resistente a los golpes, las articulaciones se mueven gracias a los servos AX-12A Dynamixel Actuators. Tiene un peso de 1.36 Kg, un alcance vertical de 51 cm y un alcance horizontal de 38 cm con una fuerza más que suficiente en el agarre de las pinzas. [11]

8.3 Dynamixel 12A Actuator.

Es un actuador específico para aplicaciones en Robótica. Este servomotor tiene la interesante capacidad de poder detectar su movimiento, velocidad, temperatura, ángulo y la carga. Por lo que para aplicaciones de enseñanza es ideal, pudiendo monitorizar y variar estas características. Este servo es digital por lo que se controla por medio de un algoritmo propio de estos motores como se ve en la implementación del proyecto. Las diferentes especificaciones de este servo se pueden ver en la *Tabla 3*. [11]

Característica	Descripción
Voltaje de operación	12V
Torque	15Kg/cm
Velocidad	0.169seg/60º
Peso	565g
Resolución Temperatura	0.29ºC
Ángulo de operación	300º
Corriente máxima	900 mA
Corriente de espera	50 mA
Protocolo	TTL Half Duplex Async Serial
Retroalimentación	Temperatura, Voltaje, Posición

Tabla 3. *Especificaciones del servomotor Dynamixel AX-12ª. Granat. K. (2017). Dynamixel AX-12 Documentation. [online] Learn.trossenrobotics.com. Available at: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx*

9. Software

A continuación, se describen con profundidad los diferentes softwares y herramientas utilizadas.

9.1 Arduino IDE.

Es un espacio de desarrollo integrado basado en Processing. Processing da soporte a lenguajes de programación como C, C++, el nuevo C# y Java [12]. El IDE de Arduino permite comunicar el microcontrolador Arbotix Std con el ordenador y así poder integrar cualquier programa escrito en C o cualquier lenguaje de programación que derive de C. Ese programa podrá interactuar con cada salida o entrada de nuestro Arduino. Una vez escrito, se carga a través del USB y la placa empezará a trabajar de forma automática. [13]

9.2 Matlab

Matlab es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M). Gracias a este software se manipulan una gran cantidad de datos, operar con las matrices en la cinemática del robot y la implementación de algoritmos, entre otras muchas cosas. En definitiva, Matlab es el software ideal para llevar a cabo proyectos en donde se encuentren implicados elevado número de cálculos matemáticos y la visualización de gráficas de los mismos.

Desde el punto de vista de control, Matlab se puede considerar un entorno matemático de simulación que puede utilizase para modelar y analizar sistemas. Para ello, dispone de dos herramientas adicionales fundamentales que expanden sus prestaciones, Simulink, que no se ha utilizado en este proyecto y GUIDE (editor de interfaz de usuario – GUI) y muchas herramientas que se añaden con extensiones, como puede ser V-Realm Builder, utilizado para este proyecto. [8] [16]

9.3 V-Realm Builder

Este software de diseño permite crear, visualizar y conectar un mundo virtual previamente diseñado mediante una sencilla interfaz de funciones orientadas a objetos y bloques (*ver Figura 7*). El lenguaje utilizado es VRML97 que se conoce comúnmente como *worlds*¸ por ello la extensión de estos archivos es .wrl.

De este modo, se pueden insertar diferentes estructuras para formar el brazo robótico y enlazarlas físicamente para que tenga el mismo comportamiento que el robot real. Hay que tener en cuenta que las medidas, por convenio, están en el sistema internacional y no se pueden modificar. [17] [18]

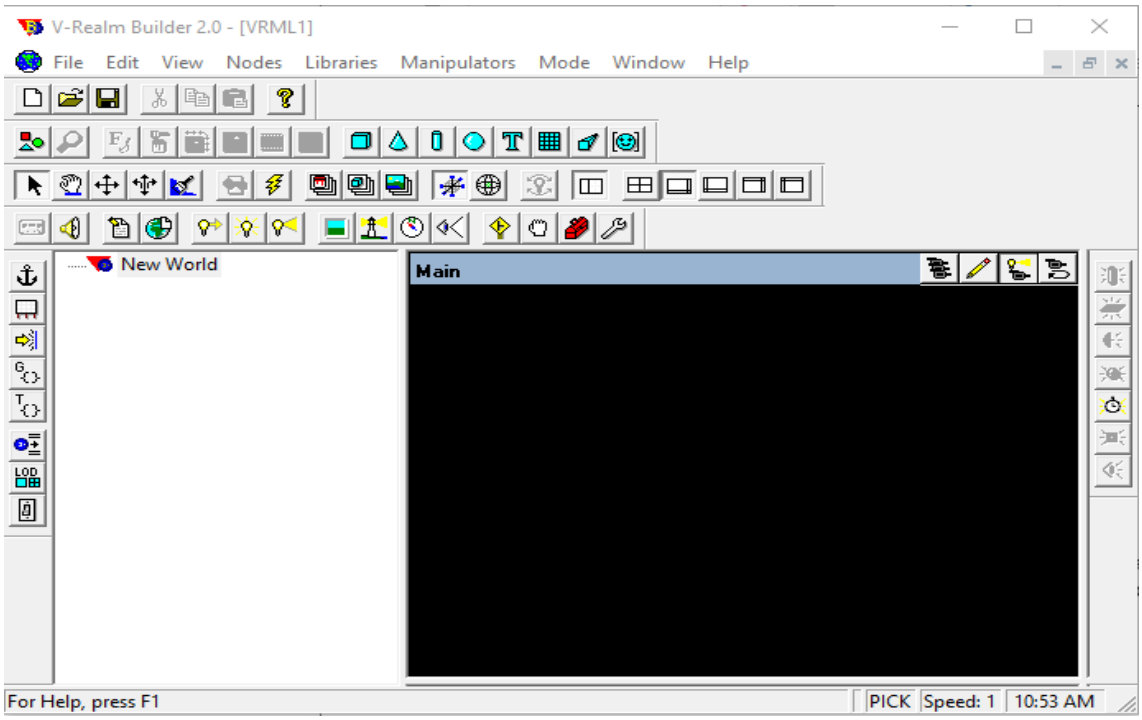

Figura 7. *Interfaz V-Realm Builder*

9.4 GUI de Matlab

Esta herramienta permite un control sencillo de las aplicaciones de software. Las Apps de Matlab, están hechas con esta herramienta como front-end y con Matlab como back-end. La GUI incluye diferentes botones y funcionalidades para poder hacer casi cualquier interfaz que se desee (*Tabla 4*).

Una vez creado la interfaz de usuario se crea automáticamente el código de Matlab para construirla, el cual se puede modificar para programar el comportamiento de la App. [19]

El código generado al crear la interfaz está compuesto por 3 tipos de funciones:

- *OpeningFcn.* En esta parte del código se incluirán las diferentes funciones que se van a ejecutar nada más se abra el programa, ya sean las imágenes de la interfaz, valores predeterminados o inicializar el mundo virtual.
- *CreateFcn.* Aunque no se ha utilizado durante el proyecto, este tipo de funciones se utilizan para otorgar condiciones iniciales a las gráficas.
- *Calbacks.* Este tipo de funciones son la parte de código que se ejecutará al darle a un botón determinado. Cada elemento de la interfaz tiene asociado tanto un *CreateFcn* como un *Callback*.

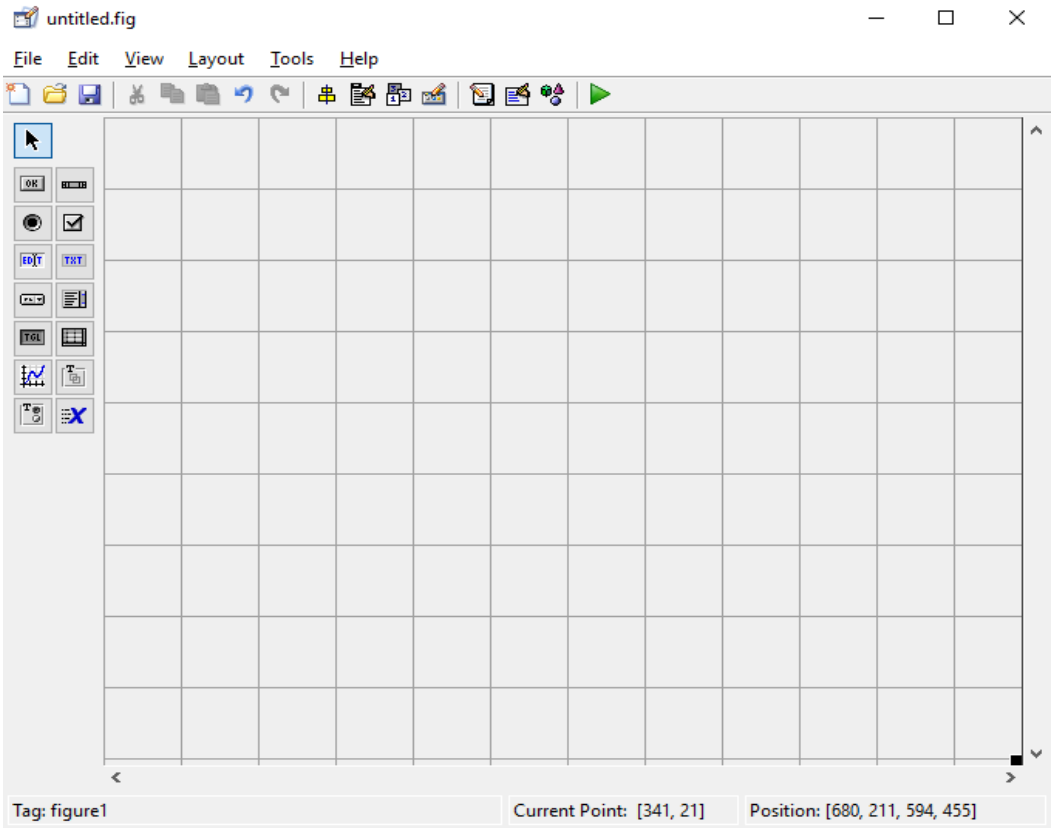

Figura 8. *Interfaz de usuario GUI*

Los diferentes botones que se pueden utilizar en el GUI de Matlab son los siguientes:

CONTROL	DESCRIPCIÓN
Push Button	Genera una acción
Slider	Representa un rango de valores
Radio Button	Representa una opción
Check Box	Inicia el estado de una opción
Edit Text	Para editar texto
Static Text	Muestra un string de texto
Pop-up Menu	Provee una lista de opciones
Listbox	Lista deslizable
Toggle Button	Genera una acción on, off
Axes	Para graficar
Panel	Visualiza grupo de controles
Button Group	Es un panel exclusivo para radio buttons y Toggle buttons
ActiveX Control	Despliega controles ActiveX en GUI

Tabla 4. *Descripción botones del GUI Matlab. Barragán, D. (2012). Manual Interfaz Gráfica Usuario de Matlab. [online] www.dspace.espol.edu.ec. Available at: Diego Orlando Barragan. (2012). Manual Interfaz Gráfica Usuario de Matlab. 2012, de Matpic Sitio web: https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf*

10. Implementación

En este apartado se describe cómo se ha llevado a cabo el proyecto en los diferentes módulos descritos con anterioridad. El *Back-end* del GUI de Matlab se le considera como parte del Matlab y el GUI propiamente dicho está compuesto por el diseño del interfaz de usuario y sus propiedades gráficas.

