Resumen Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Exploración del software de simulación V-REP

Autor: Jesús de Miguel Fernández
Tutor: José Ángel Acosta Rodríguez

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Exploración del software de simulación V-REP

Autor:
Jesús de Miguel Fernández

Tutor:
José Ángel Acosta Rodríguez
Profesor titular

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

ÍNDICE

1 INTRODUCTION

1.1 ¿Por qué V-REP®?

Un simulador es esencial para la prueba de cualquier sistema robótico ya que se evita que éste se dañe. También es muy útil para la enseñanza ya que no se precisa del robot físicamente. La mayoría de softwares que se encuentran en el mercado están diseñados para los robots de la propia empresa que desarrollaron este software y en raras ocasiones es gratuito. Por este motivo recurrimos a V-REP®. Este software de la empresa Coppelia Robotics es un simulador: libre, gratuito, que permite diferentes modos de simulación y trae unos algoritmos a primera vista bastante atractivos para el cálculo de elementos esenciales en cualquier robot como son la cinemática inversa y la detección de colisiones. El entorno gráfico te permite diseñar tu propio robot con los elementos que te aporta o importar tu propio diseño desde cualquier entorno gráfico. Eres libre en tu diseño.

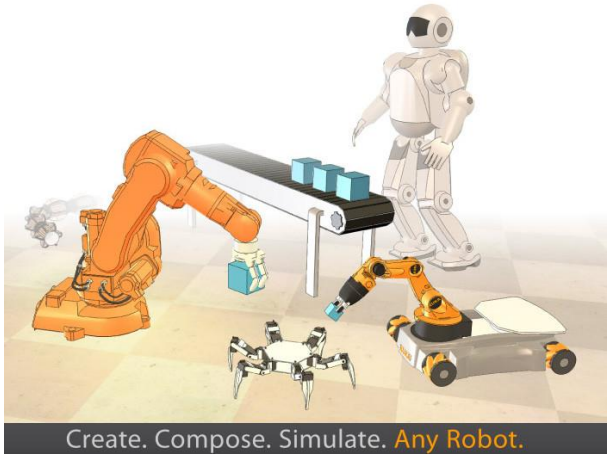

Figure 1-1.EntornoV-REP

1.2 ¿V-REP® es sencillo de manejar?

El desarrollador te proprociona un extensor manual en fromato digital en el que vienen detallados los temas necesarios para familiarizarse con el entorno. Hay ciertas temas que no vienen explicados con la profundidad que se merece, pero que serán desarrollados en este documento. Al ser una aplicación en alza, tiene una comunidad bastante grande detrás en la que vienen resueltos la mayoría de problemas con los que te puedes encontrar. Además, la propia empresa te responde en un periodo muy corto de respuesta. Desde mi punto de vista cuesta hacerse con el entorno, principalmente a la hora de construir tu propio robot.

1.3 ¿V-REP® tiene limitaciones?

Es un simulador muy interesante, pero hay ciertos elementos que en primera instancia no sabes muy bien qué estás modificando. El lenguaje de programación, LUA, es un lenguaje de muy alto nivel con funciones prediseñadas con el que a veces te sientes atado de manos. Otra limitación presente es que los algortimos de

cláculo del jacobiano (Pseudoinversa y DLS) están herméticamente cerrados y no puedes modifcar nada.

Nuestro estudio en esta aplicación comenzará con un robot muy simple como es un robot de dos grados de libertad que sólo se mueve en el plano XY para hacernos con el entorno. A continuación, vamos a pasar al caso de un robot humanoid, *Biolid Premium Tipo A*.

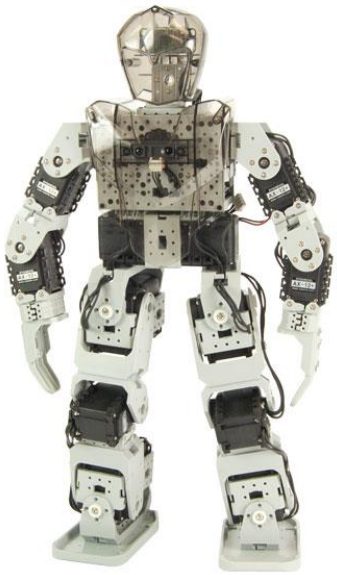

Figure 1-2.Modelo Biolid Premium Tipo A

Debido a la limitaciones que encontraremos y para entender mejor las estructuras de control que esta aplicaión implementa, buscaremos un conexión *API* con MATLAB® donde V-REP® es el servidor y MATLAB® el cliente.

2 ROBOT DE 2 GDL EN V-REP®

2.1 Construcción

Para la constucción de este robot en V-REP utilizaremos las formas básicas que te proporciona el programa. Este sería el resultado final:

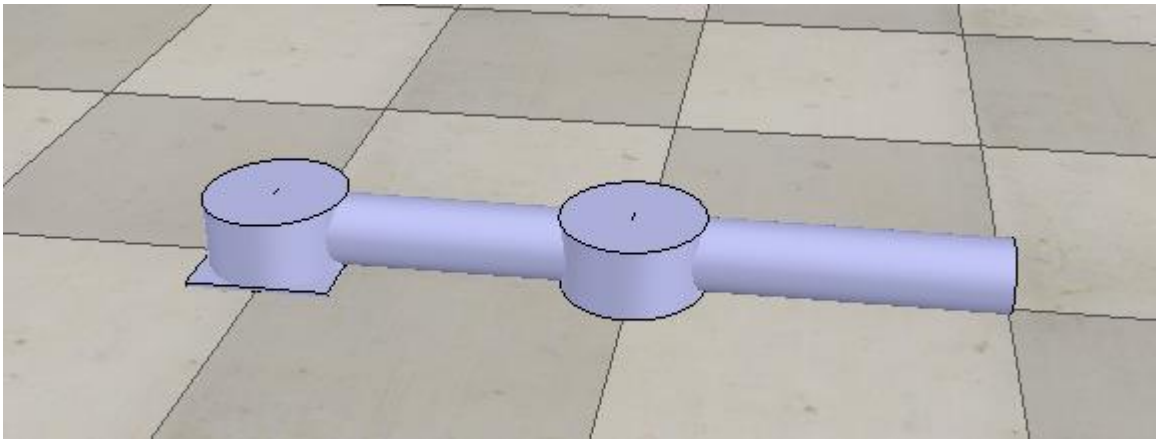

Figure 2-1.Robot 2 GDL en V-REP

Como ya he mencionado hay ciertos temas que no están explicados o no con el peso que se merecen. Un ejemplo claro es la jerarquización de los elementos al construir tu propio robot. Para nuestro robot sería de la siguiente forma:

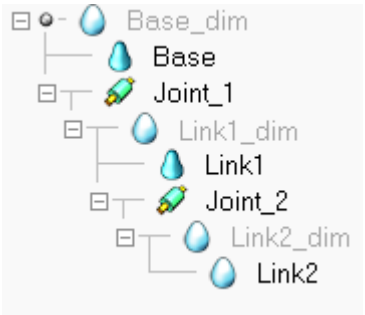

Figure 2-2.Jerarquía del modelo

Tiene que existir una base de la que dependa todo. Cada articulación debe de estar unido a un solo cuerpo, si ponemos dos cuerpos que dependan que él, el segundo no notara la presencia del motor y el robot no se comportará como se espera. Existen distintas opciones para combinar dos cuerpos en un solo y ya si hacer una correcta jerarquización.

2.2 Control dinámico

Vamos a mandar una posición de referencia para cada articulación a través de un slider a cada articulación. El diagram de control que se sigue es este:

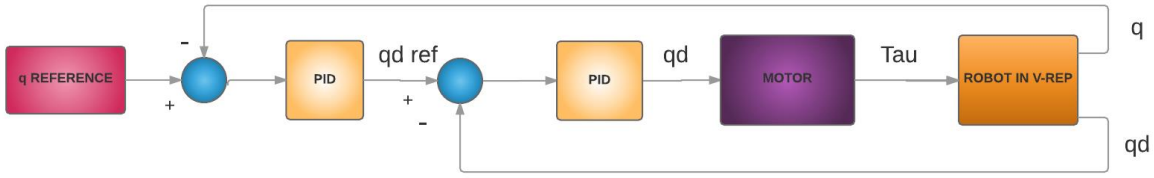

Figure 2-3.Diagrama de control dinámico

Este ejemplo correctamente implementado se encuentra implementado en la escena *2GOF_dynamicslider.ttt*.

