aHuman-Droid Prototype: Primeros pasos en robótica bípeda
Workshop del CAFR2005

Lezama, Damián Sklar, Alexander Tejera, Gonzalo

Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Resumen: En este artículo se presenta la construcción de un prototipo de robot bípedo capaz de caminar cuasiestáticamente. Entre los aportes originales se encuentra la estrategia para caminar del prototipo, que utiliza un enfoque análogo a la técnica de stop-motion utilizada en animación, la lectura de la posición de los motores en lazo semiabierto, y una técnica de coordinación de motores llamada Clamping (encepado). El Proyecto incluye elementos de electrónica, mecánica, programación a bajo nivel y embebida, programación a alto nivel y programación de interfaces gráficas, control, y tratamiento de señales..

Palabras clave: Robótica, Bípedos, Construcción de Robots, Sistemas de tiempo real, Programación embebida.

1 INTRODUCCIÓN

En este artículo se presenta la construcción de un prototipo de robot bípedo - denominado Human-Droid Protoype (HDP) - capaz de caminar cuasiestáticamente (a bajas velocidades). Se logró desarrollar una estrategia de caminata innovadora utilizando servomotores de hobby, aluminio, técnicas de construcción casera y elementos de bajo costo para obtener un desarrollo en extremo económico. Además, en este trabajo se desarrolló una placa de control para robótica móvil, y se propone una arquitectura extensible genérica para robótica, parte de la cual fue implementada para el prototipo.

2 MORFOLOGÍA

Al momento de construir un robot bípedo, hay que decidir cuál será la configuración de las articulaciones, así como la cantidad de articulaciones y segmentos. Al ser cada pie un efector del sistema, tenemos dos efectores. Para especificar cada efector se necesita un punto en el espacio y sus ángulos directores, es decir seis parámetros. Esto da un total de doce parámetros para describir los efectores del sistema. En este caso se apuntaba a que el sistema cinemático fuera compatible determinado, por lo que se eligió tener doce articulaciones, seis por pierna. Cada pierna cuenta con un tobillo, una rodilla y una cadera, así como con un muslo y una pantorrilla. El

tobillo cuenta con dos grados de libertad (DOF): el primero permite mover el pie hacia arriba y hacia abajo, mientras que el segundo es el que permite inclinar hacia la izquierda y la derecha. La rodilla cuenta con un único grado de libertad, que es el que le permite doblarse. Por último la cadera posee tres grados de libertad, uno en cada eje.

2.1. El cuerpo humano como un sistema redundante

Si bien con doce DOF se resuelve el posicionamiento de los efectores, el cuerpo humano posee más grados de libertad (es decir, es redundante), y la solución al problema de la cinemática inversa [1] a veces no es única: una persona sentada con los pies fijos en el piso, puede mover la rodilla hacia los costados. El efector está fijo, sin embargo hay un grado de libertad redundante. Esto le permite al cuerpo humano efectuar movimientos que en el HDP no son posibles, pero que no son imprescindibles para caminar.

2.2 Tipos de articulaciones e implementaciones

Las articulaciones en el reino animal son de tres tipos: "bola y cavidad", rotativa, y charnela. El primer tipo permite movimiento en los tres ejes (hombro, cadera), el segundo en dos ejes (articulación radio-cubital) y el último en un solo eje (dedos, codos, rodillas, articulación temporo-maxilar). Ya que usamos motores para implementar las articulaciones, fue necesario pensar en una forma de realizar articulaciones de varios grados de libertad. En el HDP, los ejes de los DOF correspondientes a articulaciones de más de un DOF, se intersectan en un punto imaginario que sería el centro de la articulación "bola y cavidad" o de la rotatoria. Esto es una gran ventaja sobre las implementaciones de articulaciones de múltiples DOF que utilizan otros robots [2][3] (es decir, una cadena de segmentos unidos por las articulaciones ortogonales), ya que el giro de uno de los DOF en nuestro caso no afecta la posición relativa a los demás DOF, cosa que sí sucede en las otras implementaciones.

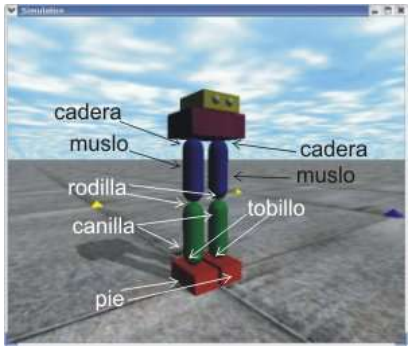

Figura 1. Esquema mecánico del prototipo.

3 ESTRUCTURA DE LA SOLUCIÓN

En la solución propuesta, la lógica se encuentra distribuida entre dos nodos. El primero es un nodo que controla la lógica de bajo nivel (mover los actuadores), y el segundo la lógica de alto nivel (dar un paso, etc.). Esto presenta varias ventajas: en primer lugar desacopla los dos niveles de lógica, lo que provee una gran flexibilidad. Por otro lado, promueve el enfoque en capas, lo que mejora la mantenibilidad y claridad de la arquitectura del sistema.

