Trabajo de Fin de Grado

Grado en Ingeniería Informática

Visualización en 3D y control con interfaces naturales de un manipulador robótico

Robotic arm control with natural interfaces and 3D visualization

David Fernando Redondo Durand

La Laguna, 5 de Septiembre de 2017

D. **Rafael Arnay del Arco**, con N.I.F. 78.569.591-G profesor Ayudante Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

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

Que la presente memoria titulada:

"Visualización en 3D y control con interfaces naturales de un manipulador robótico."

ha sido realizada bajo su dirección por D. **David Fernando Redondo Durand**, con N.I.F. 45.899.961-B.

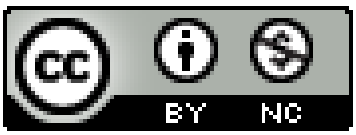
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de Septiembre de 2017.

Agradecimientos

En primer lugar agradecer a toda mi familia y a mis padres por el apoyo que me han dado durante todo el transcurso de la carrera. En especial quiero darle las gracias a mi padre por haberme inculcado una filosofía de aprendizaje continuo y haberme introducido en el mundo científico.

En segundo lugar agradecer a mi tutor académico Rafael Arnay del Arco por brindarme su tiempo siempre que lo necesité, motivándome durante el transcurso de la asignatura de robótica para que presentase mi propia propuesta para este trabajo y estar continuamente atento sobre la progresión del trabajo.

Licencia



Resumen

El objetivo de este proyecto consiste en el desarrollo de una aplicación que permita la visualización y control de un brazo robótico mediante una interfaz externa. La interfaz a utilizar será el sensor de gestos ArmBand Myo. Con este sensor en forma de pulsera, se puede detectar tanto la posición del brazo del usuario como una serie de gestos. Esos movimientos y gestos se verán reflejados en un modelo 3D de un brazo robótico en primera instancia y, posteriormente en un brazo robótico real llamado SCORBOT-ER9.

Palabras clave: *armband, myo, interfaz, natural, 3D, robot, scortbot, er9, unity*

Abstract

The objective of this project is to develop an application that allows the control and visualization of a robotic arm through an external interface. This interface will be the wearable gesture control and motion control MyoArmband. This device in the shape of a bracelet, is capable to detect the position of the user's arm and a series of gestures. At first, these movements and gestures will be reflected in a 3D robotic arm model, and then in a real robotic arm called SCORBOT-ER9.

Keywords: *armband, myo, interfaz, natural, 3D, robot, scortbot, er9, unity.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este apartado se hablará sobre el estado y la situación en la que se encuentra ambientado este Trabajo de Fin de Grado, además de sus objetivos principales y la motivación que llevó a hacerlo posible.

1.1 Estado actual

Muchas empresas están invirtiendo millones en crear sistemas que puedan desarrollar su propia lógica, y otras muchas en crear dispositivos que permitan amplificar los sentidos o las capacidades físicas de la persona que lo utiliza, ya sea sobre su propio cuerpo o remotamente.

Un ejemplo que se ha visto desde hace años pero en el cual el mercado sigue reinventándose, es el de las prótesis avanzadas que permite que los discapacitados puedan obtener diferentes capacidades, las cuales nunca han tenido o perdieron en un momento [2]. Una de las personas que ha hecho grandes avances en este sector y es una gran influencia a nivel mundial es Hugh Herr [1], escalador e ingeniero y profesor de biofísica estadounidense, el cual posee discapacidad en las dos piernas.

El otro caso, y es en el cual se basa este trabajo es el del control remoto de diversos sistemas mecánicos sobre los que una persona actúa directamente. Un ejemplo de estos sistemas pueden ser los diversos robots que utilizan actualmente los ejércitos, los cuales sirven para realizar misiones arriesgadas sin poner en peligro vidas humanas [3]. Los dispositivos que se suelen controlar son vehículos para la desactivación de bombas o reconocimiento de zonas peligrosas, drones de reconocimiento y a veces incluso aviones cargados con munición.

Debido a la complejidad que conlleva desarrollar un sistema que permita el control de una máquina de forma rápida y precisa a través de interfaces controladas por personas, se han utilizado periféricos sencillos que permiten obtener una respuesta equivalente al grado de precisión-rapidez que se requiera.

Una de las interfaces más usadas son los joysticks, los cuales utilizan desde hace décadas para este fin, pero con el inconveniente de que los movimientos que se realizan son pocos naturales.

1.2 Objetivos

Este Trabajo de Fin de Grado persigue integrar una interfaz que permita al usuario controlar un brazo robótico de forma remota y llevando a cabo movimientos lo más intuitivos posibles.

El objetivo de este Trabajo se puede desglosar en cuatro sub-objetivos.

1. Desarrollar un entorno que permita la visualización del brazo en 3D.
2. Crear una comunicación entre la interfaz externa y dicho programa de forma que se mueva el brazo de la forma más intuitiva posible.
3. Comunicar el programa con un brazo real de forma offline y que adopte la posición de este.
4. Permitir que el programa se comunique de forma online con el brazo.

1.3 Motivación

La principal motivación que mueve el desarrollo de este Trabajo de Fin de Grado es el de crear un sistema con el que sea más sencillo comprender los principios de la robótica en el entorno educativo, viendo así de una forma más intuitiva y natural los conceptos básicos de la cinemática directa e inversa.

Capítulo 2.
Tecnologías Utilizadas

Para el desarrollo de este proyecto se utilizaron tecnologías abiertas - en la medida de lo posible -, de manera que permitiera una personalización lo más amplia posible.

En los siguientes apartados, éstas se mostrarán siguiendo la secuencia de comunicación que se planeó:

Interfaz → Modelo 3D → Brazo real

2.1 Interfaz MyoArmband

A la hora de elegir la interfaz, se decidió optar por una intuitiva, que permitiera obtener los movimientos de la forma más precisa y con la menor latencia posible.

Se optó por utilizar el brazalete MyoArmband [7] de la empresa ThalmicLabs™ [6].

Figura 2.1. MyoArmband

Este brazalete es básicamente una combinación de sensores unidos por una estructura extensible que permite la adaptación para cualquier persona.

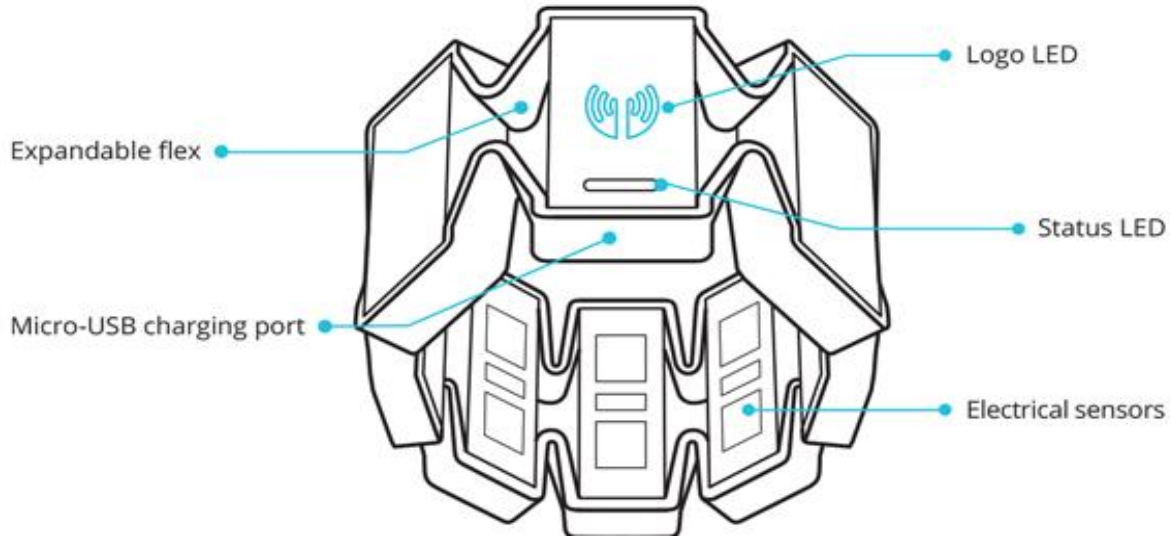


Figura 2.2. Esquema MyoArmband

2.1.1 Sensores

A pesar de la aparente simplicidad de este dispositivo, éste dispone de una amplia gama de sensores que permiten que tenga una gran precisión, tanto a la hora de calcular la rotación del brazo como a la hora de obtener señales de los músculos sobre los que la persona está actuando.

- **Magnetómetro**: Permite situar la rotación respecto a la de la tierra
- **Acelerómetro**: Permite saber la aceleración lineal.
- **Giroscopio**: Permite saber la aceleración angular.
- **Sensores eléctricos**: Permite obtener los gestos que se realizan mediante electromiografía

2.1.2 Software

Dado que éste es un dispositivo no solo apto para el uso cotidiano, ThalmicLabs™ nos provee de un SDK que permite hacer uso de sus librerías que permite a los desarrolladores darle nuevos usos de una manera más simple.

La parte más interesante de este SDK es en la cual se realizan todos los cálculos para convertir los datos crudos que obtiene de los sensores en datos más fácilmente tratables por un desarrollador.

Básicamente aquí es donde se unen los datos del magnetómetro y giroscopio para obtener la rotación del dispositivo. También se cruzan todos los datos de los sensores eléctricos y se decide qué entrada se corresponde con qué movimiento.

