Proyecto de Grado 2004

Construcción de Robots Bípedos

Facultad de Ingeniería

Universidad de la República Oriental del Uruguay

Damián Lezama Alexander Sklar

Tutores: Ing. Gonzalo Tejera, Ing. Rafael Canetti, Ing. Carlos Martínez

14 de abril de 2005

Resumen

Este documento es el informe final del Proyecto de Grado denominado "Construcción de Robots Bípedos", para las carreras de Ingeniería en Computación e Ingeniería Eléctrica. El proyecto propone el estudio del Estado del Arte en robótica bípeda, y la construcción de un prototipo de robot bípedo capaz de caminar cuasiestáticamente. Entre los aportes originales de este Proyecto 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.

Agradecimientos

Unidad de Recursos Informáticos de la Facultad de Ingeniería, Ing. Gonzalo Tejera, Ing. Eduardo Grampín, Ing. Rafael Canetti, Ing. Carlos Martínez, Daniel y Eduardo Thévenèt, Macarena Harispe, Plaza Hobbies, Aluminios del Uruguay, Panadería Panes.

Índice general

Capítulo 1

Introducción

1.1. Motivación del Proyecto

La robótica bípeda conforma un área de investigación con gran crecimiento en los últimos años. Si bien el desplazamiento mediante ruedas es más eficiente y permite mayor velocidad, los robots con patas son más versátiles y pueden desplazarse en terrenos irregulares. En particular, los bípedos son especialmente aptos para manejarse en nuestro entorno, por contar con características similares a las de los seres humanos. Es así que sin necesidad de modificar nuestros hogares y lugares de trabajo estos robots pueden realizar tareas por nosotros, siendo particularmente interesante su aplicación a trabajos que ponen en riesgo la salud o la vida de las personas. Los robots bípedos no sólo son más aptos para nuestro medio por sus capacidades en cuanto a locomoción, sino que también los humanos nos podemos adaptar más fácil a la interacción con ellos que con otro tipo de robots, por ser los bípedos más semejantes a nosotros.

Otra motivación muy importante para el desarrollo del área es su aplicación al diseño de prótesis para personas con discapacidad motriz. Gran cantidad de personas sufren de estas discapacidades, y el avance tecnológico de las prótesis se debe en parte al desarrollo de la robótica bípeda.

Los bípedos conforman la categoría más nueva en las competencias de fútbol de robots. Estas competencias brindan un ambiente apto para el desarrollo de nuevas tecnologías y su aplicación al fútbol, una tarea que requiere habilidades similares a las necesarias para realizar diversos trabajos donde se presentan ambientes dinámicos y de tiempo real. Nuestra facultad participó por primera vez en el año 2003 en el Campeonato Argentino de Fútbol de Robots en categoría simulada y este proyecto se enmarca en un primer acercamiento hacia la categoría de bípedos.

1.2. Objetivos

Este proyecto plantea el estudio del estado del arte en robótica móvil y robots bípedos, para luego diseñar y construir un prototipo de bajo costo capaz de desplazarse caminando. Los resultados obtenidos deben servir de base para trabajos futuros que apunten al desarrollo de robots bípedos capaces de participar en competencias de fútbol de robots y para el desarrollo de investigación en esta área.

1.3. Organización del documento

En el capítulo 2 se expone el estudio realizado sobre el Estado del Ate en robótica bípeda para luego en el capítulo 3 explicar el enfoque particular en el que basamos nuestro Proyecto y las decisiones y contenido más importantes del mismo. En el capítulo 4 explicamos como y para qué utilizamos simulación. El capítulo 5 explica el diseño mecánico de nuestro robot, el 6 el diseño eléctrico, el 7 el funcionamiento software embebido y el 8 la arquitectura del software que corre en el PC. Finalmente en el capítulo 9 presentamos nuestras conclusiones y proponemos trabajos a futuro.

En los apéndices se presenta información complementaria a la presentada en el cuerpo del informe y detalles técnicos: el apéndice A expone los recursos utilizados en el proyecto, el B presenta el glosario de términos que aparecen en este documento. En el apéndice C se explican algunos conceptos de control, en el D se explican los relevamientos y mediciones realizadas a lo largo del proyecto y en el E se resume información sobre cinemática inversa. El apéndice F explica como se realiza el control de servomotores de corriente continua como los de hobby.

Capítulo 2

Estudio del Estado del Arte

2.1. Los Bípedos de mayor presupuesto

Los proyectos más ambiciosos sobre robótica bípeda se encuentran actualmente en manos de las empresas Sony y Honda. Estas empresas apuntan a la creación de robots que constituyan productos de venta masiva y sus desarrollos insumen años de trabajo, miles de profesionales y cientos de millones de dólares de presupuesto. En esta sección describimos brevemente las características de sus robots.

2.1.1. Sony QRIO

Figura 2.1: Robot Sony QRIO

Este robot fue concebido con fines de entretenimiento y además del control motriz, está equipado con una gran cantidad de hardware y software dedicado a la interacción con las personas. El QRIO camina en forma dinámica utilizando el método de ZMP (Zero Moment Point, punto de momento cero) y es capaz de adaptarse a superficies irregulares y de reaccionar a fuerzas externas. De ser empujado el robot puede decidir dar un paso hacia la dirección adecuada para equilibrarse.

Para lograr todo esto este robot cuenta con varios tipos de sensores que realimentan su sistema de control. Estos mecanismos sin embargo sólo están garantizados para compensar elevaciones de hasta 1 cm en el terreno y pendientes de hasta 10 grados, las especificaciones no explicitan qué tanta fuerza externa el robot es capaz de compensar. En caso que el robot detecte que se va a caer y no pueda remediarlo adopta una estrategia de amortiguación de impacto y luego de la caída es capaz de levantarse en forma autónoma.

Agregado a las funcionalidades básicas de locomoción el robot también cuenta con sistemas de reconocimiento de objetos mediante su visión estereoscópica y detección de fuentes de sonido mediante siete micrófonos ubicados en su cabeza. Dentro del software que incluye se encuentran programas capaces de planificar trayectorias y trazar mapas entre otros.

Para la interacción con los seres humanos el QRIO está equipado con software de reconocimiento de personas mediante su cara o su voz y reconocimiento de palabras habladas. Este robot también es capaz de hablar y cuenta con un sistema que simula conversaciones y emociones. En su diseño están contemplados aspectos de seguridad, como por ejemplo detección de obstrucciones en las articulaciones que permiten evitar que las mismas lastimen al usuario, control de temperatura para que sea siempre seguro tocarlo y un peso suficientemente liviano como para no lastimar a una persona si cae sobre ella. También se tuvieron en cuenta en su diseño aspectos de confort, como por ejemplo una operación silenciosa.

El Sony QRIO es un robot autónomo con una batería que le permite una hora de operación y cuenta con interfaz de red inalámbrica con la cual puede interactuar con equipos de computación, aunque está concebido para interactuar directamente con personas mediante lenguaje hablado.

Al momento de realizar este informe el Sony QRIO aún no se ha puesto a la venta, pero el precio que se anuncia para su lanzamiento es de U$S 50.000

Desarrolladores: Sony Corporation

Homepage: http://www.sony.net/SonyInfo/QRIO/

Altura: 58 cm

Peso: 6.5 kg

Velocidad: 10 cm/s en superficies irregulares y 33 cm/s en terreno plano

Procesamiento: 2 procesadores RISC de 64 bits, 128 Mb RAM, sistema operativo de tiempo real Amperios (propietario de Sony)

DOF (Degrees of freedom, Grados de libertad): 4 en el cuello, 2 en el cuerpo, 5 en cada brazo y 6 en cada pierna. Un total de 28 grados de libertad más 5 dedos en cada mano.

Sensores: 3 Infrarrojos de distancia, 1 sensor de aceleración de 3 ejes y 2 de 2 ejes, 4 sensores de presión en cada pie, 6 sensores de temperatura, 5 sensores de tacto tipo llave en manos y hombros y 1 con control de presión en la cabeza, 7 micrófonos, 2 cámaras de video.

Actuadores: Servomotores propietarios

Alimentación: 12 baterías de alto voltaje de litio, tecnología propietaria de Sony

Otros: Leds multicolor en los ojos, parlante, slots para PC-CARD y Memory Stick, Interfaz de red inalámbrica

Cuadro 2.1: Características del robot QRIO

2.1.2. Honda ASIMO

Figura 2.2: Robot Honda ASIMO

A diferencia del robot QRIO de Sony, que se centra en el entretenimiento, el ASIMO de Honda fue concebido desde un principio pensando en asistir a las personas y realizar tareas domésticas en nuestro entorno, por ello su altura es mayor ya que fue creado teniendo en cuenta las dimensiones y posición habitual de mesas, sillas, llaves de luz y escaleras entre otras cosas. También se tuvo en cuenta en el desarrollo de ASIMO que sus ojos quedaran a la altura de los ojos de una persona

adulta sentada, con miras a que el robot fuera cómodo para su interacción en la oficina. Los últimos avances en el proyecto incorporan al robot la posibilidad de correr, o sea permanecer en el aire un instante a lo largo del paso, sin embargo los resultados aún distan mucho del paso de un ser humano ya que ASIMO sólo permanece en el aire 5 centésimas de segundo en cada paso cuando corre.

Si bien los primeros prototipos de ASIMO utilizaban patrones preestablecidos para caminar, las versiones más modernas analizan la situación y generan los movimientos necesarios teniendo en cuenta la dinámica del robot. Este sistema no sólo es reactivo sino que también actúa en forma predictiva, moviendo el centro de masa adecuadamente antes de comenzar por ejemplo un giro.

Al igual que QRIO de Sony, ASIMO cuenta con una gran cantidad de software que le permite reconocer objetos y voces, analizar su entorno y desplazarse por él y brindar muchas posibilidades en cuanto a interacción con seres humanos. También este robot actúa en forma autónoma con baterías que le permiten media hora (hasta una hora en los nuevos prototipos) de autonomía. A diferencia del QRIO, ASIMO no es capaz de levantarse si se cae.

La forma en que Honda comercializa el ASIMO es alquilándolo y en este momento el precio del alquiler es de U$S 150.000 anuales.

A continuación se exponen algunas características técnicas del robot, pero lamentablemente Honda no divulga muchos datos técnicos del mismo.

Desarrolladores: Honda

Homepage: http://world.honda.com/ASIMO/

Altura: 120 cm

Peso: 52 kg

Velocidad: 42 cm/s.

Procesamiento: No se divulga

DOF: 2 en el cuello, 5 en cada brazo, 1 en la mano y 6 en cada pierna. Un total de 26 grados de libertad.

Sensores: Sólo se divulga que cuenta con sensores de 6 ejes en el área del pie, giroscopio y sensor de aceleración.

Actuadores: Servomotores propietarios

Alimentación: Batería de níquel-metal de 40 volts

Otros: Parlante, Interfaz de red inalámbrica

Cuadro 2.2: Características del robot ASIMO

2.1.3. Resumen

La información recabada sobre estos robots nos permite conocer mejor la dificultad del problema que estamos afrontando, aunque lamentablemente por ser desarrollos que mantienen en secreto gran parte de sus características, no podemos obtener demasiado conocimiento de aquí. En cuanto

a las dificultades que quedan en evidencia se puede observar el bajo rendimiento de las baterías en ambos robots, problema que estas empresas están muy preocupadas en solucionar, así como también las limitaciones en cuanto a velocidad que todavía son muy grandes incluso contando con la más alta tecnología.

2.2. Desarrollos académicos de mediano presupuesto

Los proyectos que presentamos en esta sección fueron realizados por universidades y en algunos casos patrocinados por pequeñas o medianas empresas. Estos robots no pretenden ser productos aptos para el mercado sino prototipos de investigación. Centramos nuestra atención en algunos de los robots que fueron presentados en la RoboCup 2004 y exponemos a continuación un relevamiento de las principales características de los mismos.

2.2.1. ARIA

Figura 2.3: Robot ARIA

Desarrolladores: Isfahan University of Technology (I.U.T.)
Artificial Intelligence laboratory

Homepage: http://ece.iut.ac.ir/robocup/index.htm

Altura: 60 cm

Peso: 6.5 kg

Velocidad: 2 cm/s

Procesamiento: PC Pentium IV 512 Mb RAM, Windows 98.

DOF: Cada pierna tiene únicamente 4 DOF. Dos DOF adicionales controlan mediante rieles la posición de las baterías horizontalmente en el cuerpo. Un total de 18 grados de libertad.

Sensores: 8 sensores de fuerza para calcular el ZMP. 2 Giroscopios tipo chip. 1 Cámara de video.

Actuadores: 18 Servomotores de tipo hobby

Alimentación: 2 Baterías NiCd 3A 12 V

Cuadro 2.3: Características del robot ARIA

2.2.2. DD

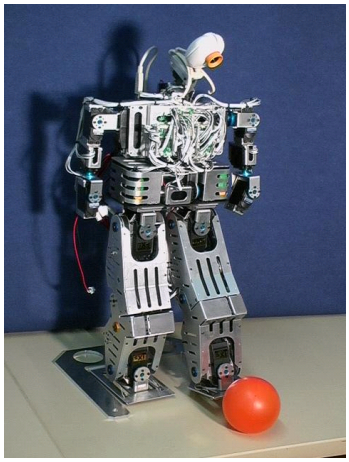


Figura 2.4: Robot DD

Desarrolladores:	Simulation and Systems Optimization Group - Technische Universität Darmstadt
Homepage:	http://www.sim.informatik.tu-darmstadt.de
Altura:	60 cm
Peso:	4.8 kg
Velocidad:	No se especifica
Procesamiento:	Procesador NEC Vr4181A 133 MHz. Linux.
DOF:	6 en cada pierna, 4 en cada brazo, 2 en la cintura y 2 en el cuello. Un total de 24 grados de libertad
Sensores:	3 sensores de fuerza en cada pie. 24 encoder angulares en las articulaciones. Cámara CCD.
Actuadores:	24 Servomotores de tipo hobby
Alimentación:	Batería incluida, no se especifican características

Cuadro 2.4: Características del robot DD

2.2.3. GuRoo

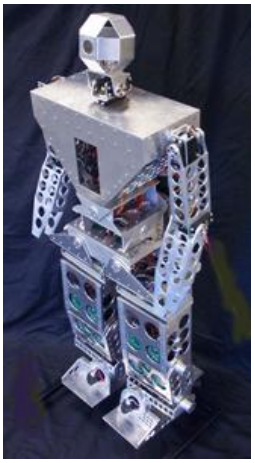


Figura 2.5: Robot GuRoo

Desarrolladores: ITEE, University of Queensland

Homepage: http://www.ite.uq.edu.au/~damien/GuRoo

Altura: 120 cm

Peso: 38 kg

Velocidad: No se especifica

Procesamiento: Compaq iPAQ PDA (208Mhz StrongArm, 32Mb RAM).

DOF: 6 en cada pierna, 3 en el cuerpo, 2 en el cuello y 3 en cada brazo. Un total de 23 grados de libertad.

Sensores: 15 encoders de 500 cuentas/revolución (exactitud 0.001°) ubicados junto con cada motor DC. Acelerómetros, giroscopios y magnetómetros conforman una unidad de medida de inercia. Dos cámaras OmniVision CMOS procesadas por un Hitachi SH4 a 20 cuadros/s

Actuadores: 15 Motores DC con escobillas para las piernas y el cuerpo: Maxon RE32 (32 V nominal). Reducción 156:1 cerámica, 72 % eficiencia. Torque máximo de 10 mNm.
8 Servomotores tipo hobby Hi-Tech HS705-MG. Torque Máximo de 1.4 Nm.

Alimentación: 2 Baterías 42 V 1.5 Ah NiCd para los motores DC y 2 Baterías 7.2 V 1400 mAh NiCd para los servos y la lógica.

Cuadro 2.5: Características del robot GuRoo

2.2.4. Isaac

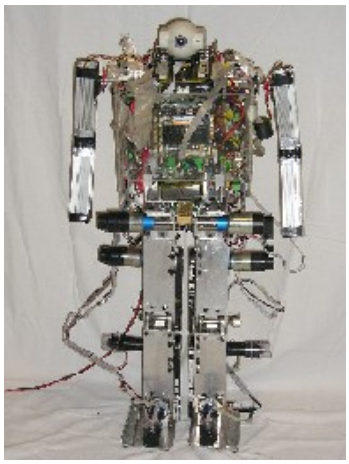


Figura 2.6: Robot Isaac

Desarrolladores: Politecnico di Torino. Dipartimento di Automatica e Informatica

Homepage: http://www.isaacrobot.it

Altura: 83 cm

Peso: 14 kg

Velocidad: No se especifica

Procesamiento: PC104 Procesador de 267 MHz 128 Mb RAM, tarjetas de I/O digital y analógica. Linux con extensiones de Tiempo Real RTAI.

DOF: 16 (6 en cada pierna, 1 en cada brazo, 2 en el cuello).

Sensores: 8 Sensores de fuerza en los pies, encoders incorporados en los motores, potenciómetros en cada articulación, 3 acelerómetros y 2 giroscopios. Webcam.

Actuadores: Motores DC con escobillas y engranajes. 8 motores de 4 Nm de torque y 8 motores de 9 Nm.

Alimentación: 3 Baterías 25 V de hasta 5 A

Cuadro 2.6: Características del robot Isaac

2.2.5. Rope

Figura 2.7: Robot Rope

Desarrolladores: Singapore Polytechnic National University of Singapore

Homepage: http://guppy.mpe.nus.edu.sg/legged_group/humanoidrobot.htm

Altura: 49.5 cm

Peso: 2.5 kg

Velocidad: No se especifica

Procesamiento: PC-104 Pentium. RTLinux, comunicación con memoria compartida entre los módulos de tiempo real y el resto del sistema. Tarjeta DAQ Diamond DMM32AT. Controlador Parallax Basic Stamp bs2p24 para generar PWM para los servos. Utiliza Reinforcement Learning.

DOF: 18

Sensores: Sensores de fuerza en la planta de los pies para determinar el centro de presión. Sensor de inclinación de 2 ejes (Crossbow). Acelerómetro de 3 ejes (Crossbow). Giróscopo para determinar hacia donde apunta el robot (Silicon Sensing System). Cámara.

Actuadores: 18 Servomotores Hitec HS-5945MG, 1.3 Nm de torque

Alimentación: No se especifica

Cuadro 2.7: Características del robot Rope

2.2.6. Resumen

El relevamiento hecho sobre estos robots nos lleva hacia varias tecnologías de sensores, actuadores, elementos de procesamiento y estrategias de control sobre las cuales profundizamos en la sección 2.5. En el caso de estos proyectos encontramos mucha información en los sitios web de los desarrolladores que nos fue muy útil en el desarrollo de nuestro proyecto.

Casi la totalidad de los componentes utilizados en estos robots son estándar y sólo una parte

muy pequeña de ellos fueron mandados hacer a medida por los desarrolladores. Sin embargo casi todos los componentes son muy específicos y por lo tanto no se consiguen en nuestro país, debiéndose recurrir en caso de necesitarlos a distribuidores en el exterior. Otro inconveniente que se vislumbra a partir de este estudio es el alto costo de la mayoría de los componentes. Si bien los desarrolladores no publican detalles de costos, un relevamiento sobre los componentes utilizados nos lleva a que el precio de estos robots oscila entre U$S 5.000 y U$S 10.000, y esto es teniendo en cuenta precios de componentes en origen. La descripción de las experiencias de los desarrolladores nos presenta que en el proceso de desarrollo tuvieron que construir varios prototipos y muchas veces, luego de evaluar el prototipo, cambiar los componentes utilizados por otros más adecuados. Estos datos nos dan una idea del costo que se afronta en este tipo de proyectos.

2.3. Robots bípedos comerciales de bajo costo

Existen varias empresas que se dedican a desarrollar robots de bajo costo para utilizar como juguetes didácticos y pasatiempos tanto para jóvenes como para adultos. Recientemente han aparecido algunos bípedos entre la oferta de robots de este tipo y en esta sección profundizamos sobre dos de ellos. A diferencia de los robots que se expusieron en las secciones anteriores, estos bípedos no tienen brazos, contando sólo con miembros inferiores. Tampoco tienen cámaras ni otros sensores caros o difíciles de procesar y su forma de desplazarse y funcionalidades son bastante limitadas.

2.3.1. Biped Scout by LynxMotion

Figura 2.8: Robot BipedScout

Este robot es vendido entre U$S 700 y U$S 1.100 dependiendo de la configuración, es un desarrollo muy nuevo y la empresa no ofrece aun demasiadas aplicaciones ni opciones. Básicamente el producto es un kit para armar y configurar, distintos sensores pueden opcionalmente ser adosados al robot. Es el usuario quien debe programar el comportamiento del robot así como la interpretación de los datos en caso de agregarle sensores.

Desarrolladores: Lynxmotion

Homepage: http://www.lynxmotion.com

Altura: 35 cm

Peso: 1.1 kg (sin baterías)

Velocidad: No se especifica

Procesamiento: Controladores Basic Atom Pro, ServoPod, u otro provisto por el usuario

DOF: 6 en cada pierna, 12 en total.

Sensores: Opcionales

Actuadores: Servomotores Hitec HS-5475 0.53 Nm (Mínimo) o Hitec HS-5645 1.18 Nm Recomendado

Alimentación: Batería de 6 V a 7.2 V provista por el usuario

Cuadro 2.8: Características del robot BipedScout

2.3.2. Toddler by Parallax

Figura 2.9: Robot Toddler

El precio de este robot es de U$S 249 sin incluir sensores ni baterías. Los sensores deben ser agregados por el usuario, sin embargo su documentación orienta más sobre posibilidades de expansión en este aspecto que la del BipedScout. Toddler tiene únicamente dos grados de libertad, cada uno actuado con un servomotor, y aún así puede caminar hacia adelante, hacia atrás y girar. Uno de los grados de libertad permite levantar un pie del piso, a la vez que inclina al robot hacia el lado opuesto para mantener el equilibrio. El otro grado del libertad es el que simula adelantar una pierna, aunque lo que realmente hace es mover ambas piernas, una hacia adelante y la otra hacia atrás, controlando el ángulo que forman. Para girar el robot mueve las piernas en el movimiento adelante-atrás sin levantar ningún pie del piso y con esto gira, pero depende de la fricción con

el suelo para lograrlo. Es el mecanismo más simple que relevamos que logra caminar levantando alternadamente cada pie, sin embargo sus movimientos están sumamente limitados y no son para nada naturales.

Desarrolladores: Parallax Inc.

Homepage: http://www.parallax.com

Altura: No se especifica

Peso: No se especifica

Velocidad: No se especifica

Procesamiento: Controlador Basic Stamp 2

DOF: 2 en total

Sensores: Opcionales

Actuadores: 2 Mini servomotores tipo hobby de 0.33 Nm

Alimentación: 4 Baterías AA

Cuadro 2.9: Características del robot Toddler

2.3.3. Resumen

Los bípedos ofrecidos en el mercado para proyectos de hobby son aún demasiado elementales. No se encuentra en ellos una alternativa válida para trabajar en robótica bípeda ya que su diseño es rígido y no está orientado a la investigación y desarrollo sino al entretenimiento. Sin embargo el estudio de estos productos sí fue productivo para nuestro proyecto, ya que los mismos se basan en componentes accesibles y relativamente simples con los que podemos trabajar a un costo bastante bajo. Por ejemplo todos estos robots usan para el procesamiento controladores del tipo que estudiaremos en la sección 2.5.5 que son relativamente sencillos y baratos.

2.4. Caminadores pasivos

Los caminadores dinámicos pasivos son una clase de modelos simples que caminan sin actuación (motores) ni control. Tad McGeer ([MG90]) mostró que un modelo plano, simple, de dos piernas, podía caminar sobre una rampa inclinada sin ninguna entrada de energía externa ni entradas de control externas. En la figura 2.10 se muestra un caminador pasivo.

Figura 2.10: Caminador pasivo

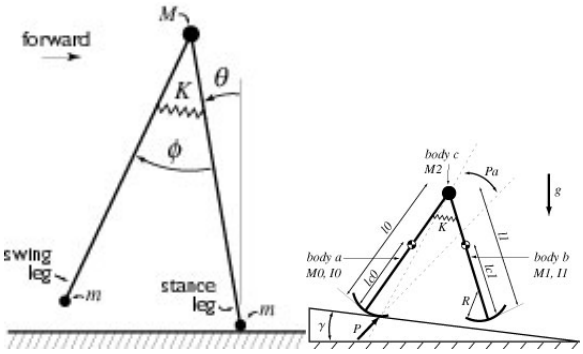


Figura 2.11: Modelo cinemático del caminador pasivo

El funcionamiento del mecanismo es el siguiente: inicialmente se lo coloca en el extremo alto de la rampa, lo que hace que el propio peso del robot ejerza un torque sobre el robot que tiende a hacerlo avanzar. Este torque convierte la energía potencial gravitatoria en energía cinética durante la fase de soporte único. Durante el impacto contra el piso del pie libre, al comienzo de la fase de doble soporte, se pierde una cantidad de energía. Es importante notar que es fundamental el correcto diseño del caminador ya que si no se dimensiona correctamente, el mecanismo simplemente se caerá. Para poder modelar el mecanismo, hay que observar que en un caminador dinámico pasivo, las extremidades actúan como péndulos acoplados: la pierna de apoyo actúa como un péndulo invertido, y la pierna libre como un péndulo libre, adosada a la pierna de apoyo en la cadera. El resultado es un andar muy natural, con una eficiencia energética alta.

Si bien su construcción puede resultar relativamente simple, el desarrollo de estos mecanismos pasivos implica la realización de complejos modelados e intrincados estudios. Con el relevamiento de esta familia de mecanismos móviles aprendimos que si bien la industria está enfocada a robots activos, existe un área de caminadores pasivos, que encuentra asiduos en el campo académico y

del entretenimiento cuando se producen como juguetes. Además, el modelo pendular es útil a la hora de obtener un modelo simplificado de un bípedo.

2.5. Tecnologías

En esta sección se presentan las tecnologías relevadas como predominantes en el área.

2.5.1. Construcción estructural

Los metales, por su alta durabilidad, son los materiales más populares cuando se trata de hacer robots. En particular, el aluminio es muy usado por ser muy liviano y accesible, además de ser fácil de trabajar. Los proyectos que disponen de presupuestos mayores utilizan plásticos o polímeros propietarios cuyas características no se divulgan. El robot ARNE ([IK03]) que vemos en la figura 2.12, utiliza polyamide para la parte principal del cuerpo, y aluminio para el resto. En algunas ocasiones, se utiliza acrílico de alta densidad para partes que estén sujetas a mayores desgastes.

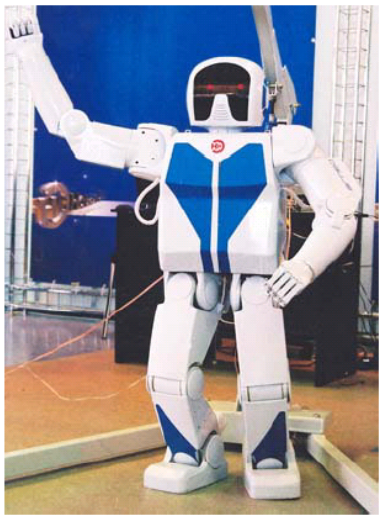


Figura 2.12: El robot ARNE

2.5.2. Sensores

Los sensores son dispositivos que contienen transductores. Los transductores traducen una magnitud de algún tipo (eléctrica, mecánica, etc.) en otra de otro tipo. Los sensores que son de interés para el proyecto son aquellos cuya magnitud de salida es eléctrica. A continuación relevamos las distintas magnitudes y cómo se pueden medir.

2.5.2.1. Velocidad angular

Tacómetro óptico (optical shaft encoder) Consiste en una rueda con ranuras equiespaciadas, que se monta sobre el eje donde se quiere medir la velocidad angular. Además hay un dispositivo emisor de luz que hace pasar un haz por alguna de las ranuras, y un fotorreceptor del

otro lado. Al girar la rueda, el fotorreceptor detecta un haz pulsado a una cierta frecuencia que tiene una relación con la velocidad angular de la rueda. Notar que con un solo par fotoemisor-fotorreceptor se obtiene la velocidad angular en módulo. Para obtener el sentido de rotación se dispone otro par de fotoemisor-fotoreceptor en cuadratura eléctrica. De esta forma se tiene un dispositivo que permite obtener la velocidad en módulo y sentido, viendo la secuencia en la que se activan los dos pares.

2.5.2.2. Posición angular

Potenciómetro La forma más sencilla de realizar un sensor de posición angular es montar un potenciómetro cuya pata de regulación va acoplada mecánicamente al eje del motor. Midiendo la resistencia que exhibe el potenciómetro podemos obtener la posición del motor. Esto tiene la desventaja de que el motor no puede girar continuamente hacia el mismo lado, dado que el potenciómetro tiene una capacidad de giro de alrededor de 180º. Están los potenciómetros de tipo tornillo, pero aún así puede girar un ángulo limitado. Esta medición, una vez calibrado el potenciómetro, es absoluta.

Integración cinemática Otra forma de realizar la medición de la posición angular consiste en integrar la salida de un tacómetro angular. De esta manera obtenemos el desplazamiento angular respecto de cierta posición de referencia (ya que la medición es relativa). Esto muchas veces no es muy útil ya que los pequeños errores, si no se cancelan, al integrarse divergen y hacen que la posición medida difiera seriamente de la real.

Optical Shaft Encoder Una forma de obtener una lectura de posición absoluta, es haciendo n calados de anchos sucesivamente mayores, y disponer n pares fotoemisor-fotoreceptor. De esta manera se tiene una medida instantánea absoluta codificada según cierto código (dependiente de los anchos de las ranuras), de la posición del eje.

2.5.2.3. Presión, fuerza o aceleración

Acelerómetro piezoeléctrico Son dispositivos por lo general caros. Funcionan a través del efecto piezoeléctrico. Cuando se ejerce presión sobre un cristal, aparece una densidad de carga libre sobre las caras del cristal, esto se traduce en una corriente que puede ser transducida en un voltaje. De esta manera, podemos transducir presión en una señal eléctrica.

Se puede usar para las tres magnitudes (fuerza, aceleración y presión) ya que son convertibles entre sí:

Definición de presión: $P = \frac{F}{A}$, por lo que $\int P.\overrightarrow{dA} = \overrightarrow{F}$

Segunda ley de Newton: $\overrightarrow{F} = m\overrightarrow{a}$.

2.5.2.4. Posición y velocidad lineal

Dependiendo de la construcción del mecanismo, se puede construir un transductor de posición o velocidad lineal a partir de uno angular. Para obtener un transductor de posición a partir de uno de velocidad, podemos integrar como en el caso angular, con cuidado de que los errores de

medición se cancelen a la larga, es decir que el error sea de media nula. En caso contrario no se podrá utilizar debido a la divergencia de este error integrado.

2.5.2.5. Inclinación

En esta categoría caen los inclinómetros y giroscopios. Son dispositivos que miden el apartamiento entre dos ejes, uno fijo y uno perteneciente al dispositivo mismo.

Inclinómetro Los inclinómetros más simples pueden hacerse sencillamente, como se describe en [WEEI], donde una caja plástica rellena con un electrolito líquido, es utilizada como un divisor resistivo para medir la inclinación.

Giroscopio Los giroscopios utilizan la precesión para mantener su eje de rotación en una posición fija, más allá de las fuerzas o pares externos que se les aplique. Como el eje de rotación permanece constante, se puede obtener la inclinación del eje del giroscopio midiendo el ángulo entre ambos. Los robots estudiados que utilizan giroscopios son el ASIMO, ARIA, Isaac, Guroo, y Rope.

2.5.2.6. Sensores ópticos

Fotorresistores o LDR (light dependent resistor) Los sensores ópticos varían ampliamente en diseño. Algunos actúan como resistores cuya resistencia depende de la intensidad de la luz que les llega. A estos se les llama fotorresistores.

Fotodiodos - fototransistores Los fotodiodos son diodos activados mediante radiación lumínica, es decir conmutan entre el estado activo (On) y cortado (Off) dependiendo de si la intensidad de la luz recibida supera cierto umbral. En los fototransistores, la luz excita la base generando una corriente que polariza al transistor en zona activa, conduciendo corriente entre colector y emisor (npn) o emisor y colector (pnp). Estos dos tipos de sensores poseen un ancho de banda reducido, es decir, detectan luz de un color único.

Video Otros sensores más complejos, son las cámaras de video. Estas permiten un nuevo nivel de percepción totalmente diferente; la contraparte es que esto no solamente encarece la construcción, sino que complica la algoritmia del procesamiento. Casi todos los robots de mediano y gran presupuesto cuentan con por lo menos una cámara de video.

2.5.3. Actuadores

En esta sección se describen algunas de las tecnologías de actuadores estudiadas ([IGOE]).

2.5.3.1. Actuadores rotatorios

En el área de los actuadores rotatorios, los motores son por lejos los dispositivos más utilizados. En este apartado nos ocuparemos de los tres tipos de motores más utilizados en robótica.