El nodo PIC es el encargado del sistema motriz de bajo nivel del robot, es decir que contiene la inteligencia necesaria para que el

cada articulación pueda moverse, aunque sea sin coordinación. Algunos animales presentan la misma división del sistema nervioso: por ejemplo, los gatos y otros mamíferos tienen una parte de su sistema motriz integrado en la columna vertebral, mientras que el resto de su inteligencia está concentrada en el encéfalo [4]. El sistema motriz de mayor nivel es el que impone la coordinación entre las articulaciones para lograr movimientos coherentes del robot. El mismo está implementado en un nodo distinto, llamado nodo PC. El nodo PC está compuesto por un PC y el software de control del robot. Este software es el responsable de la lógica motriz de mayor nivel: saber como mover el conjunto de articulaciones disponibles, coordinar los movimientos, y también recibir, interpretar y reaccionar ante estímulos, sean estos originados en el robot o en el ambiente.

Desde el punto de vista del flujo de información a nivel lógico, el nodo PC envía señales de comando (mover el motor i-ésimo a la posición i con velocidad i, encender o apagar una salida lógica, etc.), y recibe la medición de las posiciones de los motores. Además de esta información, se notifica si cada motor ha llegado a su destino, y se agrega un bit de reconocimiento del último paquete enviado (ACK).

4 ARQUITECTURA PROPUESTA

El sistema se estructura en diferentes capas, y se utiliza el patrón Observer para la notificación de eventos asincrónicos. Además, se utiliza el patrón Abstract Factory para obtener los manejadores de distintas capas. Dicho Factory es configurable a través de un archivo de configuración XML. El uso de interfaces permite reemplazar componentes existentes por otros con iguales funcionalidades. A continuación se describen las funcionalidades de las diferentes capas de la arquitectura.

4.1 pic

En este componente se realiza el control motríz a bajo nivel (generación de las ondas PWM para comandar a los servomotores, ramping, lectura de sensores, etc.)

4.2 comm

Este componente es el encargado de implementar la comunicación a nivel de flujo de bytes con el PIC. En este sentido, debe poder enviar bytes hacia el PIC y recibir bytes desde el PIC. Los servicios ofrecidos por esta capa son el envío de un arreglo de bytes del PC al PIC, algunos servicios de configuración y estadística del puerto serial, y una operación para registrarse para recibir notificaciones de bytes entrantes (paquetes desde el PIC hacia el PC).

4.3 motor

Este componente es el que conoce el formato de los paquetes que se envían desde el PIC al PC y viceversa. Además, permite esperar a que un motor determinado llegue a la posición a la que debe llegar, o que todos los motores lleguen. Cabe notar que el término "llegue a la posición" significa que el PIC envíe el bit de Reached encendido, en oposición a que el motor realmente se encuentre en la posición que deba estar. Esta distinción implica que cada motor puede seguir las órdenes del PIC sin atrasarse (de la misma manera, que las velocidades enviadas por el PIC son lo suficientemente bajas).

4.4 stance

La primera responsabilidad de este componente es la implementación de un diccionario de poses del robot. Una pose (stance) es una configuración en el espacio de valores de posición de los motores, es decir un vector de 12 elementos, y se identifica por un nombre. La segunda responsabilidad es proveer una interfaz que permita comandar al robot hacia una posición con una cierta velocidad para cada articulación. A este efecto utiliza los servicios de la capa motora.

4.5 gait

Este componente es el encargado de implementar un diccionario de andares del robot. Un andar (gait) es una máquina de estados de poses, y se identifica por un nombre. Además ofrece el servicio de realizar un andar, para lo que utiliza las operaciones de la capa stance.

4.6 walk

Este componente provee una interfaz que permite comandar al robot a que realice una caminata, concatenando servicios de la misma. Esta capa se comporta de manera similar a las

ordenes disponibles en el lenguaje LOGO. Los servicios que provee son dar un paso hacia adelante, atras, izquierda o derecha y rotar hacia la izquierda o derecha. Para ello, podrá utilizar la lectura de sensores u otra información para poder generar andares en tiempo real.

4.7 pathPlanning

Este componente es el encargado de, dado una ubicación adonde se desea llegar y sabiendo la ubicación actual, calcular el camino que debe seguir el robot. El componente debe permitir especificar ciertas restricciones, cuya especie a principio es desconocida, pero que son del estilo "no pasar por un punto dado", o "pasar por un punto dado".

4.8 biped

Este componente es el componente lógico de mas alto nivel, y es el que ofrece el servicio de desplazamiento. Dentro del mismo, se envían los pedidos de cálculo de trayectorias a la capa inferior, y los mismos se retroalimentan mediante las notificaciones que se reciben desde el PIC, a través de las lecturas de sensores analógicos, que pueden ser de proximidad, temperatura, etc.

4.9 ui

Este componente implementa la lógica de interfaz de usuario que permite comandar al robot. Está constituido por clases que implementan tres tipos de lógica: por un lado, la interfaz gráfica de los casos de uso. Por otro, clases auxiliares que implementan controles que se utilizan en la UI.

5 ELEMENTOS DE PROCESAMIENTO

Como se comentó en la descripción de la Arquitectura, por un lado se tiene el robot construido, con un elemento de procesamiento que lo controla, específicamente un microcontrolador. El mismo se encuentra dentro de una plaqueta de control embebido, y está ejecutando continuamente el software de control embebido. Este conjunto de elementos (robot, plaqueta controladora y software embebido) compone el nodo PIC.

