UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

INTERFAZ DE PROGRAMACIÓN DE APLICACIONES PARA LOCOMOCIÓN DE ROBOT HUMANOIDE BIOLOID

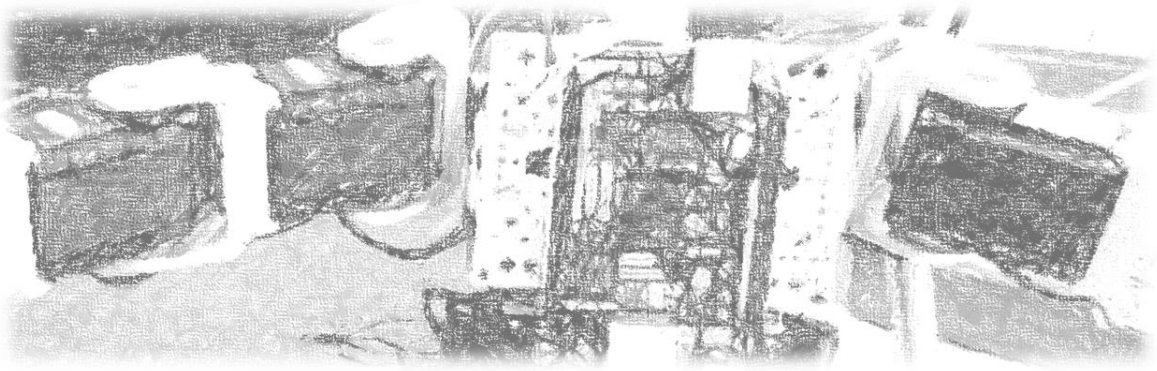
TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN AUTOMÁTICA E INFORMÁTICA INDUSTRIAL

Autor: Juan Carlos Brenes Torres

Tutor: Francisco Blanes Noguera

Septiembre 2016



"La imaginación frecuentemente nos llevará a mundos que jamás fueron. Pero sin ella, no iremos a ningún lado"- Carl Sagan

Abstract

Humanoid robots lay deep in the imagination of current society. The premise of a helper for dull or dangerous tasks have captivated the efforts of researchers and enthusiasts for decades. Their study represent a broad spectrum of challenges to be solved, and the work presented here aids solving some of the technical challenges for Bioloid humanoid robots.

In this Master's Final Project we present the design, implementation and testing of an Application Programming Interface for locomotion of Bioloid humanoid robots with the use of a BeagleBone Black embedded system.

The main contribution of this work is the development of a novel integration of technologies in order to control Bioloid robots with an embedded system. Up to date, projects on this robotics platform used the manufacturer proprietary controller and software. With the work generated, future investigations could benefit from the advantages of using an embedded system, like extended computation capacity and easy integration with cameras and IMUs.

This project presents the development done to solve the inverse cinematics problem for a Bioloid robot, the creation of a trajectories generation function and the grouping of movements into behaviors. It's also shown the improvement of the robot's foot to sense pressure and detect obstacles. The sum of this functions compose the Application Programming Interface for the locomotion of the Bioloid robot.

Contenidos

Índice de ilustraciones

Índice de tablas

Capítulo 1. Introducción

En el presente informe se plasman los conocimientos aplicados, los retos encontrados y las soluciones implementadas para el Trabajo de Fin de Máster de la titulación de Máster Universitario en Automática e Informática Industrial.

En el capítulo 1 se describe la motivación del proyecto, los objetivos, la metodología empleada, la contribución del trabajo, las tecnologías utilizadas y la estructura del reporte a utilizarse.

1.1. Motivación del proyecto

La idea de poder replicar las funcionalidades del movimiento del ser humano en una máquina ha capturado la mente y la imaginación de las personas por décadas. La ciencia ficción se encargó de materializar esa visión en la figura de una máquina con forma de humano que ayude en las labores domésticas o sea compañero de aventuras intergalácticas; incluso una visión negativa en que se convierte en enemigo de los humanos y es necesario su aniquilación luego de una gran batalla.

Los robots humanoides están firmemente arraigados en el imaginario de la sociedad actual y su estudio presenta una amplia gama de retos para investigadores y desarrolladores, pero esta dificultad se ve recompensada por el acercamiento cada vez mayor hacia esa situación de ciencia ficción de humanos conviviendo con robots [1].

En el estado actual de la tecnología los robots humanoides se ubican principalmente en ámbitos académicos para desarrollo de nuevas tecnologías y en el ámbito comercial para futuras oportunidades lucrativas y demostraciones mercadotécnicas. Sin embargo, estudiantes, investigadores y aficionados alrededor del mundo trabajan con ahínco en resolver los retos que acerquen los robots humanoides a nuevos campos de aplicación como el cuidado de adultos mayores y ejecución de tareas del hogar [2]. El futuro cercano se vislumbra con robots humanoides presentes en ambientes de manufactura, labores de rescate y viajes al espacio; donde el tener espacios y herramientas diseñadas para humanos justifica la presencia de un robot de forma humanoide [3].

En base a todas estas premisas se considera que el estudio de los robots humanoides y en específico el robot Bioloid utilizado en este trabajo de Fin de Máster está justificado y es de un valor importante para el avance en tecnologías de este campo.

1.2. Objetivo general

Desarrollar una interfaz de programación de aplicaciones (API por sus siglas en inglés) para la locomoción de un robot humanoide Bioloid que se ejecute en un sistema empotrado BeagleBone Black.

1.3. Objetivos específicos

1. Investigar el estado del arte en locomoción de robots humanoides Bioloid.
2. Desarrollar rutinas informáticas que permitan resolver la cinemática inversa de los brazos del robot y también de sus piernas y caderas.
3. Implementar generadores de trayectorias para poder crear secuencias de movimientos entre distintos puntos en cada extremidad.
4. Agrupar distintos tipos de movimientos básicos en comportamientos y crear una lógica para el cambio entre comportamientos.
5. Trabajar en la implantación de sensores en la planta del pie del robot para tener información de presión y proximidad de obstáculos, y con ello poder ajustar el tipo de movimiento a realizar.
6. Validar el sistema desarrollado en la competición nacional de robots humanoides CEABOT que se lleva a cabo como parte de las Jornadas de Automática 2016 del Comité Español de Automática.

1.4. Metodología

La metodología empleada en el trabajo consistió de varias etapas, primeramente se documentó mediante publicaciones y libros de texto las técnicas de resolución del problema de cinemática inversa en robots humanoides. Luego se procedió a diseñar las rutinas de cinemática inversa y trayectorias, las cuales se probaron mediante simulaciones. Posteriormente se realizó la implementación de los procedimientos en el lenguaje y controlador utilizado por el robot.

La implementación se probó mediante pequeñas pruebas controladas con el sistema Bioloid colgado de un marco de apoyo para evitar caídas. Luego en un ambiente de prueba controlado se probaron actividades como actividades como caminar o subir escalones. Por último se analizaron los resultados y se realizaron algunas correcciones al diseño general, y con esto se generaron conclusiones acerca de la aplicación desarrollada.

1.5. Tecnologías usadas

En este trabajo se integraron múltiples tecnologías para poder lograr los objetivos planteados. Para la fase de diseño se utilizó el programa Matlab con la herramienta Robotics Toolbox, con la cual se generaron y visualizaron las simulaciones de los movimientos de brazos y piernas del robot.

Se utilizó un robot Bioloid Premium con morfología humanoide (Figura 1), 18 grados de libertad (3 grados de libertad por brazo y 6 por pierna), servomotores Dynamixel AX-12 (Figura 2) y batería de Litio de 12V.

Figura 1. Robot Bioloid Premium A

Figura 2. Servomotor Dynamixel AX-12

Para el control del robot se utilizó un sistema empotrado BeagleBone Black (Figura 3) que posee un sistema operativo Linux. El programa a ejecutarse se desarrolló usando librerías y rutinas en C y en C++. La alimentación de la placa controladora se hizo mediante un regulador de 5V a la salida de la alimentación de la batería. Además es importante que para comunicar esta placa con el bus de comunicación Dynamixel se utilizó un dispositivo USB2Dynamixel (Figura 4) que se conecta al puerto USB y tiene salida al mencionado bus de datos.

Figura 3. Sistema empotrado BeagleBone Black

Figura 4. Interfaz USB2Dynamixel

Para la medición de presión de la planta del pie se utilizaron sensores de presión FRS de 7mm (Figura 5) y para la detección de obstáculos sensores Sharp GP2Y0A21YK0F (Figura 6). La toma de datos, filtrado y conversión de los valores de los sensores se realizó mediante una placa Arduino Nano (Figura 7) en cada planta del pie. Mediante un buffer de tres estados 74LS241 y la implementación de varias rutinas se conectó el Arduino al bus de comunicación y alimentación Dynamixel.

Figura 5. Sensor de presión FRS

Figura 6. Sensor de proximidad SHARP

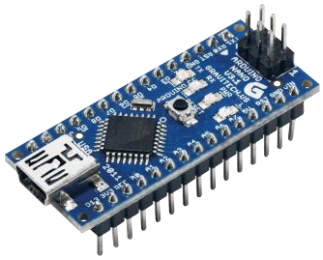


Figura 7. Placa microcontroladora Arduino Nano

1.6. Contribuciones del trabajo

El presente trabajo de fin de Máster genera un aporte a las tecnologías empleadas por el Instituto de Automática e Informática Industrial (AI2) y la Escuela Técnica Superior de Informática de la Universidad Politécnica de Valencia. Hasta este año el control de locomoción robots humanoides Bioloid se realizaba únicamente mediante el controlador CM-510 de la empresa Robotis y el programa Roboplus.

Por tanto, el principal aporte del trabajo radica en poder controlar el robot directamente de un sistema empotrado BeagleBone Black y por tanto ampliar el espectro de tecnologías existentes. Este nuevo desarrollo presenta beneficios como la facilidad de integración con otros dispositivos, entre los que se puede mencionar: IMUs[1] , cámaras e incluso conexiones por red. Además en futuros proyectos se podría hacer uso de la capacidad computacional del sistema empotrado para ejecutar tareas más complejas o que requieran paralelismo entre procesos.

[1] IMU: Unidad de Medición Inercial (por sus siglas en inglés)

Otro aporte importante fue la implementación de una estructura para la planta del pie que proporcionara información sobre valores de presión y detección de obstáculos, los cuales permitieran la toma de decisiones y modificación de comportamientos en el sistema de control.

1.7. Estructura del trabajo

En el trabajo se tendrá la siguiente estructura: el Capítulo 2 será una revisión de los trabajos existentes en el ámbito académico y comercial respecto a robots humanoides.

El Capítulo 3 abordará el diseño, simulación e implementación de la solución de cinemática inversa para los brazos y piernas del robot. El Capítulo 4 comprende los criterios utilizados en la realización de los generadores de trayectorias empleados para los movimientos de los brazos y las piernas.

En el Capítulo 5 se abordará el circuito desarrollado, los elementos utilizados y las rutinas implementadas para la parte de detección y medición en la planta del pie.

La agrupación de un conjunto de trayectorias en un comportamiento, los distintos comportamientos empleados y la máquina de estados para regir el robot se discutirán en el Capítulo 6.

El Capítulo 7 presentará los resultados obtenidos luego de la implementación de los diversos diseños para la medición de variables y locomoción del robot. Además en esta sección se abordará la prueba realizada y el resultado obtenido durante la competición CEABOT.

Para el Capítulo 8 se procederá a la etapa de conclusiones del trabajo, incluyendo además las limitaciones actuales de las tecnologías desarrolladas y las sugerencias para trabajos futuros.

Capítulo 2. Trabajos previos

En esta sección se hará una revisión de algunos trabajos hechos en robots humanoides. Dado que la literatura y fuentes sobre el tema es extensa, se mencionarán los ejemplos más significativos en los ámbitos generales, y luego se discutirán algunos trabajos más relacionados con los objetivos del trabajo.

Los robots humanoides más avanzados en la actualidad corresponden a esfuerzos de empresas u organizaciones gubernamentales que los utilizan para futuros desarrollos comerciales y hasta para exhibiciones de mercadotecnia. Estos robots usualmente son de un tamaño similar al humano y presentan sistemas de alta complejidad para lograr integrar la fuente de alimentación de energía, las unidades de cómputo presentes, los actuadores que posean y los numerosos sensores.

El ejemplo de robot humanoide de tamaño natural más famoso hasta el momento corresponde al robot Atlas de la empresa Boston Dynamics. Este robot cuenta con 28 articulaciones con actuadores hidráulicos, cada uno con control de bucle cerrado de posición y fuerza. Posee un computador de tiempo real y un radar láser. Este es un robot desarrollado por una empresa privada con financiamiento del Departamento de Defensa de Estados Unidos [4].

Un robot que alcanzó bastante popularidad en su momento fue el robot ASIMO de la empresa Honda. Este robot posee 34 grados de libertad, cada uno con un motor eléctrico. Cuenta con la capacidad tanto de caminar, como de correr y manejar el balance de su cuerpo. Este robot se ha utilizado principalmente para investigación y demostraciones comerciales de la empresa [5].

La Agencia Espacial de Estados Unidos NASA ha entrado en los últimos años al mundo del desarrollo de robots humanoides. Su último esfuerzo se llama R5 (también se conoce como Valkyrie) y es un robot con 44 grados de libertad, de los cuales la mayoría están compuestos por actuadores elásticos que proveen seguridad para interactuar con humanos. El objetivo de esta entidad gubernamental es que robots humanoides realicen muchas tareas de exploración espacial, viajes a otros planetas y reparaciones en la Estación Espacial Internacional [6]. Se han formulado una competición global que se llevará a cabo en el 2017, llamada Desafío de Robótica Espacial, la cual engloba una serie de retos que este robot debe cumplir para acercarlo a esa visión [7].

El robot HUBO es un robot humanoide desarrollado por el Instituto Coreano de Ciencia y Tecnología (KAIST). Posee 41 grados de libertad donde cada articulación está compuesta por un motor y un controlador individual. Posee cámaras en la cabeza y sensores de fuerza/par en los tobillos y muñecas. Posee un sistema de control

distribuido con buses CAN[2] entre las partes. El principal objetivo de este robot es la investigación acerca del caminado bípedo con apariencia y movimientos similares a los humanos [8].

El Instituto Nacional de Ciencia y Tecnología Industrial Avanzada de Japón (AIST) ha desarrollado el robot humanoide HRP-4 como plataforma de investigación y desarrollo, este posee 34 grados de libertad y un controlador basado en Linux. El diseño de este robot destaca respecto de otros por su forma delgada y liviana [9].

Existen universidades que han diseñado e implementado robots humanoides, tal es el caso de la Universidad Tecnológica de Georgia en Estados Unidos que creó el robot DURUS, un robot humanoide orientado a la investigación de cadencias y algoritmos para caminar. Este robot posee movimientos en el torso y las piernas, con 17 grados de libertad compuestos por motores eléctricos con cajas reductoras creadas a la medida y la utilización de elementos flexibles en los tobillos [10].

Otro esfuerzo importante de una universidad es el del robot Thor-OP de la Universidad Tecnológica de Virginia en Estados Unidos. Este robot humanoide de tamaño natural tiene la particularidad de que utilizó actuadores de propósito general de la marca Dynamixel (31 actuadores en total), a diferencia de actuadores personalizados como en los casos mostrados anteriormente. Posee una cámara en la cabeza y un radar láser en el pecho. Las comunicaciones con los actuadores y dispositivos se logran mediante un bus RS-485 y un bus USB [11].