Otra parte fundamental de este código es el que nos permite obtener estos datos de la forma más simple posible, con el inconveniente de que no se encuentra accesible desde cualquier lenguaje y/o framework.

2.1.3 Comunicación

Con respecto a este apartado, el dispositivo hace uso Bluetooth 4.0 para comunicarse con todos los dispositivos con los que es compatible, pudiendo utilizarse tanto el adaptador del cual nos provee la compañía - destinado a ordenadores -, como el bluetooth de nuestro dispositivo móvil.

2.2 Unity3D

Unity3D [4] es un motor de videojuegos multiplataforma desarrollado por Unity Technologies este se utilizó para la creación del modelo 3D e implementar la comunicación con el brazo real.

Unity posee una amplia comunidad que se está reinventando continuamente y comparte librerías que pueden llegar a reducir drásticamente el tiempo de desarrollo. Aparte de esto, utiliza un gran motor gráfico y físico, el cual es utilizado por muchísimas empresas dedicadas al desarrollo de videojuegos.

2.2.1 Lenguajes de programación

Unity utiliza tanto C# como Java para el desarrollo de scripts que permiten el control de los elementos que componen el proyecto. Dado que para utilizarlo es necesaria la instalación de Visual Studio, éste tiene integrado .NET Framework [13], que permite utilizar librerías que comunican directamente con el sistema operativo.

2.3 SCORBOT ER-9

Como brazo robótico se utilizó el SCORBOT-ER9 [9], que posee cinco articulaciones y un efector final intercambiable. Es un brazo ampliamente utilizado en el mundo de la enseñanza debido a que es un hardware muy polivalente y fácil de controlar en comparación con otros brazos similares.

El control de este brazo se hace a través de un puerto serie, el cual, mediante el lenguaje de programación ACL permite la comunicación con un ordenador.

Figura 2.3. SCORBOR ER9

2.3.1 Especificaciones técnicas

Especificaciones	
Número de articulaciones	5 más la pinza
Límites de las articulaciones	• Base: 270º • Hombro: 145º • Codo: 210º • Alabeo muñeca : 196º • Cabeceo muñeca : 737º
Velocidades efectivas y máximas	• Base: 80º/s, 140º/s • Hombro: 69º/s, 123º/s • Codo: 77º/s, 140º/s • Alabeo muñeca : 103º/s, 166º/s • Cabeceo muñeca: 175º/s, 300º/s
Rango de operación máximo	691mm (27,2”) sin la pinza
Opciones del efector final	Pinza neumática
	Pinza eléctrica
Guiado	Switch óptico y encoder
Carga máxima	• 5kg con aceleración reducida • 2kg con velocidad máxima
Precisión	±0.05mm
Peso	53.5kg
Temperatura operative	2ºC – 40ºC

Tabla 2.1. Especificaciones técnicas SCORBOT ER9

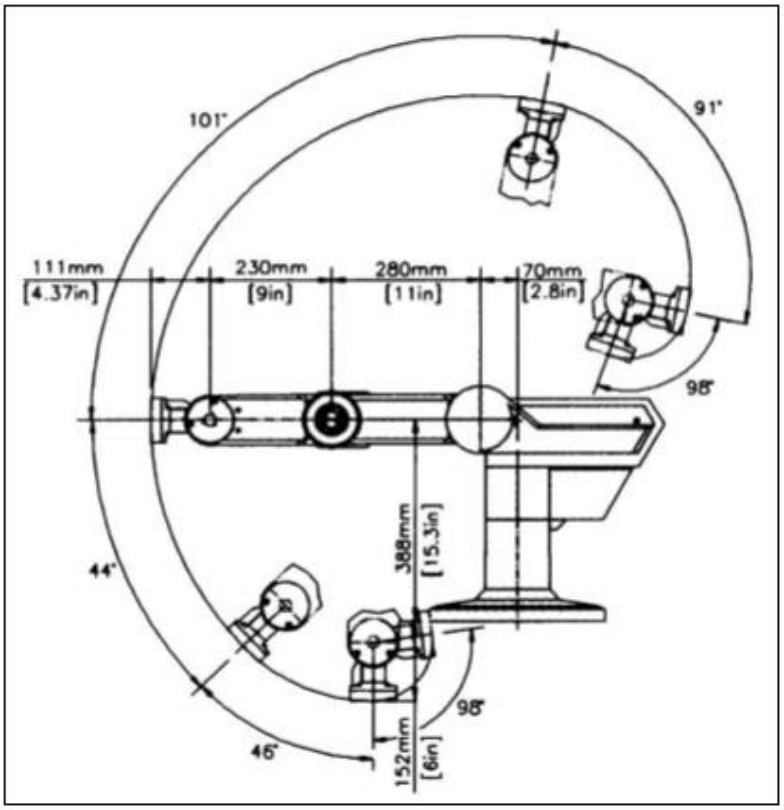


Figura 2.4. Rango Operativo (Vista lateral).

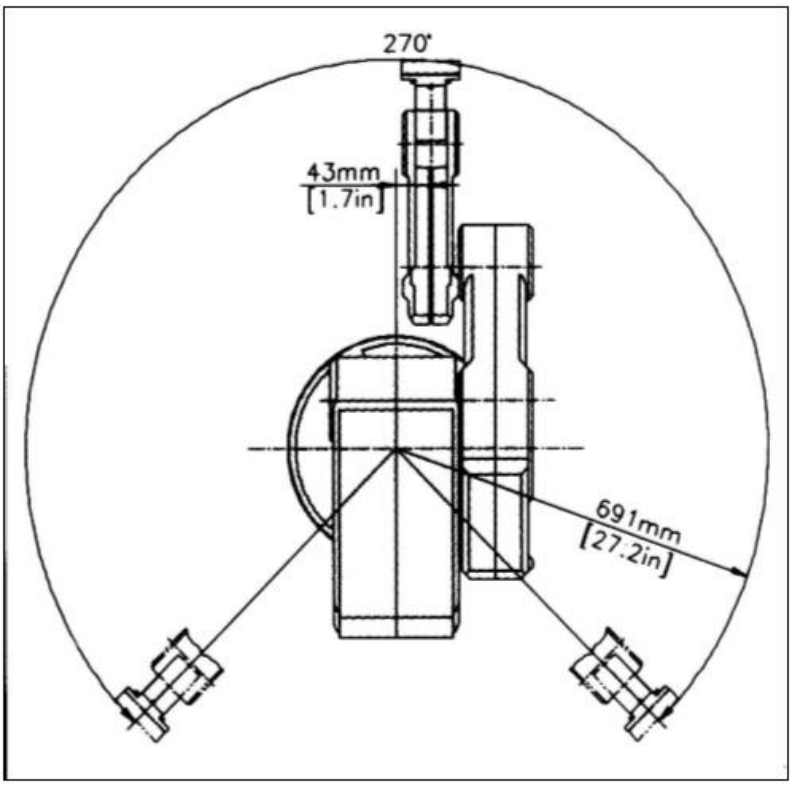


Figura 2.5. Rango Operativo (Vista superior).

Capítulo 3. Implementación

En este capítulo se describirán las fases que ha seguido el desarrollo de este proyecto, exponiendo las decisiones tomadas, durante el transcurso de la implementación.

3.1 Comunicación de MyoArmband con Unity3D

Como se comentó en el apartado de las tecnologías utilizadas, ThalmicLabs™ provee a los desarrolladores de librerías para diversos frameworks y, afortunadamente, dentro de este grupo se encuentra Unity3D.

Básicamente su uso consiste en la creación de un objeto que represente al MyoArmband dentro del framework, adjuntándole un script que lleva a cabo toda la comunicación de la interfaz con el programa.

3.2 Creación de un modelo 3D

La creación del modelo 3D que permitiera la representación del manipulador robótico en un entorno virtual consistió en varias fases.

En primera instancia, se decidió utilizar un modelo básico que permitiese ir probando la implementación de los diferentes algoritmos que se utilizaron sin preocuparse por el realismo.

Figura 3.1. Modelo 3D básico.

Una vez los aspectos técnicos estaban más perfilados, se decidió buscar un modelo a través de internet lo más semejante posible, pero fue imposible encontrar uno compatible con Unity. En su defecto, se encontraron los modelos tantos de las piezas por separado como del conjunto hecho en SolidWorks (.SLDPRT y .SLDASM), así que se decidió convertir estos modelos a un formato compatible (.fbx, .dae, .3ds, .dxf, .obj, o .skp). Para ello, se hizo uso de dos programas dado que no se encontró ningún programa que lo convirtiese directamente. En primer lugar se utilizó eDrawings [11] para hacer una conversión de .SLDPRT a .SLT, y en segundo lugar se pasó el archivo resultante por un conversor online [12] que lo convirtió a formato .OBJ, el cual es compatible directamente con Unity.

Ya teniendo cada pieza del modelo, se pasó al ensamblaje teniendo en cuenta cada uno de los ejes del modelo antiguo y haciendo que encajasen lo mejor posible.