5.1 Nodo PIC

Se decidió desarrollar una placa controladora propia en lugar de utilizar una comercial. Esta decisión estuvo respaldada por dos razones. La primera razón es que los controladores comerciales no contaban con las prestaciones necesarias, ya que no proveían una resolución de velocidades adecuada, ni una cantidad de conversores A/D suficiente. Lo primero es necesario porque es importante la correcta coordinación de los motores para poder llegar a caminar. Si bien las placas existentes en el mercado proveían ocho bits de resolución de velocidad, no se aseguraba que los 256 niveles resultantes fueran utilizables (es decir, que a la máxima velocidad admisible, el servomotor coincidiera con el nivel más alto de velocidad). Esto significa que para realizar movimientos más lentos, podríamos quedar limitados a unos pocos niveles de velocidad, lo que perjudicaría la coordinación. Lo segundo es importante debido a que queremos poder leer las posiciones de los motores y en lo posible otros dispositivos sensores para poder tomar

decisiones. Esta característica permite capturar (digitalizar) la pose en la que se encuentra actualmente el robot.

La segunda razón para crear nuestro propio controlador es que las placas representan un costo bastante alto, aumentado por el hecho de que no se consiguen en el mercado local. Adicionalmente el controlador presenta un valor agregado al proyecto, ya que el mismo puede ser utilizado en otros proyectos de robótica.

Debido a que se requería control embebido a nivel de tiempo real, y se precisaba una gran cantidad de puertos de E/S, se optó por utilizar un microcontrolador. Para implementar el control de los motores se eligió utilizar un microcontrolador de la línea PIC de Microchip. Los requerimientos que determinaron la elección se resumen a continuación:

- Velocidad de clock suficientemente alta como para poder cumplir con requerimientos de tiempo real.
- 12 generadores de onda PWM o bien suficientes salidas digitales que permitan generar las ondas. Esto es necesario para comandar los motores.
- 12 entradas analógicas (a través de conversores A/D). Esto es necesario para poder leer la posición de los motores.
- Empaquetado DIP para simplificar el montaje en Protoboards y circuitos impresos.
- Comunicación hacia afuera según algún protocolo estándar de forma simple.
- Varios timers, ya que se trata de un sistema de tiempo real que debe poder interactuar con el mundo exterior (comunicarse, manejar motores, etc.)

- Es deseable que la programación del PIC se realice sin tener que desmontar el PIC del protoboard a través del puerto serial. Esta modalidad se denomina ICSP (In-Circuit Serial Programming)
- Es deseable la posibilidad de expansión mediante la conexión de más de un PIC en cascada a través de algún tipo de bus.

El PIC elegido fue el 18F4320 de la empresa MicroChip. Algunas características del PIC que se tomaron en cuenta tanto en la elección del microcontrolador como en la programación son:

- Velocidad del clock de 40 MHz a través de un cristal de 10 MHz.
- 13 conversores A/D de 10 bits
- 4 timers
- Hasta 32 puertos de E/S
- El PIC implementa el protocolo RS-232.

5.2 Nodo PC

Los requerimientos que determinaron la elección de la estructuración del nodo PC fueron básicamente los siguientes: productividad en el desarrollo, usabilidad y amigabilidad del sistema (ya que este nodo

será el punto de entrada para el usuario), mantenibilidad y claridad respecto a la arquitectura. Es por esto que se decidió utilizar un PC para este propósito. El lenguaje que se eligió para la programación fue Java y la interfaz de usuario fue desarrollada usando Java Swing.

5.3 Comunicación entre nodos

La decisión de utilizar una interfaz serial RS-232 viene a raíz de que muchos microcontroladores vienen con el protocolo RS-232 integrado, otros proveen un puerto de comunicación inalámbrica, pero son los menos.

6 ESTRATEGIA MOTRIZ

El robot construido no es "inteligente" por si mismo, es decir que el lazo de control no se cierra dentro del robot per se, sino por afuera. Para hacer que el robot camine, se eligió un enfoque de lazo abierto y en principio determinista, en el cual el robot sigue una secuencia de movimientos, sin utilizar la información recabada por los sensores. Estos movimientos conducen a que el robot pase por una cantidad de poses, y mediante la concatenación de estas poses se logra que el robot avance. Esto está inspirado en que la locomoción es un proceso mayoritariamente repetitivo, en el sentido que esto se realiza casi sin pensar, y casi siempre siguiendo las mismas trayectorias. De esta manera, el robot actúa como una marioneta cibernética, donde el marionetista es un programa de control. Si bien entre posición y posición no es posible forzar las articulaciones a seguir otra trayectoria que la que se calcula como se describirá en la subsección Perfil de actuación, intercalando suficientes posiciones intermedias, muy cercanas unas de otras, se puede llevar al robot por una trayectoria arbitraria.

7 ACTUADORES Y SENSORES

Debido a la popularidad, facilidad de uso y relativo bajo costo de los motores en aplicaciones de robótica, se decidió usar un motor para implementar cada articulación de un grado de libertad. Las articulaciones de más de un grado de libertad se realizaron uniendo varios motores de alguna manera conveniente (ver la sección 2). Dentro de los tipos de motores disponibles, consideramos que debido a su buena relación peso/par, facilidad de interfacear y relativo bajo costo, los servomotores eran la mejor opción. Los motores elegidos fueron los servomotores Futaba S3003. Estos motores tienen además la ventaja de estar internamente realimentados, es decir que a diferencia de los motores de corriente continua, o los motores paso a paso (steppers), cuentan con un hilo de control, por donde se les indica a qué posición deben ir.