Esta misma universidad junto con la universidad de Tokio Japón han diseñado una plataforma robótica humanoide de tamaño reducido (llamada también de tamaño infantil) que se ha nombrado como Darwin-OP. Este robot es de código abierto; con sus rutinas, diseños, diagramas y planos disponibles en su totalidad. Posee 20 grados de libertad y utiliza servomotores Dynamixel para ello. Este robot se ha utilizado ampliamente en investigación y para competencias robóticas como la RoboCup [12].

En el mismo rango de robots humanoides de tamaño infantil, el robot NAO de la empresa Aldebaran ha tenido una notable popularidad. Posee 25 grados de libertad, logrados con motores DC personalizados; y un sistema de control distribuido, con un computador central en la cabeza y varios microcontroladores en el cuerpo conectados por un bus RS-485. El robot NAO se utiliza comúnmente en competiciones, en investigación y hasta en demostraciones en televisión [13].

La empresa Robotis, creadora de los servomotores Dynamixel, fabrica el robot humanoide Bioloid, el cual es un robot de tamaño infantil con 18 grados de libertad que utiliza servomotores, posee un microcontrolador central y conexiones a sensores de proximidad e inercia [14]. A pesar de tener capacidades más limitadas que los robots NAO y Darwin-OP, este robot es muy utilizado en las competencias robóticas debido a que tiene un costo más accesible y a que existe amplia documentación sobre proyectos realizados con él [15] [16] .

Entre la documentación existente para robots humanoides Bioloid se encuentran importantes esfuerzos como el de Núñez et al [17], el de Cerritos-Jasso et al [18] y el de

[2] CAN: Controller Area Network

Chiand y Tsai [19] que proporcionan soluciones explícitas encontradas de manera analítica para el problema del cálculo de la cinemática inversa de un robot humanoide Bioloid. La ventaja de una solución explícita es que aprovecha la morfología humanoide para realizar los cálculos, y evita el tener que dedicar tiempo y recursos computaciones a cálculos matriciales, como si se hiciera un tipo de solución genérica de una cadena cinemática.

Como referencia para este trabajo se tienen también los esfuerzos realizados para encontrar la solución a la cinemática inversa de otros robots, como en los trabajos de O'Flaherty et al [20] en un robot HUBO, Ali et al [21] en un robot KHR-4, y Contreras et al [22] en un robot Darwin-OP. Es de notar las similitudes que se encuentran en el análisis cinemático entre robot de diferentes tamaños, por lo cual el esfuerzo hecho en el presente trabajo se puede considerar extensible en cierta medida a plataformas de mayor tamaño.

Respecto de la generación de secuencias de caminado en robots humanoides Bioloid se encuentran trabajos como el de Akhtaruzzaman y Shafie [23] y el de Arias et al [24], donde muestran el desarrollo realizado para generar un patrón de caminado estable. También resulta interesante el trabajo de Jiwankar y Deshmuk [25] donde además de la generación de un patrón de caminado, se utiliza la información del centro de masa para crear un bucle de control. Como referencia, también resulta de valor el control en bucle cerrado del caminar que realiza Graf et al [26] para un robot NAO.

Por último, se debe notar que no se encuentran esfuerzos documentados por controlar un robot Bioloid únicamente con un sistema empotrado BeagleBone Black, por lo cual resulta de especial valor el aporte realizado mediante este trabajo y las posibilidades que proporciona para futuros desarrollos.

Capítulo 3. Cinemática Inversa del robot Bioloid

En el presente capítulo se explicará cómo se afrontó el problema de la resolución de la cinemática inversa en los brazos, piernas y cadera del robot humanoide Bioloid. Así como las pruebas hechas y las rutinas implementadas en el sistema empotrado.

3.1. Cinemática Inversa de los brazos del robot.

Por Cinemática Inversa de un brazo robótico se entiende el cálculo de los valores de los ángulos de las articulaciones que logran que la punta del brazo alcance un punto en el espacio determinado. Para este cometido, el proyecto se basó en el trabajo de Núñez et al [17].

En esa investigación se propone una solución analítica explícita para el problema de resolver la cinemática inversa, donde dada la facilidad que aporta que el brazo sólo posea 3 grados de libertad, se permite el desarrollo de ecuaciones que describen en cualquier instante las posiciones de los ángulos. En el caso de brazos robóticos más complejos, como es el caso de los brazos con 6 grados de libertad, el cálculo de la cinemática inversa resulta computacionalmente más intenso, teniendo que recurrir a herramientas como aproximaciones sucesivas para resolver algebra matricial.

Se utilizó la misma convención de ejes que en la mencionada investigación, donde se toma el eje x con una orientación hacia afuera del pecho (línea azul en la Figura 10, segunda imagen), el eje y hacia la izquierda del robot humanoide (línea roja) y el eje z hacia la cabeza (línea verde). En el caso del brazo derecho del robot humanoide, como se observa en la Figura 8 primera imagen, el eje x se dirige hacia el frente del robot, el eje z hacia arriba y el eje y negativo ($-y$) hacia afuera del robot.

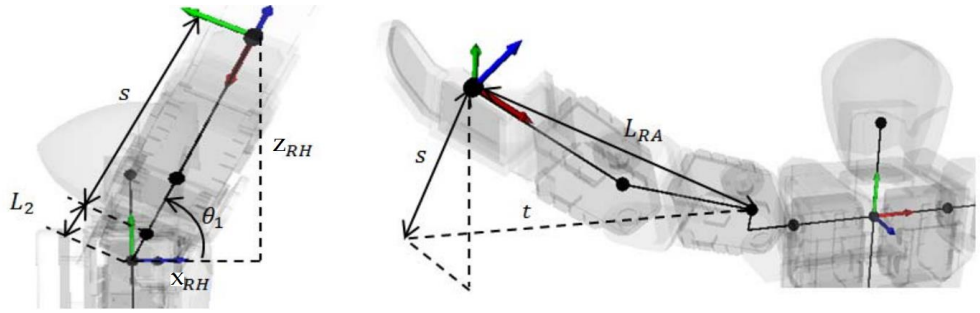


Figura 8. Ejes de referencia y nomenclaturas usadas para la cinemática inversa del brazo del robot [17]

El primer ángulo que se calcula es el de la base del brazo del robot, que en la investigación se refiere como θ_1, y que está formado por la proyección de la punta del brazo sobre el eje x y la proyección sobre el eje z (ver Figura 8, primera imagen). Para los cálculos, la simulación y la implementación se utilizó la función atan2 ya que tiene la ventaja de que calcula el arco tangente y también tiene en consideración los signos de los argumentos para calcular la ubicación del ángulo resultante. La fórmula utilizada para obtener el ángulo de la base del brazo se muestra en la Ecuación 1.

$$\theta_1 = \text{atan2}(z_{RH}, x_{RH}) \qquad \text{Ecuación 1}$$

Para el cálculo del ángulo del codo del brazo del robot, se utilizó la proyección de la punta del brazo sobre el eje y, en la investigación se refiere a este valor como t (Ecuación 2). También se utilizó la hipotenusa entre las proyecciones sobre *x* y sobre *y*, la cual es referida como s (Ecuación 3). Debe notarse que dado que el eje de referencia se toma como en el centro del pecho del robot, en las ecuaciones de *t* y *s* se restan las longitudes para llegar al hombro del robot. Posteriormente con los valores de *s* y *t* se calculó el valor de la extensión del brazo, referida como LRA, y dada por la Ecuación 4.

$$t = -y_{RH} - L_0 - L_1 \qquad \text{Ecuación 2}$$

$$s = \sqrt{x_{RH}^2 + z_{RH}^2} - L_2 \qquad \text{Ecuación 3}$$

$$L_{RA} = \sqrt{s^2 + t^2} \qquad \text{Ecuación 4}$$

Como ya se poseen todas las longitudes del triángulo formado por el antebrazo, brazo y extensión del brazo, por ley de cosenos se puede obtener el valor del coseno del ángulo del codo, el cual es referido como $cos(\gamma_{RA})$ y es referido también como *C5*. Este valor está dado por la Ecuación 5.

$$\cos(\gamma_{RA}) = \frac{L_{RA}^2 - L_3^2 - L_4^2}{2L_3L_4} \qquad \text{Ecuación 5}$$

$$\theta_5 = -\text{atan2}\left(\sqrt{1 - {C_5}^2}, C_5\right) \quad \text{Ecuación 6}$$

Para obtener el ángulo del hombro, se utilizan 2 ángulos auxiliares; el primero es referido como γ1 y está formado por s, t y la extensión del brazo LRA (ver Ecuación 7), el segundo ángulo es referido como γ2 y está formado por LRA y la extensión del brazo L3 (Ecuación 8). En la Tabla 1 se pueden observar los valores de longitud de las partes del brazo que se utilizaron y cuál es su denominación en las ecuaciones. Finalmente el valor del ángulo del hombro, referido como θ_3, está dado por la diferencia de los ángulos anteriormente encontrados (Ecuación 9).

$$\gamma_1 = \text{atan2}(s, t) \quad \text{Ecuación 7}$$

$$\gamma_2 = \text{atan2}(L_4 \sin(\theta_5), L_3 + L_4 \cos(\theta_5)) \quad \text{Ecuación 8}$$

$$\theta_3 = -(\gamma_1 - \gamma_2) \quad \text{Ecuación 9}$$

Tabla 1. Longitudes del brazo utilizadas

Denominación	Medida (mm)	Descripción
L_0	0.047	Distancia del pecho al eje del servo de la base
L_1	0.0145	Distancia del eje del servo de la base al eje del servo del hombro
L_2	0.025	Distancia del servo del hombro al eje y
L_3	0.0675	Longitud del brazo: Distancia del eje del servo del hombro al eje del servo del codo
L_4	0.0745+l_{tool}	Longitud del antebrazo: Distancia del eje del servo del codo al borde del servo del codo. Se le suma la longitud de la mano o herramienta
L_{tool}	0.070	Longitud de la mano o herramienta usada

3.2. Simulación de la Cinemática del brazo del Robot

Para probar las ecuaciones cinemáticas mencionadas en el trabajo se utilizó el programa Matlab con la Robotics ToolBox (desarrollada por Corke, P. y explicada en [27]). Con esto se creó una cadena cinemática de eslabones a los cuales se les pudo aplicar las ecuaciones anteriormente descritas (ver Figura 9), hay que notar que los ejes en Matlab se muestran en un colocación diferente a la usada en el proyecto. Mediante esta simulación se pudo comprobar una discrepancia en los planteamientos de la investigación, ya que los ángulos que aparecían como negativos en las imágenes, se tomaban como positivos en las ecuaciones. Además esta simulación permitió tener certeza de cómo ajustar los ángulos obtenidos a la nomenclatura de ángulos utilizada en el proyecto.

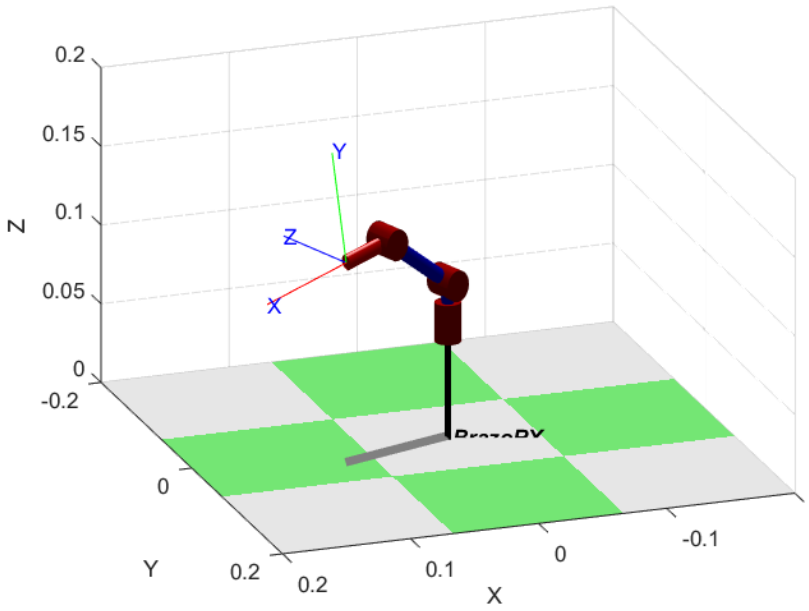


Figura 9. Simulación en Matlab del brazo del robot

3.3. Implementación de las rutinas de Cinemática Inversa del brazo del robot

Las ecuaciones de cinemática inversa se implementaron en la BeagleBone Black mediante una función en C++ llamada IK3ServoArm. Esta rutina recibe las coordenadas de un punto en el espacio y las dimensiones del brazo. Posteriormente utiliza todas las ecuaciones mencionadas y retorna los ángulos de cada articulación que alcanzan el punto en el espacio recibido. Los ángulos se retornan por referencia mediante un parámetro de salida en los argumentos.

Tabla 2. Descripción de la función IK3ServoArm

Función	IK3ServoArm
Descripción	Calcula la cinemática inversa de un brazo de 3 grados de libertad
Argumento de entrada	Real (float) con la coordenada x del punto en el espacio
Argumento de entrada	Real (float) con la coordenada y del punto en el espacio
Argumento de entrada	Real (float) con la coordenada z del punto en el espacio
Argumento de entrada	Vector (float) con las longitudes de las partes del brazo
Argumento de salida	Vector (int) con los ángulos calculados

Debido a que los servomotores Dynamixel tienen una nomenclatura de ángulos diferente a la usada en la investigación, fue necesario crear una función que convirtiera los valores de ángulos obtenidos en la cinemática inversa, los cuales corresponden al número de grados en que se desvía el motor de su eje y de coordenadas. El formato Dynamixel tiene como origen 0 grados, tiene 150 grados alineado con el eje y, y tiene como final 300 grados.

Además en esta función se implementó revisiones de posibles errores, como en el caso de que el codo y/o el hombro se les asignara un ángulo que ocasione un choque con el soporte plástico (ángulo < 60 grados o ángulo > 240 grados), en cuyo caso el ángulo se satura a su valor límite. Otro caso de error corresponde cuando el ángulo tiene un valor menor a cero o mayor a 300 grados, esto significa que el punto al que se desea llegar no es alcanzable y por ende se muestra un mensaje de error. Esta función se llamó IKAdjust3RX y recibe un vector con los 3 ángulos a corregir; retorna por referencia los ángulos corregido en el mismo vector.

Tabla 3. Descripción de la función IKAdjust3RX

Función	IKAdjust3RX
Descripción	Transforma los ángulos obtenidos en la cinemática inversa a la nomenclatura Dynamixel. Corrige posibles valores máximos en los ángulos.
Argumento de entrada/salida	Vector (int) con los ángulos a ser corregidos. Retorna el mismo vector con los ángulos corregidos.

3.4. Cinemática Inversa de piernas y cadera del robot.

La resolución del problema de cinemática inversa de las piernas y la cadera del robot se entiende como el cálculo de los valores de los ángulos de las articulaciones que logran que tomando como eje de referencia el pie, se ubique la cadera en un punto determinado en el espacio tridimensional con una orientación específica.

Además se puede utilizar la forma inversa a esta aproximación en la cual se toma como eje de referencia la cadera y se calcula los ángulos que logran ubicar el pie en determinada posición en el espacio, sin una orientación específica (debido a las limitaciones de grados de libertad en el pie). Ambas maneras de afrontar el problema son necesarias para cuando se resuelve el problema de caminar y que se verá en el siguiente capítulo.

