SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

SEP
Institutos Tecnológicos

Subsecretaría de Educación Superior
Dirección General de Educación Superior Tecnológica
Instituto Tecnológico de la Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

" Diseño, modelado y control de una muñeca esférica con módulos PowerCube "

POR

Ing. Pedro Zúñiga Flores

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS

Dr. Ricardo Emmanuel Campa Cocom

ISSN: 0188-9060

RIITEC: (07)-TMCIE-2014

Torreón, Coahuila, México
Julio, 2014

SEP

INSTITUTO TECNOLOGICO de la laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Diseño, modelado y control de una muñeca esférica con módulos PowerCube"

POR

Ing. Pedro Zúñiga Flores

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
Dr. Ricardo Emmanuel Campa Cocom

ISSN: 0188-9060

TECNOLÓGICO DE LA LAGUNA

RIITEC: (07)-TMCIE-2014
Torreón, Coahuila, México,
Julio 2014

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
Instituto Tecnológico de la Laguna

"2014, Año de Octavio Paz"

Torreón, Coah., 16/Junio/2014
Dependencia: DEPI/CPCIE
Oficio: DEPIJ/CPCIE/041/2014
Asunto: Autorización de impresión de tesis.

C. PEDRO ZÚÑIGA FLORES
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"DISEÑO, MODELADO Y CONTROL DE UNA MUÑECA ESFÉRICA CON MÓDULOS POWERCUBE"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (07)-TMCIE-2014**, para que proceda a la impresión del mismo.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO de la Laguna
División de Estudios de Posgrado e Investigación

DR. JOSÉ LUIS MEZA MEDINA
Jefe de la División de Estudios de Posgrado e Investigación
del Instituto Tecnológico de la Laguna

RHSR

Torreón, Coah., 16/Junio/2014

DR. JOSE LUIS MEZA MEDINA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"Diseño, modelado y control de una muñeca esférica con módulos PowerCube"

Desarrollado por el **C. Pedro Zuñiga Flores**, con número de control **M07131483** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Ricardo E. Campa Cocom
Asesor/Director

Dr. Victor A. Santibáñez Dávila
Comité Tutorial

M.C. Javier E. Ollervides Vázquez
Comité Tutorial

Dr. Alejandro E. Dzul López
Comité Tutorial

Dedico este trabajo a mi familia y a las personas que han sido parte importante de mi vida.

Agradecimientos

Agradezco a Dios, por darme vida y bendición. A mi asesor, Dr. Ricardo E. Campa Cocom, por su orientación y apoyo. También agradezco al proyecto CONACYT 134534, y a dicha institución, por brindarme la oportunidad de realizar mis estudios de maestría.

A mis compañeros y amigos, con los que intercambié ideas y consejos durante mis estudios de maestría: Fernando (Fer), Alejandro (Dena), Eduardo (Cruz), Fo, Alejandro (Ale) y a los integrantes del "SADIC".

A mis padres y hermanos, por todo el apoyo. Al pueblo de México, que financió mis estudios a través del CONACYT.

Diseño, modelado y control de una muñeca esférica con módulos PowerCube

Pedro Zúñiga Flores

Resumen

La mayoría de los robots industriales utiliza servomotores del tipo conocido como de CD sin escobillas (o BLDC, por las iniciales en inglés de "brushless DC"). Para su funcionamiento, estos motores requieren de manejadores (o "drivers") que pueden configurar su modo de control para que la señal de entrada a los mismos actúe como una referencia deseada de posición, velocidad o corriente. Buscando cubrir la necesidad de actuadores compactos y de precisión para aplicaciones en robótica, la compañía alemana Amtec Robotics empezó a producir los llamados módulos PowerCube, que integran en un solo elemento un servomotor BLDC con una transmisión armónica, su manejador, y un sistema de control descentralizado que emplea interfaces de comunicación estándar tales como RS-232, CAN y PROFIBUS-DP. En el 2007 Schunk adquirió Amtec Robotics y continuó así desarrollando el mercado de robótica modular. El objetivo principal de esta tesis es diseñar y poner en operación una muñeca esférica empleando dos módulos PowerCube (uno de ellos de dos grados de libertad). Después de estudiar el funcionamiento de estos módulos (especialmente lo referente a los protocolos de comunicación), diseñar y construir el mecanismo, se procede a obtener su modelo cinemático y dinámico. Al final, se realizan experimentos en los que se evalúa el funcionamiento de la muñeca esférica empleando controladores simples en modo velocidad y modo corriente.

Design, modeling and control of a spheric wrist with PowerCube modules

Pedro Zúñiga Flores

Abstract

Most of the industrial robots employ a motor type known as brushless DC (BLDC) motor. In order to operate, these motors require drives which can be configured so that their input signal acts as a desired reference of either position, velocity or current. So as to meet the need of compact and precision actuators for robotic applications, the German company Amtec Robotics started to manufacture the so-called PowerCube modules, which integrate in one element a BLDC motor with a harmonic-drive gear, the motor's drive, and a decentralized control system which employ standard communication interfaces such as RS-232, CAN and PROFIBUS-DP. In 2007 Schunk purchased Amtec Robotics, continuing in this way the development of the modular robotic market. The main goal of this thesis is to design and implement a spherical wrist using two PowerCube modules (one of them with two degrees of freedom). After studying the operation way of these modules (particularly the communication protocols), designing and building the mechanism, we proceed to obtain its kinematic and dynamic model. At the end, some experiments are carried out in order to evaluate the performance of the spherical wrist when some simple controllers in velocity and current mode were employed.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde tiempos antiguos el ser humano se ha esforzado en construir máquinas que imitan las partes del cuerpo humano. La antigua civilización egipcia trabajó en formar brazos mecánicos para las estatuas de sus dioses; los cuales eran operados por sacerdotes egipcios, quienes clamaban que el movimiento de éstos era inspiración de sus dioses. Por otra parte, la antigua civilización griega construyó estatuas que reproducían movimientos gracias a ingeniosos sistemas hidráulicos; esto con el fin de fascinar a los adoradores de los templos griegos [Niku, 2001].

Durante los siglos XVII y XVIII, en Europa, fueron construidos varios muñecos mecánicos muy ingeniosos que tenían algunas características de máquinas en forma de humanos. Jacques de Vaucansos construyó varios títeres de tamaño humano a mediados del siglo XVIII. Esencialmente se trataban de mecanismos con forma de humanos diseñados para un propósito específico: la diversión. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos; una serie de levas eran el sistema principal de control para que esta muñeca pudiese realizar tareas de escritura y dibujo. Hubo otras invenciones mecánicas durante la revolución industrial, muchas de las cuales estaban dirigidas al sector de la producción textil; entre ellas se puede citar la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de

Cartwright (1785) y el telar de Jacquard (1801), entre otros [Lui and Wu, 2001].

En la actualidad existe una gran variedad de mecanismos, unos que precisamente imitan partes del cuerpo humano y otros que no, y son empleados para llevar a cabo tareas repetitivas, peligrosas o difíciles de realizar por las personas; estos mecanismos son llamados robots. El término robot fue empleado por primera vez en una obra checoslovaca publicada en 1917 por Karel Kapek, denominada "Rossum's Universal Robots". La palabra checa "robota" significa "servidumbre" o "esclavo", y cuando se tradujo al inglés se convirtió en el término robot. Los robots hoy en día son capaces de realizar una gran cantidad de tareas que van desde el pegado de etiquetas en una botella hasta la exploración espacial, pasando por muy diversas áreas, como la medicina, la industria automotriz, la química, etc.

La ciencia que se dedica al estudio de robots es llamada robótica; este término fue introducido por el ruso Isaac Asimov, escritor de ciencia ficción que concibiera a los robots como autómatas de apariencia humana carentes de sentimientos [Sciavicco y Siciliano, 2000]. La robótica es una ciencia interdisciplinaria que combina aspectos de la ingeniería eléctrica, mecánica e industrial, con ciencias computacionales, matemáticas y economía. Y es precisamente esta combinación de elementos la que ha determinado el crecimiento y desarrollo de la robótica. La robótica se puede dividir en robótica avanzada y robótica industrial; dentro de la robótica industrial se encuentran los llamados robots manipuladores. De acuerdo a la Federación Internacional de Robótica, un robot manipulador se define como "una máquina manipuladora con varios grados de libertad, controlada automáticamente, reprogramable y de múltiples usos, pudiendo estar en un lugar fijo o móvil para su empleo en aplicaciones industriales" [Kelly y Santibáñez, 2003].

1.1. Componentes de un manipulador industrial

Un robot manipulador se puede considerar como un brazo mecánico articulado formado por eslabones conectados a través de uniones o articulaciones, el cual posee una

herramienta de trabajo (pinzas, cortador, pulidor, etc.) para efectuar una tarea. De manera general, el número de articulaciones determina el número de grados de libertad (g.d.l.) del robot manipulador. Las articulaciones generalmente pueden ser rotacionales o lineales (prismáticas). Una articulación rotacional es como una bisagra, la cual permite una rotación relativa entre dos eslabones. Una articulación prismática permite un movimiento de traslación (lineal) relativo entre dos eslabones.

Los eslabones pueden ser móviles o fijos. Un eslabón se considera móvil cuando está sujeto a una articulación que le transmite movimiento. Un eslabón fijo es aquel que no posee movimiento. La figura 1.1 muestra el esquema de un robot manipulador de 4 g.d.l., donde se pueden observar cuatro articulaciones rotacionales nombradas como A_1, A_2, A_3 y A_4, además de cuatro eslabones móviles nombrados como E_1, E_2, E_3 y E_4, y un eslabón fijo (o base del robot) llamado E_0.

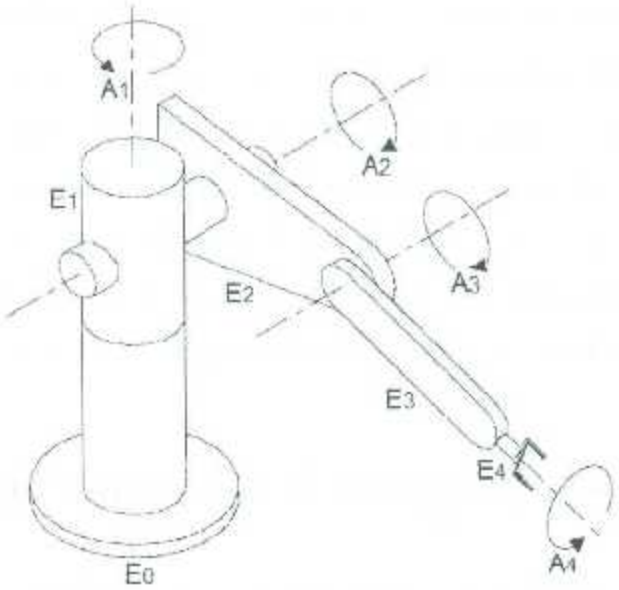


Figura 1.1: Robot manipulador de 4 g.d.l.

Como se mencionó al inicio de esta sección, un robot manipulador se puede considerar un brazo mecánico por la analogía que se puede establecer, en muchos casos, con las extremidades superiores del cuerpo humano. Por esta causa, los robots manipuladores por lo general se dividen en dos secciones: 1) ensamble de brazo y cuerpo y 2) ensamble de

muñeca. Normalmente se asocian tres articulaciones con el ensamble de brazo y cuerpo, y tres con el de muñeca. El ensamble de brazo y cuerpo tiene una función diferente del de muñeca. La función del primero es colocar un objeto u herramienta y la del segundo es orientar adecuadamente el objeto u herramienta. La orientación se relaciona con la alineación precisa del objeto con respecto a alguna posición estacionaria en el área de trabajo. En la figura 1.1 las articulaciones A_1, A_2, A_3 están asociadas al ensamble de brazo y cuerpo, y la articulación A_4 está asociada al ensamble de muñeca pero con un solo g.d.l para la función de orientación.

Para realizar estas funciones, las estructuras de brazo y cuerpo son diferentes de las de muñeca. La colocación requiere movimientos grandes, en tanto que la orientación requiere movimientos de giro y rotación para alinear el objeto u herramienta relacionados con una posición física en el lugar de trabajo. Un ensamble de brazo y cuerpo posee eslabones y articulaciones grandes, mientras que el de muñeca consta de eslabones cortos.

En los manipuladores comerciales hay cinco configuraciones básicas para el ensamble de brazo y cuerpo, mientras que para el ensamble de muñeca la configuración más usada por prácticamente todos los robots es la muñeca esférica. La muñeca esférica se caracteriza por tener 3 g.d.l. y por que todos sus ejes de rotación se intersecan en un solo punto [Sciavicco y Siciliano, 2000]. Las figuras 1.2 y 1.3 presentan las configuraciones básicas para el ensamble de brazo y cuerpo, y para el ensamble de muñeca, respectivamente.

1.2. Modelado y control de robots manipuladores

El problema de control de movimiento de robots manipuladores consiste en encontrar señales aplicadas a sus articulaciones, de tal forma que las posiciones asociadas a las coordenadas articulares del robot sigan con precisión una posición deseada en función del tiempo [Kelly y Santibáñez, 2003]. El modelado de robots juega un papel importante en la solución del problema de control de movimiento, ya que permite crear un modelo matemático del robot indispensable para la simulación de los movimientos. El modelado de

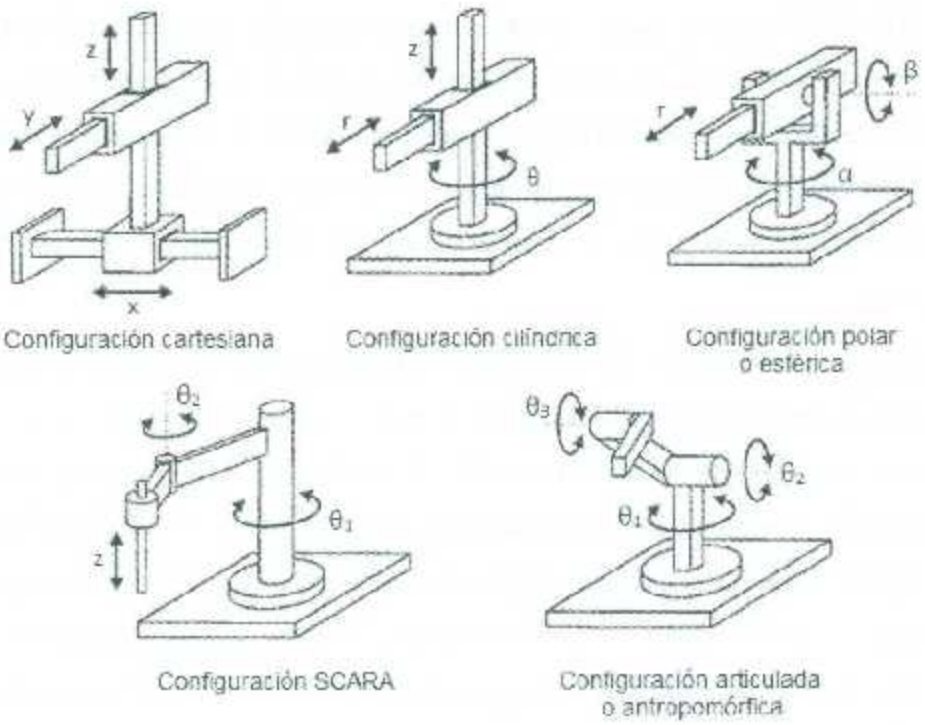


Figura 1.2: Configuraciones básicas para el ensamble de brazo y cuerpo

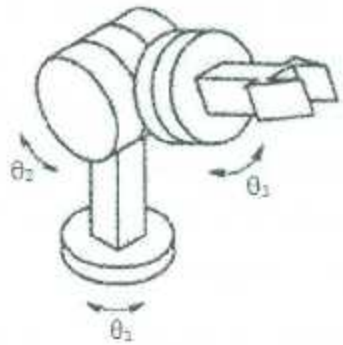


Figura 1.3: Configuración básica para el ensamble de la muñeca

robots se puede clasificar en dos categorías principales: modelado cinemático y modelado dinámico.

El modelado cinemático consiste en estudiar el movimiento prescindiendo de las fuerzas que lo producen, es decir, se trata de encontrar una función matemática que relacione el desplazamiento, la velocidad, la aceleración y el tiempo, sin hacer referencia a la causa del

movimiento. Existen varios métodos para obtener el modelo cinemático de robots manipuladores, entre los cuales destaca una metodología utilizando los parámetros geométricos del robot conocidos como parámetros de Denavit-Hartenberg. El modelado dinámico consiste en estudiar el movimiento a partir de las fuerzas que lo producen, es decir, se trata de encontrar una función vectorial que determine las fuerzas requeridas para producir cierto movimiento.

La obtención del modelo dinámico de robots manipuladores juega un papel importante para la simulación de movimientos, diseño de algoritmos de control y análisis de estructuras mecánicas de manipuladores. El cálculo de las fuerzas y pares necesarios para la ejecución de movimientos típicos provee información útil para establecer las características de diseño de articulaciones, transmisiones y actuadores. Existen varios métodos para obtener el modelo dinámico de robots manipuladores, se puede usar por ejemplo la formulación de Newton-Euler, las ecuaciones de movimiento de Hamilton o la formulación de Euler-Lagrange.

El movimiento de un robot manipulador es controlado por medio de señales aplicadas a sus actuadores, que pueden ser electromagnéticas, neumáticas o hidráulicas. Las articulaciones rotacionales son ampliamente utilizadas en los robots manipuladores industriales y generalmente poseen actuadores electromagnéticos, siendo los llamados motores de CD sin escobillas (BLDC por sus siglas en inglés: brushless DC) los más utilizados en tareas de control de movimiento de alta precisión.

Un motor BLDC está compuesto de un motor síncrono de imán permanente trifásico y un manejador electrónico ("drive" en inglés), el cual produce las señales trifásicas con la potencia requerida. Su nombre se debe a que su respuesta en estado estacionario es muy similar a la de un motor de CD con escobillas [Krause et al., 2002]. El manejador puede incluir hasta cuatro etapas básicas, tal como se muestra en la figura 1.4: un inversor de voltaje, un controlador de par, un controlador de velocidad y un controlador de posición [Campa et al., 2008].

El esquema de la figura 1.4 muestra también los tres modos de operación (indicados

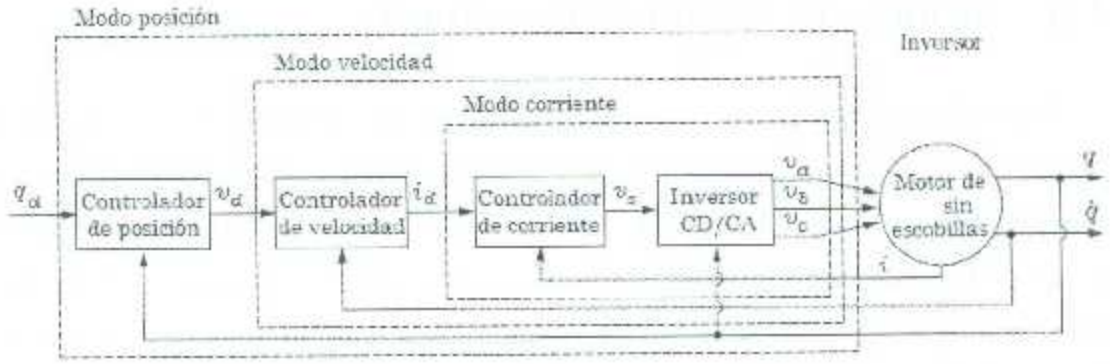


Figura 1.4: Diagrama esquemático de un sistema manejador/motor BLDC

en la figura por los rectángulos en líneas discontinuas) en los que puede configurarse un manejador de motores BLDC para cumplir con el objetivo general de control de posición. El nombre del modo (posición, velocidad o corriente) corresponde al tipo de señal que se emplea como entrada al manejador. Por ejemplo, cuando se configura el manejador en modo corriente los controladores internos de posición y velocidad son deshabilitados y la señal de entrada al manejador pasa directamente al controlador de corriente.

De lo anterior se concluye que la forma más simple de controlar la posición de un motor BLDC es configurando el manejador directamente en modo posición; la desventaja de esto es que generalmente no es posible modificar el controlador de posición interno, o sus ganancias. Otra manera es configurar el manejador en modo velocidad e implementar un lazo externo de control de posición que genere la referencia de velocidad deseada. Finalmente, se puede configurar el manejador en modo corriente y diseñar un controlador de posición que genere la consigna de corriente deseada. Dado que en un motor eléctrico la corriente en los devanados es proporcional al par mecánico entregado por el motor, al modo corriente se le conoce también como modo par.

1.3. Robots de arquitectura abierta

Dentro del contexto de robots industriales, el término arquitectura se define como el conjunto de elementos de hardware y las técnicas de software que caracterizan la estructura del sistema de control del robot. Los sistemas de control de los robots industriales del mercado poseen tradicionalmente arquitecturas cerradas, lo que provoca al usuario la imposibilidad de implementar nuevos algoritmos de control para diversas aplicaciones; esto lo hacen los fabricantes para que sus clientes obtengan equipo de expansión únicamente a través de ellos. El fabricante cobra más, pero las opciones para el usuario se reducen.

Los robots de arquitectura abierta son de especial interés para los grupos de investigación en el campo de control de robots debido a que en esta clase de robots es posible el acceso directo a las señales de control (par o velocidad); permitiendo a los investigadores la implementación y evaluación de algoritmos de diversos esquemas de control de alto desempeño [Blomdell et al., 2005]. Ejemplos populares de robots de arquitectura abierta son el Mitsubishi PA10, el Staubli RX60 y el PUMA 560, por mencionar algunos.

1.4. Robótica modular con PowerCube

El concepto de robótica modular nace a raíz de un nuevo paradigma de manufactura llamado *Sistemas de Manufactura Reconfigurables* (*RMS*, por sus siglas en inglés) que gracias a sus cualidades de modularidad, integrabilidad y personalización, permite implementar estructuras de robots con varios grados de libertad con beneficios de ser reconfigurables tanto en su estructura física (articulaciones, eslabones, espacio de trabajo, etc.) como en sus partes intangibles (software de control, métodos de comunicación, programación de tareas, etc.) para adaptarse rápidamente a diversas aplicaciones de manufactura y tareas de ensamble [Koren et al., 1999].

Una de las herramientas clave para el diseño de robótica modular son los componentes mecatrónicos inteligentes (*IMC*, por sus siglas en inglés), los cuales pueden ser considerados

como elementos o bloques básicos en la construcción de estructuras de robots bajo el paradigma *RMS* debido a que pueden ser fácilmente reemplazados y reintegrados en un ambiente de control automático existente.

Es bajo el paradigma *RMS* que aparece la compañía alemana Amtec Robotics, un pionero industrial en la construcción de elementos para robótica modular. Los módulos PowerCube desarrollados por Amtec Robotics prácticamente son *IMCs* que pueden ser fácilmente aplicados a tareas de manufactura y ensamble que requieren adaptación rápida y flexible de procesos. Básicamente los módulos PowerCube combinan en un espacio compacto, con geometría en forma de cubo, la mecánica motriz, la electrónica de control y potencia, y la infraestructura interna de comunicación e información con interfaces universales para su uso en aplicaciones flexibles de manufactura.

El método básico de control de movimiento utilizado por los módulos PowerCube es a través de una PC con una interfaz de comunicación (que puede ser RS-232, CAN o PROFIBUS-DP) y un software estándar de PowerCube instalado en la PC. El software estándar de PowerCube utiliza la interfaz de comunicación correspondiente para conectarse con los módulos PowerCube y generar la programación de movimientos a enviar paso por paso, o incluso almacenar dicha programación dentro de los módulos.

En el 2007 la compañía Schunk adquirió Amtec Robotics y continúa hasta el momento desarrollando el mercado de la robótica modular, ofreciendo un rango de hasta 31 módulos diferentes (elementos de sujeción, unidades rotacionales y de desplazamiento lineal, unidades de inclinación y rotación, etc.). Actualmente la robótica modular basada en la tecnología PowerCube de Schunk está tomando un rol de liderazgo en la construcción de manipuladores de peso ligero con la combinación flexible y compacta de módulos rotacionales y servo-pinzas actuadas eléctricamente para diseñar robots modulares como el que se muestra en la figura 1.5.

Dentro del área de robots móviles, específicamente en la clasificación de robots bípedos, también se destaca la participación de la tecnología PowerCube en la construcción de robots con aspecto humanoide como el que se muestra en la figura 1.6; diseñado y construido

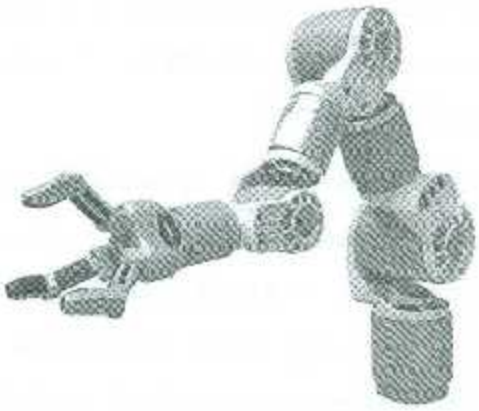


Figura 1.5: Brazo PowerCube de 7 g.d.l. con pinza

por el Instituto de Robótica de la Universidad de Johannes Kepler en Linz, Austria, y del cual se encuentran trabajos reportados como [Mayr et al., 2011a] y [Mayr et al., 2011b].

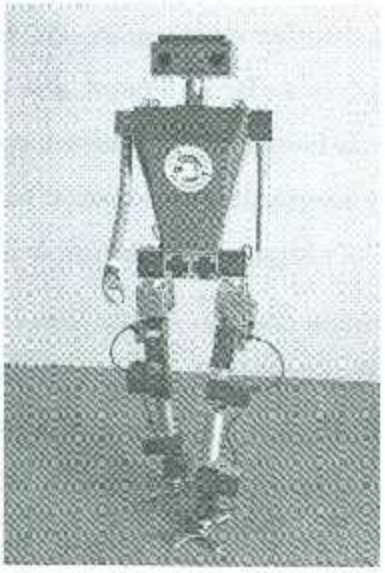


Figura 1.6: Robot bípedo de 14 g.d.l. con tecnología PowerCube

Existen diversos documentos de investigación referentes el diseño de estructuras de robots manipuladores reconfigurables a partir de la utilización de módulos PowerCube, así como su implementación dentro de sistemas de control desarrollados por el usuario para exponer problemas particulares de investigación. Por ejemplo, se encuentra el diseño de

un robot manipulador de 6 grados de libertad (g.d.l.) construido a partir de cinco módulos PowerCube [Smith y Christensen, 2009], el cual tiene como tarea atrapar un pelota con la ayuda de un sistema de visión con dos cámaras para la medición de posición de la pelota y de un órgano terminal en forma de guante. También se encuentra el diseño de un brazo robot reconfigurable de 6 g.d.l. construido con módulos PowerCube e implementado bajo el concepto de control distribuido [Strasser et al., 2008] para demostrar los beneficios de un sistema de control descentralizado y las ventajas de un robot reconfigurable con mínimos costos de diseño y programación. Otro ejemplo encontrado en la literatura es el diseño de un robot manipulador redundante de 7 g.d.l. [Wang et al., 2010] para el estudio del problema de resolución de redundancia empleando el algoritmo para obtención de la cinemática inversa en lazo cerrado (*CLIK*, por sus siglas en inglés).

1.5. Objetivos de la tesis

El presente trabajo de investigación tiene como objetivo general diseñar y construir un prototipo de muñeca esférica utilizando la tecnología de robótica modular PowerCube a través de dos módulos rotacionales del tipo PR-70 (1 g.d.l.) y PW-70 (2 g.d.l.), con el fin de contar con un mecanismo para realizar experimentos y evaluar algoritmos de control de orientación en tiempo real.

Como objetivos específicos se tienen definidos los siguientes:

1. Estudiar las características y la forma de operación de los módulos PowerCube tipo PR y PW.
2. Evaluar las diferentes interfaces de comunicación (RS-232, CAN-Bus y Profibus-DP) y seleccionar la más adecuada.
3. Diseñar la estructura mecánica para el ensamble de la muñeca esférica utilizando los módulos PR-70 y PW-70.
4. Obtener el modelo cinemático y dinámico de la muñeca esférica.

5. Desarrollar la interfaz de comunicación en tiempo real para controlar los módulos PowerCube.

6. Realizar una evaluación experimental empleando controladores simples en modo velocidad y corriente para la muñeca esférica a través de la interfaz de comunicación seleccionada.

1.6. Organización del documento

El contenido de este trabajo fue distribuido de la siguiente manera: en el capítulo 1 se presentan algunos conceptos básicos sobre robótica, enfocados al objeto de estudio de esta tesis, es decir, los robots manipuladores. Se presentan algunos antecedentes del diseño de robots seriales con módulos PowerCube, además de la descripción de las estructuras básicas de control de robots y el concepto de robótica modular de PowerCube aplicado a la construcción de robots reconfigurables. En el capítulo 2 se aborda con más detalle las características de los módulos PowerCube, se describen los controladores internos que poseen tales módulos y se analizan las diferentes interfaces de comunicación que emplean éstos para la transmisión y recepción de datos. El capítulo 3 describe la muñeca esférica que fue diseñada y construida como parte de esta tesis. El capítulo 4 describe el procedimiento para obtener de manera analítica el modelo cinemático y dinámico de la muñeca esférica utilizando los parámetros convencionales de Denavit-Hartenberg y la metodología de Euler-Lagrange, respectivamente. El capítulo 5 explica las características del protocolo de comunicación CAN utilizado para la transmisión y recepción de datos entre el sistema de control y las unidades de control de los servomotores de la muñeca. En el capítulo 6 se presentan las características principales de las herramientas de software utilizadas en el sistema de control de la muñeca esférica. Finalmente, en el capítulo 7 se presentan los resultados de la evaluación experimental que se realizó configurando los manejadores de los actuadores de la muñeca en modo velocidad y modo corriente.

Capítulo 2

Características de los módulos PowerCube

Los módulos PowerCube representan componentes mecatrónicos inteligentes (*IMC*, por sus siglas en inglés) con geometría cúbica desarrollados por Schunk. Los *IMCs* de PowerCube se caracterizan por combinar la infraestructura mecánica motriz, la electrónica de control y potencia, y la tecnología de comunicación dentro de un espacio compacto y modular. La familia de módulos PowerCube incluye hasta 31 módulos diferentes (pinzas, servomotores, actuadores de dos ejes, etc.) entre los que destacan los servomotores rotacionales eléctricos del tipo PR y PW para su empleo en robótica modular. La figura 2.1 muestra las principales familias de módulos PowerCube, los cuales, son descritos en la tabla 2.1.

En el presente capítulo se tratarán solamente los módulos tipo PR y PW, los cuales están equipados con una transmisión armónica (harmonic drive) que es directamente acoplada a un servomotor BLDC. Los módulos tipo PR y PW pueden ser de tres tamaños diferentes, y para identificarlos se les agrega un sufijo al código identificador del módulo que puede ser -70, -90 y -110; este sufijo indica el tamaño en mm de cada lado del cubo (o los cubos) del módulo correspondiente.

