Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Proyecto de Tecnológico

Monitoreo y manipulación remota de un robot humanoide

Armando Sanmiguel Luévano

209200162

Trimestre 2014 Primavera

4 de julio de 2014

M. en C. José Alfredo Estrada Soto

Profesor Titular ”C”

Departamento de Electrónica

Yo, M. en C. José Alfredo Estrada Soto, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma del asesor

Yo, Armando Sanmiguel Luévano, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma del alumno

”Si hay algo en mí que pueda ser llamado religioso es la ilimitada admiración por la estructura del mundo, hasta donde nuestra ciencia puede revelarla.”
Albert Einstein

”Si he hecho descubrimientos invaluables ha sido más por tener paciencia que cualquier otro talento.”
Isaac Newton

”Toda decisión que toma una persona proviene de sus valores y sus metas. Las personas pueden tener muchas metas y valores: fama, ganancias, amor, supervivencia, diversión y libertad son sólo algunas de las metas que una buena persona puede tener. Cuando la meta es ayudar a los demás tanto como a uno mismo, lo llamamos idealismo.”
Richard Stallman

Resumen

El desarrollo de robots es una actividad en constante evolución que requiere la utilización del conocimiento de distintas disciplinas como son: la electrónica, la mecánica, la física y la computación. La investigación en robótica responde a la necesidad de construir robots con aplicación en entornos reales que repercutan en beneficio de la sociedad. Actualmente la robótica se aplica exitosamente en la automatización de procesos productivos, la medicina y la exploración espacial, por mencionar algunos campos de aplicación.

La programación de un robot es una tarea fundamental en el diseño y construcción de sistemas robóticos y ambientes automatizados para lograr la adecuada implementación de algoritmos de control, inteligencia artificial y visión, entre otros.

En este trabajo se programó un sistema de monitoreo y manipulación remota de un robot humanoide, capaz de guiarlo a través de un entorno que puede ser censado mediante la imagen que transmite la cámara del robot. De esta manera se puede manipular al robot en una gran variedad de entornos que permitan tener un primer acercamiento con cualquier lugar de interés, tal como se hace en la exploración espacial.

Índice

Lista de figuras

1. Introducción

Actualmente hay una amplia gama de robots: móviles, industriales, aeroespaciales, humanoides, entre otros; su empleo es tan diverso que va desde simples juguetes hasta sofisticados sistemas que pueden reemplazar a los humanos en determinadas situaciones. Una de las metas de quienes se dedican a la mecatrónica -y otras disciplinas afines-, es el desarrollo de robots orientados a la exploración, búsqueda y/o rescate. De esta manera se manda a los robots para prevenir y evitar alguna situación que podría poner en riesgo a un ser humano. Generalmente se han estado empleando robots móviles construidos con una variedad de aditamentos. Sin embargo, desde hace algunos años los robots humanoides han venido ganando terreno.

Un robot humanoide es un robot en el cual la forma de su cuerpo está construida a semejanza del cuerpo humano. Un diseño humanoide puede ser utilizado para un propósito funcional y, a diferencia de los robots móviles, puede manipular herramientas hechas para el humano y sus entornos.

Darwin-OP (Dynamic Anthropomorphic Robot with Intelligence–Open Platform) es un robot humanoide creado por la compañía Robotis; es una plataforma robot-humanoide-miniatura con poder computacional avanzado, sensores sofisticados, alta capacidad de carga útil y habilidad motora dinámica; Darwin-OP es empleado principalmente en la academia para cuestiones de investigación y educación. Dadas sus características físicas y prestaciones, se empleó este robot para la realización de este proyecto.

El presente trabajo tiene como objetivo la implementación de un sistema de monitoreo y manipulación remota del robot humanoide Darwin-OP, que permitirá al usuario guiarlo a través de su entorno a lugares de interés gracias a la manipulación motriz y monitoreo con la cámara del robot, con que contará el sistema. Gracias a este sistema se podrán realizar exploraciones a distancia sin la necesidad de la presencia de un humano en el lugar.

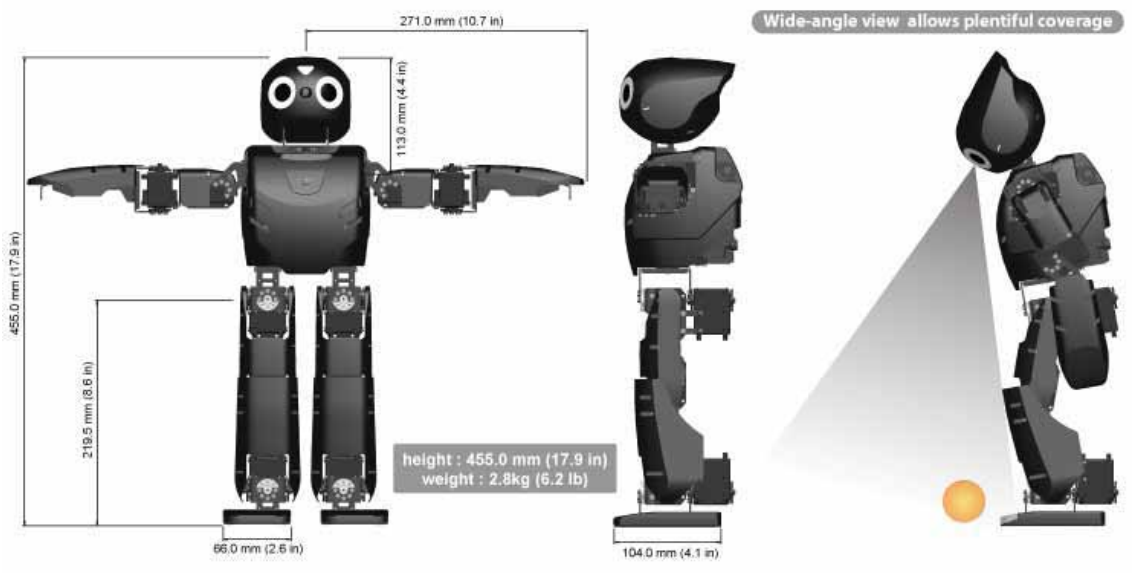

Figura 1: Robot Darwin-OP.

2. Justificación

El empleo de robots móviles en situaciones de exploración, búsqueda y/o rescate en zonas de difícil acceso o de riesgo para seres humanos ha sido ampliamente explotada. Sin embargo, hay una serie de casos en los cuales se requiere de robots con características similares a un humano, ya sea para acceder a un determinado sitio donde el movimiento de articulaciones sea la clave para lograrlo, o bien, para manipular algún tipo de herramienta u objeto como lo haría cualquier persona. Ya que el robot humanoide es una propuesta relativamente nueva en el campo de aquellas disciplinas dedicadas a la investigación y desarrollo de todo lo relacionado con la robótica, existe una gran demanda de aplicaciones dedicadas a resolver problemas mediante el empleo de este tipo de robots.

Así, se pretende implementar un sistema que permita manipular un robot humanoide a control remoto de manera tal que, mediante un dispositivo de video, el usuario pueda enterarse de lo que ocurre en el sitio, tome una decisión sobre la acción a seguir y se la haga llegar al robot para que la ejecute.

3. Antecedentes

Un robot es una entidad virtual o mecánica artificial. En la práctica, esto es por lo general un sistema electromecánico que, por su apariencia o sus movimientos, ofrece la sensación de tener un propósito propio. La independencia creada en sus movimientos hace que sus acciones sean la razón de un estudio razonable y profundo en el área de la ciencia y tecnología. La palabra robot puede referirse tanto a mecanismos físicos como a sistemas virtuales de software, aunque suele aludirse a los segundos con el término de bots.

No hay un consenso sobre qué máquinas pueden ser consideradas robots, pero sí existe un acuerdo general entre los expertos y el público sobre que los robots tienden a hacer parte o todo lo que sigue: moverse, hacer funcionar un brazo mecánico, sentir y manipular su entorno y mostrar un comportamiento inteligente, especialmente si ese comportamiento imita al de los humanos o a otros animales. Actualmente podría considerarse que un robot es una computadora con la capacidad y el propósito de movimiento que en general es capaz de desarrollar múltiples tareas de manera flexible según su programación; así que podría diferenciarse de algún electrodoméstico específico.

Aunque las historias sobre ayudantes y acompañantes artificiales, así como los intentos de crearlos, tienen una larga historia, las máquinas totalmente autónomas no aparecieron hasta el siglo XX. El primer robot programable y dirigido de forma digital, el Unimate, fue instalado en 1961 para levantar piezas calientes de metal de una máquina de tinte y colocarlas.