Así como se hizo con los brazos del robot, este trabajo tomó como base la investigación de Núñez et al [17], en la cual se utiliza la facilidad que brinda la morfología humana para resolver mediante entidades geométricas la cinemática inversa, logrando un conjunto de ecuaciones explícito que evite la necesidad de utilizar algebra matricial para encontrar la solución.

Se resolverá la aproximación que toma como referencia el pie y ubica la cadera, ya que es el procedimiento más complejo. La otra situación (tomar como referencia la cadera y ubicar el pie) utiliza las mismas ecuaciones pero con diferente convención de coordenadas. La configuración a resolver se observa en la Figura 10 (primera imagen), donde el eje de coordenadas x (azul) se encuentra hacia el frente del pie, el eje y (rojo) hacia la izquierda del pie y el eje z (verde) hacia arriba del pie. Se tomará como ejemplo la pierna y cadera derecha para la demostración.

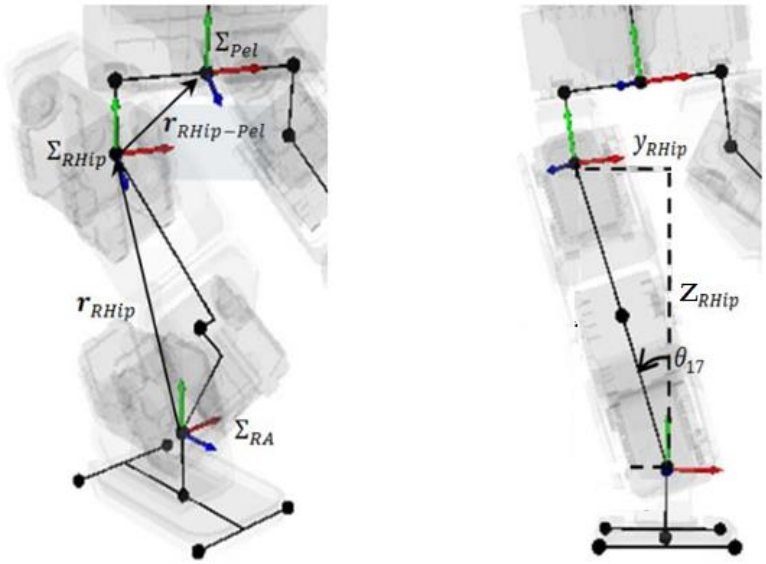


Figura 10. Ejes de referencia y nomenclaturas usadas para la cinemática inversa de la pierna y cadera del robot [17]

Por simplicidad, el primer ángulo que se resuelve es el ángulo de giro lateral del tobillo (o *Roll* del tobillo) y referido como θ_{17}, este ángulo se forma por la distancia en el eje *z* y en el eje *y* donde se desea ubicar la cadera. Para su cálculo se utilizó la función atan2, que como se explicó en el caso del brazo, tiene la ventaja de tomar en cuenta los signos para ubicar correctamente el ángulo. El valor del ángulo Roll del tobillo está dado por la Ecuación 10.

$$\theta_{17} = \text{atan2}(y_{RH}, z_{RH}) \quad \text{Ecuación 10}$$

El cálculo del ángulo de la rodilla presentó cierta complejidad debido a que el rotor del servomotor de la rodilla no está en línea con la cadera y pie, sino se encuentra desplazado cierta distancia. Para solucionar esto se formuló solucionar primero el triángulo formado por la línea recta entre pie y cadera llamada LRLeg (Ecuación 11), la línea entre cadera y centro del rotor de rodilla y la línea entre pie y centro del rotor de rodilla. Estas últimas dos líneas son equidistantes y se nombran como Lf (Ecuación 12).

$$L_{RLeg} = \sqrt{x_{RHip}^2 + y_{RHip}^2 + z_{RHip}^2} \quad \text{Ecuación 11}$$

$$L_f = \sqrt{L_7^2 + L_8^2} \quad \text{Ecuación 12}$$

En el triángulo entre L_{RLeg} y las dos líneas L_f, se puede aplicar la Ley de Cosenos para obtener el valor del coseno del ángulo en el rotor de la rodilla γ, nombrado siguientemente como Cγ (Ecuación 13). Usando nuevamente la función atan2 e identidades trigonométricas, se puede encontrar la solución al ángulo γ (Ecuación 14).

$$\cos(\gamma) = \frac{L_{RLeg}^2 - 2L_f^2}{-2L_f^2} \quad \text{Ecuación 13}$$

$$\gamma = \text{atan2}\left(\sqrt{1 - {C_\gamma}^2}, C_\gamma\right) \quad \text{Ecuación 14}$$

Sin embargo como se mencionó anteriormente, el rotor del servomotor de la rodilla se encuentra desplazado cierta distancia, por lo cual el ángulo que se aplica al servomotor es diferente del ángulo de la rodilla. Para hallar este ángulo es necesario encontrar primero el ángulo de diferencia entre la rodilla y el rotor, y mencionado como α (Ecuación 15). Luego el ángulo de la rodilla θ13 está dado por la Ecuación 16.

$$\alpha = \text{atan2}(L_7, L_8) \quad \text{Ecuación 15}$$

$$\theta_{13} = (\gamma - 2\alpha)$$ Ecuación 16

Para resolver el ángulo de giro frontal del tobillo (*Pitch* del tobillo) es necesario varios cálculos previos. Primero se encuentra la extensión de la cadera en el plano y-z y nombrada como L1RLeg (Ecuación 17); con este valor se encuentra el ángulo entre L1Rleg y la posición en z de la cadera, llamado β (Ecuación 18). Finalmente el ángulo Pitch del tobillo θ15 está dado por la Ecuación 19.

$$L^1_{RLeg} = \sqrt{z^2_{RHip} + y^2_{RHip}}$$ Ecuación 17

$$\beta = \text{atan2}(x_{RHip}, L^1_{RLeg})$$ Ecuación 18

$$\theta_{15} = -\left(\beta - \frac{\gamma}{2} + \alpha\right)$$ Ecuación 19

Para determinar el valor de los ángulos de los servomotores de la cadera se utilizan los valores de ángulos *Yaw*, *Pitch* y *Roll* deseados para la orientación. El ángulo del servomotor 7 es transparente con el valor del ángulo Yaw (Ecuación 20). El valor del servomotor 9 corresponde al ángulo Roll deseado más el ángulo Roll del tobillo (Ecuación 21). Por último ángulo del servomotor 11 está dado por la Ecuación 22.

$$\theta_7 = Yaw_{RHip}$$ Ecuación 20

$$\theta_9 = Roll_{RHip} + \theta_{17}$$ Ecuación 21

$$\theta_{11} = -Pitch_{RHip} + \beta + \frac{\gamma}{2}\,\alpha$$ Ecuación 22

Tabla 4. Longitudes de la pierna del robot utilizadas

Denominación	Medida (mm)	Descripción
L_7	0.0145	Distancia del ángulo de la rodilla al centro del rotor del servomotor
L_8	0.075	Longitud del muslo y la pantorrilla

Para el caso en que se toma el eje de referencia en la cadera y se desea ubicar el pie, las ecuaciones anteriores mostradas anteriormente son válidas. Sólo se debe hacer la salvedad que en esta situación se resuelve únicamente la ubicación del pie, no su orientación; debido a las limitaciones de la configuración de pie robótico utilizado (no hay suficientes grados de libertad para resolver la orientación). Además se debe tener en cuenta que la cadera tiene la misma convención de ejes que cuando la referencia estaba

en el pie, por lo que las posiciones en las que se ubicará el pie caerán dentro del plano *z* negativo.

3.5. Simulación de la Cinemática de las piernas del Robot

Así como se hizo para simular el brazo del robot Bioloid, para el caso de simular las piernas del robot también se utilizó el programa Matlab con la librería Robotics Toolbox [27]. Para esto se creó una cadena cinemática de 3 eslabones (uno fue sólo para simular la forma del pie) y 6 grados de libertad. Se invirtió el sentido del eje Z para que la cadena cinemática fuera de abajo hacia arriba y por ende fuera similar a una pierna. Debido a que en esta simulación la cadera es inmóvil, se comprobó el caso de ubicar el pie tomando como referencia la cadera.

En código Matlab se implementaron todas las ecuaciones de la sección anterior y se comprobó que lograran ubicar el pie correctamente en el punto en el espacio deseado. Es de notar la importancia de esta simulación, ya que permitió encontrar discrepancias en las ecuaciones de [17], y por tanto fue necesario volver a calcular las fórmulas hasta llegar a las mostradas anteriormente.

Además esta simulación permitió conocer el rango de alcance de las piernas en el espacio y por ende definir un área de trabajo permisible. Además en estas simulaciones fue posible crear algunas trayectorias entre puntos y ver algunos tipos de marcha posibles.

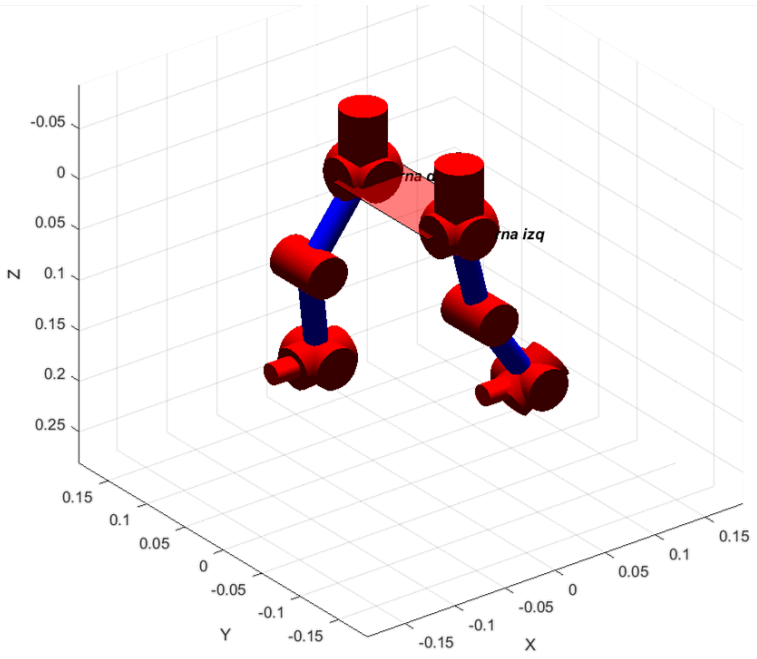


Figura 11. Simulación en Matlab de las piernas y las ecuaciones de cinemática inversa

3.6. Implementación de las rutinas de Cinemática Inversa para las piernas y cadera del robot

Para poder aplicar las ecuaciones de cinemática inversa en el sistema empotrado BeagleBone Black, se implementaron dos rutinas en C++. La primera se llama IK6ServoLegtoHip e implementa la ubicación de la cadera tomando como referencia el pie. Esta rutina recibe las coordenadas de un punto en el espacio donde se desea posicionar la cadera, la orientación deseada de la cadera y las dimensiones de la pierna. Se utiliza las ecuaciones de cinemática inversa para retornar los ángulos de cada articulación de la pierna y cadera que alcanzan el punto en el espacio y la orientación deseada. El retorno de los ángulos se hace por medio de un parámetro de salida en los argumentos donde se pasa por referencia un vector.

Tabla 5. Descripción de la función IK6ServoLegtoHip

Función	IK6ServoLegtoHip
Descripción	Calcula la cinemática inversa de la cadera teniendo como referencia el pie
Argumento de entrada	Real (float) con la coordenada x del punto en el espacio
Argumento de entrada	Real (float) con la coordenada y del punto en el espacio
Argumento de entrada	Real (float) con la coordenada z del punto en el espacio
Argumento de entrada	Real (float) con el ángulo Pitch deseado en la cadera
Argumento de entrada	Real (float) con el ángulo Roll deseado en la cadera
Argumento de entrada	Real (float) con el ángulo Yaw deseado en la cadera
Argumento de entrada	Vector (float) con las longitudes de las partes de la pierna
Argumento de salida	Vector (int) con los ángulos calculados

La segunda rutina de cinemática inversa para la pierna es la función IK6ServoLegtoFoot, la cual toma como referencia la cadera y calcula los ángulos que posicionan el pie en un punto en el espacio. Esta función recibe un grupo de coordenadas que representa la posición deseada del pie y recibe también las dimensiones de la pierna. Luego mediante las ecuaciones de cinemática inversa se calculan los ángulos de cada articulación de la pierna que satisfacen los requerimientos de entrada. En los argumentos se tiene un parámetro de salida en el cual se pasa por referencia un vector, este es modificado con los valores de salida de los ángulos.

Tabla 6. Descripción de la función IK6ServoLegtoFoot

Función	IK6ServoLegtoFoot
Descripción	Calcula la cinemática inversa de la cadera teniendo como referencia el pie
Argumento de entrada	Real (float) con la coordenada x del punto en el espacio
Argumento de entrada	Real (float) con la coordenada y del punto en el espacio
Argumento de entrada	Real (float) con la coordenada z del punto en el espacio
Argumento de entrada	Vector (float) con las longitudes de las partes de la pierna
Argumento de salida	Vector (int) con los ángulos calculados

Así como se hizo con el brazo, en este caso también fue necesario implementar una función que transformara los ángulos encontrados en la cinemática inversa a la convención utilizada por los servomotores Dynamixel. Esta función se llamó IKAdjust6ServoLeg y recibe por referencia un vector con los ángulos de la pierna creados en la función de cinemática, y en este mismo vector almacena los valores corregidos. Recibe además una variable tipo lógica (bool) indicando si se trata de la pierna derecha o no.

En la convención de ángulos empleada en los cálculos, el centro representa 0 grados, giros a la izquierda son negativos y a la derecha positivos. En la convención Dynamixel, el centro está en 150 grados y se puede girar hasta 0 grados a la izquierda y hasta 300 grados a la derecha. Debido a que la pierna derecha tiene varios servomotores en posición contraria a la pierna izquierda, fue necesario hacer 2 tipos de correcciones, según cual pierna era.

Además en esta función se implementó la funcionalidad de revisión de errores en los ángulos, es decir ángulos que pudieran ocasionar un choque en la estructura (lo que podría dañar el servomotor). Para esto se encontró experimentalmente los valores límites de cada articulación de la pierna y cadera. Se revisó además un caso de error posible, que es cuando los ángulos Pitch de la cadera y tobillo son cero, y el ángulo de la rodilla es 300, lo cual representa el resultado de buscar un punto no alcanzable en el espacio.

Tabla 7. Descripción de la función IKAdjust6ServoLeg

Función	IKAdjust6ServoLeg
Descripción	Transforma los ángulos obtenidos en la cinemática inversa a la nomenclatura Dynamixel. Corrige posibles valores máximos y mínimos en los ángulos.
Argumento de entrada/salida	Vector (int) con los ángulos a ser corregidos. Retorna el mismo vector con los ángulos corregidos.
Argumento de entrada	Variable lógica (bool) indicando si se trata de la pierna derecha o no

Capítulo 4. Generación de trayectorias

En el siguiente capítulo se explicará las técnicas abordadas para generar trayectorias entre 2 puntos en el espacio para las diferentes extremidades del robot. El enfoque general utilizado consistió en mantener una velocidad de movimiento constante en los servomotores y controlar la posición en diferentes intervalos de la trayectoria. A continuación se verá de manera detallada como se logró esto.

4.1. Trayectorias para el brazo del robot

Para lograr que el brazo del robot se desplazara entre dos puntos en el espacio describiendo un movimiento predeterminado, se implementaron 2 tipos de trayectorias en el trabajo: trayectoria libre y trayectoria lineal (o cartesiana).