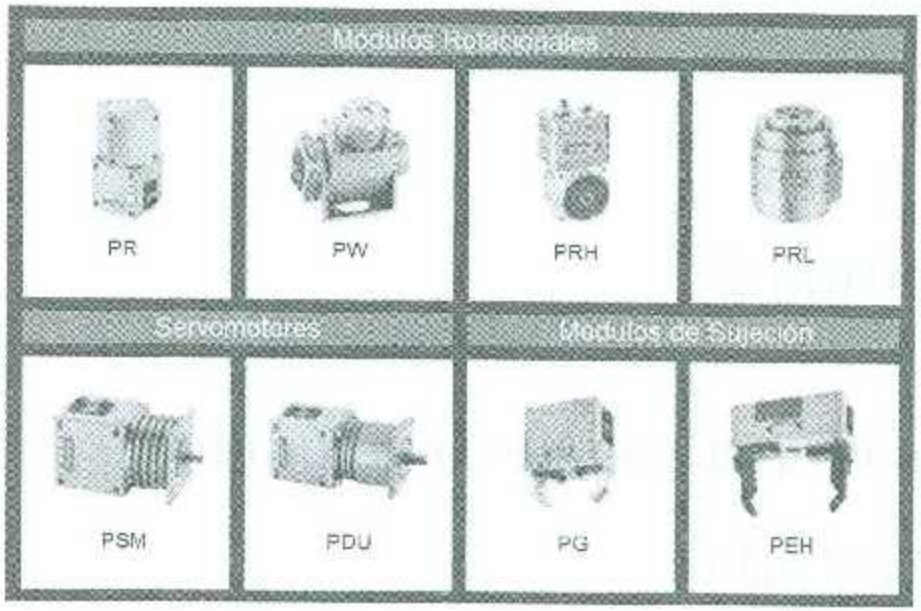


Figura 2.1: Clasificación de los principales tipos de módulos de PowerCube

Tabla 2.1: Familias de módulos PowerCube

Clasificación	*Descripción*
PR	Módulos de un giro servoeléctrico
PW	Módulos de giro inclinado, servoeléctricos, con dos ejes giratorios
PRH	Módulos de un giro servoeléctrico miniatura
PRL	Módulos con servomotor rotacional eléctrico y hueco en el centro
PSM	Servomotores compactos con control de posición integrado
PDU	Servomotores compactos con control de posición integrado y engranaje de precisión
PG	Pinzas servo-eléctricas compactas con dos dedos paralelos
PEH	Pinzas servo eléctricas con dos dedos largos paralelos

Los módulos PowerCube de Schunk se pueden considerar de arquitectura abierta debido a que se pueden utilizar en amplias aplicaciones de control con diferentes tarjetas de comunicación compatibles para transmitir las consignas de control de movimiento a sus respectivos servomotores. Los módulos PowerCube poseen conexiones simplificadas por medio de un bus de comunicación para enviar las señales de control a los manejadores (o drivers) que se encuentran en un sistema electrónico de control integrado que se denomina aquí unidad interna de control (UIC). Esta UIC junto con el servomotor, la transmisión

armónica y la interfaz del bus de comunicación se encuentran integrados en los módulos tipo PR y PW.

2.1. Servoactuadores rotatorios

Los servomotores rotacionales eléctricos pertenecen a un tipo de módulos PowerCube con características de movimiento rotacional por parte de sus ejes que van desde un rango de $\pm 120°$ hasta $\pm 360°$ (o superior). En esta clase de módulos se encuentran los tipos PR y PW que son de uso común para formar estructuras de robots manipuladores con varios grados de libertad para aplicaciones de control de posición y manipulación de objetos.

2.1.1. Módulos tipo PR

Los módulos PowerCube tipo PR consisten principalmente de un servomotor rotacional eléctrico del tipo BLDC que está acoplado a una transmisión armónica (harmonic drive) y, que además cuenta con una UIC donde se encuentra la electrónica de control y potencia (es decir, el manejador) que procesa las consignas de control de movimiento en modo posición, velocidad o corriente. La UIC posee una conexión simplificada a la que se puede acceder a través de una interfaz de comunicación utilizando alguno de los siguientes protocolos de comunicación: RS-232, CAN o PROFIBUS-DP.

El módulo tipo PR mostrado en la figura 2.2 puede generar movimientos rotacionales mayores a los ± 360 grados. La figura 2.3 muestra los principales componentes internos de este módulo (la numeración coincide con la de los elementos señalados en la figura):

1. Unidad interna de control (UIC). Se compone de la electrónica de control y potencia que, por medio de una interfaz simplificada basada en un protocolo de comunicación (RS-232, CAN o PROFIBUS-DP) recibe las consignas de movimiento proporcionadas por el usuario para operar el motor del módulo tipo PR en modo posición, velocidad

Figura 2.2: Módulo PowerCube tipo PR

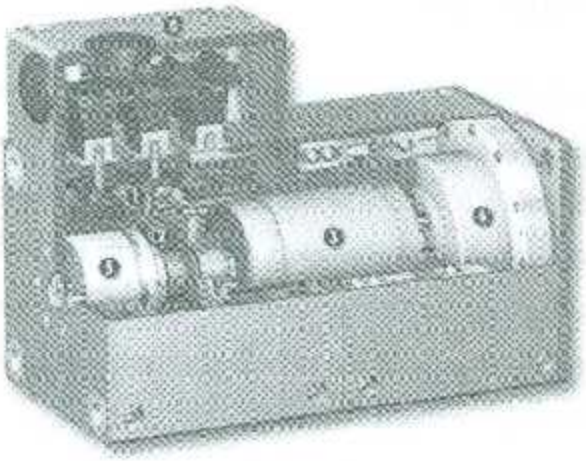


Figura 2.3: Estructura interna del módulo tipo PR

o corriente. El acceso a la UIC del módulo tipo PR se hace a través de su bloque de conexiones.

2. Encoder incremental. Transductor rotacional que transforma el movimiento angular del eje del motor en impulsos digitales. El módulo tipo PR cuenta con un encoder incremental para detectar la posición angular de su eje.

3. Motor de CD sin escobillas. Motor de 24 a 48 VCD 4 A (amperios) tipo BLDC con una capacidad de carga nominal de 7.5 a 142 N-m (después de la transmisión armónica) dependiendo del sufijo (-70, -90 o -110) del código identificador del módulo tipo PR. Es el encargado de generar el movimiento angular en el eje del módulo tipo PR.

4. Transmisión armónica. Mecanismo especial de engranajes que, por medio de un movimiento ondulatorio, transmite el movimiento angular del eje del motor al eje de la articulación del módulo con un determinado coeficiente de reducción y sin efectos de perdida de precisión causados por juegos mecánicos.

5. Freno electromagnético. Dispositivo encargado de frenar el movimiento del módulo y de mantener la posición angular fija cuando el módulo este inmóvil.

6. Bloque de conexiones (BC). Interfaz física de enlace entre los manejadores del módulo tipo PR y la estación de control del usuario. Contiene las terminales de la interfaz de comunicación para los protocolos RS-232, CAN y PROFIBUS-DP. El bloque de conexiones cuenta además con un bloque de cuatro entradas y cuatro salidas digitales de 24 VCD para procesar señales externas de control y activar alarmas, actuadores, etc. Este bloque de conexiones también proporciona la conexión física para la alimentación de potencia del motor y de la alimentación de energía de 24 VCD de 500 mA (miliamperios) para la lógica de control de la UIC.

2.1.2. Módulos tipo PW

El módulo PowerCube tipo PW consiste principalmente de dos servomotores rotacionales eléctricos del tipo BLDC que están individualmente acoplados a una transmisión armónica (harmonic drive), y que además cuentan con una UIC compartida donde se encuentra la electrónica de control y potencia (manejador) que procesa las consignas de control de movimiento en modo posición, velocidad o corriente para ambos servomotores. Al igual que en el caso de los módulos tipo PR, esta UIC posee una conexión simplificada

a la que se puede acceder a través de una interfaz de comunicación utilizando alguno de los siguientes protocolos de comunicación: RS-232, CAN o PROFIBUS-DP.

El módulo tipo PW mostrado en la figura 2.4 posee 2 ejes perpendiculares entre sí con los cuales puede generar movimientos independientes. El primer eje del módulo tipo PW puede generar movimientos de $\pm 120^{\circ}$ y el segundo eje de $\pm 360^{\circ}$ (o superior).

Figura 2.4: Módulo PowerCube tipo PW

El módulo tipo PW cuenta con los siguientes componentes principales, que se pueden visualizar en la figura 2.5 y que se describen a continuación:

1. Transmisión armónica. Mecanismo especial de engranajes que por medio de un movimiento ondulatorio transmite el movimiento angular del eje del motor al eje de la articulación del módulo. Para cada uno de los ejes del módulo tipo PW se tiene un determinado coeficiente de reducción; ambos sin efectos de perdida de precisión causados por juegos mecánicos.

2. Motor de CD sin escobillas (eje 2). Para el movimiento angular del eje 2 el módulo tipo PW tiene un motor de 24 VCD 2.1 A (amperios) tipo BLDC con una capacidad de carga nominal de 2 a 12 N-m (después de la transmisión armónica) dependiendo del sufijo (-70 o -90) del código identificador del módulo tipo PW.

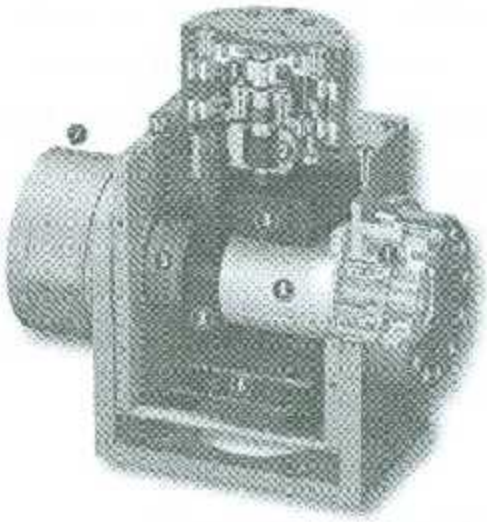


Figura 2.5: Estructura interna del módulo tipo PW

3. Encoder incremental. Transductor rotacional que transforma el movimiento angular del eje del motor en impulsos digitales. El módulo tipo PW cuenta con un encoder incremental para detectar la posición angular en cada uno de sus ejes.

4. Motor de CD sin escobillas (eje 1). Para generar el movimiento angular en el eje 1 el módulo tipo PW tiene un motor de 24 VCD 4 A (amperios) tipo BLDC con una capacidad de carga nominal de 12 a 23 N-m (después de la transmisión armónica) dependiendo del sufijo (-70 o -90) del código identificador del módulo tipo PW.

5. Freno electromagnético. Dispositivo encargado de frenar el movimiento del módulo y de mantener la posición angular fija cuando el módulo este inmóvil. El módulo tipo PW solamente tiene freno electromagnético en el eje 1.

6. Unidad interna de control (UIC). Se compone de la electrónica de control y potencia (manejadores) que, por medio de una interfaz simplificada basada en un protocolo de comunicación (RS-232, CAN o PROFIBUS-DP) recibe las consignas de movimiento proporcionadas por el usuario para operar de forma individual los motores del módulo tipo PW en modo posición, velocidad o corriente. El acceso a la UIC del módulo tipo PW se hace a través de su bloque de conexiones.

7. Bloque de conexiones (BC). Interfaz física de enlace entre los manejadores del módulo tipo PW y la estación de control del usuario. Contiene las terminales de la interfaz de comunicación (RS-232, CAN y PROFIBUS-DP). El bloque de conexiones cuenta además con un bloque de 4 entradas y 4 salidas digitales de 24 VCD para procesar señales externas de control y activar alarmas, actuadores, etc. Este bloque de conexiones también proporciona la conexión física para la alimentación de potencia del motor y de la alimentación de energía de 24 VCD de 500 mA (miliamperios) para la lógica de control de la UIC

2.2. Unidad interna de control

Las unidades internas de control (UICs) de los módulos PowerCube tipo PR y PW consisten de la electrónica de control y potencia que se caracterizan por tener una interfaz simplificada a través de un protocolo de comunicación (RS-232, CAN o PROFIBUS-DP) por medio del cual reciben las consignas de movimiento proporcionadas por el usuario para operar el o los motores de cada módulo en modo posición, velocidad o corriente, tal y como se explicó en la sección 1.2 (figura 1.4). En otras palabras, la UIC de cada módulo contiene el manejador (driver) del motor correspondiente.

Los módulos PowerCube poseen una interfaz que permite a los usuarios configurar los modos de operación (modo posición, modo velocidad y modo corriente) de los servomotores con los siguientes beneficios:

1. Minimizando el hardware de la interfaz. Gracias a un bus de comunicación que puede utilizar los protocolos RS-232, CAN y PROFIBUS-DP, se recortaron conexiones de las señales de control de los servomotores.

2. La UIC de cada módulo PowerCube está provista con ciertas funciones de "autoprotección" (límite de ángulo, error de comunicación, temperatura elevada, etc.) que generan un código (hexadecimal) de error al ser activadas, para avisar al usuario so-

bre un estado crítico de los módulos. Estas funciones de "autoprotección" permiten al usuario operar los módulos con cierto grado de seguridad en caso de descuidos en el manejo. Un resumen de las principales funciones de protección y códigos de error se presenta en la tabla 2.2. Para mayor información de estas funciones de protección y códigos de error se puede consultar [Schunk, 2012].

Tabla 2.2: Principales funciones de protección y códigos de error

Código hexadecimal	Código decimal	Descripción
0x70	112	Temperatura mínima alcanzada
0x71	113	Temperatura máxima alcanzada
0x72	114	Alimentación de la lógica de control demasiada baja
0x73	115	Alimentación de la lógica de control demasiada alta
0x74	116	Alimentación del motor demasiada baja
0x75	117	Alimentación del motor demasiada alta
0x76	118	Cable de comunicación defectuoso
0x82	130	Posición objetivo sobrepasada por disparo de movimiento
0xD5	213	Límite mínimo de movimiento definido en software fue sobrepasado
0xD6	214	Límite máximo de movimiento definido en software fue sobrepasado
0xD9	217	Se ha activado un paro de emergencia
0xDE	222	Se ha sobrepasado el límite de corriente permitido
0xDF	223	Servomotor sobrecargado
0xE0	224	El módulo no puede ser inicializado adecuadamente
0xE4	228	El límite de velocidad máxima fue sobrepasado

Los controladores integrados a las unidades internas de control cuentan con un lazo de control tipo P (Proporcional) para el control de posición, y con dos lazos de control tipo PI (Proporcional más Integral) para el control de velocidad y corriente. Es por medio de estos lazos internos de control que los módulos PowerCube pueden controlar por ellos mismos la posición y velocidad rotacional de cada uno de sus ejes así como la corriente que se aplica a los servomotores de los módulos; esto sin necesidad de contar con un algoritmo de control externo para llevar a un valor deseado los estados de posición, velocidad y corriente de los módulos PowerCube. Los lazos internos de control provistos por los manejadores disponen

de ganancias de ajuste referentes a sus controladores (P o PI) que pueden ser modificadas en caso de que sea necesario como se especifica en [Schunk, 2012].

2.2.1. Bloque de conexiones

El acceso a la UIC de los módulos tipo PR y PW se hace a través del bloque de conexiones mostrado en las figuras 2.3 y 2.5, respectivamente. Este bloque de conexiones se sujeta a los módulos tipo PR y PW a través de cuatro tornillos externos; en su interior se encuentra una tarjeta de conexiones que se fija al bloque de conexiones por medio de cuatro tornillos, tal y como se muestra en la figura 2.6.

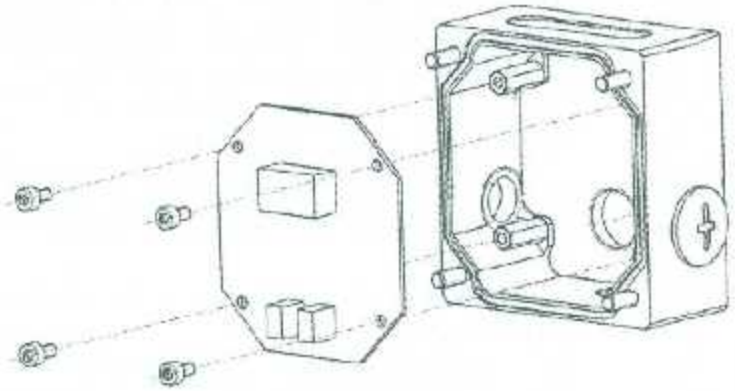


Figura 2.6: Separación de la tarjeta de conexiones

La tarjeta de conexiones es similar para los módulos tipo PR y PW. La figura 2.7 muestra la representación esquemática de la tarjeta de conexiones ubicada en el interior del bloque de conexiones.

La tarjeta de conexiones de estos módulos poseen 3 terminales de conexión principales llamadas X1, X2 y X3, además de las terminales de conexión de alimentación para el o los motores (dependiendo si es el módulo tipo PR o PW) y unos puentes de conexión para las resistencias de terminación de bus de la interfaz de comunicación seleccionada (RS-232, CAN o PROFIBUS-DP). La descripción de las terminales de conexión que aparecen en la figura 2.7 se presenta a continuación:

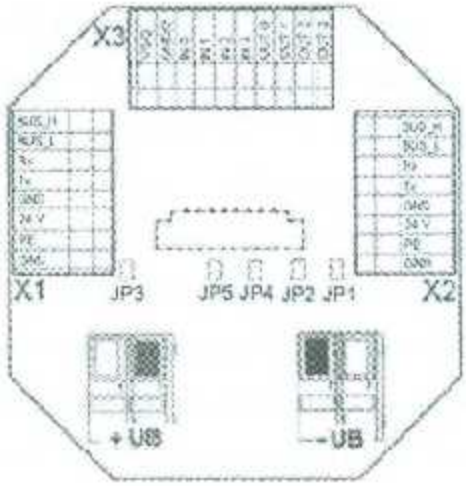


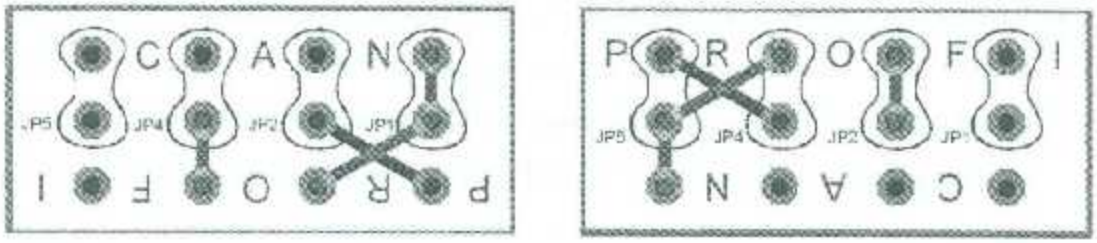
Figura 2.7: Estructura de la tarjeta de conexiones

- Terminal de conexión $X1$. Es la principal terminal de conexión para la interfaz de comunicación y la alimentación de energía para la lógica de control de la UIC del módulo. En este bloque se encuentran las dos líneas de transmisión de la interfaz de comunicación del protocolo CAN o PROFIBUS-DP llamadas *BUS-H* y *BUS-L*, así como las dos líneas de transmisión más la tierra de la interfaz de comunicación RS-232 identificadas como *Rx*, *Tx* y *GND*; también se encuentran los tres bornes de conexión para la alimentación de energía de 24 VCD a 500 mA (miliamperios) de la UIC identificadas como $24\,V$, *PE* (siglas de *Protective Earth* o tierra física) y *GND*.

- Terminal de conexión $X2$. La función de esta terminal es propagar las líneas de conexión de la terminal $X1$ para crear una conexión en cascada con otro módulo conectado a la misma interfaz de comunicación y a las mismas líneas de alimentación de la lógica de control de la UIC.

- Terminal de conexión $X3$. Esta terminal se usa para la transmisión y recepción de salidas y entradas digitales, respectivamente. Cuenta con 4 entradas (IN0, IN1, IN2 e IN3) y 4 salidas (OUT0, OUT1, OUT2 y OUT3) digitales a 24 VCD. Las salidas digitales necesitan la alimentación de una fuente externa de 5 a 24 VCD, la cual debe ir conectada a los bornes $VS/2$ y $GND/2$.

- Puentes de conexión ("*jumpers*"). Se utilizan para habilitar la o las resistencias de terminación del bus cuando el módulo es el último nodo conectado al bus. Existen 5 puentes de conexión de JP1 a JP5 (véase figura 2.7), de los cuales, JP3 es habilitado por medio de un simple puente (*jumper*) para terminar el bus cuando se este utilizando la interfaz RS-232; los 4 puentes restantes son para terminar el bus cuando se estén utilizando las interfaces CAN o PROFIBUS-DP. Para el caso de las interfaces CAN o PROFIBUS-DP, los puentes de conexión JP1, JP2, JP4 y JP5, habilitan las resistencias de terminación de bus por medio de una tarjeta de circuito impreso (proporcionada por el fabricante de los módulos) que es colocada en una determinada posición por encima de los puentes de conexión como se muestra en la figura 2.8.



(a) (b)

Figura 2.8: (a) Terminación del bus CAN y (b) terminación del bus PROFIBUS-DP

- Alimentación del motor. Se trata de dos bornes de conexión identificadas como $+UB$ y $-UB$ para la alimentación de potencia del motor o los motores correspondientes (dependiendo si es el módulo tipo PR o PW). Cada uno de los bornes $+UB$ y $-UB$, cuenta con dos entradas de cable (comunes); se puede utilizar cualquiera de los dos entradas de cada borne para alimentar el servomotor del módulo correspondiente, y la otra entrada sobrante del borne queda disponible para propagar la alimentación de potencia a otro módulo en caso de que sea necesario.

En resumen, el bloque de conexiones se considera como la interfaz entre cada módulo PowerCube (donde se encuentra la UIC), y la computadora de control (donde se encuentra la interfaz para monitoreo y configuración) que se explicará más adelante en la sección 2.4.

La figura 2.9 ilustra la función del bloque de conexiones en conjunto con la interfaz de comunicación seleccionada y la computadora de control. Nótese que las terminales X1 y X2 del bloque de conexiones coinciden con las funciones descritas anteriormente.

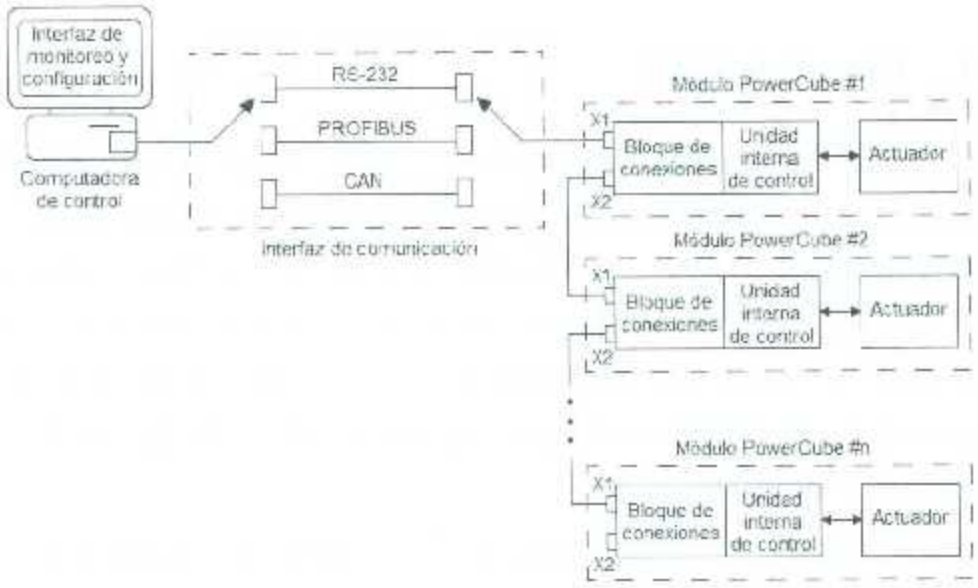


Figura 2.9: Funcionamiento del bloque de conexiones

Para mayor información acerca de las terminales de conexión del bloque de conexiones de los módulos PowerCube tipo PR y PW se puede consultar [Schunk, 2011a] y [Schunk, 2011b], respectivamente.

2.3. Protocolos de comunicación

Los módulos PowerCube tipo PR y PW manejan tres métodos diferentes de comunicación a través de los protocolos RS-232, CAN y PROFIBUS-DP para comunicarse con sus respectivas computadoras de control a través de un bus de comunicación o una red común. Estos protocolos de comunicación cuentan con sus respectivas interfaces dentro de los bloques de conexión de cada módulo. Cada protocolo de comunicación posee características de funcionamiento que se pueden adaptar a las necesidades del usuario, como por ejemplo

la velocidad de transmisión, el número de nodos o módulos conectados al bus de comunicación, la longitud máxima de transmisión, etc.; además de reducir considerablemente el hardware de conexión al utilizar menos cableado.

2.3.1. Protocolo CAN

La interfaz CAN (acrónimo de Controller Area Network) que poseen los módulos PowerCube tipo PR y PW utiliza el protocolo de comunicación CAN de la especificación 2.0a para transmitir y recibir comandos de control de sus respectivas computadoras de control a través de una red común. La especificación 2.0a del protocolo CAN introduce los comandos de control dentro de paquetes de datos conocidos como mensajes con un formato de trama de datos normal como se explica en [Schunk, 2012], y también como se explicará más adelante en el capítulo 5.

Las computadoras de control necesitan de un puerto CAN proporcionado por una tarjeta de comunicación CAN para establecer un enlace de comunicación con los módulos PowerCube. Esta tarjeta de comunicación CAN debe pertenecer a una lista de tarjetas de comunicación compatibles con los módulos PowerCube como se especifica en [Schunk, 2012].

Las principales características del protocolo de comunicación CAN utilizado en los módulos PowerCube son las siguientes:

- Velocidad de transmisión de hasta 1 Mbit/s.
- Longitud máxima de transmisión de 9 m.
- Número máximo de nodos o módulos conectados de 200 (30 por segmento).
- Red multi-maestro de acceso aleatorio.
- Método de comunicación basado en el mensaje.
- Mensajes de datos con tamaño máximo de 8 bytes.

2.3.2. Protocolo PROFIBUS-DP

La interfaz PROFIBUS-DP (acrónimo de Process Field Bus and Decentralized Periphery) que poseen los módulos PowerCube tipo PR y PW, utiliza el protocolo de comunicación PROFIBUS-DPV0 (primera versión del protocolo) para transmitir y recibir comandos de control de sus respectivas computadoras de control a través de un bus de comunicación. La primera versión del protocolo PROFIBUS-DP introduce los comandos de control dentro de mensajes de datos con un formato de empaquetamiento como se explica en [Schunk, 2012].

Las computadoras de control necesitan de un puerto PROFIBUS-DPV0 proporcionado por una tarjeta de comunicación PROFIBUS-DP que maneje la primera versión de este protocolo para establecer un enlace de comunicación con los módulos PowerCube. Esta tarjeta de comunicación PROFIBUS-DP debe pertenecer a una lista de tarjetas de comunicación compatibles con los módulos PowerCube como se especifica en [Schunk, 2012].

Las principales características del protocolo de comunicación PROFIBUS-DP utilizado en los módulos PowerCube son las siguientes:

- Velocidad de transmisión de hasta 1.5 Mbit/s.
- Longitud máxima de transmisión de 30 m.
- Número máximo de nodos o módulos conectados de 126 (32 por segmento).
- Red maestro (sistema de control)/esclavo (módulo) para envío y demanda de datos unidireccional.
- Método de comunicación basado en la dirección.
- Mensajes de datos con tamaño máximo de 8 bytes para transmisión (de maestro a esclavo) y de 16 bytes para recepción (de esclavo a maestro).

2.3.3. Protocolo RS-232

La interfaz RS-232 (acrónimo de Recommended Standard 232) que poseen los módulos PowerCube tipo PR y PW, utiliza el protocolo de comunicación serial RS-232 para transmitir y recibir comandos de control de sus respectivas computadoras de control a través de una red común. El protocolo serial RS-232 introduce los comandos de control dentro de mensajes de datos con un formato de empaquetamiento como se explica en [Schunk, 2012].

Las computadoras de control necesitan contar con un puerto RS-232 libre para establecer un enlace de comunicación con los módulos PowerCube.

Las principales características del protocolo de comunicación RS-232 utilizado en los módulos PowerCube son las siguientes:

- Velocidad de transmisión de hasta 115,200 bits por segundo.
- Longitud máxima de transmisión de 3 m.
- Número máximo de nodos o módulos conectados de 255.
- Red insegura con posible colisión de datos.
- Método de comunicación utilizado frecuentemente para la parametrización de los módulos o nodos.
- Mensajes de datos con tamaño máximo de 255 bytes.

2.4. Computadora de control

La computadora de control interactúa directamente con el usuario; consta de una computadora (generalmente una PC) donde se encuentra instalado el software de programación estándar (veáse figura 2.10) de la compañía Schunk llamado *MTS* (acrónimo de Motion Tool Schunk), que es utilizado por el usuario con el fin de generar las referencias

para las unidades internas de control según el modo de operación de los módulos PowerCube. Las referencias generadas se envían desde la computadora de control a las unidades internas de control de los módulos PowerCube a través de un bus de comunicación provisto por una tarjeta de comunicación instalada dentro de la periferia de la computadora de control. Esta tarjeta de comunicación debe pertenecer a una lista de tarjetas compatibles con el software *MTS* descritas en [Schunk, 2012].

(a) (b)

Figura 2.10: (a) Ventana principal de MTS y (b) ventana de programación del módulo correspondiente

El software *MTS* permite al usuario (a través de la ventana principal mostrada en la figura 2.10a) seleccionar y configurar la tarjeta de comunicación a utilizar para enviar las consignas de control a los módulos PowerCube. Además, por medio de la computadora de control y de la ventana de programación del módulo (véase figura 2.10b) es posible generar secuencias básicas de control en modo posición, velocidad o corriente para inmediatamente ejecutarlas o, en caso de que sea necesario, almacenar dicho control dentro de una localidad de memoria EEPROM de los módulos PowerCube para su posterior ejecución. Para una mayor descripción de las funciones básicas del software de programación mostrado en la figura 2.10 ver el apéndice A.

Los módulos PowerCube también pueden ser manejados desde un lenguaje de programación (como Visual C/C++ y Visual Basic) a través de una librería de funciones DLL que permite al usuario operar los módulos bajo el ambiente de Windows y un compilador compatible como se especifica en [Amtec, 2004].

Capítulo 3

La muñeca esférica MASKARA

3.1. Antecedentes

Desde hace algunos años en el Laboratorio de Mecatrónica y Control se tenía intención de construir una muñeca esférica que sirviera para realizar experimentos de control de orientación. Se pretendía que tal muñeca empleara motores tipo BLDC pero que no fueran demasiado grandes, con tal de reducir lo más posible las dimensiones de la muñeca.