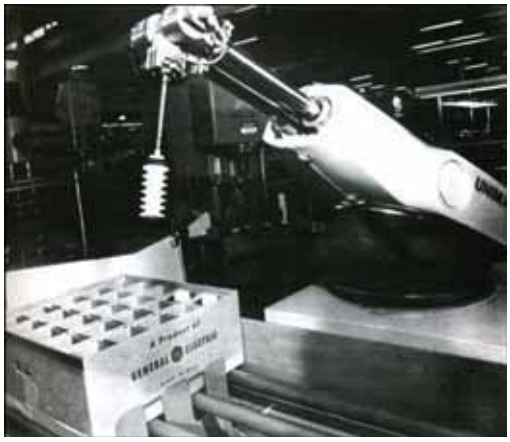

Figura 2: Robot Unimate.

Por lo general, la gente reacciona de forma positiva ante los robots con los que se encuentra. Los robots domésticos para la limpieza y mantenimiento del hogar son cada vez más comunes en los hogares. No obstante, existe una cierta ansiedad sobre el impacto económico de la automatización y la amenaza del armamento robótico, una ansiedad que se ve reflejada en el retrato a menudo

perverso y malvado de robots presentes en obras de la cultura popular. Comparados con sus colegas de ficción, los robots reales siguen siendo limitados.

3.1. Los primeros autómatas

En el siglo IV antes de Cristo, el matemático griego Arquitas de Tarento construyó un ave mecánica que funcionaba con vapor y al que llamó ”La paloma”. También el ingeniero Herón de Alejandría (10-70 d. C.) creó numerosos dispositivos automáticos que los usuarios podían modificar, y describió máquinas accionadas por presión de aire, vapor y agua. Por su parte, el estudioso chino Su Song levantó una torre de reloj en 1088 con figuras mecánicas que daban las campanadas de las horas.

Al Jazarií[1] (1136–1206), diseñó y construyó una serie de máquinas automatizadas, entre los que había útiles de cocina, autómatas musicales que funcionaban con agua, y en 1206 los primeros robots humanoides programables. Las máquinas tenían el aspecto de cuatro músicos a bordo de un bote en un lago, entreteniendo a los invitados en las fiestas reales. Su mecanismo tenía un tambor programable con clavijas que chocaban con pequeñas palancas que accionaban instrumentos de percusión. Podían cambiarse los ritmos y patrones que tocaba el tamborilero moviendo las clavijas.

3.2. Desarrollo moderno

El artesano japonés Hisashige Tanaka (1799–1881), conocido como el”Edison japonés”, creó una serie de juguetes mecánicos extremadamente complejos, algunos de los cuales servían té, disparaban flechas retiradas de un carcaj[2] e incluso trazaban un kanji[3].

Por otra parte, desde la generalización del uso de la tecnología en procesos de producción con la Revolución Industrial se intentó la construcción de dispositivos automáticos que ayudaran o sustituyeran al hombre. Entre ellos destacaron los Jaquemarts, muñecos de dos o más posiciones que golpean campanas accionados por mecanismos de relojería china y japonesa.

Robots equipados con una sola rueda fueron utilizados para llevar a cabo investigaciones sobre conducta, navegación y planeo de ruta. Cuando estuvieron listos para intentar nuevamente con los robots caminantes, comenzaron con pequeños hexápodos y otros tipos de robots de múltiples patas. Estos robots imitaban insectos y artrópodos en funciones y forma. Como se ha hecho notar anteriormente, la tendencia se dirige hacia ese tipo de cuerpos que ofrecen gran flexibilidad y han

[1] Inventor musulmán de la dinastía Artuqid.
[2] Es un cilindro de piel, madera y/o tela usada por los arqueros para transportar las flechas.
[3] Caracteres utilizados en la escritura japonesa.

probado adaptabilidad a cualquier ambiente. Con más de 4 piernas, estos robots son estáticamente estables lo que hace que el trabajar con ellos sea más sencillo. Sólo recientemente se han hecho progresos hacia los robots con locomoción bípeda.

En 2002 Honda y Sony, comenzaron a vender comercialmente robots humanoides como ”mascotas”. Los robots con forma de perro o de serpiente se encuentran, sin embargo, en una fase de producción muy amplia, el ejemplo más notorio ha sido Aibo de Sony.

3.3. La robótica en la actualidad

En la actualidad, los robots comerciales e industriales son ampliamente utilizados, y realizan tareas de forma más exacta o más barata que los humanos. También se les utiliza en trabajos demasiado sucios, peligrosos o tediosos para los humanos. Los robots son muy utilizados en plantas de manufactura, montaje y embalaje, en transporte, en exploraciones en la Tierra y en el espacio, cirugía, armamento, investigación en laboratorios y en la producción en masa de bienes industriales o de consumo.

Otras aplicaciones incluyen la limpieza de residuos tóxicos, minería, búsqueda y rescate de personas y localización de minas terrestres.

Existe una gran esperanza, especialmente en Japón, de que el cuidado del hogar para la población de edad avanzada pueda ser desempeñado por robots.

Los robots parecen estar abaratándose y reduciendo su tamaño, una tendencia relacionada con la miniaturización de los componentes electrónicos que se utilizan para controlarlos. Además, muchos robots son diseñados en simuladores mucho antes de construirse y de que interactúen con ambientes físicos reales.

Además de los campos mencionados, hay modelos trabajando en el sector educativo, servicios (por ejemplo, en lugar de recepcionistas humanos o vigilancia) y tareas de búsqueda y rescate.

3.4. Robots exploradores

Los robots exploradores son aquellos que se han diseñado con el fin de conocer y explorar un lugar en concreto con el fin de evitar poner en peligro la vida de personas o conseguir acceso a entornos en los que el hombre no llega. Este tipo de robots cuentan con cámaras integradas para capturar imágenes que posteriormente son retransmitidas a la base para su posterior análisis. Aunque los robots exploradores pueden ser controlados de forma remota, también existen unidades

capaces de tomar decisiones gracias a que cuentan con inteligencia artificial.

4. Objetivos

4.1. General

Diseñar e implementar un sistema de monitoreo y manipulación remota de un robot humanoide.

4.2. Específicos

1. Diseñar e implementar la interfaz de usuario para controlar al robot humanoide.
2. Diseñar e implementar la interfaz de usuario para visualizar la señal de video transmitida por la cámara del robot humanoide.
3. Diseñar e implementar la aplicación de control que permita al robot ejecutar las órdenes que reciba desde la interfaz de usuario.
4. Diseñar e implementar la interfaz de control que comunique la interfaz de usuario con la aplicación que opera al robot.

5. Marco teórico

Un robot es una máquina o un agente artificial virtual, generalmente una máquina electromecánica que es guiada por un programa de computadora o circuitos electrónicos. Los robots pueden ser autónomos o semi-autónomos y van desde los humanoides como Honda's Advance Step in Innovative Mobility (ASIMO) y de TOSY's TOSY Ping Pong Playing Robot (TOPIO) a robots industriales, robots enjambre programados colectivamente, e incluso microscópicos nano robots. Al imitar un aspecto realista o movimientos automatizados, un robot puede aparentar inteligencia o pensar por si mismo.

La rama de la tecnología que se ocupa del diseño, construcción, operación y aplicación de robots, así como de sistemas de computo para su control, retro-alimentación sensorial y procesamiento de información, es la robótica. Estas tecnologías se ocupan de las máquinas autómatas que pueden tomar el lugar de los humanos en entornos peligrosos o procesos de fabricación, o parecerse a los humanos en apariencia, comportamiento y/o la cognición.

5.1. Robot humanoide

Un robot humanoide es un robot cuyo cuerpo es construido a semejanza del cuerpo humano. Un diseño humanoide podría ser para fines funcionales, tales como la interacción con herramientas y entornos humanos, con fines experimentales, como el estudio de la locomoción bípeda, o para otros fines. En general, los robots humanoides tienen un torso, una cabeza, dos brazos y dos piernas, aunque algunas formas de robots humanoides pueden modelar sólo una parte del cuerpo, por ejemplo, de la cintura para arriba. Algunos robots humanoides pueden tener cabezas diseñado para replicar los rasgos faciales humanos, tales como los ojos y la boca. Los androides son robots humanoides construidos para parecer humanos estéticamente.

Los robots humanoides se utilizan como una herramienta de investigación en diversas áreas científicas. Los investigadores necesitan entender la estructura del cuerpo humano y su comportamiento biomecánica para construir y estudiar los robots humanoides. Por otro lado, el intento de la simulación del cuerpo humano conduce a una mejor comprensión de la misma.

La cognición humana es un campo de estudio que se centra en cómo los seres humanos aprenden de la información sensorial con el fin de adquirir las habilidades perceptivas y motoras. Este conocimiento se utiliza para desarrollar modelos computacionales de la conducta humana y se ha ido mejorando con el tiempo.

Además de la investigación, los robots humanoides se están desarrollando para realizar tareas humanas como la asistencia personal, en el que deben ser capaces de asistir a los enfermos y an-

cianos. Trabajos regulares como recepcionista o trabajador de línea de fabricación de automóviles también son adecuados para los humanoides. En esencia, ya que pueden utilizar herramientas y operar los equipos y el material diseñado para la forma humana, los humanoides podrían realizar en teoría cualquier tarea que un ser humano puede, siempre y cuando tengan el software adecuado. Sin embargo, la complejidad de hacerlo es aparentemente grande.