Motores paso a paso (Steppers) Los motores paso a paso se componen por un número de bobinas equiespaciadas angularmente ubicadas en el estator. En el centro de este arreglo se encuentra el rotor. Las bobinas se utilizan como electroimanes; al energizarse en una secuencia circular, los campos magnéticos generados en las bobinas arrastran al eje del motor. Hay varias formas de efectuar esto:

1. Wave Drive - Las bobinas se energizan de a una a la vez. Es el de menor consumo eléctrico.
2. High Torque - Las bobinas se energizan de a pares consecutivos. Al haber dos bobinas energizadas en todo momento, el torque es mayor pues el flujo que atraviesa la armadura es mayor.
3. Half step - En este enfoque se energiza la bobina 1, luego la 1 y 2, luego la 2, etc. La desventaja que posee es que el torque no es constante.

Un punto a tener en cuenta es que si la frecuencia de la secuencia de energización es demasiado alta, puede no moverse, vibrar, o moverse en sentido contrario. A continuación se muestra un diagrama del rotor con los bobinados del estator (ver [JO98])

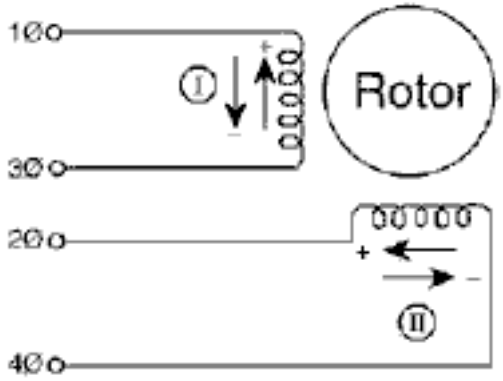


Figura 2.13: Diagrama de un motor paso a paso ([EIO])

Motores de corriente continua (DC) Para hacer funcionar estos motores, se les aplica una tensión en bornes, y a continuación acelerarán hasta girar a una velocidad constante siempre que puedan vencer el par de arranque. La velocidad a la que giran dependerá del par resistivo al que se someta al motor.

Para operar un motor de continua hace falta un circuito "driver", y un controlador. El driver energiza el motor, y el controlador es el dispositivo que comanda al driver. Una forma común de driver es el circuito llamado "puente H". Está compuesto por dos pares de transistores: dos pnp y dos npn. Cada transistor pnp se conecta a un npn a través de sus colectores, los emisores de los npn a tierra y los emisores de los pnp a la fuente de alimentación. Cada borne del motor se conecta con uno de los pares de transistores en el colector. Cuando se activa un transistor pnp de un lado y un npn del otro, el motor gira en un determinado sentido: activando los transistores del lado opuesto se hace girar al motor hacia el otro lado. Si se activan los dos transistores pnp o los

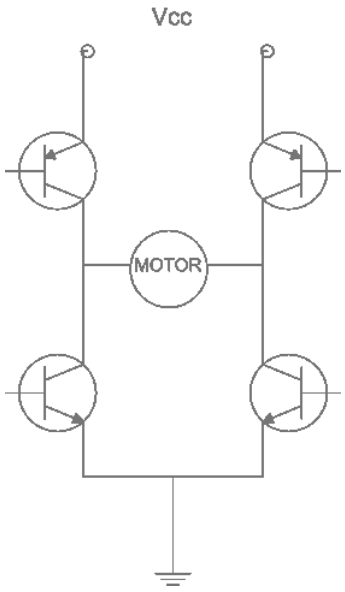


Figura 2.14: Esquemático del Puente H

dos npn, se tiene un freno. Si se activan los transistores npn y pnp del mismo lado, se tiene un cortocircuito, así que este caso debe evitarse a toda costa.

Servomotores El término servomotores se refiere a una familia de motores más que a un tipo específico. Son motores que cuentan con un lazo de realimentación interno. Esto significa que sensan la posición en la que se encuentran, la comparan con la posición a la que se le indica que debe ir, y aplica una tensión que depende de esa diferencia. El caso más común es el servomotor de corriente continua, que consiste en un motor de DC, con un potenciómetro adosado mecánicamente al eje, que traduce el ángulo del motor en un voltaje, y que a su vez este voltaje se realimenta a la entrada negativamente. De esta manera se logra un motor con un lazo de realimentación de posición interno. Los servomotores de DC poseen un conductor de alimentación, uno de tierra y uno de control, donde se le indica a donde debe ir. Si bien la señal de control puede enviarse como una señal analógica, se acostumbra codificar la posición en PWM (Pulse Width Modulation, modulación en el ancho del pulso), con un período de alrededor de 20 ms. Esto es así porque el motor DC verá el valor promedio de la onda (es decir, responde al ciclo de trabajo de la onda). De esta manera no es necesario aplicar niveles de voltajes diferentes para diferentes posiciones, y el motor puede controlarse desde un dispositivo que tenga salida de 0 y 5 V (correspondientes al 0 y 1 lógicos respectivamente). Es así que se cambia la resolución espacial (resolución en el voltaje) por una resolución temporal (en el ciclo de trabajo de una onda de período corto).

Un servomotor presenta varias ventajas respecto de un motor paso a paso. En primer lugar, alcanza una buena resolución de posición con sólo tres hilos, mientras que para un stepper se necesitarían más hilos para una precisión similar. En segundo lugar, en el hilo de control ya va codificada la posición, y el motor se dirige a esa posición, a diferencia del stepper, donde hay que ir conmutando los solenoides para arrastrar el rotor. Por último, la relación par máximo/peso del motor es mucho mejor para el servomotor que para el motor paso a paso. Ver el apéndice F.

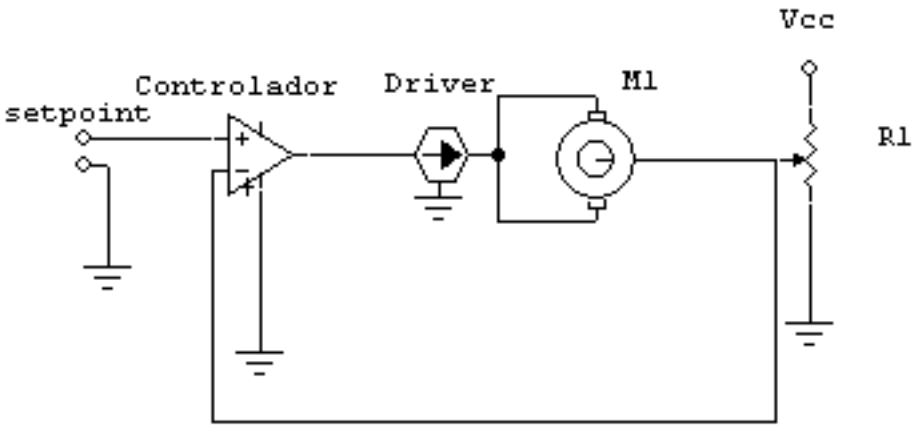


Figura 2.15: Esquema de un servomotor de corriente continua

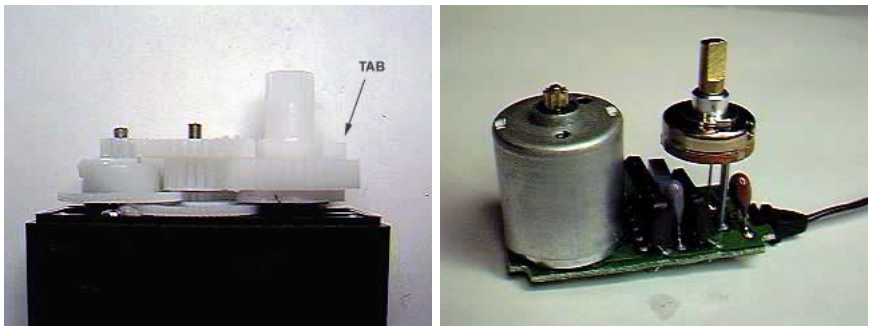


Figura 2.16: Un servomotor por dentro ([HVWT])

2.5.3.2. Mecanismos lineales

Existen diferentes tipos de actuadores que permiten realizar desplazamientos lineales en lugar de rotatorios. A continuación se comentan algunos.

Cinta móvil La forma más sencilla de obtener un movimiento lineal, es enrollar un motor rotatorio en una cinta. Al girar el motor, la cinta se desplaza a una velocidad lineal $v = \omega \cdot r$, donde ω es la velocidad angular del motor y r el radio del motor.

Actuadores lineales paso a paso En estos actuadores, la idea es "desenrollar" el arreglo de bobinas que posee un motor paso a paso y disponerlo a lo largo de la distancia máxima a recorrer. Así se obtiene un mecanismo sencillo que permite desplazar una carga siguiendo una recta.

Móvil rodado A través del cardan y de engranajes, los automóviles convierten el giro del motor en el giro del eje de las ruedas, lo que redunda en el desplazamiento del móvil.

2.5.3.3. Otros tipos de actuadores

En este apartado se comentan algunos tipos de actuadores especiales o más sofisticados.

Músculos artificiales ([WEPA]) Son actuadores cuya finalidad es la de reproducir la respuesta de un músculo, y producir movimiento a partir de la contracción del mismo. En algunos casos la expansión subsecuente puede ser controlada, utilizando esta propiedad para producir movimiento, y en otros no.

Neumáticos Consisten de elementos flexibles cuyo volumen cambia mediante la acción de compresores, que a su vez son actuados por motores.

Nitinol (Nickel Titanium Naval Ordinance Laboratory) Es una familia de materiales intermetálicos con un porcentaje de níquel del 55 %, titanio, y otros metales. Entre las características que estos materiales presentan, se encuentran la memoria de la forma (morfohistéresis) y la superelasticidad transformacional o seudoelasticidad. La memoria se refiere a que el material vuelve a su forma original cuando se le calienta. La seudoelasticidad es un comportamiento elástico, con características parecidas a la goma. Por más información, ver [NITI].

Polímeros Últimamente se vienen desarrollando polímeros en gel que reproducen algunas de las propiedades de los músculos. No tienen histéresis, y pueden llegar a contraerse y expandirse por un factor de 1000. Para que atraviesen estos cambios se debe cambiar una magnitud externa que dependiendo del material específico puede ser la temperatura, el pH, el campo eléctrico, o la composición iónica del solvente. La tasa de contracción (t, tiempo característico de contracción) es proporcional al cuadrado de la mayor dimensión característica (d), es decir $t = c \cdot d^2$ siendo c una constante dependiente de la concentración. A modo de ejemplo, para el polyacrylamide, c vale $2 \cdot 10^9 \frac{s}{m^2}$, y una fibra de 1 cm de diámetro tardaría 2.5 días en contraerse, mientras que un entramado de fibras de $25\,\mu m$ de diámetro se contrae en un segundo.

2.5.3.4. Perfiles de movimiento

En este apartado se describen las distintas formas de conducir un actuador hacia una configuración deseada. Si bien está orientado a actuadores rotatorios, los conceptos pueden aplicarse a cualquier tipo de actuador.

Con el objetivo de mover un actuador hacia una cierta posición, de la mejor manera posible, se estudia cómo se realiza el movimiento en los seres vivos. La primera aproximación al movimiento de una articulación simple como lo es una rodilla es un actuador de un grado de libertad como un motor. Para realizar el movimiento, la opción más fácil sería adoptar una velocidad constante. Esto es lo más adecuado en bajas velocidades, ya que permite coordinar distintas articulaciones con facilidad para que los movimientos de las mismas comiencen y terminen al mismo tiempo. La principal desventaja, y la razón por la cual no debe usarse a grandes velocidades, es la aceleración no nula (en realidad, es muy elevada) al comienzo y al final del movimiento. Vale notar que para que un movimiento se defina como "suave", la derivada tercera de la posición ("jerk") debe ser nula al comienzo y al final.

Definimos un perfil, como la relación que mantiene la posición de una articulación con el tiempo. Los perfiles por lo general presentan tres fases: aceleración, régimen y desaceleración. A continuación se explican algunos de los perfiles estudiados y sus características.

Trapezoidal Las etapa de aceleración y desaceleración presentan una aceleración constante y la de régimen no presenta aceleración. Tiene la ventaja de alcanzar el objetivo en un tiempo bajo, pero presenta tirones en los puntos de transición. Un caso particular, que utilizamos en el Proyecto, es el de velocidad constante. En este caso, las etapas de aceleración y desaceleración tienen una duración infinitesimal.

Acampanado Con esta familia de perfiles se puede lograr un movimiento muy suave; como contrapartida el tiempo necesario para alcanzar el objetivo se incrementa. En esta familia de perfiles podemos encontrar los siguientes:

Cosenoidal: $\omega(t) = \frac{\Delta\theta}{T}\left(1 - \cos\left(\frac{2\pi t}{T}\right)\right)$

Senoidal: $\omega(t) = \frac{\Delta\theta}{T}\sin\left(\frac{\pi t}{T}\right)$

2.5.4. Procesamiento en robots de mediano y alto presupuesto

En estos emprendimientos los elementos de procesamiento por lo general son computadoras, ya sean PC de escritorio, o un sistema multiprocesador RISC de 64 bits. La conclusión a extraer del estudio del estado del arte respecto a los elementos de procesamiento es que estos deben ser lo suficientemente poderosos para ejecutar el software y soportar el conjunto de interfaces necesario, ya que no existe una panacea universal en este ámbito; como corolario decimos también que en caso de tener que realizar una tarea de tiempo real, es común y necesario utilizar ya sea un microcontrolador dedicado, o un PC con un sistema operativo de tiempo real. El procesamiento puede distribuirse en varios elementos de procesamiento. Como se vio en la sección 2.2, los robots ARIA y Rope utilizan procesadores Pentium de propósito general, mientras que otros proyectos utilizan sistemas diseñados a medida para la aplicación.

2.5.5. Controladores de robots de hobby

Los robots diseñados para que el usuario los pueda armar, configurar, programar y modificar suelen contar con una plaqueta de control con características muy similares. Estos robots utilizan por lo general como actuadores, servomotores de hobby y por lo tanto todas estas plaquetas permiten controlar estos servos. Otra característica básica es la capacidad de interconexión con un PC. También muchas de estas plaquetas cuentan con entradas analógicas y digitales aptas para conectar sensores, así como también con salidas digitales.

Los controladores de robots que estudiamos no están diseñados para ningún robot en particular, sino que pueden ser utilizados en cualquier proyecto de robótica y muchos incluso son desarrollados por empresas que no desarrollan robots. Estos controladores están basados en un microcontrolador que resuelve todo el procesamiento. Existen algunos muy simples que sólo controlan servomotores y no cuentan con conversores A/D, incluso los más baratos ni siquiera tienen control de velocidad. Incluimos a continuación detalles de dos de los más completos que se encuentran en el mercado a la fecha de realización de este estudio.

2.5.5.1. Servio

Desarrolladores: PicoBytes Inc.

Homepage: http://www.picobytes.com

Servos que controla: 20

Resolución de posición: 16 bit

Resolución de velocidad: 8 bit

Conversores A/D: 8

E/S digitales: Hasta 30 anulando las otras E/S

Interfaz: RS-232 1.200-115.200 bps

Funcionamiento autónomo: Secuenciador sencillo

Otros: 2 salidas PWM

Precio en origen: U$S 100

Cuadro 2.10: Datos del controlador Servio

2.5.5.2. ASC16

Desarrolladores: Positive Logic Engineering

Homepage: http://www.positivelogic.net

Servos que controla: 16

Resolución de posición: 12 bit

Resolución de velocidad: 8 bit

Conversores A/D: 8

E/S digitales: Hasta 8 entradas anulando los conversores. 8 salidas

Interfaz: RS-232 9.600 bps

Funcionamiento autónomo: no

Otros: Control de aceleración de 8 bits

Precio en origen: U$S 90

Cuadro 2.11: Datos del controlador ASC16

2.5.6. Técnicas de control

A continuación presentaremos algunas formas de controlar sistemas, comenzando con las de más bajo nivel y seguiendo con las más complejas y de alto nivel.

2.5.6.1. Control en lazo abierto y lazo cerrado

En el apéndice C se presenta una introducción a los conceptos básicos de la teoría de control. En este apartado nos referiremos a sistemas deterministas.

Si conocemos cuál es la función del sistema S, y queremos que la salida a una entrada dada u sea $f(u)$, podemos intercalar antes del sistema un bloque cuya función sea $g = S^{-1} \circ f$. Decimos en este caso que el control es en lazo abierto: uno conoce que hace el sistema, sabe que es lo que uno quiere que haga, y basándose en esta información modifica la entrada efectiva al sistema. De esta manera, la salida no tiene, a priori, ningún efecto en la entrada (no hay realimentación), y la efectividad del sistema depende de que no haya perturbaciones externas (de lo contrario la función S que medimos no será exactamente la función del sistema) y la precisión depende de la calibración del sistema (si el sistema se va deteriorando, su función S se verá modificada, y la salida del sistema puede llegar a variar en gran medida).

Si por el contrario, realimentamos la salida a la entrada, mediante algún dispositivo de comparación (como ser un comparador, un sumador u otro dispositivo), tenemos un sistema en lazo cerrado. La entrada al sistema es la composición de una entrada externa y la salida misma del sistema, es decir que la salida influye en sí misma. A este tipo de sistemas se les conoce como sistemas retroalimentados, realimentados, o de lazo cerrado. Cabe notar que si realimentamos un cierto sistema en lazo abierto (es decir, "cerramos el lazo"), el sistema realimentado por lo general tendrá una ganancia menor y una estabilidad mejorada.

2.5.6.2. Cinemática inversa

Con el objetivo de conducir un efector de nuestro robot en una trayectoria específica, es que nos preguntamos cómo deben moverse sus actuadores. En este apartado describimos una técnica conocida como cinemática inversa que responde a esta pregunta.

Introducción El estado de un sistema de cuerpos rígidos con N grados de libertad es descriptible mediante un conjunto de N coordenadas $\overrightarrow{q} = \{q_i\}_{i=1..N}$. Cuando escribimos las N ecuaciones de movimiento, tenemos un sistema de ecuaciones diferenciales de segundo orden, que podemos escribir como $2N$ ecuaciones en $2N$ incógnitas añadiendo una variable nueva por cada grado de libertad para representar la derivada de este grado de libertad. Unificando las variables originales y las añadidas en un vector $\overrightarrow{Q}$ al que llamaremos configuración del sistema, obtenemos que $\dot{\overrightarrow{Q}} = \overrightarrow{F}(\overrightarrow{Q})$. Cuando hablamos del sistema, nos referimos al conjunto formado por el sistema de cuerpos rígidos, las articulaciones y las restricciones existentes.

Descripción del problema La posición de un efector $\overrightarrow{r}$ del sistema viene dada por una función que depende de las coordenadas que describen al sistema:

$$\overrightarrow{r} = \overrightarrow{g}(\overrightarrow{q}) \tag{2.1}$$

Por lo tanto, para cada valor del vector $\overrightarrow{q}$ se deduce la posición del efector $\overrightarrow{r}$ mediante simple aplicación de una función. Esto se conoce como el problema de la cinemática directa.

El problema de la cinemática inversa se define de la siguiente manera: suponiendo conocida la posición de un efector dado del sistema $\overrightarrow{r}$, deducir cuál es el valor del vector $\overrightarrow{q}$. Es decir, deducir

la configuración del sistema a partir de un efector del mismo. Las coordenadas de cualquier punto solidario al sistema, como lo es un efector (ya que pertenece al sistema) se pueden poner en función de las coordenadas generalizadas del mismo, y por eso escribimos: $\overrightarrow{r} = g(\overrightarrow{q})$.

El problema IK se reduce entonces a:

$$\overrightarrow{q} \quad = \quad g^{-1}(\overrightarrow{r}) \tag{2.2}$$

o lo que es lo mismo, a encontrar la expresión de $g^{-1}(\cdot)$. La determinación de la inversa de g no es un problema trivial. Para empezar, puede no haber inversa (es decir, puede no estar definida por ser g no inyectiva...). Por otro lado, si g no es sobreyectiva, existirá un conjunto de puntos a los cuales no será posible llegar (llamado conjunto de las configuraciones no alcanzables). Asimismo, la inversa de g debe ser inyectiva para que g esté bien definida como función.

Es importante recalcar el hecho de que para un punto $\vec{r}$, puede no existir un valor de $\vec{q}$ (decimos que $\vec{r}$ es inalcanzable), existir uno (g es inyectiva, cosa que rara vez sucede), o existir más de un valor de $\vec{q}$ (g no es inyectiva).

De esta manera, existen herramientas (como veremos en la subsección 2.5.7, y también integradas en los robots de mayor presupuesto) que persiguen la generación de determinadas trayectorias de efectores del sistema, y esto lo logran llevando sus actuadores a las posiciones que se calculan resolviendo el problema IK. En el apéndice E se amplía.

2.5.6.3. Modelo dinámico

Desafortunadamente, la cinemática inversa es un modelo en el cual no se tienen en consideración las leyes de la mecánica. Esto significa que no se toman en cuenta las fuerzas y momentos que aparecen en la realidad como resultado de contactos, campos, etc. En el problema de la dinámica, obtenemos las aceleraciones de los cuerpos como resultado de las fuerzas y momentos que actúan sobre ellos. Esto lo hacemos utilizando la segunda Ley de Newton $\vec{F} = m\ddot{\vec{x}}$. El problema de la dinámica inversa es, conociendo la aceleración deseada como función del tiempo, deducir las fuerzas y pares que deben aplicarse externamente para cumplir con dicha aceleración. Si el objetivo principal es calcular la configuración dada para una posición del efector, y bajo el supuesto de utilizar bajas velocidades, se puede utilizar el modelo cinemático. Si por el contrario las fuerzas, torques y velocidades que aparecen son mayores, el modelo cinemático no es válido y es necesario utilizar el modelo dinámico. La diferencia operacional entre dinámica inversa y cinemática inversa es que en la cinemática se tiene un problema "estático", en el sentido de que la ecuación $\overrightarrow{r} = \overrightarrow{g}(\overrightarrow{q})$ es algebraica. En el caso de la dinámica, intervienen no sólo las posiciones de los actuadores sino también sus trayectorias (velocidades, aceleraciones, que vienen influenciadas por las fuerzas y momentos), por lo que en este caso se tiene una ecuación diferencial $\ddot{\vec{r}} = \overrightarrow{g}(\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}}) = \vec{g}\left(\vec{Q}, \dot{\vec{Q}}\right)$.

En los robots estudiados, se utilizan una amplia gama de sensores: de inclinación, de presión, de fuerza, acelerómetros y demás, para poder calcular las fuerzas generalizadas que están ejerciéndose así como la ubicación del COG (centro de gravedad) o el ZMP y entonces poder deducir las fuerzas generalizadas a aplicar para compensar.

2.5.6.4. Gait generation([MS01, WT92])

Gait Generation es la formulación y selección de una secuencia de movimientos coordinados de las distintas partes de un multípodo que resultan en que el mismo avanza recorriendo una cierta trayectoria dada. Básicamente existen cuatro formas en uso para la generación de un andar:

Control Caminar o correr en sistemas multípodos puede modelarse utilizando retroalimentación. Las leyes de control generan el andar como una propiedad inherente al sistema al definir una retroalimentación estable que conduce al cuerpo y piernas alternadamente. Este enfoque es utilizado ampliamente pues permite andares más rápidos.

Conductista El enfoque conductista embebe la generación del andar en una red de control, que se diseña inspirada en el sistema neurológico de algún animal o en una abstracción de una jerarquía de comportamiento. Beer y Chiel ([BC92]) desarrollaron una versión simplificada del sistema nervioso de una cucaracha. Enfocándose en el sistema motriz, crearon una red neuronal artificial heterogénea en la cual las células nerviosas pueden excitar o inhibir los movimientos de las piernas y coordinar el movimiento del cuerpo.

Basado en reglas En los enfoques orientados al control y conductista no hay una planificación explícita, ya que el robot presenta un movimiento con un patrón bien definido. El enfoque basado en reglas, utiliza un conjunto de reglas y heurísticas para planificar un andar.

Basado en restricciones El enfoque basado en restricciones utiliza una varios factores, como son la cinemática, el terreno y la estabilidad para restringir el rango de movimientos posibles y luego ordenar el resto de los movimientos para poder utilizar búsquedas u optimizaciones para encontrar el andar óptimo según cierto criterio. Un ejemplo de esto muy utilizado es la integración de la cinemática inversa, el modelo dinámico, las restricciones de estabilidad (ya sea estática o dinámica) y la optimización en términos de velocidad o eficiencia en el uso de la energía.

2.5.6.5. Técnicas de Aprendizaje e Inteligencia Artificial ([AIDE])

En este apartado comentamos una descripción de algunas de las técnicas de la rama de la Inteligencia Artificial.

Reinforcement Learning El aprendizaje por refuerzo es un método que selecciona para cada situación, por medio de ensayo y error, la acción que resulta en el mayor beneficio a largo plazo. En la aplicación de esta técnica existen tres elementos: el ambiente donde se desarrollan las acciones, el agente que ejecuta las acciones, y la señal de recompensa que provee la información acerca del avance del educando. Esta técnica se utiliza por ejemplo para enseñar a un robot a caminar, dependiendo del tipo de terreno (ver el apartado 2.5.6.4) . En este caso el ambiente es el terreno, el agente es el robot y la señal contendrá información acerca del avance del robot.

Lógica difusa La lógica difusa (Fuzzy logic) es un superconjunto de la lógica convencional (booleana) que se extendió para manejar el concepto de la verdad a medias, es decir valuaciones entre "completamente verdadero" y "completamente falso". Es útil en problemas de toma de decisiones.

Redes neuronales Las redes neuronales artificiales (o neuromiméticas) son modelos matemáticos de neuronas y estructuras neuronales muy sencillas, con implementaciones electrónicas o simuladas en software. Se utilizan para resolver problemas de reconocimiento de patrones y toma de decisiones por adiestramiento.

Algoritmos genéticos Son algoritmos de optimización que se basan en el principio darwiniano de la evolución ("la evolución del más apto"). Se trata de algoritmos iterativos, que mantienen un conjunto de estructuras que representan posibles soluciones al problema que se quiere resolver. Se tiene además, una función de "aptitud" o fitness, que mide qué tan buena es una solución candidata. La idea detrás de esta familia de algoritmos es combinar candidatos de una generación utilizando ciertos operadores (cruza, mutación, etc.) de forma de maximizar la aptitud de los candidatos hasta llegar a una solución aceptable. Principalmente se utiliza en problemas de optimización y búsqueda.

2.5.6.6. Control de alto nivel

A continuación comentaremos algunas de las disciplinas que conforman el grupo de objetivos de alto nivel que frecuentemente aparecen en robótica.

Planificación de trayectorias ([AIDE]) Entre las disciplinas de alto nivel encontramos el problema clásico de, dado el mapa del terreno, encontrar una trayectoria que una dos puntos dados del mismo. Existen una variedad de soluciones a dicho problema: Algoritmo A*, Campos de potencial, RRT (Rapidly-exploring Random Trees), etc.

Tareas estáticas En esta categoría entran por ejemplo, jugar al ajedrez u otros juegos de mesa, e incluso los sistemas de asistencia como los que podemos encontrar en algunos centros comerciales, y los sistemas de interfaz hombre-máquina. Se caracterizan por tener un dominio estático (es decir, bien definido y que no cambia con el correr del tiempo), las entradas llegan a intervalos discretos relativamente grandes, y por lo general son entradas simbólicas (es decir que están codificadas a priori, o pueden ser decodificadas de manera relativamente simple, por ejemplo utilizando reconocimiento del habla, u otras técnicas).

Tareas en equipo y dinámicas Entre las opciones más populares se encuentran jugar al fútbol, al hockey u otros deportes robóticos. Especialmente en el fútbol robótico es donde se encuentra la comunidad más grande y activa de entusiastas, existiendo diferentes categorías en diferentes copas y torneos ([RC05, FIRA]). Son tareas en extremo complejas, pues integran grandes campos que incluyen la visión artificial, cooperación y competencia en entornos multiagentes, control de bajo nivel, problemas de tiempo real, toma de decisiones y planificación, representación del conocimiento, entre otros.

2.5.7. Simuladores

Los simuladores de sistemas de cuerpos rígidos sirven para modelar físicamente los robots y así poder experimentar sobre ellos sin construirlos. Mediante la simulación se puede por ejemplo

evaluar la estabilidad de un robot y la reacción a distintas fuerzas aplicadas sobre él. Se puede usar también el simulador para probar los algoritmos de control, aunque debido a las simplificaciones del modelo, esto no garantiza el funcionamiento correcto en el sistema real. Para que sea de utilidad, un simulador debe modelar con la mayor exactitud posible la física, pero existe un compromiso entre lo detallado del modelo y la cantidad de cómputo necesaria para implementarlo. Con el propósito de su uso en este Proyecto, se evaluaron los distintos simuladores que se presentan a continuación.

2.5.7.1. DynaMechs

DynaMechs ([MM95]) es una biblioteca open source orientada a objetos que provee simulación dinámica de robots con estructuras complejas; también modela fuerzas hidrodinámicas para la construcción de robots submarinos. Los cálculos dinámicos están basados en algoritmos recursivos eficientes y se provee visualización gráfica de los robots en el ambiente OpenGL. La página web de esta biblioteca es http://dynamechs.sourceforge.net

Para nosotros una ventaja de DynaMechs es la referencia de haber sido utilizada con éxito como núcleo para la construcción de un simulador del robot bípedo GuRoo. Dicho simulador tiene la misma interfaz que el robot y permite que se conecte el software de control indistintamente al robot o al simulador. Además de la simulación física, el simulador de GuRoo simula los motores y sus drivers, sensores y la red CAN y procesadores que implementan el control descentralizado. Con un paso de $500\,\mu s$ y actualización gráfica cada $5\,ms$, se logra una velocidad de 40 % de tiempo real. Otra ventaja de Dynamechs es que es "Open Source", por lo que puede ser modificada o mejorada según la necesidad. Sin embargo una gran desventaja es que la biblioteca se encuentra bastante en desuso. La última versión liberada es del año 2001, y no hay una comunidad activa a la que recurrir; incluso compilar la biblioteca no es sencillo dado que utiliza componentes antiguos de software.

2.5.7.2. RobotBuilder - RobotModeler

Estas herramientas, desarrolladas en Ohio State University, utilizan DynaMechs para la simulación física. Su utilidad es simplemente facilitar el uso de la biblioteca para probar diseños de robots y sistemas de control. Al igual que DynaMechs el proyecto no está muy activo actualmente. La única plataforma que soporta es Windows y no se provee el código fuente. El control del robot se hace mediante el desarrollo de una DLL de control que interactúa con la simulación. Si bien las herramientas son interesantes, al no disponerse del código fuente, el riesgo de que no se adapten a las necesidades del proyecto tiene consecuencias muy graves.

2.5.7.3. Robotics Toolbox for MATLAB

Robotics Toolbox for MATLAB es una biblioteca de dominio público que provee funciones para cálculos de cinemática, dinámica y trayectorias de robots. Provee primitivas para la construcción de robots mediante segmentos y articulaciones, pero no provee manejo de geometrías por lo que no permite modelar contacto y colisiones.

2.5.7.4. BioRobotic

Este software es "Open Source". El simulador viene como parte de un lenguaje interpretado llamado Config++, en una biblioteca llamada Biobot. Tiene limitaciones en el manejo de colisiones y el contacto entre dos objetos está representado únicamente por un punto. Según la documentación de la biblioteca, la misma no está desarrollada haciendo hincapié en la exactitud ni en la eficiencia de los algoritmos, sino que se provee como una herramienta de prototipado rápido. La última versión del software disponible es la 0.4, de enero de 2000. No se ha encontrado una comunidad activa de usuarios y la documentación no es detallada.

2.5.7.5. Yobotics

Yobotics es un software de simulación para robots desarrollado íntegramente en Java. Es comercial y la licencia para uso educativo tiene un costo de U$S 995. Requiere para su uso de JBuilder y Java3D que son de uso libre para fines educativos. Esta herramienta simula la dinámica de cuerpos rígidos y brinda una funcionalidad muy completa.

2.5.7.6. ODE

ODE (Open Dynamics Engine, [ODE]) es una biblioteca de simulación para dinámica de cuerpos rígidos. Es "Open Source" y hay una comunidad activa de usuarios. La última versión disponible es la de mayo de 2004. La biblioteca provee cuerpos básicos (cajas, cilindros y esferas) y articulaciones de varios tipos. Esta herramienta está desarrollada en C++ pero la interfaz para su uso está en lenguaje C. ODE también provee un sistema de colisiones básico, que puede ser utilizado o reemplazado por uno desarrollado por el usuario. La simulación de sistemas con muchos grados de libertad es bastante lenta y si bien provee métodos más rápidos de cálculo los mismos no son adecuados para modelar sistemas de cuerpos con estructura del tipo de la que tienen los robots con patas. De todas formas la velocidad es suficiente para experimentar siempre y cuando no se deseen utilizar técnicas de aprendizaje donde se necesiten muchas ejecuciones. La simulación utilizada en el desarrollo del robot Isaac fue realizada con esta herramienta.