En la trayectoria libre, la punta del brazo del robot se desplaza entre 2 puntos de la manera más rápida y fácil posible para sus articulaciones. Generalmente este tipo de trayectorias describe un movimiento circular o de arco (ver Figura 12), ya que al ser las articulaciones de tipo revolución, este es el movimiento más simple posible.

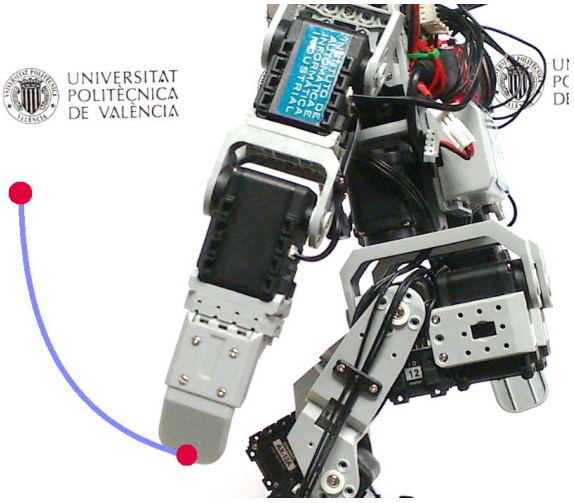


Figura 12. Ilustración de una trayectoria libre para el brazo del robot.

Para lograr describir este tipo de movimiento se implementó la función freeTraj, la cual recibe el punto en el espacio donde se va a iniciar el movimiento y el punto donde va a terminar; y también recibe las longitudes de las partes del brazo (necesarias para la cinemática inversa). Con estos datos, la función calcula la cinemática inversa para el punto de inicio y también para el punto final. Luego para cada articulación se toman los

ángulos de inicio y de fin y se calculan una serie de ángulos intermedios entre estos 2 ángulos. La cantidad de ángulos intermedios está determinado por la variable global llamada "sample". Con esos ángulos obtenidos, la función crea una matriz con los valores para cada articulación en cada instante de tiempo, la cual es retornada por referencia.

Tabla 8. Descripción de la función freeTraj

Función	freeTraj
Descripción	Calcula una trayectoria libre entre 2 puntos para el brazo del robot
Argumento de entrada	Vector (float) con las coordenadas del punto de inicio
Argumento de entrada	Vector (float) con las coordenadas del punto de fin
Argumento de entrada	Vector (float) con las longitudes de las partes del brazo
Argumento de salida	Matriz (int) con los ángulos para cada instante de tiempo de la trayectoria

En una trayectoria lineal o llamada también trayectoria cartesiana, la punta del brazo robótico describe una trayectoria que corresponde a una línea recta en el espacio cartesiano (ver Figura 13). Este tipo de movimiento es el que puede resultar más intuitivo al pensar en trayectorias, pero a la vez es mucho más complicado para el robot ya que debe variar las posiciones de sus articulaciones durante el movimiento para lograr que la punta del brazo permanezca en la trayectoria de la línea recta.

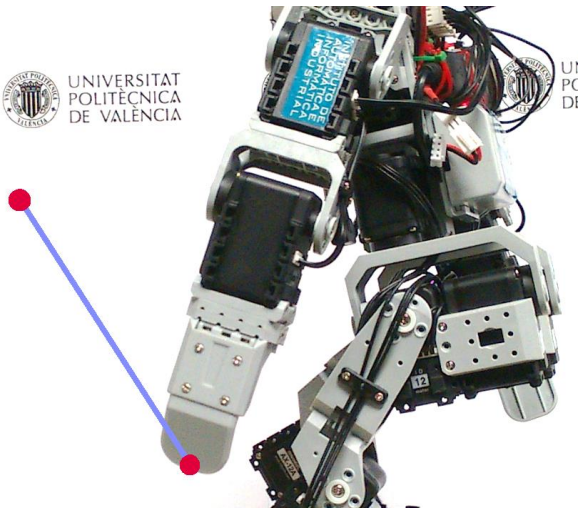


Figura 13. Ilustración de una trayectoria lineal o cartesiana para el brazo del robot

Para crear una trayectoria lineal o cartesiana se implementó la función linearTraj, la cual recibe el punto de inicio de la trayectoria lineal en el espacio y el punto de fin de la trayectoria; y también recibe las longitudes de las partes del brazo. Con estos valores de entrada, la función crea una línea entre el punto de inicio y el punto final: luego divide esta línea en una serie de puntos intermedios. Así como en el caso de la función freeTraj, en la función linearTraj la cantidad de puntos intermedios o divisiones de la línea está establecido por la variable global "sample". Para cada uno de los puntos intermedios obtenidos la función calcula la cinemática inversa del brazo robótico, y almacena dichos

valores en una matriz. Esta matriz se retorna por referencia y contiene los valores de ángulos para cada articulación en cada instante de tiempo.

Tabla 9. Descripción de la función linearTraj

Función	linearTraj
Descripción	Calcula una trayectoria linear o cartesiana entre 2 puntos para el brazo del robot
Argumento de entrada	Vector (float) con las coordenadas del punto de inicio
Argumento de entrada	Vector (float) con las coordenadas del punto de fin
Argumento de entrada	Vector (float) con las longitudes de las partes del brazo
Argumento de salida	Matriz (int) con los ángulos para cada instante de tiempo de la trayectoria

4.2. Trayectorias para la pierna y cadera del robot

Para lograr movimientos controlados entre 2 puntos en el espacio por parte de la cadera o del pie del robot, se implementaron 2 funciones de generación de trayectorias.

En el caso de los desplazamientos de la cadera en el espacio se implementó un generador de trayectoria lineal (o cartesiana), ya que es importante mantener la orientación y posición controlada durante los puntos intermedios del recorrido (cosa que no pasa con una trayectoria libre) y así evitar posiciones del robot que puedan comprometer el equilibrio. Este generador calcula todos los ángulos intermedios que debe tener cada articulación de manera que la cadera describe un movimiento lineal (Figura 14).

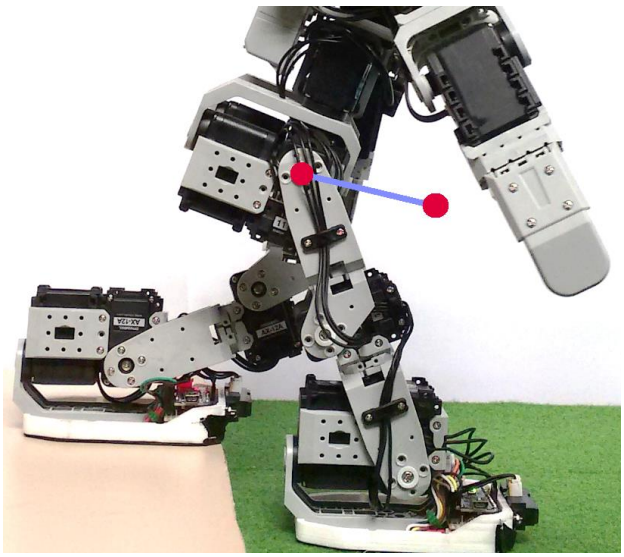


Figura 14. Ilustración de una trayectoria lineal para la cadera del robot

Para este tipo de movimiento se implementó la función linearTrajHip, la cual toma el punto final deseado y el punto de inicio y calcula una línea recta; luego esta línea se divide en segmentos (definidos por un argumento de entrada llamado "divisionQty") y calcula la cinemática inversa para cada segmento. Los ángulos encontrados para las articulaciones de la pierna y cadera son almacenados en una matriz, la cual es el argumento de salida de la función (por referencia).

La función linearTrajHip recibe el punto inicial y el punto final de la trayectoria, la orientación deseada, la matriz donde se retornará por referencia los ángulos de las articulaciones para cada segmento, una variable lógica que indica si es la pierna derecha o izquierda, y la cantidad de divisiones que tendrá el movimiento.

Tabla 10. Descripción de la función linearTrajHip

Función	linearTrajHip
Descripción	Calcula una trayectoria linear o cartesiana entre 2 puntos para la cadera del robot
Argumento de entrada	Vector (float) con las coordenadas del punto de inicio
Argumento de entrada	Vector (float) con las coordenadas del punto de fin
Argumento de entrada	Vector (float) con la orientación deseada durante el movimiento
Argumento de salida	Matriz (int) con los ángulos para cada instante de tiempo de la trayectoria
Argumento de entrada	Variable lógica (bool) indicando si se trata de la pierna derecha o no
Argumento de entrada	Cantidad de divisiones intermedias para la trayectoria

Para los desplazamientos del pie en el espacio también se eligió implementar un generador de trayectoria lineal (o cartesiana) debido a que se desea evitar posiciones que puedan ocasionar choques de partes del pie con el suelo, es decir se desea que la planta del pie se mantenga en su posición horizontal siempre. Por lo tanto, un generador de trayectorias lineal permite mantener control de la posición de las partes de la pierna durante los puntos intermedios de una trayectoria cartesiana del pie entre 2 puntos en el espacio (Figura 15).

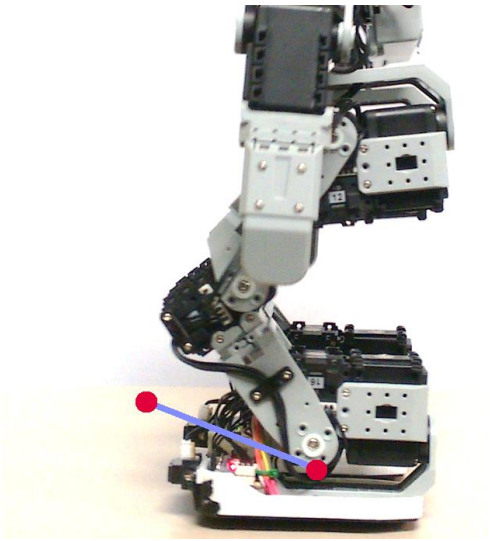


Figura 15. Ilustración de una trayectoria lineal para el pie del robot

La generación de trayectorias lineales para el pie se implementó mediante la función linearTrajFoot, que internamente realiza un cálculo similar a la función para la cadera, con puntos intermedios entre el inicio y fin, pero con la diferencia de que no toma en cuenta la orientación.

Esta función recibe la coordenada del punto inicial y la coordenada del punto final de la trayectoria, una matriz donde por referencia se retornarán los ángulos calculados para las articulaciones en cada segmento del movimiento, la variable lógica que indica si es la pierna derecha o no, y la cantidad de divisiones que tendrá la trayectoria Tabla 11.

Tabla 11. Descripción de la función linearTrajFoot

Función	linearTrajFoot
Descripción	Calcula una trayectoria linear o cartesiana entre 2 puntos para el pie del robot
Argumento de entrada	Vector (float) con las coordenadas del punto de inicio
Argumento de entrada	Vector (float) con las coordenadas del punto de fin
Argumento de salida	Matriz (int) con los ángulos para cada instante de tiempo de la trayectoria
Argumento de entrada	Variable lógica (bool) indicando si se trata de la pierna derecha o no
Argumento de entrada	Cantidad de divisiones intermedias para la trayectoria

Capítulo 5. Detección y medición en la planta del pie

En este capítulo se abordarán los criterios de diseño utilizados para la medición de datos de presión y detección de proximidad en la planta del pie del robot. También se mostrarán cómo fue implementada la comunicación del microcontrolador con el bus de comunicación Dynamixel. El objetivo de esta implementación fue utilizar los datos de los sensores para la creación posterior de lógicas de control en bucle cerrado.

5.1. Esfuerzos anteriores

La integración de la toma de datos y detección de proximidad en la planta del pie ya había sido abordada por Vañó [28], en donde se desarrolló un sistema empotrado que mide 4 sensores de presión FSR ubicados en las esquinas de la parte inferior de la planta del pie del robot Bioloid y un sensor de proximidad Sharp ubicado en la parte delantera. El sistema empotrado se programó para ser conectado a un bus de comunicaciones TTL. También se rediseñó la parte mecánica de la planta del pie para albergar los sensores y la placa del sistema empotrado.

Sin embargo al realizar pruebas a este sistema no se logró entablar comunicación con él, sospechando principalmente que alguno de los componentes se encontrara dañado. Esto representaba un problema importante para el desarrollo del proyecto con miras a la competición CEABOT, debido a que la placa diseñada y el sistema empotrado utilizado eran muy específicos y la fabricación de un sistema nuevo podría demorar muchas semanas.

Por tanto se eligió el utilizar un sistema microcontrolador genérico y de tamaño reducido (en este caso un Arduino Nano) y tomar el trabajo realizado anteriormente como base para una nueva implementación. Las partes mecánicas creadas anteriormente fueron reutilizadas sin necesidad de mayores cambios a sus especificaciones.

5.2. Medición de valores de presión y detección de proximidad

Para la medición de valores de presión se utilizaron sensores FSR en una configuración de divisor de voltaje, es decir una patilla del sensor conectada a la alimentación de 5V y la otra a una resistencia de “Pull-down” de 10 KΩ conectada a

tierra. La medición de voltaje se realizó en la conexión entre sensor y resistencia externa, y se llevó a la entrada de conversión Analógica-Digital del microcontrolador (Figura 16).

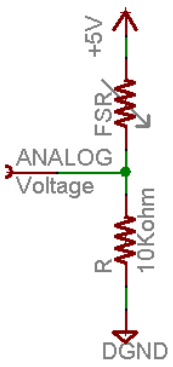


Figura 16. Circuito de medición implementado para cada sensor FSR [29]

Mediante esta configuración, conforme disminuye la resistencia del FSR (aumento de presión) la corriente total aumenta y por ende aumenta el voltaje medido entre la resistencia de 10 KΩ y tierra. En resumen, se logra que un efecto proporcional de que a mayor presión, mayor voltaje en la entrada.

Para atenuar el efecto del ruido en la medición de valores de presión, se implementó un filtro de promedio en el microcontrolador, el cual utiliza los últimos 3 valores medidos en el sensor y calcula el promedio.

Para la detección de obstáculos en la proximidad del pie se utilizó un sensor Sharp GP2Y0A21YK0F, el cual se indica en las especificaciones posee un rango efectivo de 10cm a 80cm, pero en distancias menores a 10cm también se puede utilizar, pero con menos precisión. La conexión con este dispositivo se hizo de manera directa entre el pin de datos y la entrada de conversión analógica a digital del microcontrolador.