7.1 Perfil de actuación y coordinación

Es deseable que los movimientos de todos los motores comiencen y terminen en el mismo momento, ya que de esta manera se obtiene la menor velocidad posible en cada articulación: si una articulación terminara antes que las demás, entonces podría hacerse el mismo

movimiento con una velocidad menor para poder terminar al mismo tiempo. Las velocidades más bajas, además de dar movimientos más suaves, redundarán en una aproximación más precisa al enfoque cuasiestático, es decir que cuanto más bajas sean las velocidades que utilicemos, y siempre y cuando la proyección del centro de gravedad (COG) caiga en el polígono de sustentación, el robot permanecerá en equilibrio estable.

La mejor manera de hacer que los movimientos de las articulaciones estén sincronizados es imponiéndole una velocidad constante a todos los motores, proporcional al desplazamiento que debe tener cada motor. Cuando a un servomotor se le indica que debe ir a una posición dada, la velocidad con que el motor se mueve no es constante (sigue un comportamiento de segundo orden amortiguado). Es por ello que recurrimos a una técnica que permite obtener una velocidad constante, denominada Ramping [5]. El ramping consiste en ir guiando al motor hacia posiciones consecutivamente cercanas hasta llegar a la posición deseada.

8 DISEÑO MECÁNICO

Luego de evaluar varias herramientas de simulación, decidimos utilizar ODE [6], ya que es la más adecuada para ser usada en el proyecto por proveer la funcionalidad necesaria y por existir posibilidad de adaptarla al contarse con el código fuente. También es un punto importante a favor de ODE la existencia de una comunidad activa de usuarios y desarrolladores, que es útil para obtener soporte. ODE es una biblioteca C/C++ que cuenta con tres partes:

- Manejo de cuerpos rígidos y articulaciones
- Asociación de geometrías a los cuerpos y colisiones
- Visualización de la simulación

Para diseñar una simulación se debe programar el código que genere: los cuerpos, las articulaciones, las geometrías asociadas a los cuerpos y un ciclo de simulación en el cual se avanza el tiempo mediante pasos discretos y se visualizan los objetos en cada paso.

8.1 Diseño del robot

Como se vio en la sección 2, la configuración del robot es de siete segmentos unidos por seis articulaciones. Para poder realizar una simulación más sencilla se modeló cada segmento como un cuerpo simple con masa uniforme.

8.1.1 Datos del Robot Simulado

Mediante la experimentación con la simulación se pudo comprender mejor el problema que afrontamos y obtener datos que nos ayudan para enfrentar la construcción del robot. Observamos que si las articulaciones no están correctamente coordinadas el robot se cae. Esto determinó que el controlador de los motores tenga un buen control de velocidad o de otra manera la coordinación del movimiento de las articulaciones no sería posible.

Tabla 1. Datos del robot simulado.

Dimensiones del pie	50 x 90 x 35 mm
Masa del pie	60 g
Dimensiones del muslo	8 cm de largo, 2 cm de radio
Masa del muslo	90 g
Masa del cuerpo	350 g
Altura Total	27 cm
Masa Total (g): 750	750 g
Dimensiones de la canilla	8 cm de largo, 2 cm de radio
Masa de la canilla	50 g

No existen desarrollos similares que utilicen motores de tan bajo torque. Lamentablemente los servos de mayor torque tienen un costo entre tres y diez veces superior. En la simulación observamos cómo el torque del que disponemos en los motores seleccionados es muy poco, ya que si bien el robot camina, no se cuenta con mucho margen de seguridad. La construcción del robot debe lograr por lo tanto un tamaño y peso mínimos, pero los que simulamos ya constituyen un desafío.

En cuanto a la validación de la estrategia propuesta para caminar, el hecho de que las velocidades aplicadas lleven a una secuencia de movimientos que haga caminar al robot dio un buen nivel de seguridad en cuanto a la validez cinemática de la misma. La simulación brindó por lo tanto confianza de que si las velocidades son suficientemente pequeñas un robot puede caminar con esta estrategia.

8.2 Construcción

Se debieron diseñar piezas para un prototipo funcional de robot bípedo. Como se vio en la sección 2 es deseable que los ejes de cada DOF de una articulación converjan a un punto (que define la articulación), así como también que estos puntos estén alineados a lo largo de la pierna en la posición natural del robot. La distribución de masa debe ser lo más similar posible a una distribución uniforme. La movilidad de las articulaciones no debe ser menor que el rango que utiliza la simulación para caminar y es deseable que se acerque lo más posible a la de un ser humano. Es deseable construir el robot lo más pequeño y liviano posible para que el mismo pueda funcionar con actuadores de poco torque, por tener estos un costo menor que los de mayor torque.

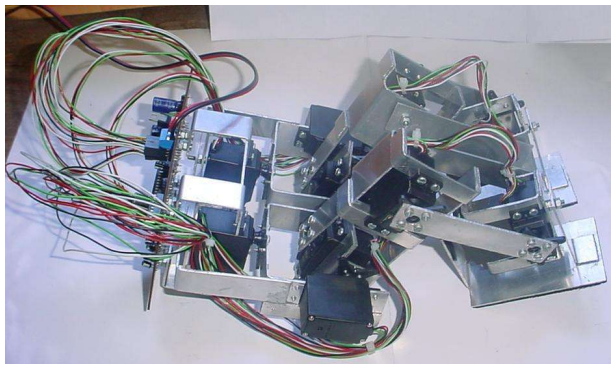

Figura 2. El robot HDP.