5.2. Darwin-OP

DARWIN-OP (Dynamic Anthropomorphic Robot with Intelligence–Open Platform) es una robot humanoide miniatura con poder computacional avanzado, sensores sofisticados, alta capacidad de carga y capacidad de movimiento dinámico desarrollado. Construido por el fabricante coreano ROBOTIS, en colaboración con el Tecnológico de Virginia, la Universidad de Purdue y la Universidad de Pennsylvania. DARWIN-OP tiene veinte grados de libertad cada uno controlado por un servomotor Dynamixel MX-28T. El MX-28T tiene un par de bloqueo de 24 kgf·cm (a 12 V, 1,5 A) y un rango de 360 gados de movimiento.

Figura 3: Diagrama de Darwin-OP.

5.2.1. Especificaciones

- Altura: 454.5 mm (17.89 pulgadas)
- Peso: 2.9 kg (6.4 libras)
- Por defecto la velocidad al caminar: 24.0 cm/s (9.5 in/s) 0.25 s/paso
- Por defecto el tiempo para levantarse desde el suelo: 2.8 s (boca abajo) y 3.9 s (boca arriba)
- Built-in PC: 1.6 GHz Intel Atom Z530 (32 bits) on-board SSD flash 4 GB
- Administrador de control (CM-730): ARM CortexM3 STM32F103RE 72 MHz
- 20 actuadores MX-28T (6 pierna DOF $\times$ 2 + 3 brazo DOF $\times$ 2 + 2 DOF cuello) con engranajes metálicos
- 3 Mbit/s bus Dynamixel de alta velocidad para el control conjunto
- Giroscopio de 3 ejes, acelerómetro de 3 ejes, botón $\times$ 3, micrófono de detección $\times$ 2
- Funcionalidad versátil (puede aceptar futuros periféricos)

Figura 4: Diagrama de servomotor.

5.2.2. Framework

Diagrama de clases del framework de Darwin

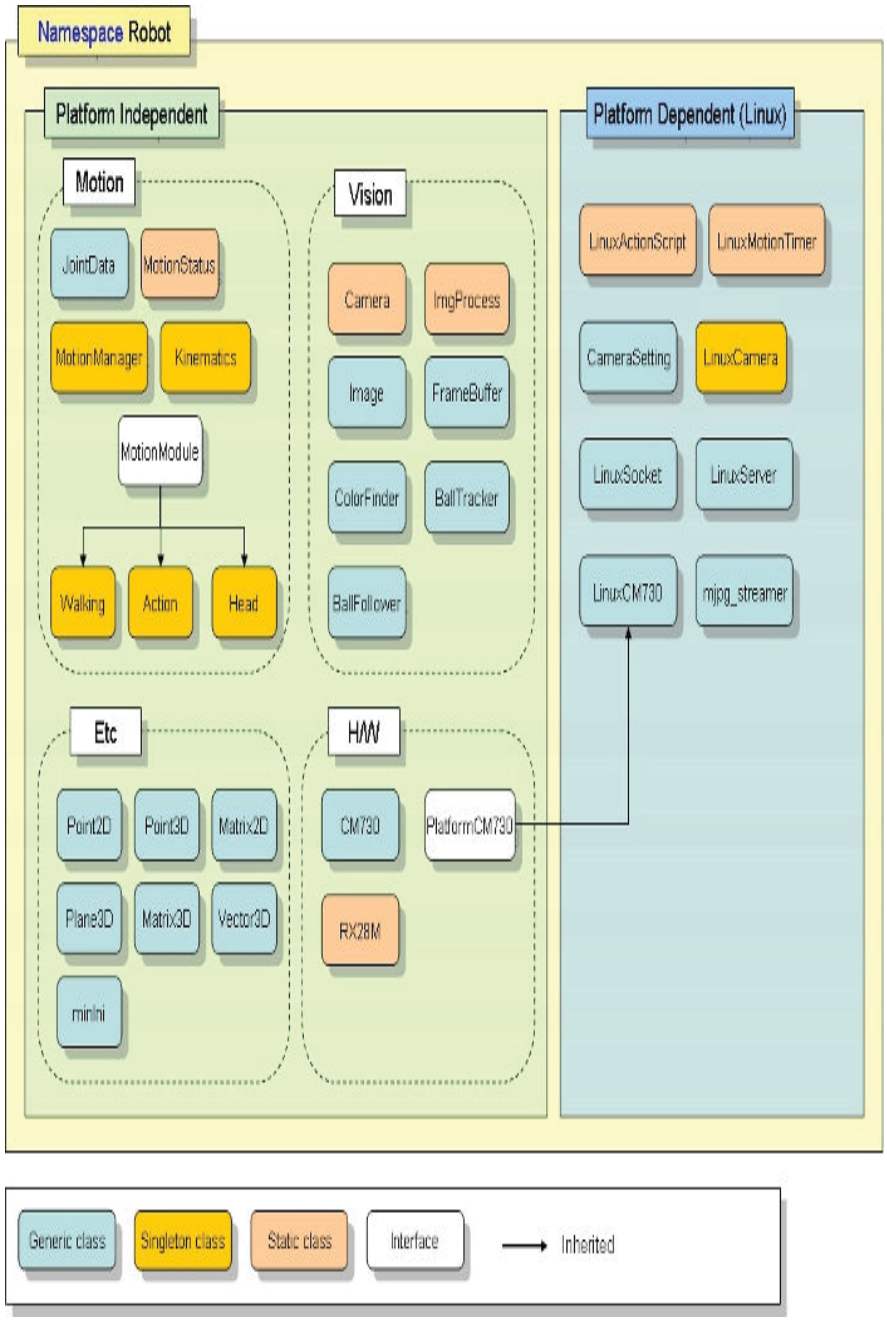

Figura 5: Diagrama de clases.

Pipeline de framework de Darwin

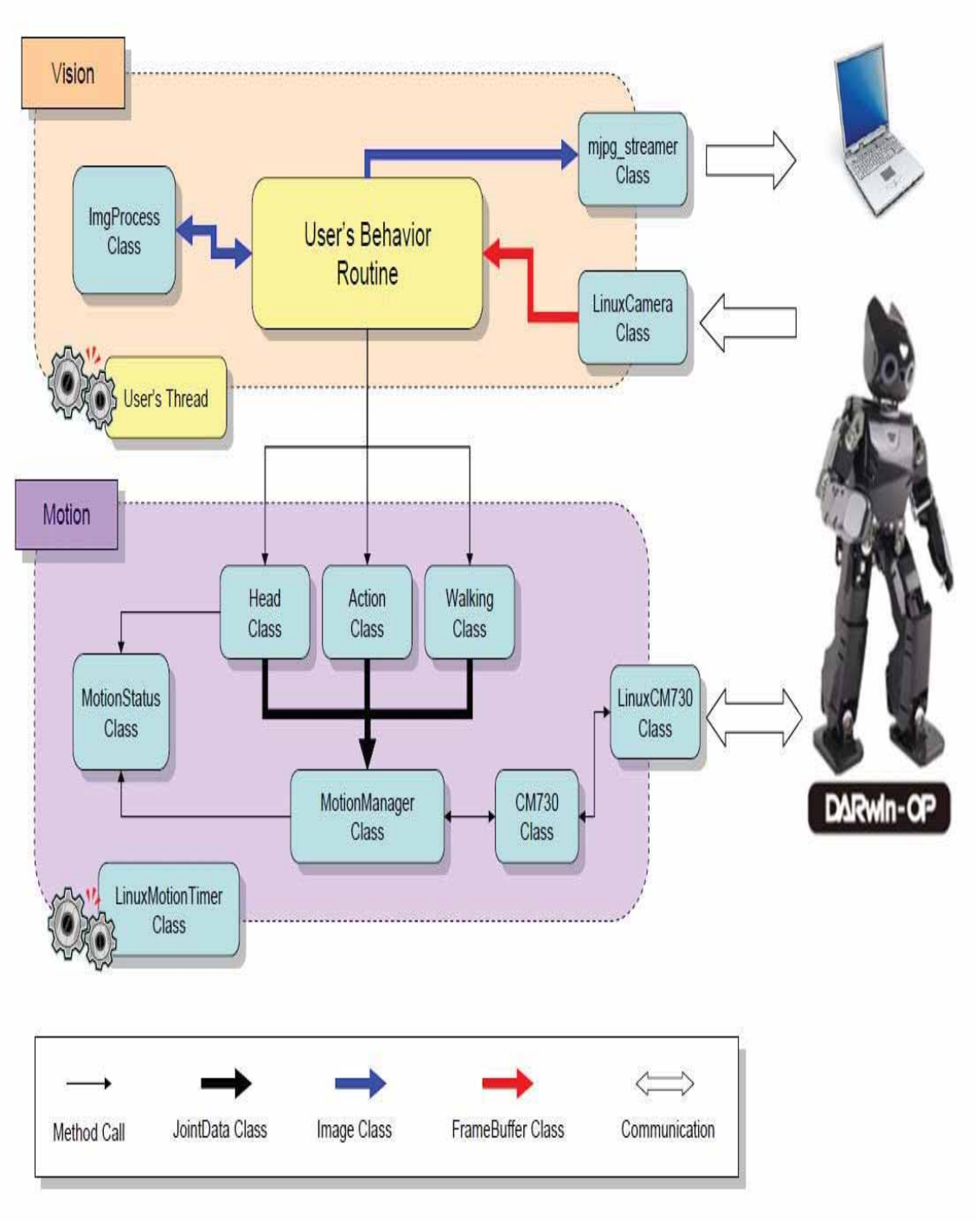

Figura 6: Diagrama de pipeline de framework.