Los valores de voltaje medidos a la salida del sensor Sharp no varían de forma lineal conforme aumenta la distancia de un objeto, sino que poseen una curva que se ajusta a una forma exponencial decreciente. Para poder facilitar la lectura de los valores, se utilizó una ecuación que se ajusta en gran medida a la curva proporcionada por el fabricante (Figura 17). Esta ecuación convierte los valores a la salida del convertidor analógico digital en valores de milímetros (Ecuación 23).

$$Distancia\ (mm) = \left(\frac{3027{,}4}{valor\ ADC}\right)^{1{,}2134} \cdot 10 \qquad \text{Ecuación 23}$$

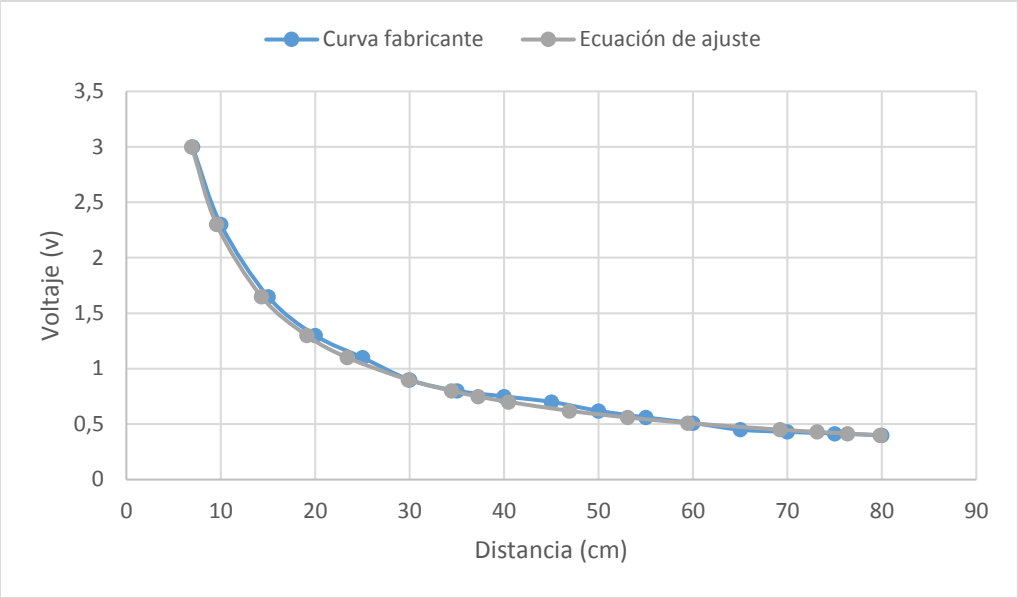


Figura 17. Voltaje de salida del Sensor Sharp según la distancia

Como se mencionó anteriormente el sensor Sharp tiene una especificación para distancias mayores de 10cm, sin embargo si se pueden realizar mediciones en distancias menores a esta. Como se observa en la Figura 18 el voltaje de salida disminuye abruptamente conforme la distancia disminuye de 10cm a 0, por lo que en estas distancias pueden haber cambios muy grandes en los voltajes medidos. Además al utilizar todo el rango de 0 a 80cm, se tendrán 2 posibles distancias para un mismo voltaje, por lo que se debe tener en cuenta este aspecto al analizar los datos obtenidos.

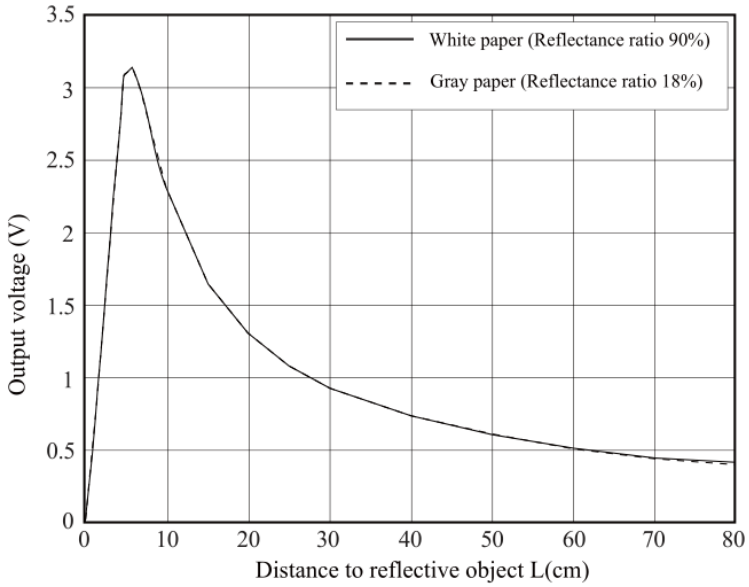


Figura 18. Medición de distancia mediante voltaje en Sharp GP2Y0A21YK0F [30]

5.3. Conexión al bus de datos Dynamixel

Para poder comunicar el microcontrolador de la planta del pie mediante el bus de datos Dynamixel se debió solventar primero las diferencias entre la comunicación. El microcontrolador de la placa Arduino Nano posee una UART[3] TTL. La comunicación por UART se caracteriza porque tiene 3 pines básicos: Tx (transmisión), Rx (recepción) y GND (tierra), no posee señal de reloj (de ahí el término asincrónico).

El bus de datos Dynamixel posee 3 líneas: Vcc (voltaje de alimentación), Data (datos) y GND (tierra). Por lo que la diferencia clara que se nota es que la UART utiliza dos líneas para enviar datos, mientras que el bus sólo usa una.

Para resolver este problema se utilizaron buffers de 3 estados, específicamente el circuito integrado 74LS241. Se implementó un circuito que conecte al pin Data del bus Dynamixel sólo uno de los dos pines de Datos de la UART (Tx o Rx) a la vez (Figura 19). Para esto es necesario además la utilización de una señal de Control que cuando está en alto habilita la transmisión de la UART y cuando está en bajo habilita la recepción.

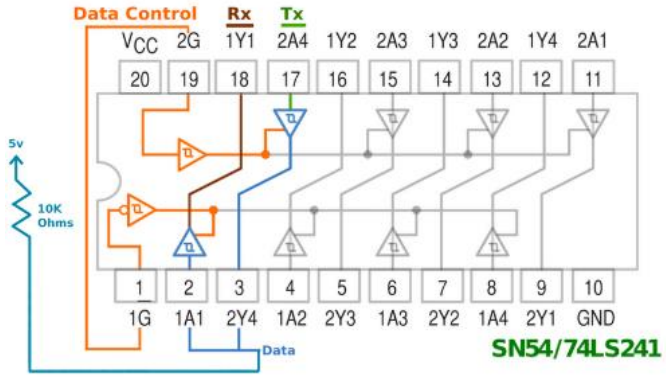


Figura 19. Circuito implementado para la conexión de la UART al bus Dynamixel [31]

La correcta habilitación de la línea de Control en el código del microcontrolador y el manejo de su temporización fue clave para asegurar una adecuada recepción de un mensaje y el envío de su respuesta. Para esto se utilizó una ecuación que calcula el tiempo que toma enviar cierta cantidad de datos en un tiempo específico (Ecuación 24). Esta ecuación utiliza la cantidad de bytes a enviar, el tamaño de la palabra (cantidad de bits por trama) y la velocidad de transmisión. Con el valor del tiempo obtenido, se pueden aplicar tiempos de espera adecuados en el microcontrolador durante la transmisión, antes de regresar al estado de recepción.

$$Tiempo\ envío = \frac{Cant.Bytes\ \cdot Tamaño\ palabra}{Velocidad\ (baudios)} \qquad \text{Ecuación 24}$$

[3] UART: Universal Asynchronous Receiver/Transmitter

5.4. Comunicación en el bus de datos Dynamixel

En el bus de datos Dynamixel existe un único Maestro y los demás dispositivos funcionan como esclavos. El maestro es el dispositivo de control del robot (la placa BeagleBone Black en nuestro caso) y por tanto el microcontrolador se debe conectar como un esclavo al bus. Esto significa que el microcontrolador deberá estar recibiendo hasta que lea un mensaje con su identificador (ID) y deberá responder según lo solicitado.

Debido a que los sensores en la planta del pie tienen las funciones de medición y detección, y ningún tipo de actuación, se diseñaron las rutinas del microcontrolador para responder únicamente a operaciones de lectura de datos. Por lo tanto el microcontrolador estará en estado de recepción hasta que reciba un mensaje según el formato de la Figura 20.

0xFF	0xFF	ID ej: 0x3A	Longitud 0x04	Instrucción 0x02	Dirección ej: 0x04	Tamaño ej: 0x02	Checksum

Figura 20. Formato del paquete de lectura con protocolo Dynamixel

En esta figura cada recuadro corresponde a un dato de tamaño 1 byte. Los 2 primeros bytes corresponden al valor 0xFF o encabezado, el tercer byte es el número que identifica a cada dispositivo. Seguidamente se envía la longitud del paquete, la cual en este caso siempre será de 0x04. El siguiente byte es la instrucción recibida, que en el caso de la planta del pie será de lectura (0x02). Luego se envía la dirección del registro donde se desea leer y el tamaño de los datos a leer (cantidad de bytes). El último dato a enviar corresponde a un valor para revisar si la transmisión sucedió sin errores; este valor se calcula como el inverso del resultado de la suma de todos los bytes del mensaje (excluyendo los 0xFF) y se conoce como Checksum.

Si el microcontrolador recibe un mensaje de este estilo de forma correcta, procederá a transmitir un mensaje de respuesta con el valor solicitado. Este mensaje tendrá un formato como el de la Figura 21, en el cual los 2 primeros bytes son de encabezado, el tercer byte es el identificador del dispositivo y el cuarto la longitud del mensaje. El siguiente byte será la indicación de si hubo algún error, en caso de no haberlos tiene un valor de cero. Los siguientes 2 bytes corresponden a la medición solicitada, para esto se debe dividir el valor en 2 bytes, enviar primero el byte bajo y luego el byte alto.

0xFF	0xFF	ID ej: 0x3A	Longitud ej: 0x04	Error 0x00	Valor (byte bajo)	Valor (byte alto)	Checksum

Figura 21. Formato del paquete de respuesta de lectura con protocolo Dynamixel

Los valores medidos en los sensores FSR y Sharp se ordenaron en una tabla de registros, donde se especifica la dirección de cada valor que puede ser consultado (Tabla

12). Al final de la tabla aparecen los valores de los centros de masa de la planta del pie, estos valores fueron calculados con la información de los cuatro sensores de presión y su ubicación en la estructura mecánica.

Tabla 12. Tabla de control de los registros del sistema de la planta del pie

Dirección	Dirección (hex)	Item	Byte
0	0x00	Sensor Sharp	Bajo
1	0x01	Sensor Sharp	Alto
2	0x02	Sensor FSR 1	Bajo
3	0x03	Sensor FSR 1	Alto
4	0x04	Sensor FSR 2	Bajo
5	0x05	Sensor FSR 2	Alto
6	0x06	Sensor FSR 3	Bajo
7	0x07	Sensor FSR 3	Alto
8	0x08	Sensor FSR 4	Bajo
9	0x09	Sensor FSR 4	Alto
10	0x0A	Centro de Masa en X	Bajo
11	0x0B	Centro de Masa en X	Alto
12	0x0C	Centro de Masa en Y	Bajo
13	0x0D	Centro de Masa en Y	Alto

En caso de que se solicite leer una dirección de un registro inexistente, el microcontrolador transmitirá un mensaje de error. Este mensaje es similar al de la Figura 21, donde el byte de error será 0x04, lo cual significa un Error de Instrucción, es decir que la instrucción recibida no se conoce o corresponde a una dirección no permitida. Además de esto en el mensaje de error no se transmitirán los bytes de valor.

5.5. Implementación de los circuitos en la estructura mecánica

El microcontrolador Arduino Nano, el circuito necesario para la comunicación con el bus Dynamixel y los elementos pasivos para la conexión con los sensores FSR, fueron montados en una placa de prototipos de manera que abarcaran un área compacta. Además en esta placa se montaron pines de conexión para el fácil enlace con el sensor Sharp, los sensores FSR y el bus Dynamixel (Figura 22).

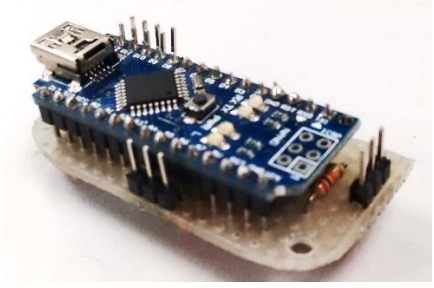


Figura 22. Circuito de medición, detección y comunicación montado en placa de prototipos

La placa de prototipos con los circuitos fue montada en la estructura mecánica diseñada en trabajos anteriores, así mismo se montaron los sensores FSR en la parte inferior de la planta del pie se sujetador con un adhesivo de contacto. En la parte frontal de la estructura se montó el sensor de proximidad Sharp mediante tornillos. Por último se conectaron los cables de comunicación de los sensores a los pines instalados en la placa de prototipos (Figura 23).

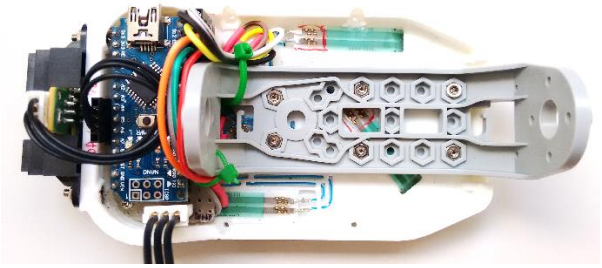


Figura 23. Sensores, circuito y cables de comunicación montados en la estructura mecánica de la planta del pie

Como paso final se desmontó la planta del pie original en el robot Bioloid y se montó la nueva planta del pie con los sensores y circuitos desarrollados. La estructura mecánica abarca un área horizontal casi igual a la de la planta original.

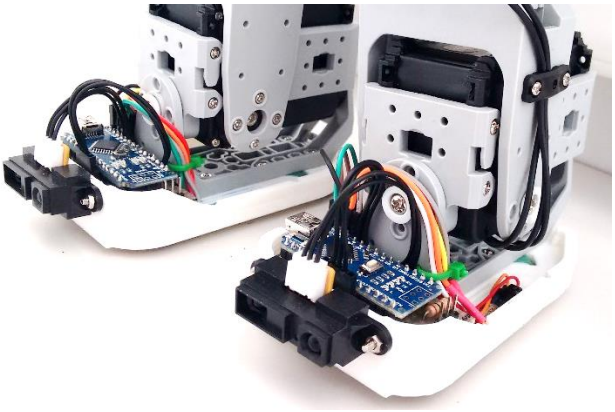


Figura 24. Nueva planta del pie montada en el robot Bioloid

Capítulo 6. Comportamientos del robot

Se crearon varias trayectorias para las extremidades del robot que juntas describen un tipo de movimiento, estas fueron agrupadas en lo que se denomina comportamiento. Por ejemplo, el conjunto de trayectorias que describe el levantar un pie, moverlo hacia adelante y bajarlo, y luego repetir el proceso con el otro pie; se agruparían dentro del comportamiento "dar un paso adelante".

A continuación se describirán los comportamientos que fueron diseñados e implementados en el robot, así como el sistema implementado para controlar los cambios entre comportamientos.

Debe resaltarse que para la escritura de valores de posición y velocidad en los servos se utilizó la librería desarrolladas por Cisternas, D. en [32]. Esta librería se encarga de las comunicaciones entre un sistema empotrado BeagleBone Black y un bus de comunicación Dynamixel. Para la conexión física al bus se utilizó el dispositivo Usb2dynamixel conectado al puerto USB del sistema empotrado.

6.1. Dar un paso adelante

Mediante las funciones que se mencionaron en los capítulos anteriores, se crearon un conjunto de trayectorias para los dos pies del robot, que juntas describen el movimiento de que el robot da un paso hacia adelante.

Estas trayectorias son una combinación de desplazamientos para la cadera, que se usan cuando el pie está en contacto con el suelo, y desplazamientos del pie, cuando este está en el aire. Así el movimiento de dar un paso adelante se compuso de las siguientes trayectorias descritas en la Tabla 13.

Tabla 13. Trayectorias para el comportamiento dar un paso adelante

Trayectoria #	Descripción
1	Se desplazan ambas caderas a un lado. De manera que el centro de gravedad quede sobre un pie.
2	Se levanta el pie que está lejos del centro de gravedad y se mueve hacia adelante.
3	Se baja el pie que está en el aire a su posición final en el suelo (adelante).
4	Con ambos pies en el suelo, se desplaza la cadera izquierda y derecha, pasando de estar sobre el pie de atrás, a estar sobre el pie de adelante. Al mismo tiempo se desplazan ambas caderas lateralmente para ubicar el centro de gravedad sobre el pie de adelante.
5	Se levanta el pie que esta atrás y se mueve adelante.
6	Se baja el pie levantado hasta que toque el suelo y quede ubicado junto al otro pie.
7	Se mueven ambas caderas de manera que el centro de gravedad quede centrado en ambos pies (posición normal).