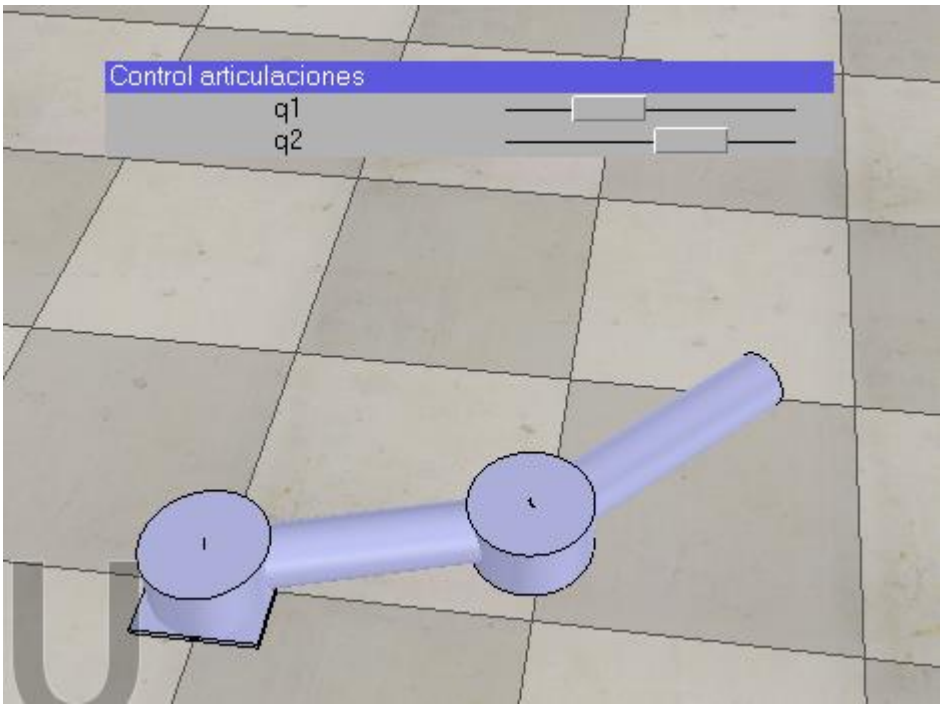

Figure 2-4.Demo control dinámico con sliders

2.3 Control cinemático

Nuestra meta en este punto es obtener el jacobiano para una determinada posición y orientación que determinamos con un punto (*dummy*):

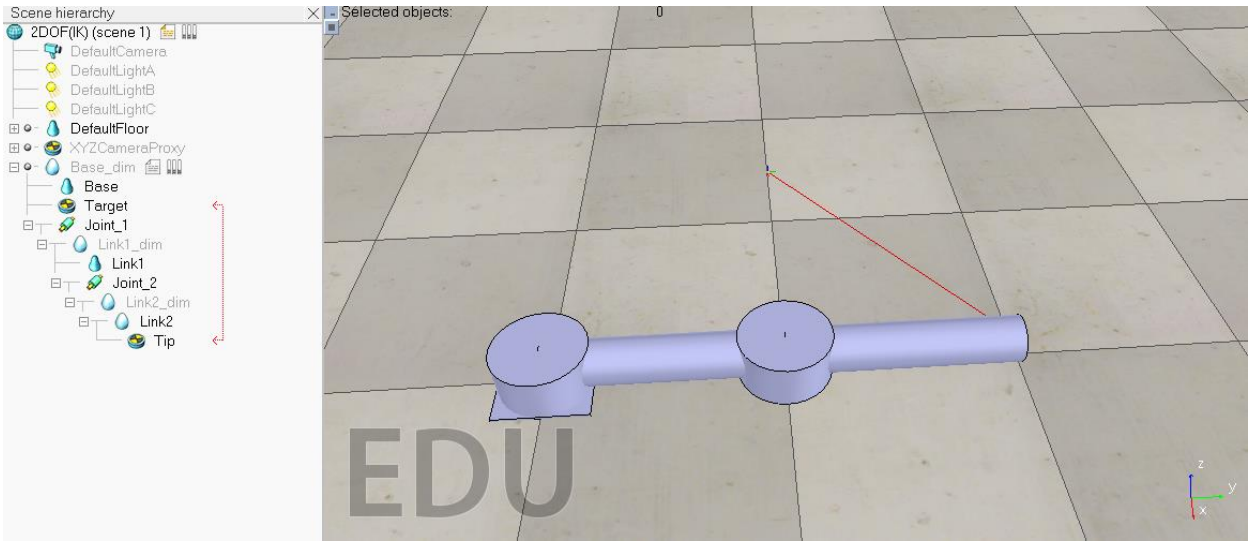

Figure 2-5.Distribución de los puntos en la jerarquía

Una vez que se llegue a la posición objetivo, aparecerá el jacobiano por pantalla. Así, se pueden poner distintas posiciones y obtener los jacobianos para interpolarlos y obtener el simbólico. El esquema de control que hemos seguido es el siguiente:

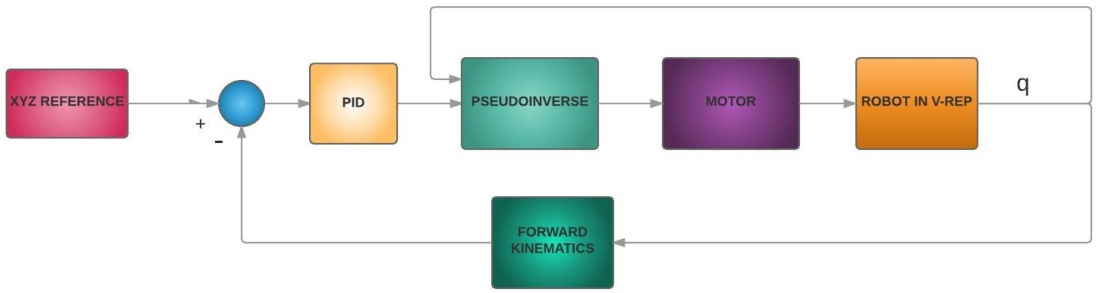

Figure 2-6. Diagrama de control cinemático

Esto se encuentra implementado en la escena *2GOF_IK.ttt* scene.

3 ROBOT HUMANOIDE EN V-REP®

En esta sección vamos a estudiar el caso de un robot humanoid (Biolid Type A) que se encuentra en el departamento Ingeniería de Sistemas y Automática de nuestra escuela. Es interesante centrarno en esta confiración para escapar de las configuraciones vistas a lo largo el grado.

3.1 Construcción

La mayoría de formas utilizadas han sido proporcionadas por Rafael Matínez Márquez en su trabajo fin de grado *Diseño de un sistema de vuelo de un robot humanoide*. Todas las piezas han sido diseñadas con *CATIA V5* y guadadas en el formato predilecto de la aplicación, *.stl* . Todas las formas se encuentran en la carpeta *Piezas*. Teniendo en cuenta las mismas consideraciones que para construir el robot anterior construimos a nuestro humanoide:

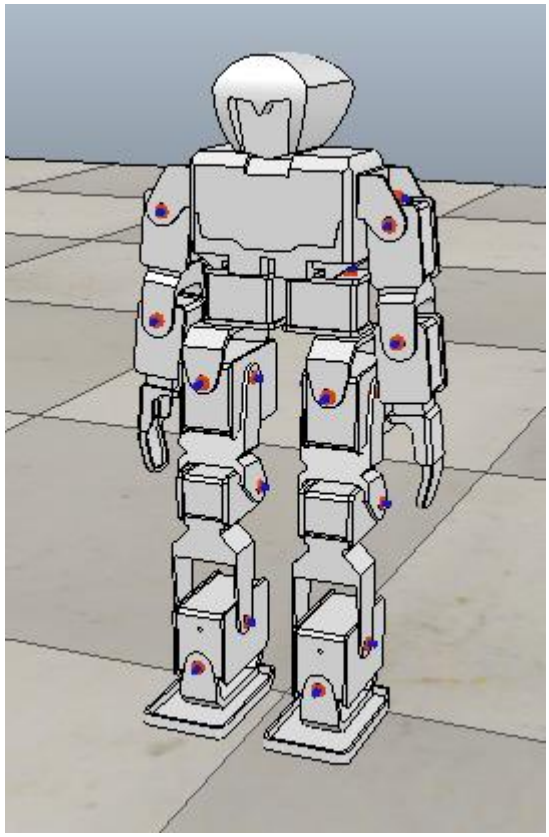

Figure 3-1.Robot humanoid en V-REP

3.2 Control dinámico

Vamos a controlar dinámicamente las articulaciones de los brazos y la de una pierna pasándole la posición mediante sliders como en el caso anterior:

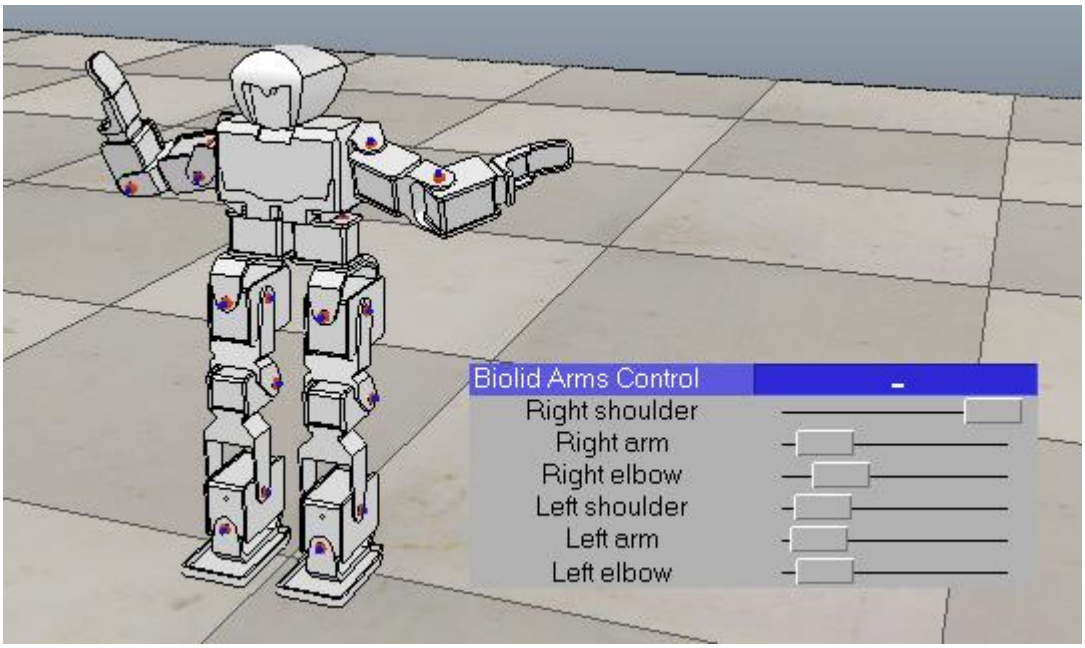

Figure 3-2.Demo control de brazos

.