5.3. Ventajas y desventajas

- Facilitan tareas pesadas y riesgosas para los seres humanos.
- Muestran un gran avance tecnológico a nivel mundial.
- Pueden ser perdidas aceptables en situaciones de riesgo.
- No necesitan condiciones optimas como: comida, agua, dormir, oxígeno, etc.
- Son fácilmente reemplazables.
- Muchas personas pueden perder sus trabajos para ser remplazados por robots.
- Son muy costosos.
- Refacciones caras.
- Dependen de tener una buena programación.
- Necesitan un suministro constante de electricidad.

6. Desarrollo del proyecto

El trabajo desarrollado en este proyecto fue dividido en tres módulos, los cuales se muestran en la figura 7.

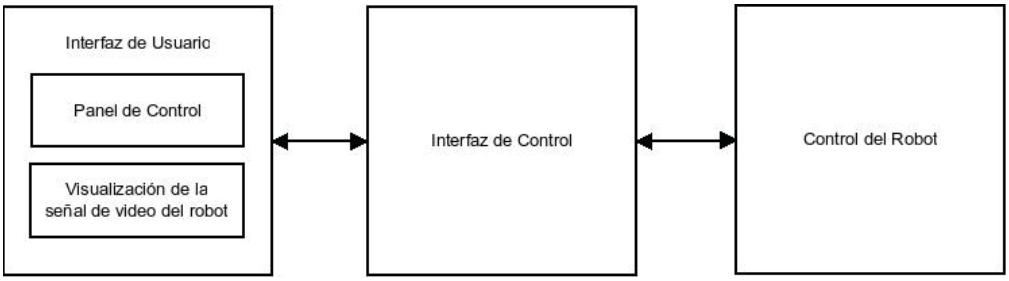

Figura 7: Descripción básica de los módulos del proyecto.

6.1. Diagrama de casos de uso

El desarrollo del proyecto se realizo con base en los siguientes casos de uso. Debido a que toda la interacción del usuario se realiza con la interfaz de usuario unicamente.

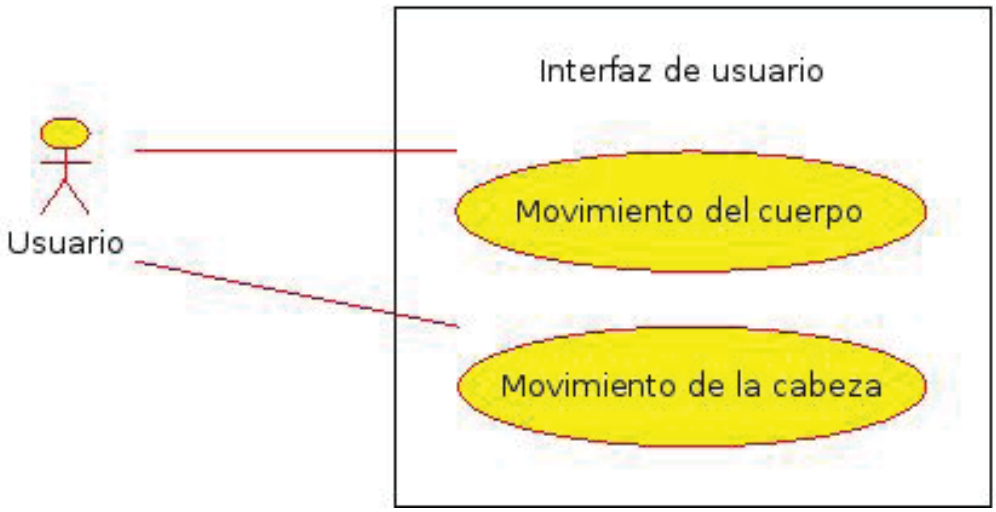

Figura 8: Caso de uso movimiento.

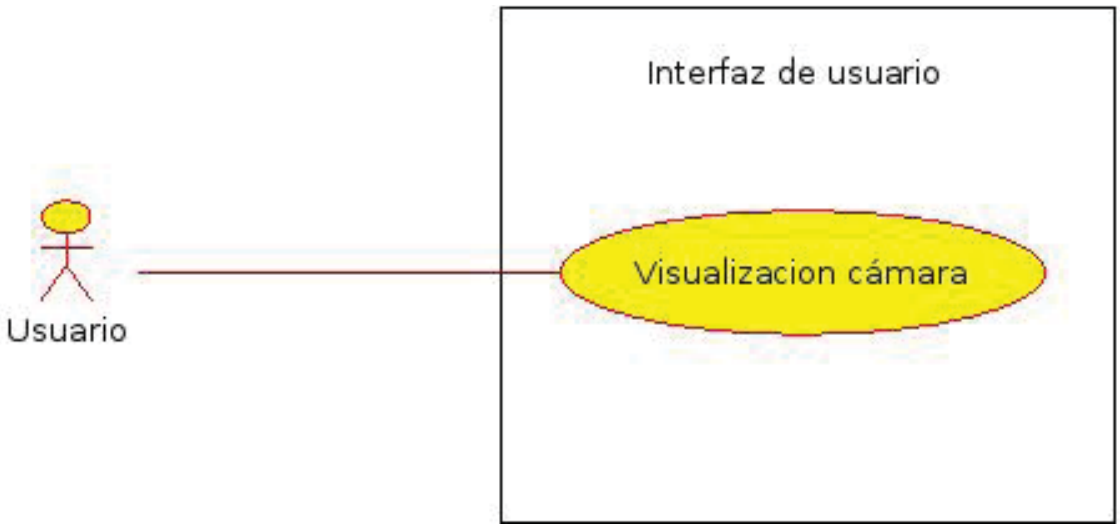

Figura 9: Caso de uso visión.

6.2. Diagrama de componentes

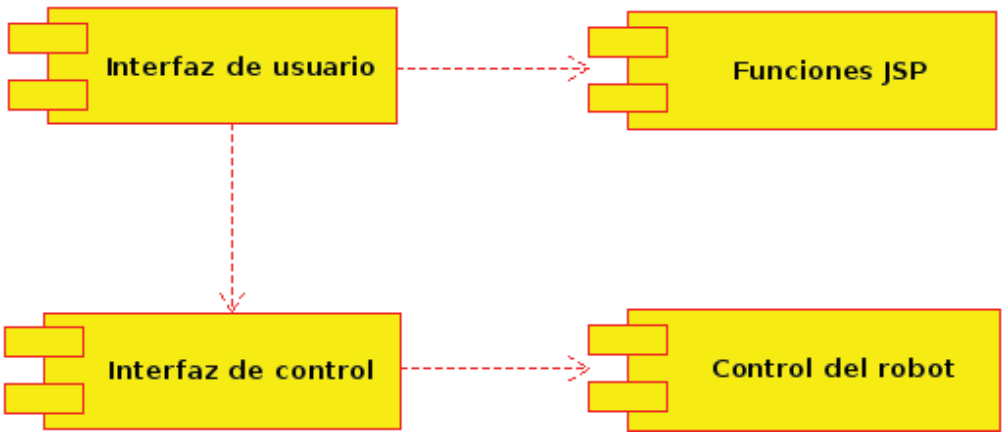

Figura 10: Diagrama de componentes.

6.3. Diagrama de secuencia

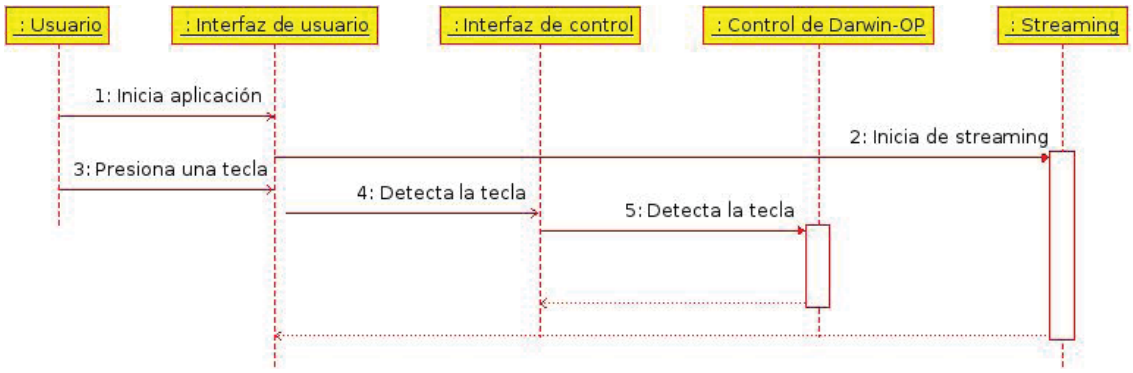

Figura 11: Diagrama de secuencia.