El tamaño de los servos impone un mínimo al tamaño que es posible lograr. El tamaño que hemos simulado ya constituye un desafío porque apenas deja el lugar necesario para los servos. También el peso supone un desafío ya que de los 750 gramos del robot simulado, 480 corresponden a los motores y sólo 270 gramos quedan disponibles para las piezas.

Las piezas deben proveer la posibilidad de fijar a ellas los actuadores con los ejes en posiciones adecuadas, y a su vez tienen que

evitar que los servos estén sujetos a fuerzas que los perjudiquen. Es importante también que las piezas mismas se deformen lo menos posible. Adicionalmente, los pies deben brindar el contacto con el piso, así como el cuerpo debe brindar el soporte para la electrónica del robot.

Las piezas deben poderse construir con las técnicas a las que tenemos acceso y su costo debe ser bajo. Una vez diseñada una pieza se requiere que la misma pueda ser construida y probada en un corto tiempo y en forma económica para continuar con el trabajo de diseño.

8.2.1 Herramientas

Para poder lograr los objetivos planteados en la sección anterior se deben utilizar en la construcción herramientas de fácil acceso y cuya operación no conlleve riesgos de seguridad ni requiera de una pericia de la cual no dispongamos. Por ello nos planteamos la construcción utilizando únicamente:

- Morsa portátil
- Sierra manual
- Taladro de baja velocidad y mechas de medidas estándar
- Punta y martillo
- Destornillador
- Trincheta

8.2.2 Materiales

Por sus características de precio, disponibilidad y adecuación para nuestro uso se seleccionaron los siguientes materiales:

- Chapa de aluminio
- Tornillos y tuercas de medidas estándar
- Cemento de soldadura plástica
- Precintos
- Goma eva

Utilizar el eje de los servomotores para articular el DOF es el enfoque más sencillo y fue con lo que se realizaron las primeras pruebas. Sin embargo al intentar completar el robot las limitaciones de esta estructura se hicieron evidentes. Tanto las piezas como los brazos de plástico de los servos y el eje de los mismos cedían, deformando cada articulación lo suficiente para que la suma de estos errores deformara completamente el robot. Además el esfuerzo provocado sobre los ejes de los servos es considerable y es probable que conduzca a roturas.

Los servos de mayor precio incluyen rulemanes en los ejes, ejes de metal y también existen brazos de metal que se pueden adquirir por separado. Si se contara con este tipo de servos quedaría aún el problema de la deformación de las piezas que habría que solucionar, cosa que tampoco es fácil con las limitaciones con las que trabajamos en este sentido.

El enfoque que finalmente se utilizó en el robot es la utilización de dos puntos de apoyo para articular cada DOF. Uno de los puntos es el eje del servo y el segundo es un rodamiento alineado con este eje y que actúa entre los dos segmentos unidos por el DOF. Si bien este enfoque soluciona el problema, el mismo complica significativamente el diseño, aumenta el peso del robot y trae un nuevo problema a afrontar, que es fabricar el

rodamiento.

El aluminio se seleccionó por su baja densidad y relativa facilidad en su manipulación. Se experimentó con distintos grosores de chapa de aluminio llegándose a la conclusión de que 1.5 mm era un grosor apropiado para la construcción.

Para construir el prototipo se podrían cortar las piezas a partir de su desarrollo plano sobre la chapa, para luego agujerear y doblar. Sin embargo los cortes internos son complicados de realizar con las herramientas con las que contamos, por lo que se optó por fabricar las piezas en partes y se utiliza pegamento de soldadura plástica para unirlas.

9 DISEÑO ELÉCTRICO

9.1 Alimentación

Para la alimentación se implementó una fuente regulada cuya entrada viene de una fuente de PC de 12 V, ya que los servomotores deben alimentarse a 6 V para obtener el mejor torque (voltaje que no se encuentra disponible). La solución más fácil hubiese sido un circuito regulador, pero las demandas de corriente de los motores son muy grandes (hasta 600 mA cada uno) por lo que hubo que pensar en una solución que diera hasta 5 A sin caída de potencial. La fuente fue construida en un circuito impreso al igual que la placa controladora, y se le adosaron disipadores en el regulador y en el transistor de potencia, y un ventilador para favorecer la disipación. Esto es muy importante ya que el transistor puede llegar a disipar 115 W, lo que eleva la temperatura del circuito muy rápidamente. La fuente construida alimenta directamente a los motores. El PIC se alimenta en 5 V, pasando esta fuente por un regulador lineal 7805.

9.2 Modificación de los motores

Con el fin de obtener un lazo de realimentación de la información de los motores, se les realizó una modificación, que conecta el borne del potenciómetro en el eje del motor con un conversor A/D del PIC. De esta forma, el PIC sensa la posición en la que realmente se encuentra el motor, y esta información se transmite al PC, donde pueden tomarse decisiones más informadas acerca de las acciones a tomar. Esta es la característica crucial y determinante que permite que nuestra estrategia funcione, ya que posibilita la digitalización de las posiciones de los motores.