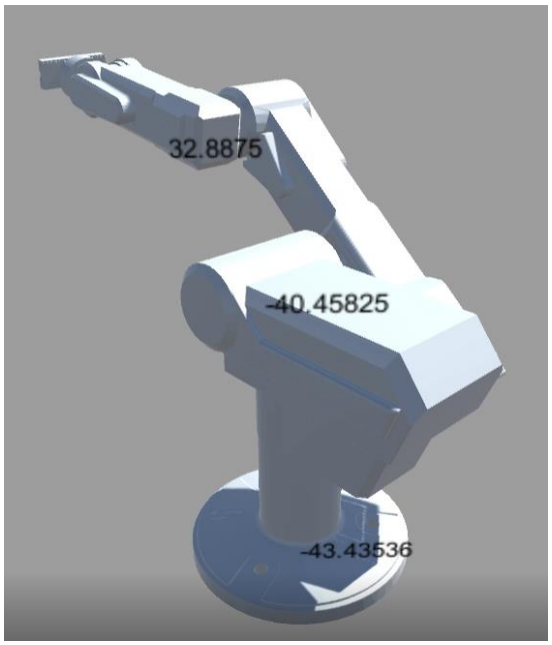


Figura 3.2. Modelo 3D complejo.

Disponiendo ya de un brazo y un modelo completamente funcional se pasó a realizar mejoras puramente estéticas, mejorando el entorno y el brazo en sí.

Figura 3.3. Modelo 3D final.

3.3 Movimiento del brazo

La implementación del algoritmo para el movimiento del brazo se dividió en dos fases:

- En primer lugar se creó un conjunto de ecuaciones que, en base a la entrada del MyoArmband – cabeceo, alabeo y guiñada – calculase una posición, la cual se correspondería con las coordenadas del efector final.
- En segundo lugar, se implementó el cálculo de la cinemática inversa con el fin de calcular los ángulos que debería de tomar cada articulación del robot para conseguir llegar a las coordenadas calculadas anteriormente.

3.3.1 Definición de las posiciones alcanzables por el robot

Para definir las posiciones que pueden ser alcanzadas por el robot se utilizaron tres ecuaciones, definidas en un sistema de coordenadas en el cual el eje ***(x, y)*** definen un plano horizontal, y el eje ***z*** se corresponde con el eje vertical:

$$x = \cos\alpha * \cos\theta * widthRadio$$

$$y = \sin\alpha * \cos\theta * widthRadio$$

$$z = l_1 + \sin\theta * heightRadio$$

$\alpha \equiv Ángulo\ en\ el\ eje\ vertical - Yaw$

$\theta \equiv Ángulo\ en\ el\ eje\ horizontal - Pitch$

$l_1 \equiv Desplazamiento\ en\ el\ eje\ vertial$

$widthRadio \equiv Radio\ del\ elipsoide\ en\ el\ eje\ horizontal$

$heightRadio \equiv Radio\ del\ elipsoide\ en\ el\ eje\ vertial$

Estas tres fórmulas describen un elipsoide como el que se muestra abajo, el cual se asemeja a las posiciones alcanzables por nuestro manipulador robótico, tomando como centro la primera articulación del robot – *correspondiente al hombro -.*

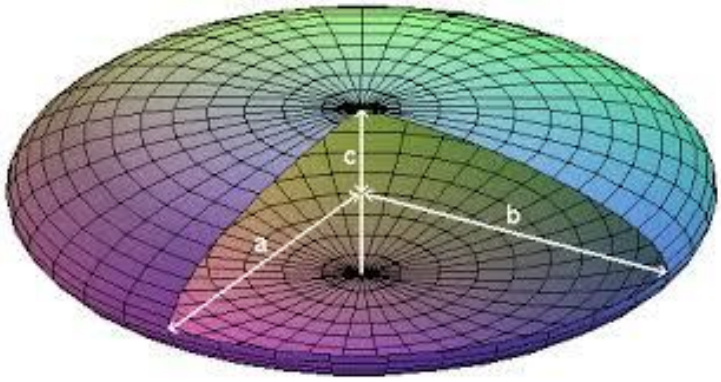


Figura 3.4. Elipsoide.

3.3.2 Cálculo de la cinemática inversa – Bases del cálculo

Para calcular los ángulos de las articulaciones se aplicó cinemática inversa, en concreto un método matemático para este manipulador.

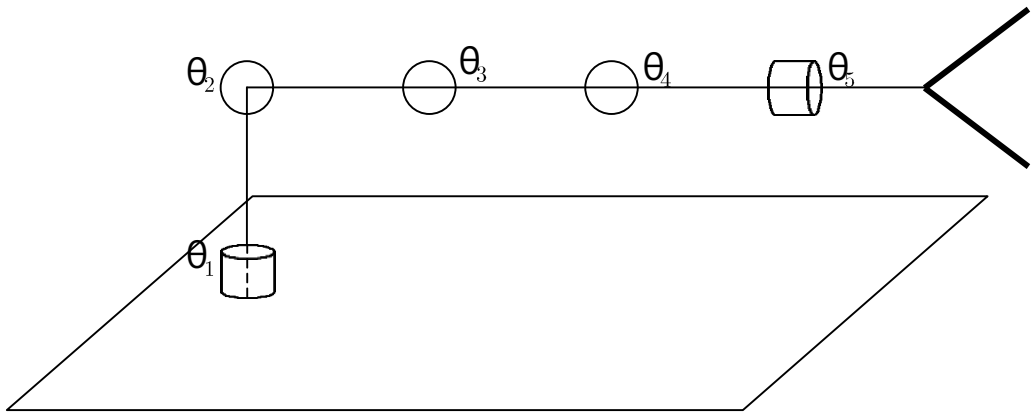


Figura 3.5. Esquema del manipulador

Tomando como referencia la nomenclatura de los ángulos definidos por el esquema anterior, se puede definir el problema como:

$$(\vec{p}, \vec{a}, \vec{s}) \rightarrow (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$$

$\vec{p} \equiv Vector\ de\ \ posición$
$\vec{a} \equiv Vector\ de\ acercamiento$
$\vec{s} \equiv Vector\ de\ cierre$

Pero, dado que en nuestro el vector de acercamiento $\vec{a}$ será siempre inverso al eje z - *debido a que la pinza siempre se encontrará apuntando hacia el suelo* -, se puede calcular θ_4 de una forma sencilla:

$$\vec{a} = -\vec{z}$$

$$\theta_4 = -\theta_1 - \theta_2 - \theta_3$$

También, se puede sacar de la ecuación el vector de cierre $\vec{s}$ y el ángulo θ_5 debido a que éste se asignará directamente tomando el ángulo en el eje horizontal *(Roll)* del MyoArmband aplicándole una cierta sensibilidad.

$$\theta_5 = \gamma * rollSensitivity$$

Esto nos deja con un problema bastante más reducido:

$$\vec{p} \rightarrow (\theta_1, \theta_2, \theta_3)$$

Donde $\vec{p}$ representa la posición del punto θ_3, asociado a la muñeca.

Una vez el problema se ha reducido, se dividió el problema en dos partes para facilitar el cálculo

- Realizar el cálculo para obtener los puntos de las articulaciones en un plano tomando solo las articulaciones θ_2 y θ_3
- Incluir la articulación θ_1 la cual introduciría un componente tridimensional y realizar el cálculo de nuestro nuevo problema sustituyendo los datos obtenidos en el punto anterior.

3.3.3 Cinemática Inversa – Manipulador planar

Considerando que el manipulador se encuentra definido en un plano, es decir $\theta_1 = cte$. Si se sustituye θ_2 por θ_1 y θ_3 por θ_2:

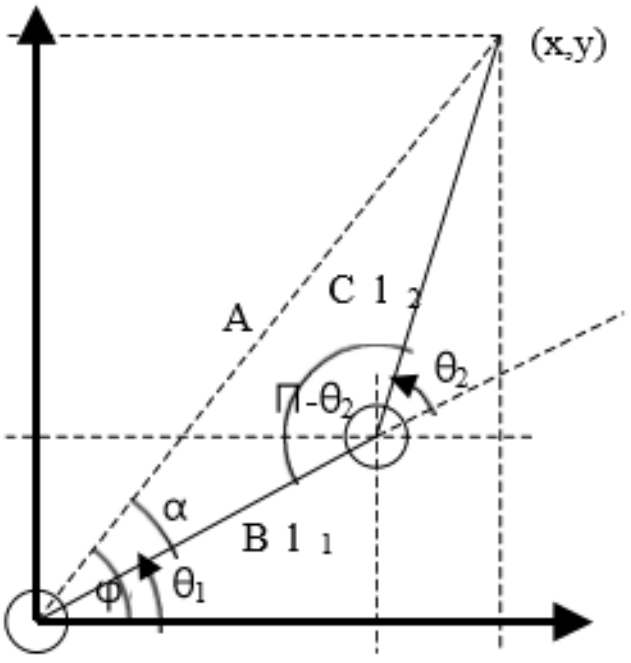
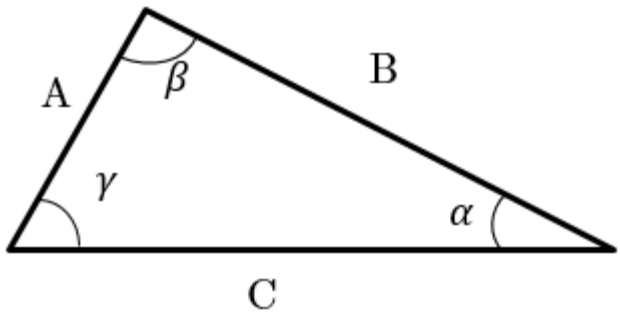


Figura 3.6. Manipulador planar. Desglose de ángulos.