10.1 V-Realm Builder

Para llevar a cabo la creación del modelo 3D de nuestro brazo robótico, el primer paso es insertar una pieza llamada Robot PhantomX. Existen nodos especiales que pueden agrupar a otros nodos si estos últimos se definen en su campo *Children*. Un nodo agrupador sirve para que varios objetos tengan las mismas propiedades como transformaciones geométricas, comportamiento, etc.

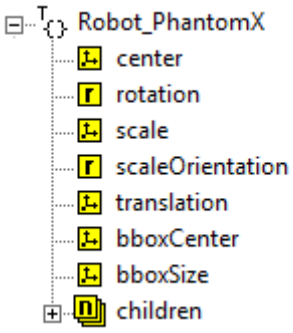

Figura 9. *Características del robot*

Como se visualiza en la *Figura 9,* se crea una pieza para el ensamble con los siguientes parámetros modificables según interese para el Robot. En este caso, no se modifica ninguno de los parámetros ya que esta transformada hace exclusivamente de conector entre las demás piezas.

La siguiente pieza por insertar será un cilindro desde la barra de tareas que hace de soporte del robot y como articulación que rotará sobre el eje Y. En este caso, la propiedad *rotation* se debe modificar, puesto que por defecto viene que gire respecto al eje Z.

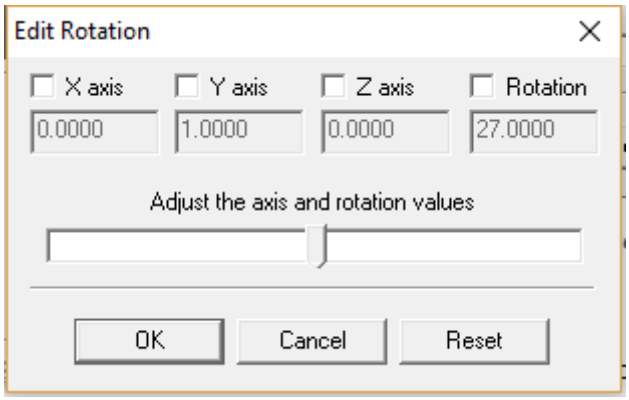

Figura 10. *Ventana de configuración para la rotación de la base del robot*

Otra de las propiedades que se tiene que modificar es *traslation*, ya que al haber insertado el suelo con anterioridad se ha de trasladar el soporte 1 cm más arriba ya que este es el grosor del suelo para que quede apoyada en él. Por último, también se dará las dimensiones correctas a la base tanto altura como radio, para que se asemeje a las medidas de la base del robot físico.

A continuación, se inserta un nuevo bloque en el apartado *Children* del anterior nodo para que tenga las mismas propiedades como transformaciones geométricas o comportamiento. Se inserta un cilindro que hará de servo en el que se le cambia las dimensiones y la rotación, que se debe colocar a 90º respecto el eje X para que adopte la posición tumbada sobre el cilindro anterior.

A la misma altura que este servo, se inserta el primer miembro del robot como un rectángulo. En el cuál se modificará la propiedad *center* para que rote sobre el servo y no sobre la mitad del miembro. Al medir el miembro 0.2 m, el centro se atrasa 0.1 m para que coincida con el servo implementado anteriormente.

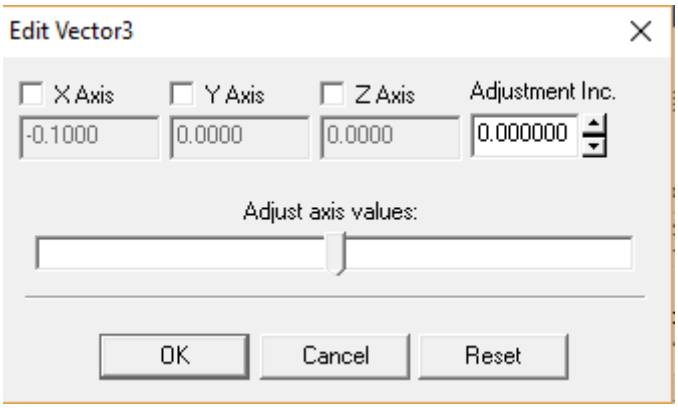

Figura 11. *Ventana de configuración del centro de rotación del brazo*

A continuación, se crea el background y los diferentes puntos de visión para la visualización de nuestro brazo. Se crean dos puntos diferentes *Frontal* y *Superior*, donde se deben de tener en cuenta sus propiedades de posición y orientación de la cámara para obtener la visión deseada.

Realizado estos pasos, se obtiene el siguiente modelo en V-Realm Builder:

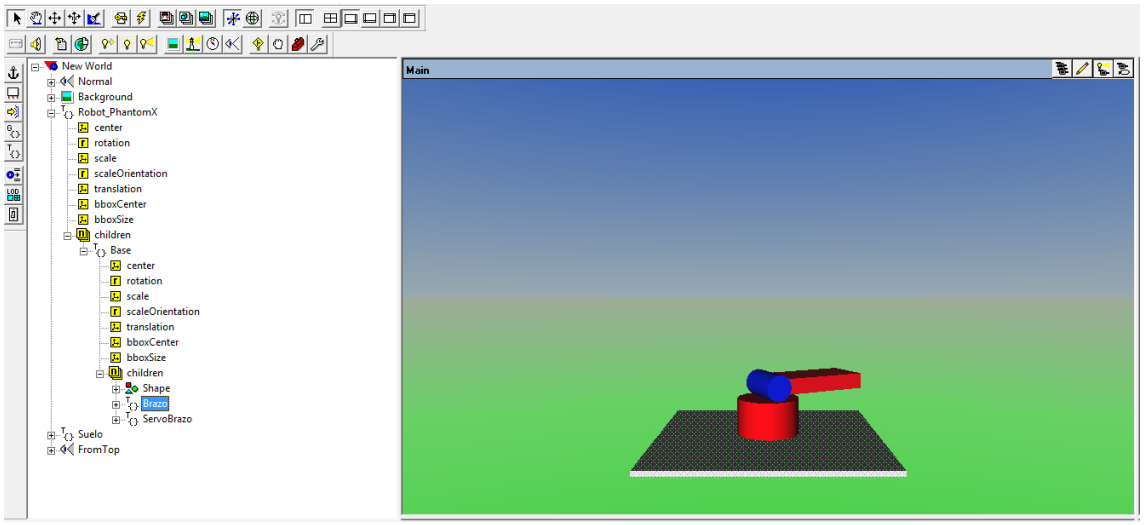

Figura 12. *Visión general del modelo sin terminar*

De igual forma, se procede a realizar la inserción de las diferentes partes del robot y la relación que tiene las unas con las otras. Cabe destacar la inserción de las pinzas, que se hizo mediante extrusión. En el cuál, se inserta un cuadrado, pero nos da la posibilidad de moldearlo como se ve en la *Figura 13*. Finalmente, cuando ya se tiene la forma deseada se modifican las mismas propiedades de igual modo que en los demás nodos, tanto la traslación y orientación como el tamaño de cada pinza.

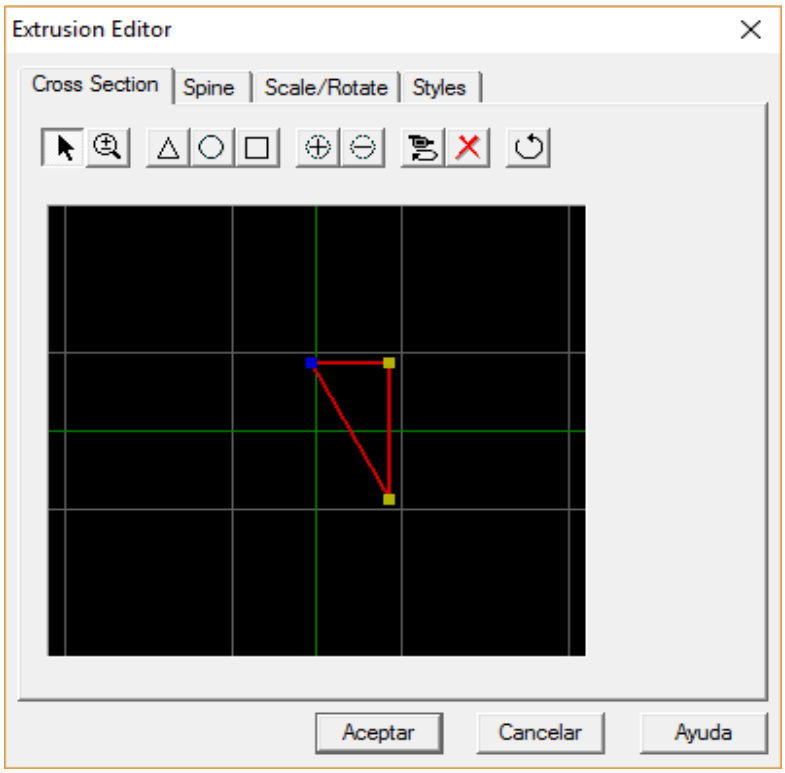

Figura 13. *Ventana de configuración de la extrusión insertada*

Una vez creado y ajustado las pinzas y darles color a las diferentes partes del robot, se concluye el proceso de modelado 3D del brazo robótico, quedando como se puede ver en la *Figura 14*.