10 SOFTWARE

Por un lado, el PIC debe poder comunicarse con un PC utilizando un protocolo descrito en la siguiente sección, a través del puerto serial. Por otro, el PIC debe poder controlar todos los motores del robot simultáneamente, sin errores perceptibles y con la mayor precisión posible, y manejar las salidas digitales presentes en el sistema. Además, el PIC sensa las posiciones de los motores, escrutinia el estado de los mismos (si han llegado a la posición deseada o no), lee las entradas digitales, y comunica dichos datos al PC a través del puerto serial.

10.1 Protocolo de comunicación

En esta sección se describe el protocolo de

comunicación que se utiliza para la comunicación entre el PIC y el PC. El protocolo define dos clases de mensajes: los mensajes de comandos, que viajan del PC al PIC y los mensajes de notificaciones, que viajan del PIC al PC. Además, se utilizan las técnicas de stop-and-wait y piggybacking para enviar los ACK. No se incluye ningún tipo de método de detección o corrección de errores, ya que en las condiciones dadas (cables cortos y sin pérdidas), consideramos el cable serial como confiable.

10.2 Software embebido

Como se desea tener una buena precisión a la hora de controlar motores y leer sensores, es necesario contar con un módulo que implemente las funciones aritmético-lógicas de 16 bits (alu16), ya que el microcontrolador no las proporciona. Este módulo es utilizado por los demás como infraestructura.

Por otro lado, el módulo ramping realiza el ramping lineal de los motores, actualizando los valores de posiciones de los motores según la velocidad asignada a dicho motor. Básicamente, si la diferencia entre la posición calculada en la que se encuentra el motor (que es a la que se comanda que vaya el motor, y que a velocidades bajas coincidirá con la real, que es la leída a través del conversor A/D desde el potenciómetro) y la posición a la que debe ir difieren en un valor mayor a un cierto umbral, la posición calculada se actualiza según la velocidad y sentido de giro necesario. Se define para ello un cierto umbral para evitar el efecto hunting (es decir que el motor no oscile en torno a la posición final).

Otro módulo importante es el encargado de calcular las máscaras de bits que se detallan más adelante para la coordinación de motores, y por ende de cómo generar las ondas PWM.

El sistema se estructura entonces en una rutina de atención a interrupción que despacha la interrupción adecuada testeando ciertas banderas, un programa principal que realiza polling de ciertas condiciones y rutinas de atención a las interrupciones. En concreto, existen tres interrupciones:

- Recepción de un byte por el puerto serie
- Timeout del timer de los motores
- Timeout del timer de inicio de transmisión de paquete con información acerca de los sensores y motores

10.2.1 Timeout del timer de los motores

Los motores utilizados son servomotores, y precisan recibir una onda PWM cuyo período debe estar entre los 18 y 22 ms. Esta rutina debe generar las ondas PWM de los motores sin atrasarse y con la máxima precisión posible. Para ello, se utilizó un enfoque basado en máscaras de bits.

Se define un evento como una dupla <motores, tiempo>, entendiendo que los motores deben bajarse en un cierto tiempo dentro del ciclo de la onda. Como varios motores pueden tener que bajarse en un mismo instante, es preferible utilizar máscaras, ya que con una única escritura a un puerto pueden habilitarse/deshabilitarse varios motores. Al

utilizar 12 motores se hace necesario usar dos puertos (ya que cada uno puede direccionar ocho motores).

El procesamiento se reparte en dos: por un lado se calculan las máscaras y los tiempos, y por otro lado se comandan los motores con esta información. Para calcular las máscaras, se tiene en cuenta cuáles motores están habilitados. Si dos motores deben bajarse en el mismo momento (es decir deben dirigirse a la misma posición), en la máscara aparecerán los dos bits en estado bajo simultáneamente. Esto se extiende en una técnica que denominamos "Clamping".

10.2.2 Clamping

El clamping (encepado) es una técnica que desarrollamos que permite optimizar el control de los motores. En esencia es simple: la rutina de control de motores va a tardar unas cuantas instrucciones en realizar su cometido. Esta cantidad se irá acumulando evento a evento. Cuando llega al último motor podemos tener un gran desfasaje con lo que deberíamos tener. Por lo tanto, si dos motores deben bajarse muy próximos uno del otro (menos del tiempo que tarda la rutina de control), es conveniente bajarlos conjuntamente, ya que de otro modo no solamente no se logrará la posición objetivo, sino que se retrasarán todos los motores y el sistema se comportará de forma errática. De esta forma, el sistema está programado para tomar en cuenta sus propias "limitaciones" de tiempo, y es de alguna manera consciente de sí mismo.

10.2.3 Coordinación de los motores

Se debe ejecutar el algoritmo que realiza la técnica de Ramping para obtener la velocidad constante deseada (ver sección 7.1).

10.2.4 El programa principal

El programa principal analiza las siguientes banderas en un bucle infinito:

- Fin de recepción de un paquete a través del puerto serial
- Se bajaron todos los motores (es decir, el valor de la onda PWM en este momento es 0 V para todos los motores)
- Inicio de transmisión del paquete saliente
- Armado del paquete saliente

Debido a que la lectura analógica involucra una conversión A/D, se precisa de cierto tiempo de setup de los conversores, y de un cierto tiempo de conversión. Es por eso que se hace un polling sobre una bandera del PIC que indica cuándo termina la conversión. Antes de la transmisión de cada byte se hace polling sobre una bandera del PIC que indica cuándo latch de transmisión se ha vaciado. Una vez que esto ocurre, se avanza la posición del puntero que indica el siguiente byte a transmitir dentro del paquete saliente.