Teniendo en cuenta el Teorema de Pitágoras Generalizado:



$$A^2 = B^2 + C^2 - 2BC\cos\alpha$$

Se puede deducir que:

$$x^2 + y^2 = {l_1}^2 + {l_2}^2 - 2 \cdot l_1 \cdot l_2 \cdot \cos(\pi - \theta_2)$$

Y despejando θ_2

$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - {l_1}^2 - {l_2}^2}{2 \cdot l_1 \cdot l_2}\right)$$

Esto devuelve dos posibles soluciones, las cuales se corresponden a cuando el manipulador se encuentra con una pose “codo arriba” o “codo abajo” la cual se puede ver en la imagen inferior:

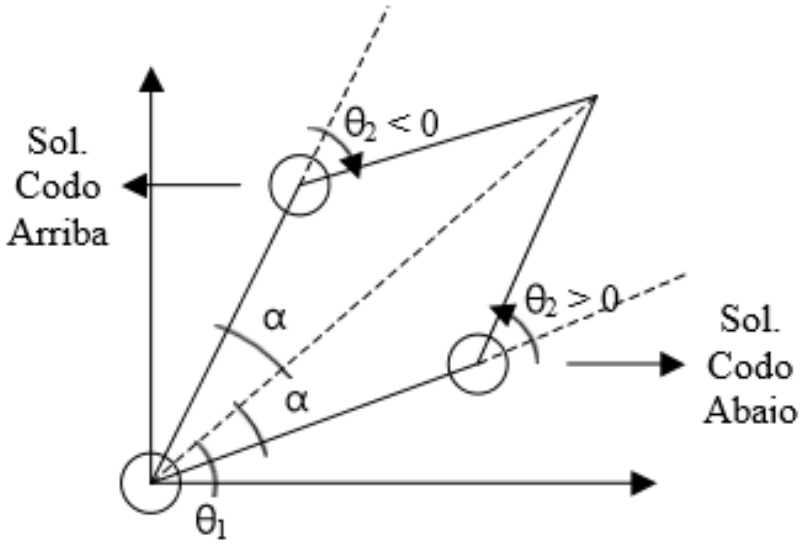


Figura 3.7. Codo arriba/Codo abajo.

Para calcular el valor de θ_1

$$\theta_1 = \varphi - \alpha$$

$$\tan\varphi = {}^{y}/_{x}$$

$$\tan\alpha = \frac{l_2 * \sin\theta_2}{l_1 + l_2 \cdot \cos\theta_2}$$

θ_1 *tendrá un resultado u otro dependiendo de si* θ_2 *se ha calculado para que sea "codo arriba" o "codo abajo".*

3.3.4 Cinemática Inversa – Manipulador antropomórfico

Debido a que ya se disponen de todas las fórmulas necesarias para el cálculo de la cinemática inversa para un manipulador planar, se pasa a definir las ecuaciones agregando θ_1 *(ángulo de la base)* introduciendo el componente tridimensional.

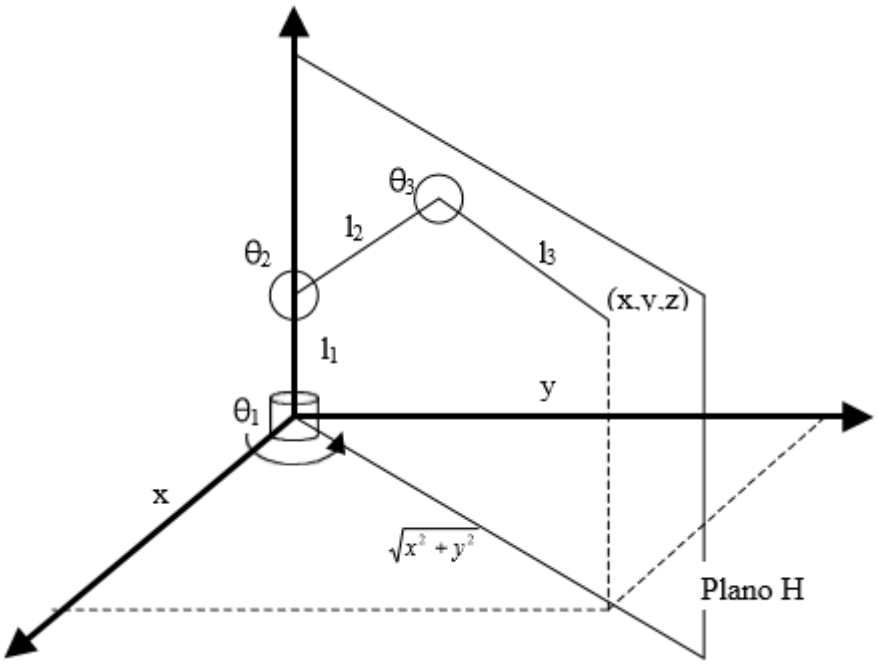


Figura 3.8. Manipulador antropomórfico.

Como se aprecia en la imagen superior, se puede definir un Plano H al cual poder aplicarle las ecuaciones obtenidas en el apartado anterior y así después solo aplicarle una sustitución a las formulas agregando el ángulo θ_1:

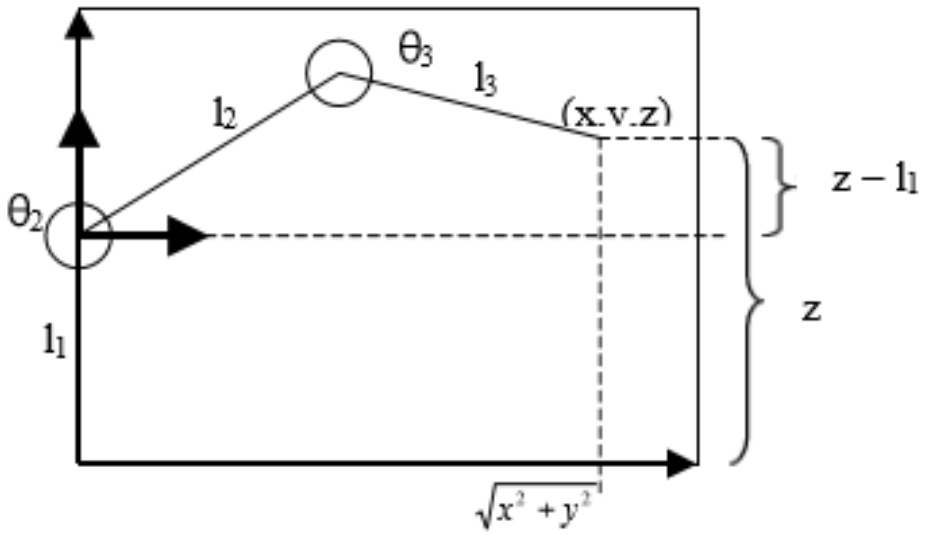


Figura 3.9. Plano H.

Manipulador Planar	Manipulador Antropomórfico
x	$\sqrt{x^2 + y^2}$
y	$z - l_1$
θ_1, l_1	θ_2, l_2
θ_2, l_2	θ_3, l_3

Tabla 3.1. Sustitución planar-antropomórfico

Si se lleva a cabo esta sustitución y se despeja.

$$\boldsymbol{\theta_3 = \cos^{-1}\left[\frac{x^2 + y^2 + (z - l_1)^2 - l_2^2 - l_3^2}{2 \cdot l_2 + l_3}\right]}$$

$$\boldsymbol{\theta_2 = \varphi - \alpha}$$

$$\varphi = \tan^{-1}\left(\frac{z - l_1}{\sqrt{x^2 + y^2}}\right)$$

$$\alpha = \tan^{-1}\left(\frac{l_3 \cdot \sin\theta_3}{l_2 + l_3 \cdot \cos\theta_3}\right)$$

$$\boldsymbol{\theta_1 = \tan^{-1}\left(\frac{y}{x}\right)}$$

3.3.5 Restricciones del movimiento

Dado que el manipulador real posee restricciones físicas, éstas se han implementado intentando que sean lo más realistas posibles. Estas restricciones se comprueban antes de que se actualice la posición del robot.

En caso de que alguna de las restricciones impida el movimiento del manipulador, el brazalete MyoArmband enviará una señal de vibración para darle un feedback al usuario.

Las restricciones se encuentran divididas en dos secciones:

- Restricciones de posición
- Restricciones angulares

Las restricciones de posición impiden que el robot alcance unas coordenadas que estén fuera de su rango. Estas restricciones son controladas mediante las ecuaciones de cinemática. Concretando un poco más, cuando $\cos\theta_3$ da un valor superior a 1, se entiende que es una posición inalcanzable.

Las restricciones angulares vienen definidas por la capacidad que tiene el robot de rotar sus articulaciones. Estos parámetros se han hallado por medio de las fichas técnicas del fabricante, y son las siguientes:

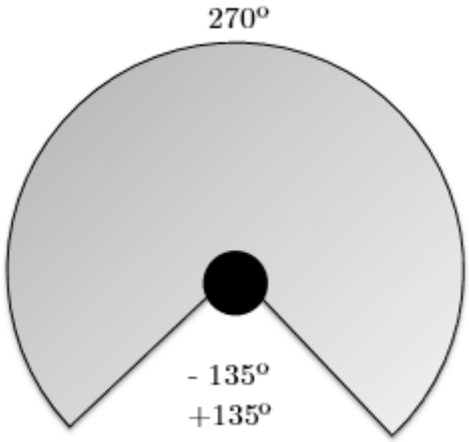


Figura 3.10. Restricción Base (Vista superior).