6.4. Implementación

La implementación del proyecto completo requirió de varios lenguajes de programación debido a las diferentes prestaciones que otorga cada lenguaje, así como a que está dedicado cada uno. Las decisiones tomadas para elegir los lenguajes fueron aprovechando los lenguajes usados por el fabricante del robot para crear su API. De igual manera, gracias a la investigación que se hizo, se encontraron varias herramientas útiles para mejorar la realización de este proyecto.

6.4.1. Interfaz de usuario

La implementación de la interfaz de usuario se dividió en dos módulos, uno para el control motriz del cuerpo y cabeza de Darwin-OP, y el otro para la visualización de la señal de video.

La interfaz de usuario fue programada con el lenguaje HTML[4], creando una página web amigable en la que se dan las instrucciones de como manejar al robot y de igual manera donde se puede observar las imágenes transmitidas por la cámara de Darwin.

Dentro del código de la página se incrusto código en javascript[5], para crear funciones que permitan la recepción de la imagen de video de manera continua.

[4]Del ingles HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.

[5]Es un lenguaje de programación interpretado, utilizado principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Por último en la interfaz de usuario se incrusto una función en PHP[6] para ejecutar del lado del servidor el control del robot, esto es para conectar la interfaz de usuario con el programa de control del robot Darwin-OP.

La imagen 12 es de la página web final que representa al control para el manejo y monitoreo del robot humanoide Darwin-OP. En esta imagen se muestra del lado izquierdo las instrucciones para el manejo del robot, la parte inferior de la pantalla contiene la representación del teclado que corresponde a los utilizados para controlar a Darwin, mientras que en la parte central de la pantalla se mostrara la imagen transmitida por la cámara con que cuenta el robot.

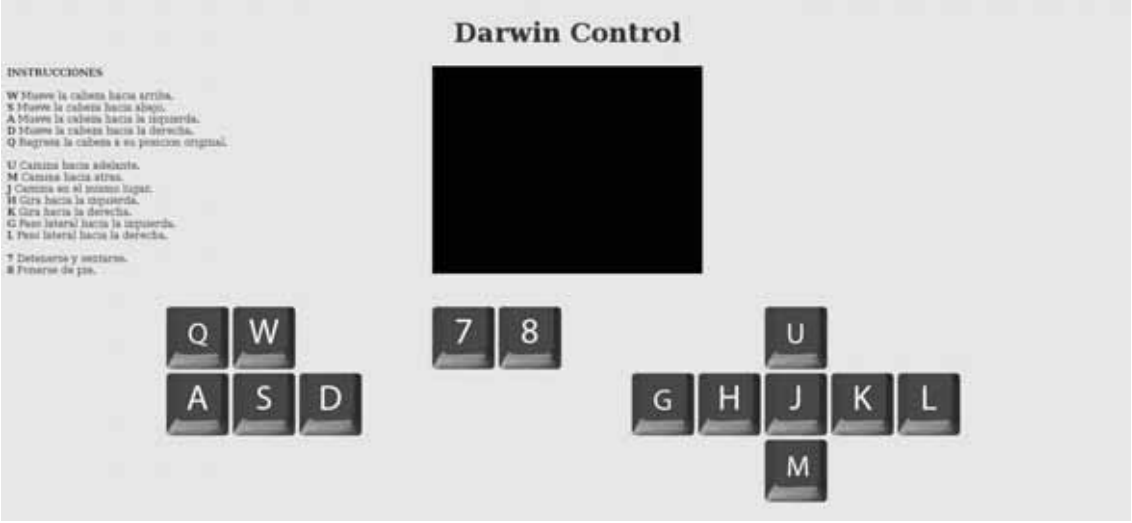

Figura 12: Imagen de página web del control.

6.4.2. Interfaz de control

Al implementar el módulo de interfaz de control se optó por hacer un manejo de una función en PHP que permitiera detectar la tecla presionada en la interfaz de usuario, de esta manera se envía el parámetro directamente al programa que controla al robot para ejecutar la instrucción correspondiente al parámetro enviado.

El siguiente fragmento de código corresponde a la interfaz de control que es quien se encuentra en espera de que el usuario presione alguna tecla para posteriormente enviarla como parámetro al programa control de Darwin.

```
<?php
  str = KeyDown(ev);
  exec('/darwin/darwin/upenn_humanoid_1.1/Player/a_control.lua str');
?>
```

[6]Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

6.4.3. Control del robot

Para realizar el módulo de control del robot se utilizó una API[7] creada por la Universidad de Pensilvania en el lenguaje de programación LUA[8], con esto se logro una programa interpretado mucho mejor para realizar un manejo ágil e intuitivo mejorando así el desempeño del control de Darwin.

El programa que controla el movimiento del robot se centra básicamente en el ciclo que se muestra a continuación, este se encuentra en espera de un parámetro que será enviado primero por la interfaz de usuario y procesado por la interfaz de control.

```
function update()
  Motion.update();
  local str=getch.get();
  if #str>0 then
    local byte=string.byte(str,1);
    if byte==string.byte("u") then targetvel[1]=targetvel[1]+0.01; -- camina al frente
    elseif byte==string.byte("h") then targetvel[3]=targetvel[3]+0.1; -- gira izquierda
    elseif byte==string.byte("j") then targetvel[1],targetvel[2],targetvel[3]=0,0,0; -- marcha en su lugar
    elseif byte==string.byte("k") then targetvel[3]=targetvel[3]-0.1; -- gira a la derecha
    elseif byte==string.byte("m") then targetvel[1]=targetvel[1]-0.01; -- camina para atras
    elseif byte==string.byte("g") then targetvel[2]=targetvel[2]+0.01; -- paso lateral izquierda
    elseif byte==string.byte("l") then targetvel[2]=targetvel[2]-0.01; -- paso lateral derecha
    elseif byte==string.byte("7") then Motion.event("sit"); b.set_head_hardness({.5,.5}); -- sentarse
  elseif byte==string.byte("8") then walk.stop(); Motion.event("standup"); -- pararse
    elseif byte==string.byte("w") then bpos[2]=bpos[2]-0.1; -- cabeza arriba
  elseif byte==string.byte("s") then bpos[2]=bpos[2]+0.1; -- cabeza abajo
  elseif byte==string.byte("a") then bpos[1]=bpos[1]-0.1; -- cabeza izquierda
  elseif byte==string.byte("d") then bpos[1]=bpos[1]+0.1; -- cabeza derecha
  elseif byte==string.byte("q") then bpos={0,0}; -- posicion orgunal
    end
    print(string.format("\n Walk Velocity: (%.2f, %.2f, %.2f)\n",unpack(targetvel)));
    walk.set_velocity(unpack(targetvel));
    b.set_head_command(bpos);
  end
end
```

[7]Del inglés Application Programming Interface es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

[8]Es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como un lenguaje interpretado con una semántica extendible.

7. Resultados

7.1. Experimento de transmisión continua de video

AL finalizar el proyecto se pudo obtener una transmisión continua del video recibido desde la cámara con que cuenta el robot humanoide Darwin-OP. El servidor de video que se implementó para transmitir, no interrumpe en ningún momento la transmisión de imagen hacia la interfaz de usuario a menos de que se de por terminado el sistema de monitoreo y manipulación del robot humanoide.

La figura 13 muestra una captura de pantalla de la interfaz de usuario en ejecución, como se puede apreciar, está transmitiendo un video en tiempo real. Para que se pueda entender esta acción, se oriento la cámara de Darwin hacia la computadora que esta recibiendo la transmisión de video, con esto se puede ver que está captando al usuario viendo su propia pantalla de video.