Después de realizar una búsqueda exhaustiva de los motores adecuados, finalmente se decidió comprar los módulos tipo PR-70 y PW-70 de PowerCube. Cuando esta tesis inició, en enero de 2013, ya se contaba con ambos módulos, pero estos no habían sido probados aún. Por esta razón, al definir la propuesta del proyecto de tesis se establecieron las siguientes actividades a realizar con los módulos:

- Revisión de la documentación para instalación y operación.
- Puesta en operación empleando el software del fabricante.
- Estudio de los diferentes protocolos de comunicación posibles.
- Evaluación del desempeño usando diferentes protocolos.

- Diseño y construcción de las piezas para ensamble de la muñeca.

Tras revisar la documentación correspondiente se decidió realizar las primeras pruebas de funcionamiento de ambos módulos empleando el software del fabricante, que es descrito en el apéndice A y el protocolo de comunicación RS-232 que, aunque es lento, para fines de control resulta útil para una primera evaluación del desempeño de los módulos.

Una vez comprobado que ambos módulos funcionaban correctamente, se procedió al estudio en forma de los protocolos de comunicación PROFIBUS-DP y CAN. Además, para poder realizar las pruebas correspondientes empleando estos protocolos se seleccionaron y adquirieron las tarjetas PCI CIF 50-PB y CAN-AC2-PCI como interfaces para crear un bus de comunicación entre la computadora de control y los módulos, empleando los protocolos PROFIBUS-DP y CAN, respectivamente.

Al final, sólo se pudo establecer comunicación con ambos módulos empleando el protocolo CAN, por lo que se decidió emplear éste de forma definitiva como medio de comunicación para la muñeca esférica. Es por eso que en la sección 3.4.2 se describe únicamente la tarjeta de comunicación CAN y el capítulo 5 está dedicado al protocolo CAN.

Y en cuanto a la construcción de la muñeca esférica en sí, se diseñaron y mandaron a maquinar algunas piezas adicionales, las cuales son descritas en el apéndice B. También en este apéndice se mencionan los pasos para el ensamble de la muñeca esférica y su instalación en un pedestal ya existente en el laboratorio de Mecatrónica y Control del I.T.L.

Al prototipo final de la muñeca esférica (que incluye al mecanismo y todos sus componentes de hardware y software) se le ha denominado MASKARA por las iniciales de "Module-Assembled Spherical-Kinematics Articulated Robot Arm" o "Brazo Robótico Articulado de Cinemática Esférica Ensamblado por Módulos".

Para una mejor descripción de sus componentes y funcionamiento, desde el punto de vista de su sistema de control, se considera que la muñeca MASKARA es un sistema jerárquico con cuatro niveles, tal como se muestra en la figura 3.1. Estos niveles son:

1. Nivel 1: Mecanismo.
2. Nivel 2: Unidades manejadoras de servos.
3. Nivel 3: Unidad de control.
4. Nivel 4: Unidad de programación.

En las siguientes secciones se describe cada uno de estos niveles. La figura 3.1 ilustra el enfoque tradicional de un sistema de control con dos computadoras, conocidas en este esquema como computadora de desarrollo y computadora de control [Monroy et al., 2001]. La descripción de estas dos computadoras se realizará más adelante en este capítulo.

3.2. Nivel 1: Mecanismo

Este es el nivel más bajo y corresponde a toda la estructura física del prototipo, desde la base hasta el órgano terminal. Básicamente consta de los siguientes elementos:

- Módulos PR-70 y PW-70 de PowerCube.
- Piezas para ensamble.
- El órgano terminal.

En la figura 3.2 se muestra la muñeca ensamblada en dos configuraciones diferentes, vertical y horizontal, sobre un pedestal ubicado en el laboratorio de Mecatrónica y Control del I.T.L.

3.2.1. Módulos PowerCube

Como ya se ha mencionado, para el diseño y construcción de la muñeca MASKARA se utilizaron los módulos PowerCube (tipo PR-70 y PW-70) explicados en el capítulo 2.

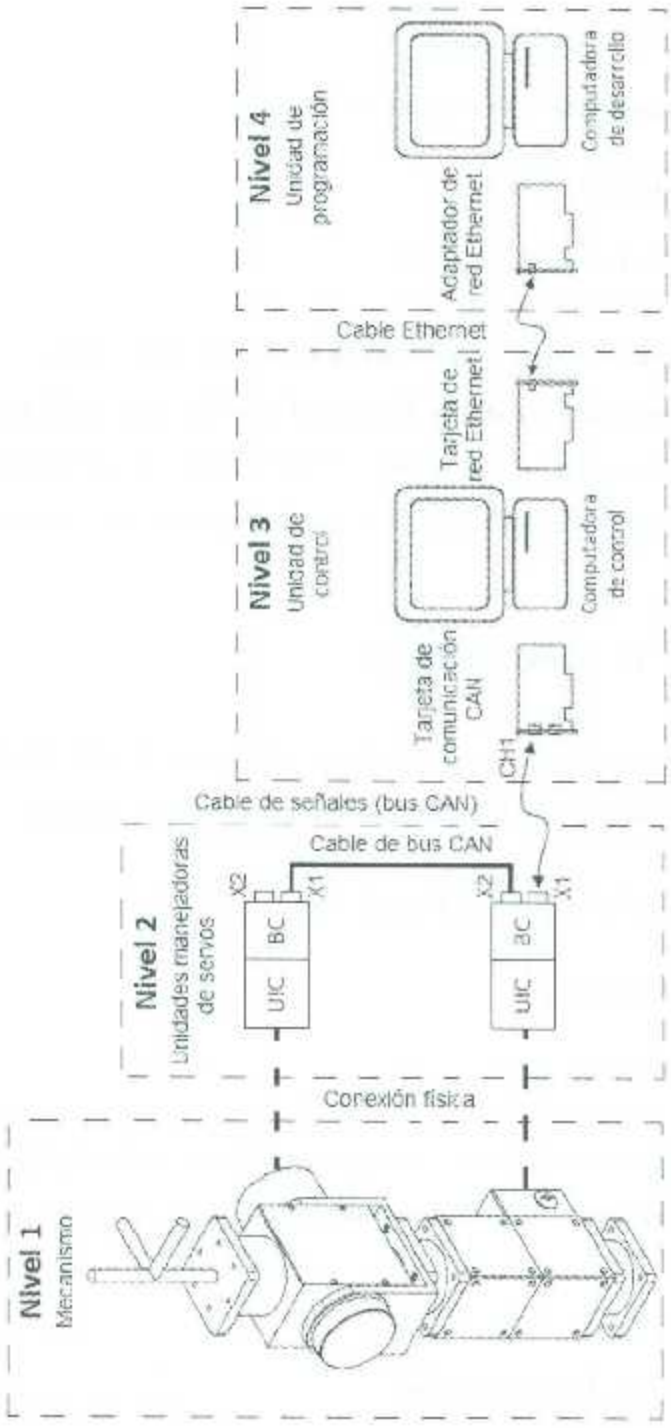
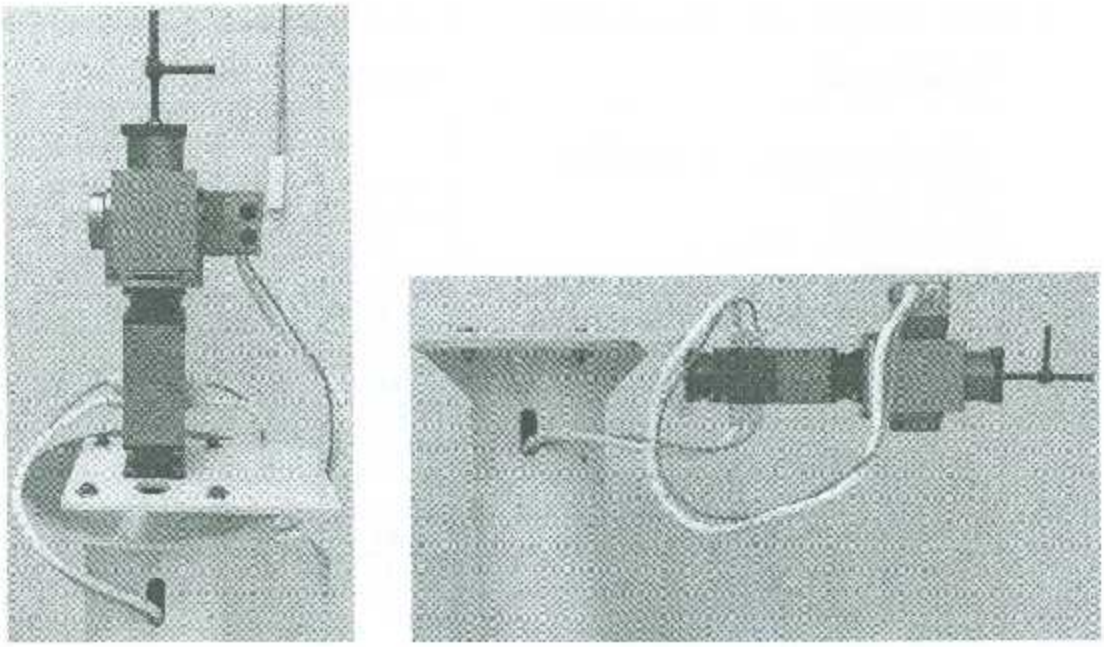


Figura 3.1: Arquitectura de la muñeca MASKARA



(a) (b)

Figura 3.2: Ensamble de la muñeca esférica. (a) Configuración vertical y (b) configuración horizontal

Estos módulos contienen servomotores del tipo BLDC y están equipados con transmisiones armónicas (harmonic drives) con un factor de reducción de 161 para el módulo tipo PR-70, mientras que para el primero y segundo eje del módulo PW-70 la reducción es de 121 y 101 respectivamente. Es importante que no se sobrepasen los límites físicos de las primeras dos articulaciones de la muñeca para que los cables externos no se dañen (en caso de que se exceda el rango de movimiento permitido) y así evitar que se disparen funciones de protección propias de los módulos referentes a los límites de posición, velocidad, corriente, par, etc. Los datos técnicos más relevantes de los módulos PowerCube de la muñeca se presentan en la tabla 3.1. Para mayor información de los datos técnicos se puede consultar [Schunk, 2008].

La muñeca cuenta con frenos electromagnéticos solamente en las primeras dos articulaciones y su propósito es el de sostener la muñeca, no de frenarlo, ni de sostener cargas externas cuando esté desenergizado.

Tabla 3.1: Datos técnicos de los módulos PowerCube usados en la muñeca esférica

Descripción	PR-70	PW-70	
		Eje 1	Eje 2
Datos mecánicos de operación			
masa (kg)	1.7	1.8	
Par nominal (Nm)	23	12	2.1
Par máximo (Nm)	46	24	4
Ángulo de rotación (grad)	> 360	± 120	> 360
Máxima velocidad angular (grad/seg)	150	240	360
Máxima aceleración angular (grad/seg^2)	600	960	1440
Factor de reducción	161	121	101
Freno magnético	Si	Si	No
Datos eléctricos del motor			
Voltaje nominal (V)	24	24	24
Corriente nominal (A)	4	4	2.1
Máxima corriente (A)	8	8	4.4
Datos eléctricos de la UIC			
Voltaje nominal (V)	24	24	24
Corriente nominal (A)	0.5	0.5	0.5
Interfaces	RS-232; Profibus-DP; CAN		

3.2.2. Piezas para ensamble

La muñeca esférica cuenta con piezas de ensamble que fueron diseñadas para permitir conectar los eslabones fijos y móviles de la muñeca esférica. Entre las piezas de ensamble que se diseñaron están una base de acero en forma de escuadra para el montaje de la muñeca en posición vertical u horizontal, dos bridas cuadradas (una de acero y una de aluminio) para la sujeción de un eslabón fijo y un eslabón móvil de la muñeca, y un marco de coordenadas de tres ejes ortogonales (fabricado con material PVC) para visualizar la orientación del extremo final de la muñeca. Tanto el diseño como la ubicación exacta de las piezas de ensamble de la muñeca se pueden consultar en el apéndice B.

3.2.3. Órgano terminal

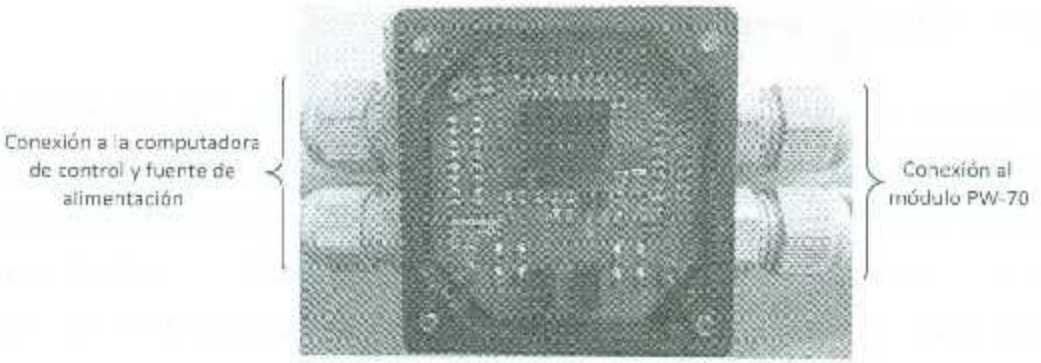
El órgano terminal se sitúa en el extremo final de la muñeca. Por el momento no se encuentra disponible una herramienta de trabajo como órgano terminal aunque se podría fácilmente instalar una con los elementos adecuados dentro de la estructura mecánica de la muñeca. Para las pruebas realizadas en esta tesis se diseñó y construyó un marco de coordenadas con tres ejes ortogonales (X, Y y Z) hecho de material PVC, con el fin de visualizar la orientación del extremo final de la muñeca.

3.3. Nivel 2: Unidades manejadoras de servos (UMS)

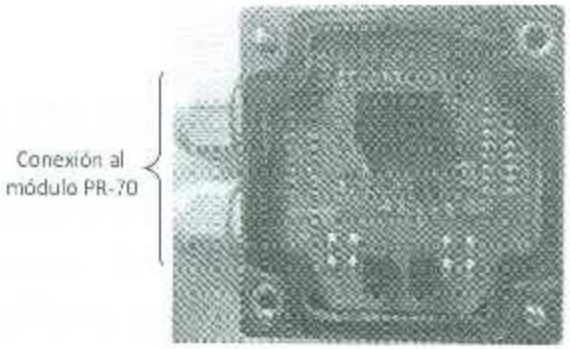
En este nivel se encuentran las UICs de los módulos PowerCube con su respectivo bloque de conexiones. Como ya se mencionó en el capítulo 2, el usuario no puede tener acceso directo a las UICs, sólo se puede acceder a éllas a través de los bloques de conexiones de cada módulo mostrados en la figura 3.3.

La muñeca MASKARA cuenta con dos UICs, una para el módulo PowerCube tipo PR-70 correspondiente a la primera articulación y otra para el módulo PowerCube tipo PW-70 correspondiente a la segunda y tercera articulación. El bloque de conexiones de cada UIC necesita como mínimo la conexión de dos cables, uno para la alimentación de potencia de los motores y la lógica de control, y otro para las señales de control bajo el protocolo CAN seleccionado; en el caso del bloque de conexiones del módulo PR-70 se necesitan conectar dos cables adicionales para transmitir las señales de control y de alimentación al módulo PW-70 (véase figura 3.3a), además de los dos cables provenientes de la computadora de control (contenida en el nivel 3) y de la fuente de alimentación.

En este nivel también se encuentra un paro de emergencia de la muñeca conectado al bloque de conexiones del primer módulo.



(a)



(b)

Figura 3.3: Bloque de conexión del módulo (a) PR-70 y (b) PW-70

3.3.1. Cables de señales y de energía

Para que el usuario pueda acceder desde el nivel 3 al nivel 2 de la arquitectura de la muñeca, es necesario utilizar un cable de señales para transportar las señales entre el mecanismo y la tarjeta bajo el protocolo CAN (especificación 2.0a), además de un cable de energía para suministrar la potencia necesaria a los servomotores y a la lógica de control de las UICs. Es importante mencionar que estos cables no estaban incluidos en los módulos PowerCube de la muñeca por lo que se procedió a armar un cable de señales especial para el protocolo CAN que cumpliera las especificaciones correspondientes, y también se armó un cable para la alimentación de energía que cumpliera con la características eléctricas de los servomotores especificadas en el capítulo 2.

La conexión de los cables de energía y de señales dentro de los bloques de conexiones

de los módulos PR-70 y PW-70 se especifica en las tablas 3.2 y 3.3, respectivamente.

Tabla 3.2: Líneas de conexión en bloque de conexiones del módulo PR-70

Terminal	*Descripción*
X1/BUS_H	Línea de transmisión CAN-H del cable de señales.
X1/BUS_L	Línea de transmisión CAN-L del cable de señales.
X1/24 V	Cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la UIC.
X1/PE	Tierra física (PE, Protective Earth) de cable de alimentación para la lógica de control de la UIC.
X1/GND	Tierra de cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la UIC.
X2/BUS_H	Línea de transmisión CAN-H del cable de bus CAN.
X2/BUS_L	Línea de transmisión CAN-L del cable de bus CAN.
X2/24 V	Cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la siguiente UIC.
X2/PE	Tierra física (PE, Protective Earth) de cable de alimentación para la lógica de control de la siguiente UIC.
X2/GND	Tierra de cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la siguiente UIC.
+UB	Cable de alimentación de 24 volts (a 10 A) para el servomotor.
-UB	Tierra de cable de alimentación de 24 volts (a 10 A) para el servomotor.

Tabla 3.3: Líneas de conexión en bloque de conexiones del módulo PW-70

Terminal	*Descripción*
X1/BUS_H	Línea de transmisión CAN-H del cable de bus CAN.
X1/BUS_L	Línea de transmisión CAN-L del cable de bus CAN.
X1/24 V	Cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la UIC.
X1/PE	Tierra física (PE, Protective Earth) de cable de alimentación para la lógica de control de la UIC.
X1/GND	Tierra de cable de alimentación de 24 volts (a 500 mA) para la lógica de control de la UIC.
+UB	Cable de alimentación de 24 volts (a 10 A) para los servomotores.
-UB	Tierra de cable de alimentación de 24 volts (a 10 A) para los servomotores.

Para mayor información de la manera de conectar los cables de energía y de señales en los módulos PowerCube de la muñeca se puede consultar [Schunk, 2011a] y

[Schunk, 2011b].

3.3.2. Botón de paro de emergencia

La muñeca MASKARA cuenta con un botón N.C. (normalmente cerrado) de paro de emergencia conectado a una entrada digital del módulo PR-70 (primera articulación de la muñeca). Por medio de este botón de paro es posible enviar una comando de paro de emergencia a cada servomotor de la muñeca y activar sus respectivos frenos magnéticos. El funcionamiento del botón de paro de emergencia es posible gracias a la constante supervisión del estado de la entrada digital (a la que se encuentra conectado) a través del algoritmo de control almacenado en el nivel 3 de la arquitectura de la muñeca. Cabe mencionar que para poder manejar la muñeca después de presionar el botón de paro de emergencia, es necesario liberar el botón, y reiniciar la fuente de alimentación externa de las UICs.

La conexión del botón de paro de emergencia dentro del bloque de conexiones del módulo PR-70 se especifica en la tabla 3.4.

Tabla 3.4: Conexión del paro de emergencia en bloque de conexiones del módulo PR-70

Terminal	*Descripción*
X3/GND/2	Tierra de entradas y salidas digitales; conectada en común con la terminal X1/GND para utilizar la terminal X1/24 V como fuente de alimentación de las señales conectadas a las entradas digitales
X3/IN 0	Entrada digital 0 conectada a un botón de paro de emergencia que conmuta una señal de 24 volts.

3.4. Nivel 3: Unidad de control

La unidad de control contiene la computadora de control que consta de una PC de escritorio que es responsable, por un lado, de recibir de la computadora de desarrollo el programa a ejecutar y, luego, una vez que este programa está en ejecución, mantiene una

constante comunicación (en tiempo real) con las UICs, enviando las referencias deseadas y recibiendo las señales de realimentación. La ejecución del programa en la computadora de control se realiza desde la computadora de desarrollo.

El monitor de la computadora de control es utilizado para visualizar en "tiempo de ejecución" variables de interés contenidas en el algoritmo de control que está almacenado en el CPU; como por ejemplo variables referentes al estado de la muñeca. En el capítulo 6 se mencionan todas las señales que pueden ser monitoreadas en la computadora de control.

Para la comunicación con los otros niveles, la computadora de control requiere tener instaladas una tarjeta de comunicación CAN y una tarjeta de red Ehernet.

3.4.1. Operación de la computadora de control

Para ejecutar el algoritmo de control, la computadora de control necesita inicializar su funcionamiento en modo kernel a través de un disco de arranque insertado en su unidad lectora de CD-ROM. El modo kernel es un modo de operación que permite administrar directamente (sin restricciones) los recursos de hardware disponibles en la PC sin necesidad de estar bajo un sistema operativo; entre estos recursos de hardware se encuentran, la memoria RAM (lugar donde se guarda el algoritmo de control) y las tarjetas de comunicación CAN y Ethernet, que se encuentran instaladas en dos puertos PCI de la PC. La creación del disco de arranque se hace a través del software de desarrollo descrito en el capítulo 6 y el apéndice C.

La computadora de control establece una comunicación vía protocolo TCP/IP con el nivel 4 de la muñeca a través de una tarjeta de red Ethernet; al mismo tiempo que se establece una comunicación en tiempo real con las UMSs de los módulos PowerCube (nivel 2) utilizando el protocolo CAN (especificación 2.0a) a través de una tarjeta de comunicación CAN que envía los datos por medio de un cable bus de dos líneas blindado.

Un resumen de los recursos de hardware requeridos en la computadora de control se muestra en la tabla 3.5.

Tabla 3.5: Recursos de hardware requeridos en la computadora de control

Hardware	Descripción
Puertos de comunicación PCI	Tarjeta de red Ethernet Tarjeta de comunicación CAN
CPU	Con procesador Intel Core 2 Quad
Dispositivos periféricos	Unidad lectora de CD-ROM Monitor VGA
Disco de arranque	CD autoarrancable en modo kernel
RAM	3063 MB de memoria RAM disponible

Se recomienda realizar una configuración de la BIOS de la computadora de control para optimizar el funcionamiento de la PC, además de establecer la secuencia de arranque para que lea primero el disco de arranque introducido en la unidad lectora de CD-ROM. Un procedimiento general para realizar la configuración de la BIOS de la computadora de control se describe en el apéndice D.

3.4.2. Tarjeta de comunicación CAN

La figura 3.4 muestra la tarjeta de comunicación CAN-AC2-PCI de la compañía Softing Industrial Automation que utiliza el protocolo CAN y que es compatible con los módulos PowerCube utilizados en la muñeca MASKARA, como se especifica en [Schunk, 2012].

Esta tarjeta cuenta con dos puertos de comunicación CAN independientes que manejan velocidades de transmisión de datos de hasta 1 Mbps; cada uno de los cuales tienen incorporados un interruptor para activar su propia resistencia de terminación de bus como se muestra en la figura 3.5. La tarjeta de comunicación CAN se encuentra instalada en un puerto PCI libre de la computadora de control para establecer un bus de comunicación a través del primer puerto CAN (CAN 1, según figura 3.5) con el fin de interconectar la computadora de control con las UICs de los módulos PowerCube de la muñeca. Mediante el enlace proporcionado por el bus de comunicación CAN es posible enviar periódicamente los comandos de control (posición, velocidad o corriente) empaquetados en mensajes de datos para el control de movimiento de los servomotores de la muñeca. Los comandos

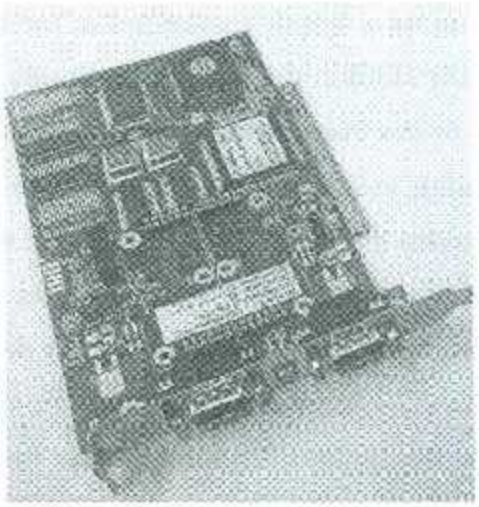


Figura 3.4: Tarjeta de comunicación CAN

de control son proporcionados por la computadora de control para especificar la posición deseada de las articulaciones de la muñeca. La computadora de control almacena los comandos de control como parte del algoritmo de control descargado en su memoria RAM desde la computadora de desarrollo; la cual utiliza una biblioteca xPC Target de Simulink para crear mensajes de datos bajo el protocolo CAN.

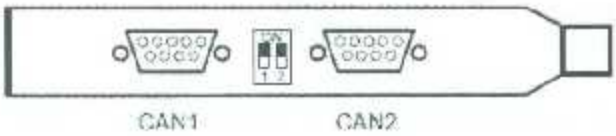


Figura 3.5: Puertos de comunicación e interruptores de terminación de bus de la tarjeta CAN

3.4.3. Conexión entre la computadora de control y las UMSs

La conexión física entre la computadora de control y las UMSs de la muñeca se hace mediante un bus de comunicación CAN, en donde el primer nodo conectado al bus es la computadora de control a través del primer puerto CAN (CAN 1) de la tarjeta de comunicación CAN, seguido de dos nodos correspondientes a las interfaces de comunicación de las UICs de los módulos PowerCube tipo PR-70 y PW-70; el cable a utilizar para

conectar cada uno de los nodos al bus de comunicación, corresponde a un cable especial definido por el estándar ISO 11898-2, del cual se presentarán más detalles en el capítulo 5. Este cable se compone de dos líneas de transmisión (conocidas como CAN-H y CAN-L) protegidas con un blindaje coaxial que debe ir conectado a la tierra física solamente en un punto como se especifica en [ESD, 2012]. Dicho cable se conecta en un extremo al primer nodo correspondiente a la computadora de control por medio de un conector DB-9 hembra que embona en el conector DB-9 macho de la tarjeta de comunicación CAN; el otro extremo del cable de bus CAN se extiende siguiendo una topología de bus (o lineal) hasta llegar a los bloques de conexión de las UICs para conectar las interfaces de comunicación CAN del segundo y tercer nodo correspondientes a los módulos PowerCube, tal y como se especificó en las tablas 3.2 y 3.3. La figura 3.6 muestra un esquema de la conexión física entre la computadora de control y las UICs.

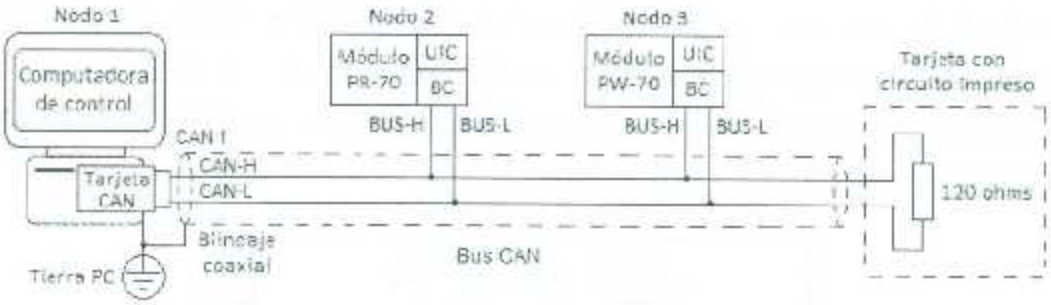


Figura 3.6: Conexión física entre la computadora de control y las UICs.

En el extremo derecho de la figura 3.6 se puede observar una tarjeta con circuito impreso (incluida con los módulos PowerCube), la cual, es responsable de activar la resistencia de terminación en el extremo final (después del tercer nodo) del bus de comunicación CAN para prevenir reflexión de señales provocadas por alta impedancia; la resistencia de terminación del otro lado del bus (después de la computadora de control) es activada a través del primer interruptor de la tarjeta CAN. Para mayor información de las resistencias de terminación del bus CAN se puede consultar el capítulo 5.

3.4.4. Tarjeta de red Ethernet

La figura 3.7 muestra la tarjeta de red Ethernet con bus PCI de la compañía Accton Technology Corporation, la cual puede alcanzar velocidades de transmisión de datos de hasta 100 Mbps y que además es compatible con el software de xPC Target utilizado en la computadora de desarrollo de la muñeca como se indica en [MathWorks, 2010b]. Esta tarjeta se inserta en un puerto PCI libre de la computadora de control para establecer un enlace de comunicación vía protocolo TCP/IP entre los niveles 3 y 4 del robot a través de un cable cruzado Ethernet que es conectado en un extremo al puerto de la tarjeta de red Ethernet instalada en la computadora de control y al otro extremo al adaptador de red Ethernet de la computadora de desarrollo. Es por medio de esta tarjeta que es posible comunicar la computadora de control con la computadora de desarrollo con el fin de cargar dentro de la computadora de control el algoritmo de control creado por la computadora de desarrollo, y de llevar a cabo la supervisión y control de ejecución de la aplicación desde el nivel 4.

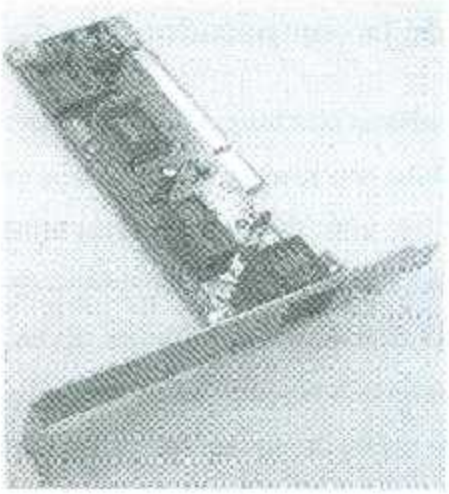


Figura 3.7: Tarjeta de red Ethernet

3.5. Nivel 4: Unidad de programación

Este es el nivel de control más alto de la arquitectura del robot con el cual puede interactuar el usuario. Esta formado por una computadora portátil llamada computadora de desarrollo que dispone de un adaptador de red Ethernet que se conecta a través de un cable cruzado Ethernet al nivel 3 de la muñeca. Esta computadora tiene instalado el paquete Matlab/Simulink con el cual se crea una interfaz gráfica que permite la programación de la muñeca MASKARA con el fin de evaluar diferentes algoritmos de control en tiempo real. Una vez creado un algoritmo de control con Simulink, éste puede ser seleccionado y descargado a la computadora de control con la ayuda de la interfaz gráfica vía protocolo TCP/IP para su ejecución y, luego, una vez que este algoritmo está en ejecución, mantiene una constante comunicación (no determinística) con la computadora de desarrollo, permitiendo al usuario detener la ejecución del algoritmo así como la oportunidad de modificar ciertos parámetros del algoritmo.