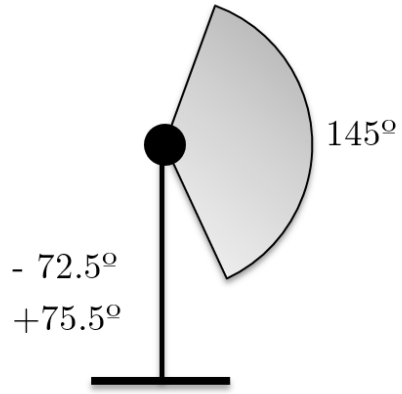


Figura 3.11. Restricción Hombro (Vista lateral).

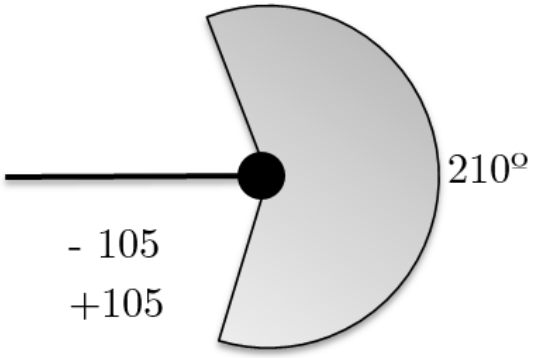


Figura 3.12. Restricción Codo (Vista lateral).

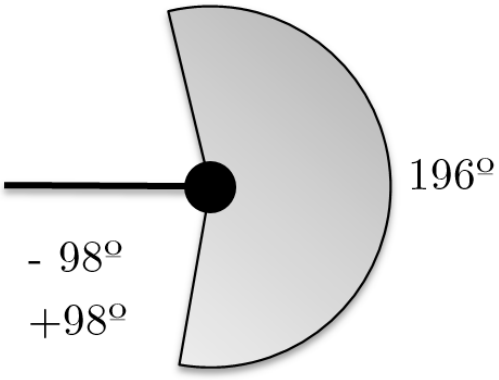


Figura 3.13. Restricción Muñeca (Vista lateral).

3.4 Comunicación de Unity3D con SCORBOT ER9

El proceso de comunicación entre el SCORBOT ER9 y el programa se realiza mediante el puerto SERIE debido a que es el único medio de comunicación del que dispone el brazo.

Para la implementación de los protocolos de comunicación se utiliza la librería de .NET Framework, que viene incluida nativamente dentro de Unity. .NET facilita bastante la implementación del proceso de comunicación gracias a una clase que representa un puerto serial, llamada SerialPort. Esta clase se utiliza definiendo los parámetros de configuración del protocolo utilizado por el SCORBOT, los cuales se ven en la tabla de abajo.

Configuración puerto serie SCORBOT ER9	
BaudRate	9600
Parity	None
DataBits	8
StopBits	1
XonXoff	No

Tabla 3.2. Configuración puerto serie SCORBOT ER 9

3.4.1 Protocolo de envío de mensajes

A la hora de definir un protocolo para el envío de mensajes dentro del programa, lo primero que se hizo fue comprobar empíricamente con que cadencia el SCORBOT podía recibir los mensajes para no saturar su buffer. El tiempo mínimo que se ha de esperar entre el envío de mensajes es de aproximadamente 10ms, lo que dificulta llevar a cabo movimientos naturales, pero no llega a ser del todo imposible.

Una vez obtenido el tiempo, se pasó a implementar un procedimiento por el cual controlarlo dentro del programa, el cual se decidió realizar mediante una cola.

Esta cola, recibe una entrada de datos cada segundo. Estos datos se corresponden con la sentencia ACL denominada *teach*. Esta sentencia permite definir una posición del robot enviando cinco parámetros. Los tres primeros parámetros son correspondientes a las coordenadas (x,y,z) a la cual queremos que el efector final llegue, el cuarto y el quinto parámetro se corresponden a los ángulos que definen el vector de acercamiento $(\vec{a})$ y el vector de cierre $(\vec{n})$.

Aparte de estos cinco mensajes hacen falta otros dos al inicio y al final. El primero se encarga de definir el nombre que le queremos dar a la posición y el último mueve el brazo a la posición indicada.

En el siguiente fragmento de código se ve la estructura que sigue el comando:

```
serialPortQueue.Enqueue ("teach pos1");
serialPortQueue.Enqueue ((X*100).ToString("0.000"));
serialPortQueue.Enqueue ((Y*100).ToString("0.000"));
serialPortQueue.Enqueue ((Z*100-80).ToString("0.000"));
serialPortQueue.Enqueue ("-90");
serialPortQueue.Enqueue ((R).ToString("0.000"));
serialPortQueue.Enqueue("move pos1");
```

Una vez se tiene el conjunto de comandos que se le quiere enviar al SCORBOT, estos tienen que irse enviando progresivamente cada segundo. Este proceso se realiza manteniendo un temporizador en el cual, al sobrepasar los 10ms se saca de la cola el último mensaje y se envía, consiguiendo que se envía en el orden y con las restricciones temporales correctas.

```
if(serialPortQueue.Count > 0 )
    command = serialPortQueue.Dequeue ();
    if(command == null || command == "")
        return;
    sendCommand(command);
```

Las diferencias de tiempo que existe entre la captura de las posiciones para la sentencia *teach* y el tiempo entre envío de mensajes se debe a que, a la hora de que el robot deba efectuar un movimiento, este ha de completar dicho movimiento antes de realizar otro, por lo que ha de existir un desfase mayor entre sentencias que definan una traslación, no ocurriendo entre los comandos.

Capítulo 4.
Control

En este capítulo se desglosarán los movimientos necesarios para llevar a cabo el movimiento del brazo robótico.

4.1 Desplazamiento del brazo

El desplazamiento, tanto del modelo en 3D como del SCORBOT ER9 se realiza mediante los diferentes ángulos dados por MyoArmband. Estos ángulos, - como se explicó en el punto 3.3 – definen un punto dentro del límite de una esfera, el cual nos da la posición a la que debería llegar el efector final.

Antes de comenzar el movimiento, se inicializa un punto como base, el cual sirve como referencia para el movimiento. Una vez se tiene este punto, se activa la entrada del MyoArmband el cual modifica las coordenadas mediante el cambio de los ángulos de este.

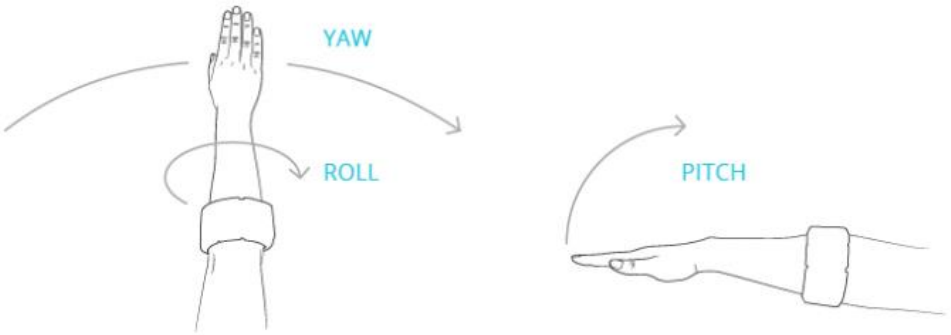


Figura 4.1. Movimientos angulares.

El eje de guiñada o yaw produce un desplazamiento en los ejes laterolatera y dorsoversal, mientras que el eje de cabeceo o pitch produce un desplazamiento en los ejes craneocaudal y dorsoventral [Figura 4.2.].

El eje de guiñada o roll se utiliza para modificar el ángulo de cierre de la pinza del robot, esta conversión se hace prácticamente directa dado que se realiza de una forma muy natural.