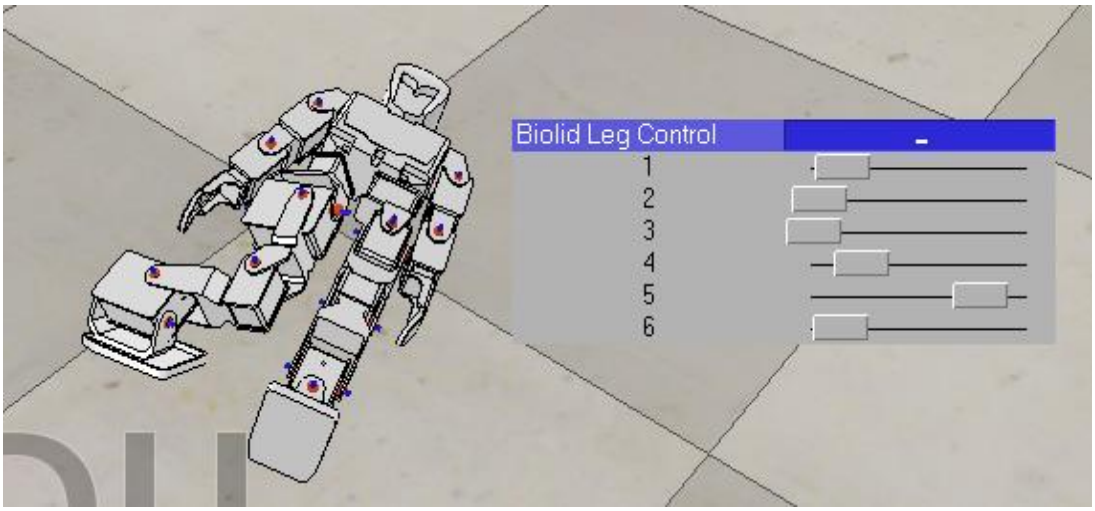

Figure 3-3.Demo control de pierna

Ambos ejemplos se encuentran implementados respectivamente en las escenas *Arms_control.ttt* y *Leg_control.ttt* scene respectivamente.

3.3 Control en bucle abierto para andar

La limitación importante de no poder modificar directamente el control cinemático estará muy presente en este punto. Al no poder controlar directamente elementos de estabilidad típicos de esta configuración como son la cadera y las piernas, vamos a abarcar el problema de dos maneras distintas.

3.3.1 Control cinemático: Cadena IK con puntos

La idea básica es establecer dos cadenas cinemáticas para cada una de las piernas. Cada cadena tiene asociada dos puntos: un origen (situado en el centro de la planta del pie) y un destino (situado en el extremo de la punta del pie). Vamos a hacer que cada pie siga un camino recto a una velocidad determinada que podemos

modificar. Cada cadena cinemática se moverá siguiendo el movimiento de los puntos a lo largo del camino. A lo largo de la ejecución de código iremos obteniendo la posición absoluta del robot y la del punto en el camino para determinar correctamente el paso. Esta idea viene expuesta en varios temas distintos del foro mencionado en la introducción. Además, viene implementada correctamente en un ejemplo que trate el propio programa *Asti robot*. En este ejemplo crea una trayectoria precisa para que exista cierto balanceo en la cadera de modo que parezca un control en bucle cerrado cuando no lo es. Para nuestro robot la cadera no presenta un gran apoyo, utiliza la más bien la inclinación de los tobillos y este método no tiene en cuenta eso. Del ejemplo antes mencionado también se extrae que hay que crear un camino muy específico y muy detallado, lo cual no resulta muy interesante desde el punto de vista del control. Por este motivo, no resulta muy interesante comentar a fondo este ejemplo. De todos modos, adjunto la implementación comentada con alto grado de detalles a nivel de código en la escena *Biolid walk ik.ttt*.

3.3.2 Control dinámico: Máquina de estados

En este caso voy a hacer una pequeña máquina de estados el que cada estado es una posición distinta del paso. El cambio entre estados vendrá condicionado en el momento en el que cierta articulación llegue a la referencia establecida. Las distintas posiciones que voy a establecer son:

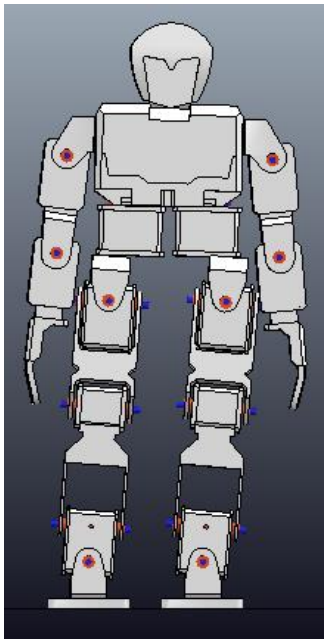

Figure 3-4.Inclinación necesaria para no perder equilibrio

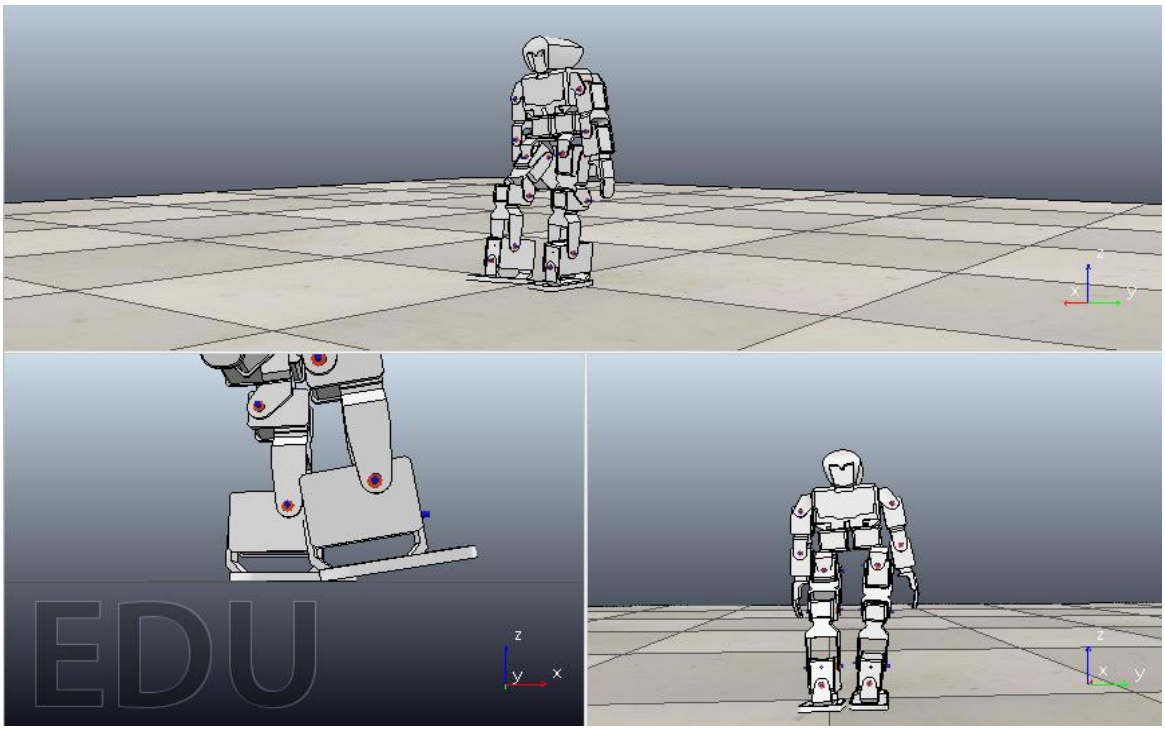

Figure 3-5.Comienzo del paso

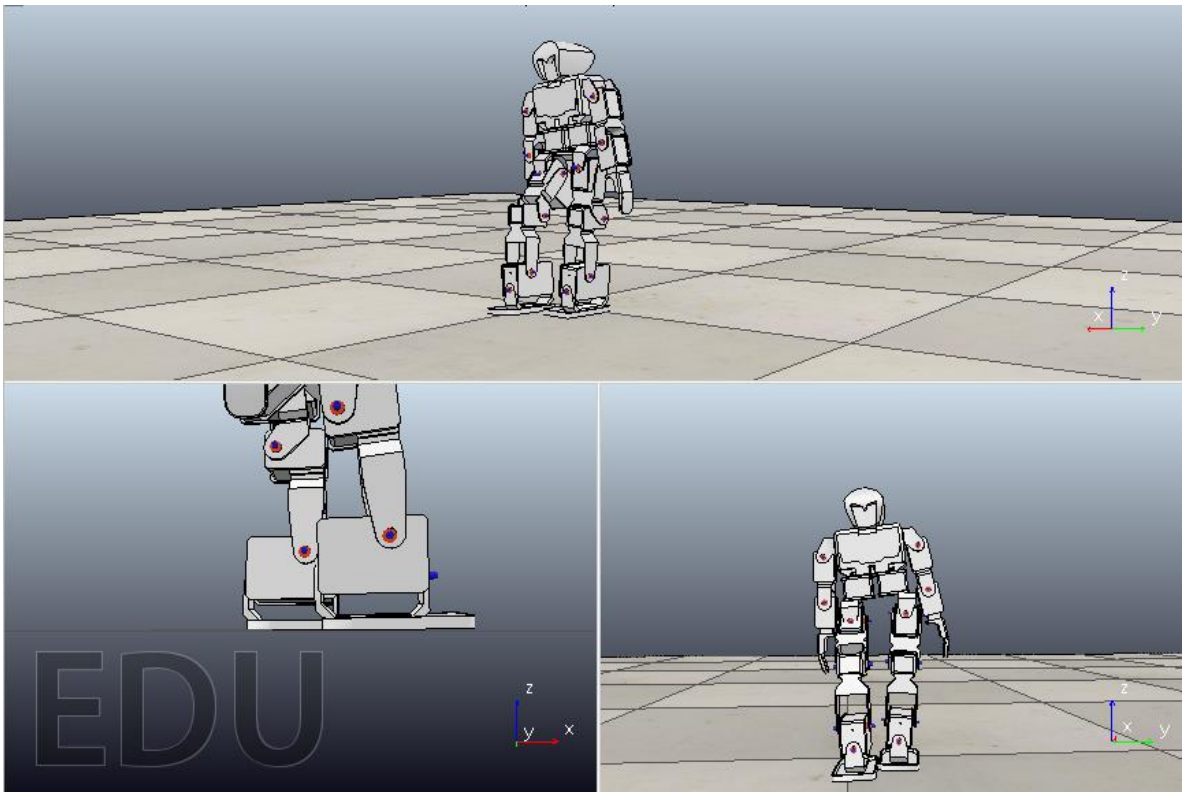

Figure 3-6.Paso terminado

A nivel de código es simple, pero algo tedioso y nada interesante desde el punto de vista del control y por eso no lo voy a extender a más pasos. La implementación se encuentra en la escena *Biolid walk reference.ttt*. Llegados a este punto debo de buscar una alternativa más seria y más eficaz para el control. Matlab parece un buen candidato para desempeñar esta tarea. En el siguiente capítulo veremos cómo establecer la conexión entre Matlab y V-REP.

4 CONTROL DINÁMICO EN MATLAB®

Llegados a este punto vamos a analizar el modelo dinámico de un robot desde Matlba y comprarlo en V-REP. En primer lugar, vamos a identificar el modelo del motor de V-REP.

El diagrama de comunicación que vamos a seguir va a ser el siguiente:

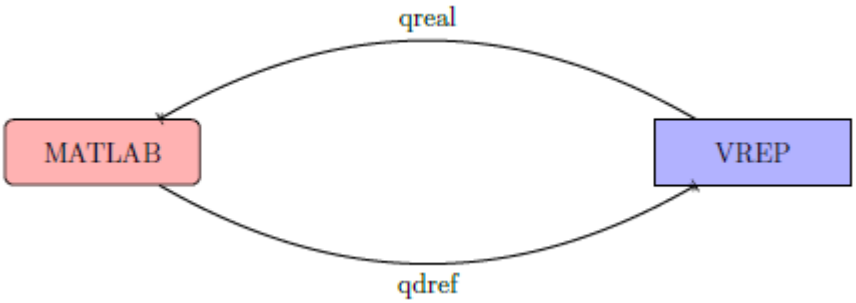

Figure 4-1.Diagrama de comunicación

La función de Matlab que se encarga de mandar la velocidad articular es *VREPROBOT.m*. Esta función recibe esta referencia e inicia la comunicación para mandarla al servidor. Esta misma función obtiene el valor de las posiciones articulares al final del movimiento solicitándolas al servidor.

4.1 Modelo del motor de V-REP

Para identificar la function de transferencia y los parámetros del motor, vamos a centrarnos en un robot con un eslabón:

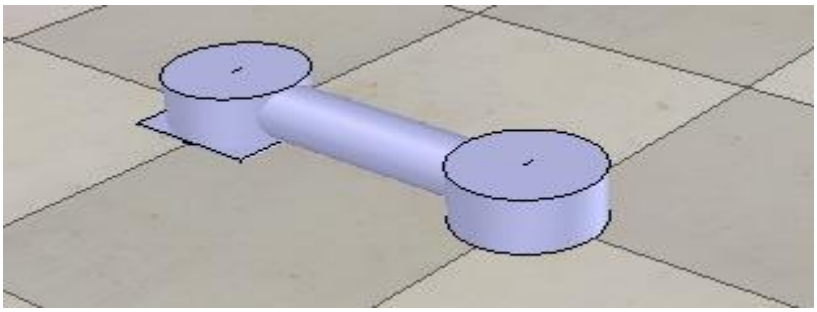

Figure 4-2.Robot de un grado de libertad

En primer lugar, desconocemos la relación de la inercia del eslabón frente a la del motor. Necesitamos hacer varios experimentos con diferentes ordes de inercia del eslabón. Simulamos en bucle abierto desde Matlab para diferentes inercias del eslabón en V-REP:

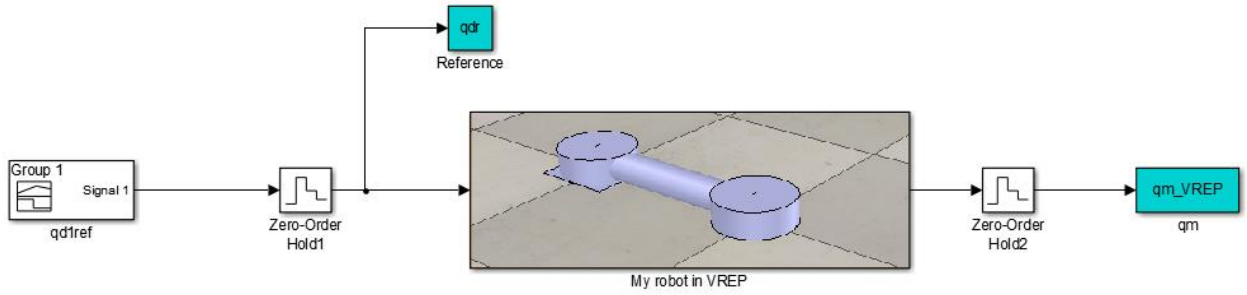

Figure 4-3.Simulación en bucle abierto

Vamos a hacer experimentos escalonados con diferentes órdenes de inercia. Vamos a ir desde e-2 a e3 incrementando un orden en cada simulación. Tenemos los diferentes valores de la velocidad articular y del par para los distintos órdenes de inercia y para la misma referencia en velocidad:

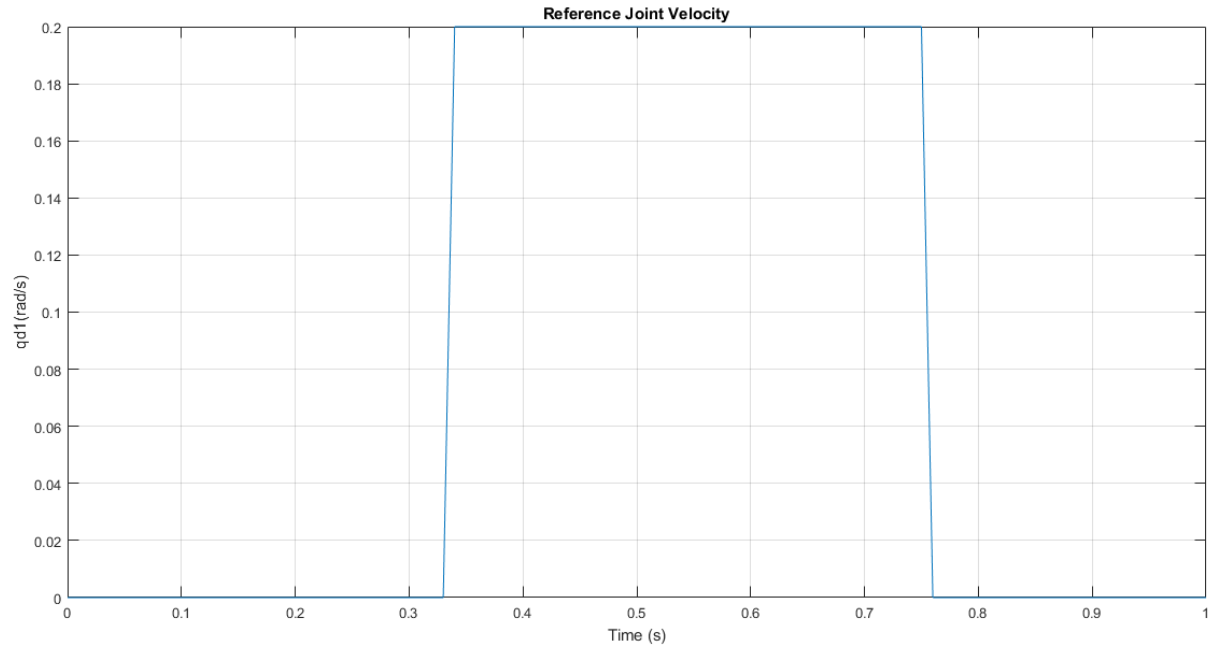

Figure 4-4.Velocidad de referencia

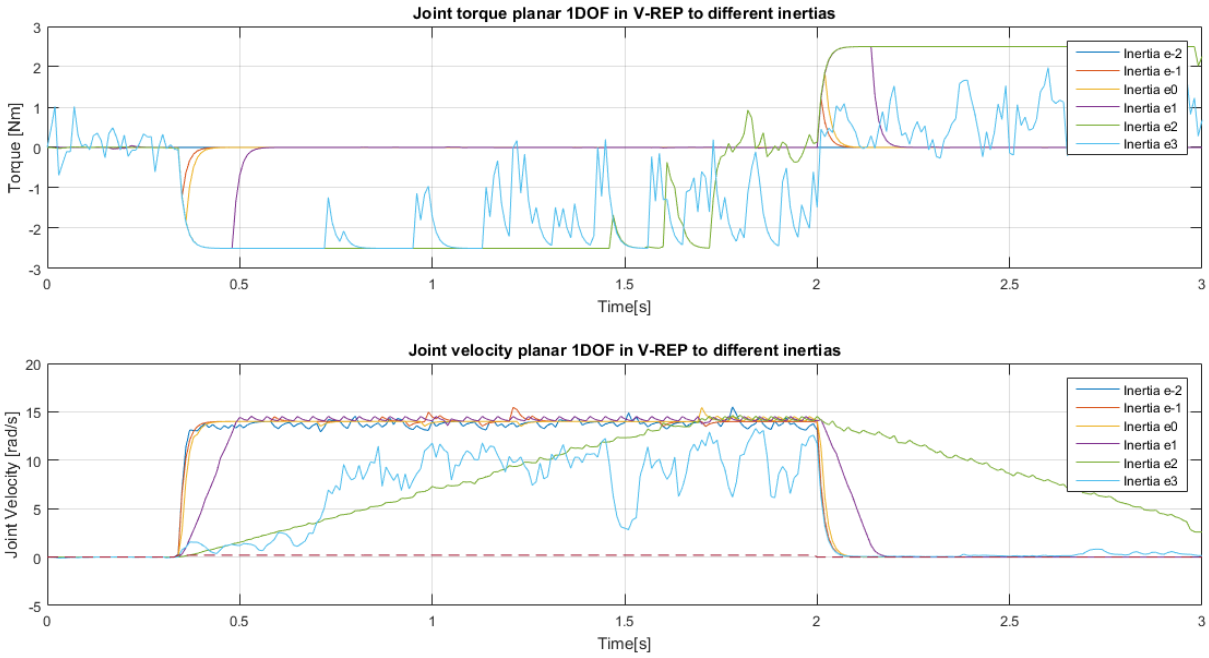

Figure 4-5.Par y velocidad articular para el robot de un grado de libertad para diferentes inercias

De la simulación se puede extraer que el motor es controlado por un proporcional. Para identificar la fnción de tranferencia y los parámetros del motor tenemos que:

$$(Jm + J_{LINK}) \cdot qdd + B \cdot qd = Torque$$

$$G(s) = \frac{1}{(Jm + J_{LINK}) \cdot s + B}$$

$$Torque = -Kp \cdot (qd - qd_{REF}); N = \frac{qd}{qd_{REF}} \qquad (1)$$

$$Torque = Kp \cdot qd \cdot \left(\frac{1}{N} - 1\right)$$

Donde *Jm* es la inercia del motor [Nm], *Jlink* es la inercia del eslabón [Nm], *B* es la viscosidad del [Nm/(rad/s)], N es la relación de transmisión y Kp es la acción proporcional del motor.

De los reultados de la simulaciones obtenemos que:

Inertia=0.0765 [Nm]

Viscosity= 6.9577 [Nm/(rad/s)]

Reduction factor=70

$$Kp \cdot \left(\frac{1}{N} - 1\right) = 1.4e7$$

$$G(s) = Kp \cdot \left(\frac{1}{N} - 1 \right) \cdot \frac{0.1437}{0.011s + 1} \qquad (2)$$

Vamos a constuir nuestro modelo con la herramienta que nos proporciona el módulo de Peter Corke y para nuestra estimación obtenemos el siguiente resultado:

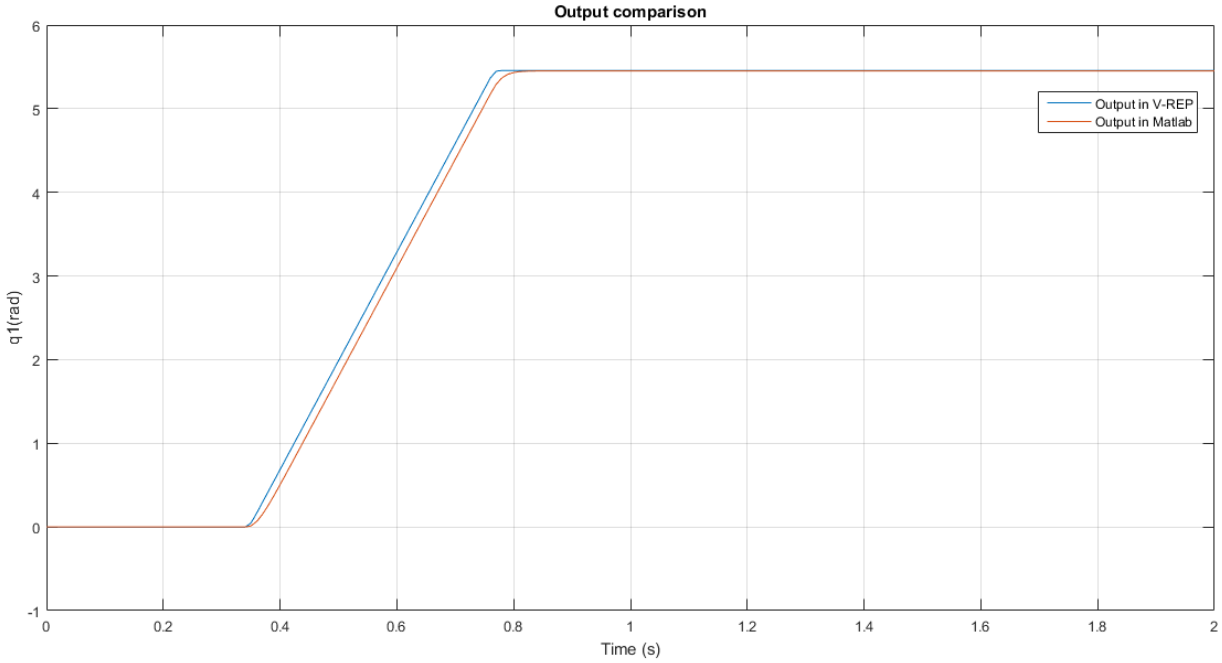

Figure 4-6.Compación de la posicón articular

4.2 2 DOF dynamic control

El diagrama de control que buscamos es el expuesto en el segundo capítulo.

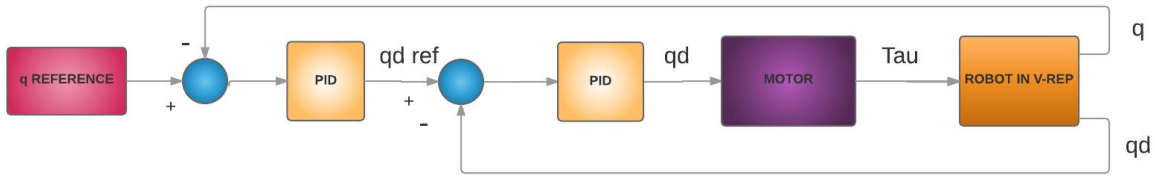

Figure 4-7.Diagrama de control dinámico

Hay que observar que no hay ningún comando API para aplicar directamente el par al modelo, sólo puedes obtenerlo. Por lo que el modelo del motor hay que simularlo en V-REP. Del mismo modo, no se puede extraer velocidad articular, sólo aplicarla por lo que habrá que derivar la posición. De este modo, se podría aplicar el siguiente esquema en Matlab para nuestro control discreto. Además, tampoco podemos modificar el segundo control, porque habría que descativar una serie de opciones, no permitiendo ejecutar correctamente los comandos API que mandamos desde Matlab. Finalmente, nuestro diagrama de control se queda de la siguiente forma:

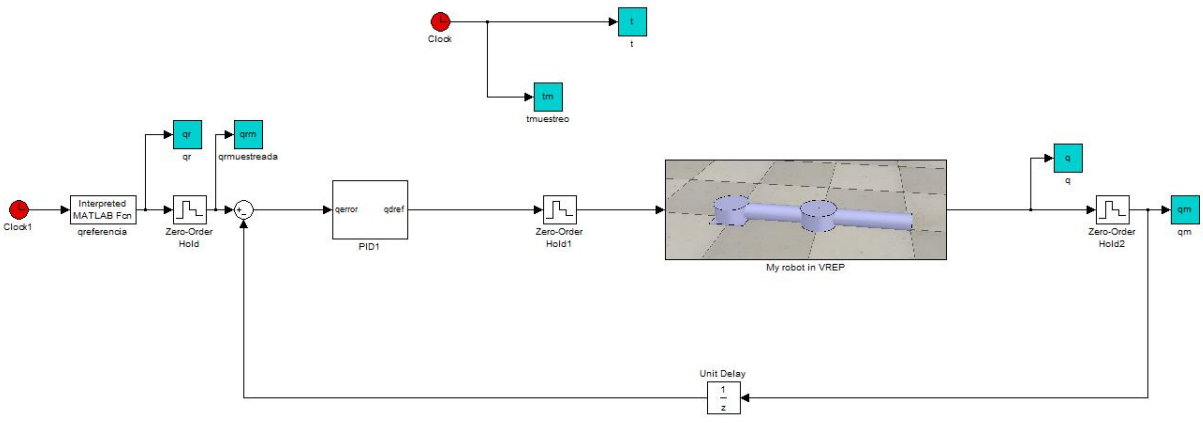

Figure 4-8.Control dinámico final en Matlab

Aplico una trayectoria articular para ambas articulaciones que comenzará en 0 [rad] hasta 0.8 [rad]. Implementamos un controlador P discreto con ganancia 0.2 [rad/s/rad].