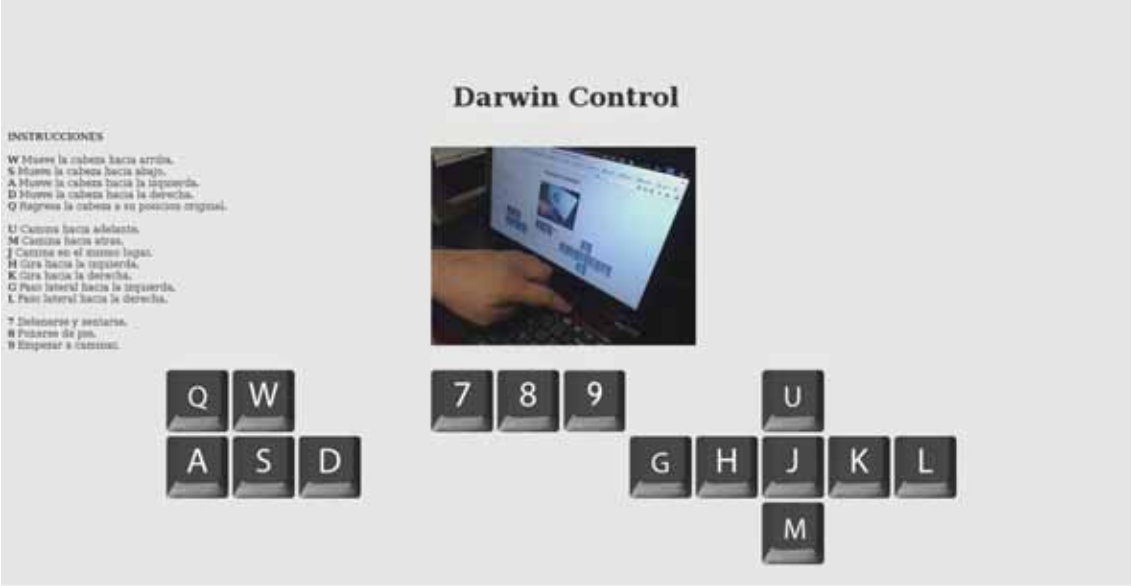

Figura 13: Prueba de visión.

7.2. Experimento de movimientos del cuerpor de Darwin-OP

7.2.1. Movimiento del cuerpo

Al término de este proyecto se consiguió que Darwin hiciera diversas caminatas en un circuito de pruebas representado en la figura 14.

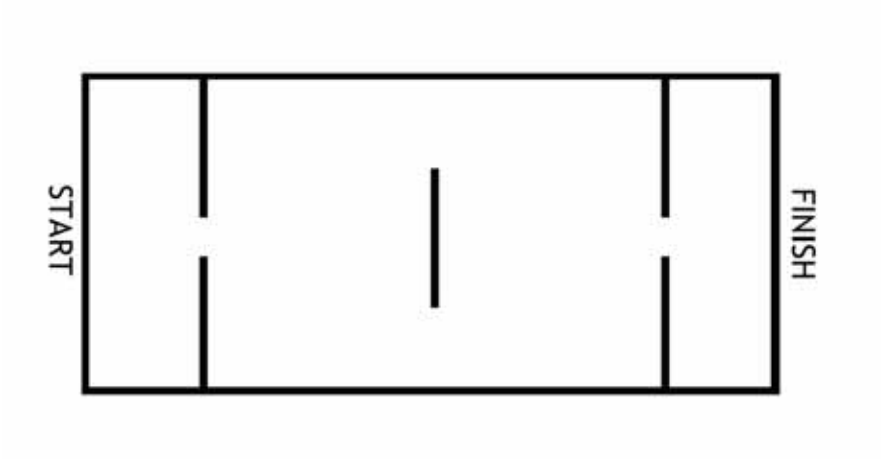

Figura 14: Circuito de pruebas.

- En un primer experimento se controló al robot para que hiciera una caminata en linea recta del punto A al punto B, sin obstáculos 15. El usuario podía ver de manera directa al robot. El recorrido se completo de manera exitosa.

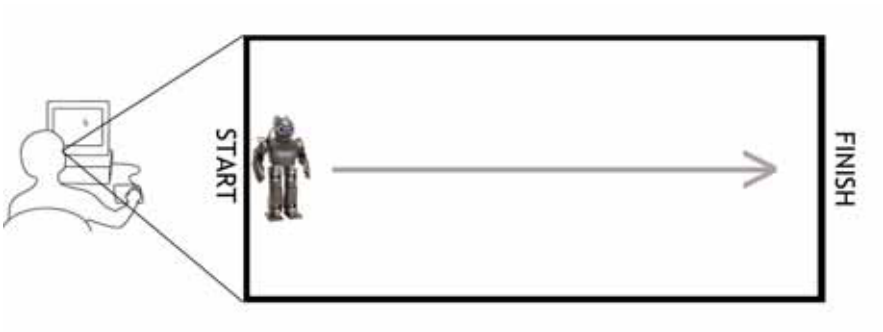

Figura 15: Caminata sin obstáculos, usuario presente.

- Se repitió el experimento anterior, pero con la diferencia de que el usuario ahora no podía ver de manera directa al robot 16, ahora solamente se podía guiar por la transmisión de video del robot. El recorrido se completó de manera satisfactoria.

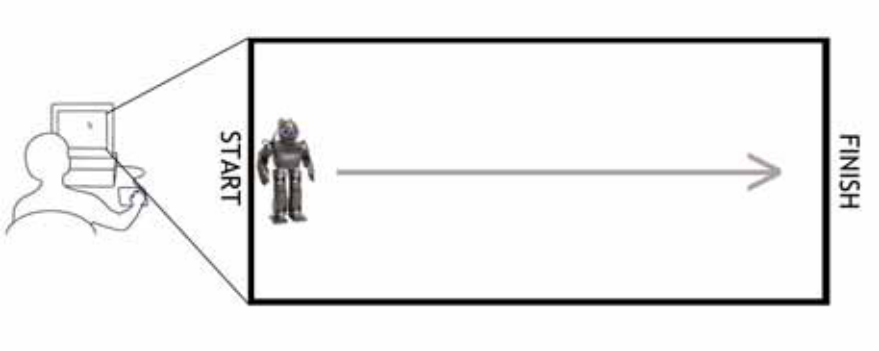

Figura 16: Caminata sin obstáculos, usuario no presente.

- Para el segundo experimento se le agregaron obstáculos que impidieran el andar recto del robot, esto para forzar al usuario a tener que hacer giros en la trayectoria. Para este segundo experimento el usuario podia ver al robot de manera directa y poder guiarlo a su destino a través del circuito 17. El recorrido de completo de manera exitosa.

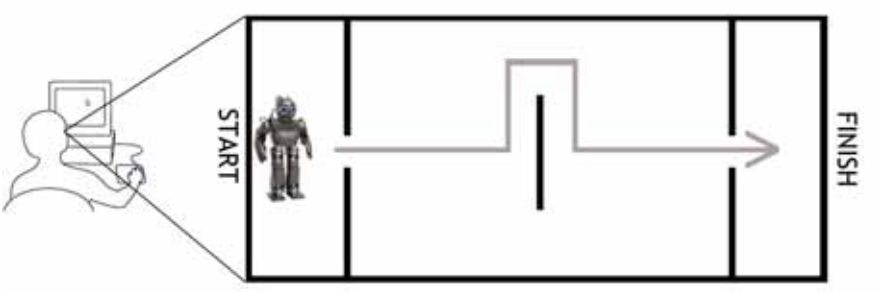

Figura 17: Caminata con obstáculos, usuario presente.

- Se repitió el experimento anterior, pero con la diferencia de que el usuario ahora no podía ver de manera directa al robot 18, ahora solamente se podía guiar por la transmisión de video del robot. El recorrido se completó de manera satisfactoria.

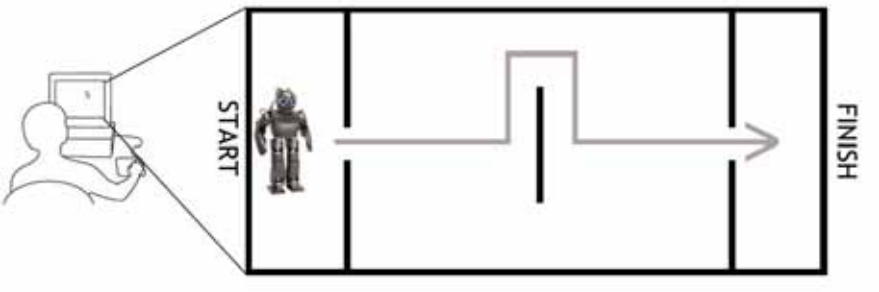

Figura 18: Caminata con obstáculos, usuario no presente.

7.2.2. Movimiento de la cabeza

Para la experimentación con el movimiento de la cabeza se realizaron pruebas en las que se movía la cabeza del robot y se corroboraba que no hubiera interrupción de la trasmisión de video al igual que no se interrumpiera el movimiento del resto del cuerpo.

- Transmisión continua de video mientras se mueve la cabeza.
- Caminar mientras se mueve la cabeza.
- Las dos anteriores.

Las tres pruebas anteriores fueron realizadas exitosamente.

8. Análisis y discusión de resultados

Las pruebas realizadas para cada uno de los experimentos fueron hechas con base a las especificaciones requeridas para la aprobación de término de este proyecto. Todas las pruebas fueron realizadas satisfactoriamente en la mesa de experimentación, así como en un circuito de pruebas hecho para este fin especifico.