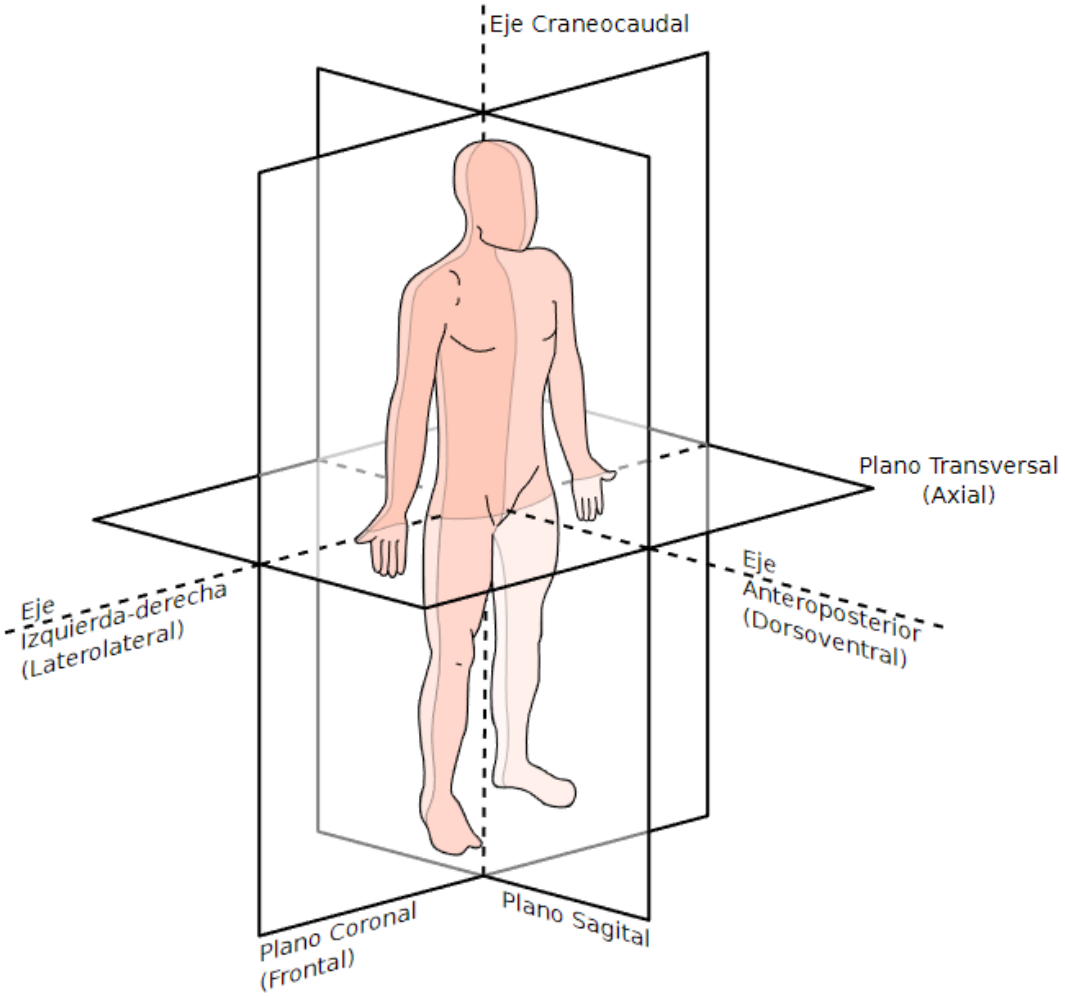


Figura 4.2. Planos anatómicos.

4.2 Gestos

Se utilizaron en total tres gestos de un total de cinco disponibles. Esta decisión fue tomaba debido a que no se necesitaban más teniendo en cuenta las características y limitaciones del robot.

4.2.1 Apertura de pinza

La apertura de pinza se realiza mediante una apertura de la palma de la mano de una forma en la que el gesto se vea reflejado en los músculos del antebrazo.

Figura 4.3. Gesto de apertura de pinza.

4.2.2 Cierre de pinza

De la misma forma que el anterior, el cierre de la pinza se realiza mediante cierre del puño de manera que este se vea reflejado en los músculos del antebrazo.

Figura 4.4. Gesto de cierre de pinza.

4.2.3 Reinicio del brazo

Para volver a la posición inicial, se ha de realizar un doble "click" entre el dedo índice y el pulgar de una forma marcada y al igual que en el caso anterior, viéndose reflejado en los músculos del antebrazo.

Figura 4.5. Gesto de reinicio del brazo.

Capítulo 5. Metodologías alternativas

En este capítulo se definirán las metodologías que se intentaron implementar para mejorar algunos aspectos de la aplicación o solventar otros.

5.1 Acelerómetro

En un principio, se comenzó a implementar un algoritmo el cual permitiera variar dinámicamente el radio del elipsoide utilizando los valores del acelerómetro. En los siguientes puntos se exponen las ideas que se implementaron y no dieron como resultado el esperado.

5.1.1 Cálculo matemático

En primera instancia, se intentó implementar esta alternativa utilizando formulas físicas básicas, con las cuales, mediante con un vector de aceleración $\vec{a}$ obtener el desplazamiento $\vec{s}$ de un punto en un tiempo t determinado.

MyoArmband nos da un vector de aceleraciones $(\vec{a}_x, \vec{a}_y, \vec{a}_z)$, el cual está situado en un sistema de coordenadas locales. Para trasladar este vector a un sistema global, se han de cruzar los datos del acelerómetro con el cuaternión que define la rotación del brazalete.

$$\vec{A} = q \cdot \vec{a}$$

El resultado es un vector de aceleración $\vec{A}$ situado en un sistema de coordenadas global con el cual se puede empezar a aplicar fórmulas para obtener la velocidad $\vec{v}$ y posteriormente el desplazamiento para un tiempo s.

$$\vec{v} = \vec{v}_0 + \vec{A} \cdot t$$
$$\vec{s} = \vec{s}_0 + \vec{v} \cdot t$$

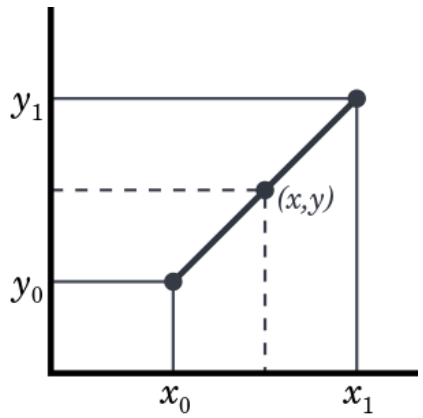
$$\vec{v}_{inicial} = \vec{0}$$
$$t \equiv tiempo\ entre\ frames$$

Esta sería la forma más sencilla de obtener la distancia recorrida por nuestro brazalete, pero tiene el gran inconveniente de que tanto el acelerómetro como el giroscopio poseen errores que se van acumulando en cada fórmula que se realice, y esto, sumado a que las muestras se obtienen discretamente, genera un grandísimo error, lo que produce que a la hora de acelerar y decelerar, las magnitudes no se contrarresten adecuadamente, por lo que la velocidad del dispositivo es muy probable que nunca sea cero, haciendo que la representación del dispositivo en el programa se mantenga en continuo movimiento, aunque este esté completamente en reposo.

5.1.2 Suavizado de señal

Otra posible solución fue la de suavizar la señal para que no se produzcan picos, los cuales son producidos por perturbaciones y errores, y que en mayor medida no provienen del movimiento que quiere realizar el usuario. Para el suavizado se utilizó la función Lerp, que recibe como parámetros el vector $\vec{x}$, el vector $\vec{y}$ y el parámetro interpolador t.

```
Vector3.Lerp(Vector3 x, Vector3 y, float t)
```

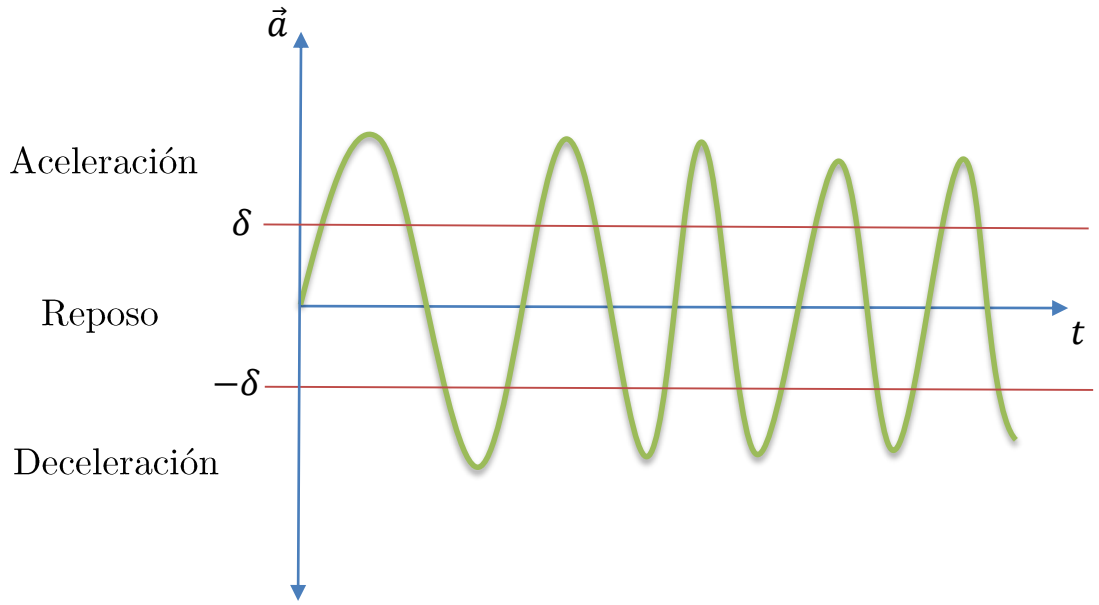


Esta función hace una interpolación lineal entre dos vectores tomando como parámetro el interpolador $t \in \mathbb{R}$, $0 < t < 1$. Cuando $t = 0$ es devuelto el vector $\vec{x}$, en caso de que $t = 1$ es devuelto el vector $\vec{y}$.

Esta interpolación se le aplicó a la señal de la aceleración, la cual, a pesar de mejorar la precisión con la que se detectaba la posición, se quedaba muy lejos de lo que haría falta para obtener una velocidad y posición precisas.