2.5.7.7. Juice

Este simulador ([JUIC]) corre únicamente en plataformas Windows. Es de uso libre pero el código fuente no está disponible. El simulador se basa en ODE para el modelado físico. Si bien este software facilita la creación de robots, brindando una interfaz gráfica para la construcción, no aporta funcionalidades para la simulación mas allá de las ya disponibles en ODE.

Capítulo 3

Enfoque del Proyecto

3.1. Alcance

De acuerdo a las características del Proyecto de Grado en las que se enmarca este trabajo, se estableció un alcance adecuado tanto académicamente como desde el punto de vista del presupuesto disponible. Se desarrolló por lo tanto un robot bípedo simple, que camina utilizando un enfoque sencillo. Todos los materiales utilizados son de muy bajo costo y se consiguen fácilmente.

3.2. Estrategia para caminar

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, elegimos 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 lo hacemos 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 3.4.2.1, intercalando suficientes posiciones intermedias, muy cercanas unas de otras, se puede llevar al robot por una trayectoria arbitraria.

3.2.1. Definiciones preliminares

El criterio de estabilidad estático (es decir en el que no se consideran fuerzas o torques, o lo que es lo mismo, en el modelo a bajas velocidades) es que la proyección del COG caiga dentro del polígono de sustentación. El polígono de sustentación es la curva convexa delimitada por los pies que se encuentran en contacto con el piso.

Para la estabilidad dinámica, el criterio es que la proyección del ZMP caiga dentro del polígono de sustentación.

Observando la interfase entre los pies y la superficie de contacto, vemos que existen dos tipos de fuerzas: las normales f_{n_i} que son perpendiculares, y las de fricción f_{t_i} que son siempre tangenciales. La posición del COP (centro de presión) de un pie se define como $P = \frac{\sum q_i f_{n_i}}{\sum f_{n_i}}$, donde q_i es el vector hacia el punto de aplicación de la fuerza f_{n_i}. Durante la fase de soporte único, el ZMP coincide aproximadamente con el COP. Ver [FM04].

3.2.2. El equilibrio

Si quisiéramos dar un paso hacia adelante con la pierna derecha, y con el objetivo de no perder el equilibrio, moveríamos el peso del cuerpo hacia la izquierda. De este modo la pierna izquierda se convierta en el apoyo del cuerpo y podamos levantar la derecha sin caernos. Si no hiciéramos esto, simplemente nos caeríamos. Esto se debe a que el centro de gravedad del cuerpo, cuya proyección sobre el piso se encuentra en el punto medio entre los pies, se encontraba en el interior del polígono de sustentación hasta que levantamos el pie derecho. Luego de levantarlo, el polígono de sustentación queda limitado a la superficie de apoyo del pie izquierdo. Como la proyección del centro de gravedad no cae dentro de esta región, existe un momento angular neto de nuestro cuerpo, lo que implica que existe una velocidad angular no nula, y nos caemos.

Si por el contrario, movemos el centro de gravedad hacia la pierna izquierda antes de levantar el pie, la situación es muy diferente, ya que la proyección del centro de gravedad estará un poco a la izquierda del centro del polígono de sustentación, y dentro de la superficie de apoyo del pie izquierdo. Cuando levantemos el pie derecho, la proyección del centro de gravedad estará dentro del nuevo polígono de sustentación y no perderemos el equilibrio.

3.3. Arquitectura de la solución

A la hora de estructurar la solución, se presentaron varias preguntas que requirieron tomar decisiones para poder ser contestadas. La primera pregunta que surge lógicamente es: ¿cómo está formado un robot bípedo? Por lo estudiado en el Estado del Arte (capítulo 2), pudimos extraer algunas conclusiones que nos ayudaron a responder nuestra pregunta.

Por un lado, todo robot móvil posee un conjunto de piezas mecánicas que interactúan a través de algún mecanismo actuador con el objetivo de moverse. Por otro lado, estos movimientos deben controlarse, coordinarse y ser monitoreados por algún agente inteligente. Además, los robots por lo general son capaces de tomar información de su entorno, o de sí mismos, analizarla y reaccionar frente a ella de alguna manera.

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. La división en estos dos niveles de funciones en módulos nos permitió concentrarnos en el sistema motriz de bajo nivel al comenzar el proyecto, y una vez implementado este, pasamos a desarrollar la lógica que provee las habilidades motrices de más alto nivel. Para realizar la comunicación entre estos dos nodos, se deberá contar con algún mecanismo de comunicación de datos estándar.

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 ([DU02]). 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 (capítulo 8). 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, vemos en la figura 3.1 que el nodo PC enviará señales de comando (mover el motor i a la posición Θ_i con velocidad Ω_i, encender o apagar una salida lógica, etc.), y recibirá la medición de las posiciones de los motores. En el capítulo 7 veremos que además de esta información, se notificará si cada motor ha llegado a su destino, así como tambien un bit de reconocimiento del último paquete enviado (ACK).

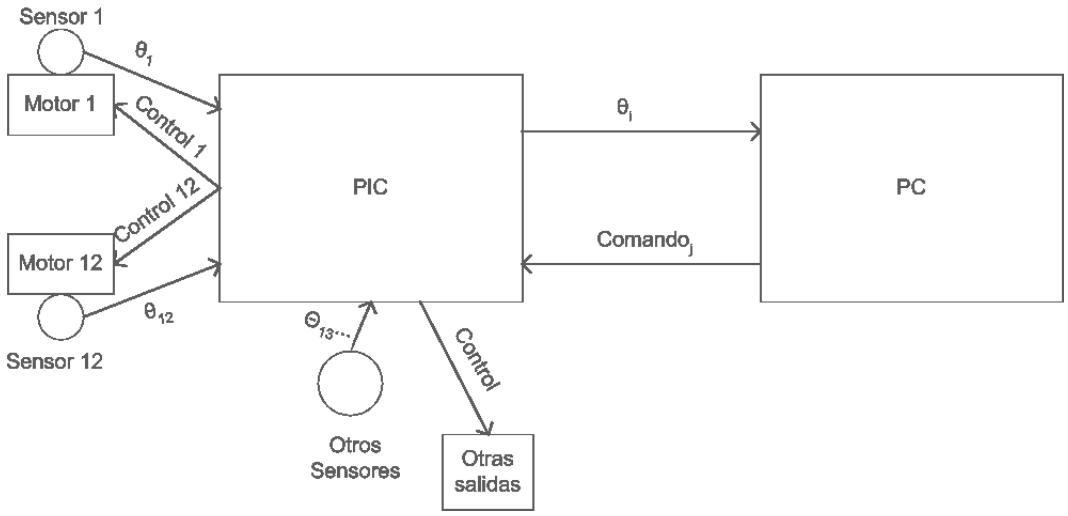


Figura 3.1: Diagrama de la arquitectura de alto nivel. Se muestra el flujo de información a nivel lógico.

3.4. Decisiones de diseño

A continuación se presentan algunas de las decisiones de diseño más importantes que se tomaron.

3.4.1. Alimentación

Basándonos en el estudio del Estado del Arte, consideramos que construir un robot bípedo autónomo (es decir, que no necesite de un cable de alimentación externa), implicaría que debe alimentarse de una cantidad de baterías relativamente grande. Esto no solamente es costoso, sino

que complicaba el diseño porque en este caso el robot debía cargar el peso de las baterías además del suyo propio, y al tratarse de un prototipo experimental, no se justificaba. Es por eso que decidimos que alimentaríamos el robot desde una fuente fija.

3.4.2. Actuadores

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 3.4.3). 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. Sus especificaciones se dan en el apéndice D.1. 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. Esto simplifica mucho su manejo a nivel del PIC.

3.4.2.1. 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 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 COG caiga en el polígono de sustentación, el robot permanecerá en equilibrio estable. Vemos que sin embargo, este enfoque es el menos eficiente desde el punto de vista energético:

$$E = \int_0^T \tau \cdot \omega \, dt = \int_0^T \tau_0 \left(1 - \frac{\omega}{\omega_0}\right) \omega \, dt = \tau_0 \left(1 - \frac{\omega}{\omega_0}\right) \omega \cdot T = \tau_0 \left(1 - \frac{\omega}{\omega_0}\right) \theta$$

donde E es la energía, τ es el momento en el eje, ω la velocidad de rotación, τ_0 el par máximo, ω_0 la velocidad máxima y θ el ángulo a recorrer. De esta forma, vemos que la mayor energía se da a bajas velocidades, y la menor energía a velocidades altas (cercanas a la máxima ω_0).

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. Esto significa que si tenemos dos motores cuyos desplazamientos son $\Delta\theta_1$ y $\Delta\theta_2$, con $\Delta\theta_1 = 2 \cdot \Delta\theta_2$, entonces se verificará que $\omega_1 = 2 \cdot \omega_2$. Cuando a un servomotor se le indica que debe ir a una posición θ, la velocidad con que el motor se mueve no es constante (puede verse la deducción de esta afirmación en los apéndices F.1 y F.2). Es por ello que recurrimos a una técnica que permite obtener una velocidad constante, denominada Ramping (ver apéndice F.3).

Al aspirar a obtener una velocidad constante por motor, estamos utilizando un perfil de actuación constante. Este perfil es una degeneración del perfil trapezoidal en el cual las fases de aceleración y desaceleración tienen una duración infinitesimal.

La forma de calcular la velocidad de un actuador es paramétrica en una velocidad máxima, a la que llamaremos ω_{max}. El procedimiento es el que sigue:

Sean θ_i, con $i = 1..12$, las articulaciones, y sea r el índice tal que $r = \arg \max_{i=1..12} \{|\theta_i^* - \theta_i|\}$, es decir el índice de la articulación que debe recorrer el mayor ángulo, donde θ_i^* es la posición deseada en la articulación i-ésima.

Las velocidades ω_i serán entonces, $\omega_i \equiv \frac{\theta_i^* - \theta_i}{|\theta_r^* - \theta_r|}\omega_{max}$. Esto hace que los tiempos que tardan las articulaciones en llegar a destino son todos iguales ($\tau_i = \frac{\theta_i^* - \theta_i}{\omega_i} = \frac{|\theta_r^* - \theta_r|}{\omega_{max}}$, que es independiente del número de motor).

3.4.3. Morfología y grados de libertad

Ya que el Proyecto propone la construcción de 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 necesitamos 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. Debido a que es deseable que el sistema cinemático sea compatible determinado, elegimos 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 (canilla). El tobillo cuenta con dos grados de libertad: 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.

3.4.3.1. 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 IK 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 nuestro robot no son posibles, pero que no son imprescindibles para caminar.

3.4.3.2. Tipos de articulaciones e implementaciones

Las articulaciones en el reino animal son del tipo "bola y cavidad", rotativa, y charnela. La primera permite los movimiento en los tres ejes (hombro, cadera), la segunda en dos ejes (articulación radio-cubital) y la última 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 nuestro robot, los ejes de los DOF correspondientes a articulaciones de más de un DOF, deben intersecarse 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 (es decir, una cadena de segmentos unidos por las articulaciones perpendiculares), 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 3.2: Esquema mecánico del prototipo

3.4.4. Construcción estructural

Como se trata de un prototipo, se supuso que iba a ser necesario modificar aspectos mecánicos a lo largo del Proyecto. Un material que permite cumplir este requerimiento es el aluminio. Su maleabilidad, bajo costo y alta disponibilidad fueron factores que justificaron la elección del mismo para la realización de los segmentos que unen las articulaciones.

3.4.5. 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 (capítulo 7). Este conjunto de elementos (robot, plaqueta controladora y software embebido) compone el nodo PIC.

3.4.5.1. Nodo PIC

Decidimos desarrollar nuestra propia placa controladora en lugar de utilizar una como las de la subsección 2.5.5. Esta decisión estuvo respaldada por dos razones. La primera razón es que los controladores estudiados en el Estado del Arte 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 8 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. Como se verá en la sección 3.3, 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.

Las opciones que se nos presentaban a la hora de elegir qué clase de elemento de procesamiento utilizaríamos en el nodo PIC, eran básicamente dos: un PC o un microcontrolador. 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 ([MICR]). Los requerimientos que determinaron la elección se resumen a continuación:

1. Velocidad de clock suficientemente alta como para poder cumplir con requerimientos de tiempo real.
2. 12 generadores de onda PWM o bien suficientes salidas digitales que permitan generar las ondas. Esto es necesario para comandar los motores.
3. 12 entradas analógicas (a través de conversores A/D). Esto es necesario para poder leer la posición de los motores.
4. Empaquetado DIP para simplificar el montaje en Protoboards y circuitos impresos.
5. Comunicación hacia afuera según algún protocolo estándar de forma simple.
6. Varios timers, ya que se trata de un sistema de tiempo real que debe poder interactuar con el mundo exterior (comunicarse, manejar motores, etc.)
7. 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)
8. 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 hubo que tomar en cuenta tanto en la elección del microcontrolador como en la programación (ver capítulo 7) son:

1. Velocidad del clock de 40 MHz a través de un cristal de 10 MHz. Los 40 MHz se logran con un circuito PLL interno al PIC.
2. 13 conversores A/D de 10 bits
3. 4 timers
4. Hasta 32 puertos de E/S (la cantidad y el tipo de los mismos es configurable)

5. El PIC implementa el protocolo RS-232 a través de un puerto direccionable USART.
6. Existencia de modos de ahorro de energía. En esta aplicación no tiene utilidad.
7. El PIC posee un bus I^2C. Si bien en este Proyecto no lo utilizaremos, es un ventaja ya que es una forma de expansión posible.

3.4.5.2. Nodo PC

Los requerimientos que determinaron la elección de la estrtucturación 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.

3.4.6. Comunicación entre nodos

La decisión de utilizar una interfaz serial RS-232 viene a raíz de lo visto en el estudio del Estado del Arte relativo a microcontroladores: muchos de ellos vienen con el protocolo RS-232 integrado en el microcontrolador, otros proveen un puerto de comunicación inalámbrica, pero son los menos. En la sección 7.2 se define la estructura de los mensajes.

3.5. Artefactos producidos

En el Proyecto se produjeron los siguientes artefactos:

- Simulación (capítulo 4)
- Prototipo de robot
  - Piezas mecánicas (capítulo 5)
  - Fuente de alimentación (capítulo 6)
  - Plaqueta de control embebido (capítulo 6)
  - Software de control embebido (capítulo 7)
- Sofware de comunicación y control para PC (capítulo 8)
- Sitio web de presentación y difusión del proyecto ([LS04W])
- Hoja de datos del prototipo ([LS04D])
- Manual del software del PC ([LS04M])
- Protocolo de comunicación ([LS04P])
- Informe final (este documento)

Capítulo 4

El Simulador

4.1. Objetivos

Como se explicó en la subsección 2.5.7 el objetivo de la simulación es poder experimentar sin necesidad de contar con el robot construido, de esta forma se pueden desarrollar y validar técnicas que luego serán transferidas al sistema real. Para poder validar nuestra estrategia para caminar desarrollamos una simulación del robot y una herramienta que nos permitiera diseñar posiciones para el mismo.

4.2. Herramienta de simulación

Luego de evaluar las herramientas expuestas en la sección 2.5.7, decidimos utilizar ODE. De las herramientas evaluadas ODE 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.

4.3. Diseño del robot

Como se vio en la subsección 3.4.3, la configuración del robot es de 7 segmentos unidos por 6 articulaciones. Para poder realizar una simulación más sencilla se modeló cada segmento como un cuerpo simple con masa uniforme. A continuación vemos una imagen del robot simulado:

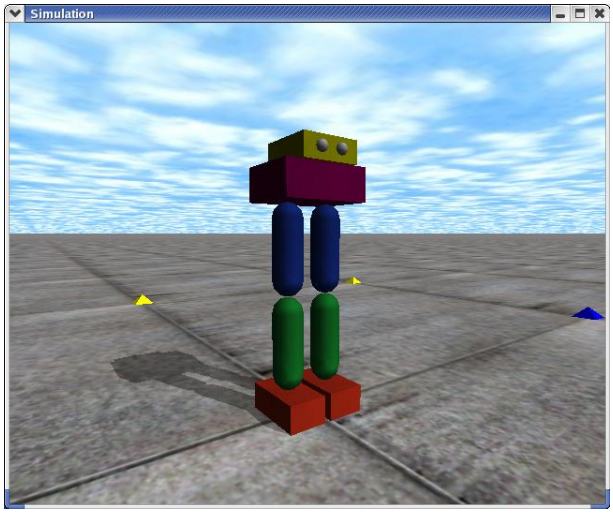


Figura 4.1: Robot Simulado

Dimensiones del pie (mm): 50 x 90 x 35

Masa del pie (g): 60

Dimensiones de la canilla (mm): 80 de largo, 20 de radio

Masa de la canilla (g): 50

Dimensiones del muslo (mm): 80 de largo, 20 de radio

Masa del muslo (g): 90

Dimensiones del cuerpo (mm): 120 x 90 x 75

Masa del cuerpo (g): 350

Altura Total (mm): 270

Masa Total (g): 750

Cuadro 4.1: Características del Robot Simulado

4.3.1. Características y limitaciones del modelo

La simulación se basa en un modelo que contiene muchas simplificaciones de la realidad. Estas simplificaciones deben ser tenidas en cuenta al momento de analizar los resultados y evaluar qué tan parecido puede ser el comportamiento del sistema real. A continuación vemos las aproximaciones que tuvimos en cuenta, muchas de ellas son debidas a limitaciones en la herramienta de simulación ([ODEU]).

4.3.1.1. Materiales

Las deformaciones de los materiales no están tenidas en cuenta en la simulación ya que ODE simula únicamente dinámica de cuerpos rígidos. Los efectos de la deformación de los materiales,

junto con el juego de los engranajes y los errores de construcción constituyen una fuente de error que aleja a la simulación de la realidad.

4.3.1.2. Integración

La herramienta de simulación resuelve las ecuaciones físicas mediante métodos numéricos. Para ello cuenta con un integrador lineal, lo que hace que sea necesario utilizar pasos muy pequeños para la simulación y que aun así la misma no sea muy exacta. Nuestra simulación utiliza pasos de una centésima de segundo. Estos pasos pequeños junto con las velocidades bajas a las que trabajamos reducen el error cometido.

4.3.1.3. Distribución de masa

Si bien es posible en principio aproximar mucho la distribución de masa en la simulación esto presenta dos problemas. Calcular las distribuciones de masa de las piezas relativamente complejas del robot es bastante trabajoso y luego cada detalle que se le agrega al modelo hace más pesados los cálculos y por lo tanto enlentece la simulación. Es por esto que nuestra simulación supone distribución de masa uniforme en cada una de las partes que conforman el robot. Esto afecta la posición del centro de masa del robot así como las características dinámicas del mismo.

4.3.1.4. Modelado del contacto

El contacto entre el pie y el piso es fundamental para la simulación y no es simple de modelar. ODE utiliza un modelo muy simplificado para la fricción. La fricción estática afortunadamente no presenta tantas complicaciones y es lo más importante al momento de caminar. Si el robot viola las restricciones necesarias al realizar movimientos con ambos pies en el piso al menos un pie puede necesitar resbalar para completar el movimiento, en este caso el modelo de fricción dinámica entra en juego y debe ser tomada en cuenta la limitación del modelo en este aspecto. Otras situaciones donde el robot resbale no son importantes si sólo vamos a modelar un desplazamiento caminando lentamente. Se experimentó con los parámetros de fricción de la simulación como parte del estudio.

La otra parte importante del contacto es el impacto del pie con el piso y cómo son transmitidas y absorbidas las fuerzas que éste genera. En este aspecto se utilizaron en un principio los parámetros por defecto de la herramienta de simulación para luego experimentar con ellos. Como el simulador utiliza un modelo de masa-amortiguador-resorte para modelar los contactos, los parámetros en ambos casos se fueron modificando hasta lograr un modelado del contacto "bien comportado", es decir que por ejemplo cuando el pie hace contacto con el piso, el piso no se hunda, el pie no rebote, que el pie no quede oscilando, etc.

Parámetros que modelan el contacto En ODE existen dos parámetros para modelar contactos y articulaciones: el primero es el ERP (Error reduction parameter), que permite aplicar fuerzas para corregir errores en las articulaciones. Los errores en las articulaciones son básicamente discordancias en la posición relativa de los dos cuerpos unidos por una articulación, por ejemplo, en una articulación de bola y cavidad, donde la bola y la cavidad no están alineadas. El segundo parámetro es el CFM (Constraint force mixing). Este parámetro controla qué tan severa

es una violación de una articulación. La restricción para cada articulación tiene la siguiente forma: $J \cdot v = c$, donde J es una matriz jacobiana con una fila por cada grado de libertad que la articulación remueve del sistema, v es un vector de velocidad de los cuerpos involucrados en la articulación y c es un vector. En el siguiente paso discreto de tiempo, se calcula un vector λ tal que las fuerzas a aplicar para preservar las restricciones cumplen $F = J^t \cdot \lambda$. En ODE, en lugar de usar la primera de estas ecuaciones, se utiliza $J \cdot v = c + CFM \cdot \lambda$, donde CFM es una matriz diagonal. El CFM mezcla la fuerza restauradora de la restricción con la restricción que produce esta fuerza. El comportamiento conjunto del ERP y el CFM es el de un sistema de una masa con un sistema de resorte-amortiguador. Si se desea una constante de resorte k y una constante de amortiguación b, se deben elegir los siguientes valores de CFM y ERP:

$$\begin{cases} ERP = & \frac{h\,k}{h\,k+b} \\ CFM = & \frac{1}{h\,k+b} \end{cases}$$

El método de resolución de IK, aunque no se encuentra documentado, es el de DLS (Damped Least Squares, mínimos cuadrados amortiguados, ver apéndice E)

4.3.1.5. Modelado de los motores

ODE implementa un modelo para motores donde puede especificarse un torque máximo. La herramienta aplica el torque necesario para lograr lo más rápido posible la velocidad que se quiere imponer al motor y luego mantenerla. ODE plantea la velocidad como una restricción y tiene en cuenta las fuerzas y torques externos para aplicar el torque necesario al motor y así acelerarlo hasta lograr la velocidad impuesta, siempre sin superar el torque máximo definido. La herramienta logra mediante este método un muy buen control de velocidad.

Hay que tener en cuenta que a diferencia de los motores reales el torque máximo a aplicar en la simulación no está relacionado con la velocidad del motor. Para lograr a pesar de ello un modelo razonable, calculamos el torque máximo a aplicar a partir de la velocidad máxima que vamos a utilizar para todos los movimientos. Modelando el motor mediante un comportamiento par-velocidad lineal y a partir de los datos del fabricante sobre par y velocidad máximos, podemos obtener un valor de torque máximo a aplicar para cada velocidad. Utilizar este torque máximo en la simulación es una aproximación pesimista en cuanto a la potencia del motor, ya que el simulador no va a utilizar torques mayores a velocidades menores. Usar esta aproximación es menos trabajoso que implementar un nuevo control de velocidad, por lo que decidimos utilizarlo y tener en cuenta las limitaciones al analizar los resultados.

Nuestro programa de simulación aplica la velocidad deseada a los motores y los mismos van a ser acelerados por ODE con la máxima aceleración que permita el torque disponible. Luego en cada paso de simulación controlamos si el motor llegó a su posición destino y en tal caso le imponemos velocidad cero. Hay que manejar para esto un margen de error de posición o de lo contrario dejaríamos al motor oscilando en torno de la posición de equilibrio en lugar de mantener la posición deseada.

Estamos simulando por lo tanto un control de velocidad con perfil trapezoidal (visto en el apartado 2.5.3.4). Cuanto mayor sea el torque que se aplica al motor, las etapas de aceleración y desaceleración ocupan menos tiempo y recorrido.

4.4. Desarrollo del andar del robot

4.4.1. Generación de posiciones mediante cinemática inversa

Para poder crear posiciones necesitamos poder calcular los ángulos de las articulaciones a partir de las posiciones deseadas de los pies con respecto al tronco. El cálculo de estos ángulos se realiza mediante cinemática inversa como se vio en el apartado 2.5.6.2. Para no tener que utilizar una nueva herramienta de software a tales efectos utilizamos el mismo simulador para resolver el problema. Para ello fijamos el tronco del robot y luego movemos los pies, imponiendo las nuevas posiciones deseadas de los mismos. Para cumplir con las nuevas posiciones de los pies, el simulador deberá mover las articulaciones adecuadamente para cumplir las restricciones. De esta manera, obtenemos los ángulos de las articulaciones correspondientes a una pose determinada.

Para que esto funcione correctamente, es necesario colocar los pies utilizando desplazamientos pequeños y dejar que la herramienta resuelva las restricciones en cada paso. Este enfoque nos permitió desarrollar con facilidad las posiciones y es interesante su analogía con el funcionamiento en nuestro sistema real.

Para hacer caminar el robot se diseñaron 15 posiciones: La posición de erguido y 7 posiciones básicas de cada lado. Presentamos a continuación dichas posiciones:

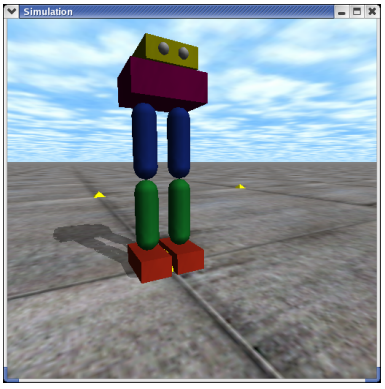


Figura 4.2: Erguido

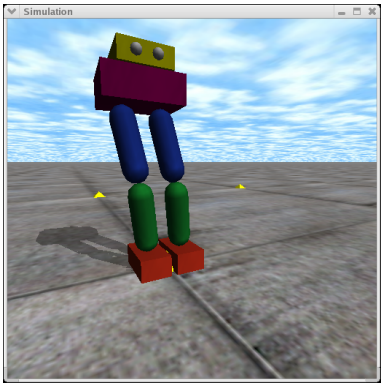


Figura 4.3: Peso sobre la derecha

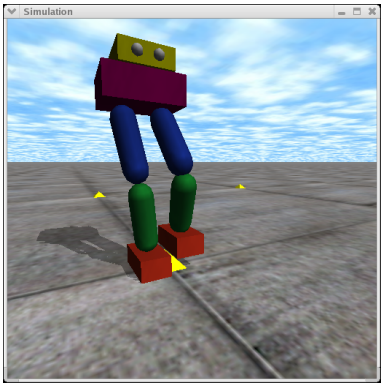


Figura 4.4: Izquierda levantada

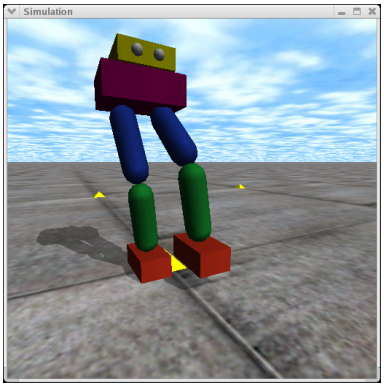


Figura 4.5: Izquierda levantada adelante

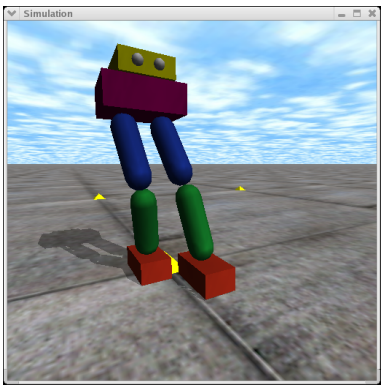


Figura 4.6: Izquierda adelante apoyada

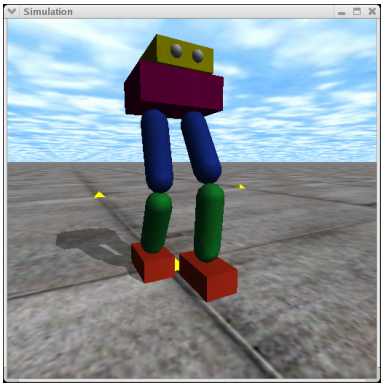


Figura 4.7: Izquierda adelante, peso en el medio

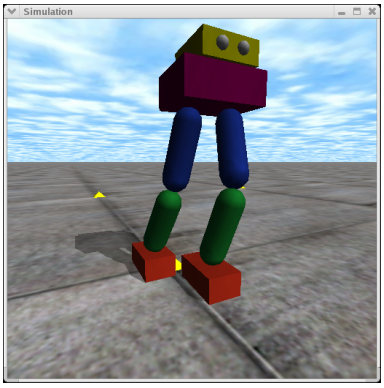


Figura 4.8: Izquierda adelante, peso sobre la izquierda

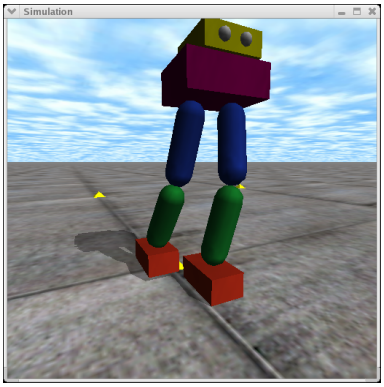


Figura 4.9: Izquierda adelante, derecha levantada

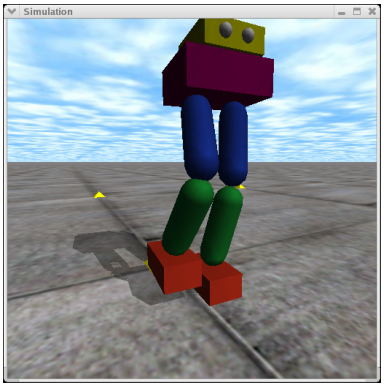


Figura 4.10: Derecha levantada

Como se puede apreciar la posición con la derecha levantada (figura 4.10) es análoga a la del lado izquierdo (figura 4.4), lo mismo pasa con el resto de las posiciones por lo que omitimos aquí las otras 6 posiciones del lado derecho. No fue necesario utilizar el programa para calcular los ángulos para las posiciones de lado derecho ya que las mismas, al ser simétricas a las del otro lado, se obtienen directamente intercambiando los ángulos.

Para validar las posiciones, se le agregó al software el cálculo de las coordenadas del centro de masa, con ello verificamos que en cada posición, la proyección del mismo quedara dentro del polígono de sustentación del robot.

4.4.2. Prueba del andar

Una vez diseñadas las posiciones, creamos un ciclo con ellas para hacer caminar al robot simulado. Para ello programamos la técnica descrita en el apartado 3.4.2.1 para el cálculo de velocidades constantes de cada articulación entre posiciones. Para cada transición se define por lo tanto una velocidad que es utilizada como máximo a partir del cual se calculan todas las demás. Si bien es posible fijar una velocidad máxima para cada transición entre posiciones, en nuestras pruebas utilizamos la misma velocidad máxima para todas las transiciones.

Para comenzar a caminar partimos de la posición Erguido en la que el robot es creado al iniciar la simulación. Luego pasamos a la siguiente posición para poner el peso sobre la pierna derecha y en tercer lugar levantamos la pierna izquierda. Estas tres posiciones corresponden a las figuras 4.2 a 4.4. Luego se siguen las posiciones en el mismo orden de las figuras hasta tener la pierna izquierda adelantada y la derecha atrás levantada (figura 4.9). Luego para la pierna derecha se repite desde la derecha levantada hasta la derecha adelante y la izquierda atrás levantada, después lo mismo para la izquierda y así sucesivamente. De estar forma el desarrollo del andar se realiza con seis posiciones de cada lado, más dos posiciones que sólo se usan para empezar a caminar y eventualmente se utilizarían para terminar de caminar, volviendo a la posición Erguido.

Puede notarse que la posición con un pie adelantado y el peso en el medio (figura 4.7) podría ser eliminada pero el robot caminaría de forma más inestable. Como se explicó en la sección 3.2, cuantas más posiciones se utilicen mejor se controla la trayectoria y se reduce el error cometido en nuestra estrategia de cálculo de velocidades.

4.5. Resultados obtenidos

Al ciclar al robot entre las distintas posiciones se logró que el mismo caminara siempre y cuando la velocidad se mantuviera pequeña. Una vez que logramos que el robot simulado caminara utilizamos el simulador para probar los límites de nuestra estrategia y obtener datos que nos permitieran sacar la mayor cantidad posible de conclusiones.

4.5.1. Límites de funcionalidad

Figura 4.11: Robot cayéndose

En el intento por encontrar los límites de nuestra estrategia comenzamos a aplicar mayores velocidades. Primero se realizaron pruebas con distintas velocidades máximas y torques asociados calculados de la forma descrita en la sección 4.3.1. Para nuestros motores el torque máximo (a velocidad 0) es de 0.40 Nm y la velocidad máxima (donde el torque es cero, sino el motor aceleraría) es de 2.76 rad/s. Trazamos entonces una recta que aproxima la respuesta del motor a distintas velocidades.