3.5.1. Operación de la computadora de desarrollo

Para las pruebas realizadas en esta tesis, se utilizó como computadora de desarrollo una computadora portátil Toshiba con procesador Intel i3, la cual tiene instalado el software Matlab R2010a, Simulink 7.5, xPC Target 4.3 y otras utilidades, instalados en Windows XP (32-bit), aunque es posible instalar cualquier sistema operativo de Windows (32 o 64-bit) compatible con Matlab/Simulink y xPC Target. Es en la computadora de desarrollo donde se diseña la programación a bloques con la ayuda de Simulink; y una vez diseñado el algoritmo de control, se utiliza la herramienta xPC Target para crear un código que será descargado a la computadora de control para su ejecución en tiempo real.

En la computadora de desarrollo, el usuario emplea una interfaz gráfica que le da la posibilidad de controlar la ejecución del algoritmo de control, así como modificar ciertos parámetros del mismo antes y durante la ejecución. Es importante mencionar que la comunicación entre la computadora de desarrollo y la computadora de control no se lleva a

cabo en tiempo real.

El algoritmo de control generado desde la computadora de desarrollo, es eficaz gracias a una biblioteca xPC Target de Simulink que contiene funciones básicas para manejar hardware externo, entre los cuales esta la tarjeta de comunicación CAN llamada CAN-AC2-PCI.

Los requerimientos mínimos de hardware que debe poseer la computadora de desarrollo se presentan en la tabla 3.6.

Tabla 3.6: Requerimientos mínimos de hardware de la computadora de desarrollo

Hardware	Descripción
Tarjeta de comunicación	Adaptador de red Ethernet.
CPU	Procesador Pentium, Athlon, o posterior.
Dispositivos periféricos	Disco duro con 60 MB de espacio libre. Unidad grabable de CD-ROM (opcional).
RAM	128 MB o más.

La unidad grabable de CD-ROM es opcional debido a que sólo se necesita para grabar un archivo de configuración del enlace de comunicación vía TCP/IP entre la computadora de desarrollo y la computadora de control en un CD de arranque en modo kernel (para el CPU de la computadora de control) creado mediante el software de MathWorks xPC Target en la computadora de desarrollo. Este archivo de configuración se puede guardar en una unidad de almacenamiento externa y grabar después en un CD autoarrancable mediante otra computadora o laptop que si disponga de una unidad grabable de CD-ROM. El procedimiento para crear el CD autoarrancable en modo kernel con el archivo de configuración se puede ver en el apéndice C.

3.5.2. Adaptador de red Ethernet

Se trata de un puerto de red Ethernet disponible en la computadora de desarrollo, con velocidades de transmisión de datos de hasta 100 Mbps. Es por medio de este adaptador de red Ethernet que es posible establecer un enlace de comunicación vía protocolo TCP/IP

con la computadora de control. Para la conexión física entre el adaptador de red Ethernet y la computadora de control, es necesario utilizar un cable cruzado Ethernet.

3.5.3. Conexión entre la computadora de desarrollo y la computadora de control

La conexión física entre la computadora de desarrollo y la computadora de control de la muñeca MASKARA se hace directamente a través de un cable cruzado de red Ethernet. Tanto la computadora de desarrollo como la computadora de control deben de contar con un adaptador o tarjeta de red Ethernet para establecer una comunicación vía protocolo TCP/IP. La computadora de control cuenta con una tarjeta de red Ethernet compatible con MathWorks xPC Target y la computadora de desarrollo cuenta con un simple puerto adaptador de red Ethernet. En la figura 3.8 se muestra un esquema de la conexión física entre la computadora de desarrollo y la computadora de control.

Figura 3.8: Conexión física entre la computadora de desarrollo y la computadora de control

Para establecer un enlace de comunicación vía protocolo TCP/IP entre la computadora de desarrollo y la computadora de control mediante el cable cruzado (crossover) de red Ethernet, es necesario especificar algunos parámetros de configuración de la red, como por ejemplo la dirección IP, la mascara de subred y la puerta de enlace. En el caso de la computadora de desarrollo, sólo es necesario definir una dirección IP fija y una máscara de subred con la ayuda de la plataforma de Windows XP como se especifica en el apéndice D.

Capítulo 4

Modelado de la muñeca MASKARA

En el presente capítulo se explican primero los conceptos y métodos generales para realizar el modelado cinemático y dinámico de robots manipuladores. Posteriormente se emplea esa teoría para obtener el modelo cinemático y dinámico de la muñeca esférica MASKARA que se describió en el capítulo 3. Se consideran dos posibles configuraciones de la muñeca esférica: configuración vertical y configuración horizontal.

4.1. Modelado de robots manipuladores

Los robots manipuladores son sistemas mecánicos formados por eslabones y conectados entre ellos por medio de articulaciones. Las articulaciones generalmente son de dos tipos: rotacionales y prismáticas. Cada articulación representa la interconexión entre dos eslabones. Tanto el modelado cinemático como dinámico de un manipulador son aspectos importantes en el momento de planear e implementar una aplicación específica para un robot manipulador. El modelado cinemático y dinámico de robots manipuladores ha sido tratado en un gran número de textos sobre mecánica y robótica [Spong y Vidyasagar, 1989, Sciavicco y Siciliano, 2000, Craig, 1989, Fu et al., 1988]. En esta sección se presentan brevemente los conceptos fundamentales del modelado cinemático

y dinámico de robots manipuladores.

4.1.1. Modelado cinemático

La cinemática de robots establece las relaciones entre las diferentes variables que intervienen en el movimiento del robot desde un punto de vista puramente geométrico, con el fin de especificar matemáticamente la ubicación espacial del robot. La ubicación espacial o configuración de un robot manipulador se puede especificar a través de la postura (es decir, la posición y la orientación) del órgano terminal. Para esto, generalmente se utiliza un vector de posición $\boldsymbol{p} \in \mathrm{R}^3$ y una matriz de rotación $R \in \mathrm{SO}(3)$, las cuales se pueden agrupar en la siguiente matriz conocida como matriz de transformación homogénea:

$$T = \begin{bmatrix} R & \boldsymbol{p} \\ 0 & 1 \end{bmatrix} \in \mathrm{R}^3 \times \mathrm{SO}(3) \equiv \mathrm{SE}(3). \tag{4.1}$$

Una propiedad importante de las matrices de transformación homogénea es que si ${}^0T_1$ representa la postura relativa del marco Σ_1 con respecto al marco Σ_0 y ${}^1T_2$ da la postura del marco Σ_2 con respecto al marco Σ_1, entonces el producto ${}^0T_1{}^1T_2$, se denota como ${}^0T_2$ y da la postura relativa de Σ_2 con respecto a Σ_0.

Se le llama modelo cinemático directo al conjunto de expresiones que permite expresar la postura del órgano terminal $(\boldsymbol{p}, R)$ en términos de las llamadas variables articulares que describen el movimiento relativo de un eslabón con respecto al anterior dentro de la cadena cinemática.

Por otra parte, la cinemática de velocidad establece la relación entre las velocidades articulares y las correspondientes velocidades lineales y angulares del órgano terminal.

A continuación se presenta el análisis de la cinemática directa de posición y de la cinemática de velocidad aplicado a manipuladores seriales.

Cinemática directa de posición

Un método muy usado para obtener el modelo cinemático directo de robots manipuladores seriales es el propuesto por J. Denavit y E. Hartenberg, conocido como método de Denavit-Hartenberg (o método D-H). Básicamente, el método consiste en asignar a cada eslabón de la cadena cinemática del manipulador un marco coordenado, luego se determinan las matrices de transformación homogénea que dan la postura relativa de un marco con respecto al anterior dentro de la cadena. Luego, de acuerdo a la propiedad mencionada anteriormente el producto de todas las matrices ${}^{i-1}T_i$ para $i = 1, 2, \ldots, n$ daría la postura del marco Σ_n (asociado al último eslabón) con respecto a Σ_0 que es el marco asociado a la base del robot. Y como en general la matriz ${}^{i-1}T_i$ queda en función de la variable articular q_i entonces

$$ {}^{0}T_n(q) = {}^{0}T_1(q_1)\,{}^{1}T_2(q_2)\ldots{}^{n-1}T_n(q_n) $$

donde $q = [q_1\ q_2\ \ldots\ q_n]^T \in \mathbb{R}^n$. Debe resultar claro ahora que ${}^{0}T_n(q)$ es una forma de representar el modelo cinemático directo del robot manipulador. Quizás la parte más difícil del método es la obtención de las matrices de transformación homogénea entre eslabones consecutivos. Afortunadamente, Denavit y Hartenberg observaron que si se siguen ciertas reglas para colocar los marcos de cada eslabón es posible obtener la postura relativa entre los marcos $i-1$ e i empleando sólo cuatro parámetros escalares, denotados como d_i, θ_i, a_i y α_i que son conocidos comúnmente como parámetros Denavit-Hartenberg (o simplemente parámetros D-H).

A continuación se presenta el algoritmo completo propuesto por Denavit y Hartenberg para obtener el modelo cinemático directo de manipuladores seriales:

Este algoritmo fue tomado de [Barrientos et al., 1997]

1. **Asignación de los marcos coordenados**

   Para la colocación de los $n+1$ marcos coordenados (uno por cada eslabón móvil más el de la base) asociados a un manipulador serial de n g.d.l. se deben seguir las

siguientes reglas:

a) Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

b) Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.

c) Localizar el eje de cada articulación. Si ésta es rotacional, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

d) Para i de 0 a $n-1$ situar el eje z_i sobre el eje de la articulación $i+1$.

e) Situar el origen del marco de la base Σ_0 en cualquier punto del eje z_0. Los ejes x_0 e y_0 se colocan de modo que formen un sistema dextrógiro (que cumpla con la regla de la mano derecha) con z_0.

f) Para i de 1 a $n-1$, situar el marco Σ_i (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría Σ_i en el punto de corte. Si fuesen paralelos Σ_i se situaría en la articulación $i+1$.

g) Situar x_i en la línea normal común a z_{i-1} y z_i.

h) Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

i) Situar el sistema Σ_n en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

2. **Obtención de los parámetros Denavit-Hartenberg**

Un robot manipulador serial de n grados de libertad requiere $4n$ parámetros D-H para quedar completamente caracterizado. Estos cuatro parámetros D-H (de cada articulación) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que lo unen con el eslabón anterior. Considérese la figura 4.1

en donde se presentan los eslabones $i-1$ e i de una cadena cinemática con sus correspondientes marcos coordenados más un marco adicional ($\Sigma_{i'}$) que tiene la característica de que su eje $z_{i'}$ coincide con el eje z_{i-1} pero los ejes $x_{i'}$ y $y_{i'}$ coinciden con x_i y y_i. Este marco se usa para definir la postura relativa entre los marcos Σ_{i-1} y Σ_i como se explica abajo.

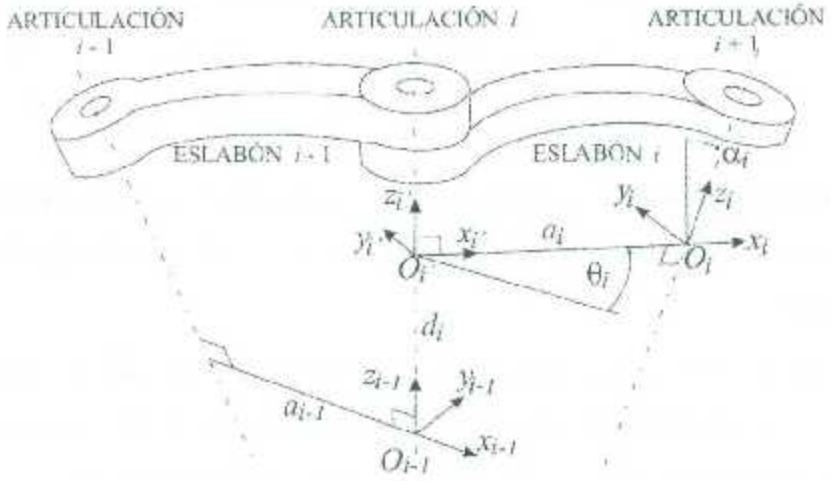


Figura 4.1: Parámetros geométricos Denavit-Hartenberg

Los cuatro parámetros D-H para la articulación i denominados d_i, θ_i, a_i y α_i se definen de la siguiente manera:

d_i: distancia entre los ejes x_{i-1} y x_i, medida a lo largo del eje z_{i-1}.

θ_i: ángulo entre los ejes x_{i-1} y x_i, medido alrededor del eje z_{i-1}.

a_i: distancia entre los ejes z_{i-1} y z_i, medida a lo largo del eje x_i.

α_i: ángulo entre los ejes z_{i-1} y z_i, medido alrededor del eje x_i.

Los parámetros a_i y α_i son siempre constantes y dependen de la geometría de los eslabones. Los parámetros θ_i y d_i dependen del tipo de articulación que conecta el eslabón *i-1* al eslabón *i*. En particular, si la articulación es rotacional, la variable es θ_i; si la articulación es traslacional (prismática), la variable es d_i.

3. **Cálculo de las matrices de transformación homogénea** ${}^{i-1}T_i$

Si los marcos coordenados se colocan siguiendo las reglas del paso 1, la obtención

de los parámetros D-H es directa, y la matriz de transformación homogénea ${}^{i-1}T_i$ para $i = 1, 2, \ldots, n$ se obtiene simplemente sustituyendo los parámetros D-H en la expresión siguiente [Barrientos et al., 1997]

$$T_i^{i-1}(q_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

De este modo, basta con identificar los parámetros D-H de cada eslabón para obtener las matrices de transformación T_i^{i-1} y relacionar así todos y cada uno de los eslabones del robot.

Debe observarse también que en (4.2) ${}^{i-1}T_i$ se puso explícitamente en función de q_i; esto es porque dependiendo del tipo de articulación entre los eslabones $i-1$ e i, ya sea el parámetro θ_i o el d_i son función de esta variable articular.

4. **Cálculo del modelo cinemático directo**

El modelo cinemático directo que da la postura del elemento terminal (en el eslabón n) con respecto al marco de referencia (Σ_0, en la base del robot), se obtiene simplemente aplicando la propiedad de multiplicación de las matrices de rotación

$${}^0T_n(q) = {}^0T_1(q_1){}^1T_2(q_2)\ldots{}^{n-1}T_n(q_n)$$

Cinemática de velocidad

Para establecer la relación entre las velocidades articulares y las correspondientes velocidades lineales y angulares del órgano terminal, se utiliza una matriz llamada *jacobiano geométrico* $J(\boldsymbol{q})$, es decir, si $\dot{\boldsymbol{q}} \in \mathbb{R}^3$ es el vector de velocidades articulares, $\boldsymbol{v} \in \mathbb{R}^3$ es el vector velocidad lineal y $\boldsymbol{\omega} \in \mathbb{R}^3$ es el vector velocidad angular del órgano terminal,

entonces

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J(q)\dot{q} \tag{4.3}$$

donde $J(q) \in \mathrm{R}^{6\times n}$. Es importante decir que la velocidad lineal del órgano terminal no es otra cosa más que la derivada temporal del vector de posición $p \in \mathbb{R}^3$, es decir:

$$v = \dot{p}. \tag{4.4}$$

La ecuación (4.3) se puede separar en dos:

$$\begin{aligned} v &= J_P(q)\dot{q} \\ \omega &= J_O(q)\dot{q}; \end{aligned}$$

donde $J_P, J_O \in \mathrm{R}^{3\times n}$ son, respectivamente, los jacobianos de posición y orientación tales que

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}. \tag{4.5}$$

Separando la matriz jacobiana (4.5) en vectores columna de (3×1) como se muestra a continuación:

$$J = \begin{bmatrix} j_{P1} & \cdots & j_{Pn} \\ j_{O1} & & j_{On} \end{bmatrix}. \tag{4.6}$$

Para calcular la matriz J es conveniente calcular las contribuciones de cada articulación distinguiendo el caso de una articulación prismática, del caso de una articulación

rotacional. Según [Sciavicco y Siciliano, 2000], se tiene que

$$\begin{bmatrix} j_{Pi} \\ j_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{para una articulación prismática} \\ \begin{bmatrix} z_{i-1} \times (p - p_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{para una articulación rotacional} \end{cases} \tag{4.7}$$

donde los vectores z_{i-1}, p y p_{i-1} están todos en función de variables articulares. En particular:

- z_{i-1} está dado por la tercera columna de la matriz de rotación R^0_{i-1},

$$z_{i-1} = R^0_1(q_1)\dots R^{i-2}_{i-1}(q_{i-1})z_0 \tag{4.8}$$

  donde $z_0 = [0\ 0\ 1]^T$ permite seleccionar la tercera columna.

- p está dado por los primeros tres elementos de la cuarta columna de la matriz de transformación T^0_n, es decir, los tres primeros elementos del vector $\bar{p} \in \mathbb{R}^4$ definido por

$$\bar{p} = T^0_1(q_1)\dots T^{n-1}_n(q_n)\bar{p}_0 \tag{4.9}$$

  donde $\bar{p}_0 = [0\ 0\ 0\ 1]^T$ permite seleccionar la cuarta columna.

- p_{i-1} está dado por los primeros tres elementos de la cuarta columna de la matriz de transformación T^0_{i-1}, es decir, los tres primeros elementos de $\bar{p}_{i-1}$

$$\bar{p}_{i-1} = T^0_1(q_1)\dots T^{i-2}_{i-1}(q_{i-1})\bar{p}_0 \tag{4.10}$$

  usando el mismo vector $\bar{p}_0 \in \mathbb{R}^4$ que en el punto anterior.

4.1.2. Modelado dinámico

El modelo dinámico de un robot manipulador describe la relación analítica entre su movimiento y las fuerzas que lo producen (accionadores, gravedad, etc.); para el modelado dinámico de un robot manipulador, se toma en cuenta el modelo cinemático del mismo y algunos parámetros físicos de la estructura, tales como masa, inercia, fricción, entre otros.

En general, el modelo dinámico de un robot manipulador se puede expresar como

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau,$$

donde $M(q) \in \mathbb{R}^{n\times n}$ es la matriz de inercias y $\ddot{q} \in \mathbb{R}^n$ es el vector de aceleraciones articulares, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ es la matriz de fuerzas centrífugas y de coriolis, $g(q) \in \mathbb{R}^n$ es el vector de pares gravitacionales y $\tau \in \mathbb{R}^n$ es el vector de pares aplicados al robot manipulador.

El procedimiento que se estudiará a continuación para obtener el modelo dinámico de un robot serial de n g.d.l., se basa en una metodología presentada en [Ramírez, 2008] y [Sciavicco y Siciliano, 2000], la cual requiere conocer los parámetros de Denavit-Hartenberg. Esta metodología se puede resumir en los siguientes 7 pasos sin considerar los efectos de la fricción para el cálculo del modelo dinámico:

1. **Calcular las matrices de transformación de los centros de masa de los eslabones con respecto a la base (eslabón 0).**
   Las matrices de transformación homogénea de cada uno de los centros de masa de los eslabones con respecto a la base ($T_{ci}^0 \in \mathrm{SE}(3)$) se obtienen a partir del producto de las matrices de transformación entre los marcos de los eslabones consecutivos referenciados a la base ($T_i^0 \in \mathrm{SE}(3)$, obtenida con ayuda de la ecuación (4.2)) y de las matrices de transformación de los centros de masa de cada eslabón a su respectiva

articulación ($T_{ii}^{i} \in \mathrm{SE}(3)$). Las matrices T_{ii}^{i} tienen la siguiente forma

$$T_{ii}^{i} = \begin{pmatrix} 1 & 0 & 0 & p_{xi} \\ 0 & 1 & 0 & p_{yi} \\ 0 & 0 & 1 & p_{zi} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.11}$$

donde la cuarta columna de la matriz denota el vector de posición $\boldsymbol{p}_{ii} \in R^4$ que contiene las tres coordenadas de posición (x, y, z) del centro de masa del eslabón i con respecto al marco Σ_i de la misma articulación. Finalmente la matriz T_{ii}^{0} se obtiene de

$$T_{ii}^{0} = T_{i}^{0} T_{ii}^{i}. \tag{4.12}$$

2. **Encontrar los jacobianos geométricos de posición y orientación del centro de masa de cada eslabón.**

El método del cálculo del jacobiano geométrico puede ser aplicado para los eslabones intermedios y no sólo para el órgano terminal como se explicó en la subsección 4.1.1. Los jacobianos que se consideran para el eslabón i son:

$$J_{P}^{(i_i)} = \begin{bmatrix} \boldsymbol{j}_{P_1}^{(i_i)} & \cdots & \boldsymbol{j}_{P_i}^{(i_i)} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \tag{4.13}$$

$$J_{O}^{(i_i)} = \begin{bmatrix} \boldsymbol{j}_{O_1}^{(i_i)} & \cdots & \boldsymbol{j}_{O_i}^{(i_i)} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}. \tag{4.14}$$

Las columnas de las matrices (4.13) y (4.14) pueden calcularse de acuerdo a (4.7), resultando

$$\boldsymbol{j}_{Pj}^{(i_i)} = \begin{cases} \boldsymbol{z}_{j-1} & \text{para una articulación prismática} \\ \boldsymbol{z}_{j-1} \times (\boldsymbol{p}_{i_i} - \boldsymbol{p}_{j-1}) & \text{para una articulación rotacional} \end{cases} \tag{4.15}$$

$$j_{O_j}^{(i)} = \begin{cases} 0 & \text{para una articulación prismática} \\ z_{j-1} & \text{para una articulación rotatoria} \end{cases} \tag{4.16}$$

donde p_i es el vector de posición del centro de masa del eslabón i, p_{j-1} es el vector de posición del origen del marco $j-1$ y z_{j-1} es el vector unitario del eje z del marco $j-1$.

3. **Calcular la energía cinética total del robot.**

La energía cinética total de un robot serial se obtiene sumando las contribuciones de cada uno de los eslabones. Con este fin, la ecuación a utilizar está expresada por

$$K_{l_i} = \frac{1}{2} m_{l_i} \dot{q}^T J_P^{(l_i)T} J_P^{(l_i)} \dot{q} + \frac{1}{2} q^T J_O^{(l_i)T} R_i^0 I_{l_i}^i R_i^{0T} J_O^{(l_i)} \dot{q} \tag{4.17}$$

donde m_{l_i} es la masa correspondiente al eslabón i, $\dot{q}$ es el vector de velocidades articulares, $J_P^{(l_i)}$ y $J_O^{(l_i)}$ son los jacobianos geométricos de posición y orientación del centro de masa de cada eslabón, R_i^0 son las matrices de rotación de los marcos de cada eslabón referenciados a la base y $I_{l_i}^i$ es una matriz simétrica que representa el tensor de inercia relativo al centro de masa de cada eslabón i representada por:

$$I_{l_i}^i = \begin{bmatrix} I_{xx_{l_i}} & -I_{xy_{l_i}} & -I_{xz_{l_i}} \\ -I_{xy_{l_i}} & I_{yy_{l_i}} & -I_{yz_{l_i}} \\ -I_{xz_{l_i}} & -I_{yz_{l_i}} & I_{zz_{l_i}} \end{bmatrix}.$$

Una vez calculadas las contribuciones de cada eslabón con la ecuación (4.17), la suma de cada una de estas contribuciones representa la energía cinética total del robot, resultando en

$$K = \sum_{i=1}^{n} K_{l_i} = \frac{1}{2} \dot{q}^T M(q) \dot{q} \tag{4.18}$$

donde $M(q) \in R^{n \times n}$ es una matriz simétrica definida positiva denominada matriz

de inercia.

4. **Calcular la energía potencial total del robot.**

   La energía potencial total de un robot serial se obtiene sumando las contribuciones de cada uno de los eslabones. Con este fin, la ecuación a utilizar está expresada por

$$\mathcal{U}_i = -m_i \boldsymbol{g}_0^T \boldsymbol{p}_i \tag{4.19}$$

   donde m_i es la masa correspondiente al eslabón i, $\boldsymbol{g}_0$ es el vector de aceleración debido a la gravedad referenciada al marco de la base (e.g., $\boldsymbol{g}_0 = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$ si z es el eje vertical), $\boldsymbol{p}_i$ es el vector de posición del centro de masa del eslabón i referenciado al marco de la base. La energía potencial total resulta entonces en

$$\mathcal{U} = \sum_{i=1}^{n} \mathcal{U}_i. \tag{4.20}$$

5. **Obtener la matriz $M(\boldsymbol{q})$ a partir de la energía cinética.**

   La matriz de inercia $M(\boldsymbol{q})$ de un robot serial, se extrae directamente de la ecuación (4.18) agrupando términos.

6. **Obtener la matriz $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ con los coeficientes de Christoffel.**

   La matriz $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathrm{R}^{n \times n}$ llamada matriz de fuerzas centrífugas y de Coriolis, se puede obtener a través de los coeficientes o símbolos de Christoffel $c_{ijk}(\boldsymbol{q})$ utilizados en [Kelly y Santibáñez, 2003] y definidos como:

$$c_{ijk}(\boldsymbol{q}) = \frac{1}{2}\left[\frac{\partial M_{kj}(\boldsymbol{q})}{\partial q_i} + \frac{\partial M_{ki}(\boldsymbol{q})}{\partial q_j} - \frac{\partial M_{ij}(\boldsymbol{q})}{\partial q_k}\right], \tag{4.21}$$

   donde $M_{ij}(\boldsymbol{q})$ denota el ij-ésimo elemento de la matriz de inercia $M(\boldsymbol{q})$. En efecto, el kj-ésimo elemento $C_{kj}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ de la matriz $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ puede obtenerse de la siguiente

manera:

$$C_{kj}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} c_{1jk}(\boldsymbol{q}) \\ c_{2jk}(\boldsymbol{q}) \\ \vdots \\ c_{njk}(\boldsymbol{q}) \end{bmatrix}^T \dot{q}. \tag{4.22}$$

7. **Obtener $g(q)$ con el gradiente de la energía potencial.**

   El vector $g(q) \in \mathrm{R}^n$ conocido como vector de fuerzas o pares gravitacionales puede obtenerse con el gradiente de la energía potencial $U(q)$ descrita en la ecuación (4.20), por lo tanto puede calcularse con la fórmula

$$g(\boldsymbol{q}) = \frac{\partial U(\boldsymbol{q})}{\partial q}. \tag{4.23}$$

4.2. Modelo cinemático de la muñeca

Considerando la estructura de la muñeca mostrada en la figura 4.2 se puede observar que los ejes correspondientes a las tres variables articulares (q_1,q_2 y q_3) están numerados de forma ascendente comenzando por la articulación más próxima a la base y terminando con la articulación donde se situará el órgano terminal; aunque por el momento el órgano terminal es representado de manera simbólica con un marco de coordenadas solamente. Cabe recordar que una muñeca esférica se caracteriza por tener un centro geométrico en donde se intersecan cada uno de sus ejes rotacionales.

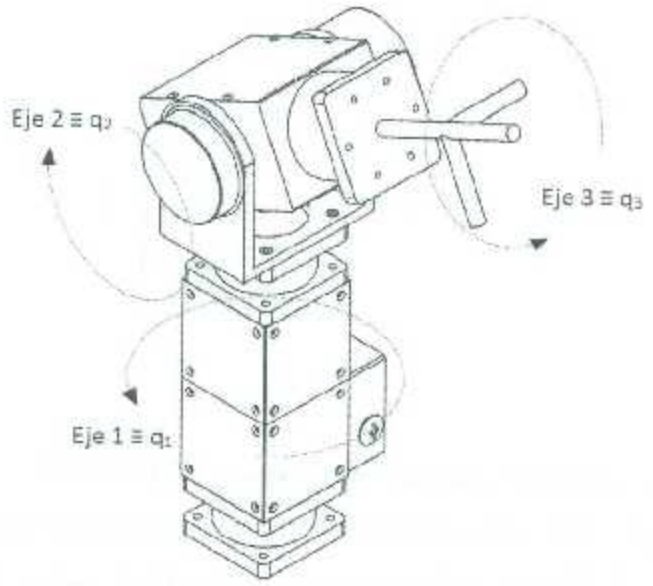


Figura 4.2: Muñeca esférica MASKARA

Tabla 4.1: Parámetros de Denavit-Hartenberg para la muñeca esférica

Eslabón	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	$d_1 - 0.292\,\mathrm{m}$	q_1
2	0	$\pi/2$	0	q_2
3	0	0	$d_2 = 0.1651\,\mathrm{m}$	q_3

4.2.1. Cinemática de posición

Para obtener el modelo cinemático directo de la muñeca esférica MASKARA se hará uso del método de Denavit-Hartenberg (método D-H) presentado en la subsección 4.1.1. El método D-H establece como primer paso un algoritmo para la asignación de los marcos coordenados asociados a cada eslabón; al ser aplicado se obtiene la figura 4.3.

Una vez asignados los marcos coordenados asociados a cada eslabón, se procede con el siguiente paso del método D-H correspondiente a los parámetros D-H. La muñeca esférica cuenta con 12 parámetros D-H mostrados en la tabla 4.1.

El siguiente paso del método D-H, consiste en utilizar los parámetros D-H encontrados

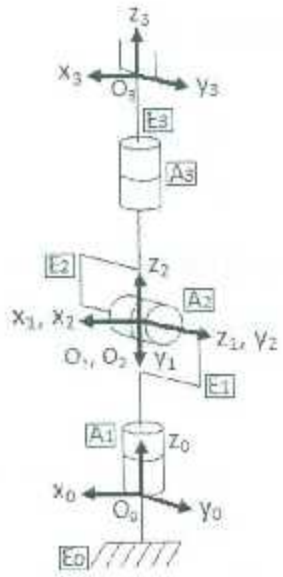


Figura 4.3: Asignación de los marcos coordenados de la muñeca.

para formar las matrices de transformación homogénea T_i^{i-1} que relacionan los marcos coordenados asociados a cada eslabón. Las ecuaciones (4.24), (4.25) y (4.26) establecen las matrices de transformación homogénea de la muñeca según la ecuación (4.2).

$$T_1^0(q_1) = \begin{bmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

$$T_2^1(q_2) = \begin{bmatrix} \cos(q_2) & 0 & \sin(q_2) & 0 \\ \sin(q_2) & 0 & -\cos(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

$$T_3^2(q_3) = \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 & 0 \\ \sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

Como último paso del método D-H, se calcula el modelo cinemático directo de la muñeca multiplicando las matrices de transformación relativas como se muestra en la ecuación (4.27).

$$T_3^0(q) = T_1^0 T_2^1 T_3^2 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 & d_2c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 & d_2s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 & d_1 + d_2c_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.27}$$

Para simplificar la notación de funciones trigonométricas en los elementos de la matriz descrita en la ecuación (4.27), se hizo uso de la siguiente sustitución:

$$c_i = \cos(q_i) \qquad y \qquad s_i = \sin(q_i) \tag{4.28}$$

4.2.2. Cinemática de velocidad

En base al estudio de la cinemática de velocidad presentada en la subsección 4.1.1, se obtendrá el jacobiano geométrico $J(\boldsymbol{q})$ propio de la muñeca esférica. Para simplificar la notación en las ecuaciones que contienen funciones trigonométricas se hace uso de la sustitución mostrada en la ecuación (4.28).