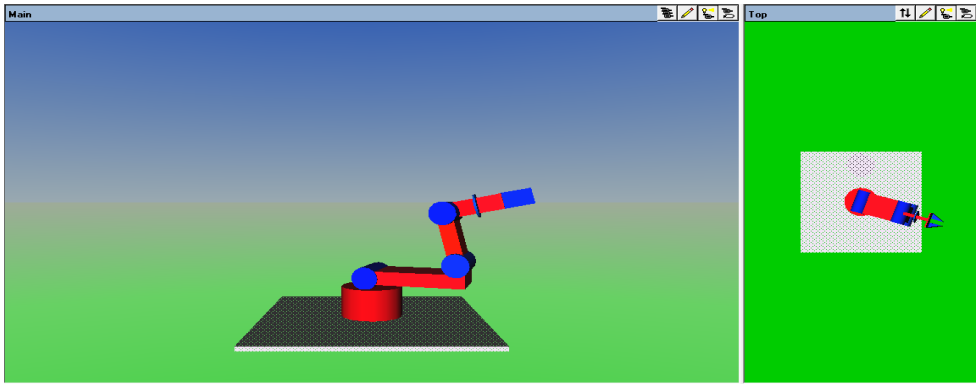

Figura 14. *Modelo 3D del brazo robótico en dos vistas. Frontal y Superior*

10.2 GUI de Matlab

En este apartado se explicará en profundidad el front-end de la aplicación para el manejo del brazo, es decir, la parte del proyecto que ve y utiliza el usuario. Posteriormente, cuando se explique la implementación en Matlab, se detallará el back-end, es decir, qué función se ejecuta una vez pulsado un determinado botón.

Para crear la interfaz en Matlab hay que escribir en la ventana de comandos el comando *guide*, de esta manera permite la opción de abrir una nueva ventana en blanco para realizar la interfaz.

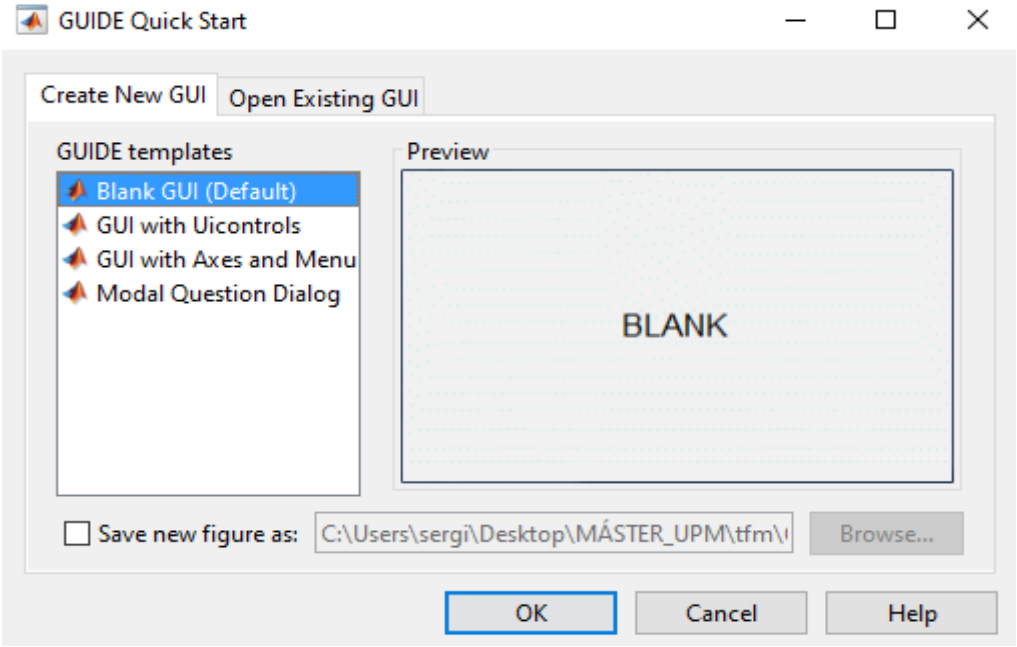

Figura 15. *Creación de la interfaz de Matlab*

De esta forma se crea un nuevo proyecto y se podrá empezar a insertar botones como se puede observar en la *Figura 8* anteriormente mencionada.

Una vez insertado los botones necesarios en nuestro GUI, queda de la siguiente forma:

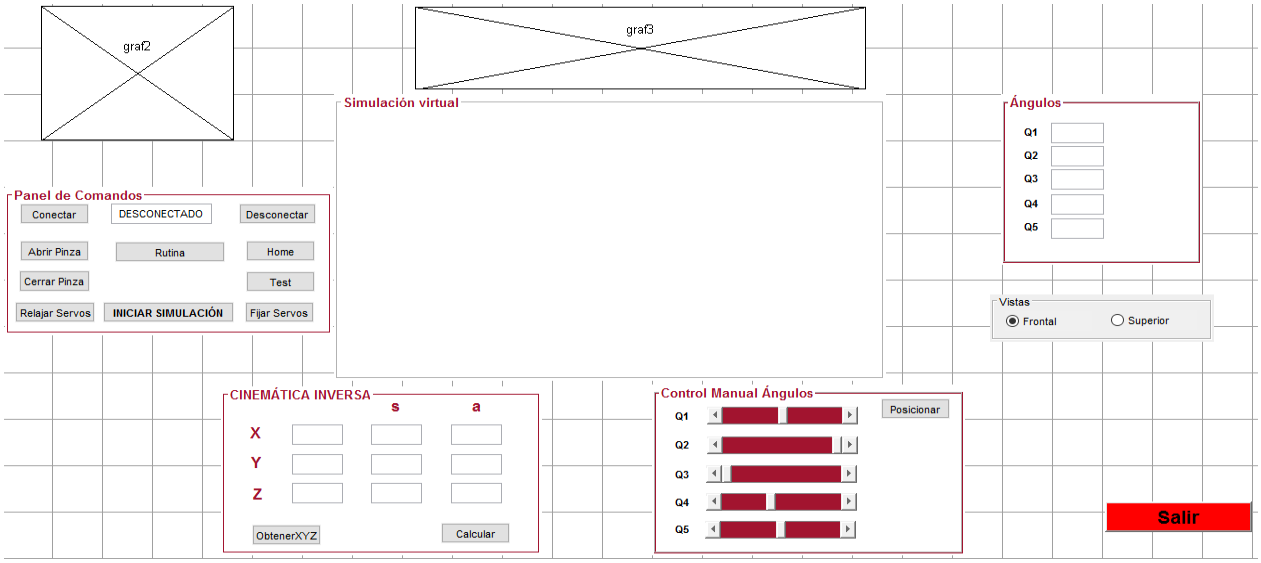

Figura 16. *Interfaz GUI Matlab sin ejecutar*

A continuación, se describe que tipo es cada botón, sus propiedades y que es lo que hace:

- Tanto la gráfica *graf2* como la gráfica *graf3* se utilizan para insertar imágenes en el GUI. De estas dos graficas no se cambian sus propiedades, solo su forma y el nombre de la función que se genera en el Matlab.
- La cajetilla de Simulación virtual es donde se va a situar la simulación una vez se ejecute el GUI.
- Con los botones de radio insertado en el panel de *Vistas* se puede cambiar la vista de frontal a superior y viceversa.
- Por otro lado, en el panel de comandos se puede diferenciar los siguientes botones.
    - Con el botón *Conectar*, se conectará al puerto serie actual.
    - Con el botón *Desconectar*, se desconectará del puerto serie actual.
    - Con los botones Abrir y Cerrar pinza se actúa sobre la pinza
    - Con el botón *Home*, el robot vuelve a su posición de partida
    - Con el botón *Test*, el robot realiza un Test para comprobar el correcto funcionamiento de todos sus servos.
    - El botón Iniciar simulación es un botón tipo *Toggle*, es decir, que una vez pulsado se ejecuta el código en su interior de forma cíclica hasta que sea pulsado nuevamente.
- En la cajetilla de *Ángulos* se monitoriza la posición en la que se encuentra cada uno de los servos en radianes.
- Con la cajetilla de Cinemática Inversa, se ejecuta la función que hace la cinemática inversa una vez se haya rellenado las casillas de las coordenadas generalizadas, el vector de desplazamiento y el de aproximación. También con el botón *Obtener*, se puede reescribir en las casillas de las coordenadas generalizadas la posición actual de la punta de la pinza del robot.
- Con los diferentes Sliders, se controla manualmente cada servo. En este tipo de botones, se debe cambiar en propiedades tanto el máximo y el mínimo, según sean los ángulos límite para cada servo, como se especifica en la *Tabla 2.*

- Con el botón *Posicionar* dentro de la cajetilla de los sliders, al presionarlo, se actualiza la posición de los diferentes sliders para que coincida con la posición actual de cada uno de los servos.
- Con el botón de *Salir* se cierra la aplicación de forma segura.

Para otorgar propiedades a los diferentes botones o apariencias como el color, tamaño de fuente o nombre de la llamada a este objeto, se selecciona con doble clic cada estructura que se quiera modificar y se abrirá una nueva ventana llamada *Property Inspector* en la que se puede cambiar todas las características que convengan.