Experimentando con torques y velocidades en esta recta logramos que el robot caminara utilizando una velocidad máxima de 0.7 rad/s con un torque máximo de 0.3 Nm. Con esta velocidad el robot se desplaza a 2 cm/s. Con velocidades mayores algunas articulaciones no logran el suficiente torque para realizar los movimientos requeridos y el robot se cae o se tranca sin poder caminar. En este límite nuestra aproximación ya no es válida porque en realidad los motores pueden hacer más torque si la velocidad de la articulación que falla es más baja que la máxima, por lo que sería necesario un modelo más fino para encontrar límites más ajustados de velocidad.

Para la siguiente prueba eliminamos la restricción de torque y probamos con distintas velocidades para averiguar dónde fallaría el robot si tuviera motores más potentes. El primer problema que nos encontramos fue la absorción del impacto con el piso. Cuando el robot apoya un pie en el

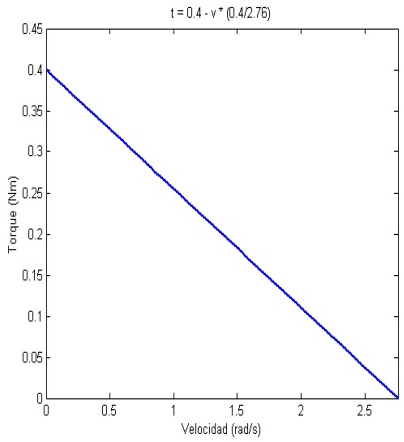


Figura 4.12: Torque vs Velocidad

piso con cierta velocidad el impacto provoca una reacción en el robot y esta lo desestabiliza. Para solucionar el problema en la simulación aplicamos distintas propiedades al contacto con el piso, logramos superar este problema al modelar el contacto con un efecto de resorte/amortiguador. De esta forma parte de la energía del impacto es absorbida por el amortiguador y la deformación de la superficie de contacto es restaurada por el resorte. Se requeriría un estudio detallado del comportamiento de distintos materiales para saber si este modelo puede ajustarse a algún material para la planta del pie del robot o a algún tipo de suspensión que se pudiera construir.

El siguiente límite de velocidad lo encontramos cuando la estrategia falla como aproximación estática al problema dinámico del equilibrio. Cuando llegamos a una velocidad máxima de las articulaciones de 1.8 rad/s el robot se desplaza a 5.5 cm/s, al aumentar esta velocidad el robot se cae por la el efecto de la inercia. Una vez más este límite está sujeto a las simplificaciones del modelo.

4.5.2. Conclusiones

Mediante la experimentación con la simulación pudimos 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. De acuerdo a esto sabemos que el controlador de los motores tiene que tener un buen control de velocidad o la coordinación del movimiento de las articulaciones no sería posible.

De acuerdo a lo que vimos en el estudio del Estado del Arte, no existen robots bípedos que utilicen motores de tan poco 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 con nuestros motores es muy poco, ya que si bien el robot camina no contamos 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 que utilizamos para caminar, el hecho de que las velocidades aplicadas lleven a una secuencia de movimientos que haga caminar al robot nos da un

buen nivel de seguridad en cuanto a la validez cinemática de la misma. La simulación nos brinda por lo tanto confianza de que si las velocidades son suficientemente pequeñas un robot puede caminar con esta estrategia.

4.6. Documentación del sistema de simulación

4.6.1. Módulos del sistema

Robot Este módulo se encarga de la simulación del robot y es el que interactúa con ODE. En el archivo robot.h se encuentran las definiciones de dimensiones y masa de los segmentos del robot, así como también los topes de las articulaciones. En cada paso de simulación se llama a la función `controlar` del módulo Control, pasándole las posiciones actuales de las articulaciones y recibiendo las nuevas velocidades a aplicar. En el caso de que estemos utilizando el simulador para cinemática inversa, este módulo actúa solo y no se utilizan los otros dos. Para mover los pies a las nuevas posiciones, espera del usuario comandos mediante el teclado. El orden de los DOF del robot que deben respetar los tres módulos para comunicarse correctamente es:

1. Pie izquierdo, rotación izquierda-derecha
2. Pie izquierdo, rotación adelante-atrás
3. Pie derecho, rotación izquierda-derecha
4. Pie derecho, rotación adelante-atrás
5. Rodilla izquierda
6. Rodilla derecha
7. Cadera izquierda, rotación izquierda-derecha
8. Cadera izquierda, rotación adelante-atrás
9. Cadera izquierda, rotación con eje vertical
10. Cadera derecha, rotación izquierda-derecha
11. Cadera derecha, rotación adelante-atrás
12. Cadera derecha, rotación con eje vertical

Control El módulo Control se encarga de verificar si cada motor llegó a la posición destino o no. Para ello cuenta con las posiciones destino y las velocidades a utilizar brindadas por el módulo Estrategia. En caso de que un DOF esté en la posición de destino (salvo un margen de error) la velocidad del motor correspondiente se setea en cero. Si en cambio el DOF se encuentra fuera del entorno de la posición destino se setea la velocidad correspondiente con el signo apropiado. En cada paso se llama a la función decidir del módulo Estrategia pasándole la información sobre si llegó o no cada DOF a su posición destino.

Estrategia Este módulo exporta la función decidir. La misma recibe únicamente la información sobre si cada DOF llegó o no a su posición destino. Esta función le dice al módulo Control si desea especificar un nuevo destino con nuevas velocidades o no. El funcionamiento en el andar del robot es calcular las velocidades de acuerdo a nuestra estrategia y enviar al módulo Control la siguiente posición con las velocidades correspondientes. Luego no se actúa hasta que se recibe la información de que todos los DOF llegaron a su destino, calculando entonces las velocidades apropiadas para ir a la siguiente posición y ordenando al módulo Control que lo haga.

4.6.2. Compilación

El simulador puede ser compilado tanto en Windows como en Linux. En ambos casos debe contarse con ODE instalado en el sistema. Para el caso de Linux se provee un Makefile, este archivo debe ser editado para especificarle los caminos correctos para los cabezales y la biblioteca de ODE. En el caso de Windows se entregan los archivos robot7.dsp y robot7.dsw para ser utilizados con Microsoft Visual C++ 6.0, donde también deben especificarse los caminos para cabezales y bibliotecas de ODE.

4.6.3. Ejecutables

En todos los ejecutables se visualiza gráficamente al robot y la posición y ángulo de la cámara pueden cambiarse con el ratón. Al arrastrar el ratón: con el botón izquierdo presionado se rota la cámara, con el botón derecho presionado se mueve la cámara izquierda-derecha/adelante-atrás y con el botón del medio (o izquierdo y derecho a la vez) se mueve la cámara izquierda-derecha/arriba-abajo.

Robot_ik Este programa es utilizado para obtener los ángulos de las articulaciones que podrían obtenerse realizando los cálculos de cinemática inversa. Al ejecutarlo el robot aparece erguido, colgado en el aire con el tronco y los pies fijados en sus posiciones. Al presionar la tecla *a* el programa imprime las coordenadas y rotaciones de los pies y el tronco, la masa y el centro de masa del robot y los ángulos de las articulaciones, que es el dato que nos interesa obtener para utilizar las posiciones. Cabe recalcar que ni el simulador ni la simulación realiza cálculos de cinemática inversa, sino que el simulador impone restricciones en las articulaciones y cuerpos, utilizando los métodos descritos en [ODEU].

Para mover el pie izquierdo se utilizan las teclas *f, h, t, g, y, n.* Cuya función es mover el pie en sentidos izquierda-derecha, adelante-atrás y arriba-abajo respectivamente. Para el pie derecho las teclas son: *j, l, i, k, u, m.* El desplazamiento al apretar una tecla es de 1 cm, el mismo puede ser incrementado y decrementado de a un milímetro con la teclas *s* y *x* respectivamente. Es aconsejable comenzar a trabajar oprimiendo las teclas *y* y *u* para acercar los pies al tronco y permitir la movilidad. No se puede con este programa cambiar la rotación de los pies, ya que fue pensado para desarrollar posiciones para caminar con los pies planos, sin embargo sería muy simple agregar esta funcionalidad.

Es posible, cuando los pies son llevados a posiciones en las que el sistema no puede satisfacer las restricciones, que la simulación se torne inestable y quede inusable, en este caso se puede utilizar la

tecla *r* para reiniciar el programa. En este ejecutable está deshabilitada la detección de colisiones, por lo que las piernas del robot pueden superponerse.

Robot_colgado Este ejecutable realiza todos los movimientos definidos en el módulo Estrategia y detecta colisiones entre las diferentes partes del robot. Al iniciar el programa el robot se encuentra, al igual que en el ejecutable anterior, erguido en el aire con el tronco en una posición y rotación fijas. Se puede utilizar este ejecutable para observar la transición entre las diferentes posiciones independientemente de los efectos del contacto con el piso.

Robot Este es el ejecutable que simula al robot caminando sobre el piso. Una vez que las posiciones están desarrolladas lo utilizamos para observar el resultado final del movimiento. Cada 10 segundos de simulación se despliega la velocidad promedio de su desplazamiento en el eje Ox. El robot inicialmente se encuentra mirando en el sentido positivo de dicho eje, por lo que si no se le imprimen giros (o suceden por problemas del andar) se puede tener una buena aproximación a la velocidad obtenida al caminar.

Capítulo 5

Diseño mecánico

5.1. Objetivos

Se deben diseñar piezas para un prototipo funcional de robot bípedo. Como se vio en la sección 3.4.3 deseamos 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. Como se vio en el capítulo anterior, 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.

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. Muchas veces es muy difícil saber si la pieza va a funcionar correctamente hasta que se construyó, por lo que se preevió que gran parte de las piezas iban a ser desechadas durante el proceso. Por estas determinantes la construcción no debe requerir de mano de obra especializada ni de materiales relativamente caros, así como tampoco de demasiadas horas de trabajo.

El diseño básico de las piezas que desarrollamos podría ser reutilizado para construir una versión mejorada del robot de contarse con más medios en el futuro, pero el diseño deberá ser adaptado a las técnicas disponibles en ese caso.

5.2. Técnicas disponibles

5.2.1. 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

5.2.2. 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

5.3. Diseño general

5.3.1. Fijación de los servos a las piezas

El servomotor para poder actuar debe estar fijado a dos piezas del robot. A una de ellas se fija la carcasa y a la otra el eje de salida. Los servomotores cuentan para esto con cuatro agujeros para tornillos en la carcasa y con un brazo de plástico que se adosa al eje. Este brazo se une con el eje mediante un calce para el engranaje dentado del eje y un tornillo para fijarlo.

Figura 5.1: Servomotor Futaba S3003 con brazo de plástico en X

Figura 5.2: Fijación del eje y de la carcasa del servo

Fijación de las piezas al eje del servo Si bien en un principio podría considerarse fijar las piezas directamente al eje del servo, esto requeriría de tecnología que no poseemos para fabricar el calce apropiado. Por ello decidimos fijar las piezas al brazo de plástico, que es además una práctica común. Si bien los agujeritos en los brazos de los servos están pensados para calzar alambres que los conecten a la pieza a mover, los mismos pueden ser utilizados para atornillar tornillos autorroscantes. El diámetro de estos agujeros es de tan sólo 1 mm y los tornillos no pueden sobresalir más de 5 mm hacia el servo o se trancarían con la carcasa. Utilizamos por lo tanto los tornillos autorroscantes estándar más pequeños. Los mismos tienen 2,9 mm de diámetro exterior y 6,5 mm de largo. Para montar estos tornillos se realizan en las piezas agujeros de 1/8" (3,175 mm) para que pasen los tornillos que luego se enroscan en los brazos de plástico. En el centro del eje se hace en la pieza un agujero de 8 mm de diámetro para que el tornillo que une el brazo con el servo pueda ser removido, permitiendo así separar el brazo del servo para desmontar la articulación o para ajustar la ubicación de la trayectoria del servo.

Fijación de las piezas a la carcasa del servo Los servos traen cuatro tornillos para ser utilizados con este fin. Los mismos son tornillos autorroscantes de una medida no estándar, es necesario además para utilizar estos tornillos agregar al servo dos pequeñas piezas de goma y cuatro arandelas de metal (todo esto viene incluido con el servo). Para evitar problemas en caso de pérdida o rotura de alguna de estas piezas y simplificar la construcción utilizamos en cambio los mismos tornillos que para fijar los brazos, que además se pueden utilizar sin las piezas extra. Los agujeros de las piezas tienen que ser en este caso de 2 mm para permitir crear la rosca adecuada para los tornillos.

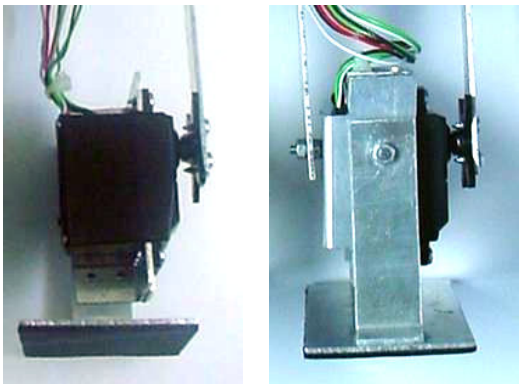


Figura 5.3: Pie con ejes simples y pie con ejes dobles

5.3.2. Articulaciones

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. Para hacerlo basta con fijar al brazo del servo la pieza que constituye el segmento adyacente en cada DOF del robot. Con estructuras simples y livianas este enfoque es adecuado, pero cuando se trata de un robot complejo comienzan a aparecer problemas. De esta forma se llegó a armar una pierna completa y lograr movimientos en la misma. 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. Para esto último consideramos el uso de rulemanes, pero el costo y complejidad que conlleva llevó a que nos decidiéramos por utilizar un tornillo como perno y un agujero en la pieza para que el tornillo rodara. Esta solución mostró ser suficiente para el funcionamiento de nuestro prototipo.

5.3.3. Plaqueta y cables

Para fijar la plaqueta al cuerpo utilizamos tornillos con tuerca de 1/8" (3.175 mm) de ancho externo y 1/2" (1,27 mm) de largo. En la plaqueta son necesarios para esto agujeros de 1/8" (3.175 mm) de ancho en la posición adecuada que deben ser tenidos en cuenta en el diseño de la misma.

Los cables de los servos se agruparon utilizando precintos y se dejaron libres con largo suficiente para que la articulación pueda moverse. Para ello se calcularon las medidas de cable necesarias para cada servo, las que se presentan a continuación, numerando los servos de acuerdo al orden en que unen segmentos si se parte del cuerpo hacia el pie:

Número de servo	Longitud del cable
1	20 cm
2	40 cm
3	50 cm
4	65 cm
5	75 cm
6	80 cm

Cuadro 5.1: Largo de los cables de los motores

5.4. Diseño de las piezas del robot

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.

En total se utilizó una plancha de aluminio de 30 cm x 50 cm y goma eva para la planta de los pies. Los agujeros son de 3 medidas diferentes: 2 mm, 1/8" (3.175 mm) y 8 mm. Se necesitan para armar el robot 96 tornillos autorroscantes de 2.9 mm de ancho y 6.5 mm de largo y 20 tornillos 1/8" (3.175 mm) de ancho y 1/2" (1.27 mm) de largo con 48 tuercas.

A continuación se presentan diagramas de las piezas. En los diagramas la línea punteada indica un doblez y las superficies cuadriculadas son los lugares donde se pegan las piezas. Para mayor legibilidad se muestran aquí sólo algunas cotas. Para diseñar las piezas se utilizó el programa Autocad y uno de los archivos entregados en este proyecto es el llamado *hdp.dwg* que contiene el diseño de todas las piezas.

5.4.1. Pie

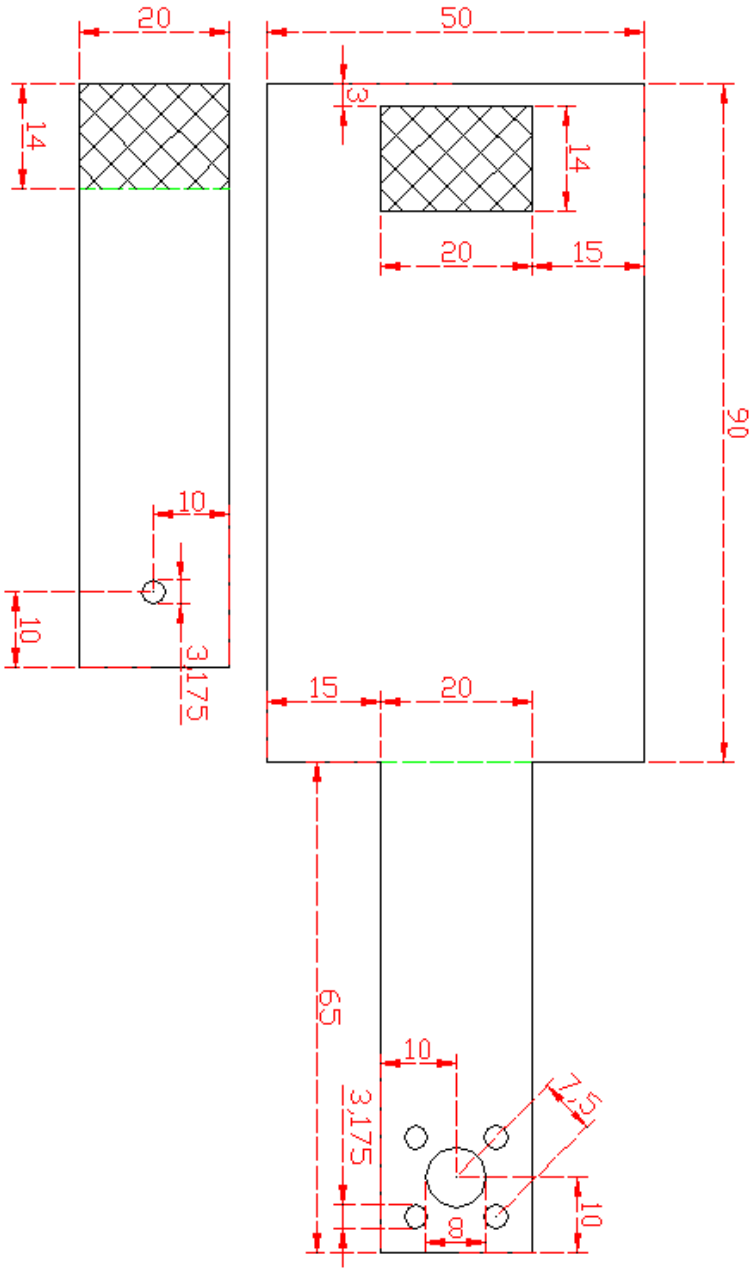


Figura 5.4: Pie

5.4.2. Tobillo

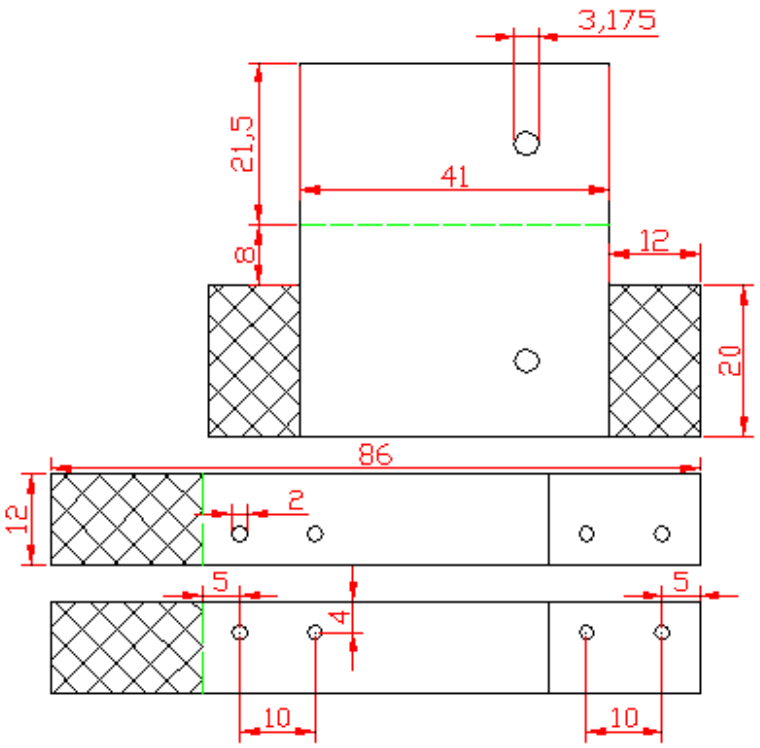


Figura 5.5: Tobillo

5.4.3. Canilla

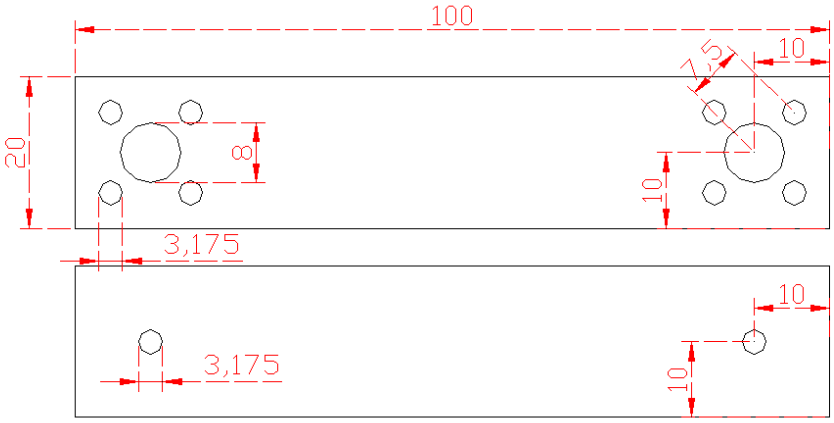


Figura 5.6: Canilla

5.4.4. Muslo

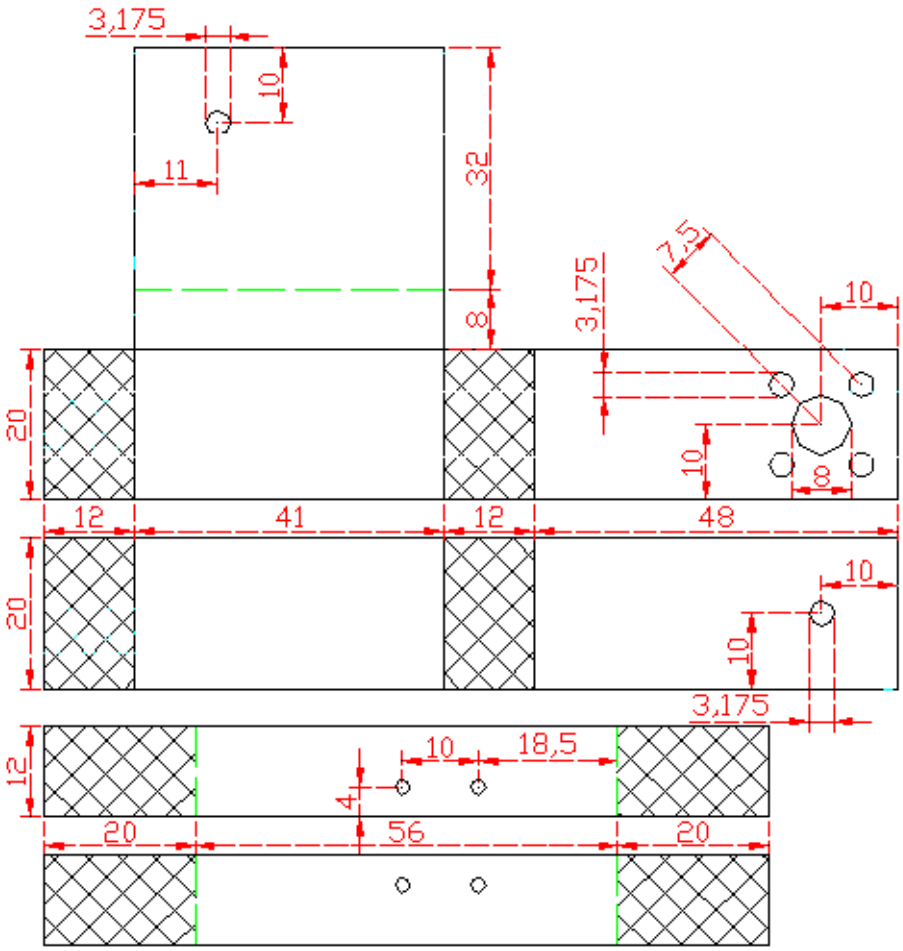


Figura 5.7: Muslo

5.4.5. Muslo-Cadera

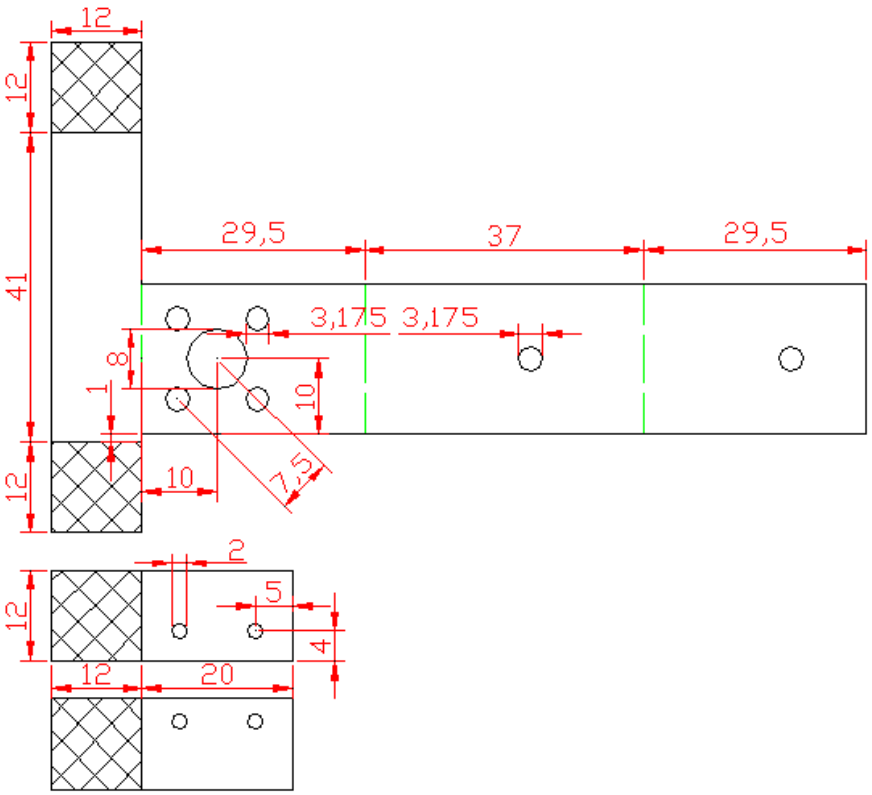


Figura 5.8: Muslo-Cadera

5.4.6. Cadera-Cuerpo

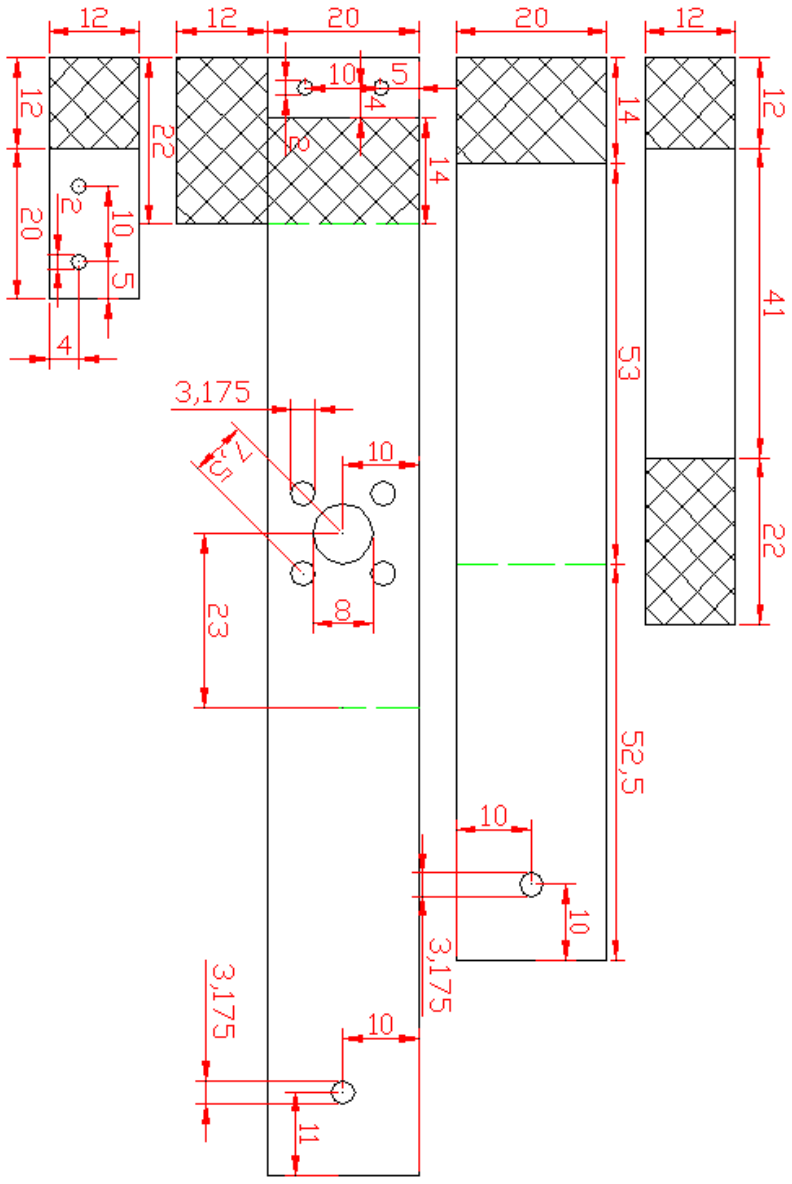


Figura 5.9: Cadera-Cuerpo

5.4.7. Cuerpo

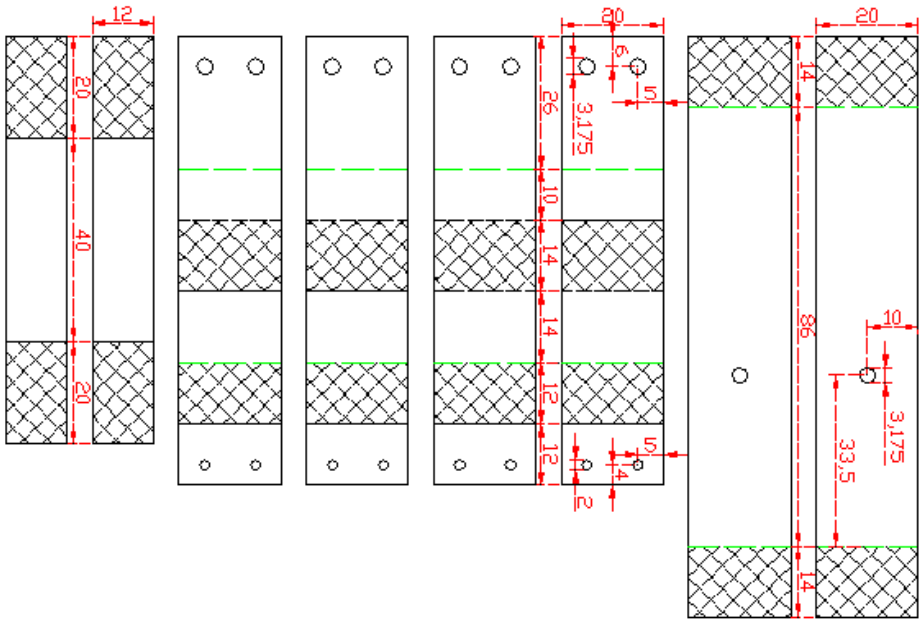


Figura 5.10: Cuerpo

A continuación se presentan algunos diagramas que ilustran cómo se disponen las distintas piezas. Cada pieza se marca con una flecha, las articulaciones están marcadas por su eje de rotación y con su nombre en cursiva, y los números que aparecen son los correspondientes a cada motor según la tabla 5.1. También se muestra un detalle de cómo se fija el servomotor de la articulación de la rodilla.