Según la ecuación (4.6) el jacobiano geométrico de la muñeca esférica $J(\boldsymbol{q}) \in \mathbb{R}^{6\times3}$ tiene la forma

$$J = \begin{bmatrix} \boldsymbol{j}_{P_1} & \boldsymbol{j}_{P_2} & \boldsymbol{j}_{P_3} \\ \boldsymbol{j}_{O_1} & \boldsymbol{j}_{O_2} & \boldsymbol{j}_{O_3} \end{bmatrix} \tag{4.29}$$

donde los elementos de cada columna se calculan con las ecuaciones (4.7) a (4.10), resultando en:

$$\boldsymbol{j}_{P_1} = \boldsymbol{z}_0 \times (\boldsymbol{p} - \boldsymbol{p}_0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(\begin{bmatrix} d_2 c_1 s_2 \\ d_2 s_1 s_2 \\ d_1 + d_2 c_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) - \begin{bmatrix} -d_2 s_1 s_2 \\ d_2 c_1 s_2 \\ 0 \end{bmatrix},$$

$$\boldsymbol{j}_{O_1} = \boldsymbol{z}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\boldsymbol{j}_{P_2} = \boldsymbol{z}_1 \times (\boldsymbol{p} - \boldsymbol{p}_1) = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \times \left(\begin{bmatrix} d_2 c_1 s_2 \\ d_2 s_1 s_2 \\ d_1 + d_2 c_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \right) - \begin{bmatrix} d_2 c_1 c_2 \\ d_2 s_1 c_2 \\ -d_2 s_2 \end{bmatrix},$$

$$\boldsymbol{j}_{O_2} = \boldsymbol{z}_1 = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix},$$

$$\boldsymbol{j}_{P_3} = \boldsymbol{z}_2 \times (\boldsymbol{p} - \boldsymbol{p}_2) = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \times \left(\begin{bmatrix} d_2 c_1 s_2 \\ d_2 s_1 s_2 \\ d_1 + d_2 c_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\boldsymbol{j}_{O_3} = \boldsymbol{z}_2 = \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix}.$$

Reescribiendo (4.29) con sus respectivos valores de columnas, se obtiene

$$J(q) = \begin{bmatrix} -d_2 s_1 s_2 & d_2 c_1 c_2 & 0 \\ d_2 c_1 s_2 & d_2 s_1 c_2 & 0 \\ 0 & -d_2 s_2 & 0 \\ 0 & -s_1 & c_1 s_2 \\ 0 & c_1 & s_1 s_2 \\ 1 & 0 & c_2 \end{bmatrix}. \tag{4.30}$$

La ecuación (4.30) muestra que el jacobiano geométrico de la muñeca tiene singularidades cuando $q_2 = n\pi$ con $n \subset \mathbb{Z}$.

4.3. Modelo dinámico de la muñeca

En esta sección se muestra paso a paso cual es el procedimiento para obtener el modelo dinámico de la muñeca esférica MASKARA. El procedimiento para la obtención del modelo dinámico se basa en la metodología presentada en la subsección 4.1.2.

Para simplificar la notación en las ecuaciones que contienen funciones trigonométricas se hace uso de la sustitución mostrada en la ecuación (4.28).

4.3.1. Matrices de transformación

Debido a que el fabricante de los módulos PowerCube (utilizados en la muñeca MASKARA) no proporciona una hoja técnica referente a los centros de masa de cada módulo, se procede a calcular las matrices de transformación de los centros de masa de los tres eslabones (de la muñeca) referidos a su marco asociado utilizando la ecuación (4.11), es

decir:

$$T^1_{lc1} = \begin{bmatrix} 1 & 0 & 0 & p_{x_1} \\ 0 & 1 & 0 & p_{y_1} \\ 0 & 0 & 1 & p_{z_1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T^2_{lc2} = \begin{bmatrix} 1 & 0 & 0 & p_{x_2} \\ 0 & 1 & 0 & p_{y_2} \\ 0 & 0 & 1 & p_{z_2} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T^3_{lc3} = \begin{bmatrix} 1 & 0 & 0 & p_{x_3} \\ 0 & 1 & 0 & p_{y_3} \\ 0 & 0 & 1 & p_{z_3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y utilizando la ecuación (4.12) se obtienen las siguientes matrices de transformación de los centros de masa de cada eslabón referidas a la base de la muñeca esférica (eslabón 0):

$$T^0_{lc1} = T^0_1 T^1_{lc1} = \begin{bmatrix} c_1 & 0 & -s_1 & c_1 p_{x_1} - p_{z_1} s_1 \\ s_1 & 0 & c_1 & c_1 p_{z_1} + p_{x_1} s_1 \\ 0 & -1 & 0 & d_1 - p_{y_1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.31}$$

$$T^0_2 = T^0_1 T^1_2 = \begin{bmatrix} c_1 c_2 & -s_1 & c_1 s_2 & 0 \\ c_2 s_1 & c_1 & s_1 s_2 & 0 \\ -s_2 & 0 & c_2 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.32}$$

$$T^0_{lc2} = T^0_2 T^2_{lc2} = \begin{bmatrix} c_1 c_2 & -s_1 & c_1 s_2 & c_1 c_2 p_{x_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 \\ c_2 s_1 & c_1 & s_1 s_2 & c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2 \\ -s_2 & 0 & c_2 & d_1 + c_2 p_{z_2} - p_{x_2} s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.33}$$

$$T^0_{lc3} = T^0_3 T^3_{lc3} = \begin{bmatrix} c_1 c_2 c_3 - s_1 s_3 & -c_3 s_1 - c_1 c_2 s_3 & c_1 s_2 & p_{x_{lc3}} \\ c_1 s_3 + c_2 c_3 s_1 & c_1 c_3 - c_2 s_1 s_3 & s_1 s_2 & p_{y_{lc3}} \\ -c_3 s_2 & s_2 s_3 & c_2 & p_{z_{lc3}} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.34}$$

donde los elementos del vector de posición de la matriz de transformación homogénea

descrita por la ecuación (4.34) son:

$$p_{x_{cg3}} = c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2,$$
$$p_{y_{cg3}} = p_{z_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2,$$
$$p_{z_{cg3}} = d_1 + c_3 d_2 + c_2 p_{z_3} - c_3 p_{z_3} s_2 + p_{y_3} s_2 s_3.$$

4.3.2. Jacobianos geométricos

En esta subsección se emplea el método de cálculo del jacobiano geométrico especificado en la subsección 4.1.2 y basado en [Sciavicco y Siciliano, 2000], para obtener los jacobianos geométricos de posición y orientación del centro de masa de cada eslabón de la muñeca esférica con la ayuda de las ecuaciones (4.13) a (4.16). Para el análisis, el marco 0 se toma como el marco de la base de la muñeca.

Eslabón 1

Como la articulación que sujeta el eslabón 1 es rotacional, y tomando en cuenta que los vectores $\boldsymbol{z}_0$ y $\boldsymbol{p}_0$ se forman con los primeros 3 elementos de las columnas 3 y 4 de la matriz de transformación del marco 0 referenciado al marco de la base, es decir, T_0^0; y considerando que $\boldsymbol{p}_{l_1}$ es equivalente al vector de posición de la matriz de transformación del centro de masa para el eslabón $i = 1$ referenciado a la base, se obtienen las siguientes columnas para los jacobianos geométricos de posición y orientación del eslabón 1:

$$J_{P1}^{(l_1)} = \boldsymbol{z}_0 \times (\boldsymbol{p}_{l_1} - \boldsymbol{p}_0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(\begin{bmatrix} c_1 p_{x_1} - p_{y_1} s_1 \\ c_1 p_{z_1} + p_{x_1} s_1 \\ d_1 - p_{y_1} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$
$$= \begin{bmatrix} -c_1 p_{z_1} - p_{x_1} s_1 \\ c_1 p_{x_1} - p_{z_1} s_1 \\ 0 \end{bmatrix},$$

$$J_{O1}^{(l_1)} = \boldsymbol{z}_0.$$

las cuales forman los jacobianos

$$J_P^{(l_1)} = \begin{bmatrix} -c_1 p_{z_1} - p_{x_1} s_1 & 0 & 0 \\ c_1 p_{x_1} - p_{z_1} s_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad J_O^{(l_1)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Eslabón 2

Como la articulación que sujeta al eslabón 2 es rotacional y respetando el mismo procedimiento realizado con el eslabón anterior, las columnas de los jacobianos geométricos de posición y orientación del eslabón 2 son:

$$\boldsymbol{j}_{P1}^{(l_2)} = \boldsymbol{z}_0 \times (\boldsymbol{p}_{l_2} - \boldsymbol{p}_0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(\begin{bmatrix} c_1 c_2 p_{x_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 \\ c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2 \\ d_1 + c_2 p_{z_2} - p_{x_2} s_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right),$$

$$= \begin{bmatrix} -c_1 p_{y_2} - c_2 p_{x_2} s_1 - p_{z_2} s_1 s_2 \\ c_1 c_2 p_{z_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 \\ 0 \end{bmatrix},$$

$$\boldsymbol{j}_{O1}^{(l_2)} = \boldsymbol{z}_0,$$

$$\boldsymbol{j}_{P2}^{(l_2)} = \boldsymbol{z}_1 \times (\boldsymbol{p}_{l_2} - \boldsymbol{p}_1) = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \times \left(\begin{bmatrix} c_1 c_2 p_{x_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 \\ c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2 \\ d_1 + c_2 p_{z_2} - p_{x_2} s_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \right),$$

$$= \begin{bmatrix} c_1 (c_2 p_{z_2} - p_{x_2} s_2) \\ s_1 (c_2 p_{z_2} - p_{x_2} s_2) \\ -c_2 p_{x_2} - p_{z_2} s_2 \end{bmatrix},$$

$$\boldsymbol{j}_{O2}^{(l_2)} = \boldsymbol{z}_1,$$

las cuales forman los jacobianos

$$J_P^{(l_2)} = \begin{bmatrix} -c_1 p_{y_2} - c_2 p_{x_2} s_1 - p_{z_2} s_1 s_2 & c_1(c_2 p_{z_2} - p_{x_2} s_2) & 0 \\ c_1 c_2 p_{x_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 & s_1(c_2 p_{z_2} - p_{x_2} s_2) & 0 \\ 0 & -c_2 p_{x_2} - p_{z_2} s_2 & 0 \end{bmatrix},$$

$$J_O^{(l_2)} = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Eslabón 3

Como la articulación que sujeta al eslabón 3 es rotacional y respetando el mismo procedimiento realizado con el eslabón anterior, las columnas de los jacobianos geométricos de posición y orientación del eslabón 3 son:

$$j_{P1}^{(l_3)} = z_0 \times (p_{l_3} - p_0),$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \left(\begin{bmatrix} c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2 \\ p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2 \\ d_1 + c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right),$$

$$= \begin{bmatrix} -p_{z_3}(c_1 s_3 - c_2 c_3 s_1) & p_{y_3}(c_1 c_3 - c_2 s_1 s_3) - d_2 s_1 s_2 - p_{z_3} s_1 s_2 \\ c_1 d_2 s_2 - p_{y_3}(c_2 s_1 - c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2 \\ 0 \end{bmatrix},$$

$$j_{O1}^{(l_3)} = z_{0_1}$$

$$j_{P2}^{(l_3)} = z_1 \times (p_{l_3} - p_1),$$

$$= \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix} \times \left(\begin{bmatrix} c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2 \\ p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2 \\ d_1 + c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \right),$$

$$= \begin{bmatrix} c_1(c_2 d_2 - c_2 p_{z_3} - c_3 p_{x_3} s_2 - p_{y_3} s_2 s_3) \\ s_1(c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3) \\ -d_2 s_2 - p_{z_3} s_2 - c_2 c_3 p_{x_3} - c_2 p_{y_3} s_3 \end{bmatrix},$$

$$\boldsymbol{j}_{O2}^{(l_3)} = \boldsymbol{z}_1,$$

$$\boldsymbol{j}_{P3}^{(l_3)} = \boldsymbol{z}_2 \times (\boldsymbol{p}_{l_3} - \boldsymbol{p}_2),$$

$$= \begin{bmatrix} c_1 s_2 \\ s_1 s_2 \\ c_2 \end{bmatrix} \times \left(\begin{bmatrix} c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2 \\ p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) - d_2 s_1 s_2 + p_{z_3} s_1 s_2 \\ d_1 + c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \right),$$

$$= \begin{bmatrix} p_{y_3} s_1 s_3 - c_3 p_{x_3} s_1 - c_1 c_2 p_{x_3} s_3 - c_1 c_2 c_3 p_{y_3} \\ c_1 c_3 p_{x_3} - c_1 p_{y_3} s_3 - c_2 c_3 p_{y_3} s_1 - c_2 p_{x_3} s_1 s_3 \\ s_2(c_3 p_{y_3} - p_{x_3} s_3) \end{bmatrix},$$

$$\boldsymbol{j}_{O3}^{(l_3)} = \boldsymbol{z}_2,$$

las cuales forman los jacobianos

$$J_P^{(l_3)} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}, \quad J_O^{(l_3)} = \begin{bmatrix} 0 & -s_1 & c_1 s_2 \\ 0 & c_1 & s_1 s_2 \\ 1 & 0 & c_2 \end{bmatrix},$$

donde los elementos del jacobiano $J_P^{(l_3)}$ son:

$$a_{1,1} = -p_{z_3}(c_1 s_3 - c_2 c_3 s_1) - p_{y_3}(c_1 c_3 - c_2 s_1 s_3) - d_2 s_1 s_2 - p_{z_3} s_1 s_2$$

$$a_{1,2} = c_1(c_2 d_2 - c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3)$$

$$a_{1,3} = p_{y_3} s_1 s_3 - c_3 p_{x_3} s_1 - c_1 c_2 p_{x_3} s_3 - c_1 c_2 c_3 p_{y_3}$$

$$a_{2,1} = c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) + c_1 p_{z_3} s_2$$

$a_{2,2} = s_1(c_2 d_2 + c_2 p_{x_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3)$

$a_{2,3} = c_1 c_3 p_{x_3} - c_1 p_{y_3} s_3 - c_2 c_3 p_{y_3} s_1 - c_2 p_{x_3} s_1 s_3$

$a_{3,1} = 0$

$a_{3,2} = -d_2 s_2 - p_{z_3} s_2 - c_2 c_3 p_{z_3} + c_2 p_{y_3} s_3$

$a_{3,3} = s_2(c_3 p_{y_3} + p_{x_3} s_3)$

4.3.3. Cálculo de la energía cinética

A continuación se utilizará la ecuación (4.17) para obtener la contribución de energía cinética de cada eslabón de la muñeca.

Eslabón 1

La energía cinética de este eslabón es:

$$\mathcal{K}_{l_1} = \frac{1}{2} m_{l_1} \dot{\boldsymbol{q}}^T J_P^{(l_1)^T} J_P^{(l_1)} \dot{\boldsymbol{q}} + \frac{1}{2} \dot{\boldsymbol{q}}^T J_O^{(l_1)^T} R_1^0 I_{l_1}^1 R_1^{0^T} J_O^{(l_1)} \dot{\boldsymbol{q}},$$

$$\mathcal{K}_{l_1} = \frac{1}{2} \dot{\boldsymbol{q}}^T \begin{bmatrix} I_{y_{3'}} + m_{l_1}(p_{x_1}^2 + p_{z_1}^2) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\boldsymbol{q}}.$$

Eslabón 2

La energía cinética de este eslabón es:

$$\mathcal{K}_{l_2} = \frac{1}{2} m_{l_2} \dot{\boldsymbol{q}}^T J_P^{(l_2)^T} J_P^{(l_2)} \dot{\boldsymbol{q}} + \frac{1}{2} \dot{\boldsymbol{q}}^T J_O^{(l_2)^T} R_2^0 I_{l_2}^2 R_2^{0^T} J_O^{(l_2)} \dot{\boldsymbol{q}},$$

$$\mathcal{K}_{l_2} = \frac{1}{2} \dot{\boldsymbol{q}}^T M_2 \dot{\boldsymbol{q}},$$

donde los elementos de la matriz M_2 son:

$$M_{2_{(1,1)}} = m_{l_2}(c_2^2 p_{x_2}^2 + 2c_2 p_{x_2} p_{z_2} s_2 - p_{y_2}^2 + p_{z_2}^2 s_2^2) + I_{zz_2} c_2^2 + I_{xx_2} s_2^2 - 2I_{xz_2} c_2 s_2$$
$$M_{2_{(1,2)}} = M_{2_{(2,1)}} = I_{xy_2} s_2 \quad I_{yz_2} c_2 - m_{l_2} p_{y_2}(c_2 p_{z_2} - p_{x_2} s_2)$$
$$M_{2_{(1,3)}} = M_{2_{(3,1)}} = 0$$
$$M_{2_{(2,2)}} = I_{yy_2} + m_{l_2}(p_{x_2}^2 + p_{z_2}^2)$$
$$M_{2_{(2,3)}} = M_{2_{(3,2)}} = 0$$
$$M_{2_{(3,3)}} = 0$$

Eslabón 3

La energía cinética de este eslabón es:

$$K_{l_3} = \frac{1}{2} m_{l_3} \dot{\boldsymbol{q}}^T J_P^{(l_3)^T} J_P^{(l_3)} \dot{\boldsymbol{q}} + \frac{1}{2} \dot{\boldsymbol{q}}^T J_O^{(l_3)^T} R_3^0 I_{l_3}^3 R_3^{0^T} J_O^{(l_3)} \dot{\boldsymbol{q}},$$
$$K_{l_3} = \frac{1}{2} \dot{\boldsymbol{q}}^T M_3 \dot{\boldsymbol{q}},$$

donde los elementos de la matriz M_3 son:

$$M_{3_{(1,1)}} = m_{l_3}((p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2)^2 + (p_{x_3}(s_1 s_3 - c_1 c_2 c_3) - p_{y_3}(c_3 s_1 - c_1 c_2 s_3) - c_1 d_2 s_2 - c_1 p_{z_3} s_2)^2) + I_{zz_3} c_2^2 + I_{xx_3} c_3^2 s_2^2 + I_{yy_3} s_2^2 s_3^2 + 2I_{xy_3} c_3 s_2^2 s_3 + 2I_{xz_3} c_2 c_3 s_2 - 2I_{yz_3} c_2 s_2 s_3$$
$$M_{3_{(1,2)}} = M_{3_{(2,1)}} = 2I_{xy_3} c_3^2 s_2 - I_{yz_3} c_2 c_3 - I_{xz_3} c_2 s_3 - I_{xy_3} s_2 - m_{l_3} p_{x_3} p_{y_3} s_2 - I_{xx_3} c_3 s_2 s_3 + I_{yy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{y_3} - c_2 c_3 m_{l_3} p_{y_3} p_{z_3} - c_2 d_2 m_{l_3} p_{x_3} s_3 - c_2 m_{l_3} p_{x_3} p_{z_3} s_3 + 2c_3^2 m_{l_3} p_{x_3} p_{y_3} s_2 + c_3 m_{l_3} p_{x_3}^2 s_2 s_3 - c_3 m_{l_3} p_{y_3}^2 s_2 s_3$$
$$M_{3_{(1,3)}} = M_{3_{(3,1)}} = I_{xz_3} c_2 - I_{xz_3} c_3 s_2 - I_{yz_3} s_2 s_3 + c_2 m_{l_3} p_{x_3}^2 + c_2 m_{l_3} p_{y_3}^2 + c_3 d_2 m_{l_3} p_{x_3} s_2 + c_3 m_{l_3} p_{x_3} p_{z_3} s_2 - d_2 m_{l_3} p_{y_3} s_2 s_3 - m_{l_3} p_{y_3} p_{z_3} s_2 s_3$$
$$M_{3_{(2,2)}} = I_{xx_3} - I_{xx_3} c_3^2 - I_{yy_3} c_3^2 + d_2^2 m_{l_3} + m_{l_3} p_{y_3}^2 + m_{l_3} p_{z_3}^2 + c_3^2 m_{l_3} p_{x_3}^2 - c_3^2 m_{l_3} p_{y_3}^2 - 2I_{xy_3} c_3 s_3 + 2d_2 m_{l_3} p_{z_3} - 2c_3 m_{l_3} p_{z_3} p_{y_3} s_3$$
$$M_{3_{(2,3)}} = M_{3_{(3,2)}} = -I_{yz_3} c_3 - I_{xz_3} s_3 - m_{l_3}(d_2 - p_{z_3})(c_3 p_{y_3} + p_{x_3} s_3)$$
$$M_{3_{(3,3)}} = I_{zz_3} + m_{l_3}(p_{x_3}^2 + p_{y_3}^2)$$

Finalmente la energía total del robot se obtiene sumando la energía de los 3 eslabones,

resultando en:

$$\begin{aligned} \mathcal{K} &= \mathcal{K}_{l_1} + \mathcal{K}_{l_2} + \mathcal{K}_{l_3}, \\ &= \frac{1}{2}\dot{q}^T M(q)\dot{q}, \end{aligned} \tag{4.35}$$

donde los elementos de la matriz $M(q)$ son:

$$\begin{aligned} M_{(1,1)} &= I_{yy_1} + m_{l_2}(c_2^2 p_{x_2}^2 + 2c_2 p_{x_2} p_{z_2} s_2 + p_{y_2}^2 + p_{z_2}^2 s_2^2) + m_{l_3}((p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - \\ &\quad c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2)^2 + (p_{x_3}(s_1 s_3 - c_1 c_2 c_3) - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - c_1 d_2 s_2 - \\ &\quad c_1 p_{z_3} s_2)^2) + I_{xx_2} c_2^2 + I_{zz_3} c_2^2 + I_{xx_3} s_2^2 + m_{l_1}(p_{x_1}^2 + p_{z_1}^2) + I_{xx_3} c_3^2 s_2^2 + I_{yy_3} s_2^2 s_3^2 + 2I_{xz_3} c_2 s_2 + \\ &\quad 2I_{xy_3} c_3 s_2^2 s_3 + 2I_{xz_3} c_2 c_3 s_2 - 2I_{yz_3} c_2 s_2 s_3 \\ M_{(1,2)} &= M_{(2,1)} = I_{xy_2} s_2 - I_{yz_2} c_2 - I_{xy_3} s_2 - m_{l_2} p_{y_2}(c_2 p_{x_1} - p_{x_2} s_2) - I_{yz_3} c_2 c_3 - I_{xz_3} c_2 s_3 + \\ &\quad 2I_{xy_3} c_3^2 s_2 - m_{l_3} p_{x_3} p_{y_3} s_2 - I_{xx_3} c_3 s_2 s_3 + I_{yy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{y_3} - c_2 c_3 m_{l_3} p_{y_3} p_{z_3} - \\ &\quad c_2 d_2 m_{l_3} p_{x_3} s_3 - c_2 m_{l_3} p_{x_3} p_{z_3} s_3 + 2c_3^2 m_{l_3} p_{x_3} p_{y_3} s_2 + c_3 m_{l_3} p_{x_3}^2 s_2 s_3 - c_3 m_{l_3} p_{y_3}^2 s_2 s_3 \\ M_{(1,3)} &= M_{(3,1)} = I_{zz_3} c_2 + I_{xz_3} c_3 s_2 - I_{yz_3} s_2 s_3 - c_2 m_{l_3} p_{x_3}^2 + c_2 m_{l_3} p_{y_3}^2 + c_3 d_2 m_{l_3} p_{x_3} s_2 + \\ &\quad c_3 m_{l_3} p_{x_3} p_{z_3} s_2 - d_2 m_{l_3} p_{y_3} s_2 s_3 - m_{l_3} p_{y_3} p_{z_3} s_2 s_3 \\ M_{(2,2)} &= I_{xx_2} + I_{yy_3} - I_{xx_3} c_3^2 + I_{yy_3} c_3^2 - d_2^2 m_{l_3} + m_{l_3} p_{y_3}^2 - m_{l_3} p_{z_3}^2 - m_{l_2}(p_{x_2}^2 + p_{z_2}^2) + c_3^2 m_{l_3} p_{x_3}^2 - \\ &\quad c_3^2 m_{l_3} p_{y_3}^2 - 2I_{xy_3} c_3 s_3 + 2d_2 m_{l_3} p_{z_3} - 2c_3 m_{l_3} p_{x_3} p_{y_3} s_3 \\ M_{(2,3)} &= M_{(3,2)} = -I_{yz_3} c_3 - I_{xz_3} s_3 - m_{l_3}(d_2 + p_{z_3})(c_3 p_{y_3} + p_{x_3} s_3) \\ M_{(3,3)} &= I_{zz_3} + m_{l_3}(p_{x_3}^2 + p_{y_3}^2) \end{aligned}$$

4.3.4. Cálculo de la energía potencial

En esta subsección se calcula la energía potencial total para las dos diferentes configuraciones de posición (vertical y horizontal) de la muñeca esférica. Considérese la variable g como la fuerza ejercida por la gravedad. Para el análisis de configuración vertical de la muñeca se toma como referencia la figura 4.3, y para el análisis de configuración horizontal de la muñeca se toma como referencia la misma figura pero girada +90 grados con respecto al eje x del marco 0 y con el eje y del marco coordenado (de la herramienta) apuntando hacia arriba en lugar del eje z, tal y como se muestra en las figuras B.4 y B.5. Utilizando la ecua-

ción (4.19) se obtiene la contribución de cada eslabón de la energía potencial de la muñeca.

Configuración vertical

Eslabón 1

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_1} = -m_{l_1}\boldsymbol{g}_0^T\boldsymbol{p}_{l_1} = -m_{l_1}\begin{bmatrix} 0 & 0 & -g \end{bmatrix}\begin{bmatrix} c_1 p_{x_1} - p_{z_1}s_1 \\ c_1 p_{z_1} - p_{x_1}s_1 \\ d_1 - p_{y_1} \end{bmatrix} = m_{l_1}g(d_1 - p_{y_1}).$$

Eslabón 2

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_2} = -m_{l_2}\boldsymbol{g}_0^T\boldsymbol{p}_{l_2} = -m_{l_2}\begin{bmatrix} 0 & 0 & -g \end{bmatrix}\begin{bmatrix} c_1c_2p_{x_2} - p_{y_2}s_1 + c_1p_{z_2}s_2 \\ c_1p_{y_2} - c_2p_{x_2}s_1 + p_{z_2}s_1s_2 \\ d_1 + c_2p_{z_2} - p_{x_2}s_2 \end{bmatrix}$$

$$= m_{l_2}g(d_1 + c_2p_{z_2} - p_{x_2}s_2).$$

Eslabón 3

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_3} = -m_{l_3}\boldsymbol{g}_0^T\boldsymbol{p}_{l_3}$$

$$= -m_{l_3}\begin{bmatrix} 0 & 0 & -g \end{bmatrix}\begin{bmatrix} c_1d_2s_2 - p_{y_3}(c_3s_1 - c_1c_2s_3) - p_{x_3}(s_1s_3 - c_1c_2c_3) + c_1p_{z_3}s_2 \\ p_{x_3}(c_1s_3 + c_2c_3s_1) + p_{z_3}(c_1c_3 - c_2s_1s_3) + d_2s_1s_2 + p_{z_3}s_1s_2 \\ d_1 + c_2d_2 + c_2p_{z_3} - c_3p_{z_3}s_2 + p_{y_3}s_2s_3 \end{bmatrix}$$

$$- m_{l_3}g(d_1 + c_2d_2 + c_2p_{z_3} - c_3p_{z_3}s_2 + p_{y_3}s_2s_3).$$

La energía potencial total de la muñeca en su configuración vertical está dada por la suma de las energías potenciales de todos sus eslabones (como se mostró en la ecuación (4.20)),

resultando en:

$$\mathcal{U} = (m_{l_1}(d_1 - p_{y_1}) + m_{l_2}(d_1 + c_2 p_{z_2} - p_{x_2} s_2) - m_{l_3}(d_1 + c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3))g. \quad (4.36)$$

Configuración horizontal

Eslabón 1

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_1} = -m_{l_1} \boldsymbol{g}_0^T \boldsymbol{p}_{l_1} = -m_{l_1} \begin{bmatrix} 0 & -g & 0 \end{bmatrix} \begin{bmatrix} c_1 p_{x_1} - p_{z_1} s_1 \\ c_1 p_{z_1} - p_{x_1} s_1 \\ d_1 - p_{y_1} \end{bmatrix} = m_{l_1} g (c_1 p_{z_1} - p_{z_1} s_1).$$

Eslabón 2

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_2} = -m_{l_2} \boldsymbol{g}_0^T \boldsymbol{p}_{l_2} = -m_{l_2} \begin{bmatrix} 0 & -g & 0 \end{bmatrix} \begin{bmatrix} c_1 c_2 p_{x_2} - p_{y_2} s_1 + c_1 p_{z_2} s_2 \\ c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2 \\ d_1 - c_2 p_{z_2} - p_{x_2} s_2 \end{bmatrix}$$

$$= m_{l_2} g (c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2).$$

Eslabón 3

La energía potencial de este eslabón es:

$$\mathcal{U}_{l_3} = -m_{l_3} \boldsymbol{g}_0^T \boldsymbol{p}_{l_3}$$

$$= -m_{l_3} \begin{bmatrix} 0 & -g & 0 \end{bmatrix} \begin{bmatrix} c_1 d_2 s_2 - p_{y_3}(c_3 s_1 + c_1 c_2 s_3) - p_{x_3}(s_1 s_3 - c_1 c_2 c_3) - c_1 p_{z_2} s_2 \\ p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 - p_{z_3} s_1 s_2 \\ d_1 + c_2 d_2 + c_2 p_{z_3} - c_3 p_{x_3} s_2 + p_{y_3} s_2 s_3 \end{bmatrix}$$

$$= m_{l_3} g (p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2).$$

La energía potencial total de la muñeca en su configuración horizontal está dada por la

suma de las energías potenciales de todos sus eslabones (véase ecuación (4.20)), resultando en:

$$\mathcal{U} = (m_{l_1}(c_1 p_{x_1} + p_{x_1} s_1) + m_{l_2}(c_1 p_{y_2} + c_2 p_{x_2} s_1 + p_{z_2} s_1 s_2) + m_{l_3}(p_{x_3}(c_1 s_3 + c_2 c_3 s_1) + p_{y_3}(c_1 c_3 - c_2 s_1 s_3) + d_2 s_1 s_2 + p_{z_3} s_1 s_2))g. \tag{4.37}$$

4.3.5. Matriz de inercias $M(q)$

Como se explica en [Kelly y Santibáñez, 2003], la matriz de inercias puede extraerse directamente de la energía cinética del manipulador. La matriz de inercias para la muñeca esférica se obtiene de la ecuación (4.35), donde $M(q) \in \mathrm{R}^{3\times 3}$ es una matriz simétrica definida positiva.