Cabe destacar que, en las características de los *Sliders*, se da la posibilidad de cambiar los valores máximos y mínimos a los diferentes límites de movilidad de cada servo (como se especifica en la *Tabla 2*) para que, de este modo, el modelo 3D no actúe fuera de los límites físicos del robot. Además, también se cambia la variación de los ángulos tanto si es pulsada en el mismo slider (por defecto cambiaba de 1 radián en 1 radián y se modifica a 0.18 radián por clic) como si se pulsa las flechas situadas en los extremos del slider (por defecto 0.5 radián y lo se debe de modificar a 0.09 radián por clic). De este modo, se visualizará de mejor manera el movimiento del robot real y del modelo 3D

Una vez ejecutado el GUI, queda de la siguiente manera:

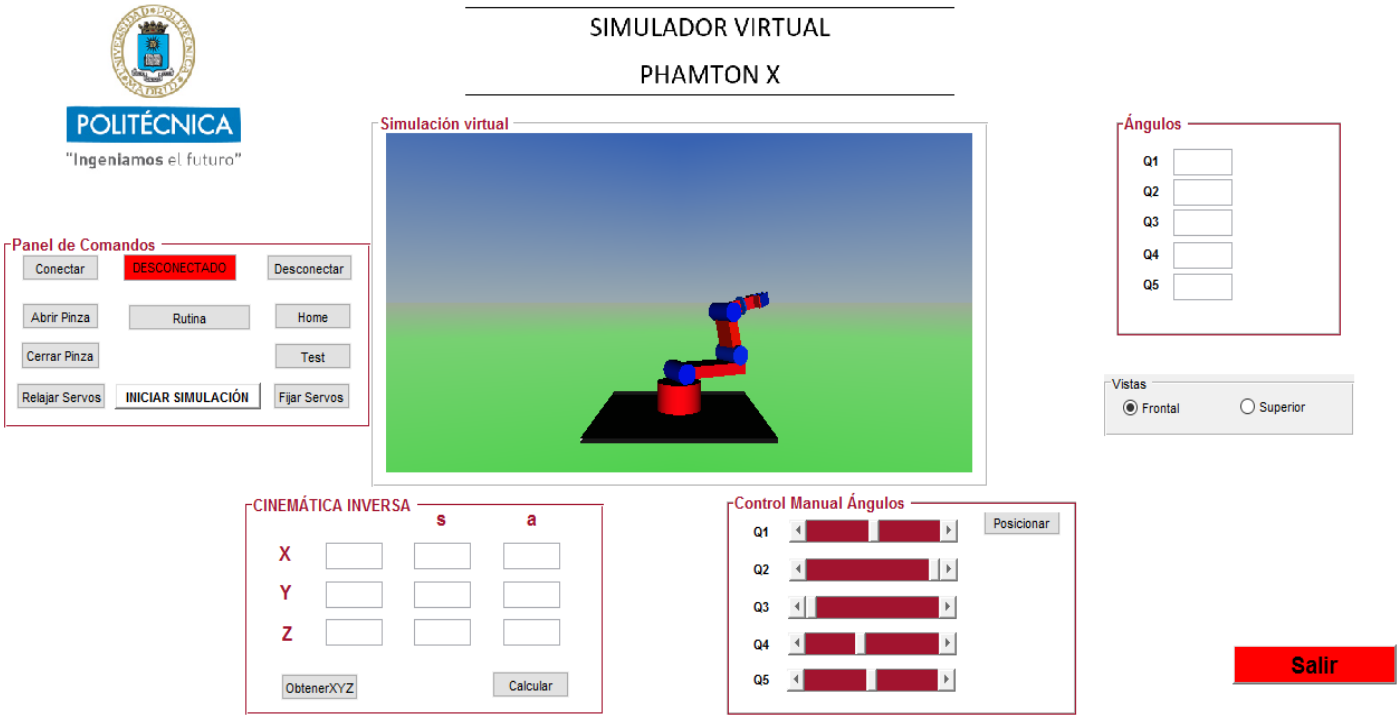

***Figura 17**. GUI de Matlab ejecutado*

10.3 Matlab

Aunque es preferible utilizar Simulink para controlar el mundo virtual por sencillez e intuición, se han implementado archivos .m y programación en Matlab para controlar dicho entorno.

Matlab será el entorno de desarrollo principal del proyecto, con Matlab se programa las diferentes funciones a ejecutar según la acción a tomar, algunos scripts útiles y el back-end del interfaz usuario.

Se empezará explicando cómo se llevó a cabo la lectura del estado de los servos que nos llega por el puerto serie.

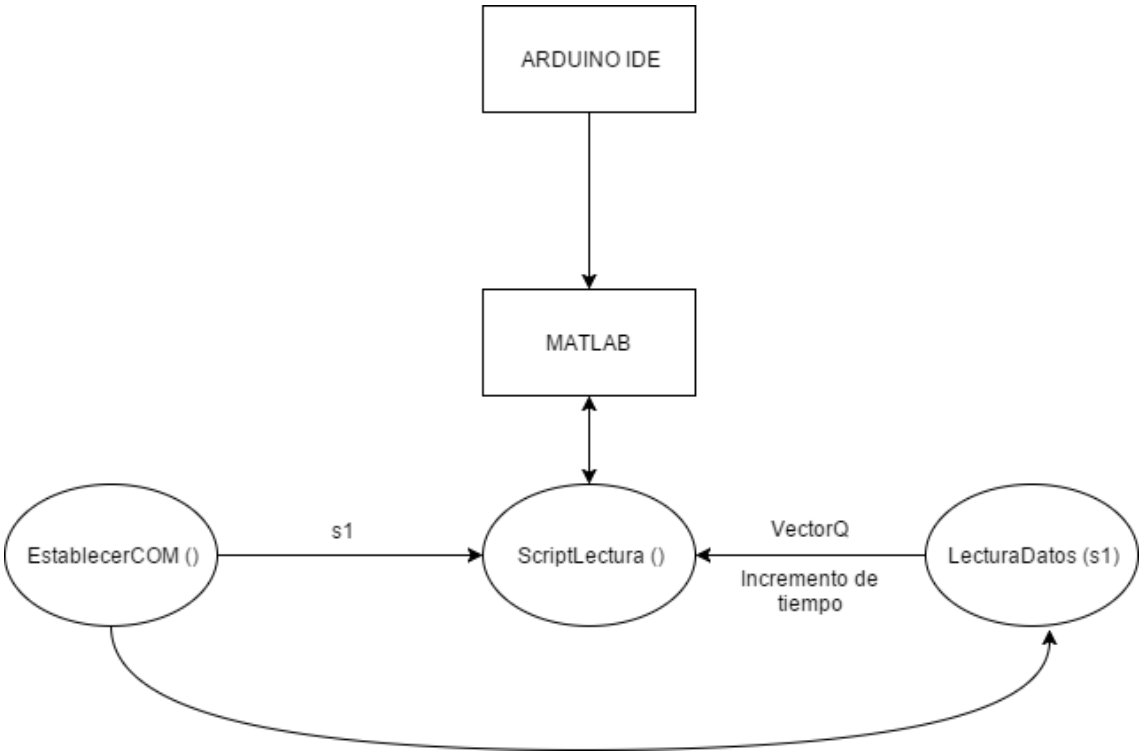

Figura 18. *Esquema de recepción de datos con Matlab*

Para la lectura de datos del puerto serie se utiliza como script principal *ScriptLectura (),* que con ayuda de la función *EstablecerCOM ()* y *LecturaDatos (s1)* se puede llegar a cabo satisfactoriamente la lectura.

El primer paso de todos es establecer la conexión con el puerto serie para poder leer los datos que llegan al ordenador desde el robot pasando por el microcontrolador. De esta forma, se ejecuta dentro del script la función *EstablecerCOM ()* en la cual se elige las distintas propiedades de transmisión de datos como la velocidad en baudios de transmisión (en este caso 115200) o la paridad, entre otras cosas. Después de definir las propiedades se abre el puerto y ya estaría listo para recibir los datos.

Posteriormente, se inicializa la matriz donde se guarda los vectores y también, declarar el tiempo de ejecución. A continuación, se lleva a cabo la lectura de los datos del puerto serie, con ayuda de la función *LecturaDatos(s1)* donde s1 es el puerto serie de interés en el cual se lee los datos que tengan una estructura similar a la de un vector, es decir, que el carácter de inicialización sea "[", el carácter de separación la coma (",") y el carácter de finalización el cierre de corchetes "]". De este modo, cuando se lea todo el vector, se reinicializa de nuevo los parámetros para leer la siguiente trama. Si en algún momento nos interesa parar el script, presionando el botón "q" se finaliza el programa.

A continuación, se desarrollará la creación del back-end del GUI de Matlab y, sobre esta explicación, se irá explicando las diferentes funciones creadas en Matlab, invocadas en el back-end.

Después de crear el GUI como se indicó en la descripción del software, se crean dos archivos, un .fig, que es la interfaz usuario de la aplicación y un .m asociado, que es la programación que hay detrás de la interfaz, en el cual se inserta automáticamente las funciones que se comentó con anterioridad, OpeningFcn CreateFcn y Callbacks.

Antes de empezar a describir el código detrás de cada botón se programará qué se desea que se inicialice cuando se abra el programa. Por un lado, se inicializa las dos graficas correspondientes a cada una de las imágenes en el GUI y también las cajas de edición de texto para que se inicialicen en blanco o con un cero, según nos convenga. Además, se inicializará también el modelo 3D hecho previamente en V-Realm Builder. Para ello, se utilizará una librería específica para abrir y manipular este tipo de objetos en Matlab, como se puede ver en la *Figura 22*.

vrworld	Create new vrworld object associated with virtual world
vrdrawnow	Update virtual world
vrclear	Remove all closed virtual worlds from memory
vrwho	List virtual worlds in memory
vrwhos	List details about virtual worlds in memory
vrnode	Create node or handle to existing node
vr.canvas	Create virtual reality canvas
vr.canvas.capture	Capture virtual reality canvas image
vrfigure	Create virtual reality figure
vrfigure.capture	Capture virtual reality figure image
vrfigure.close	Close virtual reality figure
vrfigure.get	Return property value of vrfigure object
vrfigure.isvalid	Check validity of vrfigure object handles
vrfigure.set	Set property values of vrfigure object
vrgcf	Handle for active virtual reality figure
vrgcbf	Current callback vrfigure object
vrclose	Close virtual reality figure windows
vr.utils.stereo3d	Stereoscopic vision settings for vr.canvas and vr.figure objects
vrdir2ori	Convert viewpoint direction to orientation
vrori2dir	Convert viewpoint orientation to direction
vrrotvec	Calculate rotation between two vectors

Figura 19. *Lista de las funciones más importantes para la manipulación del modelo 3D. Es.mathworks.com. (2017). Simulink 3D Animation Documentation - MathWorks United Kingdom. [online] Available at: https://es.mathworks.com/help/sl3d/functionlist.html [Accessed 8 Jul. 2017].*

Por último, en la fase de inicialización se hace uso de la función *vr.canvas* para insertar el modelo 3D dentro del GUI. Al principio, al no saber la existencia de esta función se creaba el modelo en una ventana aparte y se tenían que regular manualmente los tamaños de ambas ventanas para que no se solaparan. Gracias a esta función, se puede elegir tanto el tamaño como la posición del modelo 3D en nuestra interfaz. Una vez que se haya inicializado el modelo, se puede también elegir con que vista se empezará a ver el brazo robótico, en este caso se eligió la vista frontal.