Algunas cuestiones que deben de ser tomadas en cuenta para futuros proyectos, ya sea de continuación o de nuevas implementaciones, son presentadas a continuación:

- El robot cuenta con algunos bloqueos de seguridad, los cuales no le permiten ser a Darwin quien emita la señal Wi-Fi para su conexión, es necesario utilizar una red inalámbrica preferentemente ajena a la red provista por la universidad, ya que hace lenta la comunicación entre la interfaz de usuario y el robot.
- De igual manera, estos bloqueos de seguridad tienen código que permiten a Darwin detectar si se ha caído de frente o de espaldas, posterior a la detección de caída el robot se incorporará de manera autónoma.
- Cada una de las pruebas fueron repetidas al menos 30 veces cada una, para detectar algún fallo en el sistema.
- Cada prueba tenia una duración aproximada de 10 minutos desde el momento en que se iniciaba el sistema y el fin de la prueba. Esto equivale a un aproximado de 20 horas de pruebas.
- Dada las condiciones del robot, la mitad de las pruebas se hicieron con Darwin conectado a la corriente de manera directa y la otra mitad se hicieron con las baterías que proporciona el fabricante. Esto demuestra que el sistema es capaz de trabajar tanto con el robot conectado a la corriente como con baterías.
- Todas las pruebas fueron realizadas con una superficie plana y lisa, en el área provista por el asesor para trabajar en este proyecto.
- Algunas de las primeras pruebas resultaban con fallos en la manipulación del robot debido a un problema de calentamiento del robot. Una vez detectado este problema se optó por hacer sesiones de descanso para el robot por cada hora de pruebas. Esto evito el calentamiento del robot así como fallos en el sistema.

9. Conclusiones

El trabajo de implementar el sistema de monitoreo y manipulación de un robot humanoide ha cumplido con el objetivo principal de este proyecto. Las pruebas realizadas han sido satisfactorias y concuerdan con lo estipulado en la propuesta de proyecto de integración.

Las pruebas realizadas con respecto a la visualización de la cámara integrada en el robot tuvieron un resultado de acuerdo a lo esperado, se logró una transmisión continua de video con la cual la interfaz de usuario es capaz de recibir dicha imagen de manera continua. Cabe aclara que la fluidez de la transmisión depende en gran medida a la red con que se cuente para hacer la conexión. En ocasiones el video puede no ser tan fluida como se desearía, esto no quiere decir que este mal codificado el servidor de video o la interfaz de usuario, esto responde más bien, como se mencionó antes, a la red dedicada a este servicio.

Ahora hablando del control del robot se utilizó una API creada por la Universidad de Pensilvania, la cual cuenta con múltiples librerías para el movimiento del cuerpo de Darwin. La implementación de este modulo se realizó de manera más amigable debido a que la API mencionada está hecha con el lenguaje de programación LUA, que es un lenguaje de tipo script. Teniendo en cuenta lo anterior, se pudo crear el programa para controlar al robot de tal manera que se ajustara a las necesidades del sistema, sin el engorroso trabajo de recompilar cada vez que se hiciera un ajuste al programa. Con esto se gano tiempo valioso para llevar a cabo las pruebas pertinentes para controlar el caminar y giros de Darwin.

La cuestión del caminado fue en particular un reto. Se debieron de tener en cuentas muchas métricas para lograr coordinar todos los servomotores con que cuentan las piernas. Una vez simulado el movimiento natural de una pierna al caminar se tenían que hacer pruebas de equilibrio, como es lógico pensar, si el robot no mantenía un eje de equilibrio inevitablemente se iba al suelo. Este trabajo representó un completo estudio de mecánica, electrónica y por supuesto computación, del mismo modo que se tuvieron que tomar en cuenta cuestiones biológicas tales como la locomoción bípeda.

Otro aspecto a considerar es que Darwin no esta actuando de manera autónoma, depende casi en su totalidad del usuario, debido a la naturaleza de este proyecto que buscaba controlar a un robot para explorar lugares a los que el usuario lo guiara. Por esta razón se debe de tener siempre en consideración las posturas que adquiere el robot, cuando se inicia el sistema el robot parte de estar en cuclillas y se pone de pie. También cuando se quiere regresar a la posición inicial (en cuclillas) el usuario primero tiene que detener a Darwin para que quede sobre sus dos pies de manera casi erguida, para posteriormente poder pedir que se ponga de cuclillas, de lo contrario se corre el riesgo de que Darwin no se encuentre con sus dos piernas bien situadas sobre el piso y en el momento que se le pida regresar a la posición inicial provoque una caída del robot.

Siguiendo este camino, aún hay bastante trabajo por realizar para futuros proyectos. Se puede ampliar todavía más este proyecto, tal vez haciendo que el robot ya no solo explore con ayuda de su andar, sino que también recolecte objetos gracias a sus brazos que son capaces de sujetar cosas y cargarlas. La robótica es un campo relativamente joven, así que aún hay mucho que hacer e investigar. Solo se necesita ver todas las investigaciones que se están llevando a cabo a nivel mundial, no solo por universidades, sino por grandes empresas que ven un futuro prometedor en esta área. Los robots serán y son de gran ayuda para los humanos, tanto en la industria realizando trabajos pesados o riesgosos para las personas, como también en la vida cotidiana asistiendo a personas mayores o con capacidades diferente.

Referencias

[1] Robotis, [Web en línea]. <>. `support.robotis.com/en/techsupport_eng.htm#product/darwin-op.htm`. [Consulta 10/07/2014]

[2] Pennalizers, [Web en línea]. <>. `https://fling.seas.upenn.edu/~robocup/wiki/index.php?n=main.code`. [Consulta 10/07/2014]

[3] Penn Enginnering, [Web en línea]. <>. `http://www.seas.upenn.edu/~robocup/files/DARwIn-OP_UPenn_Tutorial.pdf`. [Consulta 10/07/2014]

[4] A. Ollero Baturone, *Robótica: manipulación y robots móviles*, Marcombo, España, 2001.

[5] John. J. Craig, *Robótica*, Pearson Educación, 3a Edición, 2006.

[6] C. Aubry, *HTML5 y CSS3 - Revolucione el diseño de sus sitios web*, Ediciones ENI, 2012.

[7] L. Van Lancker, *HTML5 y CSS3: Domine los estándares de las aplicaciones Web*, Ediciones ENI, 2a. Edición, 2013.

[8] J. Diego Gauchat, *El gran libro de HTML5, CSS3 y Javascript*, Marcombo, 2012.

[9] M. Angel Sánchez Maza, *Javascript*, Innovación Y Cualificación, 2012.

[10] E. Gutierrez, *JavaScript: Conceptos básicos y avanzados (bibliotecas Prototype y Script.aculo.us)*, Ediciones ENI, 2009.

[11] O. Heurtel, *PHP 5.3: Desarrollar un sitio Web dinámico e interactivo*, Ediciones ENI, 2011.

[12] K. Tatroe, P. Maclntyre and R. Lerdorf, *Programming PHP*, O'Reilly Media, Inc, 2013.

[13] K. Jung, A. Brown, *Beginning Lua Programming*, John Wiley & Sons, 2011.

[14] R. Ierusalimschy, *Programming in Lua*, LUA.org, 2a Edición, 2006.

[15] Andrew G. Curioso, *Ajax with PHP 5*, O'Reilly Media, Inc, 2007.

10. Código fuente

A continuación se muestra el código fuente de las partes que componen el sistema de monitoreo y manipulación del robot humanoide Darwin-OP.

10.1. Página web del sistema de monitoreo y manipulación

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <link type="text/css" rel="stylesheet" href="stylesheet.css"/>
    <title>Darwin Control</title>
    <style type="text/css">
      /* tab init */
      .tabcontent {
        display:none;
      }
      #control {
        font-size: 12px;
      }
    </style>
    <script language="javascript" type="text/javascript"
      src="/functions.js"></script>
    <script language="javascript">
      var pretab = "";
      var pretd = "";

      function extendmenu(mid, aobject) {
        var tab_name = "sc"+mid;
        var td_name = "td"+mid;
        if(pretab != "") {
          document.getElementById(pretab).style.display = "none"
          document.getElementById(pretd).style.backgroundColor="#FFFFFF";
          document.getElementById(pretd).style.fontWeight="normal";
          document.getElementById(pretd).style.color="#000000";
        }
        document.getElementById(tab_name).style.display = "block"
        document.getElementById(td_name).style.backgroundColor="navy";
        document.getElementById(td_name).style.fontWeight="bold";
        document.getElementById(td_name).style.color="#FFFFFF";
        pretab = tab_name;
        pretd = td_name;
      }

          function AJAX_response(text) {
```

```
            document.getElementById('hints').firstChild.nodeValue = "Got
                response: " + text;
        }