4.3.6. Matriz de fuerzas centrífugas y de Coriolis $C(q, \dot{q})$

Esta matriz se calcula con los símbolos o coeficientes de Christoffel presentados en la subsección 4.1.2 con las ecuaciones (4.21) y (4.22). Por la complejidad de las operaciones, se usó el software matemático MATLAB (Matrix Laboratory) para hacer las operaciones relacionadas a los símbolos de Christoffel y simplificar las ecuaciones resultantes lo más posible. Los elementos de $C(q, \dot{q})$ que resultaron fueron los siguientes:

$$C_{(1,1)} = \dot{q}_2(I_{yz_3}s_3 - I_{xz_3}c_3 \quad I_{xz_2} + 2I_{xz_2}c_2^2 + I_{xz_2}c_2 s_2 + I_{yy_3}c_2 s_2 - I_{zz_3}c_2 s_2 - I_{xz_3}c_2 s_2 - m_{l_2}p_{x_2}p_{z_2} + 2I_{xz_3}c_2^2 c_3 - 2I_{yz_3}c_2^2 s_3 - c_3 d_2 m_{l_3}p_{x_3} - c_3 m_{l_3}p_{x_3}p_{z_3} + d_2 m_{l_3}p_{y_3}s_3 - m_{l_3}p_{y_3}p_{z_3}s_3 + I_{xz_3}c_2 c_3^2 s_2 - I_{yy_3}c_2 c_3^2 s_2 + c_2 d_2^2 m_{l_3}s_2 + 2c_2^2 m_{l_2}p_{x_2}p_{z_2} - c_2 m_{l_2}p_{x_2}^2 s_2 - c_2 m_{l_3}p_{y_3}^2 s_2 + c_2 m_{l_2}p_{z_2}^2 s_2 + c_2 m_{l_3}p_{z_3}^2 s_2 - c_2 c_3^2 m_{l_3}p_{x_3}^2 s_2 + c_2 c_3^2 m_{l_3}p_{y_3}^2 s_2 + 2I_{xy_3}c_2 c_3 s_2 s_3 + 2c_2 d_2 m_{l_3}p_{x_3}s_2 + 2c_3^2 c_3 d_2 m_{l_3}p_{x_3} - 2c_2^2 c_3 m_{l_3}p_{x_3}p_{z_3} - 2c_2^2 d_2 m_{l_3}p_{y_3}s_3 - 2c_2^2 m_{l_3}p_{y_3}p_{z_3}s_3 + 2c_2 c_3 m_{l_3}p_{x_3}p_{y_3}s_2 s_3) - \dot{q}_3(I_{xy_3} - I_{zy_3}c_2^2 - 2I_{xy_3}c_3^2 + 2I_{xy_3}c_2^2 c_3^2 + I_{xz_3}c_3 s_3 - I_{yy_3}c_3 s_3 + m_{l_3}p_{x_3}p_{y_3} - I_{xz_3}c_2^2 c_3 s_3 + I_{yz_3}c_2^2 c_3 s_3 - c_2^2 m_{l_3}p_{x_3}p_{y_3} - 2c_2^2 m_{l_3}p_{z_3}p_{y_3} - c_3 m_{l_3}p_{x_3}^2 s_3 + c_3 m_{l_3}p_{y_3}^2 s_3 + I_{yz_3}c_2 c_3 s_2 + I_{xz_3}c_2 s_2 s_3 + 2c_2^2 c_3^2 m_{l_3}p_{x_3}p_{y_3} + c_2^2 c_3 m_{l_3}p_{x_3}^2 s_3 - c_2^2 c_3 m_{l_3}p_{y_3}^2 s_3 - c_2 c_3 d_2 m_{l_3}p_{y_3}s_2 + c_2 c_3 m_{l_3}p_{y_3}p_{z_3}s_2 +$$

$c_2 d_2 m_{l_3} p_{x_3} s_2 s_3 + c_2 m_{l_3} p_{x_3} p_{z_3} s_2 s_3)$

$C_{(1,2)} = \dot{q}_1(I_{yz_3} s_3 - I_{xz_3} c_3 - I_{xz_2} + 2I_{xz_2} c_2^2 + I_{xz_2} c_2 s_2 + I_{yy_3} c_2 s_2 - I_{zz_2} c_2 s_2 - I_{zz_3} c_2 s_2 -$
$m_{l_2} p_{x_2} p_{z_2} - 2I_{xz_3} c_2^2 c_3 - 2I_{yz_3} c_2^2 s_3 - c_3 d_2 m_{l_3} p_{x_3} - c_3 m_{l_3} p_{x_3} p_{z_3} + d_2 m_{l_3} p_{y_3} s_3 + m_{l_3} p_{y_3} p_{z_3} s_3$
$+ I_{xx_3} c_2 c_3^2 s_2 - I_{yy_3} c_2 c_3^2 s_2 + c_2 d_2^2 m_{l_3} s_2 + 2c_2^2 m_{l_2} p_{x_2} p_{z_2} - c_2 m_{l_2} p_{x_2}^2 s_2 - c_2 m_{l_3} p_{y_3}^2 s_2 +$
$c_2 m_{l_3} p_{x_3}^2 s_2 + c_2 m_{l_3} p_{z_3}^2 s_2 - c_2 c_3^2 m_{l_3} p_{x_3}^2 s_2 - c_2 c_3^2 m_{l_3} p_{y_3}^2 s_2 + 2I_{xy_3} c_2 c_3 s_2 s_3 + 2c_2 d_2 m_{l_3} p_{z_3} s_2 +$
$2c_2^2 c_3 d_2 m_{l_3} p_{x_3} + 2c_2^2 c_3 m_{l_3} p_{x_3} p_{z_3} - 2c_2^2 d_2 m_{l_3} p_{y_3} s_3 - 2c_2^2 m_{l_3} p_{y_3} p_{z_3} s_3 + 2c_2 c_3 m_{l_3} p_{x_3} p_{y_3} s_2 s_3) +$
$\dot{q}_2(I_{xy_2} c_2 - I_{xy_3} c_2 + I_{yz_2} s_2 + m_{l_2} p_{y_2}(c_2 p_{x_2} + p_{z_2} s_2) + I_{yz_3} c_3 s_2 + I_{xz_3} s_2 s_3 + 2I_{xy_3} c_2 c_3^2 -$
$c_2 m_{l_3} p_{x_3} p_{y_3} - I_{xx_3} c_2 c_3 s_3 + I_{yy_3} c_2 c_3 s_3 + c_3 d_2 m_{l_3} p_{y_3} s_2 + c_3 m_{l_3} p_{y_3} p_{z_3} s_2 + d_2 m_{l_3} p_{x_3} s_2 s_3 +$
$m_{l_3} p_{x_3} p_{z_3} s_2 s_3 + 2c_2 c_3^2 m_{l_3} p_{x_3} p_{y_3} + c_2 c_3 m_{l_3} p_{x_3}^2 s_3 - c_2 c_3 m_{l_3} p_{y_3}^2 s_3) - \dot{q}_3 s_2(I_{zz_3}/2 + (m_{l_3} p_{z_3}^2)/2$
$+ (m_{l_3} p_{y_3}^2)/2 + (I_{xx_3} cos(2q_3))/2 - (I_{yy_3} cos(2q_3))/2 + I_{xy_3} sin(2q_3) - (m_{l_3} p_{x_3}^2 cos(2q_3))/2$
$- (m_{l_3} p_{y_3}^2 cos(2q_3))/2 + m_{l_3} p_{x_3} p_{y_3} sin(2q_3))$

$C_{(1,3)} = -\dot{q}_1(I_{xy_3} - I_{xy_3} c_2^2 - 2I_{xy_2} c_3^2 + 2I_{xy_3} c_2^2 c_3^2 + I_{xx_3} c_3 s_3 - I_{yy_3} c_3 s_3 + m_{l_3} p_{x_3} p_{y_3} - I_{xx_3} c_2^2 c_3 s_3 +$
$I_{yy_3} c_2^2 c_3 s_3 - c_2^2 m_{l_3} p_{x_3} p_{y_3} - 2c_2^2 m_{l_3} p_{x_3} p_{y_3} - c_3 m_{l_3} p_{z_3}^2 s_3 - c_3 m_{l_3} p_{y_3}^2 s_3 + I_{yz_3} c_2 c_3 s_2 +$
$I_{xz_3} c_2 s_2 s_3 + 2c_2^2 c_3^2 m_{l_3} p_{x_3} p_{y_3} + c_2^2 c_3 m_{l_3} p_{z_3}^2 s_3 - c_2^2 c_3 m_{l_3} p_{y_3}^2 s_3 + c_2 c_3 d_2 m_{l_3} p_{y_3} s_2 +$
$c_2 c_3 m_{l_3} p_{y_3} p_{z_3} s_2 + c_2 d_2 m_{l_3} p_{x_3} s_2 s_3 + c_2 m_{l_3} p_{x_3} p_{z_3} s_2 s_3) - \dot{q}_3 s_2(I_{yz_3} c_3 + I_{xz_3} s_3 + c_3 d_2 m_{l_3} p_{y_3}$
$+ c_3 m_{l_3} p_{y_3} p_{z_3} + d_2 m_{l_3} p_{z_3} s_3 + m_{l_3} p_{x_3} p_{z_3} s_3) - \dot{q}_2 s_2(I_{zz_3}/2 + (m_{l_3} p_{x_3}^2)/2 + (m_{l_3} p_{y_3}^2)/2 +$
$(I_{xx_3} cos(2q_3))/2 - (I_{yy_3} cos(2q_3))/2 - I_{xy_3} sin(2q_3) - (m_{l_3} p_{x_3}^2 cos(2q_3))/2 +$
$(m_{l_3} p_{y_3}^2 cos(2q_3))/2 - m^{l_3} p_{x_3} p_{y_3} sin(2q_3))$

$C_{(2,1)} = \dot{q}_3((I_{xx_3} s_2)/2 - (I_{yy_3} s_2)/2 + (I_{zz_3} s_2)/2 + m_{l_3} p_{y_3}^2 s_2 - I_{xz_3} c_2 c_3 - I_{yz_3} c_2 s_3 - I_{xx_3} c_3^2 s_2 +$
$I_{yy_3} c_3^2 s_2 + c_3^2 m_{l_3} p_{x_3}^2 s_2 - c_3^2 m_{l_3} p_{y_3}^2 s_2 - 2I_{xy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{x_3} - c_2 c_3 m_{l_3} p_{x_3} p_{z_3} +$
$c_2 d_2 m_{l_3} p_{y_3} s_3 + c_2 m_{l_3} p_{y_3} p_{z_3} s_3 - 2c_3 m_{l_3} p_{x_3} p_{y_3} s_2 s_3) - \dot{q}_1(I_{yz_3} s_3 - I_{xz_3} c_3 - I_{xz_2} - 2I_{xz_2} c_2^2 -$
$I_{xx_3} c_2 s_2 + I_{yy_3} c_2 s_2 \quad I_{zz_2} c_2 s_2 - I_{zz_3} c_2 s_2 - m_{l_2} p_{x_2} p_{z_2} + 2I_{xz_3} c_2^2 c_3 - 2I_{yz_3} c_2^2 s_3 - c_3 d_2 m_{l_3} p_{x_3}$
$- c_3 m_{l_3} p_{x_3} p_{z_3} + d_2 m_{l_3} p_{y_3} s_3 + m_{l_3} p_{y_3} p_{z_3} s_3 + I_{xz_3} c_2 c_3^2 s_2 - I_{yy_3} c_2 c_3^2 s_2 + c_2 d_2^2 m_{l_3} s_2 +$
$2c_2^2 m_{l_2} p_{x_2} p_{z_2} - c_2 m_{l_2} p_{x_2}^2 s_2 - c_2 m_{l_3} p_{y_2}^2 s_2 - c_2 m_{l_2} p_{z_2}^2 s_2 + c_2 m_{l_3} p_{z_3}^2 s_2 - c_2 c_3^2 m_{l_3} p_{x_3}^2 s_2 +$
$c_2 c_3^2 m_{l_3} p_{y_3}^2 s_2 + 2I_{xy_3} c_2 c_3 s_2 s_3 + 2c_2 d_2 m_{l_3} p_{z_3} s_2 + 2c_2^2 c_3 d_2 m_{l_3} p_{x_3} + 2c_2^2 c_3 m_{l_3} p_{x_3} p_{z_3} -$
$2c_2^2 d_2 m_{l_3} p_{y_3} s_3 - 2c_2^2 m_{l_3} p_{y_3} p_{z_3} s_3 + 2c_2 c_3 m_{l_3} p_{z_3} p_{y_3} s_2 s_3)$

$C_{(2,2)} = -\dot{q}_3((m_{l_3} sin(2q_3) p_{x_3}^2)/2 + m_{l_3} cos(2q_3) p_{x_3} p_{z_3} - (m_{l_3} sin(2q_3) p_{y_3}^2)/2 + I_{xy_3} cos(2q_3) -$
$(I_{xx_3} sin(2q_3))/2 - (I_{yy_3} sin(2q_3))/2)$

$C_{(2,3)} = \dot{q}_1((I_{xx_3} s_2)/2 - (I_{yy_3} s_2)/2 + (I_{zz_3} s_2)/2 + m_{l_3} p_{y_3}^2 s_2 - I_{xz_3} c_2 c_3 + I_{yz_3} c_2 s_3 - I_{xx_3} c_3^2 s_2 +$

$$I_{yy3}c_3^2 s_2 + c_3^2 m_{l_3} p_{x_3}^2 s_2 - c_3^2 m_{l_3} p_{y_3}^2 s_2 - 2I_{xy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{x_3} - c_2 c_3 m_{l_3} p_{x_3} p_{z_3} + c_2 d_2 m_{l_3} p_{y_3} s_3 + c_2 m_{l_3} p_{y_3} p_{z_3} s_3 - 2c_3 m_{l_3} p_{x_3} p_{y_3} s_2 s_3) - \dot{q}_2((m_{l_3} sin(2q_3) p_{x_3}^2)/2 + m_{l_3} cos(2q_3) p_{x_3} p_{y_3} - (m_{l_3} sin(2q_3) p_{y_3}^2)/2 + I_{xy_3} cos(2q_3) - (I_{xx_3} sin(2q_3))/2 + I_{yy_3} sin(2q_3))/2) - \dot{q}_3(I_{xz_3} c_3 - I_{yz_3} s_3 + m_{l_3}(d_2 + p_{z_3})(c_3 p_{x_3} - p_{y_3} s_3))$$

$$C_{(3,1)} = \dot{q}_1(I_{xy_3} - I_{xy_3} c_2^2 - 2I_{xy_3} c_3^2 + 2I_{xy_3} c_2^2 c_3^2 + I_{xz_3} c_3 s_3 - I_{yy_3} c_3 s_3 - m_{l_3} p_{x_3} p_{y_3} - I_{xx_3} c_2^2 c_3 s_3 + I_{yy_3} c_2^2 c_3 s_3 - c_2^2 m_{l_3} p_{x_3} p_{y_3} - 2c_3^2 m_{l_3} p_{x_3} p_{y_3} - c_3 m_{l_3} p_{x_3}^2 s_3 + c_3 m_{l_3} p_{y_3}^2 s_3 + I_{yz_3} c_2 c_3 s_2 + I_{xz_3} c_2 s_2 s_3 + 2c_2^2 c_3^2 m_{l_3} p_{x_3} p_{y_3} + c_2^2 c_3 m_{l_3} p_{x_3}^2 s_3 - c_2^2 c_3 m_{l_3} p_{y_3}^2 s_3 + c_2 c_3 d_2 m_{l_3} p_{y_3} s_2 - c_2 c_3 m_{l_3} p_{y_3} p_{z_3} s_2 + c_2 d_2 m_{l_3} p_{z_3} s_2 s_3 + c_2 m_{l_3} p_{x_3} p_{z_3} s_2 s_3) - \dot{q}_2((I_{xz_3} s_2)/2 - (I_{yy_3} s_2)/2 + (I_{zz_3} s_2)/2 + m_{l_3} p_{y_3}^2 s_2 - I_{xz_3} c_2 c_3 + I_{yz_3} c_2 s_3 - I_{xz_3} c_3^2 s_2 + I_{yy_3} c_3^2 s_2 + c_3^2 m_{l_3} p_{x_3}^2 s_2 - c_3^2 m_{l_3} p_{y_3}^2 s_2 - 2I_{xy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{x_3} - c_2 c_3 m_{l_3} p_{x_3} p_{z_3} + c_2 d_2 m_{l_3} p_{y_3} s_3 + c_2 m_{l_3} p_{y_3} p_{z_3} s_3 - 2c_3 m_{l_3} p_{x_3} p_{y_3} s_2 s_3)$$

$$C_{(3,2)} = \dot{q}_2((m_{l_3} sin(2q_3) p_{x_3}^2)/2 + m_{l_3} cos(2q_3) p_{x_3} p_{y_3} - (m_{l_3} sin(2q_3) p_{y_3}^2)/2 - I_{xy_3} cos(2q_3) - (I_{xx_3} sin(2q_3))/2 + (I_{yy_3} sin(2q_3))/2) - \dot{q}_1((I_{xz_3} s_2)/2 - (I_{yz_3} s_2)/2 + (I_{zz_3} s_2)/2 - m_{l_3} p_{y_3}^2 s_2 - I_{xz_3} c_2 c_3 + I_{yz_3} c_2 s_3 - I_{xz_3} c_3^2 s_2 + I_{yy_3} c_3^2 s_2 + c_3^2 m_{l_3} p_{x_3}^2 s_2 - c_3^2 m_{l_3} p_{y_3}^2 s_2 - 2I_{xy_3} c_3 s_2 s_3 - c_2 c_3 d_2 m_{l_3} p_{x_3} - c_2 c_3 m_{l_3} p_{x_3} p_{z_3} - c_2 d_2 m_{l_3} p_{y_3} s_3 + c_2 m_{l_3} p_{y_3} p_{z_3} s_3 - 2c_3 m_{l_3} p_{x_3} p_{y_3} s_2 s_3)$$

$$C_{(3,3)} = 0$$

4.3.7. Vector de pares gravitacionales $g(q)$

El vector de gravedad es el gradiente de la energía potencial $\mathcal{U}(q)$ descrita en las ecuaciones (4.36) y (4.37), para las configuraciones de posición vertical y horizontal de la muñeca, respectivamente. El gradiente de $\mathcal{U}(q)$ se obtiene con la ecuación (4.23), teniendo como resultado para la configuración de posición vertical de la muñeca:

$$g(q) = \begin{bmatrix} 0 \\ m_{l_3}(c_2 p_{y_3} s_3 - d_2 s_2 - p_{z_3} s_2 - c_2 c_3 p_{x_3}) g - m_{l_2}(p_{x_2} s_2 + c_2 p_{x_2}) g \\ m_{l_3}(p_{x_3} s_2 s_3 + p_{y_3} s_2 c_3) g \end{bmatrix}$$

y para la configuración de posición horizontal de la muñeca:

$$g(\boldsymbol{q}) = \begin{bmatrix} g_1(q) \\ g_2(q) \\ g_3(q) \end{bmatrix}$$

donde los elementos $g_1(q)$, $g_2(q)$ y $g_3(q)$ son:

$g_1(q) = gm_{l_2}(p_{x_2}c_1c_2 - p_{y_2}s_1 + p_{z_2}c_1s_2) - gm_{l_3}(p_{x_3}(s_1s_3 - c_1c_2c_3) + p_{y_3}(c_3s_1 + c_1c_2s_3) - d_2c_1$
$s_2 - p_{z_3}c_1s_2) + gm_{l_1}(p_{x_1}c_1 - p_{y_1}s_1)$

$g_2(q) = gs_1(d_2m_{l_3}c_2 + m_{l_2}p_{z_2}c_2 + m_{l_3}p_{z_3}c_2 - m_{l_2}p_{z_2}s_2 + m_{l_3}p_{y_3}s_2s_3 - m_{l_3}p_{x_3}c_3s_2)$

$g_3(q) = gm_{l_3}p_{x_3}(c_1c_3 - c_2s_1s_3) - gm_{l_3}p_{y_3}(c_1s_3 + c_2c_3s_1)$

Capítulo 5

El protocolo CAN 2.0b

El protocolo CAN (acrónimo de *Controller Area Network*) es un concepto de bus serie multi-maestro desarrollado por la firma alemana Robert Bosch GmbH, en 1986, para el enlace entre controladores, actuadores y sensores, y provisto de una elevada velocidad de transmisión combinada con una importante inmunidad frente a interferencias electromagnéticas. Nace como respuesta a la necesidad de disponer de una comunicación fiable, eficaz y rápida para aplicaciones de la industria automotriz. Por sus notables características técnicas, eficiencia y soporte, la ISO (acrónimo de *International Standard Organization*) adoptó como estándares dos versiones de CAN: la versión para aplicaciones de alta velocidad (hasta 1 Mbps) estandarizada como ISO 11898 y la versión para aplicaciones de baja velocidad (hasta 125 kbps) estandarizada como ISO 11519-2 [Domingo et al., 2003]. En este capítulo se describen las principales especificaciones del protocolo CAN, el tipo de cable y conector que se utiliza, así como el proceso general de transmisión y recepción de datos, y el proceso particular de comunicación aplicado a la muñeca MASKARA. Para mayor información sobre el protocolo CAN puede consultarse [Di Natale et al., 2012].

5.1. Especificaciones

El protocolo CAN ha sido creado para proveer comunicación determinística en complejos sistemas distribuidos con las siguientes características y capacidades:

- Asignación de mensajes por prioridad y garantía de máximas latencias.
- Comunicación multicast (o multidifusión) con sincronización basada en bits.
- Consistencia de datos en todo el sistema.
- Acceso multi-maestro al bus.
- Detección y señalización de errores con retransmisión automática de mensajes corruptos.
- Detección de fallas permanentes en nodos, y desconexión automática para aislar nodo averiado.

Los dispositivos conectados al bus (también conocidos como nodos) requieren de un circuito integrado controlador CAN y un transceptor (transmisor y receptor) que usualmente se encuentran integrados entre los componentes electrónicos de la tarjeta de comunicación. A continuación se describen las principales especificaciones de la arquitectura del protocolo de comunicación CAN, las cuales son:

- El circuito integrado *controlador CAN* es el componente (*hardware*) que es responsable del acceso al medio físico de transmisión. Proporciona registros para la configuración de la conexión al bus con características definidas, incluyendo la selección de la velocidad de transferencia de bits, el tiempo de muestreo por bit, la longitud del espacio binario del marco intermedio. El chip controlador CAN también proporciona la funcionalidad de supervisar todos los aspectos del protocolo CAN, incluyendo la operación de los modos de transmisión y el manejo del bus en estado inactivo. El chip controlador CAN ofrece un número de registros de datos para guardar mensajes

de transmisiones salientes y datos de entrada, y proporciona soporte para filtrado y enmascaramiento de mensajes en la recepción.

- El *transceptor CAN* es la interfaz entre el chip controlador CAN y el medio de transmisión (en la mayoría de los casos un bus de dos líneas en modo diferencial) que consiste básicamente de un amplificador de transmisión y de recepción. El transceptor convierte el nivel eléctrico binario del chip controlador CAN al nivel eléctrico del medio de transmisión o bus. Como un transmisor, éste proporciona la salida de corriente suficiente para manejar los niveles eléctricos del bus, y protege el chip controlador CAN contra sobrecargas. Como un receptor, éste proporciona el nivel de la señal recesiva y protege la entrada comparadora del chip controlador CAN contra voltajes excesivos sobre las líneas del bus. Además, ayuda a detectar errores del bus tales como ruptura de línea, cortocircuitos y derivaciones a tierra. Finalmente, opera como un aislamiento galvánico entre un nodo CAN y las líneas del bus.

- La *topología de red* se refiere al arreglo de cables, nodos CAN y repetidores en la red. El protocolo CAN se basa principalmente en una topología bus (o lineal) que puede consistir de dos líneas de transmisión en modo diferencial en una red de alta velocidad o de solamente una línea de transmisión en una red de baja velocidad. En este último caso el uso de una topología bus abierta (o ramificada) es posible siempre y cuando la red de baja velocidad sea de corta longitud y cuente también con una resistencia de terminación de 180 ohms en cada ramificación derivada del bus. Cabe mencionar que el número máximo de nodos en una topología lineal CAN de alta velocidad está limitado por la carga eléctrica del bus, pudiendo alcanzar hasta 30 nodos por segmento o hasta 200 nodos con la ayuda de repetidores de bus.

- La *longitud del bus* es de especial interés para alcanzar la máxima velocidad de transmisión. El retardo de la señal de propagación a considerar en el cálculo de la máxima longitud de bus permitida incluye varios escenarios, con retardos variados, dependiendo de la calidad de los siguientes componentes seleccionados: el chip controlador CAN (50-62 ns), el optoacoplador (40-140 ns), el transceptor (120-250 ns)

y el cable (cerca de 5 ns/m). La tabla 5.1 muestra una referencia (no obligatoria) de las velocidades de transmisión y sus longitudes de bus máximas permitidas.

Tabla 5.1: Velocidades de transmisión típicas y sus correspondientes longitudes de bus

Velocidad de transmisión	Tiempo por bit (μs)	Longitud del bus (m)
1 Mb/s	1	30
800 kb/s	1.25	50
500 kb/s	2	100
250 kb/s	4	250
125 kb/s	8	500
62.5 kb/s	16	1000
20 kb/s	50	2500
10 kb/s	100	5000

- La *terminación de bus* es una medida preventiva para evitar que las señales eléctricas transmitidas a lo largo del bus sean reflejadas (al origen) en los extremos de las líneas de transmisión. La terminación de bus también ayuda a evitar errores cuando un nodo lee el estado eléctrico (activo o inactivo) del bus. Terminando las líneas de transmisión con una resistencia de terminación en ambos extremos del bus y evitando largas e innecesarias ramificaciones del bus es la mejor acción preventiva. El método de terminación de bus varía dependiendo de la velocidad de la red (velocidad alta o velocidad baja). Para una red CAN de alta velocidad, ambos extremos del par de cables de señales (CAN_H y CAN_L) deben ser terminados. Las resistencias de terminación en el cable deben coincidir con la impedancia nominal del cable. El estándar ISO 11898 requiere un cable con una impedancia nominal de 120 ohms, y por lo tanto unas resistencias de 120 ohms deben ser usadas para terminación del cable. Si multiples dispositivos están conectados a lo largo del cable, sólo los dispositivos en los extremos del cable necesitan resistencias de terminación. La figura 5.1 muestra un ejemplo de como terminar una red de alta velocidad. A diferencia de la red CAN de alta velocidad, la red CAN de baja velocidad (estándar ISO 115_9-2) requiere terminación en el transceptor en lugar de en el cable. La figura 5.2 muestra el lugar donde las resistencias de terminación deben ser activadas (en el transceptor) en una

red CAN con una sola línea de transmisión de baja velocidad. Por ejemplo, la tarjeta CAN de baja velocidad (para una sola línea de transmisión) de la compañía National Instruments incluye una resistencia integrada de 9.09 KΩ para el transceptor.

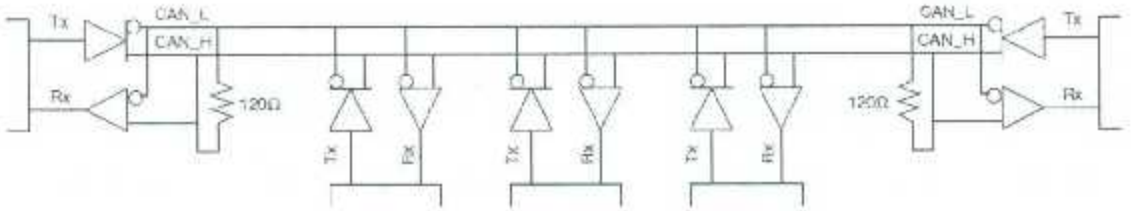


Figura 5.1: Terminación del bus en una red de alta velocidad

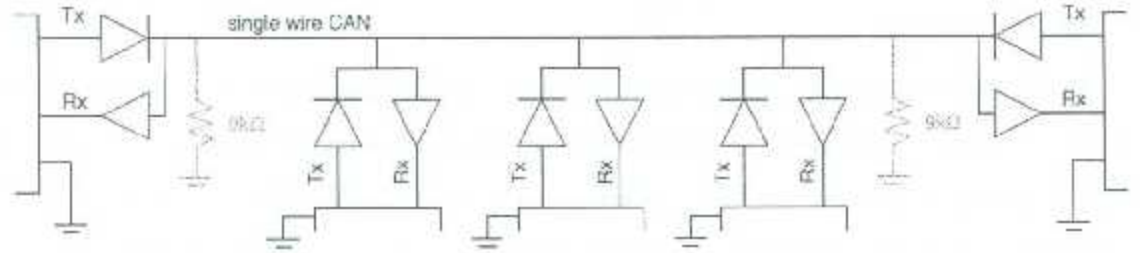


Figura 5.2: Terminación del bus en una red de baja velocidad

5.2. Cable y conector

El cable más común y utilizado como medio físico de transmisión de las señales del protocolo CAN es el definido por el estándar ISO 11898-2, un cable de dos líneas en modo balanceado. Por otro lado, el estándar SAE J2411 (SAE es el acrónimo para Society of Automotive Engineers) define una sola línea de transmisión (más tierra, por supuesto) como medio físico de transmisión para redes CAN de baja velocidad. También existe el uso de fibra óptica como medio físico de transmisión, aunque éste no se encuentre regulado por algún estándar. Cada tipo de cable tiene sus diferentes aplicaciones, como por ejemplo el uso de dos líneas de transmisión en modo balanceado es más utilizado en redes de alta

velocidad (hasta 1 Mb/s) y el uso de una sola línea (más tierra) se utiliza en redes de baja velocidad (hasta 125 kb/s), así como el uso de fibra óptica es recomendable para ambientes con alta perturbación electromagnética. La tabla 5.2 muestra posibles tipos y secciones transversales de cable para las configuraciones de red más utilizadas.

Tabla 5.2: Características del cable del bus

Velocidad del bus	Tipo de cable	Resistencia/m	Terminación	Longitud del bus
50 kb/s a 1,000 m	0.75-0.8 mm^2 (AWG18)	70 mΩ	150-300 Ω	600-1.000 m
100 kb/s a 500 m	0.5-0.6 mm^2 (AWG20)	<60 mΩ	150-300 Ω	300-600 m
500 kb/s a 100 m	0.34-0.6 mm^2 (AWG22, AWC20)	<40 mΩ	127 Ω	40-300 m
1000 kb/s a 40 m	0.25-0.34 mm^2 (AWG23, AWG22)	<26 mΩ	124 Ω	0-40 m

Los tipos de conectores a utilizar para las líneas de transmisión no están definidos por el estándar original desarrollado por Robert Bosch CmbH [Bosch, 1991] pero si fueron establecidos por otros estándares. La figura 5.3 muestra la distribución de pines de un conector blindado muy popular de 9 pines establecido por el estándar CIA DS 102-1.