El rango de movimiento de los pies y el diseño de las secuencias de movimientos a describir se comprobaron mediante simulaciones en Matlab (ver Figura 25). Se verificaron la correcta ubicación de los pies según lo indicado y la adecuada descripción de las trayectorias creadas.

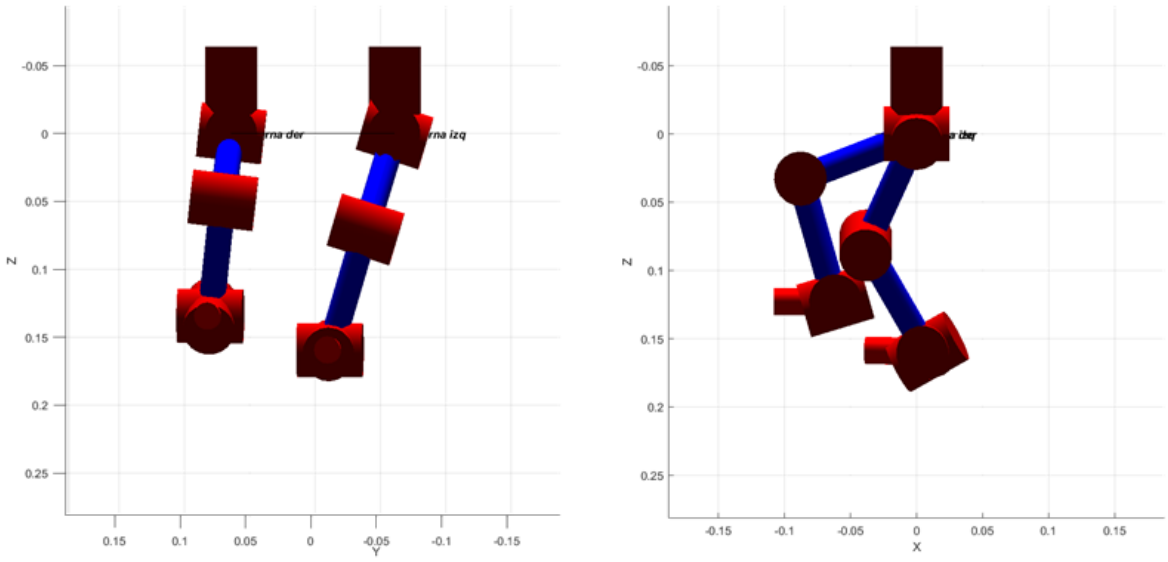


Figura 25. Vista frontal de la simulación el movimiento de dar un paso adelante

Este comportamiento se implementó en el sistema empotrado mediante 2 funciones: StepForwardR y StepForwardL. Ambas funciones son muy similares pero con la diferencia que en la primera el pie derecho es el primero en moverse y en la segunda es el pie izquierdo. Estas funciones no reciben ni devuelven ningún argumento, sino que una vez llamadas se encargan de ejecutar el movimiento por completo. En esta secuencia, se utiliza una velocidad constante en los servomotores mediante la función set_MovSpeed de la librería de comunicación con el bus Dynamixel, y luego se envía la posición de cada articulación mediante la función SetPosSync.

Tabla 14. Descripción de la función StepForwardR

Función	StepForwardR
Descripción	Genera los movimientos en el robot para dar un paso adelante, empezando con el pie derecho
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 15. Descripción de la función StepForwardL

Función	StepForwardL
Descripción	Genera los movimientos en el robot para dar un paso adelante, empezando con el pie izquierdo
Argumento de entrada	ninguno
Argumento de salida	ninguno

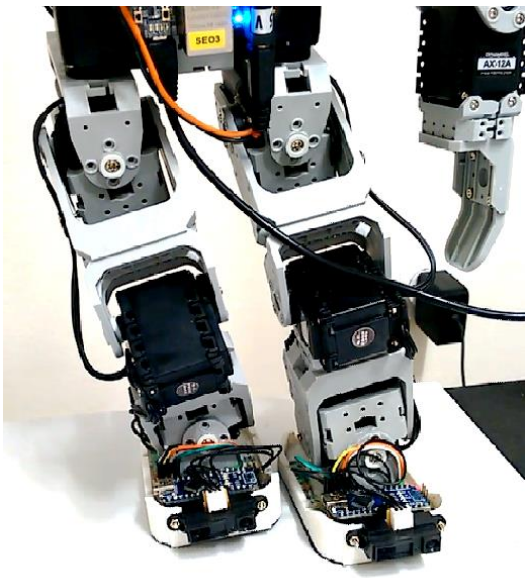


Figura 26. Piernas del robot durante el movimiento dar un paso adelante

6.2. Comportamientos de subir y bajar escalones

Otra manera en que se crearon secuencias de trayectorias en el robot fue mediante el uso del programa Roboplus. Esta aplicación es desarrollada por la empresa Robotis (creadores del robot Bioloid) y posee un modelo tridimensional del robot, en el cual se pueden variar las posiciones de las articulaciones y luego exportar los ángulos de estas. También posee una serie de movimientos predeterminados que vienen con la aplicación.

Usando este modelo se crearon los comportamientos de subir gradas y el de bajar gradas, para los cuales las trayectorias y ángulos fueron desarrolladas por Alcover, C. en [33]. La decisión de utilizar esta metodología se basó en el cronograma de actividades, por la inminente necesidad de validar el robot y el sistema en la competición CEABOT 2016, y la complejidad que lleva desarrollar desde el inicio este tipo de movimientos.

A estas secuencias fue necesario realizarles ajustes en los valores los ángulos, ya que la diferencia que existe en las partes del robot de este trabajo y el utilizado en [33], ocasionaba pérdidas de equilibrio en varios momentos del desplazamiento.

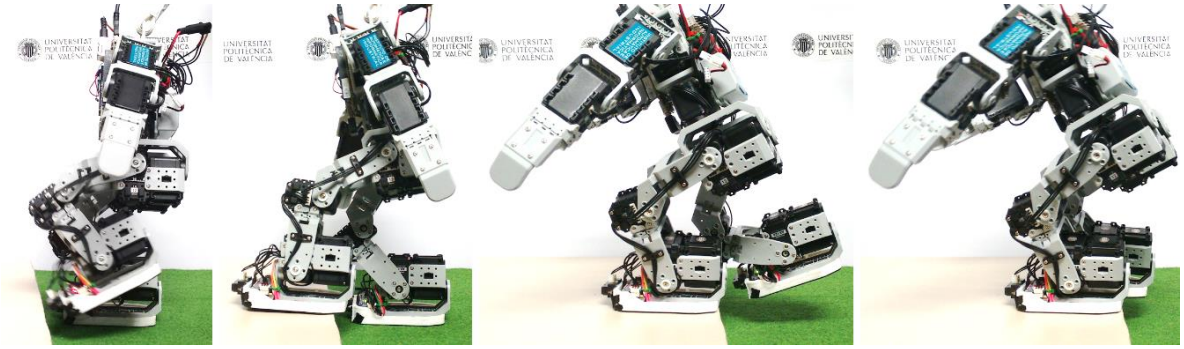


Figura 27. Secuencia de imágenes del Robot en el comportamiento UpStair

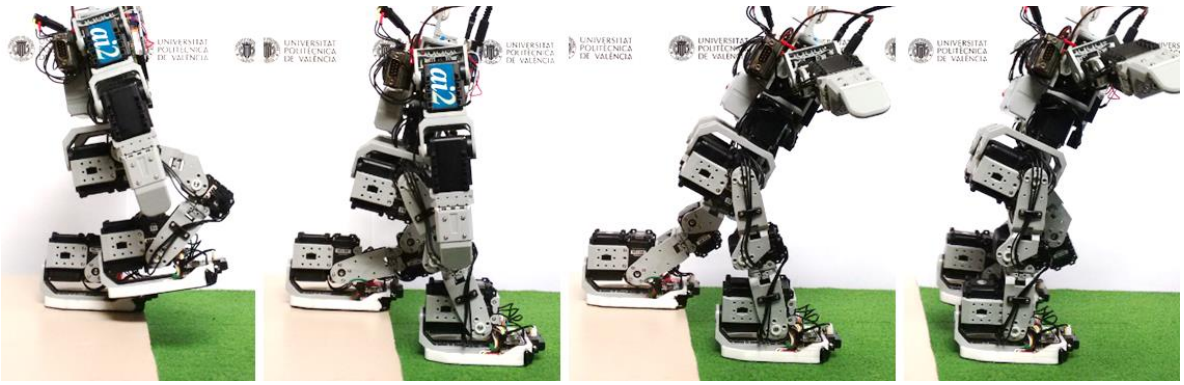


Figura 28. Secuencia de imágenes del Robot en el comportamiento DownStair

Estos movimientos fueron exportados como una matriz de ángulos y una matriz de velocidades, donde cada columna es una articulación y cada fila un instante de tiempo. Las matrices fueron importadas al fichero BioloidPosDef.h y se crearon las funciones UpStair y DownStair. Las funciones consisten en un bucle que recorre cada fila de las matrices y envía a los servomotores el valor de posición y velocidad deseado mediante la función set_VelSync y set_PosSync de la librería de comunicación con el bus Dynamixel. Entre cada fila recorrida fue necesario implementar un retardo de tiempo para que las articulaciones pudieran llegar a la posición deseada.

Tabla 16. Descripción de la función UpStair

Función	UpStair
Descripción	Genera los movimientos en el robot para subir un escalón
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 17. Descripción de la función DownStair

Función	DownStair
Descripción	Genera los movimientos en el robot para bajar un escalón
Argumento de entrada	ninguno
Argumento de salida	ninguno

6.3. Medio paso adelante y medio paso atrás

Debido a la lógica implementada para gobernar todos los comportamientos (que se explicará más adelante) fue necesario crear una función que diera medio paso adelante (sólo mover un pie) y luego otra función complementaria que diera el otro medio paso (mover el pie que quedó atrás). También fue necesario implementar una función que devolviera un pie posicionado adelante, hacia atrás hasta que estuviera con el otro pie. Para esto se utilizaron las matrices estándar de caminado que posee Roboplus, que se exportaron y añadieron al fichero BioloidPosDef.h.

Las funciones que implementan el comportamiento se hicieron de forma similar que para subir o bajar gradas, es decir un bucle que recorre cada línea de la matriz y envía la posición a los servomotores. La velocidad de los movimientos es constante y se define al inicio de la función.

Se crearon funciones complementarias para cuando el pie derecho es el primero en moverse: WalkSR1, WalkSR2 y BackSR; y otras funciones complementarias para cuando es el pie izquierdo el primero en moverse: WalkSL1, WalkSL2y BackSL. Ninguna de estas funciones recibe o devuelve ningún argumento.

Tabla 18. Descripción de la función WalkSR1

Función	WalkSR1
Descripción	Genera los movimientos en el robot para poner el pie derecho adelante
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 19. Descripción de la función WalkSR2

Función	WalkSR2
Descripción	Genera los movimientos en el robot para recoger el pie izquierdo cuando está atrás y ponerlo junto al derecho
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 20. Descripción de la función BackSR

Función	BackSR
Descripción	Genera los movimientos en el robot para recoger el pie derecho cuando está adelante y ponerlo junto al izquierdo
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 21. Descripción de la función WalkSL1

Función	WalkSL1
Descripción	Genera los movimientos en el robot para poner el pie izquierdo adelante
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 22. Descripción de la función WalkSL2

Función	WalkSL2
Descripción	Genera los movimientos en el robot para recoger el pie derecho cuando está atrás y ponerlo junto al izquierdo
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 23. Descripción de la función BackSL

Función	BackSL
Descripción	Genera los movimientos en el robot para recoger el pie izquierdo cuando está adelante y ponerlo junto al derecho
Argumento de entrada	ninguno
Argumento de salida	ninguno

6.4. Giro a la derecha, giro a la izquierda

Utilizando los movimientos preestablecidos para el robot en Roboplus, también se utilizaron las matrices de ángulos que describen un movimiento de giro corto a la izquierda y un movimiento de giro corto a la derecha; las cuales se importaron a BioloidPosDef.h.

Se crearon las funciones RightTurnMedium y LeftTurnMedium para crear los movimientos en base a las matrices mencionadas.

Tabla 24. Descripción de la función RightTurnMedium

Función	RightTurnMedium
Descripción	Genera los movimientos en el robot para subir un escalón
Argumento de entrada	ninguno
Argumento de salida	ninguno

Tabla 25. Descripción de la función LeftTurnMedium

Función	LeftTurnMedium
Descripción	Genera los movimientos en el robot para subir un escalón
Argumento de entrada	ninguno
Argumento de salida	ninguno

6.5. Máquina de estados del sistema

La manera de gobernar el conjunto de comportamientos creados (con miras a la validación del sistema en la competición CEABOT) fue mediante una máquina de estados, donde cada estado es un comportamiento y las decisiones de cambio de estado se toman en base a los valores de los sensores de los pies.

La premisa de la máquina de estados es que el robot estará de manera periódica en el comportamiento de dar un paso adelante, y toma valores de los sensores del pie cada vez que posiciona completamente un pie en el suelo. Específicamente se alternó entre dar un paso adelante derecho y dar un paso adelante izquierdo, para evitar posibles desviaciones en el caminar.

El movimiento de dar un paso adelante se implementó tanto con las funciones StepForwardR y StepForwardL, como con las funciones WalSR1, WalkSR2, WalkSL1 y WalkSL2. En la competición se utilizaron estas últimas, ya que describieron un movimiento más suave como se verá en el capítulo de resultados.

Si se detecta que en un pie los dos sensores de presión delanteros no tienen presión (lo que indica el inicio de un escalón hacia abajo), y el pie está posicionado hacia

adelante, se devolverá el pie hacia atrás y se iniciará el comportamiento de bajar escalones. Si el pie que detecta la baja presión esta junto al otro pie, se iniciará el movimiento de bajar escalón de una vez.

Estos pasos adelante se dan hasta que se detecte un obstáculo por medio de los sensores Sharp (de cualquier pie), en cuyo caso se analizará si existe diferencia entre el sensor derecho y el izquierdo, lo que implicaría que un pie está más cerca de una grada que el otro, y si la hubiera realiza giros a la derecha o izquierda para corregir esa diferencia (hasta un máximo de 3 giros). Luego se pasa al comportamiento de subir grada que realiza en el robot el movimiento de subir un escalón. Ahora el robot volverá a dar pasos y revisar sensores de manera periódica.

En siguiente diagrama (Figura 29) se muestra la lógica detrás de la máquina de estados implementada. Cada estado posee el nombre de la función que implementa ese comportamiento. Las flechas de cambio de estado obedecen a valores límites y condiciones creadas para los datos de los sensores.