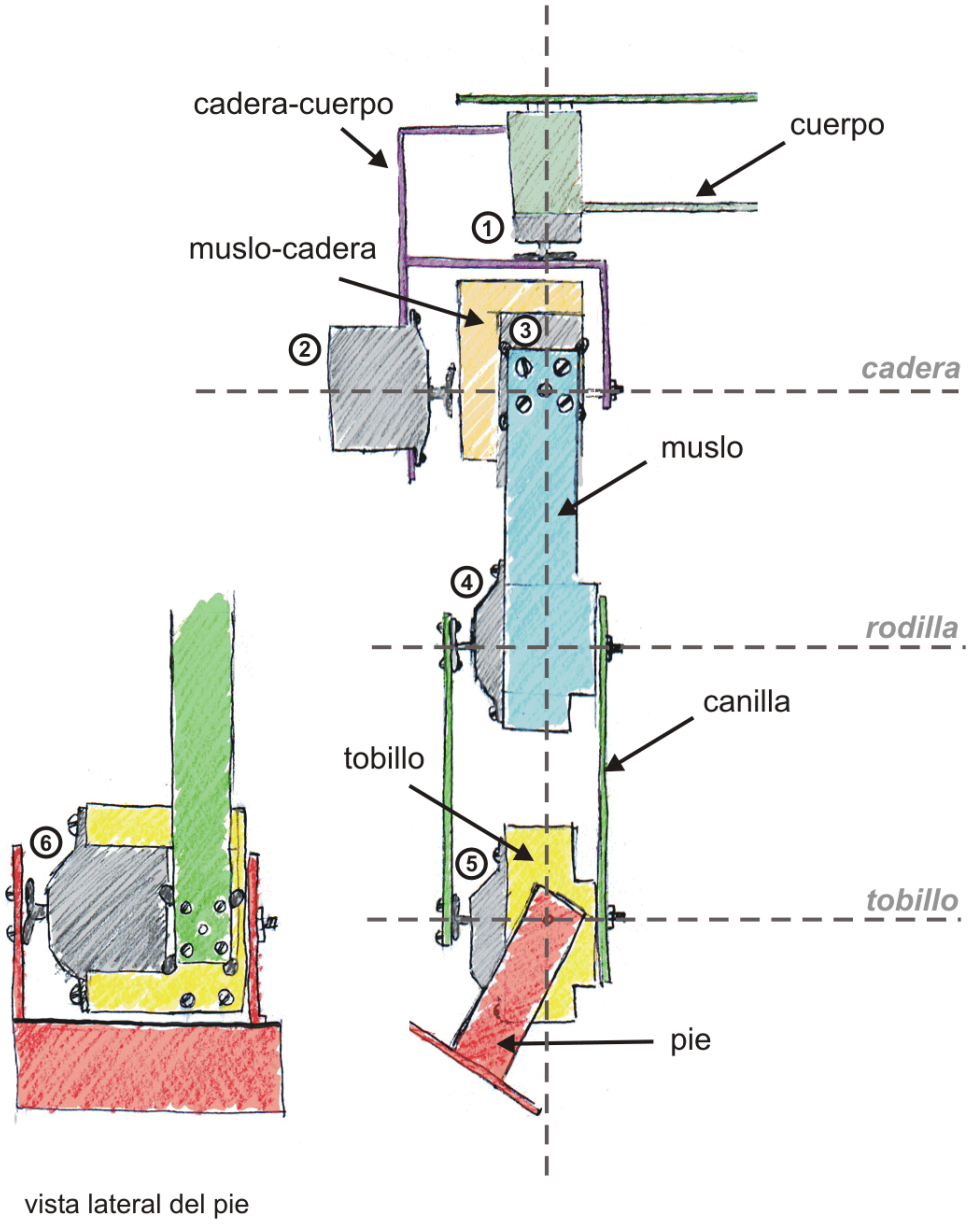


Figura 5.11: Diagrama de armado

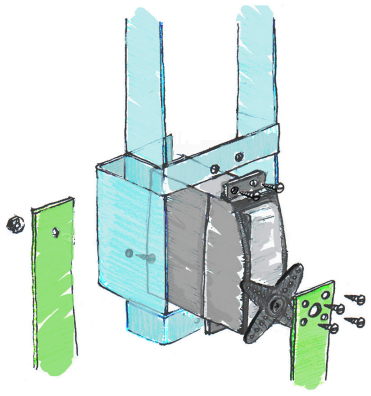


Figura 5.12: Explosión de la articulación de la rodilla (motor número 4). Este motor une el muslo (cuerpo del motor) y la canilla (eje del motor).

5.5. Resultados obtenidos

5.5.1. Descripción de los resultados

Se diseñaron y construyeron las piezas para el robot bípedo cumpliendo con la mayoría de los requerimientos planteados. Para lograr la movilidad requerida no se pudo mantener la altura propuesta en el caso del pie y los ejes del tobillo se encuentran a 5.5 cm de altura en lugar de 3.5 cm como se había propuesto. El robot es por lo tanto 2 cm más alto de lo esperado. El peso fue otra restricción que no se pudo mantener ya que el robot pesa 1.00 kg en lugar de 750 g como se esperaba. La altura total del robot construido es de 36 cm (más 10 cm aproximadamente por los cables), pero el centro de la articulación de la cadera se encuentra sólo a 21.5 cm del piso (19.5 cm era la medida en el robot simulado).

Una limitación del prototipo es que las piezas ceden un poco cuando son sometidas a mucho esfuerzo, cambiando momentáneamente la estructura del robot. Esto se nota sobre todo en posiciones con un pie en el aire. Otra limitación es que los tornillos que ofician de eje a las articulaciones y que se encuentran hacia adentro en el robot se chocan si se juntan mucho las piernas, por lo que el robot debe mantener las piernas ligeramente separadas. También el rozamiento en estos rodamientos es mayor que el deseado, pudiendo ser disminuido aceitando los mismos. Las uniones de las piezas pegadas con cemento de soldadura plástica por su parte presentan cierta fragilidad, y si bien en la operación normal del robot no se rompen, no son resistentes a caídas o un tratamiento descuidado.

Estas limitaciones pueden ser superadas si se construyen las piezas contando con otras técnicas que permitan construirlas más resistentes, con medidas más ajustadas, dotarlas de nervios y ángulos y adosarle rulemanes en las articulaciones entre otras mejoras posibles. De todas formas como se verá en el capítulo 9 lo que más limita la capacidad del robot son los servomotores utilizados.

5.5.2. Fotos del robot

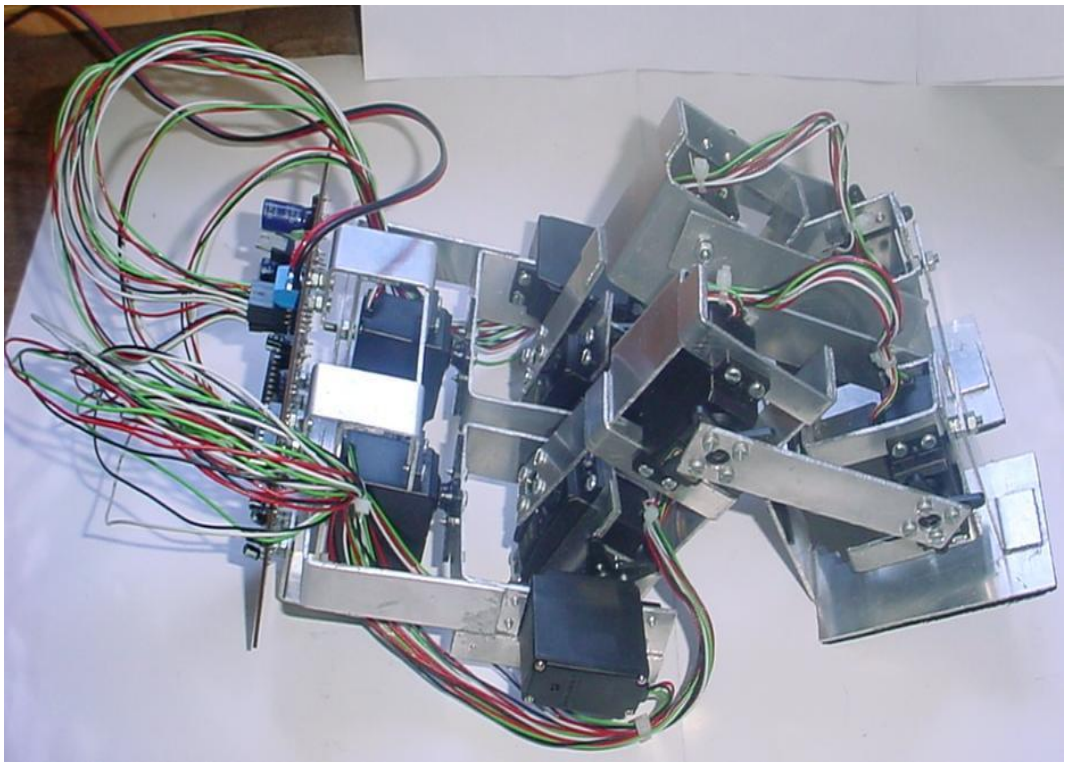


Figura 5.13: Nuestro Robot

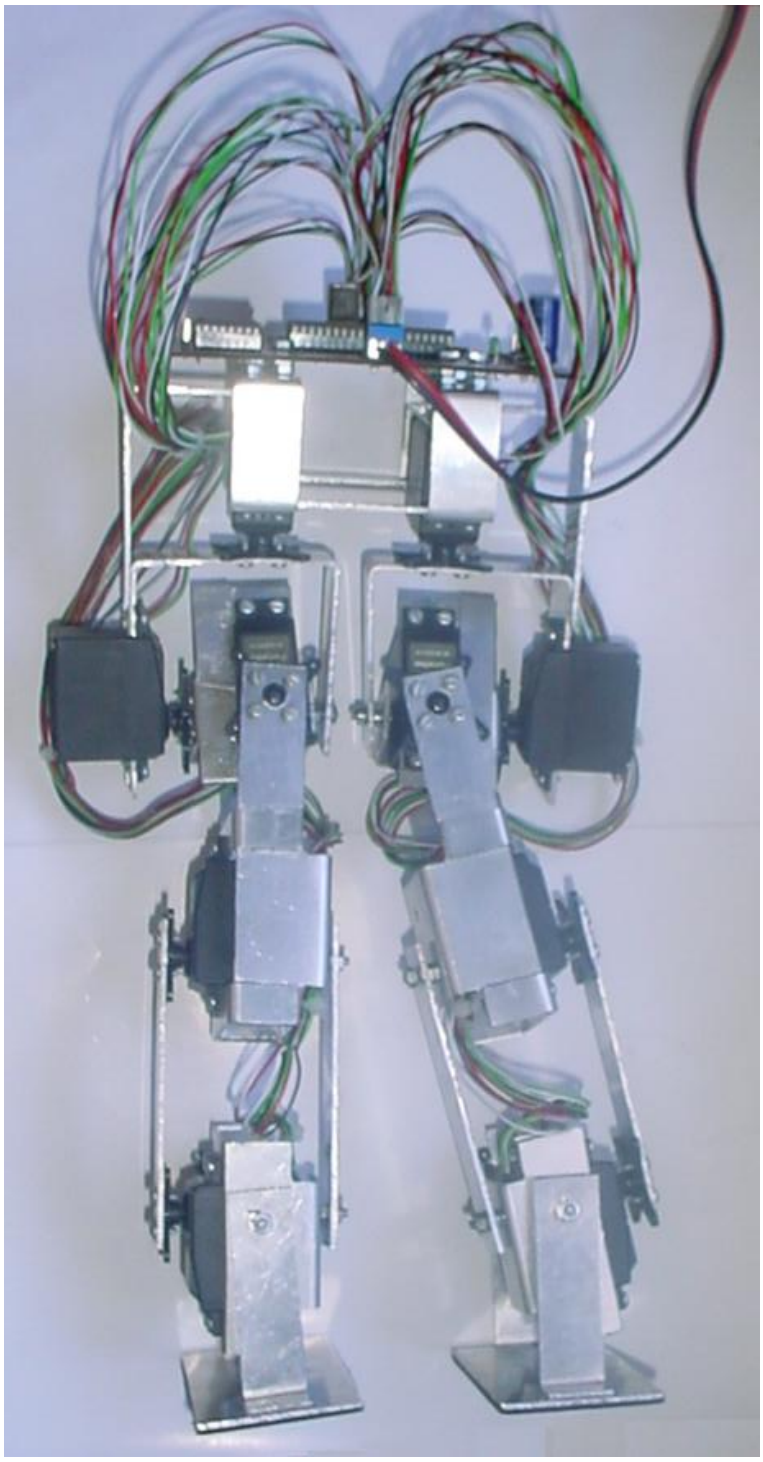


Figura 5.14: Nuestro Robot II

Capítulo 6

Descripción eléctrica del Sistema

Como se describió en la sección 3.3, el sistema consta de dos nodos bien diferenciados. Por un lado está el PIC que implementa el manejo de los motores y los sensores. Por otro lado está el PC donde se ejecuta el software de más alto nivel, y se comunica con el PIC a través del puerto serie. En lo que se refiere al diseño eléctrico, el nodo PC no reviste mayor dificultad que el cable serial, por lo que en esta sección nos concentraremos en el diseño eléctrico del nodo PIC.

6.1. Diseño a alto nivel

En la siguiente figura podemos ver el sentido en que fluyen las señales entre los distintos nodos, actuadores, y sensores del sistema. Se muestra además del sentido de flujo, los niveles de voltaje en el caso de señales analógicas, o 0/1 en el caso de señales digitales, la forma de codificación y la cantidad de bits de resolución. En el caso de la interfaz entre cada servomotor y su respectivo sensor, se indica que la transferencia de información está dada por la posición del potenciómetro en el eje.

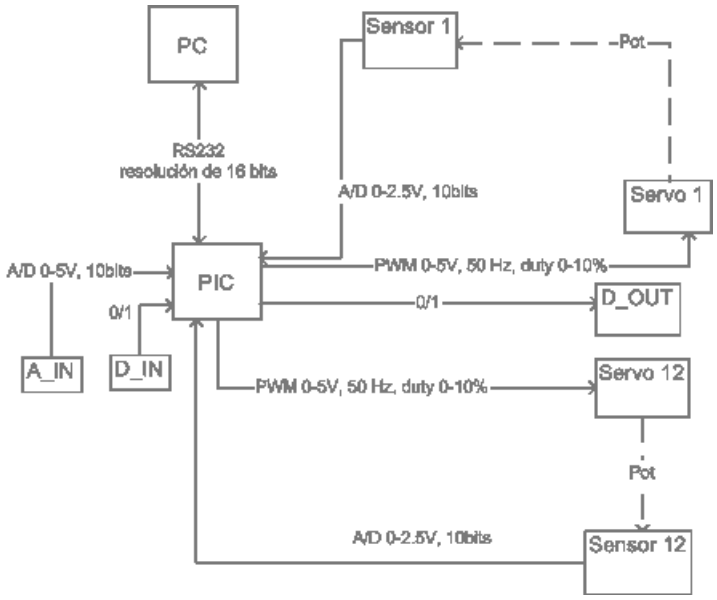


Figura 6.1: Diseño a alto nivel

6.2. Entorno de desarrollo

6.2.1. Alimentación

La alimentación del nodo PIC se obtiene de una fuente de alimentación de PC de 300W, en +12 V, +5 V y tierra. Los servomotores operan entre 4.8 y 6 V, y consumen una corriente menor a 10 mA en reposo sin carga. Según la página del fabricante (Futaba), los motores pueden consumir grandes cantidades de corriente, por lo que se hace necesario una fuente de potencia suficiente.

A modo experimental se implementó una etapa de potencia para la alimentación de los motores que es independiente de la alimentación de la electrónica. La misma se implementó como una fuente de voltaje de unos 6 V y 25 W, y está implementada con la fuente de 12 V, diodos zener de 6.8 V, resistencias y un transistor de potencia de la serie TIP.

Para el diseño de la fuente, se estimó el consumo de los motores como una fuente de corriente de onda cuadrada, con valores entre $40\,mA$ y $4\,A$, y un ciclo de trabajo de $10\,\%$ en la parte alta. Esta información, junto con la corriente necesaria para polarizar los diodos zener, determinan el valor de la resistencia de polarización.

La fuente que alimenta a esta es de 12 V, el transistor tiene una caída de 0.8 V aproximadamente, y queremos tener 6 V a la salida, por lo que los diodos zener deben ser de 6.8 V. La corriente de polarización de los zener es de 3 x 30 mA = 90 mA. La corriente de base del transistor será igual a la corriente del colector dividido entre el β del transistor (alrededor de 30). Eso hace que la corriente por la resistencia sea $i_R = 90\,mA + \frac{4\,A}{30} = 223\,mA$. Por lo que el valor de la resistencia debe ser $R = \frac{12-6.8}{i_R} = 23.3\,\Omega$.

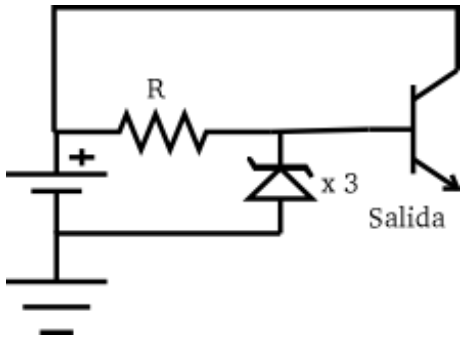


Figura 6.2: Fuente inicial

El principal componente del nodo en la etapa de diseño y testeo es un Protoboard donde se monta el PIC 18f4320, el cable serial, un chip MAX232, los motores y salidas digitales.

A continuación se enumeran algunas características eléctricas de interés:

1. Consumo de 1 W
2. Máxima corriente desde V_{SS} de 300 mA
3. Máxima corriente hacia V_{DD} de 250 mA
4. Máxima corriente por cada pin de entrada/salida de $\pm$25 mA
5. Máxima corriente por todos los puertos de $\pm$200 mA
6. Voltaje de alimentación a 40 MHz entre 4.2 V y 5.5 V
7. Soporta ICSP. Esta característica se utilizó en la etapa de desarrollo para poder programar el dispositivo.

6.2.2. Motores

En lo que respecta al aspecto eléctrico, los motores elegidos tienen tres conductores, uno de tierra, uno de alimentación y uno de control. Según el fabricante ([FUTA]), los motores pueden llegar a consumir una potencia considerable (3.6 W). Las especificaciones de los motores se dan en el apéndice D.1.

6.2.2.1. 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.

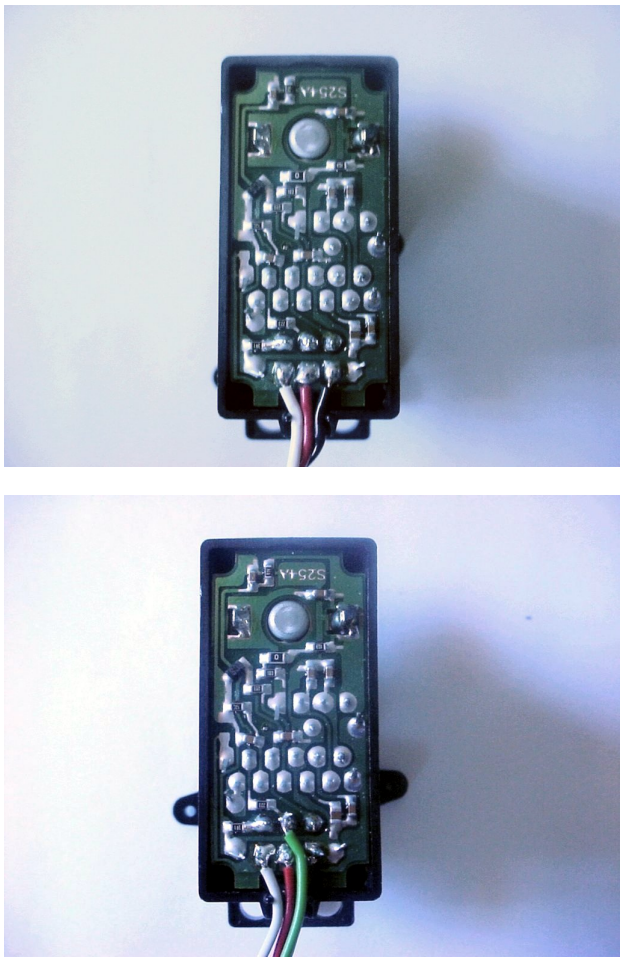


Figura 6.3: Modificación del servomotor: antes y después. Notar la adición del cable que contiene la señal de lectura de posición.

6.2.3. Interfaz Serial

El cable serial se implementa con los siguientes hilos: RX, TX, RTS, DTR y CTS. No se utiliza control de flujo, por lo que los cables RTS, DTR y CTS se conectan de forma que cumplan con las especificaciones de los programadores ICSP.

6.2.4. MAX232

El chip MAX232 convierte los niveles de tensión RS-232 a niveles TTL y viceversa. Esto es necesario ya que el "1" en RS-232 se representa como una tensión entre -3 y -12 V, y en TTL como 5 V, y el "0" en RS-232 es una tensión entre 3 y 12 V y en TTL 0 V.

6.3. Entorno de producción

Para la etapa de puesta en producción, el sistema se porta a un circuito impreso (PCB), que se adosa al cuerpo del robot, como se muestra en la siguiente figura.

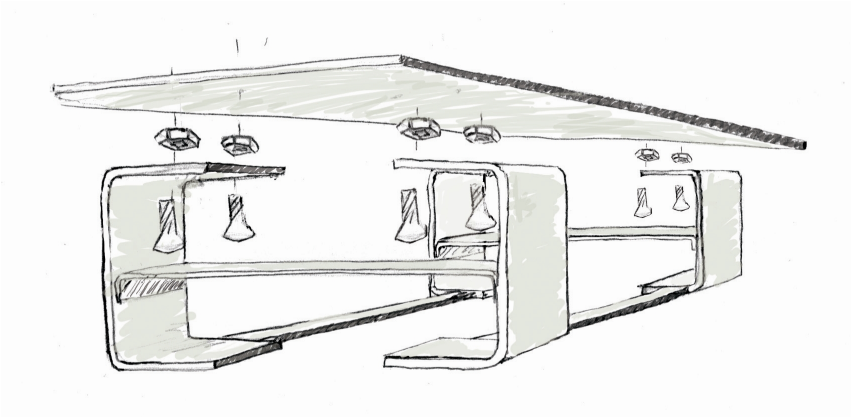


Figura 6.4: Adosado de la placa al cuerpo del robot

El circuito impreso fue diseñado con un software gratuito llamado ExpressPCB, siguiendo las siguientes guías de diseño y restricciones:

1. Minimización del área de trabajo.
2. Integración con el diseño mecánico.
3. Modularidad en los conectores de los motores.
4. Posibilidad de alimentar los motores a 6 V y la electrónica a 5 V. Debido a que los motores pueden alimentarse entre 4.8 V y 6 V, es deseable que esté la posibilidad de alimentarlos a 6 V para obtener mayor torque.
5. Conexión serial RS-232 simple.
6. Conexión estándar para las señales digitales y analógicas de entrada y salida.

A continuación se enumeran las medidas y consideraciones tomadas para resolver los requerimientos y directrices:

1. Se tomó un área de PCB de 12 cm x 9 cm.
2. La placa posee los agujeros que permitan adosar los motores que irán contra la misma (es decir, los primeros dos motores de la cadera).
3. Los motores se operan para sacar hacia afuera el hilo de lectura de posición. De esta manera se tienen cuatro hilos (posición, alimentación, tierra, control). Se proveen dos grupos de conectores de cuatro pines a ambos lados del microcontrolador en la placa: un primer grupo de 8 conectores y otro de 4.
4. Para la electrónica se dispuso un integrado LM2940CT que es un regulador de 5 V de baja caída. La entrada de dicho regulador viene de la alimentación general.
5. La alimentación general se provee en 6 V y 5 A máximo, a través de la fuente construida.

6. Para las entradas y salidas digitales se provee un conector de dos pines, uno de ellos a tierra. Además, las lecturas digitales poseen una resistencia pull-up a 5 V. Para el sensor analógico se provee un conector de tres pines: alimentación, lectura y tierra. Esto permite conectar por ejemplo un potenciómetro.

7. Para la comunicación serial se provee un conector de tres pines: GND (tierra), TX (transmisión) y RX (recepción). Este conector está al lado del integrado MAX-232 para minimizar el largo de las pistas.

Tipo de conectores Para todos los conectores de la electrónica se utilizaron conectores internos de audio de PC de 2 (entrada digital), 3 (cable serial y entrada analógica) y 4 pines (motores).

Figura 6.5: Un cable con conectores internos de audio

Alimentación La fuente fue implementada tomando la alimentación desde los 12 V de la fuente de PC utilizada. 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. Los capacitores son de $1\,\mu F$ y la resistencia es de $2.2\,\Omega$.

Esta fuente alimenta directamente a los motores. El PIC se alimenta en 5 V, pasando esta fuente por un regulador 7805 como se muestra en el esquemático 6.9. El circuito de la fuente es el que se muestra a continuación.

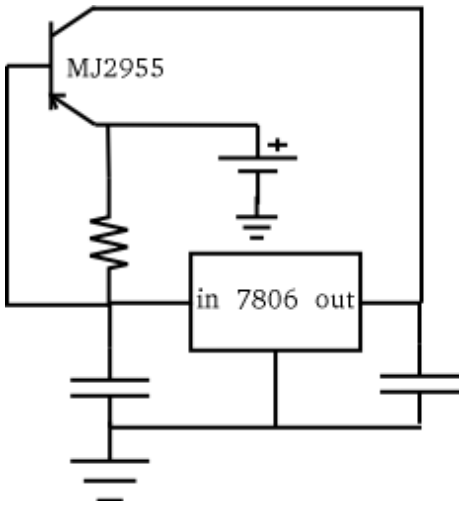
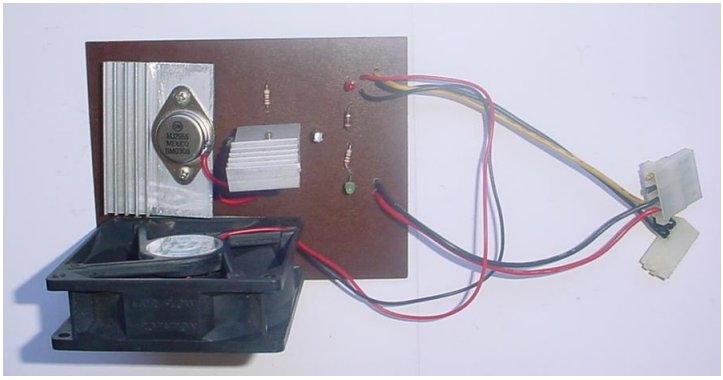


Figura 6.6: Fuente de 6 V / 5 A DC



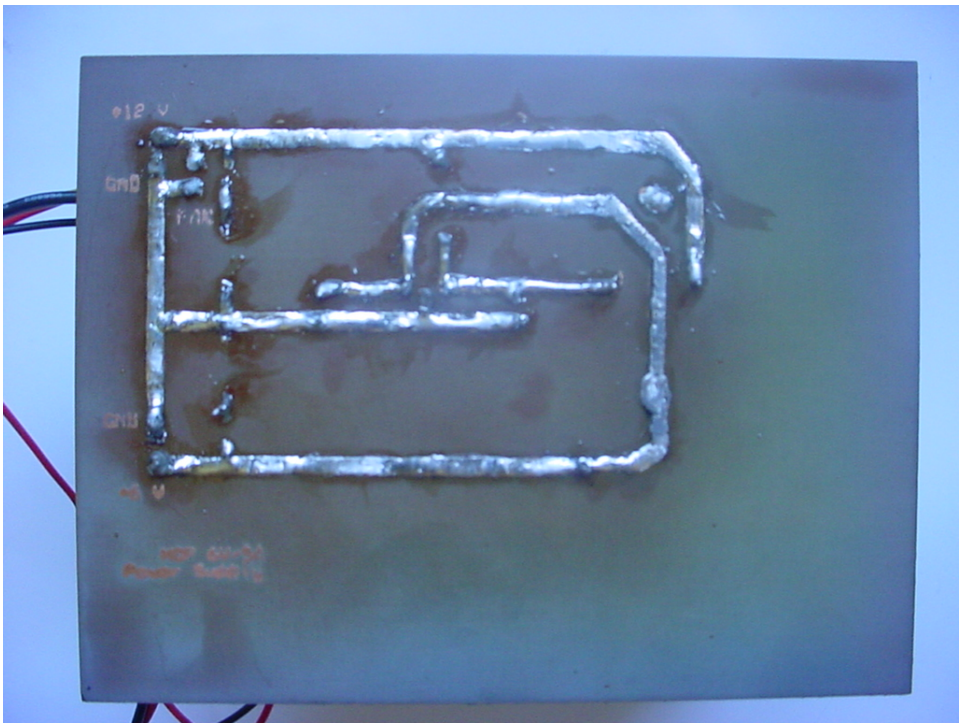


Figura 6.7: Fuente de 6 V / 5 A DC

6.3.1. Desacople

Para desacoplar tierra y fuente a nivel general se pusieron dos condensadores: uno a la entrada del regulador, entre V_{IN} y tierra y otro a la salida, entre V_{OUT} y tierra. Ambos condensadores son electrolíticos de 1000 μF.

Además se dispuso un condensador electrolítico de 47 μF para desacoplar V_{DD} y V_{SS} en la electrónica del PIC.

6.3.2. Proceso de producción del circuito impreso

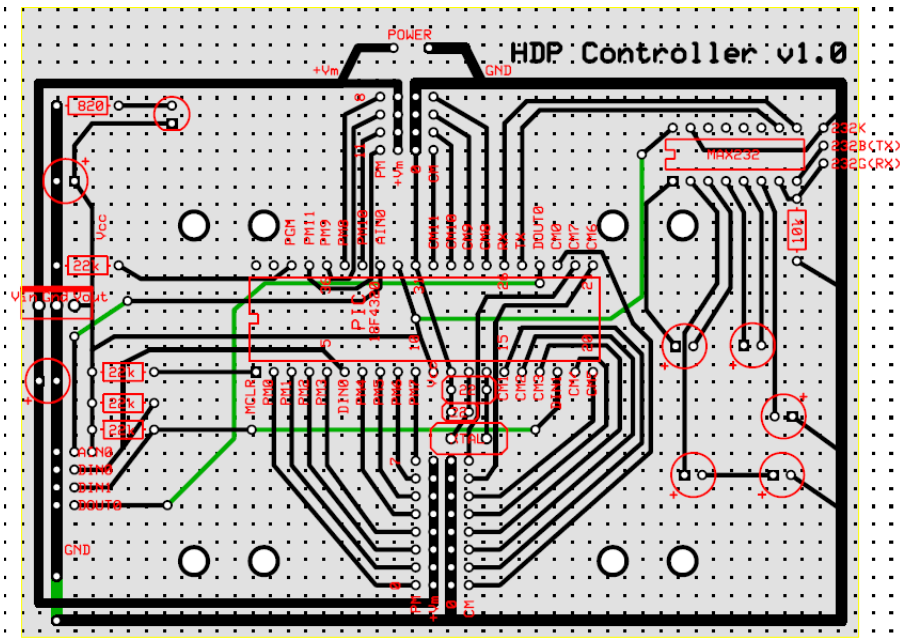


Figura 6.8: Diseño del circuito impreso del controlador

Luego de diseñar el circuito impreso siguiendo las guías de diseño y respetando el circuito implementado en el protoboard, se pasó a la producción de las placas (controladora y fuente de alimentación). Para ello se imprimió una transparencia cuyo tóner luego se transfirió a la placa mediante la aplicación del calor de una plancha e inmediatamente la sumersión de la placa en agua fría. Una vez transferido el tóner a la placa, se repasaron las pistas que no quedaron completas con un marcador indeleble, y se sumergió en un baño maría de percloruro de hierro durante 15 minutos. Luego de lavar la placa se agujerearon los hoyos para las conexiones necesarias y se soldaron los componentes. Si bien este no es un método de producción estrictamente profesional, da buenos resultados y es relativamente simple.

En la figura 6.8 se muestra la máscara que se imprimió en la transparencia. A continuación se incluye el esquemático del circuito.