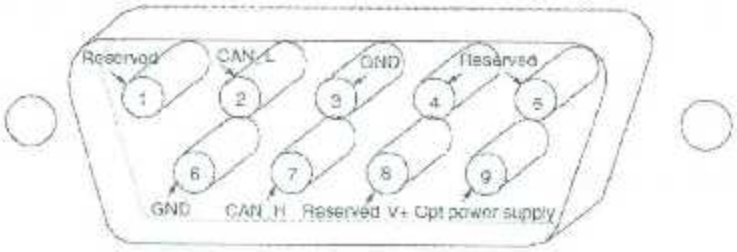


Figura 5.3: Conector para CAN definido por CIA (DS 102-1)

5.3. Capa de datos

Los bits que forman las tramas de mensajes CAN se generan de acuerdo a un método de codificación denominado NRZ (acrónimo de *No Return to Zero*) con un bit de relleno (*bit stuffing*). La codificación NRZ simplemente mantiene un determinado estado dominante

(nivel lógico '0') o recesivo (nivel lógico '1') durante todo el tiempo de bit, a diferencia de una codificación por modulación de anchura de pulso, donde el nivel del bit siempre volverá a cero en algún momento del tiempo de bit.

La codificación NRZ presenta un problema potencial de sincronización. El nodo receptor será incapaz de decodificar toda aquella información que incluya más de 10 bits seguidos en el mismo estado. Para resolver este problema, las especificaciones CAN indican que el nodo transmisor debe generar un bit de relleno (*bit stuffing*) si se transmiten más de 5 bits con el mismo estado lógico; el estado de este bit será el opuesto al del grupo de cinco. Este hecho podría provocar un conflicto de sincronización, sin embargo, el nodo receptor elimina este bit extra para que no tenga efecto en el mensaje de datos resultante de la transmisión. El bit de relleno se aplica a la mayoría de campos de las tramas de datos y tramas remotas. El bit de relleno también sirve como método de detección de errores en el bus. Si más de cinco bits del mismo valor son recibidos, se produce la violación de la regla del *bit stuffing* y el error es descubierto inmediatamente.

5.3.1. Formatos de la trama del mensaje

En el protocolo CAN, existen cuatro diferentes tipos de tramas, definidas de acuerdo a su contenido y función:

- La *trama de datos* contiene datos de información de una fuente a (posiblemente) multiples receptores.
- La *trama remota* es usada para solicitar el envío de una correspondiente trama de datos.
- La *trama de error* es transmitida cada vez que un nodo en la red detecta un error.
- La *trama de carga excesiva* es usada en el control de flujo para solicitar un tiempo adicional de espera antes del envió de la trama de datos o remota.

Trama de datos

Las tramas de datos son usadas para transmitir información entre un nodo emisor y uno o más nodos receptores. Las tramas de datos no usan explícitamente direcciones para identificar los receptores del mensaje. En su lugar, cada nodo receptor establece los mensajes que serán recibidos según el contenido de la información, la cual está codificada en el campo identificador de la trama. Existen dos formatos de tramas de datos definidos en la especificación CAN 2.0: la normal (CAN 2.0a) y la extendida (CAN 2.0b). La trama normal consta de 11 bits para la identificación del mensaje (ID) y la trama extendida de 29 bits. La figura 5.4 muestra el formato de la trama de datos que consta de un conjunto de campos que contienen información de distinta naturaleza.

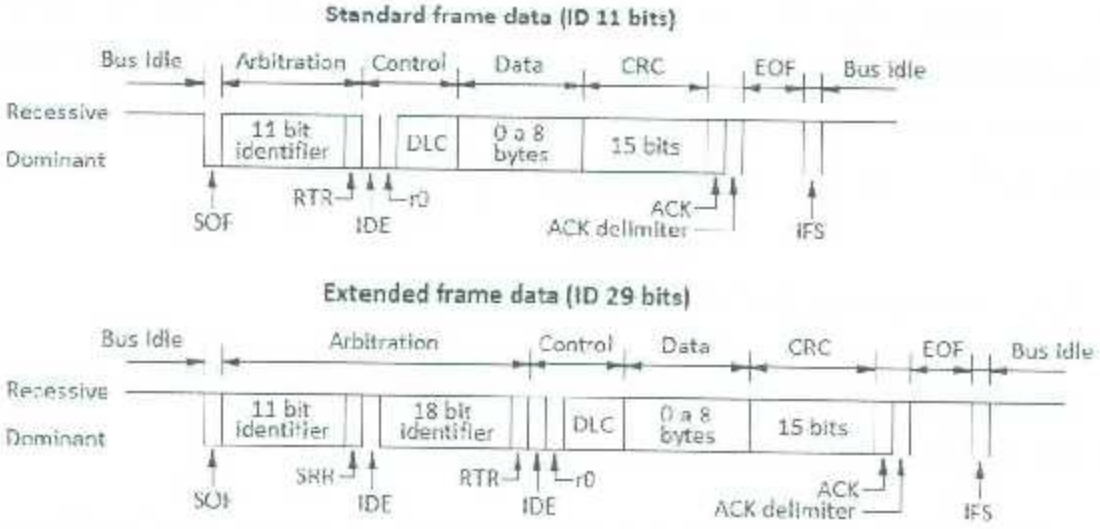


Figura 5.4: Formato de la trama de datos de CAN

Así pues se tiene:

- **Inicio de trama (SOF, *start of frame*).** Es un bit de estado dominante que marca el inicio de envío de un mensaje cuando el bus está libre, o después de finalizar la transmisión de una trama.
- **Campo de arbitraje (o identificador).** Utilizado para identificar el mensaje y para determinar la prioridad cuando dos mensajes colisionan al acceder al bus.

Incluye el bit de demanda de transmisión RTR (acrónimo de *Remote Transmission Request*) que, para tramas de datos, siempre será dominante. En la trama extendida, el bit sustituto de demanda remota (SRR) y el bit de extensión del identificador (IDE) separa en dos partes (11 bits y 18 bits) los 29 bits del identificador.

- **Campo de control.** Incluye cuatro bits que sirven para marcar la longitud del campo de datos (DLC, *data length code*). Además, incluye dos bits reservados para futuras ampliaciones que han de ser transmitidos como estados dominantes (IDE y r0 en la trama estándar, y r1 y r0 en la trama extendida).

- **Campo de datos.** Siempre será una longitud entera de entre 0 y 8 bytes, de acuerdo con lo que indique el DLC. El MSB es el byte que primero se envía.

- **Campo de código de redundancia cíclica (CRC).** Incluye la secuencia CRC de la trama y un bit final dominante como delimitador. Este código es generado y decodificado por *hardware*, constituyendo un método optimizado de verificación de errores para flujos de menos de 127 bits. El polinomio generador que se utiliza es el $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$.

- **Campo de reconocimiento (ACK, *acknowledge*).** En este campo el nodo transmisor envía dos bits en estado recesivo ('1' lógico): un bit de reconocimiento (ACK) y un bit delimitador de ACK. El nodo o nodos receptores fuerzan el bit ACK a un estado dominante si el mensaje se ha recibido adecuadamente.

- **Campo de fin de trama (EOF, *end of frame*).** La trama finaliza con el envío de siete bits recesivos consecutivos.

- **Campo entre tramas (IFS, *interframe space*).** El espacio entre tramas separa una trama de la siguiente y ha de constar de al menos tres bits recesivos.

Trama remota

Una trama remota (*remote frame*) se utiliza para solicitar la transmisión de un mensaje

(datos) con un determinado identificador de un nodo remoto. Una trama remota tiene el mismo formato de una trama de datos con las siguientes características:

- El campo del identificador se usa para indicar el identificador del mensaje solicitado.
- El campo de datos está siempre vacío (0 bytes).
- El campo DLC indica la longitud de datos del mensaje solicitado (no del mensaje transmitido).
- El bit RTR dentro del campo de arbitraje está siempre puesto en estado recesivo.

El hecho de que el bit RTR sea recesivo en esta trama y dominante en la trama de datos provoca que si en cualquier momento un nodo A solicita un dato de un nodo B, justamente en el momento en que el nodo B intenta transmitir un dato por el bus, el nodo B tendrá prioridad en el envío. De esta manera, se permite al nodo B deshacerse primero del dato que tiene, antes de generar otro.

Trama de error

Cualquier nodo que detecte un error en el bus puede generar una trama de error. La trama de error está formada por dos campos: el primero es un conjunto de 6 a 12 bits del error (*error flag field*) y, el segundo, un campo delimitador de error (*error delimiter field*) formado por ocho bits recesivos. El campo delimitador de error permite que los nodos del bus puedan reiniciar limpiamente la comunicación después de un error. El campo de bits de error puede, sin embargo, adoptar dos formas dependiendo del estado del nodo que detecta el error:

1. **Error activo**

   Si un nodo en estado de *error activo* detecta un error en el bus, interrumpe la transmisión del mensaje actual generando una secuencia de bits de error compuesta por seis bits dominantes consecutivos. Esta secuencia de bits activos viola la regla

de *bit stuffing* y las demás estaciones del bus generan, a su vez, la correspondiente trama de error. El campo de bits de error presentará, por consiguiente, entre 6 y 12 bits dominantes consecutivos (generados por uno o más nodos). La trama de error se completa, finalmente, con el campo delimitador de error. Después de completada la trama de error, la actividad del bus vuelve a la normalidad y el nodo interrumpido reenvía el mensaje abortado.

2. **Error pasivo**

   Si un nodo en estado de *error pasivo* detecta un error en el bus, transmite una secuencia de bits de error compuesta por 6 bits recesivos consecutivos y los 8 bits del campo delimitador de error (14 bits recesivos en total). En esta situación, el envío de una trama de error pasivo no afectará a ningún otro nodo de la red. Esta manera de operar se explica porque no es deseable, cuando se produce un fallo permanente de *hardware* en uno de los nodos receptores, que éste bloquee el bus enviando tramas de error activo continuamente. Para llevar a cabo esta operación, la especificación CAN 2.0b prevé un contador de errores. El nodo comienza funcionando en modo de *error activo* y conmuta internamente a modo de *error pasivo* cuando se ha alcanzado el límite de transmisiones previstas por el contador (128).

Esta manera de manejar los errores puede parecer laboriosa y excesiva, sin embargo, este procedimiento asegura el éxito de la comunicación de datos; característica importante de sistemas de control en tiempo real en los que se da con frecuencia el intercambio síncrono de datos. Cuando la trama de datos original es destruida por la superposición de alguna trama de error, el nodo que transmitió el mensaje original retransmitirá nuevamente los datos a todos los nodos del bus con un intervalo máximo equivalente a 29 bits.

Trama de carga excesiva

La trama de carga excesiva tiene el mismo formato que la trama de error activo. Sin embargo, se diferencia de ésta en que sólo puede generarse durante el espacio entre tramas

y no durante la transmisión de un mensaje. La trama de carga excesiva consiste en dos campos: un primer campo formado por los bits de carga excesiva (*overload flag*), seguido de un campo delimitador de carga excesiva (*overload delimiter*) consistente en ocho bits recesivos. El campo de carga excesiva está constituido por seis bits dominantes, seguidos de los bits de carga excesiva generados por otros nodos (resultando un máximo de 12 bits dominantes). Un nodo puede generar una trama de carga excesiva debido a la imposibilidad de comenzar la recepción del siguiente mensaje, pudiendo retrasarlo mediante la generación secuencial de, como máximo, dos tramas de carga excesiva. Si se emplea un buffer de recepción de suficiente longitud esta trama es difícil que se produzca.

5.3.2. Proceso de comunicación

El protocolo CAN está basado en el *mensaje* en lugar de en la *dirección*, es decir, se trata de un protocolo *cliente-servidor* en el que los dispositivos del bus serie y los mensajes intercambiados no están referenciados explícitamente por las direcciones de los distintos nodos. Por esta razón, en el identificador (ID) del mensaje se acostumbra definir el contenido y prioridad de los datos (dentro del mensaje) en lugar de especificar una dirección de referencia.

Este simple concepto aporta enormes ventajas en aplicaciones de automatización, puesto que un nodo puede transmitir una única información dirigida al resto de nodos conectados al bus (*comunicación broadcast*).

Todos los dispositivos del bus que necesiten esta información recibirán y harán uso del mensaje al mismo tiempo. Si sólo un dispositivo del bus tiene acceso directo a un sensor específico, los demás elementos del bus podrán recibir simultáneamente el valor actualizado en el momento en el que aquél transmita el mensaje.

Recíprocamente, si un dispositivo del bus necesita de cierta información, puede enviar la correspondiente demanda transmitiendo un mensaje de identificación y un bit especial de demanda de transmisión remota (bit RTR). Cualquier nodo del bus que posea esa

información podrá responder a la demanda (*comunicación multicast*).

Este método de comunicación, basado en las características que conforman el mensaje, permitirá fácilmente la integración de un nuevo nodo receptor de información en el sistema CAN. En estos casos, no hay ninguna necesidad de configurar los otros nodos del bus. Simplemente se habrá de conectar físicamente el dispositivo al bus, y a partir de este momento podrá recibir los mensajes que, con los ID apropiados, contienen la información deseada.

En el protocolo CAN, todos los nodos tienen la misma oportunidad para empezar a transmitir una trama del mensaje en el momento que el bus esté inactivo. Este hecho podría representar un problema cuando dos nodos descubren el bus inactivo (detección de portadora o detección de medio disponible) y proceden a enviar sus respectivos mensajes en el mismo instante (acceso multiple), ya que se produce una colisión de datos y una alteración de los mensajes que se intentan transmitir. Sin embargo, el protocolo CAN soporta la detección no destructiva de colisiones y el arbitraje. En el caso de que dos dispositivos del bus comiencen la transmisión al mismo tiempo, el protocolo CAN puede descubrir y resolver la colisión de datos resultante en el bus basándose en la prioridad del mensaje. Por esta razón el protocolo CAN es del tipo CSMA/CD+AMP (acrónimo de *Carrier Sense Multiple Access/Collision Detect and Arbitration on Message Priority* o, en español, acceso multiple por detección de portadora y detección de colisiones basado en la prioridad del mensaje).

Dos características son esenciales para la detección no destructiva de colisiones y el arbitraje:

1. Los estados lógicos definidos como: *recesivo* ('1' lógico) y *dominante* ('0' lógico).
2. El nodo que transmite debe supervisar el bus para determinar si los bits que envía están realmente presentes en el bus.

Los términos recesivo y dominante indican qué estado ganará en caso de una colisión en el bus; un estado dominante siempre anulará un estado recesivo.

La detección de colisiones y el arbitraje se podrá realizar si desde cada nodo se supervisa constantemente el estado del bus. Si un nodo transmite un estado recesivo y en ese instante detecta un estado dominante, es señal de que se ha producido una colisión en el bus.

El nodo que detecta la colisión debe cesar la transmisión inmediatamente y debe permitir el control del bus al nodo que transmitió el bit dominante. Si se define un nivel lógico '0' para el estado dominante, el nodo que transmita el número más bajo (mensaje ID de mayor prioridad) ganará el arbitraje del bus; en la figura 5.5 el nodo A gana el bus con el ID del mensaje 0xF0.

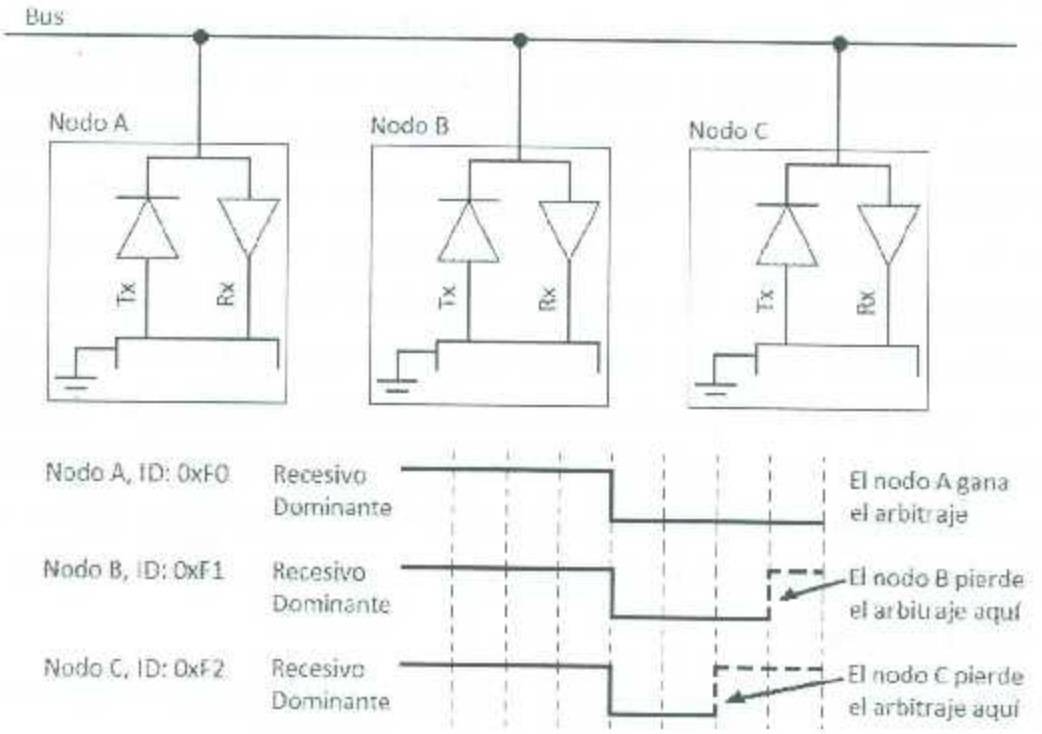


Figura 5.5: Arbitraje por estado del bit recesivo/dominante

5.4. Comunicación CAN 2.0a para la MASKARA

La topología de CAN que emplean los módulos PowerCube se conoce como bus o lineal. Cada módulo se conecta al bus por medio de las terminales BUS-H y BUS-L que

se encuentran en el bloque de conexiones.

Como ya se ha explicado, la muñeca esférica MASKARA emplea una tarjeta de comunicación CAN de la compañía Softing Industrial Automation. El paquete Matlab de MathWorks, en el toolbox xPC Target incluye una biblioteca de bloques para esta tarjeta con diferentes funciones tales como enviar y recibir mensajes CAN personalizados así como para supervisar el estado de la red [MathWorks, 2002].

El enlace entre la tarjeta de comunicación CAN de la computadora de control y las UICs de los módulos PowerCube de la muñeca esférica se realiza a través de un cable de dos hilos blindado, tal y como lo especifica el estandar ISO 11898-2. Además, se establece una relación cliente-servidor entre la computadora de control (cliente) y los módulos PowerCube (servidores) respectivamente; como es propio del protocolo CAN de la tarjeta de comunicación utilizada. Se emplea la velocidad de transmisión máxima para la red CAN de alta velocidad que es de 1 Mb/s; los números de identificación de cada uno de los nodos correspondientes a los módulos PowerCube que conforman los ejes 1, 2 y 3 del robot, son 11, 13 y 14 (0x0B, 0x0D y 0x0E en su representación hexadecimal), respectivamente. Estos números de identificación no son utilizados explícitamente en el protocolo CAN, debido a que el método de comunicación utilizado por este protocolo está basado en la prioridad del mensaje y no en la dirección o número de identificación del nodo. En [Schunk, 2012] se explica que cada mensaje CAN enviado por la computadora de control (el cliente) puede tener en total entre 5 (mínimo) y 11 bytes (máximo), y que además, debe seguir un formato como el que se muestra en el esquema de la figura 5.6.

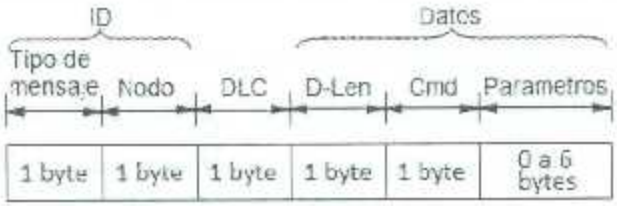


Figura 5.6: Contenido de un mensaje CAN

Los primeros dos bytes del mensaje se utilizan para especificar el identificador (ID) del mensaje en el formato de trama de datos estándar (o normal) de la especificación CAN

2.0a; lo que significa que se emplearán únicamente 11 bits de los primeros 2 bytes. El byte más significativo del mensaje proporciona tres bits (los menos significativos) para codificar en hexadecimal el contenido del mensaje en tres formas:

- 0x03 para mensajes con información de errores provenientes de los módulos del robot.
- 0x05 para mensajes provenientes de la computadora de control.
- 0x07 para mensajes provenientes de los módulos del robot.

El siguiente byte es usado para codificar en hexadecimal el número de nodo correspondiente al módulo PowerCube del robot, los cuales pueden ser 11, 13 o 14 (0x0B, 0x0D y 0x0E). El byte del mensaje identificado como DLC define el número de bytes de datos subsecuentes (máximo 8) que contiene el mensaje CAN. El primer byte de datos (D-Len) define el número de bytes subsecuentes (máximo 7) de datos válidos del cliente o servidor. El segundo byte de datos (Cmd) define el tipo de comando de control (consigna de posición, velocidad, corriente, etc.) a ejecutar por el módulo PowerCube correspondiente del robot. Los bytes de datos restantes (máximo 6) definen parámetros de información correspondientes a un comando de control. Para el caso de la muñeca MASKARA, estos parámetros de información pueden contener consignas de posición, velocidad, o corriente, solicitud de estados de posición y corriente de los módulos del robot, acuse de mensaje recibido por un nodo receptor, e información de códigos de error presentes en los módulos PowerCube. El acuse de mensaje corresponde a un mensaje de notificación que envía un nodo receptor a un nodo emisor para confirmar la recepción del mensaje.

La computadora de control de la muñeca a través del algoritmo de control, envía periódicamente (cada 2.5 ms) siete mensajes CAN para especificar la consigna deseada de control (un mensaje por cada servo), solicitar los estados actuales de posición y corriente (un mensaje por cada servo), y solicitar el estado actual de la entrada digital X3/IN 0 (un mensaje para el primer módulo, según la tabla 3.4) donde se encuentra instalado el paro de emergencia del robot. En la tabla 5.3 se enlistan estos mensajes en el orden correspondiente, su contenido y el número de bytes que requiere cada uno.

Tabla 5.3: Mensajes de la computadora de control a los módulos PowerCube

No. Mensaje	Contenido	No. bytes
1	Consigna descada de servomotor 1	11
2	Consigna deseada de servomotor 2	11
3	Consigna deseada de servomotor 3	11
4	Solicitud de estados (posición y corriente) de servomotor 1	11
5	Solicitud de estados (posición y corriente) de servomotor 2	11
6	Solicitud de estados (posición y corriente) de servomotor 3	11
7	Solicitud de estado de entrada digital X3/IN 0 de servomotor 1	11
	Total de datos	77

El número de bytes de cada mensaje de la tabla 5.3 se definió como el máximo permitido por el protocolo CAN de los módulos PowerCube; esto con el fin de facilitar la programación del algoritmo de control encargado de enviar estos mensajes desde la computadora de control, aunque es posible disminuir el número de bytes por cada mensaje conociendo la variación de los bytes de datos por mensaje. El contenido de los bytes de datos de los mensajes de la tabla 5.3 varía dependiendo del comando de control (CMD) y sus parámetros, es decir, para el caso de una consigna de control deseada de velocidad o corriente (mensajes 1, 2 y 3), el valor hexadecimal del CMD es 0xB5 o 0xB3, respectivamente, y para especificar los bytes de parámetros de la consigna deseada se emplean solamente cuatro bytes (D-Len es igual a 5). Para el caso de la solicitud de los estados de posición y corriente del servo (mensajes 4, 5 y 6), el valor hexadecimal del CMD es 0x95 y para especificar los bytes de parámetros se emplean solamente cinco bytes (D-Len es igual a 6). En el caso de la solicitud de estado de la entrada digital X3/IN 0 del servomotor 1 (mensaje 7), el valor hexadecimal del CMD es 0xE1 y para especificar los bytes de parámetros no se emplea ningún byte (D-Len igual a 1). Cada uno de los mensajes especificados en la tabla 5.3 genera uno o varios mensajes de respuesta (como es el caso de los mensajes 4, 5 y 6) por parte de los módulos del robot. Toda esta información se presenta en [Schunk, 2012], en esta sección sólo se presenta un breve resumen.

La computadora de control recibe periódicamente de 10 a 13 mensajes CAN provenientes de la muñeca para confirmar las consignas deseadas de control (un mensaje por cada servo), proporcionar los estados actuales de posición y corriente (dos mensajes por cada servo), proporcionar el estado actual de la entrada digital X3/IN 0 (un mensaje proveniente del primer módulo) donde se encuentra instalado el paro de emergencia de la muñeca y proporcionar la información eventual de códigos de error presentes en la muñeca (un mensaje eventual por cada servo). La información eventual de códigos de error presentes en la muñeca corresponde a mensajes aleatorios que pueden aparecer en caso de algún imprevisto en la operación de la muñeca (paro de emergencia, caída de voltaje en fuente de alimentación de los servos, servomotores obstruidos, etc.); por lo que es necesario considerar la longitud (número de bytes) de dichos mensajes en el tiempo de comunicación o periodo de muestreo del algoritmo de control de la computadora de control. En la tabla 5.4 se enlistan estos mensajes en el orden correspondiente que se recibe, su contenido y el número de bytes que requiere cada uno.

La secuencia de los mensajes entre la computadora de control y las UICs, inicia con un primer ciclo para asegurar que no existan errores presentes en los módulos y que la muñeca se encuentre en la posición de casa ("Home"). Después de ejecutar el primer ciclo y revisar que no esté activado el paro de emergencia (botón físico) o el paro de movimiento de la muñeca (botón por software, véase sección 6.3), se entra en el ciclo central que se repite por tiempo indefinido (hasta no oprimirse el botón por software de paro de movimiento o el botón de paro de emergencia) donde se envían las consignas de control a los módulos y se hace la solicitud de los estados de posición, corriente y de una entrada digital de la muñeca correspondiente al paro de emergencia; es también en este ciclo donde se recibe la confirmación de las consignas de control enviadas y los estados del robot solicitados (véase tabla 5.3), además de revisar que no exista perdida de datos en los mensajes recibidos. El ciclo central finaliza de forma normal en un último ciclo al momento de oprimirse el botón por software de paro de movimiento del robot. La figura 5.7 ilustra la secuencia de los comandos de control de estas tres etapas para operar de forma normal la muñeca.

Tabla 5.4: Mensajes de los módulos PowerCube a la computadora de control

No. Mensaje	Contenido	No. bytes
1	Confirmación de consigna deseada de servomotor 1	7
2	Confirmación de consigna deseada de servomotor 2	7
3	Confirmación de consigna deseada de servomotor 3	7
4	Desplazamiento angular y corriente actual de servomotor 1	11
5	Desplazamiento angular y corriente actual de servomotor 1 (continuación de mensaje 4)	10
6	Desplazamiento angular y corriente actual de servomotor 2	11
7	Desplazamiento angular y corriente actual de servomotor 2 (continuación de mensaje 6)	10
8	Desplazamiento angular y corriente actual de servomotor 3	11
9	Desplazamiento angular y corriente actual de servomotor 3 (continuación de mensaje 8)	10
10	Estado de entrada digital X3/IN 0 de servomotor 1	8
11	Información de error de servomotor 1 (eventual, sólo en caso de algún error en el servomotor 1)	6
12	Información de error de servomotor 2 (eventual, sólo en caso de algún error en el servomotor 2)	6
13	Información de error de servomotor 3 (eventual, sólo en caso de algún error en el servomotor 3)	6
	Total de datos	≥92 y ≤110

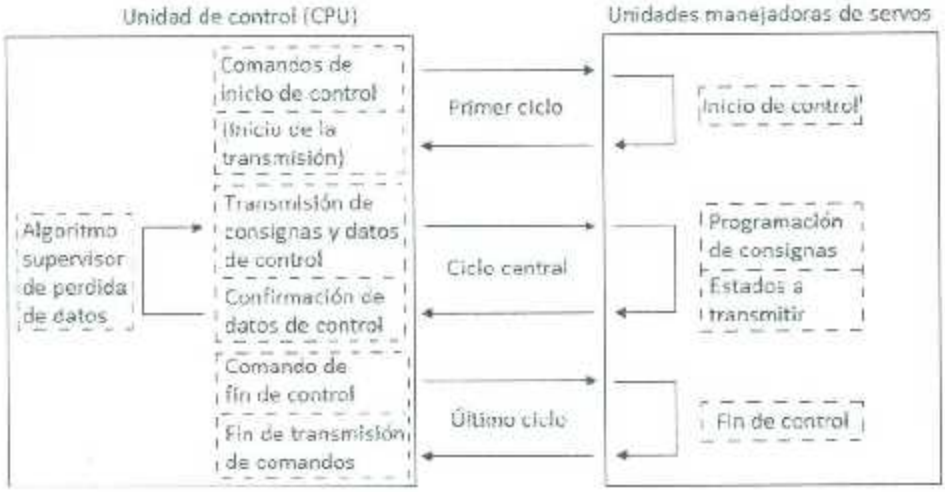


Figura 5.7: Secuencia de comandos de control

Capítulo 6

Interfaz de software

La muñeca esférica se puede manipular desde una interfaz gráfica de usuario (GUI) de Matlab, por medio de la cual se tiene acceso de forma sencilla a las funciones básicas de la misma. Con este fin se trabajó en el software Matlab R2010a, Simulink 7.5 y xPC Target 4.3, para desarrollar una interfaz gráfica que permitiera el control de un modelo de Simulink para generar código en lenguaje C y C++ dentro de la computadora de control (descrita en el capítulo 3) para su posterior ejecución en aplicaciones de control de la muñeca esférica. Para la generación y ejecución del código en lenguaje C y C++ se utilizó la herramienta Real-Time Workshop 7.5 y un compilador de código en lenguage C/C++, el Microsoft Visual Studio 2008. Todo el software antes mencionado se instaló en la computadora de desarrollo bajo el ambiente Windows XP SP3, que se encuentra en el Laboratorio de Mecatrónica y Control del ITL.

En este capítulo se presenta una breve descripción de las herramientas de software utilizadas, siguiendo con la descripción de los programas desarrollados con estas herramientas y después con la explicación de la operación de la interfaz gráfica; finalmente se presenta una lista de los errores más comunes que pueden aparecer durante la puesta en operación de la muñeca esférica.