5.1.3 Discretización la señal y máquina de estado

La última solución que se implementó fue la de crear unos rangos de valores que representen un estado de la señal. Se utilizaron tres rangos, uno que represente una aceleración positiva, uno que represente una aceleración negativa y otro que represente un estado de reposo.



Esta primera aproximación se hizo sin tener en cuenta el tiempo, tomando como referencia para la discretización solo el paso de un intervalo a otro.

t-1	t	Resultado
$-\delta < \vec{a}_x < \delta$	$\vec{a}_x > \delta$	$\vec{v}_x = \vec{v}_x + w$
$-\delta < \vec{a}_x < \delta$	$\vec{a}_x < \delta$	$\vec{v}_x = \vec{v}_x - w$
$\vec{a}_x < \delta$ ó $\vec{a}_x > \delta$	$-\delta < \vec{a}_x < \delta$	$\vec{v}_x = \vec{v}_x$

Tabla 5.1. Rango de valores

$$w \equiv Incremento\ de\ velocidad$$

$$\delta \equiv Límite\ de\ las\ zonas\ de\ aceleración$$

Esto permite que no se tengan tanto en cuenta los errores que puede cometer el acelerómetro, perdiendo bastante precisión a la hora de saber exactamente la aceleración que se está produciendo pero obteniendo una mejor aproximación de la velocidad que lleva el brazalete.

En términos generales esta solución fue bastante acertada debido a que se podía obtener de forma muy precisa el vector $\vec{v}$, y en cierto grado su magnitud. Pero como inconveniente, este método no funcionaba muy bien cuando la magnitud de la aceleración era muy pequeña y había que reducir el límite δ.

Esto es debido a que en términos de aceleración pequeños, los errores cometidos por los sensores se hacen más notables.

5.2 Acelerómetro - Solución

Una posible solución para el problema descrito en el punto anterior es el uso de dos MyoArmband, uno en el antebrazo (representando el vector que une el codo y la muñeca) y otro en el brazo (representando el vector que une el hombro y el codo). De esta forma se podría obtener la extensión del brazo de una forma muy simple y extrapolar estos datos modificando así el radio de actuación del robot de una forma muy precisa.

5.3 Alternativa a Unity3D

Al principio del desarrollo, se barajaron diferentes alternativas a Unity3D, en concreto, cabe a destacar el motor Unreal Engine 4.

Este motor posee una muy buena potencia y un muy buen motor gráfico a la par que facilita bastante la implementación de algoritmos sencillos mediante los llamados Blueprints – que no son más que diagramas de flujo -.

El inconveniente de utilizar este software es claro, y es que ThalmicLabs™ no ha desarrollado librerías compatibles con él, por lo que el desarrollo desde creo de estas sería contenido que daría para otro proyecto.

5.4 Líneas futuras de investigación

Este proyecto no empezó con el enfoque que se le ha dado durante esta memoria. En un principio fue pensado de forma mucho más general, permitiéndole al usuario crear de forma dinámica un brazo que se adaptara a las características que quisiese. Dichas características se le trasladarían al programa mediante los parámetros de Denavit-Hartenberg.

Esta primera versión del programa fue lograda, sin llegar a empezar a implementar la comunicación con el brazalete debido a la complejidad añadida que supondría diseñar e implementar el algoritmo correspondiente y acabar todos los puntos de este proyecto.

5.4.1 Implementación del primer proyecto

En esta primera versión del proyecto se consiguió implementar un algoritmo el cual permitiese crear de forma dinámica el robot que se quería. Para ello, se creó una serie de pantallas que permitían elegir al usuario un sistema de coordenadas base el cual tomaría como referencia para comenzar la construcción del robot.

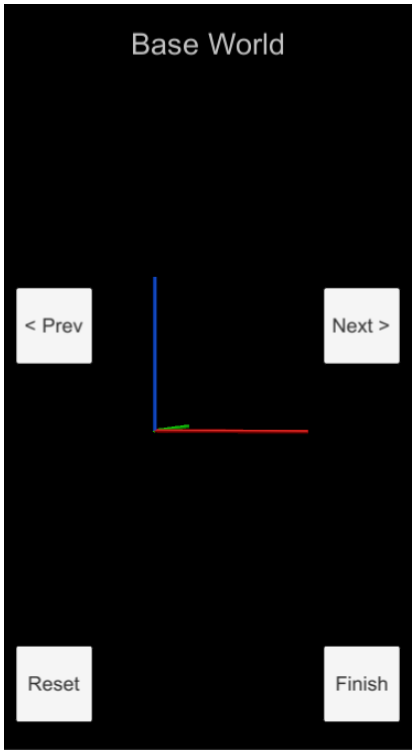


Figura 5.1. Selección del sistema de coordenadas.

Una vez elegido este sistema de coordenadas, el usuario puede ir introduciendo los cuatro parámetros de la matriz de Denavit-Hartenberg (a, b alfa y omega), cambiando entre los mundos que representan cada articulación y permitiendo definir tantas como el usuario quisiera.

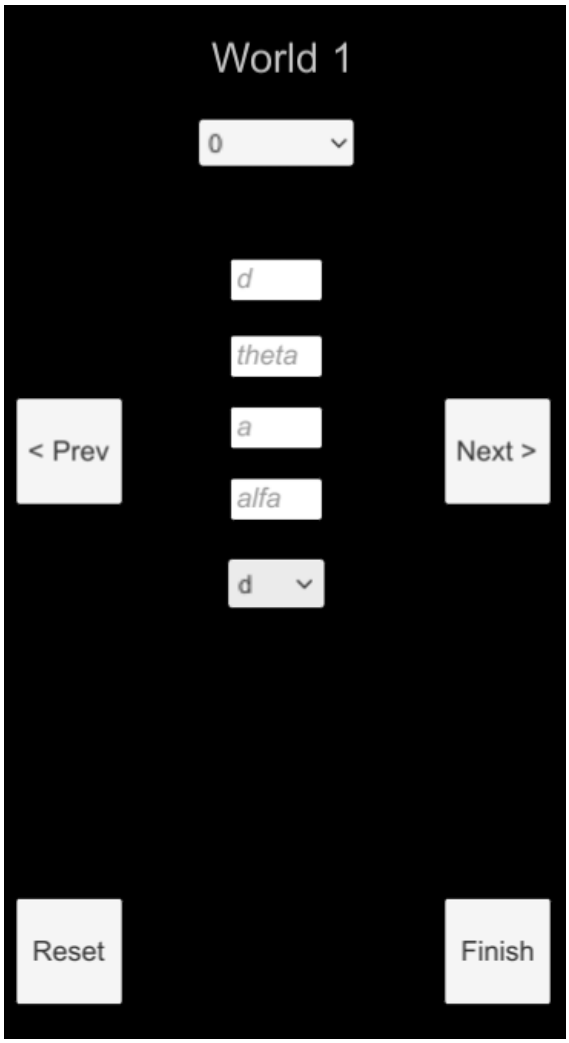


Figura 5.2. Inicialización de los datos.

Ya definidos todos los parámetros, el programa se encarga de crear un brazo ensamblando los distintos mundos uno tras otros hasta que se consigue el resultado final.

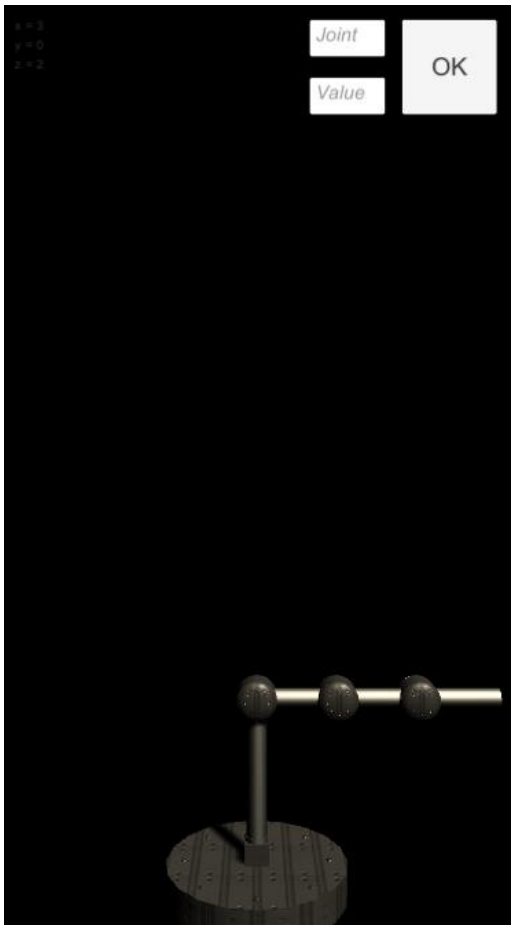


Figura 5.3. Ensamblado del brazo.

Parámetros D-H	Mundo 1	Mundo 2	Mundo 3	Mundo 4
d_i	q_0	0	0	0
θ_i	0	q_1	q_2	q_3
a_i	0	1	1	1
α_i	0	90	0	0

Tabla 5.2. Matriz D-H del ejemplo

5.4.2 Movimiento del robot

En esta primera versión también permite el movimiento del brazo de una forma muy simple. Este sistema consiste que el usuario introduzca dos parámetros, en la primera casilla el número de articulación que desea mover, y en la segunda el valor que se desea incrementar.