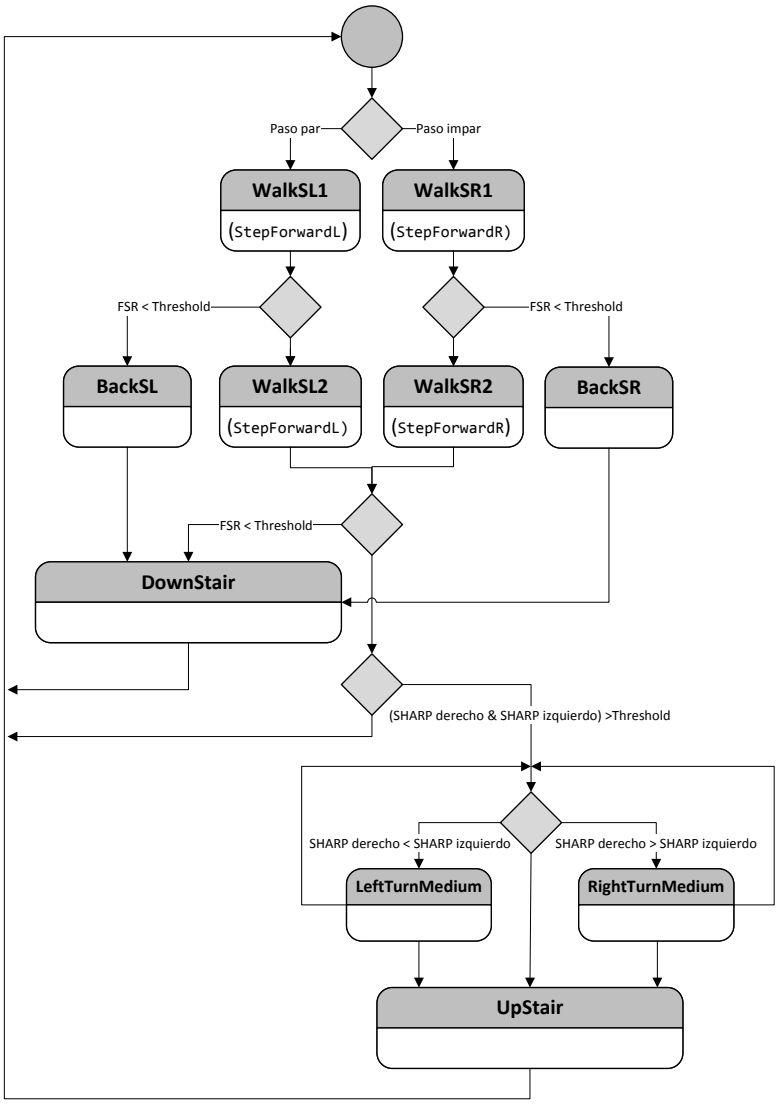


Figura 29. Diagrama de la máquina de estados implementada

Capítulo 7. Resultados experimentales

En este capítulo se analizarán los resultados obtenidos luego de la implementación de las diferentes rutinas en el robot Bioloid. Primero se mostrarán y analizarán los resultados cuantitativos obtenidos de los generadores de trayectorias y los sensores del robot durante el caminar. Luego se discutirán los resultados cualitativos de la validación del sistema en la competencia CEABOT.

7.1. Secuencias de caminado

Para analizar las secuencias de caminado se almacenaron los valores de los ángulos utilizados en las articulaciones del robot. En este caso se analizarán los valores de ángulos generados durante la primera mitad del comportamiento dar un paso adelante (empezando con el pie izquierdo) y el comportamiento WalkSL1 que da medio paso adelante con el pie izquierdo. Como se ven ambos comportamientos pretenden generar el mismo movimiento, pero uno se realizó utilizando las rutinas de generación de trayectorias y cinemática inversa que se generaron en este trabajo, y el otro comportamiento utilizó la matriz de ángulos generada por el software Roboplus.

En la Figura 30 se muestran los ángulos de las articulaciones de la pierna izquierda durante la primera mitad del comportamiento dar un paso adelante (creado a base funciones de trayectorias y cinemática inversa). En este gráfico se observa cómo los ángulos Roll de la cadera y del tobillo (las líneas están superpuestas) describen una ligera curva durante todo la secuencia, esto corresponde al movimiento de inclinar el cuerpo hacia un lado para ubicar el centro de masa sobre una pierna. Luego se observa el cambio más pronunciado en los valores de ángulos de Pitch en la cadera y la rodilla, los cuales corresponden al momento en que se levanta y luego se baja el pie (espacio entre las línea punteadas).

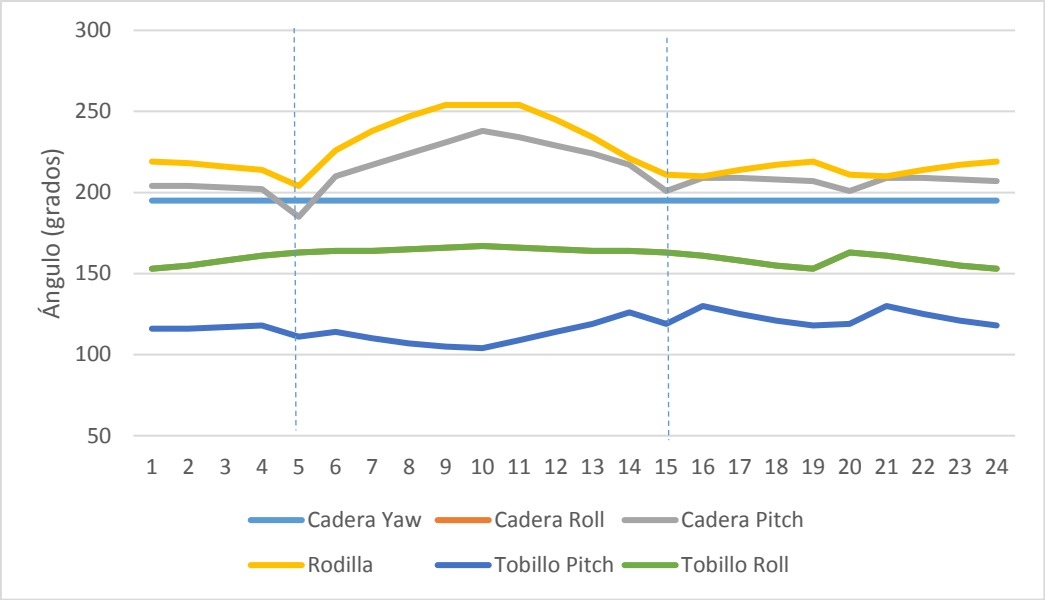


Figura 30. Ángulos de las articulaciones de la pierna izquierda del robot durante el comportamiento dar un paso adelante

El escenario paralelo al anterior, es el comportamiento WalkSL1 que da medio paso hacia adelante basándose en ángulos calculados por medio de Roboplus. La secuencia de estos ángulos durante el movimiento se muestra en la Figura 31, el cual tiene muchas similitudes con el gráfico de la Figura 30, por ejemplo la ligera curva que describen los ángulos Roll de la cadera y el tobillo y que corresponden al movimiento de inclinarse antes de levantar el pie. También se ve la similitud del movimiento del ángulo Pitch del tobillo, que incrementa levemente su valor, es decir se termina con el tobillo más flexionado hacia adelante.

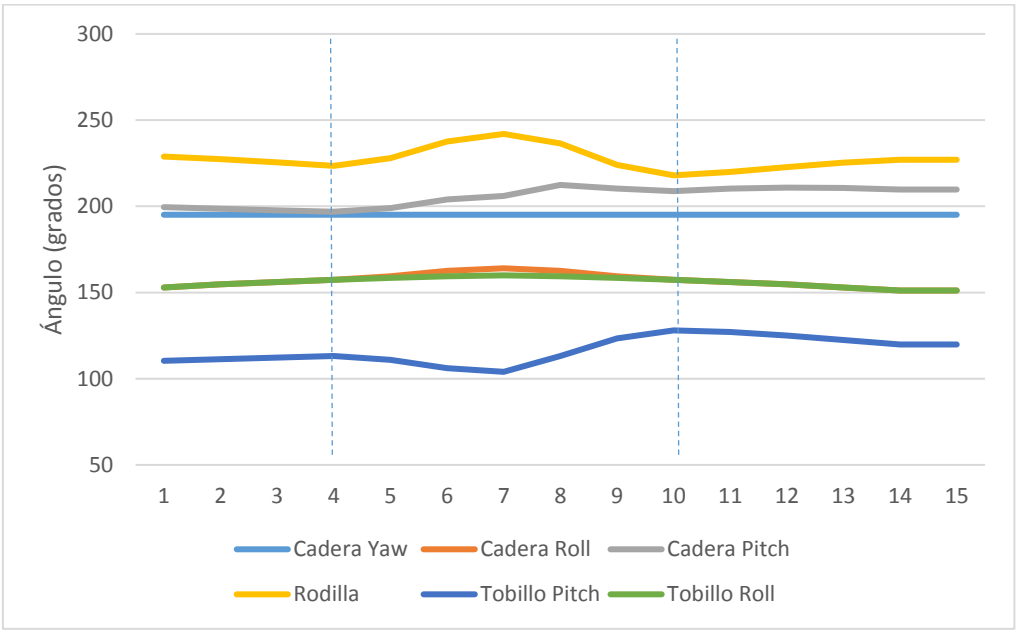


Figura 31. Ángulos de las articulaciones de la pierna izquierda del robot durante el comportamiento WalkSL1

Los comportamientos de ambos gráficos se diferencian en la magnitud del movimiento de levantar la pierna (entre líneas punteadas), ya que en la Figura 30 se observa un movimiento más abrupto del servomotor Pitch de la cadera y de la rodilla, lo que implica que se levanta más el pie. En la Figura 31, se observa un movimiento de los mismos ángulos, pero mucho más sutil, lo que indica que el pie se levanta menos o casi que se llega sólo a arrastrar. Esta diferencia fue un aprendizaje importante ya que en la práctica el movimiento de la segunda gráfica resultó mucho más suave y estable a la vista que el de la primera.

La diferencia que se observa en ambos gráficos en la cantidad de segmentos en que está divido el movimiento total (eje x) se debe a que el programa Roboplus divide sus secuencias en matrices de tamaño definido. En cambio en el generador de trayectorias desarrollado en el proyecto la cantidad de segmentos es configurable mediante una variable, por ejemplo cada trayectoria del movimiento dar un paso adelante se subdividió en 5 segmentos.

7.2. Sensores en la planta del pie

Para esta sección se almacenaron los valores de los sensores de presión y del sensor Sharp de ambos pies durante una secuencia de caminado. Se debe recordar que la toma de datos sucede cada vez que alguno de los pies se pone en el suelo.

En el caso del sensor Sharp, en la Figura 32 se observan los valores medidos durante una secuencia de caminado hasta que llega a un escalón. En este gráfico se observa como el valor de distancia al objeto va disminuyendo hasta que vuelve a subir. Esto se debe a la curva característica del sensor Sharp, en la cual entre 0 y 10 cm el comportamiento es inverso (ver Figura 18). En esta zona de funcionamiento es que se definió el umbral en el cual el robot decidirá que está lo suficientemente cerca para subir la grada. Este umbral se almaceno en la variable SharpInFrontThreshold.

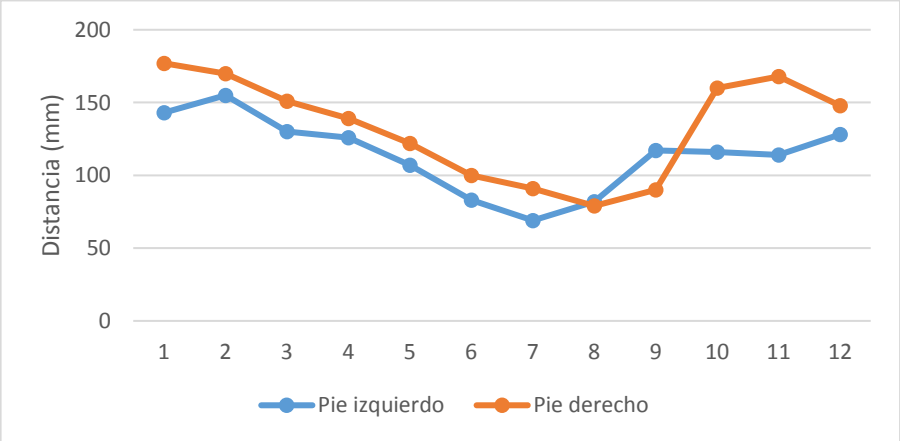


Figura 32. Lecturas de los sensores Sharp durante una secuencia de caminar hacia un escalón de subida

Como los valores de distancia medidos pueden tener 2 valores a los que corresponden, cuando el robot se va acercando al escalón y alcanza el valor mínimo, se valida una variable lógica que nos indica que el robot ya está en la proximidad del escalón. Así cuando mida un valor de distancia alto de nuevo, ya sabrá que se refiere en realidad a una distancia entre 0 y 10 cm.

Para la presión medida en el pie, se utilizaron los cuatro sensores FSR puestos cerca de las esquinas en la planta del pie. Se almacenaron los valores medidos durante un movimiento de caminado hasta que se detectó un escalón de bajada. En la Figura 33 se puede observar los valores de voltaje obtenidos en los sensores FSR del pie izquierda, ya que en este caso ese fue el que detectó el escalón. Se debe recordar que según el circuito implementado, el voltaje aumenta de manera proporcional a la presión.

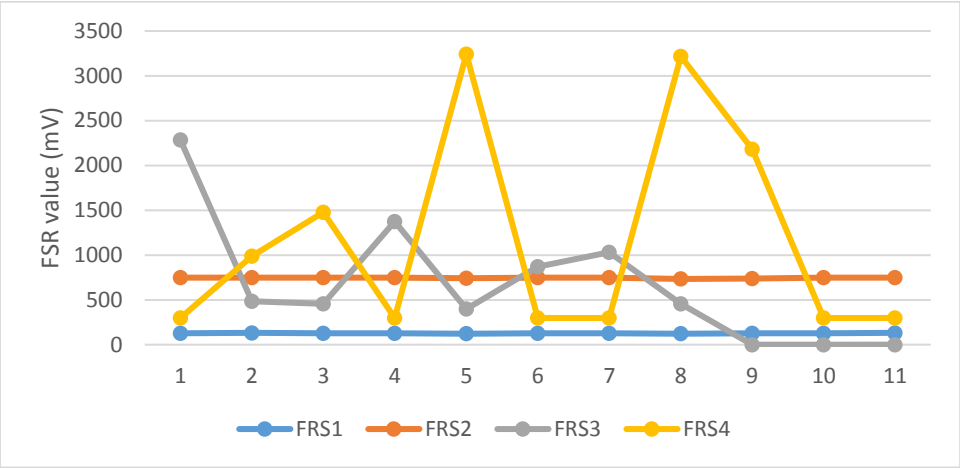


Figura 33. Lecturas de los sensores FSR durante una secuencia de caminar hacia un escalón de bajada

En este gráfico se observan varios comportamientos interesantes y que pueden servir para futuras mejoras del robot. Primero se observa que la presión no está dividida por igual en todos los sensores durante ninguna parte del movimiento. De hecho los sensores de atrás (FSR1 y FSR2) mantienen valores de voltaje (y por ende de presión) muy bajos durante todo el movimiento, estos valores incluso son constantes. Por tanto se podría concluir que en el pie, el peso del robot está soportado por la parte delantera de la planta.

Respecto a los sensores de presión de la parte delantera del pie (FSR3 y FSR4), estos tampoco muestran que la presión se reparta de manera equitativa adelante, sino que durante el transcurso del movimiento se intercala el lado que soporta mayor presión. Esto fue un aspecto que dificultó la lógica de detección de escalones de bajada, y al final se optó por utilizar el criterio de necesitar dos mediciones continuas con valores bajos de presión en ambos sensores para iniciar el movimiento de bajar escalón.