A continuación, se describe los diferentes botones insertados en la interfaz gráfica. Referenciando la *Figura 19*, se empezará por los botones de la ventana de comandos.

- Conectar. Con este botón se invoca a la función *EstablecerConexion ()* explicada anteriormente y, una vez pulsado, cambia de color y descripción la caja de texto de su lado, dibujándose de verde y poniendo CONECTADO.
- Desconectar. Al pulsar sobre este botón, se invoca la función CerrarCOM (), el cual se encarga de encontrar todos los puertos series visibles y ocultos y los elimina. Además, actúa sobre el texto estático escribiendo DESCONECTADO y dibujándose en rojo.
- Iniciar simulación. Este botón ejecuta la función principal de nuestra interfaz. Pulsándolo, se inicializa la lectura de datos por parte del PC del estado de los servos. El funcionamiento consta de:
    - Se realiza la lectura de datos con la función *LecturaDatos(objeto_puerto_serie)* del puerto serie.
    - A continuación, se debe de hacer una transformación de ángulos, puesto que no coincide el sistema de referencia del PhantomX Reactor con el del modelo 3D. Esta transformación se explicará en un punto aparte, más adelante.
    - Se crea para cada uno de los servos, un nodo, de la misma manera que se hará posteriormente para las pinzas, con el comando *vrnode*. Se especifica el modelo 3D creado y también el nombre de la transformación creado en dicho modelo 3D
    - Para cada uno de los nodos creado, se ejecuta la operación *rotation* especificando sobre que eje rota cada nodo, como se puede ver en la siguiente figura.

```
Nodo1.rotation = [0 1 0 VectorVLMR(1)];
Nodo2.rotation = [0 0 1 VectorVLMR(2)];
Nodo3.rotation = [0 0 1 VectorVLMR(3)];
Nodo4.rotation = [0 0 1 VectorVLMR(4)];
Nodo5.rotation = [1 0 0 VectorVLMR(5)];
```

Figura 20. *Configuración de la rotación de cada una de las articulaciones de nuestro modelo 3D.*

Donde VectorVLMR es el ángulo recibido del robot físico aplicándole las transformaciones oportunas para que concuerde con los ángulos en el sistema de referencia del modelo 3D.

- Por último, cada uno de los ángulos obtenidos y procesados se insertan en las casillas correspondiente a cada uno de los ángulos.

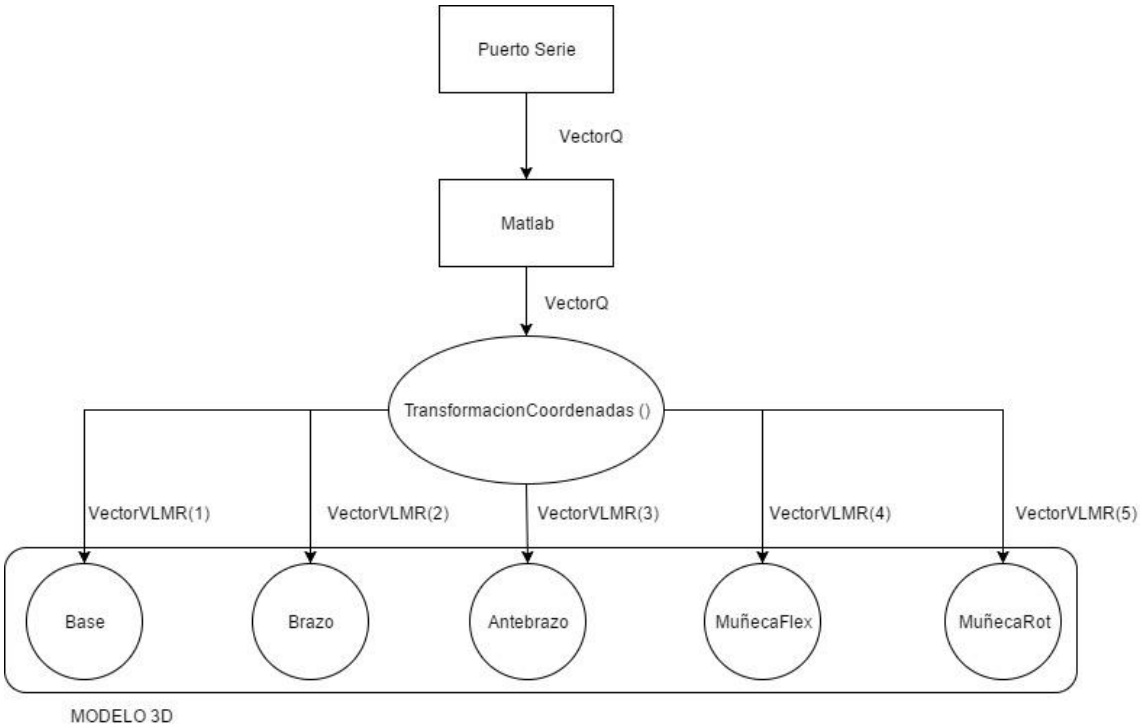

Figura 21. *Funcionamiento de la lectura y asignación de los estados de los servos*

- Abrir pinza. Por un lado, cuando se presiona este botón se escribe la cadena [OPEN] que en Arduino ejecuta la función GripperOpen () que se encarga de abrir la pinza. Por otro lado, para actuar sobre el modelo 3D de manera similar es un poco más complicado. Su función se definirá en diferentes pasos:
    - Primero de todo, se crea un bucle que recorra la variable *i* que vaya desde la posición en la que está la pinza cerrada, hasta que se abre por completo.
    - Se crea el nodo con la función *vrnode*, donde asigna el nodo creado en el V-Realm Builder a una variable en Matlab.
    - Una vez el nodo esté creado, se hace diferentes operaciones sobre él, como la traslación de la pinza, en la cual se mantiene las coordenadas X e Y constantes y se va variando el eje Z que es sobre el eje que se desliza la pinza.
    - Del mismo modo, se hace sobre la otra pinza, pero en dirección contraria, es decir, en vez de insertar *i* en el eje Z, se insertará *-i*.
- Cerrar pinza. Ocurre lo mismo que en el botón abrir pinza, pero el bucle se ejecuta, al contrario, es decir, tiene que ir en decremento, desde la posición de las pinzas cuando están completamente abiertas hasta que se cierran.
- Relajar servos. Al pulsar este botón, se escribe la cadena [RELA] y se manda por el puerto serie, para que el Arbotix le mande dicha orden al robot.
- Fijar servos. Al pulsar este botón, se escribe la cadena [FIJA] y se manda por el puerto serie, para que el Arbotix le mande dicha orden al robot.
- Home. Al pulsar este botón, se escribe la cadena [HOME] que ejecuta la función en Arduino, la cual manda al robot a una posición previamente definida.
- Test. Al pulsar este botón, se manda la cadena [TEST] al puerto serie para ejecutar el test del robot. Que es una función implementada en el Arbotix que explicará más adelante.

En la cajetilla de la cinemática inversa:

- Obtener. Con este botón, se asigna los valores de cada servo como parámetros de entrada a la función cinemática directa, que se encarga de obtener las coordenadas XYZ en las que se encuentra la punta de la pinza del robot para dichos valores de los servos. Finalmente, los valores obtenidos se escriben en las casillas de las coordenadas X,Y y Z, redondeando a 3 decimales.
- Calcular. Al pulsar este botón, se asigna los valores introducidos en cada una de las casillas de cinemática inversa a una variable y la pasa de String a Double, puesto que desde el GUI todos los valores que se obtienen son en formato String. Estas variables obtenidas se pasan como parámetro de entrada a la función de cinemática inversa, se obtiene de este modo un vector con los 5 valores en los que deberían de estar los servos para que estuviera en dicha posición y orientación. Este vector, se pasa como parámetro de entrada a la función *MoverBrazo(vQ)* que se encarga de concatenar distintos caracteres y cadenas para que al enviarlo por el puerto serie sea reconocido por la placa Arbotix, de manera que [MOVE, Q1,Q2,Q3,Q4,Q5] es el formato utilizado para que la placa reconozca que se quiera llevar a cabo dicha función. En la siguiente figura se facilita un esquema de funcionamiento de este botón.

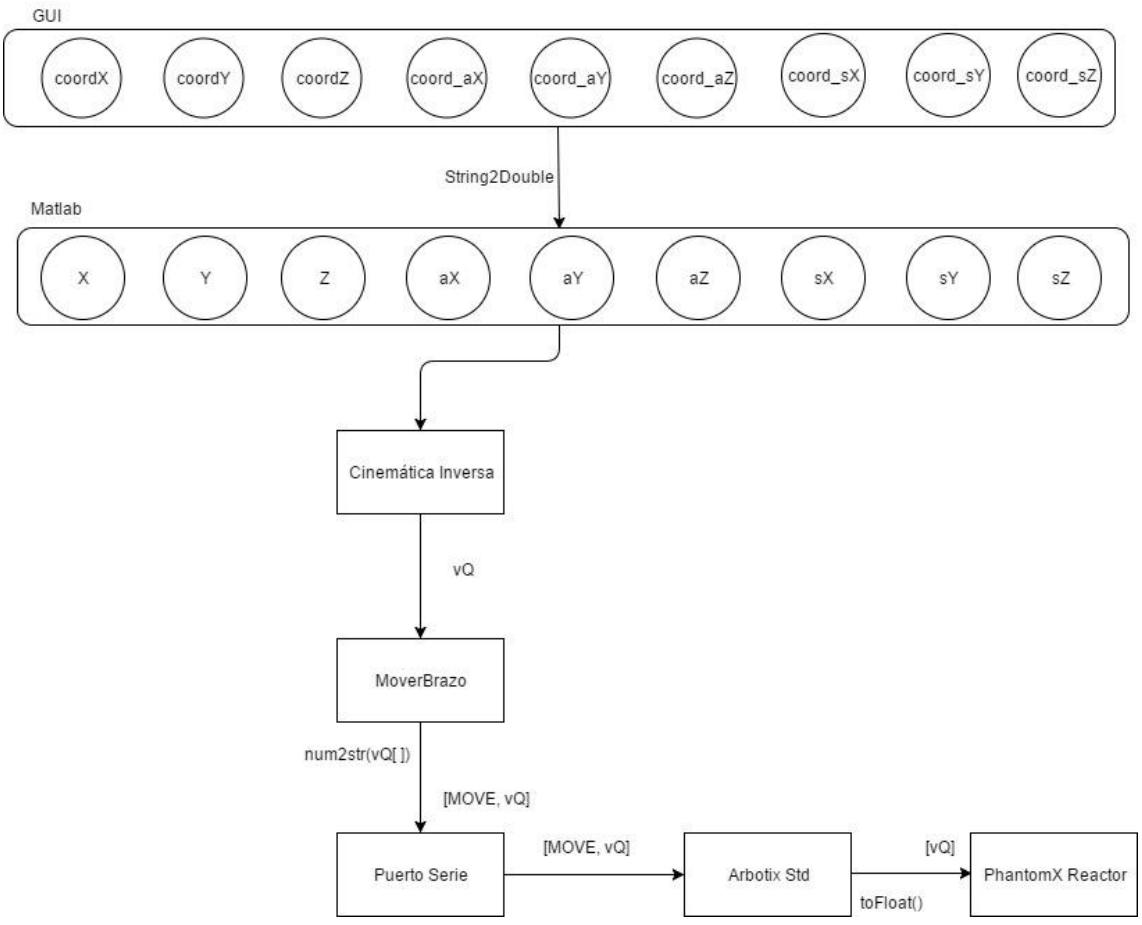

Figura 22. *Proceso al ejecutar la cinemática inversa*

La cajetilla de los sliders:

- SliderQ1-Q5. Con los sliders se modifica el ángulo de cada uno de los servos según el slider que se actúe. Al mover un slider se obtiene el valor de dicho slider en una variable y con esa variable se va moviendo el servo indicado, mientras que los demás ángulos están intactos. Finalmente, la variación de dicho servo se manda a la placa Arbotix Std mediante la función mencionada anteriormente, *MoverBrazo (var)*.
- Posicionar. Con este botón se actualizará la posición de los Slider según la posición en la que se encuentre los servos en ese momento. De este modo, se evita que, al mover el robot con otros botones, no haga movimientos inesperados al mover los Sliders desactualizados.

Para concluir:

- Salir. Con este botón se genera un cuadro de dialogo, gracias a la función *questdlg* con la cual se puede crear el texto y las diferentes opciones a escoger. Si se presiona en 'Si', se cierra definitivamente la interfaz y si se presiona en 'No', vuelve a la interfaz
- Frontal/Superior. Estos botones cambian el punto de visión del modelo 3D gracias a un método de VR, *Viewpoint*.

Hay que tener en cuenta algunas cosas para que el programa funcione correctamente, una vez se simuló se apreció que el sistema de referencia del robot físico no coincidía con el del modelo 3D, puesto que, al leer los valores de los servos y mandarlo a nuestro modelo, éste se salía del límite físico establecido. Entonces se creó una función que se encargaba de transformar los ángulos de un sistema a otro (TransformarCoordenadas()) el cual se basa en la ecuación de la recta para la transformación debido a que si se tiene la relación que existe entre ambos sistemas de referencia (m en la ecuación 9.1) se puede transformar los ángulos.

$$y = mx + n \qquad 9.1$$

De este modo, se obtuvo un par de valores de cada articulación (del robot físico mediante el puerto serie del Arduino y del modelo 3D mediante el V-Realm Builder probando rotaciones) y se insertó en la función como parámetro de entrada, recibiendo como salida la relación que existe entre ellos en cada articulación, puesto que está divido en ejes locales y no todos son el mismo.

Finalmente, a la hora de operar con ellos, el vector proporcionado por los servos fue multiplicado matricialmente por cada constante multiplicativa según fuera el servo.

Otro de los problemas que se produjo en la simulación, fue la saturación que se generaba en el puerto serie. El método que nos ha sido muy útil para borrar el buffer cuando el puerto estaba saturado fue flushinput (), el cual fue utilizado en la lectura de datos.

10.4 Arduino IDE

En este apartado se explicar cómo se realiza la lectura de datos en el puerto serie y cómo se envían los datos necesarios para la monitorización de los servos.

10.4.1 Envío de datos

Desde la placa Arbotix hasta el ordenador, se tienen que enviar los datos de la posición de los servos. De esta manera, se monitoriza el estado de los servos y se puede pasar como entrada al modelo virtual creado.

Para ello se utiliza la función *ROBOT_GetJointPos* () disponible en las librerías, la cual se encarga de llamar a otra función llamada *SERVOS_GetServosPos ()* en un bucle que pasa por todos los servos.

Esta última función se encarga de leer los datos de un registro especifico de un solo servo, cómo parámetros de entrada se insertan la ID del servo, el registro desde donde tiene que empezar a leer y la longitud del registro a leer.

Por tanto, con un bucle se lee el estado de todos los servos, tal y como hace la función previamente mencionada *SERVOS_GetServosPos ().*

Después de invocar a la función, se obtendría el estado de cada uno de los servos, pero quedaría enviarlo por el puerto serie para poder trabajar con ellos en Matlab. Con ese fin, se imprime el estado de cada servo con la función *Serial.print()* dándole el formato deseado para la lectura en Matlab, en este caso, sería formato vector [Q1, Q2, Q3, Q4, Q5] siendo Qi el ángulo en radianes de cada servo.

10.4.2 Recepción de datos

Para la recepción de datos por parte de la placa, se enviará a la placa la información deseada según el botón que se haya pulsado en la interfaz. De esta manera, se decide hacer en forma de vectores, en el cual la primera posición corresponde a un *String* a modo de bandera con el fin de que dependiendo de la bandera que lea en el Arduino IDE, entraría en una condición u otra, como se puede ver en la *Figura 23*. Cabe destacar que además de la recepción de datos en la tarjeta, se ocupa de escribirlo en el puerto serie para enviarle las diferentes órdenes al robot.

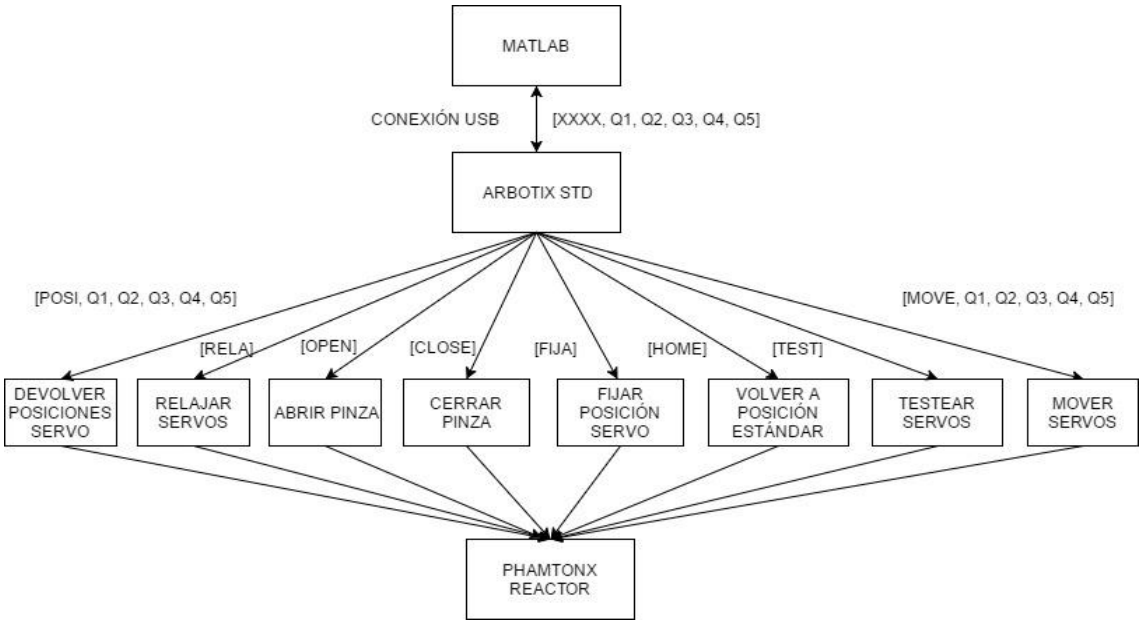

Figura 23. *Esquema de funcionamiento de recepción de datos en Arbotix Std*

A continuación, se describirá que hace cada una de las ordenes de izquierda a derecha:

- Relajar los servos. Cuando se ejecuta esta orden, los servos se relajan gracias a la llamada de la función *SERVOS_ServosOff* que a su vez llama a la función *dxlTorqueOffAll ()* que es una función de la librería de *trossen robotics* aplicable al microcontrolador Arbotix Std, con la condición de que los servos sean compatibles, en este caso, sirve para los servos AX que son los que están ensamblados en el robot. Esta función utiliza la dirección de transmisión para desactivar el par a cada uno de los servos conectados.
- Abrir pinzas. Con esta orden, las pinzas del brazo se abrirán. Se ejecuta la función *SERVOS_MoveGripper (pos)* en el que se debe de especificar el ID del servo donde se pretende actuar, y como parámetro de entrada *pos* se ingresa la constante 512, que se refiere a la apertura de las pinzas. Si la posición actual coincide con la orden, dada, no afecta en absoluto al robot.
- Cerrar pinzas. Con esta orden, las pinzas del brazo se abrirán. Se ejecuta la función *SERVOS_MoveGripper (pos)* en el que se debe de especificar el ID del servo donde se quiere actuar, y como parámetro de entrada *pos* se ingresa la constante 140,
  que se refiere al cierre casi por completo de las pinzas. Si la posición actual coincide con la orden, dada, no afecta en absoluto al robot.
- Fijar los servos. Cuando se ejecuta esta orden, los servos se quedan fijos gracias a la invocación de la función *SERVOS_ServosOn* que trabaja de la misma forma que en el caso anterior, usa la dirección de transmisión para activar el par de cada uno de los servos conectados.
- Volver a la posición inicial. Si se ejecuta esta orden el robot volverá al estado inicial previamente definido. Se ejecutará la función creada *Home ()* en la que se definirá un vector con los ángulos de cada servo para que el robot esté en la posición inicial. Posteriormente se invoca la función *ROBOT_SetSingleTrajectory (pos_ini, time, trajectory_type)* en el que el robot llevará a cabo la trayectoria entre el punto en el que se encuentra y la posición de inicio (*pos_ini*), en un tiempo especificado en milisegundos (*time)* y con un tipo de trayectoria determinado, ya puede ser LINEAL (0) o CUBIC (1).