10.3 Software del PC

El nodo PC debe manejar los motores a nivel de la posición de cada uno de ellos, así como de encargarse de la coordinación del conjunto y de concatenación de movimientos. El software del PC es el que realiza estas funcionalidades. Las funcionalidades clave que presenta el software se resumen a

continuación:

- Permite comandar al conjunto de motores hacia una pose (stance) dada con velocidades dadas para cada motor. Una pose consiste del conjunto de las posiciones de todos los motores.
- Calibrar el conjunto de motores. Esto es, encontrar parámetros que mapeen los valores de las lecturas de posición con los valores de los comandos que se envían al PIC correspondientes.
- Capturar (digitalizar) la pose en la que se encuentra el robot actualmente, y guardarla para su futura utilización.
- Definir un andar (gait). Un andar es una secuencia de instrucciones, donde una instrucción puede ser una de las siguientes:
- Ir a una pose con una velocidad máxima dada. La articulación que debe recorrer más ángulo irá con esta velocidad, mientras que las demás lo harán con velocidades proporcionales a su recorrido.
- Repetir un conjunto de instrucciones un número finito de veces. Esto se especifica con las instrucciones Repeat [n] y EndRepeat.
- Ir a un paso determinado dentro del andar. Cada instrucción queda naturalmente numerada por el orden en el que se define dentro del andar. Cuando se ingresa una instrucción Goto [n], se está indicando que al llegar a ese paso dentro del andar, se debe continuar en el paso n -ésimo.

Es deseable que el software siga una arquitectura extensible, y que esta arquitectura permita extender el sistema de forma de integrar información que excede el alcance de este Proyecto, como ser por ejemplo tener en cuenta la lectura de los sensores para generar trayectorias en tiempo real, etc. (ver Trabajos a futuro). Notar que debido a esta flexibilidad, es posible por ejemplo definir más instrucciones de ser necesario.

11 CONCLUSIONES

Una vez integradas todas las partes del sistema (salvo el simulador, que se utiliza por separado) se cuenta con un robot controlado desde un PC que puede realizar movimientos programados. El proceso de calibración dde los servomotores asocia valores de posición leídos de los motores con los comandos enviados a los mismos. Con los motores calibrados, estando los servomotores desactivados (velocidad 0) se posicionó manualmente al robot en la posición deseada y se digitalizó dicha posición.

Luego de comprobar la funcionalidad del sistema, nos dispusimos a hacer que el robot caminara. Para ello se diseñaron posiciones similares a las simuladas. Aparecen varios inconvenientes al intentar cumplir este objetivo. Primero, hubo que cambiar el diseño para constreñir a la estructura doblemente en lugar de tener un solo punto de restricción, ya que de lo contrario el juego que resultaba hacia imposible realizar los movimientos deseados.

El torque de los motores no resultó ser suficiente para caminar igual que en el simulador. Para solucionar este problema, se inclina el cuerpo (articulaciones de la cadera) hacia adelante para ayudar al movimiento adecuado del centro de masa, y de esa manera se redujo el torque necesario.

Las posiciones con ambos pies en el suelo no presentaron mayores dificultades, el robot se mueve con facilidad entre las mismas. Cuando el robot se agacha, el torque necesario aumenta y también aparece la limitación vista en la sección 2.1 por la falta de movilidad, que lleva a que algunas posiciones de los pies no permitan que el robot se agache sin que choquen sus rodillas.

Se diseñó un andar a través del cual el robot puede dar cuatro pasos seguidos, partiendo y terminando en la posición de Erguido.

11.1 Evaluación de resultados

Se logró plantear una estrategia original para hacer caminar a un robot bípedo y demostrar que la misma funciona correctamente. Otro aporte original fue la modificación de los servomotores para lograr lecturas de posición de los mismos, que es imprescindible para poder construir a bajo costo un prototipo que permita utilizar eficazmente la estrategia planteada. Esta modificación de los servomotores también permite, eventualmente, el desarrollo de otras estrategias de control más avanzadas sobre un robot de bajo costo, por ejemplo, usando cinemática inversa, u alguna técnica de control. También se considera exitoso el desarrollo de una plaqueta de control embebido basada en un microcontrolador y de la interacción de la misma con un PC. Esta plaqueta nos proveyó de funcionalidades que no encontramos en productos comerciales similares, y constituye un producto en sí mismo que puede ser utilizado en cualquier otro proyecto de robótica que necesite un controlador de servomotores. En cuanto al diseño del prototipo los resultados fueron buenos, más aún si tenemos en cuenta que carecemos de formación en cuanto a diseño mecánico.

Si bien la construcción del prototipo de robot funcionó correctamente, el mismo muestra una funcionalidad limitada y es bastante frágil.