A continuación, implementamos nuestro robot de dos grados de libertad del mismo modo que hicimos anteriormente con el módulo de Pter Corke y respresentamos los resultados frente a los de V-REP:

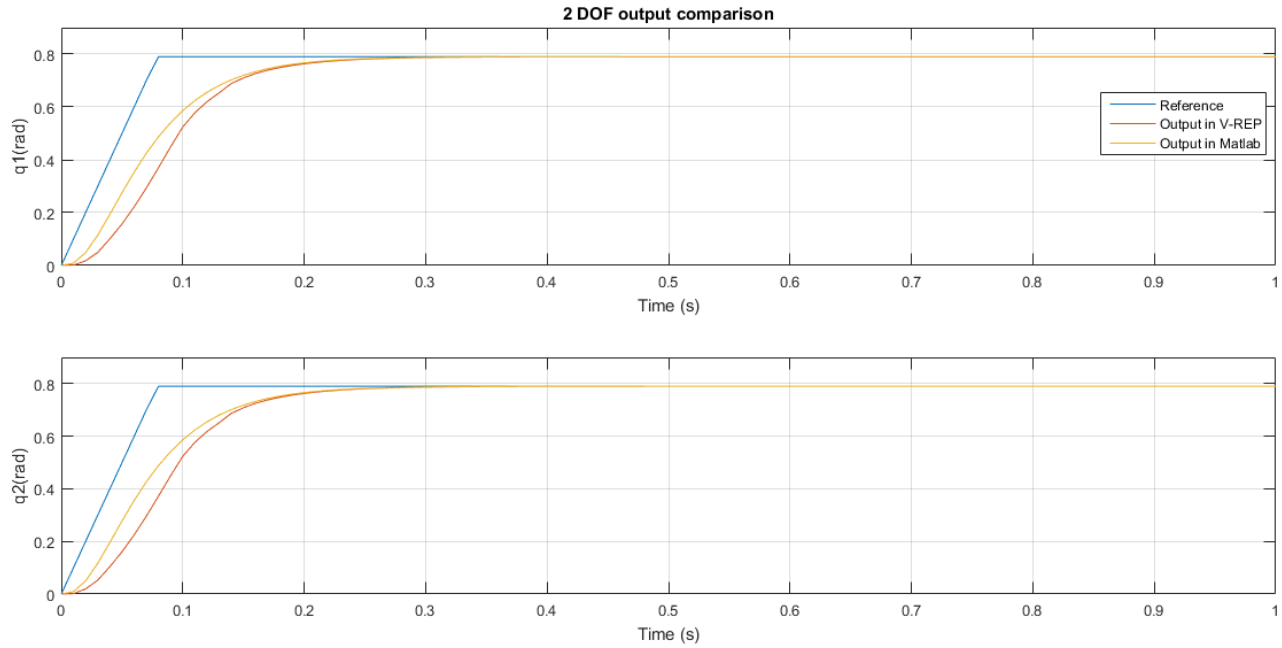

Figure 4-9. Compación de la posicón articular

En conclusion, no ha sido fácil estimar el modelo dinámico de V-REP, pero hemos obtenido una muy Buena estimación y hemos comprendido las dinámicas de V-REP al complete.

5 CONTROL CINEMÁTICO DESDE MATLAB

5.1 Control cinemático mediante la Pseudoinversa

En este punto vamos a abrir el campo de opciones que nos daba el programa y vamos a crear nuestro propio control cinemático mediante Pseudoinversa. El diagrama que debemos de tener en la cabeza es el siguiente:

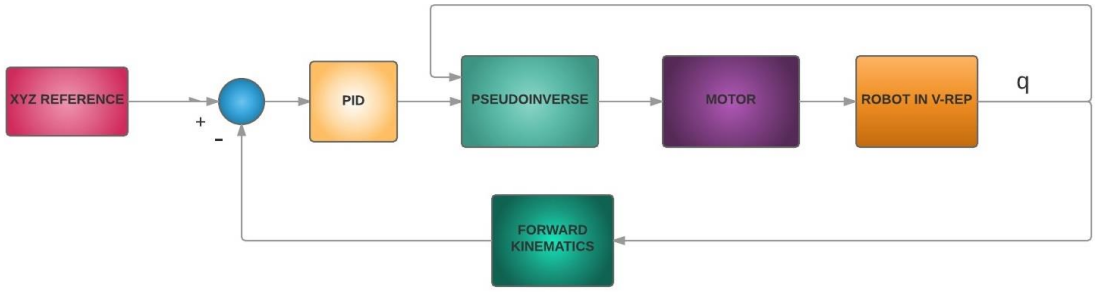

Figure 5-1.Diagrama de control cinemático

Como ya he mencionado antes, el modelo del motor hay que simularlo en V-REP. Este mismo esquema planteadoen Matlab queda así:

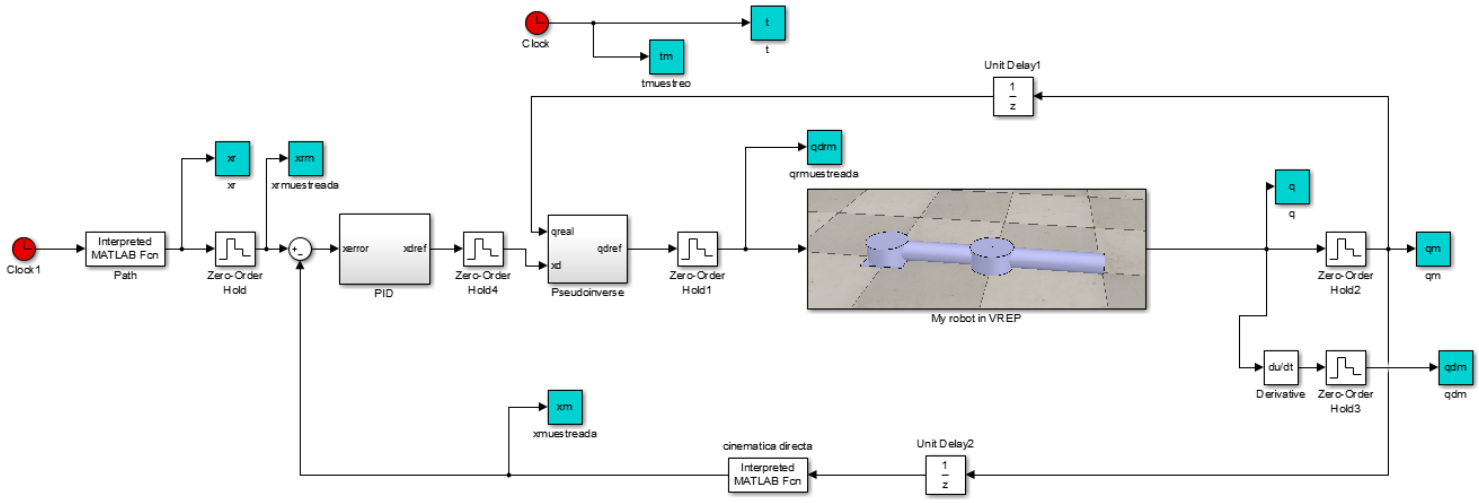

Figure 5-2.Control cinemático en Matlab

Para comprobar su funcionamiento, voy a hacer una simulación en la que reciba como referencia una trayectoria circular. El control de posición que voy a implementar sera un PI con Ti=100 [s] y Kp=0.5[(m/s)/m]. Finalemente, los resultados para la simulación completa:

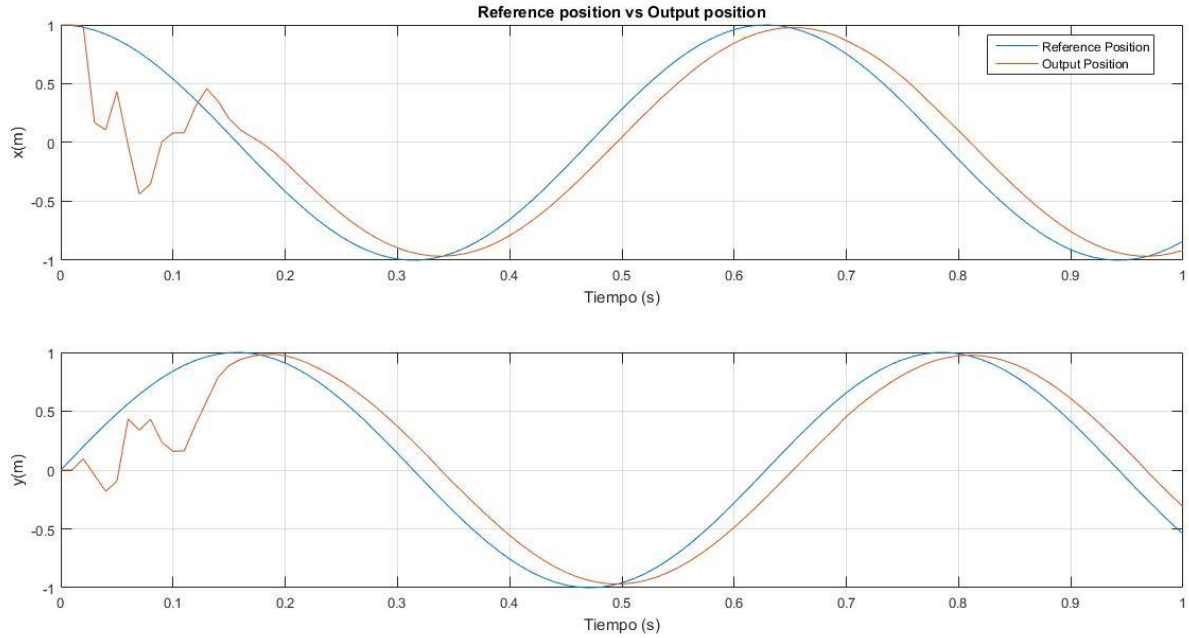

Figure 5-3.Referencia frente a la salidad de la posición XY

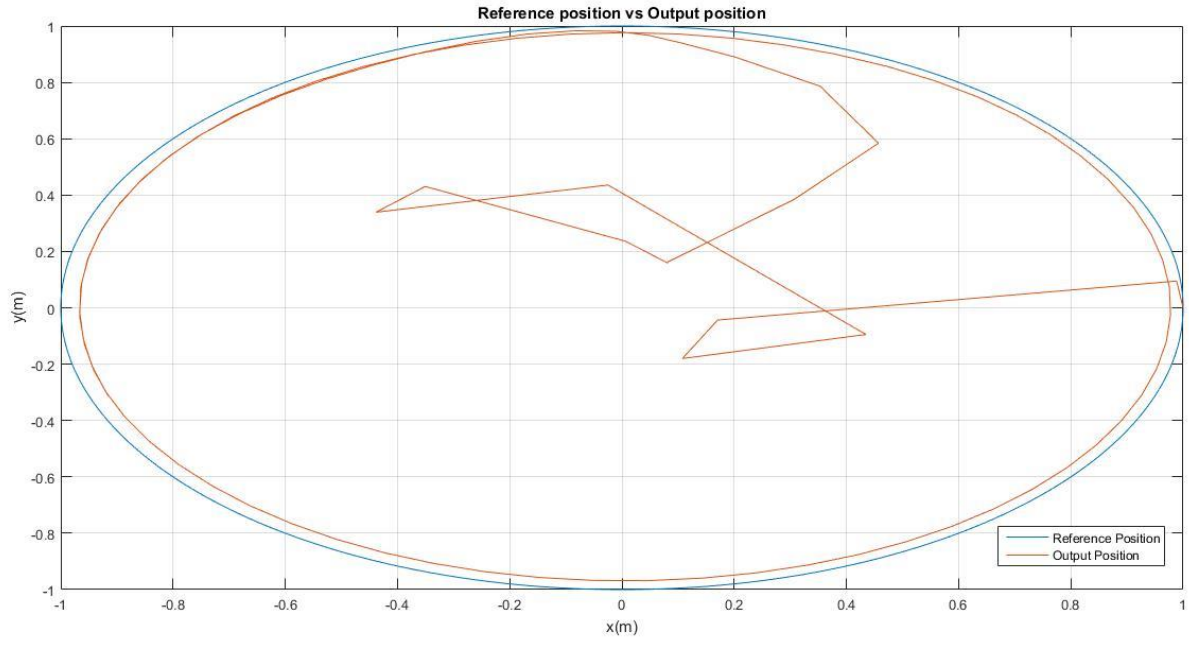

Figure 5-4. Referencia frente a la salidad de la posición XY

5.2 Control cinemático mediante la DLS

Voy a buscar los mismos objectivos que con la Pseudoinversa, pero utilizando la DLS. En este caso, voy a cambiar a un robot redundante como es un robot planar de 4 GDL:

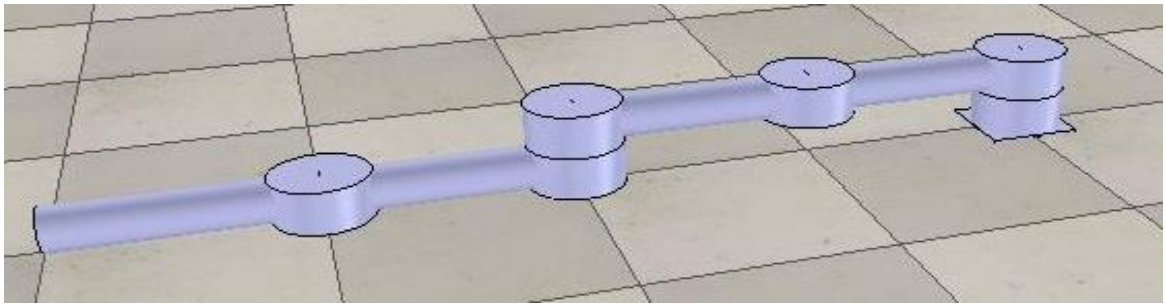

Figure 5-5.Robot planar con 4 GDL en V-REP

Con esta nueva configuración tenemos que prestart especial atención a estar cerca de una posción singular. Para obtener resultados aceptables vamos a mejorar el algortimo de la DLS que V-REP trae. El algoritmo de la DLS viene dado por:

$$J_{DLS}^{-1} := J^T \cdot (J \cdot J^T + k^2 \cdot I)^{-1} \qquad (1)$$

Donde *k* es el factor de amortiguamiento e *I* es la matriz identidad. En nuestro caso, k no sera un valor estático como en V-REP. Este factor tundra un valor máximo k_0 en configuraciones singulares y cero en otro caso. De este modo, la variable se debe adaptar a las configuraciones singulares y sus vecindades:

$$k := k_0 \cdot \exp\left(-\frac{\det(J \cdot J^T)}{2 \cdot \varepsilon^2}\right) \qquad (2)$$

Donde ε es un factor de forma que varía entre 0 y 1.

Implementamos la misma estructura de control que en el caso anterior, sólo que modificamos el algoritmo de cálculo del jacobiano. El control de posición que voy a implementar sera un PI con Ti=100 [s] y Kp=1.2[(m/s)/m]. El algoritmo DLS utilizará un factor de amortiguamiento igual a 0.9 y factor de forma 0.9. Finalmente, los resultados completos de simulación son:

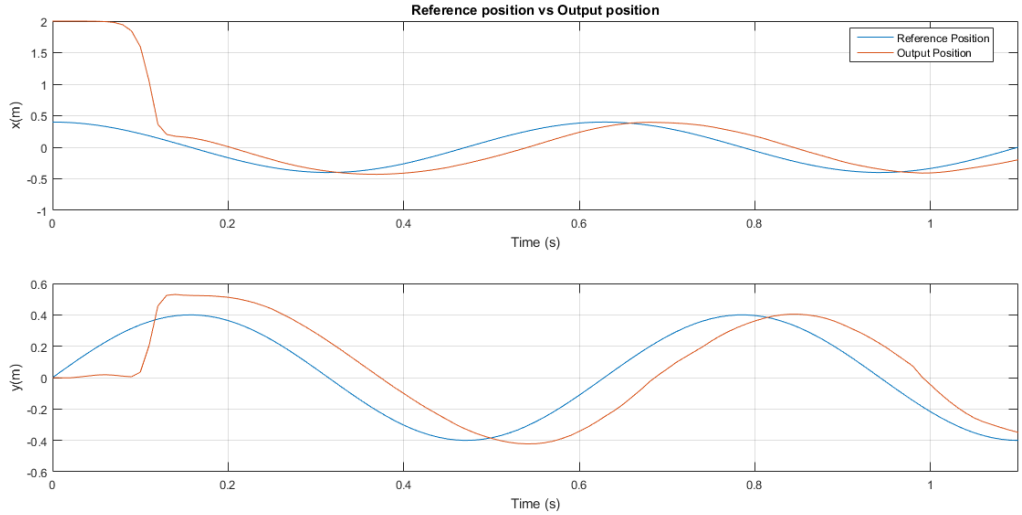

Figure 5-6. Referencia frente a la salidad de la posición XY

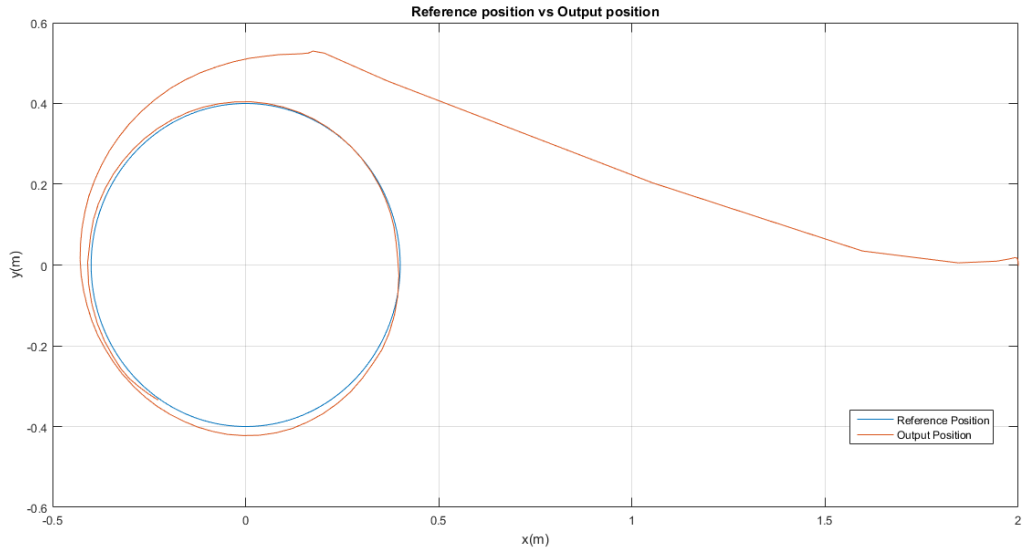

Figure 5-7. Referencia frente a la salidad de la posición XY

6 SENSOR KINECT Y V-REP

Nuestra última experiencia con V-REP sera conectar el sensor Kinect de la X-BOX a V-REP para manejar cualquier robot con el moviemiento de nuestro cuerpo.

Figure 6-1.Sensor Kinect

Conectarlo con V-REP no es nada sencillo, porque las aplicaciones que utiliza están obsoletas y son muy difíciles de encontrar. Hay mucha gente que tiene este mismo problema en el foro. Investigando de una manera más exhaustiva, he encontrado las aplicaciones necesarias y ya estamos listas para probarlo en V-REP. Añadimos a una escena vacía un nuevo elemento denominado *interface to kinect.ttm*. Este element ejecuta a aplicación *KinectServer.exe* que sigue el siguiente diagrama:

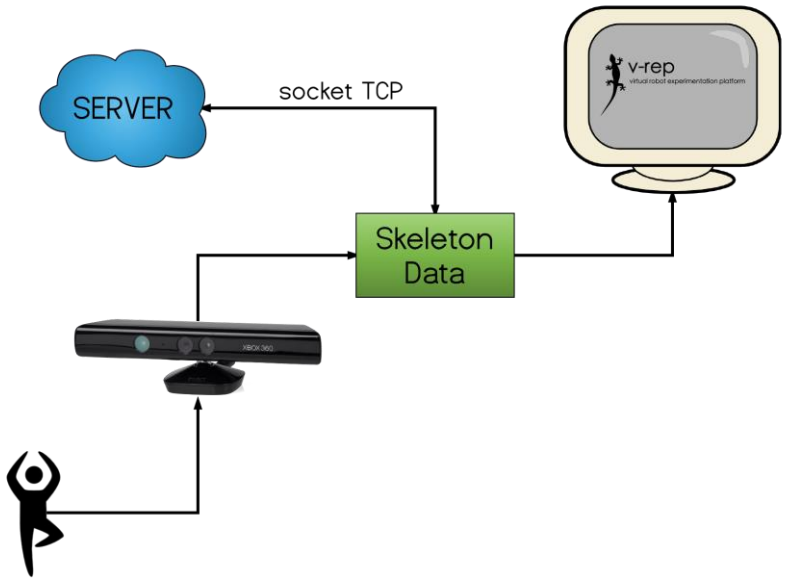

Figure 6-2.Diagrama de conexión

Si corremos esta escena, vemos que simula correctamente:

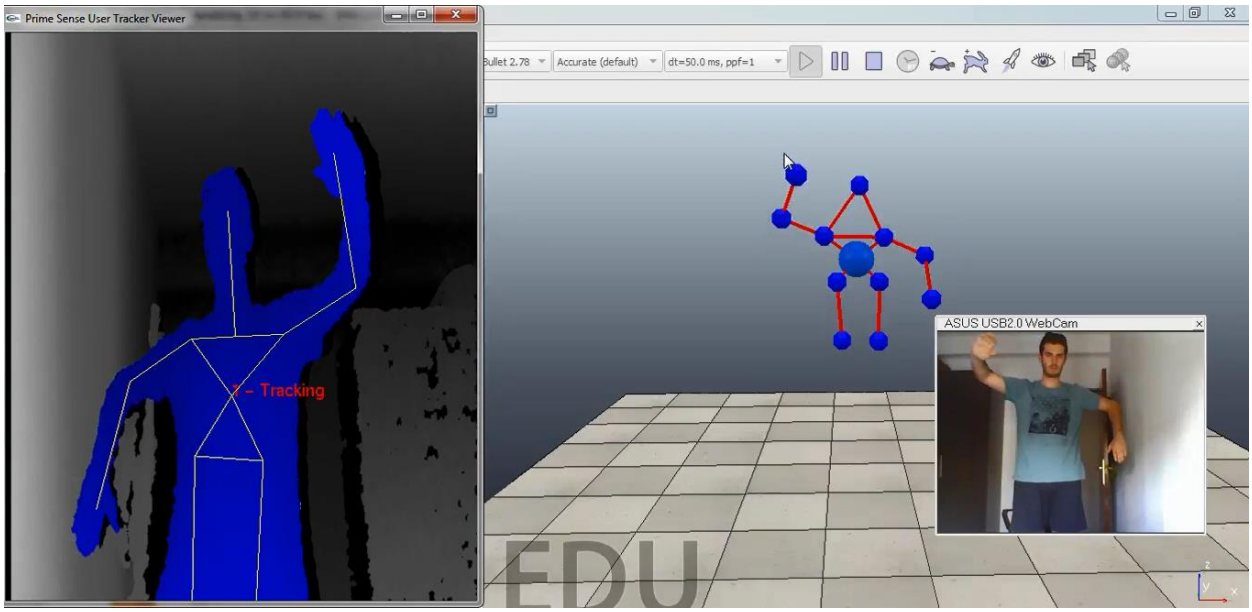

Figure 6-3.Simulación Kinect

Ya que sabemos sacar la información del esqueleto, lo que sigue es asociar cada articulación de nuestro humanoide con las de este esqueleto. En un ejemplo proporcionado por el propio V-REP, *astiKinectControl.ttt* está implementado esto. Sin embargo, el código tiene algunos errores, como que hay ciertas variables declras como locales cuando son globales. Un calro ejemplo de este ocurre en las líneas 233 y 234, Corrigiendo esto, corremos nuestra simulación. Finalmente, obtenemos el resultado que buscábamos*:*

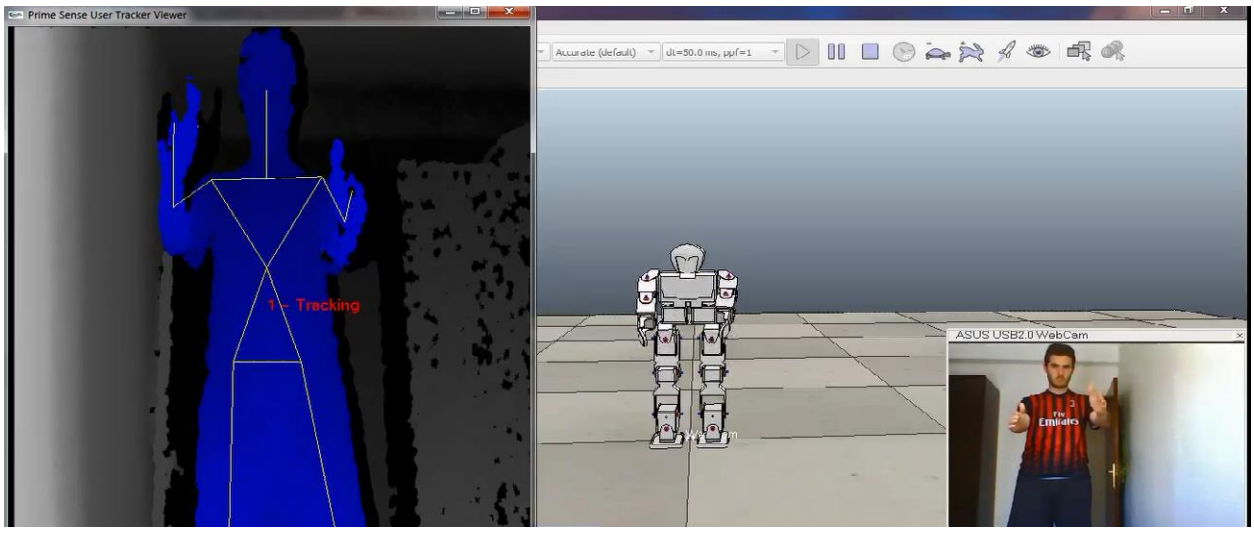

Figure 6-4.Simulación Kinect con Robot Humanoide

Esta escena se encuentra correctamente implementada en *Biolid_Kinect_Control.ttt*