7.3. Validación del sistema en competición CEABOT

El sistema implementado en el trabajo fue validado en la competición nacional de Robots Humanoides de España, llamada CEABOT, y llevada a cabo como parte de las Jornadas de Automática 2017 en Madrid que organizó el Comité Español de Automática.

En esta competición se validó el sistema Bioloid en la prueba de subir y bajar escalones. Esta prueba consiste en subir una estructura compuesta por 3 escalones y luego bajarla (otros 3 escalones) en el menor tiempo posible y con la menor cantidad de fallos posibles.

En dicha prueba se tuvieron 2 intentos para realizar el recorrido. En el primer intento el robot sufrió la desconexión accidental del bus Dynamixel hacia el brazo izquierdo, lo cual causó que este brazo no se moviera y por ende afectó seriamente el equilibrio; y produjo que no pudiera subir o bajar exitosamente ningún escalón.

En el segundo intento el robot Bioloid se recuperó en buena manera de su fracaso inicial, logrando subir sin ninguna ayuda 2 escalones y bajando 1 escalón sin ayuda. Sin embargo el robot sufrió problemas de equilibrio en un momento de la secuencia de subida y en 2 momentos de la secuencia de bajada, lo que ocasionó su caída y que fuera necesario volverlo a su sitio. En el resultado final de la prueba se obtuvo el tercer lugar entre los participantes.

Los inconvenientes del robot en la competición no obedecieron a fallos en las rutinas generadas, las cuales funcionaron sin ningún tipo de problema y hasta resultaron de mucho interés para los participantes de otras universidades, debido a la novedad que implica controlar el robot Bioloid directamente con un sistema empotrado BeagleBone Black.

Los fallos se atribuyen a la estrategia seleccionada para el reto de subir y bajar escalones, ya que durante todos los movimientos de los comportamientos se buscó mantener una posición del robot similar a la humana, buscando recrear la manera en que la cinemática de un humano resolvería ese problema. Sin embargo en la competición, los mejores resultados obtenidos en la prueba de subir y bajar escalones los lograron los equipos que adaptaron los movimientos del robot a resolver el problema de manera eficiente aunque resultara extraño a la vista el tipo de movimientos. Por ejemplo, utilizaron secuencias de caminado con la cadera prácticamente sentada en los tobillos (para bajar el centro de masa) y para subir el escalón se utilizó un principio similar en donde se bajó el centro de masa lo más posible.

Capítulo 8. Conclusiones

En este capítulo se presentarán las conclusiones que se formulan como fin al desarrollo del trabajo de Fin de Máster. También se expondrán algunas limitaciones que se encontraron durante la implementación y se plantearán recomendaciones para futuros trabajos con el mismo sistema.

Primeramente en el Capítulo 2 se hizo una revisión del estado del arte en lo que respecta a robots humanoides. Se empezó por los sistemas más complejos y famosos, desarrollados por empresas o entidades gubernamentales. Luego se explicaron los esfuerzos de robots humanoides desarrollados por universidades alrededor del mundo y se pasó a los robots utilizados en competencias. Dentro de la gama de robots de pequeño tamaño se revisaron publicaciones que han ahondado en la resolución del problema de la cinemática inversa, y se pasó a revisar otras publicaciones que han tratado la generación de secuencias de caminado para estos robots.

La formulación en el trabajo de las ecuaciones que resuelven la cinemática inversa para los brazos y piernas del robot se dio de manera exitosa y permitió generar movimientos a puntos específicos del espacio; esto se pudo observar mediante las simulaciones realizadas y la implementación hecha en el robot.

El generador de trayectorias permitió realizar movimientos de manera controlada entre dos puntos y el conjunto de trayectorias generado se agrupó satisfactoriamente en comportamientos para describir distintos tipos de movimiento en el robot. Esta estructura de funciones quedó demostrada mediante los resultados analizados en el capítulo 7 para el comportamiento de dar un paso adelante.

El esfuerzo de integración de sensores en el pie para detectar obstáculos y en la planta del pie para medir presión, permitió obtener las gráficas expuestas en los resultados que validan la utilidad de esta funcionalidad. Los datos de los sensores fueron clave para realizar lógicas de control en bucle cerrado que permitieran la realización de la máquina de estados que gobierna el funcionamiento del robot.

El funcionamiento general del sistema fue validado en la competición de robots humanoides CEABOT. En la prueba de subir y bajar escalones, se obtuvo el tercer lugar y en la posición general se terminó también en la tercera posición.

Por los resultados expuestos anteriormente es que se comprueba que el esfuerzo de crear una interfaz de generación de aplicaciones para la locomoción del robot Bioloid utilizando un sistema empotrado BeagleBone Black, fue exitoso.

8.1. Limitaciones del sistema

Durante el desarrollo de las rutinas para mover el robot y durante las pruebas realizadas, se hizo notoria una limitación que presenta el sistema respecto a la velocidad de comunicación en el bus Dynamixel. En los robots Bioloid que utilizan el controlador del fabricante, la comunicación se realiza a una velocidad de 1 millón de baudios. La comunicación que se implementó en el sistema se realiza a 115200 baudios, esto significa una décima parte de la velocidad de comunicación del sistema original.

Esta limitante se debe a la librería serial que se utiliza para la comunicación en el sistema empotrado BeagleBone Black, la cual presenta como velocidad máxima 115200 baudios. A esta velocidad, con tramas de 10 bits y un total de 9 bytes a enviar (ejemplo típico de mensaje enviado), el mensaje dura aproximadamente 10 ms en ser enviado. A esto se le suma el tiempo de espera de la respuesta y la longitud de la respuesta, que en la práctica fueron otros 10 ms. Por tanto, la cantidad de datos que se elijan preguntar o la cantidad de mensajes que se envíen a los motores, pueden tener un efecto detectable en el tiempo de duración de una secuencia de movimientos. Por ejemplo, para consultar los valores de los 5 sensores de un solo pie se dura 1 segundo. Este tiempo en que el bus está siendo utilizado es un tiempo que se detecta a la vista.

Otra limitación del sistema es el tipo de sensor Sharp utilizado, ya que como se explicó en el desarrollo del trabajo, este viene especificado para detección de objetos en el rango de 10 cm a 80 cm; pero debido a la particularidad de la curva característica, se pudo usar en el rango de 0 a 10 cm sólo que con una precisión menor. Esta menor precisión hace que la lógica de control sea mucho más compleja y la posibilidad de errores sea mayor.

8.2. Recomendaciones para futuros trabajos

Retomando lo expuesto en las limitaciones, como primera recomendación se tiene el explorar la configuración del sistema para lograr mayores velocidades seriales, esto puede ser utilizando otra librería serial o directamente en la configuración del sistema empotrado. El tener mayores velocidades de comunicación con el bus Dynamixel (idealmente en el orden de los microsegundos) permitiría generar aplicaciones que requieren tiempos de monitoreo y respuesta rápidos.

Por ejemplo, el implementar un control en bucle cerrado del equilibrio del robot durante el movimiento de subir o bajar escalones requiere una velocidad muy alta, ya que desde el momento en que el robot comienza a perder el equilibrio y cuando ya es necesario una acción de control para corregir esta situación, sólo pasan fracciones de segundo.

Como segunda recomendación se tiene una mejora para la función de generación de trayectorias. Como se explicó en el capítulo 4, para la generación de trayectorias se traza una línea recta en el espacio entre el punto de inicio y el punto final, luego esta línea se divide en segmentos y se calcula la cinemática inversa para cada punto intermedio de la línea recta. Este tipo de control de posición se hace manera lineal, por lo cual resulta de valor explorar otro tipo de trayectorias que describan una curva sinusoidal o polinómica por ejemplo. Esto tendría la ventaja de presentar aceleraciones más graduales y por tanto movimientos del robot más suaves.

Otra recomendación que se desprende de las limitaciones expuestas, es el cambio del sensor Sharp por uno que esté creado para ser utilizado en un rango de entre 0 y 20 cm por ejemplo. El uso de un sensor de mayor precisión en este rango de distancia permitiría no sólo detectar mejor el escalón de frente, sino poder trabajar en el ajuste de la posición del robot antes de empezar el comportamiento de subir escalones, ya que podría asegurarse de estar a una distancia correcta y con ambos pies a una distancia similar.

Como última recomendación está la estrategia respecto a la prueba de subir y bajar escalones en la competición CEABOT, ya que como se discutió en los resultados, es importante modificar las secuencias de movimientos para trabajar con el centro de masa lo más bajo posible (aunque se vean "extraños" los movimientos del robot). Esto permitiría reducir las posibilidades de que el robot pierda en equilibrio durante el movimiento.

Capítulo 9. Referencias

[1] S. Behnke, «Humanoid Robots-From Fiction to Reality?,» *KI,* pp. 5-9, 2008.

[2] B. Gates, «A robot in every home,» *Scientific American,* pp. 58-65, 2007.

[3] C. B. A. S. P. J. S. R. &. Z. E. Balaguer, Libro Blanco de la Robotica. De la investigación al desarrollo y futuras aplicaciones, Espana: CEA-GTROB, 2007.

[4] Boston Dynamics, «ATLAS - Datasheet,» [En línea]. Available: http://archive.darpa.mil/roboticschallengetrialsarchive/files/ATLAS-Datasheet_v15_DARPA.PDF.

[5] Honda, «Asimo Technical FAQ,» [En línea]. Available: http://asimo.honda.com/downloads/pdf/asimo-technical-faq.pdf.

[6] E. Kisliuk, «Valkyrie - NASA,» Septiembre 2015. [En línea]. Available: http://www.nasa.gov/feature/valkyrie.

[7] NASA, «NASA Facts,» Agosto 2015. [En línea]. Available: https://www.nasa.gov/sites/default/files/atoms/files/fs_space_robotics_150908.pdf.

[8] I. W. K. J. Y. L. J. &. O. J. H. Park, «Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot 3: HUBO),» *In 5th IEEE-RAS International Conference on Humanoid Robots,* nº December, pp. 321-326, 2005.

[9] K. K. F. M. M. A. K. M. G. H. A. &. K. N. Kaneko, «Humanoid robot hrp-4-humanoid robotics platform with lightweight and slim body,» *In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems,* nº September, pp. 4400-4407, 2011.

[10] J. C. E. A. H. A. H. C. M. &. A. A. D. Reher, «Realizing Dynamic and Efficient Bipedal Locomotion on the Humanoid Robot DURUS,» *ICRA,* 2016.

[11] S. J. M. S. G. V. L. H. Q. H. I. H. J. .. &. Y. M. Yi, «Team THOR's Entry in the DARPA Robotics Challenge Trials 2013,» *Journal of Field Robotics,* nº 32, pp. 315-335, 2015.

[12] I. T. Y. A. H. H. J. &. H. D. W. Ha, «Development of open humanoid platform DARwIn-OP,» *In SICE Annual Conference (SICE),* nº Septiembre, pp. 2178-2181, 2011.

[13] D. H. V. B. P. K. C. M. J. L. P. .. &. M. B. Gouaillier, «Mechatronic design of NAO humanoid,» *IEEE International Conference in Robotics and Automation ICRA'09,* nº May, pp. 769-774, 2009.

[14] J. H. I. &. K. B. S. Han, «Educational robotic construction kit: Bioloid,» de *Proceedings of the 17th World Congress of the International Federation of Automatic Control (IFAC 2008)*, Seoul, Korea, 2008.

[15] C. N. &. P. M. Thai, «Using robotis bioloid systems for instructional robotics,» de *Proceedings of IEEE Southeastcon*, 2011.

[16] J. H. P. R. P. &. C. P. Wolf, «Bioloid based humanoid soccer robot design,» de *Proc. of the Second Workshop on Humanoid Soccer Robots*, 2007.

[17] J. V. B. A. R. D. A. I. J. M. &. R. V. M. Nunez, «Explicit analytic solution for inverse kinematics of bioloid humanoid robot,» de *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)*, Brazil, 2012 .

[18] J. R. C.-G. K. A. M.-M. J. A. C.-A. G. P.-S. G. I. &. P.-G. J. A. Cerritos-Jasso, «Kinematic Modeling of a Humanoid Soccer–Player: Applied to BIOLOID Premium Type A Robot.,» de *FIRA RoboWorld Congress*, Springer Berlin Heidelberg, 2013.

[19] H. D. &. T. C. S. Chiang, «Kinematics Analysis of a Biped Robot,» de *Proceedings of 2011 International Conference on Service and Interactive Robots*, Taichung, Taiwan, 2011.

[20] R. V. P. G. M. O. P. B. A. E. M. &. S. M. O'Flaherty, «Kinematics and Inverse Kinematics for the Humanoid Robot HUBO2+.,» Georgia Institute of Technology, 2013.

[21] M. A. P. H. A. &. L. C. G. Ali, «Closed-form inverse kinematic joint solution for humanoid robots,» de *International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[22] L. M. G. V. J. A. &. M. B. J. C. Contreras Tapia, «Análisis Cinemático del Robot Humanoide: Darwin OP.,» de *Memorias del primer Concurso de Investigación, Desarrollo e Innovación Tecnológica IDIT*, 2012.

[23] M. &. S. A. A. Akhtaruzzaman, «Joint demeanors of an anthropomorphic robot in designing the novel walking gait.,» de *8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011.

[24] L. E. O. L. I. P. J. A. &. N. J. V. Arias, «Patrón de marcha 3D de tipo cicloidal para humanoides y su aplicación al robot bioloid.,» *Revista Iberoamericana de Ingeniería Mecánica,* vol. 18, nº 1, pp. 3-22, 2013.

[25] A. &. D. F. Jiwankar, «Control of floor projection of center of mass for ascending stairs in BIOLOID,» de *International Conference on Computer, Communication and Control (IC4)*, 2015.

[26] C. H. A. R. T. &. L. T. Graf, «A robust closed-loop gait for the standard platform league humanoid,» de *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, París, Francia, 2009.

[27] P. Corke, Robotics, vision and control: fundamental algorithms in MATLAB, Springer, 2011.

[28] J. Vañó, *Sistema empotrado para sensorización en robot Bioloid,* Valencia: Tesis TFG, ETSInf. UPV, 2015.

[29] L. ". Fried, «Using an FSR,» Adafruit, 19 11 2015. [En línea]. Available: https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr.

[30] SHARP, «GP2Y0A21YK0F Distance Measuring Sensor Unit,» Diciembre 2006. [En línea]. Available: http://www.sharpsma.com/webfm_send/1489.

[31] J. A. Savage, «Arduino y Dynamixel AX-12,» SAVAGE ELECTRONICS, Enero 2011 . [En línea]. Available: http://savageelectronics.blogspot.com.es/2011/01/arduino-y-dynamixel-ax-12.html.

[32] D. Cisternes, *API para la interacción de sensores y actuadores de robot Bioloid, basado en BeagleBone Black,* Valencia: Tesis, DISA UPV, 2016.

[33] C. Alcover, *Desarrollo e implementación de un sistema de control de movimientos mediante sensores para robot humanoide,* Valencia: Tésis, ETSII UPV, 2015.

Capítulo 10. Anexos

Código generado en el proyecto

La lista completa de archivos de código usados para el sistema BeagleBone Black en el proyecto, se encuentra disponible en un proyecto de GitHub en la siguiente dirección web:

https://github.com/jcbrenes/Bioloid-Locomotion-API.git

El código Arduino usado para el controlador de la planta del pie se encuentra en la siguiente dirección web:

https://gist.github.com/jcbrenes/2e53f0fdceb828af672f261cadfb0e34