A continuación se presentan las conclusiones más importantes que aparecen luego de realizado el proyecto:

- El trabajo en robótica bípeda, para poder desarrollar un robot desde cero, debe ser necesariamente multidisciplinario, requiriéndose conocimientos muy avanzados en Ingeniería Mecánica, Física, Electrónica, Computación y Biología entre otros.
- El presupuesto de los proyectos de esta área debe ser necesariamente elevado. Lamentablemente, no es posible trabajar en robótica bípeda sin materiales altamente especializados, cuyos costos son altos. Los proyectos de investigación que se presentan a eventos como la RoboCup [7] llevan robots con costos oscilando entre los

U$S 5.000 y U$S 10.000, aproximadamente.

12 TRABAJOS A FUTURO

12.1 Construcción de un robot más robusto

Si bien el prototipo logró demostrar las posibilidades de nuestro enfoque, no es lo suficientemente robusto debido a sus motores de bajo precio y a su construcción mediante técnicas poco avanzadas. Un trabajo interesante a realizar es la construcción de un robot con un diseño mecánico mejorado y adecuado para técnicas de construcción más sofisticadas y de ser necesario materiales más apropiados. Independientemente de contar con las nuevas piezas, sería importante también contar con mejores servomotores. Las metas de este trabajo son bastante concretas y el mismo es acotado de tal forma que creemos que puede ser realizado en relativamente poco tiempo si se cuenta con acceso a la tecnología y presupuesto apropiados. Basándonos en nuestra experiencia, pensamos que con las herramientas disponibles en una carpintería de aluminio y con un presupuesto entre U$S 1.000 y U$S 1.200 se puede lograr una buena construcción. A continuación mencionamos algunos puntos a tener en cuenta como posibles mejoras:

- Material de la estructura
- Amortiguamiento de las oscilaciones
- Protección contra caídas
- Motores de mayor potencia
- Separación de las señales de la alimentación
- Fuente conmutada en lugar de regulada
- Protección contra cortocircuitos en la fuente
- Mejora de la precisión en la lectura de posiciones

12.2 Construcción de un robot humanoide

Un siguiente paso lógico sería construir un robot humanoide, con cuerpo, brazos y cabeza. Para ello sería necesario ampliar la plaqueta de control y realizar un estudio para seleccionar nuevos actuadores para las piernas, ya que el peso sería considerablemente mayor. También sería importante dotar al robot de la posibilidad de incluir los sensores necesarios para otros trabajos propuestos, como el de la siguiente sección. Este es un proyecto multidisciplinario y está muy ligado al propuesto en la siguiente sección.

12.3 Estudios sobre locomoción bípeda

Si se dispone de un robot robusto y con sensores apropiados podemos trabajar para entender y mejorar la forma de caminar. Para esto se necesitaría estudiar el problema desde el punto de vista físico-matemático.

12.4 Trabajos a más alto nivel

La arquitectura planteada posee varios puntos no implementados que hacen a un sistema más completo. Como primera medida cabe destacar que la arquitectura propone tres capas adicionales: walk, pathPlanning y biped. Es importante notar que el lazo de realimentación de la lectura de los sensores se da en estas capas de más alto nivel, y es ahí donde el robot puede empezar a interactuar con su entorno de

forma menos "ciega". También hay que observar que si bien el enfoque de este proyecto fue más bien interdisciplinario, por integrar conocimientos y experiencias de electrónica, mecánica y computación, el proyecto que surja para completar la arquitectura, al disponer de la parte física resuelta, podrá enfocarse en áreas más especializadas. Esto se debe principalmente al enfoque en capas, ya que para este nuevo proyecto, el robot ya existe, y no hay que conocer su estructura interna ni su funcionamiento; basta con conocer cuál es su interfaz de servicios.

Otro punto a considerar es la integración del simulador como herramienta de predicción y corrección de la trayectoria del robot. En otras palabras, cerrar el lazo de control a un nivel mucho más alto, utilizando la predicción del simulador (que a su vez utiliza las lecturas de los sensores) para corregir la trayectoria real.

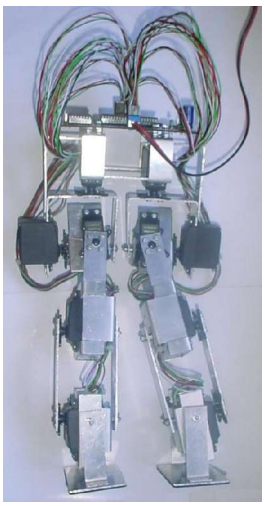

Figura 3. El robot HDP.

REFERENCIA BIBLIOGRÁFICA

[1] Samuel R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Square methods", 2004.

[2] Lynx Motion, Biped Scout Robot, http://www.lynxmotion.com, visitada mayo de 2005.

[3] Simulation and Systems Optimization Group - Technische Universität Darmstadt, DD Robot, http://www.sim.informatik.tu-darmstadt.de, visitada mayo de 2005.

[4] J. Duysen, "Human gait as a step in evolution", Brain, Vol. 125, No. 12, pp. 2589-2590, Oxford University Press, 2002.

[5] Lezama D. y Alexander Sklar, "Construcción de Robots Bípedos", Proyecto de Grado y Proyecto de Fin de Carrera, Instituto de Computación - Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, http://www.fing.edu.uy/~pgrobip/, Mayo, 2005.

[6] Russell Smith, Open Dynamics Engine, http://www.ode.org, visitada mayo de 2005.

[7] Robocup, http://www.robocup2005.org, visitada mayo de 2005.