- Testear cada uno de los servos. Para testear todos los servos se hace de la siguiente manera:
    - Se dibuja una trayectoria del mismo modo descrito anteriormente *ROBOT_SetSingleTrajectory (pos_ini, time, trajectory_type).*
    - Posteriormente, se hace un par de bucles, en el que uno de ellos recorrerá los diferentes servos y el otro los *steps* para que el robot haga el movimiento. De este modo, dentro de los bucles se le irá pasando diferentes posiciones a los servos para que lo vayan testeando.
    - Con lo anterior, se revisa el movimiento de todos los servos exceptos el de las pinzas, que simplemente se tiene que invocar las funciones descritas anteriormente para finalizar la operación.
- Mover cada uno de los servos. Como se describió anteriormente, la condición para que entre dentro de la rutina, en este caso, es que el vector contenga la orden MOVE. Una vez leído el primer *String*, entra en la condición actual de mover los servos y pasa a leer los siguientes datos. Desde Matlab, se envía los datos como *String*, de esta manera con una simple función se captura los diferentes S*tring* del vector (con formato [MOVE, Q1, Q2, Q3, Q4, Q5]) poniéndole como separador las comas.

  Aunque antes de mandarlo por el puerto serie para actuar sobre los servos, se vuelve a pasar a Float para que éstos reaccionen a los datos recibidos. Una vez pasados a Float, se insertan como vector en la función anteriormente descrita *ROBOT_SetSingleTrajectory (pos_act, time, trajectory_type),* especificando también el tiempo que transcurre desde que empieza el movimiento en el punto en el que esté hasta que lo finaliza en el punto de interés y el tipo de trayectoria que se pretende que siga.

11. Conclusiones

Con este trabajo se obtiene una útil herramienta para la simulación del Robot PhantomX Reactor en un entorno virtual 3D y en tiempo real. Uno de los principales aspectos ha sido la importancia de la integración de diferentes tecnologías y ámbitos de programación para llegar a un fin común. La irrupción de los simuladores para simular distintos escenarios con un determinado robot es muy importante para entender su funcionamiento y hacer mejoras en diferentes aspectos que se observen. Por otro lado, la inserción de los simuladores junto a los robots en el área médica es un hecho y los beneficios que obtienen tanto los cirujanos como los pacientes son numerosos, por ende, es normal que la robótica esté empezando a ser el presente y futuro en esta área, sobre todo en el ámbito quirúrgico.

Para este TFM se usaron diferentes softwares para la implementación de diferentes módulos en los que se dividió el proyecto, de modo que, conforme se realizaba el proyecto, se iba testeando la comunicación entre los diferentes módulos para comprobar el correcto funcionamiento del sistema. Fue tan importante la implementación de los diferentes módulos, como la comunicación existente entre ellos, para que cada módulo aportara la información pertinente a los demás.

Se logró desarrollar una interfaz de usuario de fácil manejo y correcto funcionamiento gracias a la funcionalidad de la herramienta GUI de Matlab. Además de la integración con el entorno 3D, que representa las características del robot, lo que constituye una muy interesante herramienta educativa e investigativa.

Además, se consiguió que los objetos creados en cada herramienta de Matlab y las funciones creadas en este mismo programa, tuviera un nivel de encapsulamiento de datos y funciones necesario para la posterior incorporación de diferentes mejoras o nuevos dispositivos sin afectar el funcionamiento de los demás.

Con el software y la interfaz creada durante la realización de este proyecto se podrían manejar numerosos robots manipuladores haciendo pequeños cambios en la arquitectura del software y ser implementado en, por ejemplo, un robot quirúrgico. Aunque hay que tener en cuenta que ni el robot de laboratorio tiene las especificaciones necesarias para ello ni el software realizado es lo suficientemente robusto y exacto que debería ser para desempeñar esta importante función.

12. Líneas futuras

A pesar de cumplir los objetivos propuesto inicialmente para el trabajo, hay varios aspectos que se pueden mejorar de cara al futuro o continuar el desarrollo implementando nuevas funcionalidades o modificando las ya existentes.

- Se puede llegar a simular sistemas mecánicos más complejos que los robots manipuladores, como por ejemplo una mano robótica que incluye un mayor número de articulaciones, o simular varios robots manipuladores en paralelo.
- El trabajo se centró en la herramienta de simulación por lo que no se realizó un análisis más profundizado de la eficiencia del código para la actuación del robot y el simulador 3D a las diferentes tareas que se le asigne mediante la interfaz.
- Con el fin de hacer el manejo del robot más amigable, se puede incorporar un joystick o un mando de consola para generar comandos de posición sobre el robot, eligiendo la articulación a mover y el movimiento deseado.
- Se puede cambiar la manera de comunicarse el ordenador con el robot, en vez de por USB por vía Bluetooth, con un XBee [21], de este modo también se puede añadir otros dispositivos como una Webcam, para trabajar con una visión estéreo

13. Bibliografía

[1] Platea.pntic.mec.es. (2017). 5.1 ¿Qué es un robot? [online] Available at: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.1.htm

[2] Ollero Baturone, A. (2007). Robótica; manipuladores y robots móviles. España: Marcombo. PP. 2

[3] Iñigo-Madrigal, R y Vidal-Idiarte, E. Robots industriales manipuladores, Ed Maracambo editores, PP. 26, 2002

[4] Dormido-Bencomo, S. y Segado-Bernal, A, Robótica Industrial; Tecnología, programación y aplicaciones, Ed. McGraw-Hill, pp 405-515, 1990.

[5] neurocirugía a las neurociencias, Ed. Academia Nacional de Medicina, PP. 273, 2004.

[6] Laparoscopica.es. (2017). ¿Cuáles son las ventajas de la cirugía robótica? | Laparoscópica.es. [online] Available at: http://www.laparoscopica.es/robotica

[7] Es.mathworks.com. (2017). MATLAB frente a R - MATLAB & Simulink. [online] Available at: https://es.mathworks.com/discovery/matlab-vs-r.html

[8] Granat, K. (2017). Arbotix 1.6 Documentation. [online] Learn.trossenrobotics.com. Available at: http://learn.trossenrobotics.com/projects/181-arbotix-1-6-documentation.html

[9] Gutiérrez, Á. and Monasterio, F. (2017). Cinemática. [Apuntes] Madrid. Available at: robolabo.etsit.upm.es

[10] Informaticamoderna.com. (2017). Puerto USB 3.0, 2.0 y 1, características y capacidades. www.informaticamoderna.com :: [online] Available at: http://www.informaticamoderna.com/El_puerto_USB.htm [Accessed 8 Jul. 2017].

[11] Granat. K. (2017). Dynamixel AX-12 Documentation. [online] Learn.trossenrobotics.com. Available at: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

[12] Processing.org. (2017). Processing.org. [online] Available at: https://processing.org/

[13] Arduino.cc. (2017). Arduino - Environment. [online] Available at: https://www.arduino.cc/en/Guide/Environment

[14] Wiki.robolabo.etsit.upm.es. (2017). PhantomX Reactor Robot - robolaboWiki. [online] Available at: http://wiki.robolabo.etsit.upm.es/index.php/PhantomX_Reactor_Robot

[15]Arduino.cc. (2017). Arduino - Home. [online] Available at: https://www.arduino.cc/

[16]Es.mathworks.com. (2017). MathWorks - Makers of MATLAB and Simulink. [online] Available at: https://es.mathworks.com/

[17]Bibing. (2010). Memoria Descriptiva La escena 3D. [online] Available at: http://bibing.us.es/proyectos/abreproy/4577/fichero/PARTE+I+-+MEMORIA+DESCRIPTIVA%252F05+Cap%C3%ADtulo+5+-+La+escena+3D.pdf

[18]http://cda.psych.uiuc.edu. (2004). Virtual Reality Toolbox. [online] Available at: http://cda.psych.uiuc.edu/matlab_pdf/vr.pdf

[19]Barragán, D. (2012). Manual Interfaz Gráfica Usuario de Matlab. [online] www.dspace.espol.edu.ec. Available at: Diego Orlando Barragan. (2012). Manual Interfaz Gráfica Usuario de Matlab. 2012, de Matpic Sitio web: https://www.dspace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf

[20]Tesis.ipn.mx. (2017). Bitstream – Implementación brazo robótico. [online] Available at: http://tesis.ipn.mx/bitstream/handle/123456789/4047/BRAZOROBOTICO.pdf?sequence=1

[21]XBee.cl. (2017). ¿Qué es XBee? ~ XBee.cl. [online] Available at: http://xbee.cl/que-es-xbee/

14. ANEXO I. MANUAL DE USUARIO ARDUINO IDE.

La interfaz del software de Arduino es el siguiente:

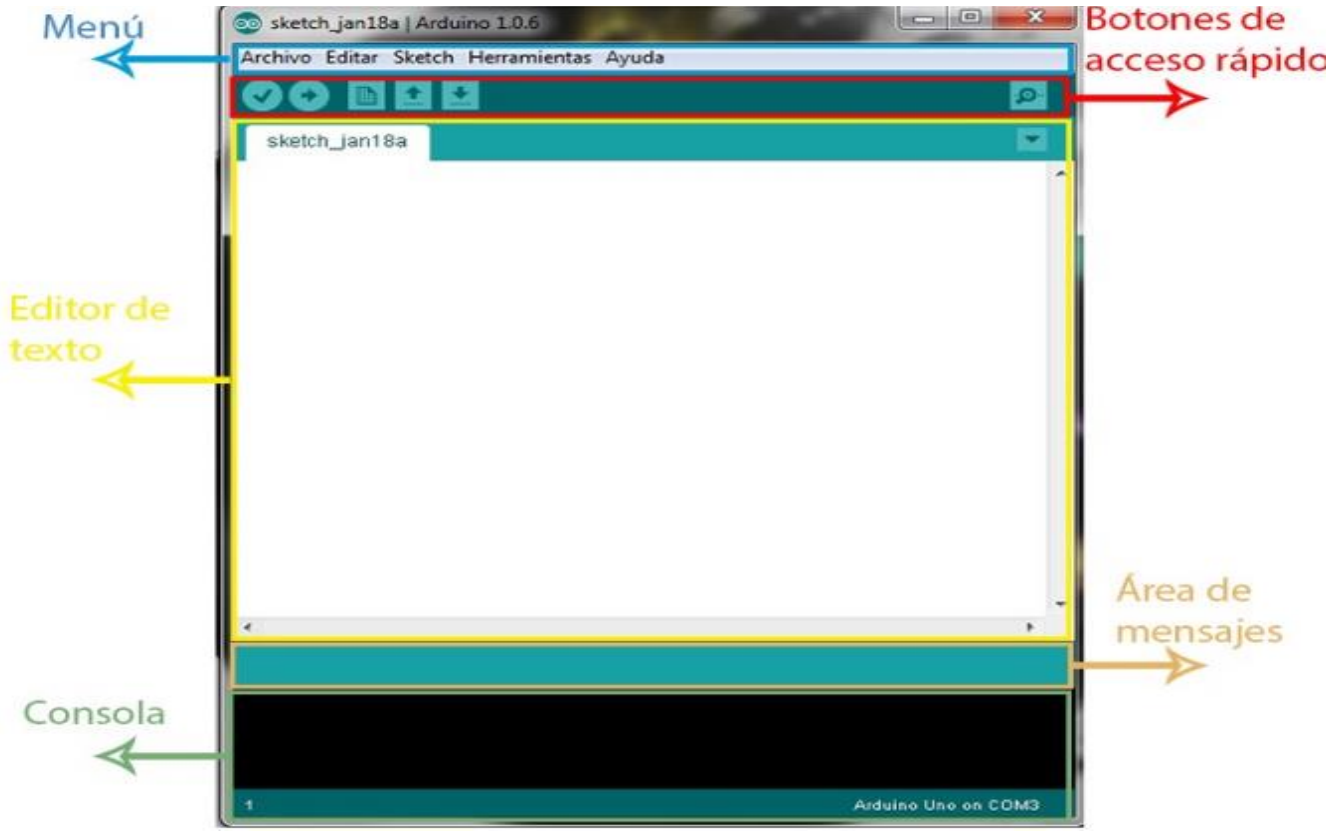

Figura 24. *Interfaz de IDE de Arduino*

En la parte de menú, hay una zona para acceder a funciones de carga de archivos, edición de texto del código, carga de librerías… etc.

En los botones de acceso rápido están los siguientes iconos:

Verificar que el programa está bien escrito y puede funcionar

Carga el programa a la placa

Crea un programa nuevo

Abre un programa

Guarda el programa

Monitor Serial: abre una ventana de comunicación con la placa de Arduino en la que se puede ver las respuestas que nuestro Arduino nos está dando, siempre y cuando esté el USB conectado.

En el cuadro del editor de texto se escribirá el código a ejecutar por el *Arbotix Std.*

Finalmente, en el área de mensajes y la consola Arduino irá dando información sobre las acciones que lleva a cabo la placa (compilar, verificar, guardar…) y sobre los fallos o errores que se produzcan en el condigo como en el propio IDE.

Para empezar a programar en la placa, se debe configurar nuestro IDE para que se comunique con la placa Arbotix-M robocontroller. Para ello, se conectará el Arbotix mediante cable USB al PC y después de instalar librerías y realizar test de funcionamiento se podrá seleccionar dicha placa, se pulsa sobre *Herramientas/Placa* y se selecciona la placa que se esté utilizando, en este caso, Arbotix Std.

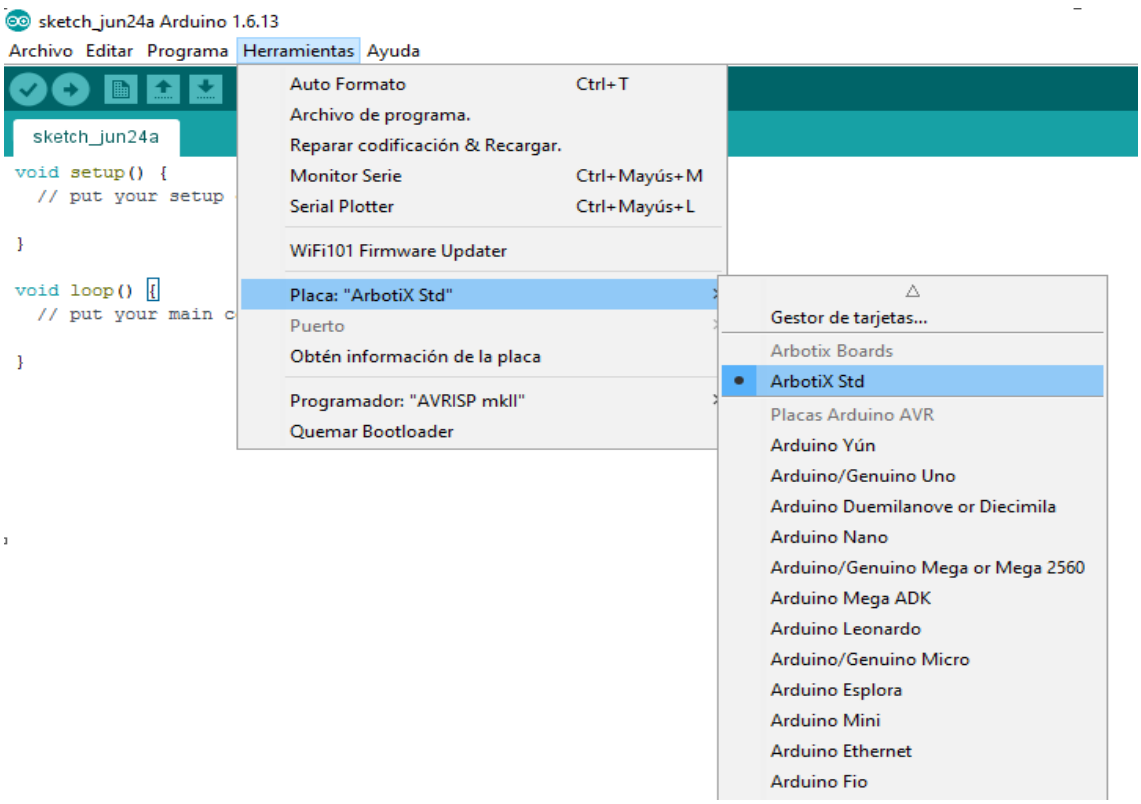

Figura 25. *Elección de placa en IDE*

Para la configuración del puerto de comunicación se debe de entrar en *Herramientas/Puerto* y elegir el puerto de comunicación en la que está conectada la placa.

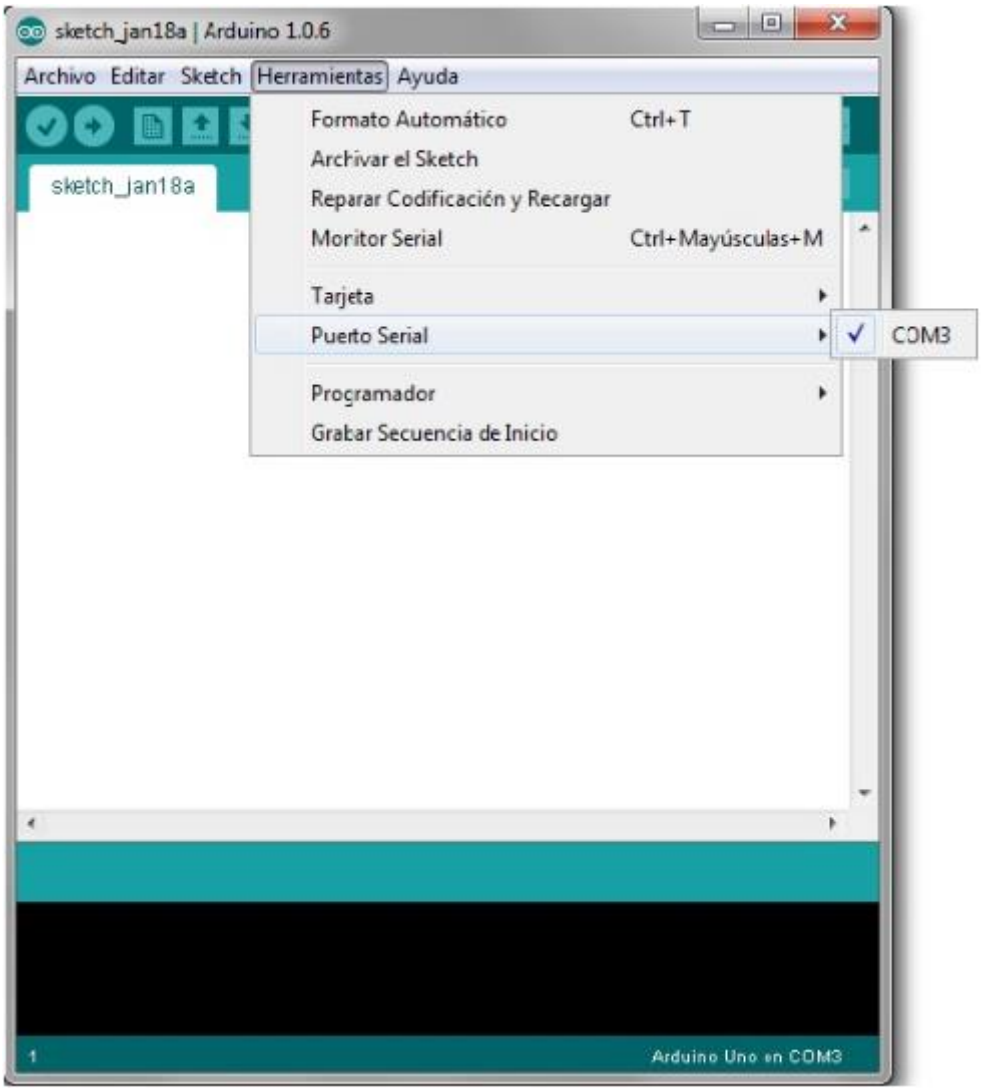

Figura 26. *Selección de puerto COM*

Si aparecieran varios COM activos, para saber cuál de ellos es el que se comunica con la placa, solo hay que buscar *Panel de control/Hardware/Administrador de dispositivos.* Se busca la pestaña (Puertos COM y LPT) y aparecerá el *Arbotix Std* y el COM en el que está conectado. Con esto, ya se puede empezar a programar la placa.