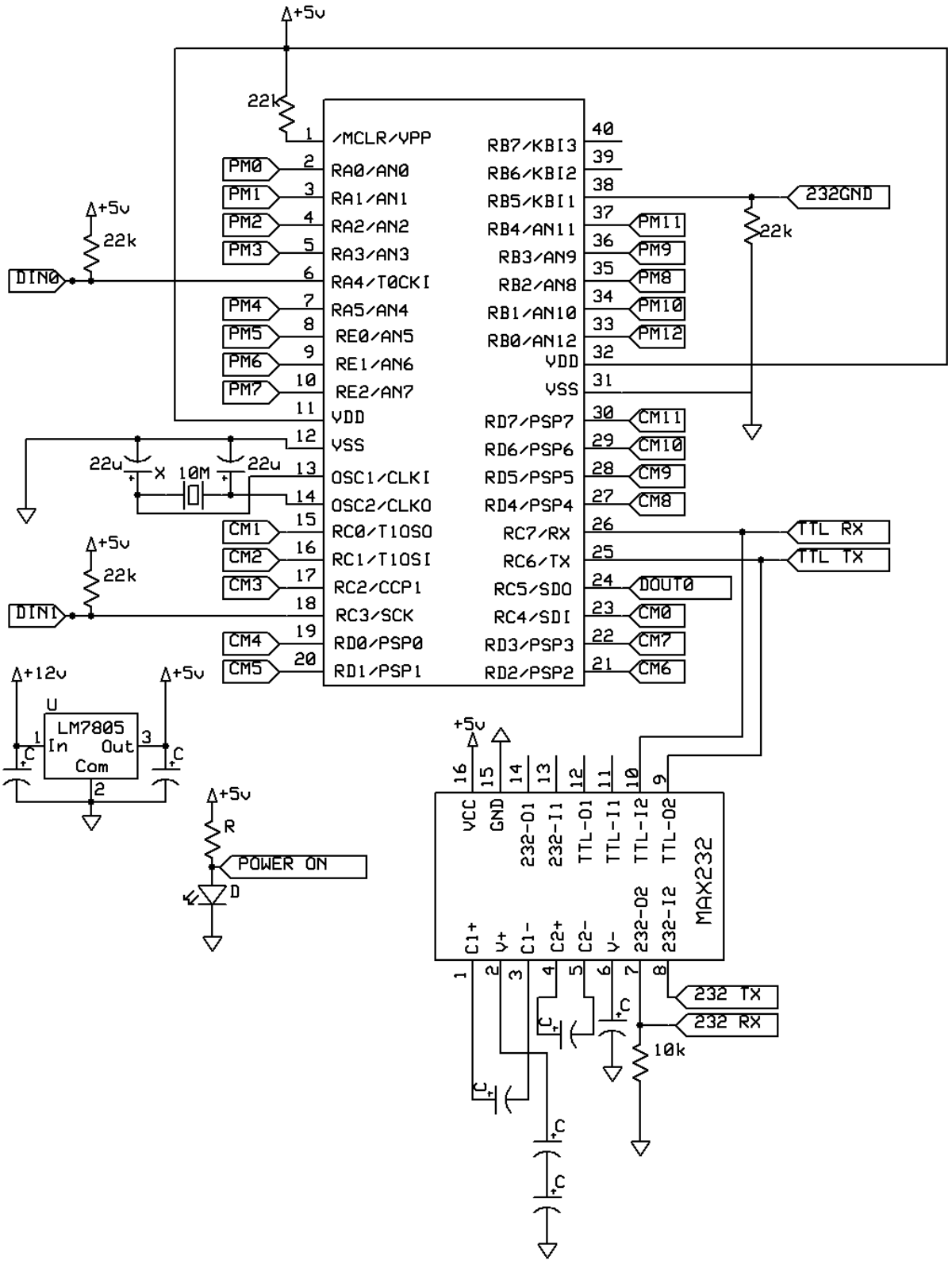


Figura 6.9: Circuito del controlador

En la figura 6.10 se muestra la placa y su conexionado.

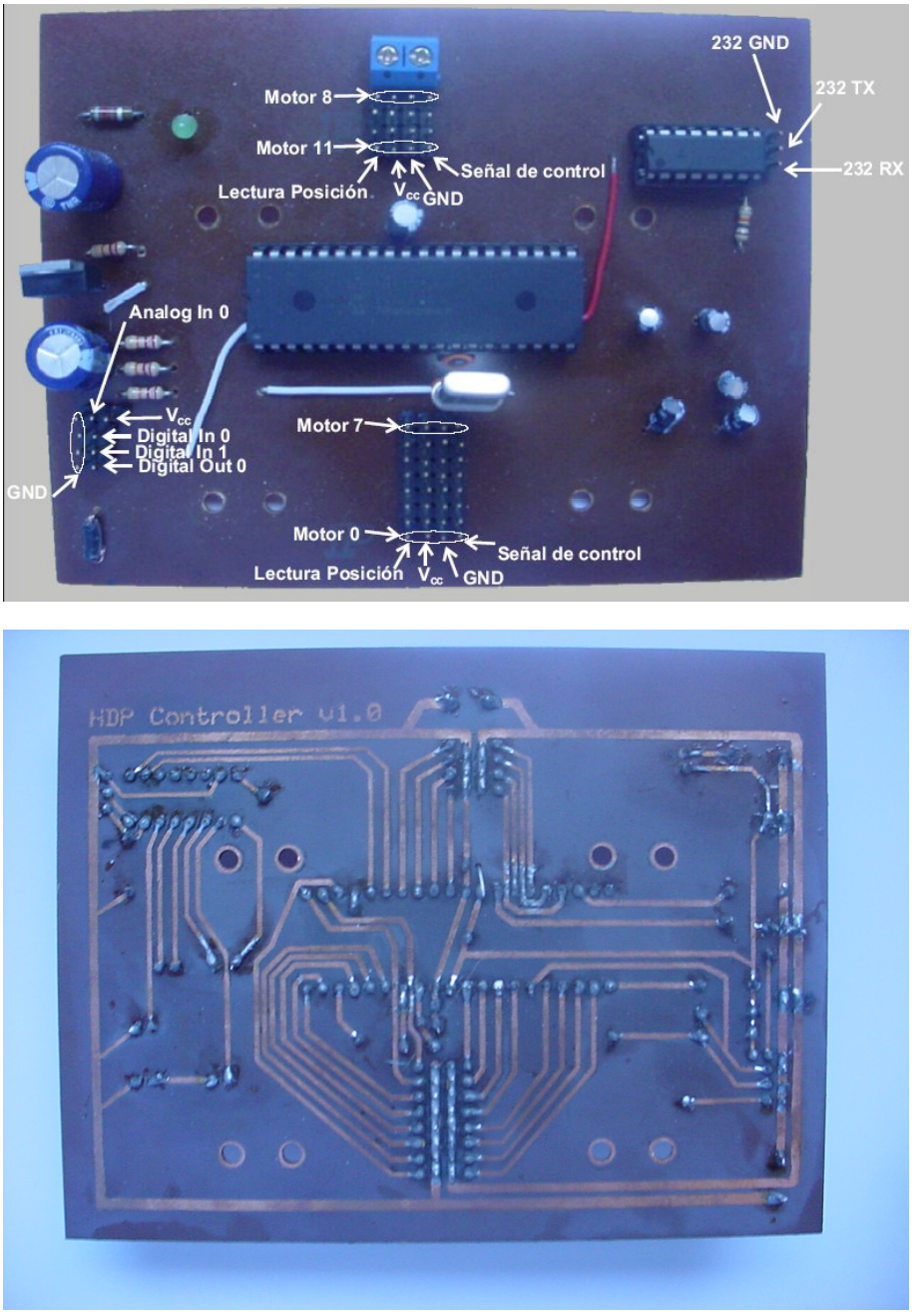


Figura 6.10: El controlador

Capítulo 7

Software integrado

En esta sección se describe el software que se ejecuta en el microcontrolador, así como la infraestructura necesaria para dicha ejecución.

Como ya se mencionó, el microcontrolador elegido para manejar los motores es el Microchip PIC 18F4320 ([MPIC]). A nivel de programación, tiene varios modos de direccionamiento (directo, indirecto, indexado) y el modo indirecto permite acceder a los distintos bancos de memoria de forma plana. Como limitaciones, el PIC es de 8 bits, posee una memoria de programa de 8 KB y cada banco de memoria datos es de sólo 256 bytes (este modelo de PIC cuenta con un sólo banco). El PIC provee un puerto USART (Universal Synchronous/Asynchronous Receiver/Transmitter) direccionable como si fuera un registro. El mismo posee un buffer de un byte.

7.1. Requerimientos funcionales

En esta sección se describen los requerimientos funcionales del software que se ejecuta en el PIC.

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.

7.2. 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. Los diagramas SDL se pueden encontrar en [LS04P].

7.2.1. Mensajes de comandos (Paquetes del PC al PIC)

Los mensajes de comandos tienen la siguiente estructura, expresada en EBNF:

MensajeComando ::= STX Posiciones Velocidades SalidasDigitales ETX
Posiciones ::= (PosicionH PosicionL)^{12}
Velocidades ::= (VelocidadH VelocidadL)^{12}

Donde:
STX es el mensaje de comienzo de transmisión (FF FF en hexadecimal)
PosicionH es el byte alto de un valor de posición de motor de 16 bits
PosicionL es el byte bajo de un valor de posición de motor de 16 bits
VelocidadH es el byte alto de un valor de velocidad de motor de 16 bits
VelocidadL es el byte bajo de un valor de velocidad de motor de 16 bits
SalidasDigitales es un byte en el cual el bit i-ésimo tiene el valor de la salida digital i-ésima.
ETX es el mensaje de fin de transmisión (FE FE en hexadecimal)

Observaciones

- Este paquete tiene un largo fijo de 53 bytes
- Si se envía un valor de velocidad de 0 para un motor, se estará deshabilitando dicho motor (es decir, el mismo no ejercerá par alguno).

7.2.2. Mensajes de notificaciones (Paquetes del PIC al PC)

Los mensajes de notificaciones tienen la siguiente estructura, expresada en EBNF:

MensajeNotificacion ::= STX ACKEstado LecturasAnalogicas LecturasDigitales ETX
LecturasAnalogicas ::= (LecturaAnalogicaH LecturaAnalogicaL)^{13}

Donde:
STX es el mensaje de comienzo de transmisión (FF FF en hexadecimal)
ACKEstado en el bit 15 de esta palabra de 16 bits se encuentra el bit de ACK (reconocimiento) de un mensaje del PC al PIC anterior. Entre las posiciones 0 a 11 se encuentran los bits que indican si el motor de esa posición ha llegado a la posición deseada.
LecturaAnalogicaH es el byte alto de la lectura analógica de 16 bits
LecturaAnalogicaL es el byte bajo de la lectura analógica de 16 bits
LecturasDigitales es un byte en el cual el bit i-ésimo tiene el valor de la entrada digital i-ésima.
ETX es el mensaje de fin de transmisión (FE FE en hexadecimal)

7.3. Descripción del software

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).

Entre las operaciones que implementa este módulo se encuentran las siguientes: add16, sub16, setBit, clrBit, y funciones de comparación (mayor, mayor o igual, menor, menor o igual). 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: $|\theta_i - \theta_i[k]| \geq \Lambda \therefore \theta_i[k+1] = \theta_i[k] + \omega_i \cdot sign\left(\theta_i - \theta_i[k]\right)$, donde θ_i es la posición final deseada del motor i-ésimo, $\theta_i[k]$ es la posición calculada en el momento k, Λ es el umbral para evitar el efecto "hunting" (es decir que no oscile el motor en torno a la posición final), y $\omega_i$es el valor de velocidad para el motor i-ésimo.

Otro módulo importante es el de motor_control. Este módulo se encarga 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.

A continuación se muestra un diagrama de bloques de la estructura del software integrado:

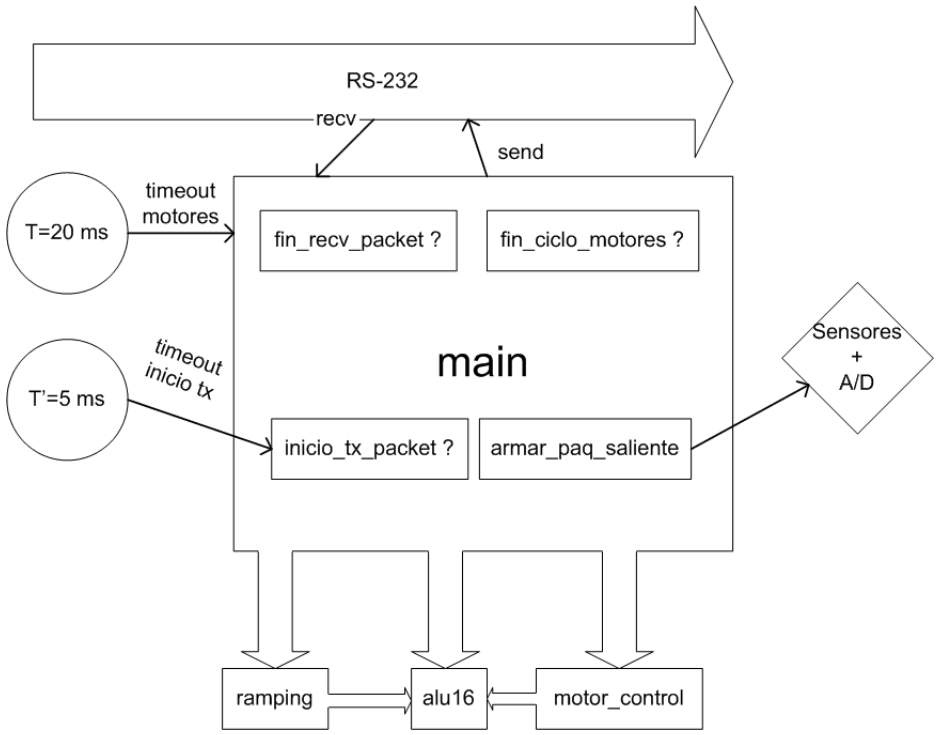


Figura 7.1: Interconexión de los módulos en el software integrado

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:

1. Recepción de un byte por el puerto serie

2. Timeout del timer de los motores
3. Timeout del timer de inicio de transmisión de paquete

A continuación se explica cada una de ellas.

7.3.1. 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.

Definimos 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.

Existe un array motoresABajar[13] cuyo registro tiene 4 bytes:

- MascaraC
- MascaraD
- Timer High
- Timer Low

El puntero dentro del array se llama indiceMAB, el tope de elementos útiles en el array es topeMAB

En el comando de los motores se ejecuta el siguiente algoritmo:

```
Si indiceMAB es 0 (principio del ciclo) subo todos los motores
Obtengo los motores habilitados del PORTC y los subo
Obtengo los motores habilitados del PORTD y los subo
Si no (adentro del ciclo)
    Cargo la mascaraC
    Hago and con PORTC
    Cargo la mascaraD
    Hago and con PORTD
Fin Si
Si hay mas motores a bajar escalo el timer 1:1 sino 4:1
Programo Timer
indice++
```

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”

7.3.1.1. 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 llegamos 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.

```
Marcar todos los motores con velocidad mayor que cero
anchoPulsoActual = 0
ptr = 0
mascaras[0..12] = motoresHabilitados
motorABajar = minimo(motores)
Mientras (motorABajar != 0xFF) {
   no considerar motorABajar en la proxima llamada a minimo(motores)
   bajar el bit motorABajar en las mascaras
   tiempo[ptr] = posicion(motorABajar) - anchoPulsoActual - RETARDO_BAJAR_MOTOR
   si (posicion(motorABajar)-anchoPulsoActual >RETARDO_BAJAR_MOTOR)
      ptr++;
   motorABajar = minimo(motores)
}
```

Este algoritmo resuelve el problema de manejar los motores.

Coordinación de los motores Además de la ejecución del algoritmo anterior, se debe ejecutar el algoritmo que realiza la técnica de Ramping para obtener la velocidad constante deseada (ver sección 3.4.2.1, apéndice F.3).

7.3.2. Recepción de un byte por el puerto serie

Cuando llega un byte a través del serial, se dispara una interrupción y la ejecución llega a una rutina que copia el byte recibido a un buffer temporal. Para ello se cuenta con un área de memoria reservada para el paquete entrante, así como un puntero que apunta a la posición donde se debe escribir el siguiente byte. Si se respeta el stop-and-wait, no hay peligro de buffer overrun, pero si no es así, no puede asegurarse el comportamiento del sistema. La otra situación en la que podría darse un buffer overrun, es si llegaran dos bytes muy juntos, tanto que la rutina que copia el byte del latch del puerto al buffer de memoria no llega a leer el primer byte, sino que lo pierde y lee el segundo. Esto no puede ocurrir ya que las rutinas están hechas de tal manera que no tardan lo suficiente para que ocurra esta situación, todo esto sin utilizar las líneas CTS (Clear to send) y RTS (Request to send), que se usan por lo general para realizar control de flujo a nivel de byte.

Cuando termina de llegar el paquete, el programa principal verifica que el STX y el ETX sean correctos. Si no lo son, se descarta todo el paquete. En caso de que sean correctos, copia el paquete

entrante a un área de memoria donde se guarda el último paquete recibido.

7.3.3. Timeout del timer de inicio de transmisión de paquete

Cuando vence un timer (configurado para 5 ms), se incrementa una variable que cuenta múltiplos de esos 5 ms. Si el valor de la variable es igual a una cierta constante definida con el valor 4, es decir, cada 20ms, se comienza el envío del último paquete armado. Para ello se levanta una bandera que hace que el programa principal cuando no tiene otras tareas más importantes que realizar, envíe el siguiente byte del paquete, hasta terminar de enviar todos los bytes. En ese momento se conmuta el valor de la bandera. El tiempo de transmisión del paquete del PIC al PC es de $t_{T,PIC \to PC} = \frac{33\,bytes}{115200\,bps} \approx 0.3\,ms$, asi que la transmisión no será un cuello de botella.

7.3.4. El programa principal

En esta sección se detalla la estructura del programa principal.

7.3.4.1. Banderas y rutinas de no interrupción

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

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

A continuación se describen cada uno de estos

Fin de recepción de paquete a través del puerto serial Cuando el puntero que apunta al último byte recibido apunta más allá del largo del paquete, el programa principal verifica la validez del paquete entrante, y en caso de ser válido, copia el paquete al área de memoria donde se almacena el último paquete recibido.

Bajada de todos los motores Cuando se detecta que todos los motores fueron bajados, se ejecuta la rutina más importante del nodo, que consiste en realizar el ramping, calcular las máscaras de bits, y escribir estas máscaras en los puertos que controlan los motores.

Inicio de transmisión del paquete saliente Cuando el sistema es interrumpido por el timer de inicio de transmisión de paquete saliente, se incrementa una variable y se compara este valor con una constante que, junto con el timer de 16 bits, conforma un timer de 24 bits. Esto permite tener tiempos más largos entre envíos. Cuando la variable es igual a la constante, se comienza a enviar el primer byte.

Armado de 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.

7.3.5. Análisis de la solución

La solución implementada resuelve todos los requerimientos descritos anteriormente. En particular permite manejar varios motores simultáneamente, con gran precisión. Para hacer el ajuste fino del sistema, fue necesario determinar las posiciones y velocidades admisibles de los servomotores, así como medir el tiempo que tarda la rutina de bajar motores. Dichos procedimientos se describen en el apéndice D.1.2.

Capítulo 8

Software del PC

8.1. Funcionalidad

Como se mencionó en las secciones 3.2 y 3.3, 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:

1. 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.

2. 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.

3. Capturar (digitalizar) la pose en la que se encuentra el robot actualmente, y guardarla para su futura utilización.

4. Definir un andar (gait). Un andar es una secuencia de instrucciones, donde una instrucción puede ser una de las siguientes:

   a) 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.

   b) Repetir un conjunto de instrucciones un número finito de veces. Esto se especifica con las instrucciones Repeat [n] y EndRepeat.

   c) 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.

5. 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 9.3.4). Notar que debido a esta flexibilidad, es posible por ejemplo definir más instrucciones de ser necesario.

En el CD se entrega un archivo que contiene la calibración de los motores, un andar para una caminata cuasiestática de cuatro pasos, y una cantidad de poses que fueron diseñadas con la finalidad de implementar dicho andar. Por información sobre la utilización del software, ver [LS04M].

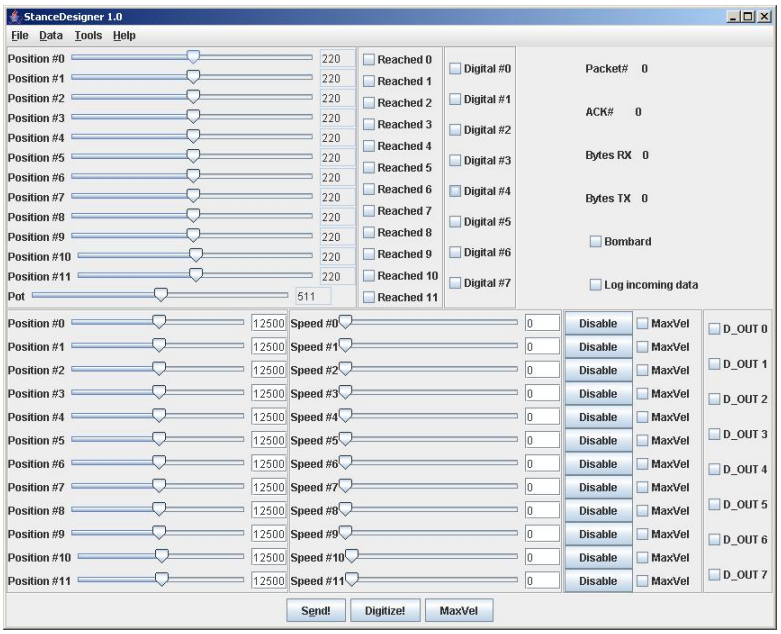


Figura 8.1: Pantalla principal del software del PC

8.2. Arquitectura

En la presente sección se describe la arquitectura del software que se ejecuta en el nodo PC. La arquitectura descrita excede el alcance del Proyecto actual y está planteada de forma de ser implementada en su totalidad por futuros proyectos. Esto significa que el proyecto actual construirá la parte de la arquitectura necesaria para cumplir con el alcance del proyecto.

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 (ver sección 8.3). 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.

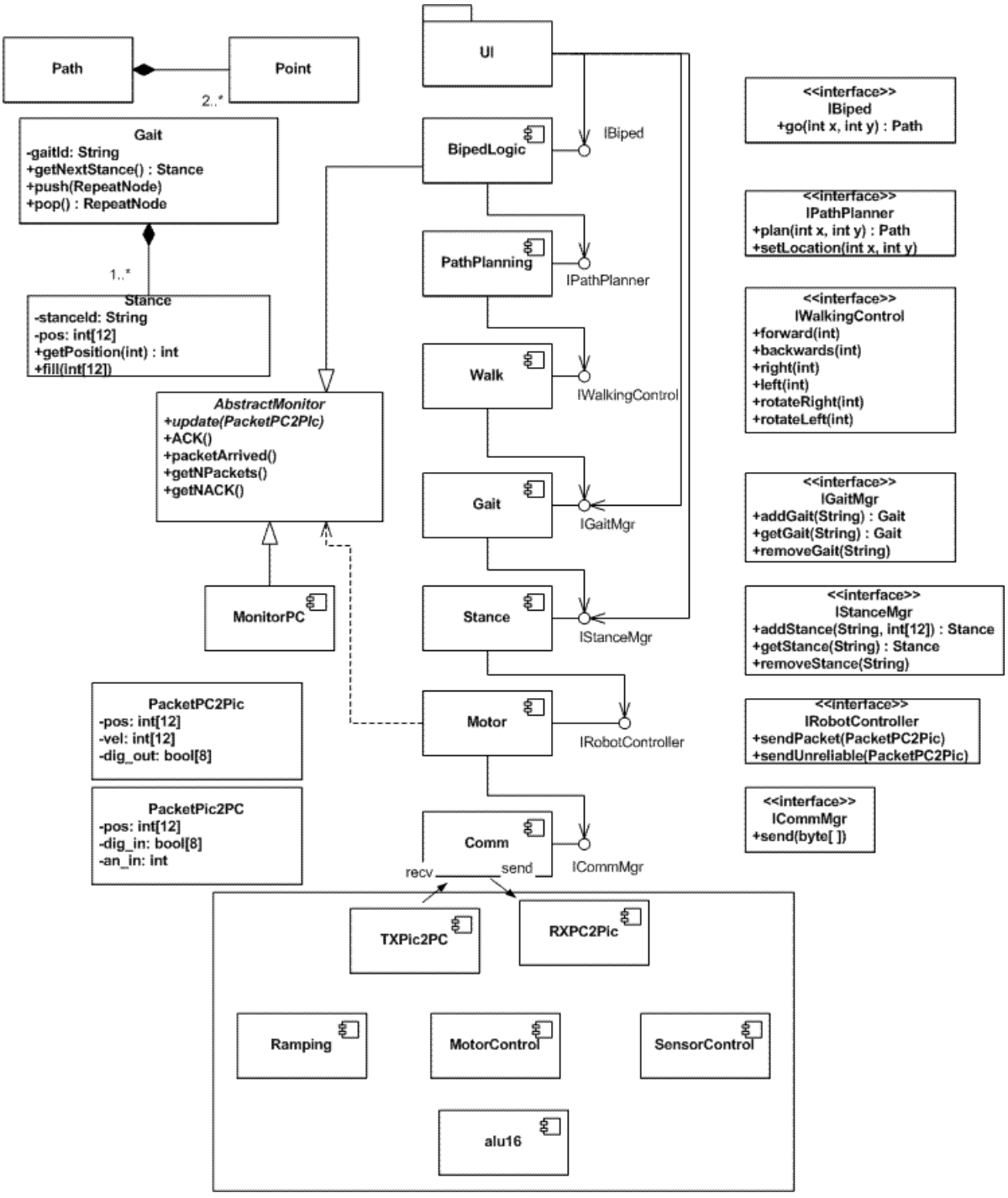


Figura 8.2: Diagrama de la arquitectura del software del PC

8.2.1. 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. La

operación de envío es sincrónica y la de recepción es asincrónica (por notificación, según el patrón de diseño Observer). 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).

8.2.2. motor

Este componente es el que conoce el formato de los paquetes que se envían desde el PIC al PC y viceversa. El mismo recibe un objeto PacketPC2Pic y lo envía ya sea de manera insegura (confiando en el envío de las capas inferiores) o de manera segura (esperando el bit de reconocimiento ACK, en algún paquete subsiguiente entrante). Además los componentes que deseen recibir notificaciones acerca de llegadas de paquetes desde el PIC, deben registrarse con el componente que realice la interfaz IRobotController. Dichos componentes deben extender clase AbstractMonitor.

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).

8.2.3. 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. A este efecto utiliza los servicios de la capa motora.

8.2.4. 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.

8.2.5. 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.

8.2.6. 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".

8.2.7. 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.

8.2.8. 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. Por último, existen clases que ayudan a implementar el patrón Observer. Esto se utiliza para realizar la actualización automática de las ventanas que listan las poses y andares. Este patrón está implementado por dos componentes: un INotifier, que despacha notificaciones de un cierto tipo hacia todos los observadores que tiene registrados para ese tipo, y varios IObserver que implementan la lógica a ejecutar bajo notificaciones. Cuando se ejecutan operaciones del sistema de interés, se invoca el despacho de notificaciones de un cierto tipo. Por ejemplo, si tenemos una ventana que queremos actualizar cuando haya un cambio en el subsistema de poses, haremos lo siguiente:

```
class Ventana extends JFrame {
   IObserver o;
   public Ventana (...) {
   INotifier i = ...
   i.addObserver(o = new IObserver() {
      public void notifyUpdate(Object o) {
         Ventana.this.actualizar(...)
      }
    /* cuando se cierra la ventana hay que
    llamar a i.removeObserver(o) */
   }
}
```

8.2.9. factory

Por último, este componente provee algunas operaciones que son utilizadas por las diferentes capas. En esta capa es donde se exportan las interfaces de los manejadores de cada capa, así como también donde se obtiene el singleton del factory que permite obtener los primeros. Además, aquí reside una clase de utilidades varias llamada Utility, donde se define la cantidad de motores a manejar, etc.

8.3. Configuración

En el archivo properties.xml se definen los nombres de las clases que realizan las interfaces de interés para la aplicación. Las mismas son el controlador de comunicaciones (CommMgr), que normalmente estará implementado por la clase comm.CommMgr, y el controlador motor (RobotController) que normalmente estará implementado por la clase motor.MotorCommand. Para poder desarrollar sin tener el robot *in situ*, se implementó un stub del robot, que reside en la clase motor.MotorStub. La misma instancia stubs del PIC y simula una comunicación con un nodo PIC virtual.

Capítulo 9

Experiencias realizadas y conclusiones

9.1. Experiencias realizadas

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. Lo primero que realizamos cuando contamos con el sistema completo funcional es la calibración de los servomotores, utilizando el software de control desde el PC ([LS04M]). El proceso de calibración asocia valores de posición leídos de los motores con los enviados a los mismos. Con los motores calibrados, estando los servomotores desactivados (velocidad 0) pusimos manualmente al robot en la posición deseada y la digitalizamos. Luego enviamos al robot a dicha posición observando el error cometido y apreciamos que es muy pequeño, el robot se queda prácticamente donde lo pusimos.

Este error se origina como consecuencia de pequeñas diferencias en la estimación de la calibración: por un lado, usamos una calibración con un modelo lineal, y por otro lado, la calibración se hace tomando en cuenta solamente dos puntos. Si el comportamiento real del motor se desvía un poco de la linealidad, aparecerán errores, pero hemos constatado que esto no es el caso general. Por el contrario, las fuentes más comunes de errores son tres: la primera y más grave es cuando en el proceso de calibración el motor se topa con algún obstáculo, hecho que introduce errores en la lectura de posición que se utiliza para calibrar el motor. Esto hace que los parámetros de calibración difieran en gran medida de los reales.

La segunda fuente de error está relacionada al efecto que tiene la corriente consumida por un motor en la lectura de posición del mismo. Cuando un motor está consumiendo mucha corriente (es decir que está ejerciendo un par grande), la lectura de posición no es buena. Esto se debe presumiblemente a efectos inductivos de la corriente de alimentación sobre la señal que está siendo leída, pero no hemos podido determinar con certeza esta afirmación.

La última fuente y la menos grave es que, como hemos comprobado, el parámetro de la pendiente tiene un valor elevado, y esto puede hacer que una mínima variación en la lectura de posiciones genera una variación grande en el comando de posiciones. Esto es lo que ocurre con

mayor frecuencia, primero porque la medición de posición no es de gran precisión: en primer lugar, se utiliza la lectura del potenciómetro directamente. La posición del eje del motor, esta sometida a deformaciones que no están representadas en la posición del potenciómetro (la retroalimentación por la lectura del potenciómetro en este caso decimos que es de lazo semiabierto). Además, esta señal pasa por una conversor A/D de 10 bits, de los que se usan 9 para los sensores de los motores, ya que la referencia de voltaje del conversor está en 2.5 V. Esto hace que la lectura oscile en torno de la posición real en un rango de aproximadamente ± 2 unidades, que corresponden a $V_{ref} \times \frac{\Delta\theta}{\delta\theta} = 2.5\,V \times \frac{2-(-2)}{2^{10}} = 2.5\,V \times \frac{4}{1024} \approx 10\,mV$ (pico a pico), lo que es totalmente razonable que ocurra en estas condiciones. Para clarificar este punto, supongamos calibramos los motores llevándolos a posiciones comandadas $c_2 = 10000$ y $c_1 = 8000$, y que las lecturas en dichas posiciones valen $s_2 = 200 + \epsilon_2$ y $s_1 = 150 + \epsilon_1$. Examinaremos qué pasa al variar las lecturas (es decir, cuando ϵ_1 y ϵ_2 son no nulos) y nos concentraremos en el peor caso. La pendiente a de la recta de calibración la calculamos como $a = \frac{c_2 - c_1}{s_2 - s_1} = \frac{10000-8000}{200+\epsilon_2-150-\epsilon_1} = \frac{2000}{50+\epsilon_2-\epsilon_1}$. Hay dos peores casos: cuando $\epsilon_2 - \epsilon_1 = 4$ y cuando $\epsilon_2 - \epsilon_1 = -4$. Llamemos a los a correspondientes $a_{mn} = \frac{2000}{54} \approx 37$ y $a_{mx} = \frac{2000}{46} \approx 43.5$ (la pendiente cuando no hay oscilaciones sería $a_0 = \frac{2000}{50} = 40$.

Las ordenadas en el origen correspondientes serían $b_{mn} = c_2 - a_{mn} \cdot s_2 = 10000 - 37 \cdot 202 = 2526$, $b_{mx} = c_2 - a_{mx} \cdot s_2 = 10000 - 43.5 \cdot 198 = 1391$ y $b_0 = c_2 - a_0 \cdot s_2 = 10000 - 40 \cdot 200 = 2000$. Esto hace que los parámetros tengan una cierta variabilidad dependiendo de que la lectura oscile o no. En la siguiente figura pueden verse las tres rectas (ideal, pendiente máxima y pendiente mínima).

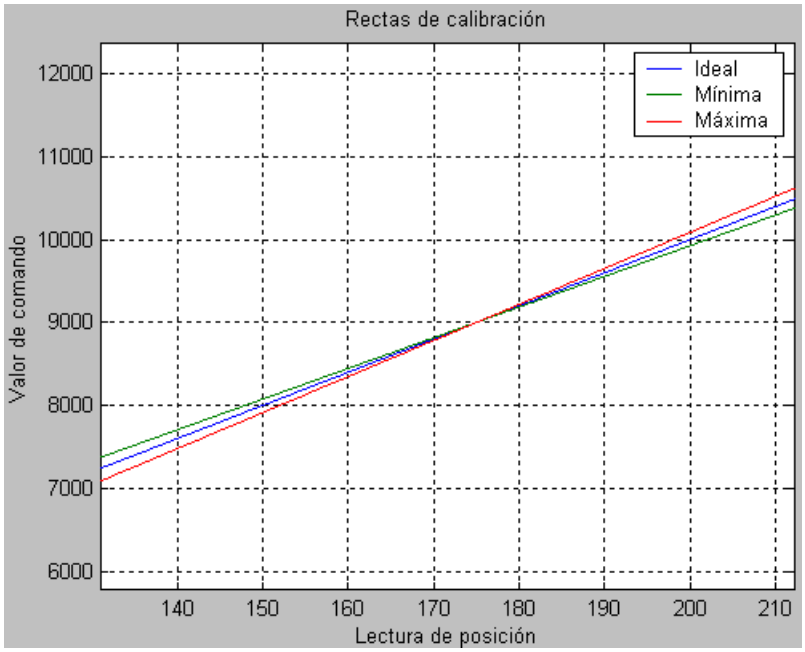


Figura 9.1: Variabilidad de las rectas de calibración

Luego de contar con muchas posiciones comenzamos a diseñar movimientos completos que las usan. También se hicieron pruebas con las entradas y salidas digitales y la entrada analógica. Para ello contamos con botones, un potenciómetro y un LED, todos soldados a los conectores apropiados.