        function KeyDown(ev) { // Funcion para visualizar la tecla
            presionada
          ev = ev || window.event;
          pressed = ev.which || ev.keyCode;
          switch (pressed) {
            case 87: /*send_command();*/
                document.getElementById('w').src="boton/wp.png"; break;
            case 81: /*send_command();*/
                document.getElementById('q').src="boton/qp.png"; break;
            case 65: /*send_command();*/
                document.getElementById('a').src="boton/ap.png"; break;
            case 83: /*send_command();*/
                document.getElementById('s').src="boton/sp.png"; break;
            case 68: /*send_command();*/
                document.getElementById('d').src="boton/dp.png"; break;
            case 85: /*send_command();*/
                document.getElementById('u').src="boton/up.png"; break;
            case 71: /*send_command();*/
                document.getElementById('g').src="boton/gp.png"; break;
            case 72: /*send_command();*/
                document.getElementById('h').src="boton/hp.png"; break;
            case 74: /*send_command();*/
                document.getElementById('j').src="boton/jp.png"; break;
            case 75: /*send_command();*/
                document.getElementById('k').src="boton/kp.png"; break;
            case 76: /*send_command();*/
                document.getElementById('l').src="boton/lp.png"; break;
            case 55: /*send_command();*/
                document.getElementById('7').src="boton/7p.png"; break;
            case 56: /*send_command();*/
                document.getElementById('8').src="boton/8p.png"; break;
            case 77: /*send_command();*/
                document.getElementById('m').src="boton/mp.png"; break;
                   default: break;
              }
            }

        document.onkeydown = KeyDown;

      </script>

    </head>

    <script type="text/javascript">

    var imageNr = 0; // Número de serie de la imagen actual
```

```
var finished = new Array(); // Las referencias a objetos img que se
    han terminado de descargar
var paused = false;

function createImageLayer() {
    var img = new Image();
    img.style.position = "absolute";
    img.style.zIndex = -1;
    img.onload = imageOnload;
    img.onclick = imageOnclick;
    img.src = "/?action=snapshot&n=" + (++imageNr);
    var webcam = document.getElementById("webcam");
    webcam.insertBefore(img, webcam.firstChild);
}

// Dos layers siempre están presentes (excepto en el comienzo),
    para evitar el parpadeo
function imageOnload() {
    this.style.zIndex = imageNr; // La imagen terminada, se trae al
        frente
    while (1 < finished.length) {
        var del = finished.shift(); // Se elimina imagen antigua
            del documento
        del.parentNode.removeChild(del);
    }
    finished.push(this);
    if (!paused)
    createImageLayer();
}

function imageOnclick() { // Un click en la imagen pausara el strem
    paused = !paused;
    if (!paused)
    createImageLayer();
}

</script>
<body onload="createImageLayer();extendmenu('1', this);">
  <?php
    str = KeyDown(ev);
    exec(/darwin/darwin/upenn_humanoid_1.1/Player/a_control.lua
        str);
  ?>

  <!-- Titulo de la página -->
  <h1>Darwin Control</h1>

  <!-- Imagen de video -->
  <div id="webcam"><noscript><img src="/?action=snapshot"
      /></noscript></div>
```

```
<!-- Set de instrucciones -->
<p>
  <b>INSTRUCCIONES</b><br><br>
  <b>W </b>Mueve la cabeza hacia arriba.<br>
  <b>S </b>Mueve la cabeza hacia abajo.<br>
  <b>A </b>Mueve la cabeza hacia la izquierda.<br>
  <b>D </b>Mueve la cabeza hacia la derecha.<br>
  <b>Q </b>Regresa la cabeza a su posicion original.<br><br>
  <b>U </b>Camina hacia adelante.<br>
  <b>M </b>Camina hacia atras.<br>
  <b>J </b>Camina en el mismo lugar.<br>
  <b>H </b>Gira hacia la izquierda.<br>
  <b>K </b>Gira hacia la derecha.<br>
  <b>G </b>Paso lateral hacia la izquierda.<br>
  <b>L </b>Paso lateral hacia la derecha.<br><br>
  <b>7 </b>Detenerse y sentarse.<br>
  <b>8 </b>Ponerse de pie.<br>
</p>

<div class="stream"></div>

<input type="image" src="boton/q.png" id="q" class="boton00"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/w.png" id="w" class="boton01"
    onkeydown="KeyDown(ev);"/>

<input type="image" src="boton/a.png" id="a" class="boton02"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/s.png" id="s" class="boton03"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/d.png" id="d" class="boton04"
    onkeydown="KeyDown(ev);"/>

<input type="image" src="boton/u.png" id="u" class="boton05"
    onkeydown="KeyDown(ev);"/>

<input type="image" src="boton/g.png" id="g" class="boton06"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/h.png" id="h" class="boton07"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/j.png" id="j" class="boton08"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/k.png" id="k" class="boton09"
    onkeydown="KeyDown(ev);"/>
<input type="image" src="boton/l.png" id="l" class="boton10"
    onkeydown="KeyDown(ev);"/>

<input type="image" src="boton/m.png" id="m" class="boton11"
    onkeydown="KeyDown(ev);"/>
```

```
            <input type="image" src="boton/7.png" id="7" class="boton12"
                onkeydown="KeyDown(ev);"/>
            <input type="image" src="boton/8.png" id="8" class="boton13"
                onkeydown="KeyDown(ev);"/>

        </body>
    </html>
```

10.2. Control del robot

```
---------------------------------
-- Darwin Control
---------------------------------

module(... or "", package.seeall)
require('unix')
webots = false;
darwin = true;
local cwd = unix.getcwd();
-- the webots sim is run from the WebotsController dir (not Player)
if string.find(cwd, "WebotsController") then
  webots = true;
  cwd = cwd.."/Player"
  package.path = cwd.."/?.lua;"..package.path;
end

computer = os.getenv('COMPUTER') or "";
if (string.find(computer, "Darwin")) then
   -- MacOS X uses .dylib:
   package.cpath = cwd.."/Lib/?.dylib;"..package.cpath;
else
   package.cpath = cwd.."/Lib/?.so;"..package.cpath;
end

package.path = cwd.."/Util/?.lua;"..package.path;
package.path = cwd.."/Config/?.lua;"..package.path;
package.path = cwd.."/Lib/?.lua;"..package.path;
package.path = cwd.."/Dev/?.lua;"..package.path;
package.path = cwd.."/Motion/?.lua;"..package.path;

require('Config');
require('Body')
require('vector')
require('Motion')
require('walk')
require("getch")
require('kick')
b=require 'DarwinOPBody'

Motion.entry();
--Motion.event("standup");

-- condiciones iniciales para el robot
getch.enableblock(1);
targetvel=vector.new({0,0,0});
bpos=vector.new({0,0});
b.set_head_hardness({.5,.5});
b.set_head_command(bpos);
```

```
print("␣Key␣commands␣\n␣7:␣sit␣down␣8:␣stand␣up\n␣i/j/l/,/h/;␣:␣control␣
    walk␣velocity\n␣k␣:␣walk␣in␣place\n␣1/2␣:␣kick");

-- ciclo principal
function update()
  Motion.update();
  local str=getch.get();
  if #str>0 then
    local byte=string.byte(str,1);
    if byte==string.byte("u") then targetvel[1]=targetvel[1]+0.01; --
       camina al frente
    elseif byte==string.byte("h") then targetvel[3]=targetvel[3]+0.1;
       -- gira izquierda
    elseif byte==string.byte("j") then
       targetvel[1],targetvel[2],targetvel[3]=0,0,0; -- marcha en su
       lugar
    elseif byte==string.byte("k") then targetvel[3]=targetvel[3]-0.1;
       -- gira a la derecha
    elseif byte==string.byte("m") then
       targetvel[1]=targetvel[1]-0.01; -- camina para atras
    elseif byte==string.byte("g") then
       targetvel[2]=targetvel[2]+0.01; -- paso lateral izquierda
    elseif byte==string.byte("l") then
       targetvel[2]=targetvel[2]-0.01; -- paso lateral derecha
    elseif byte==string.byte("7") then Motion.event("sit");
       b.set_head_hardness({.5,.5}); -- sentarse
  elseif byte==string.byte("8") then walk.stop();
     Motion.event("standup"); -- pararse
    elseif byte==string.byte("w") then bpos[2]=bpos[2]-0.1; -- cabeza
       arriba
  elseif byte==string.byte("s") then bpos[2]=bpos[2]+0.1; -- cabeza
     abajo
  elseif byte==string.byte("a") then bpos[1]=bpos[1]-0.1; -- cabeza
     izquierda
  elseif byte==string.byte("d") then bpos[1]=bpos[1]+0.1; -- cabeza
     derecha
  elseif byte==string.byte("q") then bpos={0,0}; -- posicion origunal
    end
    print(string.format("\n␣Walk␣Velocity:␣(%.2f,␣%.2f,␣
       %.2f)\n",unpack(targetvel)));
    walk.set_velocity(unpack(targetvel));
    b.set_head_command(bpos);
  end
end

while 1 do
 update()
end
```