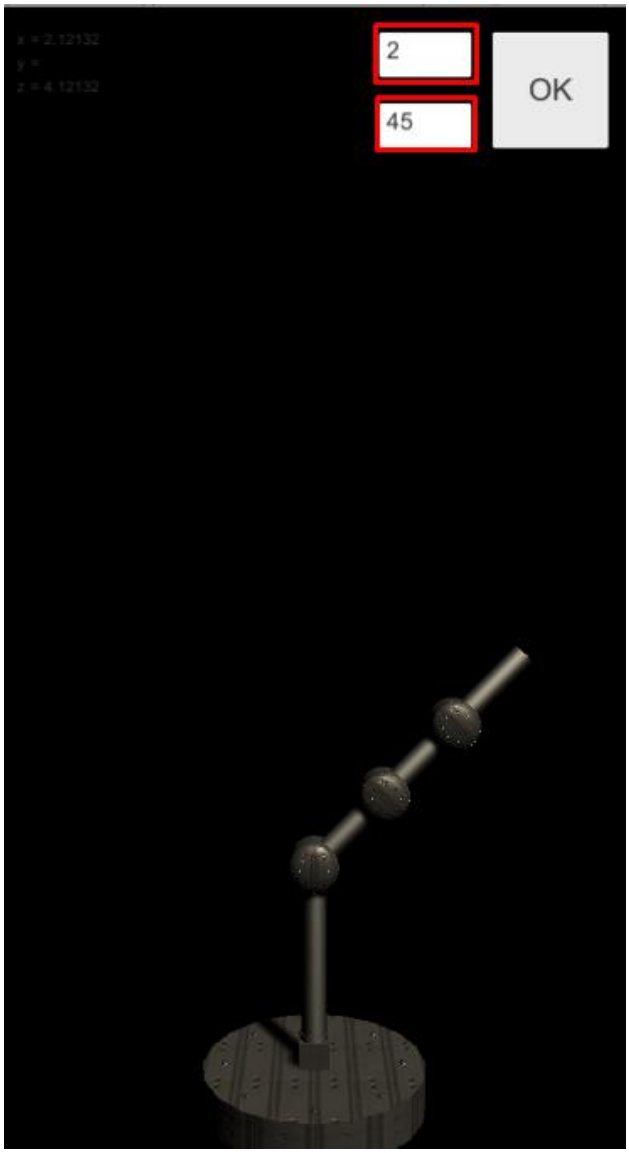


Figura 5.4. Movimiento de las articulaciones.

Este movimiento se creó con el fin de implementar una interfaz que le permita mover el brazo de una forma sencilla y recibir comandos de una clase que calcule la ingeniería inversa.

En el siguiente ejemplo se muestra una deformación del brazo más vistosa. También se puede apreciar el resultado de la aplicación de las fórmulas de cinemática directa.

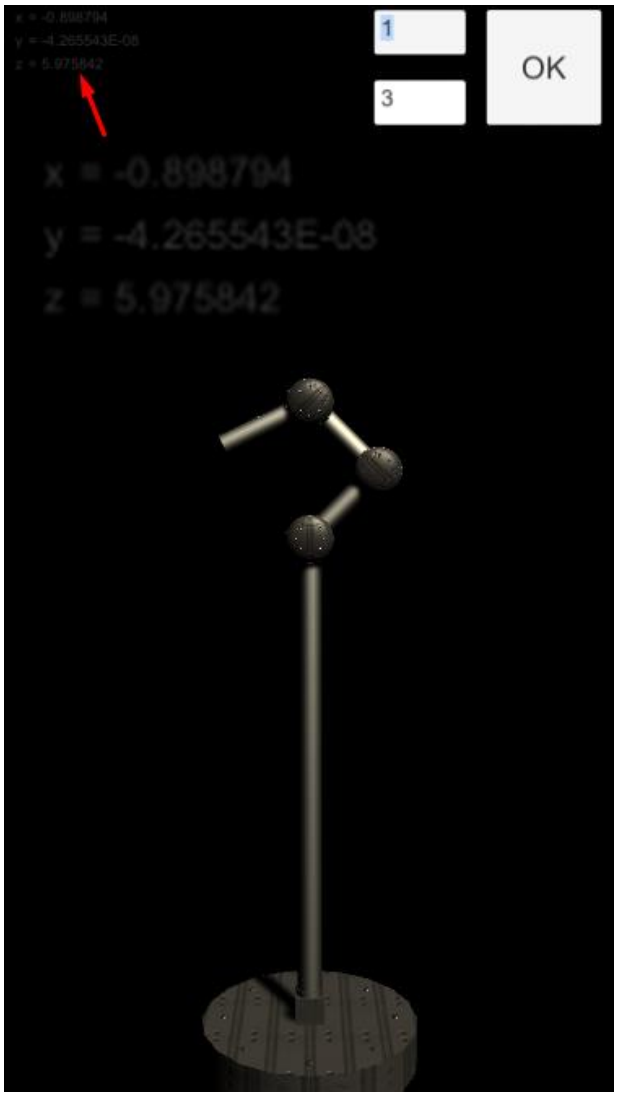


Figura 5.5. Cálculo cinemática directa.

5.4.3 Restricción

La principal restricción es que este programa no permitía que un mundo tuviese dos mundos hijos, no permitiendo que se puedan representar pinzas o

La principal restricción de este programa es que no permitía que un mundo padre tuviese dos hijos, es decir, la representación de un brazo el cual tuviera unas pinzas o un sistema similar es imposible de representar.

5.4.4 Mejoras

Sería muy interesante seguir la progresión de este Trabajo de Fin de Grado incluyendo las características desarrolladas en la primera fase del proyecto y que no fueron implementadas en la versión final, para permitirles a los usuarios experimentar y probar por su cuenta diferentes configuraciones.

Otra mejora sustancial sería la de incorporar las modificaciones necesarias para solventar la restricción del punto 5.4.3, de tal forma que se le pueda dar aún más posibilidades al usuario.

Capítulo 6.
Resumen y Conclusiones

Al finalizar este proyecto se puede decir que se ha superado con éxito los objetivos que se pretendían conseguir, logrando crear un sistema que de una forma intuitiva permita el control tanto de un brazo virtual como de uno real, aplicando principios cinemáticos y diversos conocimientos adquiridos durante todo el transcurso del grado.

Si bien es cierto que se podría haber creado un sistema mucho más completo, fluido y dinámico, se han tenido unos resultados bastantes satisfactorios, obteniendo un sistema bastante estable y con muy buenas características teniendo en cuenta las restricciones materiales y temporales con las que se contaba.

Por otro lado, y mirándolo desde un punto académico, este proyecto puede llegar a hacer ver y comprender a los alumnos las restricciones y el comportamiento de un sistema robótico. Además de la visión que puede aportar este trabajo, también se puede trabajar sobre éste creando y ampliando sus características de forma que los estudiantes se sientan más atraídos y vean la robótica desde un punto de vista más cercano.

En cuanto al desarrollo, cabe destacar el grandísimo reto personal que ha supuesto el comunicar diversos tipos de tecnologías, pasando por diferentes entornos sin tener ningún conocimiento de la tecnología que habría de usarse ni las formas que se tenían para comunicarlas.

Capítulo 7.
Summary and Conclusions

Once this project is finished it can be said that the objectives have been successfully overcome, creating a system that allows an intuitive control of the virtual and real arm, applying kinematic principles and a lot of knowledge acquired during the grade.

While it is true that the system can be more complete, a fluid and dynamic system could have been created. However, satisfactory results have been obtained, creating a fairly stable system with very good characteristics despite of physical and temporal constraints.

Looking at this project from academic point of view, it can make the students to understand the restrictions and behavior of a robotic system. In addition to the vision that can bring this project, you can also work on it by creating and expanding its features so that students feel more attracted and see the robotics from a closer point of view.

In terms of development, it is important to highlight the great personal challenge of communicating different types of technologies, going through different environments without having any knowledge of the technologies to be used or the ways to communicate them.

Bibliografía

[1] Hugh Herr Wikipedia. https://es.wikipedia.org/wiki/Hugh_Herr.

[2] CONSALUD. Prótesis inteligentes e implantes que mejoran el cuerpo incluso de personas sanas. https://consalud.es/saludigital/revista/protesis-inteligentes-e-implantes-que-mejoran-el-cuerpo-incluso-de-personas-sanas--246.

[3] Control de vehículos remotos. Wikipedia. https://en.wikipedia.org/wiki/Remote_control_vehicle

[4] Unity 3D. Webpage. https://unity3d.com/es.

[5] Unity 3D. Wikipedia. https://es.wikipedia.org/wiki/Unity_(motor_de_juego)

[6] Thalamic Lab Webpage. https://www.thalmic.com/

[7] Myo Armband Webpage. https://www.myo.com/

[8] Intilek Webpage. http://www.intelitek.com/

[9] SCORBOT ER9. Documentación. ftp://ftp.robotec.co.il/Techsup/er9pro/Books/200034%20Rev%20A%20Scorbot%20ER%209Pro.pdf

[10] Solidworks Webpage. http://www.solidworks.es/

[11] eDrawings SLDPRT to STL. http://www.edrawingsviewer.com/.

[12] STL to OBJ Converter. http://www.greentoken.de/onlineconv

[13] .NET Framework. https://es.wikipedia.org/wiki/Microsoft_.NET