Luego de comprobar la funcionalidad del sistema, nos dispusimos a hacer que el robot caminara. Para ello diseñamos posiciones similares a las simuladas. Nos enfrentamos a varios inconvenientes al intentar cumplir este objetivo. Primero, como se describió en el capítulo 5, 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.

Lamentablemente 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. Es de esta manera que logramos que el robot caminara satisfactoriamente.

Enfrentados al robot real aparecen muchos problemas que no teníamos en la simulación, muchos de ellos debidos a un funcionamiento poco adecuado de los servomotores cuando se acercan al límite de torque. Incluso experimentamos la rotura de los engranajes de uno de los servomotores, engranajes que en este modelo de servo son de plástico.

Otro elemento nuevo, que influye en el equilibrio, es el peso y disposición de los cables. El robot se conecta mediante un cable al puerto serial del PC y otro cable lo alimenta de corriente. Este último cable es el más pesado y si se orienta de distinta forma puede hacer que una posición se vuelva inestable.

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 3.4.3.1 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.

9.1.1. Andar "*Cuatro pasos*"

Como se comentó, se diseñó un andar a través del cual el robot puede dar cuatro pasos seguidos, partiendo y terminando en la posición de Erguido. Con el propósito de ilustrar este andar se registraron los datos de los sensores correspondientes a las posiciones de los motores en un archivo de bitácora. Este archivo contiene los datos de los trece sensores, separados por tabuladores entre sí y por fin de línea entre lectura y lectura. Los datos presentes en este archivo (caminata.dat) fueron cargados y procesados en MATLAB. Primero se separaron los datos por sensor. Esto permite poder procesar cada articulación por separado. Debido a las oscilaciones de los datos, así como también a la resolución de los sensores (9 bits útiles), y la utilización de un conversor A/D que produce números enteros y no de punto flotante, fue necesario filtrar las señales. Para ello se uso un filtro pasabajos digital de Butterworth de orden 2 y frecuencia de corte digital 0.1. Ya que el PIC transmite 50 paquetes por segundo, y estos se registran en la bitácora a razón de 1 de cada 5, la "frecuencia de muestreo" es de 10 Hz. Esto hace que la frecuencia de corte sea de $0.1\ \frac{f_s}{2} = 0.5$ Hz, es decir una variación de 2 valores por segundo. La respuesta en frecuencia de dicho filtro se muestra a continuación:

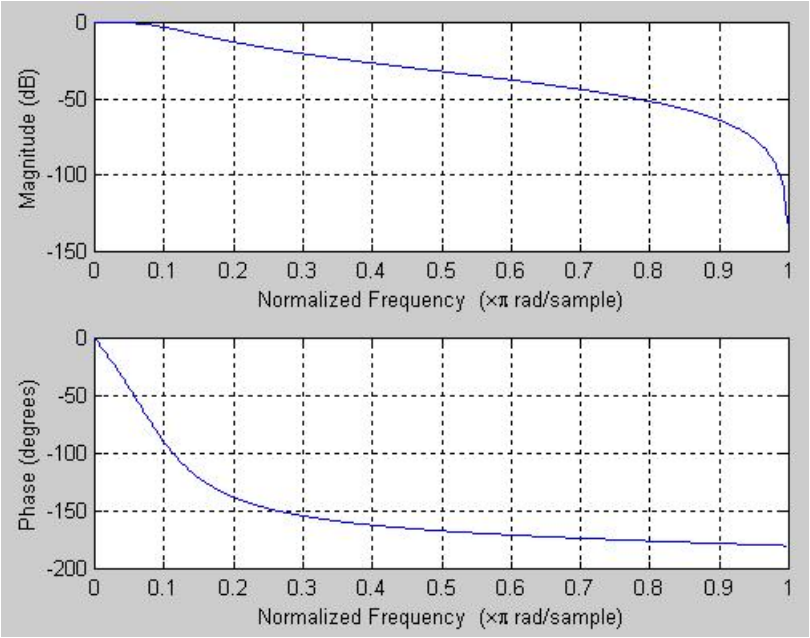


Figura 9.2: Respuesta en frecuencia del filtro utilizado

En las figura 9.3 a 9.8 se grafican las articulaciones de ambas piernas simultáneamente, para cada grado de libertad. Debido a que no todas las articulaciones están dispuestas de la misma forma, el centro de una articulación puede estar en en valor de lectura de 200 mientras que el de otra puede estar en 100. Para mejorar la claridad de la representación, lo que se hizo fue hacer coincidir valores cero para cada articulación de cada pierna. Si llamamos θ_I a la articulación de la pierna izquierda y θ_D, y θ_{I_0} y θ_{D_0} a sus valores iniciales, hacemos:

$$\begin{cases} \theta'_I = \theta_I - \theta_{I_0} + \mu \\ \theta'_D = \theta_D - \theta_{D_0} + \mu \end{cases}$$, donde $\mu = \frac{\theta_{I_0} + \theta_{D_0}}{2}$ y θ'_I y θ'_D las nuevas trayectorias.

Esto es lo mismo que $\begin{cases} \theta'_I = \theta_I - \frac{\Delta}{2} \\ \theta'_D = \theta_D + \frac{\Delta}{2} \end{cases}$, donde $\Delta = \theta_{I_0} - \theta_{D_0}$.

A continuación se grafican las trayectorias (recentradas) de los motores en función del tiempo al utilizar un andar de caminata de cuatro pasos (un video del mismo se encuentra disponible en el CD y en la página web). Observar que las trayectorias están bastante correlacionadas: Salvo en el eje lateral del tobillo, donde los movimientos amplios ocurren a la vez (cuando se cambia el peso de un costado al otro), todos los demás motores presentan un mismo patrón, con la particularidad que el motor izquierdo y el derecho se encuentran defasados medio período (es decir, un paso). Esto se aprecia claramente, por ejemplo, en el caso de la rodilla. Algunos motores tienen la lectura invertida porque por construcción giran al revés al realizar un mismo movimiento.

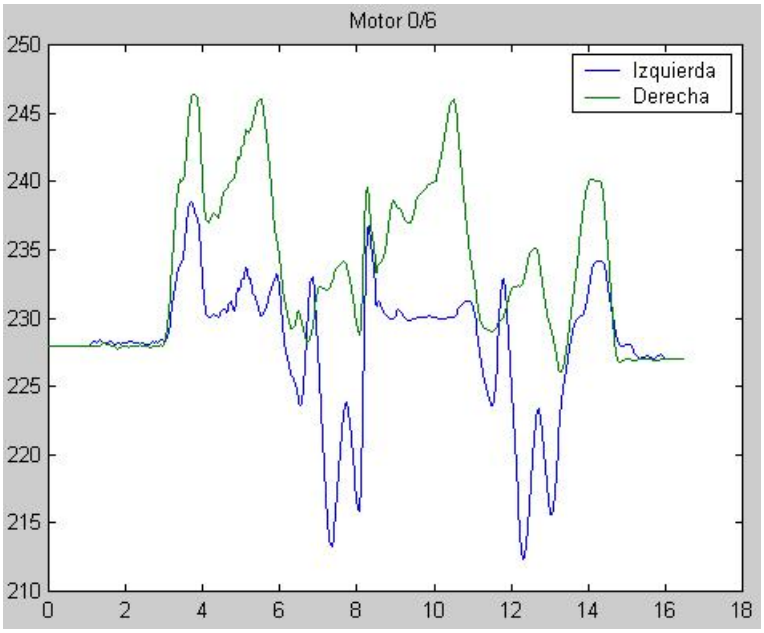


Figura 9.3: Motores 0 y 6 (Cadera)

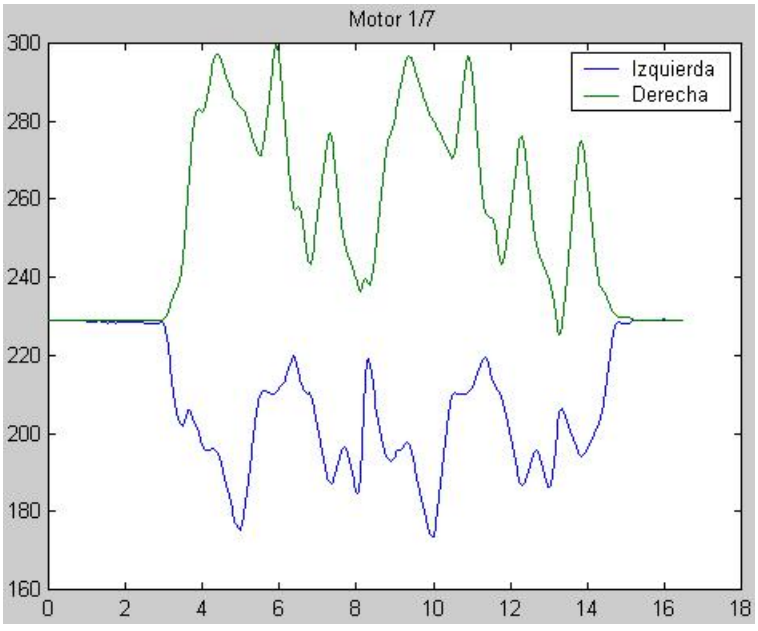


Figura 9.4: Motores 1 y 7 (Cadera)

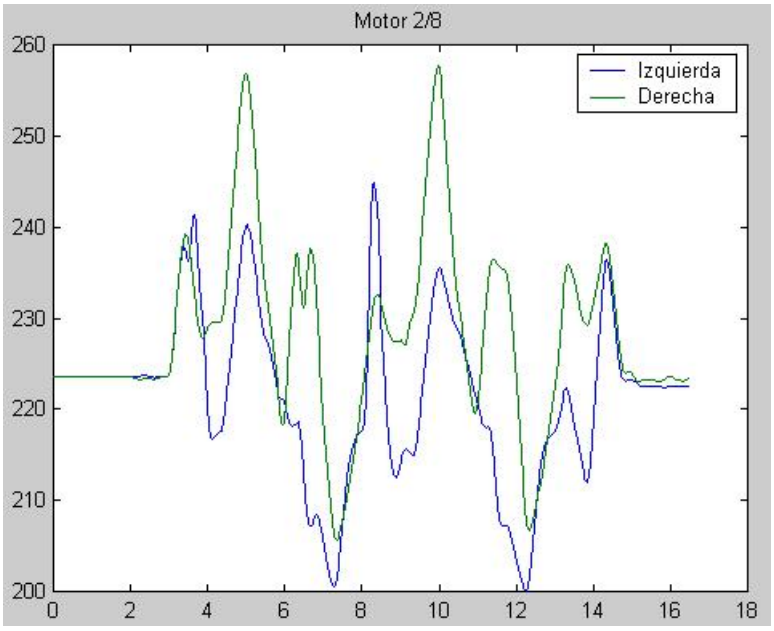


Figura 9.5: Motores 2 y 8 (Cadera)

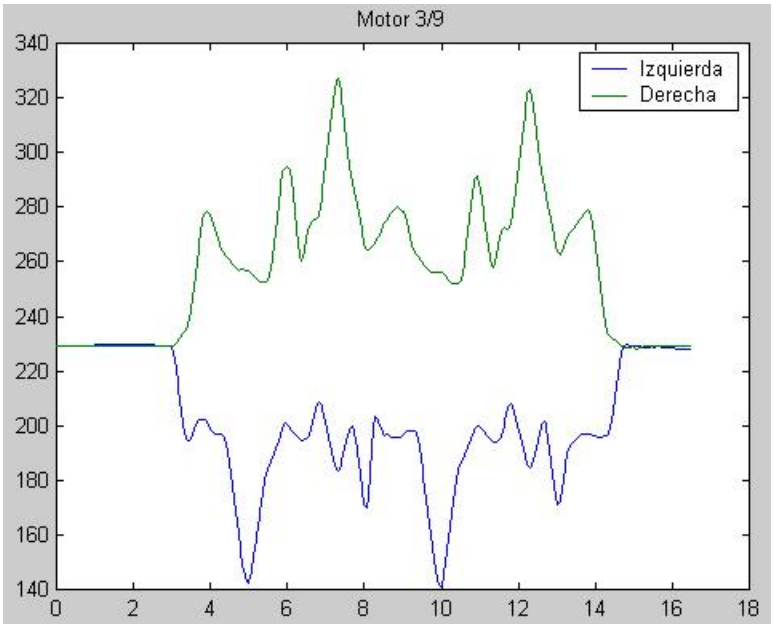


Figura 9.6: Motores 3 y 9 (Rodilla)

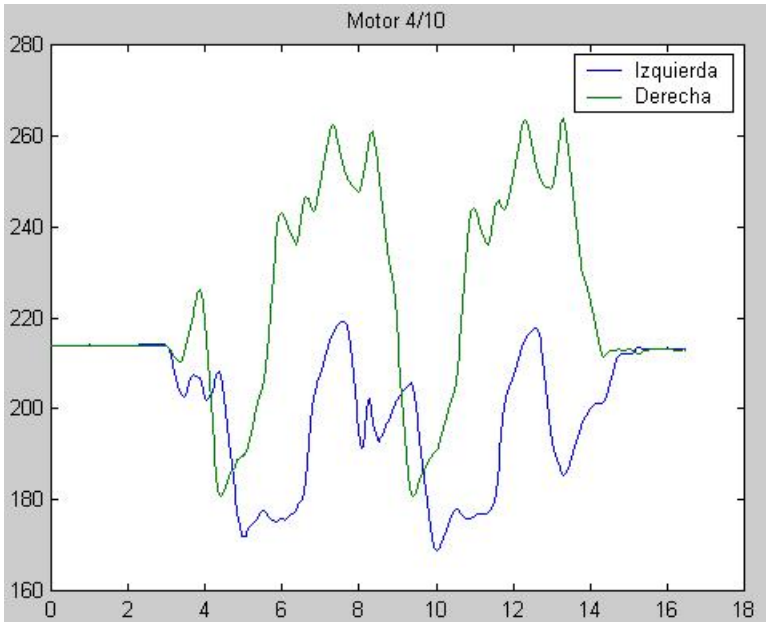


Figura 9.7: Motores 4 y 10 (Tobillo)

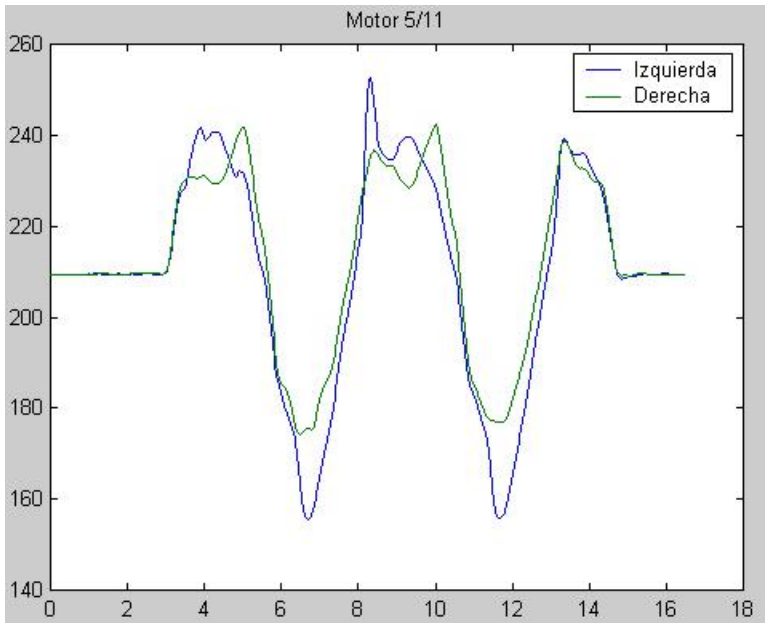


Figura 9.8: Motores 5 y 11 (Tobillo)

9.2. Evaluación de resultados

Nuestro Proyecto 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 IK (apéndice E), u alguna técnica de control. También consideramos 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. De todas formas, a la luz del muy bajo presupuesto del que disponemos y de nuestra inexperiencia en el área mecánica, el resultado es muy bueno.

De nuestra experiencia en este Proyecto rescatamos también algunas conclusiones que exponemos a continuación:

- 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. Como vimos en la sección 2.2.6, los proyectos de investigación que se presentan a eventos como la RoboCup llevan robots con costos oscilando entre los U$S 5.000 y U$S 10.000, aproximadamente. Una forma de trabajar con un presupuesto un poco menor, es restringir el estudio a algún punto particular y construir un robot que sirva apenas para investigar en el tema elegido. Se puede tomar como referencia nuestro trabajo: la construcción de nuestro robot como la planteamos en el trabajo a futuro de la sección 9.3.1, calculamos que tendría un costo entre U$S 1.000 y U$S 1.200. Debe agregarse que para realizar trabajos de investigación con este robot, es probable que sea necesario ampliarlo en algún sentido, lo que aumentaría el costo.

9.3. Trabajos a futuro

9.3.1. 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. Existe gran variedad de servomotores de hobby, variando el precio entre U$S 10 y U$S 100 en origen, que pueden ser utilizados en el robot sin necesidad de

modificar nada del sistema. Nosotros utilizamos el más barato, debido a nuestro limitado presupuesto, pero una opción que evaluamos como muy adecuada es el servomotor Futaba S3305. El mismo duplica el torque de nuestro S3003 y cuenta con engranajes de metal y dos rulemanes en el eje de salida. El precio del S3305, en el momento de escribir este informe, es el triple que el del S3003, costando U$S 35 en origen. Es necesario tener en cuenta que los precios en nuestro país duplican aproximadamente los precios en origen.

Contándose con un robot construído con estas características, se abrirían una serie de posibles trabajos que se exponen en las siguientes secciones. Este trabajo, dadas sus características, es adecuado para las áreas de Ingeniería Industrial y Diseño Industrial. 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 Si bien el aluminio funciona de manera satisfactoria para nuestro prototipo, su vida útil se ve afectada por deformarse con relativa facilidad.

Amortiguamiento de las oscilaciones Para mejorar el tiempo de establecimiento de las articulaciones (es decir, reducir el tiempo que tarda en desaparecer una oscilación y en estabilizarse el robot), habría que incorporar elementos mecánicos, como ser resortes y amortiguadores.

Protección contra caídas Para prolongar la vida del prototipo sería importante protegerlo contra caídas utilizando algún material extremadamente liviano pero que lo amortigüe (por ejemplo, alguna espuma tipo telgopor).

Motores Para poder obtener mayores pares sería necesario incorporar servomotores de mayor potencia. Esto estaría justificado en las articulaciones que necesitan más par, como lo son el tobillo y la articulación en la cadera que levanta la pierna hacia adelante.

Separación de las señales de la alimentación Debido a que las corrientes relativamente grandes de la alimentación pueden causar cierta inducción en los cables de señales en forma de interferencia, es deseable separar estos dos conjuntos de conductores.

Fuente conmutada en lugar de regulada Las fuentes conmutadas, si bien son más complejas de realizar, presentan una eficiencia mucho mayor que las reguladas, además de ser un desafío de diseño por si mismas.

Protección contra cortocircuitos en la fuente La fuente que desarrollamos, no contiene protección contra cortocircuitos. Esto puede mejorarse utilizando diodos flyback, fusibles u otros elementos. Además es deseable dotar la fuente de un fusible.

Lectura de posiciones Debido a que el PIC realiza la lectura de posiciones con un voltaje de referencia para la conversión de 2.5 V, se puede mejorar la precisión de la lectura de posiciones, ubicando un amplificador con una ganancia de 2 en la lectura de los motores, y acondicionando la señal con un filtro pasabajos de frecuencia de corte baja.

9.3.2. Construcción de un robot humanoide

Nuestro proyecto comenzó el estudio sobre robótica bípeda construyendo un robot de dos patas pero sin cuerpo articulado o brazos. 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.

9.3.3. Estudios sobre locomoción bípeda

Si contamos con 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. Este proyecto puede ser realizado sobre el robot original (sería necesario ampliar la plaqueta de control para incluir los nuevos sensores que sean necesarios), pero es más interesante si se trabaja sobre el robot humanoide. Este es un proyecto adecuado para el trabajo en el área de física, matemáticas y computación pero también puede ser necesario (o al menos deseable) contar con colaboración por ejemplo de biólogos y otros científicos que puedan aportar sobre la locomoción en los seres vivos.

9.3.4. Trabajos a más alto nivel

Si bien el Proyectó logró sus objetivos de forma satisfactoria, es decir, se construyó un robot de dos piernas que es capaz de caminar, 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.

Este proyecto depende de que se cuente con el robot construido que se plantea en la sección 9.3.1, pero puede ser encarado independientemente de otros, si bien seguramente sea necesario para el mismo contar con mejor retroalimentación de sensores, esto se puede lograr adosando por ejemplo cámaras cuya interfaz no tiene por que pasar por la misma placa de control (ni la

capacidad del puerto serial lo permitiría). También las cámaras pueden ser externas al robot como en el caso de algunas categorías de fútbol de robots.

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.

Bibliografía

[AIDE] AI Depot. http://ai-depot.com

[BC92] Randall Beer & Hillel Chiel, "A Biological Perspective on Autonomous Agent Design", *Journal of Robotics and Autonomous Systems 6*, pp. 169-186, reimpreso en *New Architectures for Autonomous Agents*, MIT Press, 1992.

[BU04] Samuel R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Square methods", 2004.

[BU00] Lee Buse, "Variable Speed Modification to the Futaba S3003 RC Servo", http://www.seattlerobotics.org/encoder/200009/S3003C.html, 2000.

[DU02] J. Duysens, "Human gait as a step in evolution", *Brain, Vol. 125, No. 12, pp. 2589-2590*, Diciembre 2002, Oxford University Press, http://brain.oupjournals.org/cgi/content/full/125/12/2589.

[EIO] EIO, "Stepper motors Part 1", http://www.eio.com/StepperMotors1.pdf

[FIRA] Federation of International RoboSoccer, http://www.fira.net

[FM04] Davide Faconti, Giuseppe Menga, "Isaac: Theoretical Analysis and Implementation of a humanoid Robot", http://www.isaacrobot.it, 2004

[FUTA] Futaba, http://www.futaba-rc.com

[HVWT] HVW Technologies, "Modifying the FP-S148 for Continuous Rotation", http://www.hvwtech.com/appnotes/servos_mod_fut.asp

[IGOE] Tom Igoe, "Motors", *Curso Physical Computing, NYU*, http://stage.itp.nyu.edu/~tigoe/pcomp/motors.shtml

[IK03] Alexander Iakovlev & Alexei Kritchoun, "ARNE robot, Team Description Paper", *RobotCup 2003*, 2003.

[JO98] Douglas Jones, "Control of Stepping Motors", http://www.cs.uiowa.edu/~jones/step, 1998. Reimpreso en *Handbook of Small Electric Motors*, secciones 5.2.10, 10.8, 10.9, 10.10, editado por Yeadon & Yeadon, McGraw-Hill, 2001.

[JUIC] Juice, http://www.natew.com/juice

[KA00] Teresa Ge Kang, "Solving Inverse Kinematics Constraint Problems for Highly Articulated Models" (Tesis de Maestría), http://www.cgl.uwaterloo.ca/~ktge/thesis_pre/thesis.pdf, 2000.

[LS04W] Damián Lezama & Alexander Sklar, "Proyecto de Grado - Construcción de Robots Bípedos", http://www.fing.edu.uy/~pgrobip, 2004

[LS04D] Damián Lezama & Alexander Sklar, "Proyecto de Grado Construcción de Robots Bípedos, Hoja de datos del prototipo", 2005.

[LS04M] Damián Lezama & Alexander Sklar, "Proyecto de Grado Construcción de Robots Bípedos, Manual del Software de Control", 2005.

[LS04P] Damián Lezama & Alexander Sklar, "Proyecto de Grado Construcción de Robots Bípedos, Protocolo de Comunicación", 2005.

[MS01] Josep MPorta & Enric Selaya, "Efficient Gait Generation using Reinforcement Learning", 2001.

[MG90] Tad McGeer, "Passive dynamic walking", *International Journal of Robotics Research*, Vol. 9, No., 2, pp. 62-82, 1990.

[MM95] McMillan, "Dynamechs", http://dynamechs.sourceforge.net, 1995.

[MICR] Microchip, http://www.microchip.com

[MPIC] Microchip, "PIC 18F2220/2320/4220/4320 Data Sheet", http://ww1.microchip.com/downloads/en/DeviceDoc/39599c.pdf

[NITI] Nitinol, http://www.nitinol.com/3tech.com

[ODE] Rusell Smith, Open Dynamics Engine, http://www.ode.org

[ODEU] Open Dynamics Engine, Manual de usuario, http://www.ode.org/ode-latest-userguide.html

[PRI] Primatics (Conversión de unidades), http://www.primatics.com/units.htm

[RC05] Robocup, http://www.robocup2005.org

[WEEI] Web-EE, "Adjustment free Inclinometer operates on +2.7V", http://www.web-ee.com/Schematics/Inclinometer/Inclinometer.htm

[WT92] D. Wettergreen & C. Thorpe, "Gait Generation for Legged Robots", *Proceedings of the IROS '92 Conference, Vol. 2, Jul 1992*, pp. 1413 - 1420.

[WEPA] Worldwide Electroactive Polymer Actuators, http://ndeaa.jpl.nasa.gov/nasa-nde/lommas/eap/EAP-web.htm

Apéndice A

Recursos utilizados

A.1. Recursos de Software

Producto	Versión	Sistema Operativo	Licencia	Descripción
LyX	1.3.4	Linux	GPL	Editor LaTeX WYSIWYM
MPLAB IDE	6.50	Windows		IDE para PIC
MPLAB C18		Windows		Compilador C para PIC
Broccoli		Linux		Programador de PIC
IC-Prog		Windows		Programador de PIC
g++		Linux	GNU	Compilador C++ para Linux
Eclipse IDE	2.0	Windows	Eclipse	Entorno de desarrollo
Java SDK	1.5	Windows	Sun	Kit de desarrollo
JavaComm	1.0	Windows	Sun	Biblioteca de comunicación RS-232
Redhat Linux	9			Sistema operativo
Windows XP	SP2			Sistema operativo
ExpressPCB	4.2.2	Windows		Diseño de circuitos impresos
ODE	0.5	Linux/Windows	BSD	Simulador de sistemas de cuerpos rígidos

A.2. Recursos de Hardware

Producto	Versión/Modelo	Descripción
Fuente de PC	N/A	Fuente de alimentación de PC de 300W
PIC	18F4320	Microcontrolador
MAX232	N/A	Conversor TTL↔RS-232
PC	Pentium IV	Estación de trabajo de desarrollo y producción
Motores	Futaba S3003	12 Servomotores

A.3. Resumen de gastos en el prototipo

Descripción	Cantidad	Precio unitario (U$S)	Total (U$S)
Componentes electrónicos discretos			20
Servomotores	12	22	264
Aluminio	30x50 cm		4
Tornillos			4
Placa PCB	2	1	2
Ácido para el PCB	1	3	3
Transparencias	2	1	2
Cables y Conectores			20
Total			319

Apéndice B

Glosario

En esta sección definiremos algunos términos básicos para el entendimiento del material contenido en este Informe.

Actuador: Dispositivo que realiza trabajo mecánico a partir de cierto estimulo, por lo general eléctrico (motor eléctrico).

A/D: Analógico/Digital (conversor).

Autómata: Un mecanismo automatizado.

Autónomo: Que no precisa de alimentación externa.

Androide: Un autómata que parece humano.

Bípedo: De dos piernas o patas.

COG: Center of gravity (centro de gravedad), ver COM.

COM: Center of mass (centro de masa). Definido como $\vec{r_0} = \frac{\int\int\int \vec{r} dm}{\int\int\int dm}$, donde $\vec{r}$ recorre los puntos del cuerpo y dm es el elemento de masa.

COP: Center of pressure (centro de presión).

DOF: Degrees of freedom (grados de libertad). Cantidad de variables libres que son restringidas cuando se agrega la articulación.

DLS: Damped Least Squares (Método de Mínimos cuadrados amortiguado)

Gait: Referido al andar de un multípodo.

EBNF: Extended Backus-Naur Form (forma de Backus-Naur extendida)

Efector: Un punto de interés de un robot, por ejemplo, una extremidad.

E/S: Entrada/Salida

Humanoide: ver Androide.

IK: Inverse Kinematics (Problema de la Cinemática Inversa)

Manipulador: ver Efector

PC: Personal Computer (Computadora personal)

PIC: Programmable Interrupt Controller (Controlador de interrupciones programables).

PWM: Pulse width modulation (Modulación en el ancho del pulso). Forma de codificar un valor analógico en una onda digital, de forma que el ciclo de trabajo está en relación con el valor a codificar.

Robot: Máquina o mecanismo que realiza una tarea. El término proviene del cirílico Robota (trabajo).

PC-Card: Anteriormente llamada tarjeta PCMCIA, es una plaqueta del tamaño de una tarjeta de crédito que puede interfacear con computadoras personales, entre otros dispositivos.

PCMCIA: Personal Computer Memory Card International Association. Estándar para tarjetas de expansión para laptops. Ver PC-Card.

Perfil (de velocidad, de actuación): Relación de la posición o velocidad de un actuador con respecto al tiempo. Por ejemplo, un perfil de posición lineal es de la forma $\theta(t) = \frac{\theta_0}{T} \cdot t$.

Sensor: Dispositivo que transduce una magnitud en otra mas fácil de medir.

Stance: Posición, pose. Conjunto de los valores de posición de las articulaciones del robot.

SDL: Software Description Language (Lenguaje de descripción de software).

SVD: Singular Value Decomposition (Descomposición en valores singulares).

Transductor ver Sensor.

USART: Universal Synchronous/Asynchronous Receiver/Transmitter (Tranceptor universal sincrónico/asincrónico).

ZMP: Zero Moment Point. Punto en el espacio donde se anulan todos los pares mecánicos.

Apéndice C

Conceptos de control

En este apéndice haremos un breve repaso del concepto de control y haremos énfasis en los elementos relevantes para el contexto del Proyecto.

C.1. Sistemas

Un sistema es un dispositivo que transforma señales de entrada en señales de salida. Si notamos las entradas como u y las salidas como y, podemos escribir $y = S\{u\}$.

Decimos que un sistema es determinista si dada la entrada u queda determinada la salida y de manera unívoca.

Un sistema es causal si su salida no puede anticipar la entrada, es decir, dadas dos entradas u_1, u_2 tal que $u_1(t) = u_2(t)\ \forall t \leq t_0$, sus salidas correspondientes serán iguales hasta ese mismo momento, es decir $y_1(t) = y_2(t)\, \forall t \leq t_0$.

Dos entradas u_1 y $u_2$se dirán equivalentes en t_0 si las salidas producidas son iguales desde $t = t_0$. En otras palabras, $y_{1[t_c,t]} = S\{u_{1[t_c,t_0]} \oplus r_{[t_0,t]}\} = y_{2[t_c,t]} = S\{u_{2[t_c,t_0]} \oplus r_{[t_0,t]}\}$, donde $f_{[t_a,t_b]}$ representa la señal f restringida al intervalo de tiempo $[t_a, t_b]$, y el operador $\oplus$ es el de concatenación de señales en el tiempo. La relación de equivalencia de señales de entrada es de hecho una relación de equivalencia (es reflexiva, simétrica y transitiva) y por tanto particiona a las señales en clases de equivalencia. Llamaremos a estas clases de equivalencia los "estados del sistema".

C.2. Clasificación de sistemas por cardinalidad del conjunto de estados

1. Sistemas algebraicos. Para cualquier instante de tiempo, todas las entradas pertenecen a la misma clase, el estado es único y el sistema se dice algebraico. $y(t) = S\{u(t)\}$, por ejemplo $y(t) = 2u(t) - 4$.

2. Autómata finito. Para cualquier instante de tiempo, el número de clases de equivalencia es finito. Un ejemplo es una máquina de estados.

3. Autómata infinito. En estos sistemas se puede establecer una biyección entre el conjunto de clases de equivalencia y los números naturales. Por ejemplo, una máquina de Turing.

4. Sistema de parámetros concentrados. En estos sistemas se puede establecer una biyección entre el conjunto de clases de equivalencia y los números reales. Por ejemplo, un circuito RC.

5. Sistema de parámetros distribuidos. En estos sistemas se puede establecer una biyección entre el conjunto de clases de equivalencia y el conjunto L_1de funciones módulo-integrables.

Un sistema es lineal si $S\{a+b\} = S\{a\} + S\{b\}$ y $S\{k \cdot a\} = k \cdot S\{a\}$ para k una constante.

El sistema retardo está definido por la siguiente ecuación: $r_\tau\{u(t)\} = u(t-\tau)$.

Un sistema es invariante en el tiempo si conmuta con el sistema retardo: $S\{r_\tau\{u\}\} = r_\tau\{S\{u\}\}$.

Por ejemplo, por definición, el sistema retardo es invariante en el tiempo.

Apéndice D

Mediciones y especificaciones

En este apéndice se documentan los resultados obtenidos de los diferentes relevamientos y calibraciones de los motores utilizados en el prototipo final, así como medidas realizadas sobre el prototipo y estimaciones y especificaciones de algunas magnitudes relevantes del mismo.

D.1. Motores

Como ya se mencionó, los motores utilizados son Futaba S3003. Las especificaciones del fabricante se listan a continuación.

Parámetro	4.8 V	6 V
Velocidad	.23 s/60º ($4.55\ \frac{rad}{s}$)	.19 s/60º ($5.51\ \frac{rad}{s}$)
Par	$310.64\,mN \cdot m$	$401\,mN \cdot m$

Tamaño	41 mm x 20 mm x 36 mm (largo x ancho x altura)
Peso	37.2 g
Conector	tipo "J" con cable de 5"

La velocidad está dada a par cero, y el par está dado a velocidad cero. Ver [FUTA].

D.1.1. Consumo

Mediante el uso de un multímetro digital, se obtuvo el consumo de los servomotores bajo tres condiciones: idle (es decir, deshabilitado), stall (trabado en una posición), y stall con par opuesto (es decir, intentando moverlo de la posición en la que está trabado). El consumo para estas condiciones se midió en 10 mA, 200 mA, 600 mA, respectivamente.

D.1.2. Ensayos de posición y velocidad

Para realizar las medidas de posiciones y velocidades admisibles, se realizaron dos ensayos, que se describen a continuación.

D.1.2.1. Ensayo de posición

Utilizando el software de monitoreo y diseño de posiciones en el PC, se relevaron las posiciones en las que se verificaba que el motor estaba muy cercano a un extremo de su recorrido, pero

que aún ejercía par resistente. Los relevamientos realizados determinaron que los motores pueden moverse sin estacionarse con valores de posición ubicados entre 2500 y 22500 aproximadamente.

D.1.2.2. Ensayo de velocidad

Al usar velocidades muy altas, el PIC enviaba la notificación de que el motor había llegado a destino antes de que esto hubiese ocurrido efectivamente. Esto se debe a que el servomotor tiene una cierta velocidad máxima que intentamos determinar en este ensayo. Para medir la velocidad máxima admisible, se fueron probando distintos valores de velocidades usando el software de monitoreo y diseño de posiciones en el PC, hasta que la notificación de que el motor había llegado a su posición deseada llegaba en el momento correcto (es decir, cuando esto efectivamente ocurría). Las velocidades máximas admisibles que los servomotores adquiridos pueden seguir, están en el entorno de 512. El ensayo se realizó de forma iterativa: se empezaron con velocidades altas y se notó que el motor atrasaba con respecto a la llegada de la notificación de arribo. Se fue bajando la velocidad hasta corroborar que la notificación llegaba al mismo tiempo que el motor.

Luego de realizar la modificación a los servomotores, y utilizando la lectura de posición, se verificó que los valores relevados por la lectura de posición fueran coherentes con los movimientos que se le imprimían al motor.

D.1.3. Calibración

Se llama calibración al proceso de encontrar parámetros de una cierta función que convierten las unidades de lectura de posiciones en unidades de control. En el prototipo se optó por una función de calibración lineal, por lo que sólo se necesitan dos puntos para determinar sus parámetros.

Las calibraciones se realizan como sigue: primero se mueve el motor hacia el centro con la máxima velocidad. Luego se mueve hacia la posición $LeftCal$ con una velocidad baja (50). La velocidad baja permite que el valor de posición del PIC no se desfase de la posición real, y asegura que cuando llega el bit Reached del motor en cuestión encendido, el motor haya llegado realmente. Se espera un segundo para estabilizar las vibraciones mecánicas y se toman las lecturas en este instante. Luego se mueve hacia la posición $RightCal$ también con velocidad 50, se espera un segundo y se toma otro conjunto de lecturas. Luego de esto, se calculan los parámetros en base a estas dos lecturas. La naturaleza pluggable del diseño de la aplicación, permite establecer otros criterios de calibración (por ejemplo, mínimos cuadrados, splines cúbicas, etc.). En el caso del prototipo implementado se optó por trazar la recta que pasa por esos dos puntos para cada uno de los motores. $LeftCal$ vale 30° desde el centro hacia la izquierda, y $RightCal$ vale 30° desde el centro hacia la derecha.

Las calibraciones dan por lo general valores bastante concentrados (con poca dispersión), con a (la pendiente de la recta) valiendo entre 52 y 58, y b (el valor en el origen) valiendo entre 200 y 600.

D.2. Especificaciones del Sistema

Llamamos Sistema al conjunto formado por el nodo PC y el nodo PIC, es decir, el prototipo (motores, plaqueta, y segmentos), fuente, interfaz serial.

D.2.1. Retardo de bajar los motores

Como se mencionó en la descripción del software embebido, éste debe tener en cuenta el tiempo que transcurre entre el momento en que se llama a la rutina de interrupción para bajar un motor y el momento en que el motor se baja efectivamente. Utilizando el simulador del MPLAB IDE, que permite realizar mediciones de tiempo, se midió que el tiempo insumido entre que se llama a la rutina de interrupción y se baja un motor es de 26 tics del timer en 10 MHz (2.6 μs). Este valor se definió como una constante en el código assembler del controlador.

D.2.2. Alimentación del prototipo

La fuente del prototipo debe alimentarse con 6.8 V DC como mínimo para funcionar correctamente, ya que de lo contrario el regulador de voltaje no funcionará correctamente por su voltaje de dropout.

D.2.3. Consumo

A continuación se presenta una tabla con los consumos estimados de los diferentes componentes del sistema.

Dispositivo	Consumo Típico (W)	Consumo Mínimo (W)	Consumo Máximo (W)
PIC	1	1	1
Motores	18	0.7	43.2
MAX232	$<$0.1	$<$0.1	$<$0.1
Fuente	-	-	-30
Total	19	2	30

D.2.4. Dimensiones del prototipo

- Alto: 36 cm (más 10 cm de cables en la parte superior).
- Ancho: 15 cm
- Profundidad: 9 cm
- Todas las medidas se realizaron con el prototipo en la posición *Erguido*.

D.2.5. Peso del prototipo

El peso del prototipo es de 1.0 kg.

Apéndice E

Cinemática inversa

Un enfoque numérico es el más apropiado para poder realizar cálculos cinemáticos inversos. La idea entonces es obtener un método que nos permita, en un tiempo razonable de cálculo, converger a una configuración que sea solución de la ecuación (2.2).

E.1. Marco de aplicación en este proyecto

La cinemática inversa como herramienta de control en lazo cerrado de la trayectoria de los efectores del sistema, se enmarca dentro de la capa walk de la arquitectura (capítulo 8). De esta forma, esta capa calcula, dada una trayectoria de los efectores, el andar correspondiente, y hace que el robot camine usando este andar.

E.2. Modelo de pequeños desplazamientos

Supondremos que los vectores r estarán en torno a un vector r_0, y procedemos a linealizar la función configuración en torno a la posición de equilibrio. A este vector le corresponde una (o más de una, pero tomamos una de las posibles) configuraciones a la que notaremos Q_0. Luego, llamemos $\tilde{r} = r - r_0$, y $\tilde{Q} = Q - Q_0$. Por lo que se tiene la siguiente ecuación:

$Q = Q_0 + \tilde{Q} \simeq Q_0 + \left(\frac{\partial r}{\partial Q}\right)^{-1} (r - r_0)$ (entendiendo la derivada como un tensor-gradiente).

Resumiendo, tenemos

$$\tilde{r} \quad = \quad \left(\frac{\partial r}{\partial Q}\right) \cdot \tilde{Q} = J \cdot \tilde{Q}$$

por lo que

$$\tilde{Q} \quad \equiv \quad \left(\frac{\partial r}{\partial Q}\right)^{-1} \cdot \tilde{r} \tag{E.1}$$

La matriz $J = \frac{\partial r}{\partial Q}$ es el jacobiano de la posición respecto a la configuración. Para calcular el jacobiano, consideramos dos tipos de articulaciones: las rotacionales y las traslacionales. En el

caso de una articulación rotacional q_j, se tiene que $\frac{\partial r}{\partial q_j} = v_j \times (r - p_j)$, donde v_j es un vector con la dirección del eje de rotación y de norma 1, y p_j es la posición de la articulación.

Si la articulación es traslacional q_j, se tiene que $\frac{\partial r}{\partial q_j} = v_j$. En el caso de que la articulación no afecte el efector se tiene $\frac{\partial r}{\partial q_j} = 0$.

Resumiendo, obtuvimos que $\tilde{Q} = J^{-1}\tilde{r}$.

Ahora, el jacobiano no necesariamente será una matriz cuadrada, eso implica que no necesariamente tiene una inversa. Para subsanar este hecho, definimos la inversa generalizada.

Decimos que una matriz X es inversa generalizada de una matriz J, $X = J^{-1}$ si cumple alguna de estas propiedades:

1) $J \cdot X \cdot J = J$

2) $X \cdot J \cdot X = X$

3) $(X \cdot J)^t = X \cdot J$

4) $(J \cdot X)^t = J \cdot X$

Si cumple todas estas, decimos que X es la seudoinversa de J. La seudoinversa es siempre única, y se encuentra minimizando $\|J \cdot X - \mathrm{I}\|_2$. Esto quiere decir que la matriz X es la matriz que más se acerca a ser la inversa de J, y por eso es la que utilizaremos para los procedimientos de cálculo.

E.3. Métodos de cálculo

El problema de usar la seudoinversa es que el proceso de cálculo es lento. Más precisamente, lleva $O(m^2 n)$ operaciones. Además hay problemas de inestabilidad numérica, originadas en las singularidades de las configuraciones. Esto es, el problema puede estar mal condicionado si partimos de una Q_0 singular (por ejemplo, una en la que todas las articulaciones están alineadas).

E.3.1. Método de la seudoinversa

Como mencionamos anteriormente, la resolución del problema (E.1), requiere la inversión de la matriz J, que no necesariamente es cuadrada. La seudoinversa o inversa de Moore-Penrose, J^+, minimiza $\| J \cdot J^+ - I \|_2$. Si la configuración está en una singularidad exactamente, entonces el método de la seudoinversa no producirá movimientos hacia posiciones imposibles, y el método se comporta bien. Si por el contrario, la configuración es cercana a una singularidad, pequeños cambios en la posición determinarán grandes cambios en la configuración. En la práctica, los errores de redondeo o truncamiento implican que nunca se llega a una singularidad exactamente y hay que verificar la singularidad detectando valores cercanos a cero.

La seudoinversa tiene además la propiedad de que la matriz $I - J^+ J$ realiza una proyección sobre el núcleo de J. Esto es, para cualquier vector $\overrightarrow{v}$, se cumple $J\left(I - J^+ J\right)\overrightarrow{v} = \overrightarrow{0}$. Esto significa que podemos establecer $\tilde{Q} = J^+ \tilde{r} + \left(I - J^+ J\right)\overrightarrow{v}$ para cualquier $\overrightarrow{v}$. Eligiendo apropiadamente el vector $\overrightarrow{v}$, podemos alcanzar objetivos secundarios, como ser devolver las articulaciones a las posiciones originales, evitar configuraciones singulares, etc.

E.3.2. Método de mínimos cuadrados amortiguado

El método DLS (Damped Least Squares, o mínimos cuadrados amortiguado), resuelve muchos de los problemas que tiene la seudoinversa con las singularidades. La idea del método es la siguiente: en lugar de encontrar $\tilde{Q} \therefore min \parallel \tilde{r} - J\tilde{Q} \parallel$, encontraremos $\tilde{Q} \therefore min \parallel \tilde{r} - J\tilde{Q} \parallel^2 + \lambda^2 \parallel \tilde{Q} \parallel^2$, con $\lambda \in \mathbb{R}$. Esto equivale a minimizar $\parallel \begin{pmatrix} J \\ \lambda I \end{pmatrix} \tilde{Q} - \begin{pmatrix} \tilde{r} \\ 0 \end{pmatrix} \parallel$. La ecuación normal correspondiente es $\begin{pmatrix} J \\ \lambda I \end{pmatrix}^t \begin{pmatrix} J \\ \lambda I \end{pmatrix} \tilde{Q} = \begin{pmatrix} J \\ \lambda I \end{pmatrix}^t \tilde{r}$.

Lo anterior puede reescribirse como $\left(J^t J + \lambda^2 I\right) \tilde{Q} = J^t \tilde{r}$.

Puede verse que la matriz $\left(J^t J + \lambda^2 I\right)$ es no-singular para $\lambda \neq 0$. Ergo, tenemos que $\tilde{Q} = \left(J^t J + \lambda^2 I\right)^{-1} J^t \tilde{r}$.

Además, $\left(J^t J + \lambda^2 I\right)^{-1} J^t = J^t \left(J J^t + \lambda^2 I\right)^{-1}$, por lo que $\tilde{Q} = J^t \left(J J^t + \lambda^2 I\right)^{-1} \tilde{r}$. Se puede encontrar un vector $\overrightarrow{f}$ tal que $\left(J J^t + \lambda^2 I\right) . \overrightarrow{f} = \tilde{r}$, y luego hacer $\tilde{Q} = J^t . \overrightarrow{f}$.

E.4. Comparación de los métodos

Para poder comparar ambos métodos, utilizaremos una herramienta del álgebra lineal llamada descomposición en valores singulares que presentamos a continuación.

E.4.1. Descomposición en valores singulares

Dada una matriz J, definimos la descomposición en valores singulares (SVD) como la descomposición en tres matrices cuyo producto da J de la forma: $J = U.D.V^t$, con U y V ortogonales y D diagonal.

$$J = \sum_{i=0}^{r} \sigma_i u_i v_i^t$$

donde r es el rango de J.

E.4.2. SVD del método de la Seudoinversa

La matriz D^+, correspondiente a la SVD de J^+, cumple $d_{ii}^+ = \begin{cases} 0 & si\ d_{ii} = 0 \\ \frac{1}{d_{i,i}} & si\ d_{ii} \neq 0 \end{cases}$, donde $J^+ = ((d))_{ij}$. La seudoinversa J^+ es entonces

$$J^+ = V D^+ U^t = \sum_{i=1}^{r} \sigma_i^{-1} v_i u_i^t$$

E.4.3. SVD del método de Mínimos cuadrados amortiguado

La matriz con la que se trabaja en DLS es $\Phi = J \cdot J^t + \lambda^2 I$. Además, podemos escribir

$$\Phi = J J^t + \lambda^2 I \quad = \quad \left(U D V^t\right) \left(V D^t U^t\right) + \lambda^2 I = U \left(D D^t + \lambda^2 I\right) U^t$$

La matriz $\left(DD^t + \lambda^2 I\right)$es la matriz diagonal cuyas entradas son $\sigma_i^2 + \lambda^2$. La inversa de esta matriz, es diagonal y sus componentes son $\frac{1}{\sigma_i^2+\lambda^2}$.

Por lo tanto,

$$J^t \cdot \Phi^{-1} \quad = \quad J^t \left(JJ^t + \lambda^2 I\right)^{-1} = \left(VD^t \left(DD^t + \lambda^2 I\right)^{-1} U^t\right) = VEU^t$$

donde $E = ((e))_{ij}$ cumple $e_{ii} = \frac{\sigma_i}{\sigma_i^2+\lambda^2}$.

La matriz solución del problema de mínimos cuadrados amortiguados puede escribirse como

$$J^t\Phi^{-1} \quad = \quad J^t \left(JJ^t + \lambda^2 I\right)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^t$$

.

E.4.4. Comparación de los métodos según SVD

En ambos casos, la matriz solución "invierte" el jacobiano haciendo $S = \sum_{i=1}^{r} \tau_i v_i u_i^t$. Para la seudoinversa, $\tau_i = \sigma_i^{-1}$. Para DLS, $\tau_i = \frac{\sigma_i}{\sigma_i^2+\lambda^2}$. El método de la seudoinversa es inestable cerca de las singularidades, ya que $\sigma_i \to 0$. Para valores de σ_i grandes respecto de λ, el método DLS no difiere mucho de la seudoinversa, ya que $\frac{\sigma_i}{\sigma_i^2+\lambda^2} \approx \frac{1}{\sigma_i}$, pero cuando son del mismo orden, las dos cantidades difieren. En particular, cuando son iguales, $\tau_i = \frac{1}{2\lambda}$, y

$$\lim_{\sigma_i \to 0} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} = 0$$

Por lo tanto, el método DLS actúa como la seudoinversa lejos de las singularidades, y cerca de ellas suaviza el problema. Por más información consultar [KA00, BU04].

Apéndice F

Control de servomotores

F.1. Introducción

Debido a la flexibilidad de los servomotores, facilidad de interfacear con un microcontrolador y alto par, hemos decidido utilizarlos para implementar las articulaciones. Dichos actuadores poseen tres cables, que se conectan a tierra, alimentación, y control. La pata de control lleva una señal modulada en ancho de pulso (PWM). Cuanto mayor sea el ciclo de trabajo de la onda, estará codificado un valor mayor del ángulo al que se quiere llegar. Esto es así pues permite interfacear facilmente con equipamiento digital, ya que los valores de la onda PWM son de 0 V para el nivel bajo (inactivo) y 5 V para el nivel alto, que corresponden a los niveles lógicos 0 y 1 respectivamente.

Este tipo de motor en su versión más simple consiste de un motor de corriente continua, reducciones y cierta circuitería para manejar el PWM y un lazo de realimentación interno. La onda PWM pasa por un filtro pasabajos cuya frecuencia de corte es muy baja. Esto promedia el valor alto y bajo en un ciclo de la onda, por lo que luego de este filtro se tiene una medida de la posición a la que se desea ir. Luego, se resta el valor del ángulo en el que se encuentra el motor de este valor, y el resultado, a menos de un factor de ganancia, es la entrada al motor de continua. Para sensar la posición actual se utiliza un potenciómetro adosado mecánicamente al eje, alimentado entre V_{CC} y tierra, de modo que en la pata del medio haya un valor que depende de la posición (divisor resistivo). Entre las versiones más sofisticadas, además de realimentar la posición del eje, se realimenta también la velocidad (que se obtiene de la fem reversa del motor), y/o el momento producido en el eje (que se obtiene sensando la corriente). Los servomotores digitales, funcionan con el mismo principio básico, pero la diferencia es que tienen integrado un microcontrolador que puede configurarse para obtener distintos comportamientos (tiempos de levantamiento, sobretiro, etc.) variando parámetros programables.

A continuación deduciremos las ecuaciones de un servomotor simple, a modo de ejemplificar su funcionamiento.

F.2. Modelo matemático del servomotor

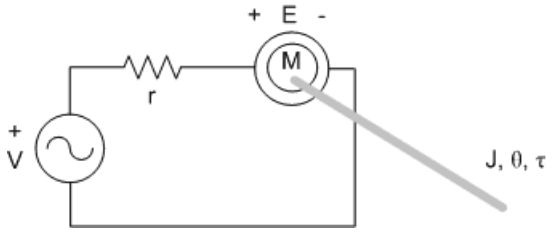


Figura F.1: Esquemático del circuito del motor de continua

Las ecuaciones que modelan el motor son entonces:

$E = A\Phi\omega$

$\tau = A\Phi i$ Ecuaciones del motor de CC

$J\dot{\omega} = \tau - \tau_{resistente}$ Segunda cardinal en el eje del motor

$V = r \cdot i + E$ Ley de Kirchhoff

$\omega = \dot{\theta}$ Definición de velocidad

donde:

E es la fuerza electromotríz del motor

τ es el par realizado por el motor

$\tau_{resistente}$ es el par resistente que se le impone al motor

J es el momento de inercia en el eje del motor

$A\Phi$ es la constante del motor

ω es la velocidad angular del motor

r es la resistencia de armadura del motor

i es la intensidad que atraviesa la armadura del motor

V es la tensión en bornes del circuito

Operando en el dominio de Laplace (el dominio de la variable "s"),

$$\begin{cases} V = r\,i + A\Phi s\theta \\ Js^2\theta = A\Phi i - \tau_{resistente} \end{cases}$$

Despreciando rozamientos y en condición de no tener carga, imponemos $\tau_{resistente} \approx 0$, por lo que $i = \frac{Js^2\theta}{A\Phi}$, y $V(s) = \frac{rJs^2\theta}{A\Phi} + A\Phi s\theta$.

$$V(s) = \theta(s)\left\{\frac{rJs^2}{A\Phi} + A\Phi s\right\} \Rightarrow V(s) = \theta(s)\cdot\frac{rJs^2 + A^2\Phi^2 s}{A\Phi}$$

Así obtenemos la función de transferencia de lazo abierto:

$$H_{ol}(s) = \frac{\theta(s)}{V(s)} = \frac{A\Phi}{s(A^2\Phi^2 + rJs)} = \frac{1}{A\Phi}\frac{1}{s(1+sT)}, T = \frac{rJ}{A^2\Phi^2}$$

Una vez obtenida esta transferencia de lazo abierto, calculemos la de lazo cerrado:

$$H(s) = \frac{H_{ol}(s)}{1+\beta H_{ol}(s)} = \frac{\frac{1}{A\Phi}\frac{1}{s(1+sT)}}{1+\beta\frac{1}{A\Phi}\frac{1}{s(1+sT)}} = \frac{1}{A\Phi\, s(1+sT)+\beta} = \frac{1}{\beta + A\Phi s + A\Phi T s^2}$$

Vemos entonces que la respuesta típica de un servomotor visto como motor de continua realimentado por posición es una respuesta de segundo orden.

Por lo general, este sistema será sobreamortiguado, por lo que podemos escribirlo de la siguiente manera: $H(s) = \frac{1}{A\Phi T}\cdot\frac{1}{(s-p_0)(s-p_1)}$, donde p_0 y p_1 son las raíces de $A\Phi T s^2 + A\Phi s + \beta = 0$. También generalmente podemos decir que habrá un polo dominante sobre el otro, por lo que la ecuación aproximada de la transferencia será $H(s) \approx \frac{k}{1+sT}$. La respuesta al escalón es entonces $k(1-e^{-\frac{t}{T}})Y(t)$. El tiempo en el cual el valor de la respuesta llega a $\frac{1}{e}$ de su valor final, es decir el tiempo de subida, es T. Vemos que la velocidad del motor es una exponencial decreciente, cuyo valor inicial es $\omega_0 = \frac{\theta^*}{T}$, donde θ^* es el valor de la señal de control.

F.3. Ramping lineal

Para uniformizar la velocidad del motor a lo largo de su desplazamiento, se propone utilizar una estrategia conocida como "linear ramping", o rampeado lineal. Dado que el motor responde muy rápidamente en el inicio de su trayectoria y muy lentamente sobre el final, es deseable uniformizar su velocidad. Esto no solamente es útil desde el punto de vista operativo, sino que reduce el "jerk" (derivada tercera de la posición) de manera considerable.

El linear ramping consiste en dividir el intervalo de desplazamiento ($\theta^* = \theta_0$) en N partes equiespaciadas (de ahí el término "linear") e ir "guiando" al motor, engañándolo al indicarle que vaya a una posición más cercana.

La entrada al motor es entonces la siguiente señal:

$$x(t) = \sum_{n=0}^{n=N-1}\frac{\theta_0}{N}Y(t-n\tau)$$

De esta forma, cada τ segundos, avanzamos la entrada en $\frac{\theta_0}{N}$ radianes. La salida correspondiente

a esta entrada será entonces

$$\theta(t) \quad = \quad \frac{\theta_0}{N} \sum_{n=0}^{n=N-1} (1 - e^{-\frac{t-n\tau}{T}}) Y(t - n\tau)$$

Para simplificar las cuentas, usaremos $\tau = T$. Entonces

$$\theta(t) = \frac{\theta_0}{N} \sum_{n=0}^{n=N-1} \left(1 - e^{-\frac{t-nT}{T}}\right) Y(t - nT) \Rightarrow$$
$$\theta(t) = \frac{\theta_0}{N} \sum_{n=0}^{n=N-1} \left(1 - e^{-\frac{t}{T}} e^n\right) Y(t - nT) \Rightarrow$$
$$\theta(t) = \frac{\theta_0}{N} \left(\sum_{n=0}^{n=N-1} Y(t - nT) - \sum_{n=0}^{n=N-1} e^{-\frac{t}{T}} e^n Y(t - nT) \right)$$

Para $t < NT$,

$$\begin{cases} \sum_{n=0}^{n=N-1} Y(t - nT) \approx \frac{t}{T} + \frac{1}{2} \\ \sum_{n=0}^{n=N-1} e^n Y(t - nT) \approx \frac{e^{\frac{t}{T}+1} - 1}{e-1} \end{cases}$$

por lo que

$$\theta(t) \quad \approx \quad \frac{\theta_0}{N} (\frac{t}{T} + \frac{1}{2} - e^{-\frac{t}{T}} \frac{e^{\frac{t}{T}+1} - 1}{e-1}) = \frac{\theta_0}{N} (\frac{t}{T} + \frac{1}{2} - \frac{e - e^{-\frac{t}{T}}}{e-1})$$

De esta forma llegamos a

$$\omega(t) \quad \approx \quad \frac{\theta_0 t}{NT} - \frac{1}{(e-1)T} e^{-\frac{t}{T}} = \frac{\omega_0}{N} - \frac{1}{(e-1)T} e^{-\frac{t}{T}}$$

Esta es la velocidad instantánea que desarrollará el motor. Observar que es ligeramente menor a $\frac{\omega_0}{N}$, pero que tiende a esa velocidad a partir de $t = T$. Esto es lo mismo que decir que si N es suficientemente grande, conseguiremos una velocidad suficientemente pareja. Esto redundará también en un par motor constante.

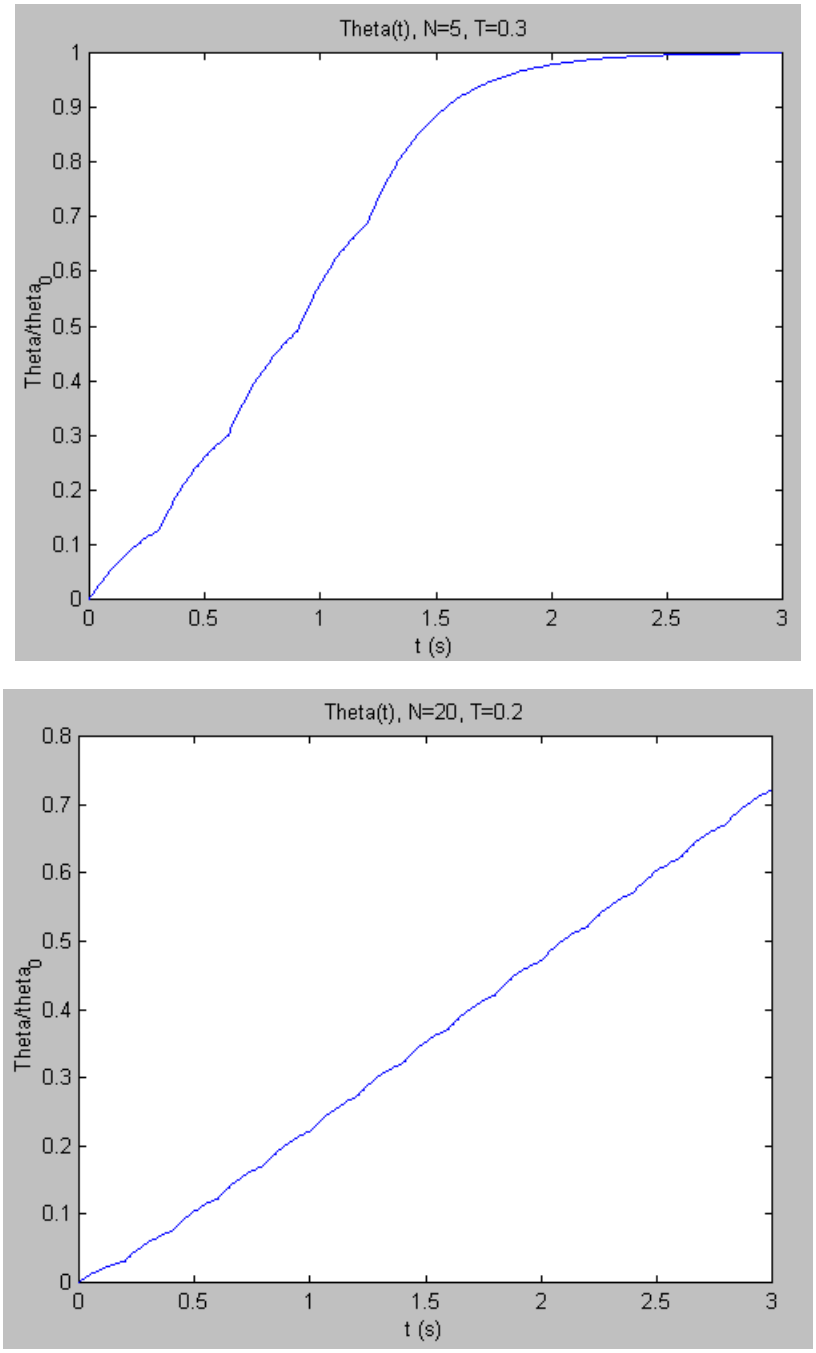


Figura F.2: Ángulo versus tiempo con distintos valores de N y T. Vemos que en el segundo caso se obtiene una velocidad casi constante.

F.4. Un modelo más complejo

El trabajo de Lee Buse ([BU00]), describe un modelo de mayor complejidad para los servomotores S3003 de Futaba. Cabe recalcar que el mismo está basado en suposiciones y no es de ninguna manera oficial. En su trabajo, Buse propone una forma de modificar el servomotor para

utilizarlo como rueda, con control de velocidad en lugar de posición. A continuación se presenta el esquemático de su propuesta:

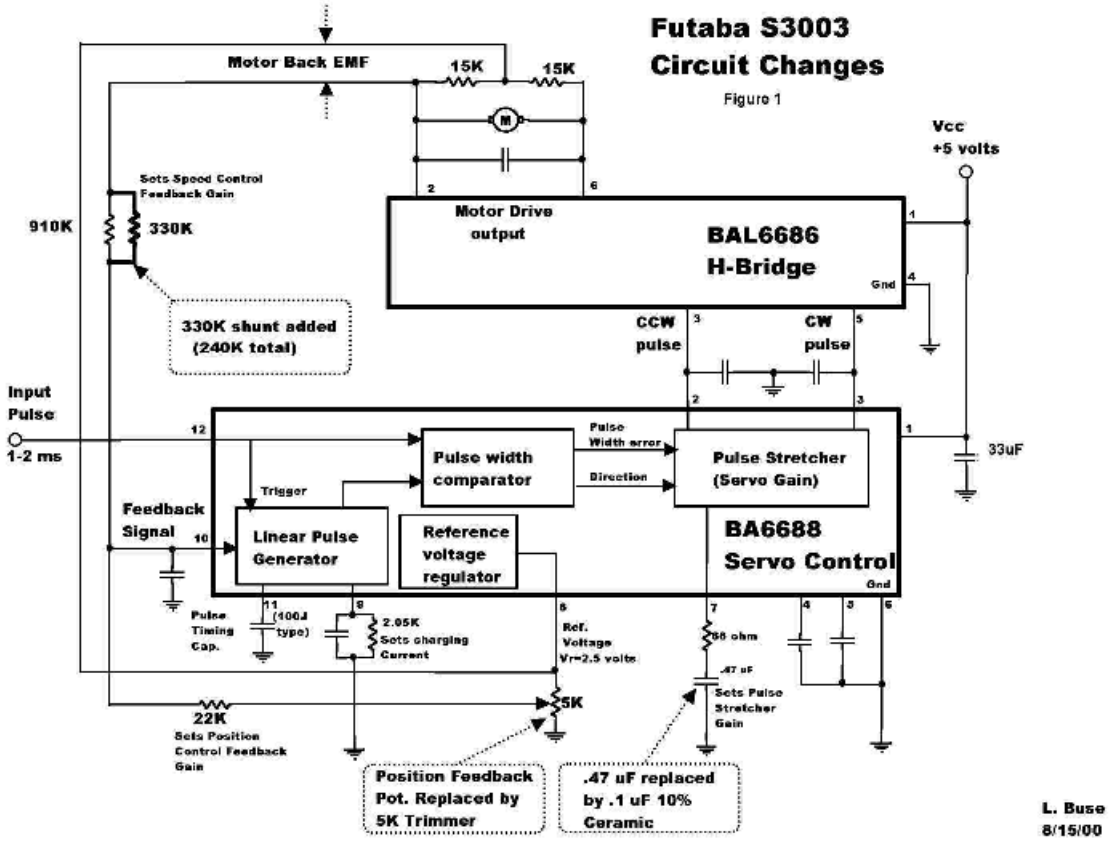


Figura F.3: Esquema del servomotor S3003 según Buse