6.1. Herramientas de software

Para crear la interfaz de control de la muñeca esférica se empleó software de desarrollo de MathWorks, Inc, así como algunas otras utilidades de software de terceros. Simulink es un software de modelado, simulación y análisis de sistemas dinámicos, que cuenta con la posibilidad de conectarse con hardware externo para observar su rendimiento en tiempo real; como es el caso de la interfaz de control que se desarrolló para la muñeca esférica. Simulink se integra con Matlab, lo que permite exportar los resultados de experimentación bajo el ambiente de Matlab para su posible esquematización y análisis personalizado, además de fungir como una interfaz virtual de comandos de entrada y salida para la ejecución de modelos en Simulink. También es necesaria la utilización de una herramienta adicional a Simulink, llamada xPC Target, que permite la descarga y ejecución de modelos desarrollados en Simulink a la computadora de control. Para la descarga y ejecución del modelo es necesaria la utilización de algunas utilidades de dos herramientas conocidas como Real-Time Workshop y un compilador de código en lenguaje C/C++ (ambas integradas a Simulink). Real-Time Workshop provee las utilidades para convertir los modelos de Simulink en código C/C++ y luego, con un compilador externo de código en lenguaje C/C++, convertirlo en código ejecutable en tiempo real.

6.1.1. Matlab

Matlab (acrónimo de Matrix Laboratory) es un lenguaje de alto nivel para cálculo técnico. En él se integra cálculo, visualización y programación en un ambiente amigable donde problemas y soluciones de carácter técnico son expresados en notación matemática común. Típicos usos de Matlab incluyen:

- Matemáticas y cálculo.
- Desarrollo de algoritmos.
- Modelado, simulación y creación de prototipos.

- Análisis de datos, exploración y visualización.
- Gráficas científicas y de ingeniería.
- Desarrollo de aplicaciones, incluyendo la construcción de interfaces gráficas de usuario.

Matlab cuenta con una familia de herramientas con soluciones a aplicaciones específicas llamadas "*toolboxes*". Estas herramientas son de gran ayuda a los usuarios de Matlab porque les permite aprender y aplicar tecnología especializada. Los *toolboxes* son una colección de herramientas basadas en funciones de Matlab (*M-files*) que extienden el enfoque de Matlab para resolver clases particulares de problemas. Algunas áreas en las cuales los *toolboxes* pueden actuar incluyen el procesamiento de señales, sistemas de control, redes neuronales, lógica difusa, simulaciones, y muchas otras. En particular, Matlab puede trabajar en conjunto con Simulink para la creación de prototipos mediante un *toolbox* llamado xPC Target que permite la descarga y ejecución de aplicaciones en tiempo real. Matlab provee una interfaz de línea de comandos para xPC Target desde donde se puede tener control completo de la computadora de control y de su aplicación a través de funciones tecleadas desde Matlab. El usuario de xPC Target puede teclear funciones desde la interfaz de línea de comandos de Matlab para:

- **Control de aplicaciones en tiempo real.** Descargar, ejecutar y detener la aplicación contenida en la computadora de control.
- **Adquisición y análisis de señales.** Guardar datos de señales mientras la aplicación se esta ejecutando en la computadora de control y analizar los datos después de que la aplicación se haya detenido, o desplegar datos de señales mientras que la aplicación se esta ejecutando en tiempo real.
- **Sintonización de parámetros.** Cambiar valores de parámetros mientras que la aplicación dentro de la computadora de control se está ejecutando en tiempo real.

6.1.2. Simulink

Simulink es un paquete de software para modelado, simulación y análisis de sistemas dinámicos. Es compatible con sistemas lineales y no lineales, modelado en tiempo continuo y tiempo discreto. Los sistemas simulados pueden tener multiples velocidades de operación, es decir, pueden tener diferentes componentes que son muestreados o actualizados a diferentes múltiplos de velocidades.

Para el modelado, Simulink proporciona una interfaz gráfica en la que se pueden conectar bloques de osciloscopios, fuentes, conectores, y muchos más; con el fin de construir la representación a bloques de sistemas dinámicos físicos y de controladores mediante operaciones de clic y arrastre de un ratón y un teclado para editar los parámetros de los bloques. En el modelado el usuario puede ver el diagrama de bloques a un nivel superior o inferior, dando doble clic sobre los bloques del diagrama para tener una percepción con más detalle de como el modelo está organizado y de como sus partes interaccionan. Después de que el usuario ha construido un modelo, tiene la posibilidad de seleccionar un modo de simulación mediante una barra de despliege del ambiente de Simulink. Seleccionando el modo de simulación externo, el usuario permite una comunicación entre el modelo y algún hardware externo (módulos I/O, tarjetas de comunicación, etc.) basada en una arquitectura cliente-servidor. En una arquitectura cliente-servidor el usuario tiene la posibilidad de modificar ciertos parámetros del modelo en cualquier momento de la simulación y de actualizar datos de señales provenientes del hardware externo. Por si fuera poco, el usuario puede poner los resultados de la simulación en un espacio de trabajo de Matlab para su posprocesamiento y visualización.

Al igual que Matlab, Simulink cuenta con una familia de herramientas con soluciones a aplicaciones específicas llamadas "*toolboxes*". Algunas áreas en las cuales los *toolboxes* de Simulink pueden actuar incluyen simulación de sistemas dinámicos en tiempo real, sistemas de comunicaciones, sistemas de visión, animaciones en 3D, creación de prototipos, y muchas otras. En el área de creación de prototipos, Simulink cuenta con un *toolbox* llamado xPC Target que trabaja en conjunto con Matlab para proveer aplicaciones de control de tiempo

real en puestas a prueba de hardware externo (sistemas de control, sensores, servomotores, etc.) mediante módulos I/O (de entradas y salidas digitales o analógicas) o tarjetas de comunicación instaladas en una computadora de control separada de la computadora en la cual se diseña la aplicación de control. Y debido a que Matlab y Simulink están integrados en un mismo paquete de software de MathWorks, Inc, el usuario puede simular, analizar, y revisar sus modelos bajo estos ambientes en cualquier momento.

6.1.3. xPC Target

La herramienta xPC Target de Simulink proporciona una solución para la creación rápida de prototipos y equipos de control en lazo cerrado, prueba e implementación de sistemas en tiempo real usando equipos de cómputo estándar. Es un ambiente que utiliza cualquier computadora como una computadora de control.

La computadora de control se encuentra separada de una computadora principal llamada computadora de desarrollo encargada de crear la aplicación de control y descargarla dentro de la computadora de control en modo kernel para su ejecución en tiempo real. El modo kernel es un modo de operación de la computadora de control que permite administrar directamente (sin restricciones) los recursos de hardware disponibles en el CPU sin necesidad de estar bajo un sistema operativo. La computadora de control se inicia en modo kernel mediante el uso de un disco de arranque insertado en su unidad lectora de CD-ROM o disquete de 3.5"; este disco de arranque se crea utilizando la interfaz de linea de comandos de Matlab como lo muestra el apéndice C.

El usuario de xPC Target puede crear un entorno de prueba en tiempo real para los modelos desarrollados en Simulink mediante la conexión de una computadora de desarrollo, una computadora de control y el equipo bajo prueba (sistemas de control, sensores, servomotores, etc.). El usuario conecta la computadora de desarrollo (donde se encuentra instalado xPC Target, Simulink, Real-Time Workshop o Simulink Coder, y un compilador de código en lenguaje C/C++) a la computadora de control sólo a través de un enlace

de comunicación TCP/IP o RS-232; luego se conecta la computadora de control al equipo bajo prueba y, mediante la computadora de desarrollo se descarga el código generado por Real-Time Workshop de un modelo de Simulink a la computadora de control en modo kernel vía el enlace de comunicación TCP/IP o RS-232. Para la conexión y comunicación de la computadora de control con el hardware externo, xPC Target proporciona una biblioteca de bloques compatibles con módulos I/O y protocolos de comunicación que permiten enviar y recibir información de hardware externo.

La biblioteca xPC Target dispone de programas de prueba desarrollados en Simulink para aprender a utilizar cada uno de sus bloques de programación. Los bloques extraídos de la biblioteca xPC Target pueden ser agregados dentro de modelos desarrollados en Simulink y descargados (junto con los drivers del correspondiente hardware) dentro de la computadora de control para su correcta operación. La figura 6.1 muestra los componentes que interfieren en un entorno de prueba en tiempo real utilizando xPC Target.

Figura 6.1: Componentes de un entorno de prueba xPC Target

Una vez realizadas las conexiones de la figura 6.1, el usuario puede:

- Acceder a la computadora de control y controlar de forma interactiva la aplicación descargada desde la computadora de desarrollo.
- Sintonizar parámetros antes, durante y después de la ejecución en tiempo real a través

de la computadora de desarrollo y de Simulink, en modo de simulación externo.

- Adquirir, visualizar y registrar datos de señales desde la computadora de desarrollo.

Un resumen de los requerimientos de hardware y software necesarios para utilizar xPC Target se muestra en las tablas 6.1 y 6.2, respectivamente.

Tabla 6.1: Requerimientos de hardware para xPC Target

Hardware	Características
Computadora de desarrollo	Computadora de escritorio o laptop.
Computadora de control	Computadora o CPU de escritorio, computadora industrial, PC/104, PC/104+, o computadora CompactPCI.
Módulos I/O o tarjetas de comunicación instaladas en la computadora de control	Módulos I/O o tarjetas de comunicación compatibles con xPC Target.

Tabla 6.2: Requerimientos de software para xPC Target

Software (instalado en la computadora de desarrollo)	Características
Matlab	Versión compatible con xPC Target.
Simulink	Versión compatible con xPC Target.
Convertidor de modelos de Simulink a código C/C++	Real-Time Workshop o Simulink Coder, ambos compatibles con xPC Target.
Compilador de código en lenguaje C/C++	Compilador Microsoft Visual C/C++, Open Watcom C/C++ o algún otro compatible con xPC Target.

Las computadoras de desarrollo y control deben de disponer de algunos recursos mínimos de hardware especificados en las tablas 6.3 y 6.4, respectivamente.

Los dispositivos periféricos como el monitor VGA y el teclado se pueden considerar de uso opcional debido a que no se consideran componentes esenciales para el funcionamiento básico de la computadora de control, es decir, la función básica de la computadora de control es la de almacenar la aplicación que se crea con el software de la computadora de desarrollo para comunicarse y tener control del equipo externo bajo prueba. Sin embargo, los componentes opcionales pueden ser utilizados para ampliar las funciones básicas de la computadora de control en el caso que se desee prescindir de algunas funciones básicas de

Tabla 6.3: Recursos mínimos de hardware de la computadora de desarrollo

Hardware	Descripción
Puerto de comunicación	Adaptador de red Ethernet o puerto serial. Empleado para establecer comunicación con la computadora de control.
CPU	Procesador Pentium, Athlon, o posterior.
Dispositivos periféricos	1. Disco duro con 60 MB de espacio libre. 2. Unidad grabable de CD-ROM o disquete 3.5". Empleado para crear el disco de arranque desde la unidad de programación.
RAM	128 MB o más.

Tabla 6.4: Recursos mínimos de hardware de la computadora de control

Hardware	Descripción
Puertos de comunicación	1. Tarjeta de red Ethernet (compatible con MathWorks xPC Target) o puerto serial. Empleado para comunicarse con la computadora de desarrollo. 2. Modulo I/O o tarjeta de comunicación (compatible con MathWorks xPC Target). Empleado para establecer comunicación entre la computadora de control y el equipo externo bajo prueba.
CPU	Intel 386/486/Pentium 32-64 bit o AMD K5/K6/Athlon con o sin un coprocesador de punto flotante.
Dispositivos periféricos	1. Unidad lectora de CD-ROM o disquetes 3.5". Empleado para insertar el disco de arranque. 2. Monitor VGA (opcional). 3. Teclado (opcional).
Medios extraíbles	CD-ROM o disquete de 3.5" autoarrancable. Empleado para arrancar el CPU en modo kernel.
RAM	8 MB o más.
Configuración de de la BIOS	1. Secuencia de arranque del sistema (unidad de CD-ROM o disquete) de la BIOS. 2. Inhabilitar todos los puertos USB. 3. Inhabilitar tecnología Hyper-threading.

la computadora de desarrollo; como por ejemplo la visualización de señales y el control de ejecución de la aplicación (desde la computadora de desarrollo) se pueden realizar desde la computadora de control con la ayuda de estos componentes opcionales. Finalmente, para el caso de computadoras de escritorio se recomienda hacer unos cambios a la configuración

de la BIOS del sistema para la optimización del funcionamiento de la computadora de control; además de establecer la secuencia de arranque para inicializar en modo kernel con ayuda del disco de arranque (CD-ROM o disquete de 3.5") creado mediante el software de MathWorks Matlab y xPC Target como se muestra en el apéndice C.

Una vez que se tiene una aplicación trabajando en el entorno de xPC Target, es posible poner en marcha la aplicación almacenada en la computadora de control sin necesidad de contar con una computadora de desarrollo. Para hacer esto es necesario la utilización de otra herramienta de Simulink llamada xPC Target Embedded Option, de la cual se hace mención sin profundizar en detalles, pero se puede encontrar más información en [MathWorks, 2010a].

6.1.4. Real-Time Workshop

Real-Time Workshop es una herramienta que proporciona las utilidades para generar el código fuente en lenguaje C/C++ y los archivos ejecutables para los algoritmos que se modelan gráficamente en el entorno de Simulink. El usuario puede generar código para la mayoría de los bloque de Simulink y funciones de Matlab que sean útiles para su aplicación embebida en tiempo real. Además, con la ayuda de Real-Time Workshop, xPC Target y un compilador de C/C++ de un proveedor externo se puede crear un aplicación ejecutable para ser almacenada en una computadora de control. Esta aplicación creada usa los parámetros iniciales establecidos en el modelo de Simulink que estaba disponible al mismo tiempo de la generación del código. En resumen, usando Real-Time Workshop el usuario puede:

- Generar el código fuente y los archivos ejecutables para sistemas de tiempo discreto, tiempo continuo, o sistemas híbridos modelados en Simulink.
- Usar el código generado tanto para aplicaciones de tiempo real y no real, incluyendo la puesta en marcha rápida de prototipos, y equipos o controladores de prueba en lazo cerrado.

- Sintonizar y monitorear el código generado por medio de bloques de Simulink.
- Producir código fuente para muchos productos y bloques de Simulink provistos por MathWorks, Inc. y otros fabricantes.

Real-Time Workshop es una herramienta que está completamente integrada con los productos de Matlab y Simulink, aunque para su utilización con xPC Target debe estar correctamente configurado como lo muestra el apéndice D.

6.1.5. Compilador de lenguaje C/C++

Un compilador de lenguaje C/C++ es un paquete que traduce un programa escrito en un lenguaje de programación C/C++ a otro lenguaje de programación, generando un programa equivalente que un dispositivo programable será capaz de interpretar.

Para el usuario de xPC Target, el compilador de código en lenguaje C/C++ se encarga de crear un código ejecutable a partir del código C/C++ generado por Real-Time Workshop. Después xPC Target usa este código ejecutable para crear una aplicación que sea ejecutable dentro del modo kernel de una computadora de control. El usuario de xPC Target debe descargar en la computadora de desarrollo un compilador de lenguaje C/C++ proveniente de un fabricante externo debido a que esta herramienta no está integrada al software proporcionado por MathWorks, Inc.

Para configurar el compilador que usará la herramienta de xPC Target, el usuario debe de usar el apéndice C y corroborar que los parámetros referentes al tipo y ruta de almacenamiento del compilador a utilizar sean los correctos. Además, el usuario debe corroborar que el compilador instalado sea compatible con el software de MathWorks, Inc, como se menciona en [MathWorks, 2010a].

6.2. Programa en Simulink

El programa desarrollado en Simulink desde la computadora de desarrollo corresponde a un modelo de programación de bloques que permite manejar una tarjeta de comunicación CAN (modelo CAN-AC2-PCI de la compañía Softing Industrial Automation) compatible con el software de xPC Target. La tarjeta CAN se encuentra instalada en la computadora de control con el fin de enviar mensajes para solicitar información del estado de la muñeca y de enviar las consignas de control para manipular la muñeca esférica. El programa (o modelo) de Simulink establece la programación necesaria para controlar el tráfico de mensajes de la computadora de control hacia las UICs de la muñeca y viceversa.

La figura 6.2 muestra un diagrama de flujo del funcionamiento del programa en Simulink; en él se muestra que al momento de correr el programa de Simulink la primera acción es configurar la tarjeta de comunicación CAN, es decir, establecer el puerto de comunicación, la velocidad de comunicación y el formato del mensaje de datos (normal o extendido), entre otros. Después se prosigue con verificar que el botón de paro de emergencia no esté activado para evitar maniobrar la muñeca por razones de seguridad; también se verifica que el control de paro por software no esté activado para impedir el movimiento de la muñeca. Las subrutinas de "Desactivar robot" y "Detener robot" permiten al usuario detener de forma normal la ejecución del programa en Simulink.

Una vez revisados los puntos anteriores, y antes de mover la muñeca, se debe revisar que la muñeca se encuentre en la posición de casa para posteriormente ejecutar una trayectoria de movimiento articular con su respectiva ley de control.

Para la ejecución de la ley de control se deben construir mensajes CAN de datos que contengan las consignas de control así como mensajes CAN que soliciten los estados de posición y corriente para cada articulación de la muñeca. La velocidad de cada articulación de la muñeca se calcula numéricamente a partir de la posición. Es importante mencionar que las consignas de control enviadas están acotadas según la tabla 6.5, la cual, muestra las consignas máximas definidas en el programa de Simulink que se pueden aplicar a las

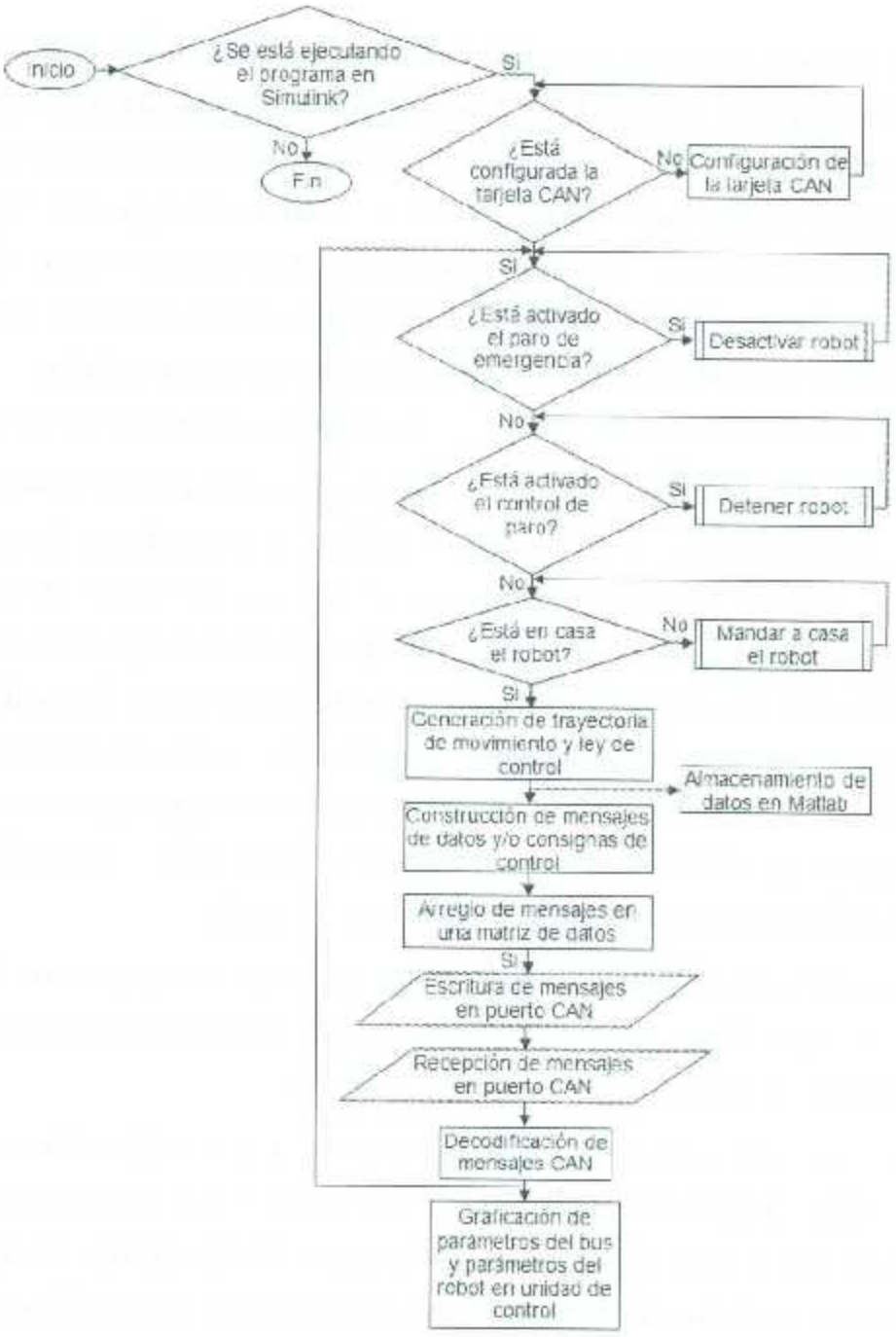


Figura 6.2: Diagrama de flujo del programa de Simulink

articulaciones de la muñeca para evitar sobrepasar los límites de la muñeca esférica. La articulación correspondiente al eje 3 no posee límite de posición articular, por lo que en la tabla 6.5 aparece la leyenda "*No aplica*".

Tabla 6.5: Consignas máximas definidas por software para las articulaciones de la MAS-KARA

Articulación	*Posición* [grad]	*Velocidad* [grad/s]	*Corriente* [A]
Eje 1	±360	+150	±3.6
Eje 2	±120	±240	±4
Eje 3	No aplica	+360	±2.1

Una vez construidos los mensajes de control y de solicitud de datos, se prosigue a ordenarlos dentro de una matriz de datos para su posterior envío en el puerto de comunicación CAN.

La recepción y decodificación de mensajes en el puerto CAN tiene lugar después de haber enviado los mensajes contenidos en la matriz de datos. Los mensajes decodificados corresponden a los estados de posición y corriente de cada articulación, y a la confirmación de consigna de control recibida por cada articulación. Además, en esta parte del programa es posible recibir mensajes espontáneos correspondientes a errores presentes en los módulos PowerCube de la muñeca, como por ejemplo, paro de emergencia, límites de articulaciones alcanzados, etcétera. Por último, la realimentación de los datos (extraídos de la decodificación de mensajes) se lleva a cabo para continuar con el lazo de programación.

La supervisión del bus de comunicación también es posible a través de este programa para conocer parámetros como el número de mensajes recibidos por la computadora de control, número de mensajes perdidos, número de mensajes vacíos y errores presentes en el bus de comunicación de la computadora de control. Además, el almacenamiento de datos de señales de control y de datos provenientes de la muñeca esférica también es realizado por el programa de Simulink para su posterior graficación bajo el ambiente de Matlab.

6.2.1. Algoritmos de control

Se trata de una serie de modelos de Simulink en los que se implementa una ley de control diferente con un periodo de muestreo de 2.5 ms especificado por Simulink. Cada uno de estos modelos alberga, a manera de subsistema, la programación de la ley de control con sus correspondientes entradas y salidas. Las señales de entrada y salida que están presentes hasta el momento en cada uno de estos modelos se describen en la tabla 6.6.

Tabla 6.6: Señales de entrada y salida de los controladores

Nombre	Descripción
Señales de entrada	
tc	Tiempo de control transcurrido
ts	Periodo de muestreo
q	Vector de posiciones del robot
qp	Vector de velocidades del robot
curr	Vector de corrientes del robot
Señales de salida	
qd	Vector de posiciones deseadas
qpd	Vector de velocidades deseadas
sC	Vector de señales de control
i	Vector de corrientes
modo_operacion	Señala el modo de operación del robot: - 1 para modo velocidad - 2 para modo corriente

Una vez creado el modelo de Simulink con su respectiva ley de control, éste es incluido como un bloque dentro del modelo principal de Simulink descrito en la sección 6.2. El número de controladores puede incrementarse conforme se agreguen nuevos algoritmos como se especifica en el apéndice E. Los modelos de Simulink que están disponibles hasta el momento con controladores en modo velocidad son:

- Controlador proporcional de regulación con saturación (tanh) de velocidad.
- Controlador proporcional de seguimiento con saturación (tanh) de velocidad.

Los modelos de Simulink que están disponibles hasta el momento con controladores en modo corriente son:

- Controlador PID de regulación.
- Controlador PID de seguimiento.

Los controladores se explican más a detalle en el capítulo 7.

6.3. Interfaz gráfica

La interfaz gráfica se desarrolla dentro de la computadora de desarrollo con el fin de proporcionar al usuario un ambiente amigable con las funciones necesarias para controlar la ejecución del programa explicado en la sección 6.2 y por ende controlar la operación de la muñeca esférica. La interfaz aquí desarrollada corresponde a una GUI de Matlab que es llamada al momento de teclear su nombre en la ventana de comandos de Matlab.

Al abrir la GUI de Matlab aparece en segundo plano el programa del modelo de Simulink que el usuario descargará a la computadora de control vía protocolo TCP/IP con ayuda de la GUI. La GUI cuenta con cuatro paneles que agrupan diferentes funciones para controlar la muñeca. Cabe mencionar que es mediante esta GUI que se manipula la comunicación entre el nivel 2 y nivel 3 de control de la muñeca, vía protocolo CAN. Esta comunicación inicia al presionar el botón 'Establecer comunicación' y termina al presionar 'Terminar comunicación' o 'Exit'. A continuación se describen los paneles con sus respectivas funciones que hasta al momento se encuentran en la GUI de la figura 6.3.

1. *Modos de operación.* Este panel permite seleccionar el modo de operación (velocidad o corriente) que se desea usar para controlar la muñeca.
2. *Controlador.* Este panel presenta una lista desplegable donde se selecciona el controlador que se desea probar. La lista desplegable muestra sólo los controladores

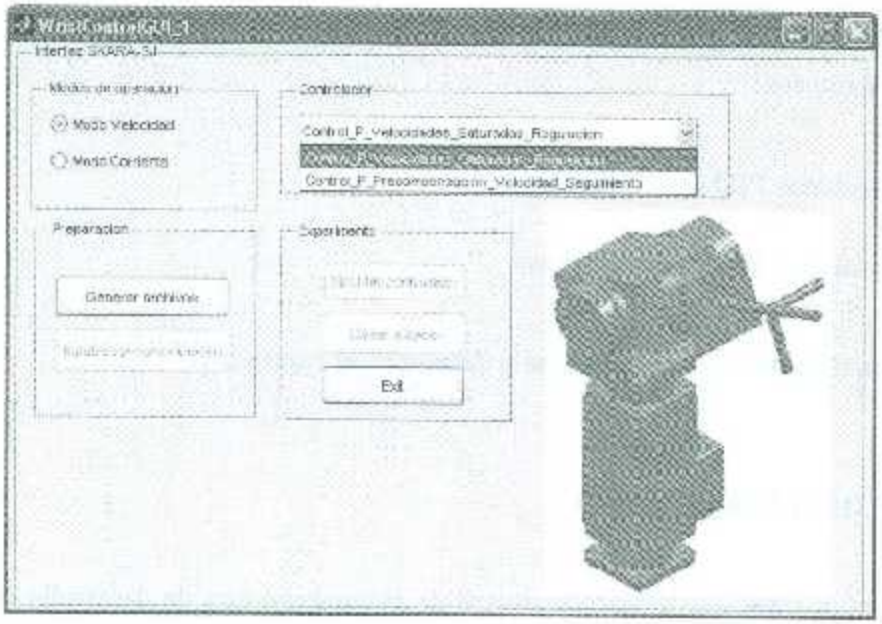


Figura 6.3: GUI de Matlab

disponibles para el modo de operación que se seleccione. En la subsección 6.2.1 se puede encontrar más información de los tipos de controladores que están disponibles y que pueden ser seleccionados.

3. *Preparación.* Este panel permite preparar la comunicación entre la computadora de control y las UICs de los módulos PowerCube antes de ejecutar el controlador seleccionado en los paneles "*Modos de operación*" y "*Controlador*".

   - *Generar archivos.* Este botón inicia el proceso de compilación del programa de Simulink explicado en la sección 6.2 con el fin de descargar los archivos generados en código C dentro de la computadora de control. Para la generación de archivos se emplea el software de Real-Time Workshop y un compilador de C/C++ proporcionado por Microsoft Visual Studio 2008. Si se encuentra alguna anomalía en el diseño del modelo de Simulink, el proceso de compilación no terminará satisfactoriamente indicando la causa del error.
   - *Establecer comunicación/Terminar comunicación.* Con este botón conmutador

se inicia o termina la comunicación entre la computadora de control y las UICs de la muñeca.

4. *Experimento.* Este panel presenta los controles necesarios para llevar a cabo un experimento después de haber manejado los paneles anteriores.

   - *Llevar a casa.* Al presionar este botón se le indica a la muñeca que se coloque en su posición de casa sin importar lo que se encuentre haciendo.
   - *Ejecutar controlador.* Una vez que se seleccionó el controlador, se inició la comunicación con la muñeca y se ubicó la muñeca en su posición de casa, con este botón se inicia la acción del controlador.
   - *Exit/STOP.* Con este botón conmutador se puede salir con toda seguridad de la GUI de Matlab y del programa de Simulink al mismo tiempo que se detiene la comunicación entre la computadora de control y las UICs de la muñeca, siempre y cuando no se esté ejecutando la acción del controlador; por otra parte, si la acción del controlador está siendo ejecutada en el robot, por medio de este mismo botón es posible detener la acción del controlador.

6.4. Programa ejecutable

El programa en código C ejecutable se encuentra almacenado en la computadora de control y es equivalente al programa en Simulink explicado en la sección 6.2. La conversión del programa de Simulink en código C ejecutable se logra utilizando el software de Real-Time Workshop y un compilador de código C/C++ provisto por Microsoft Visual Studio 2008, ambos instalados en la computadora de desarrollo al igual que el programa de Simulink y la interfaz gráfica. La interfaz gráfica es la herramienta del usuario con la cual se descarga el programa en código C en la computadora de control y se tiene control total de él desde la computadora de desarrollo.

La figura 6.4 muestra la interrelación explicada en el párrafo anterior entre el programa

en código C, el programa en Simulink y la GUI de Matlab.

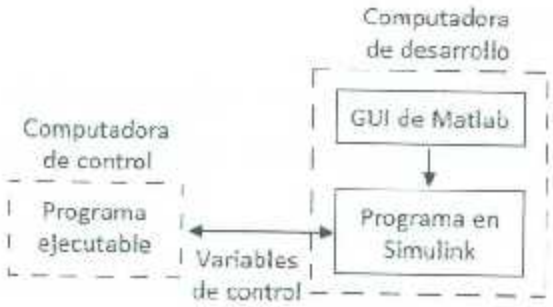


Figura 6.4: Interrelación del programa en código C, el programa en Simulink y la GUI de Matlab

El programa en código C se ejecuta en tiempo real dentro de la computadora de control y mantiene una comunicación periódica con las UICs de la muñeca vía protocolo CAN; es también por medio de este programa que es posible visualizar en tiempo real (con ayuda de un monitor) variables referentes al estado de la muñeca y del bus de comunicación, como se muestra en la figura 6.5.

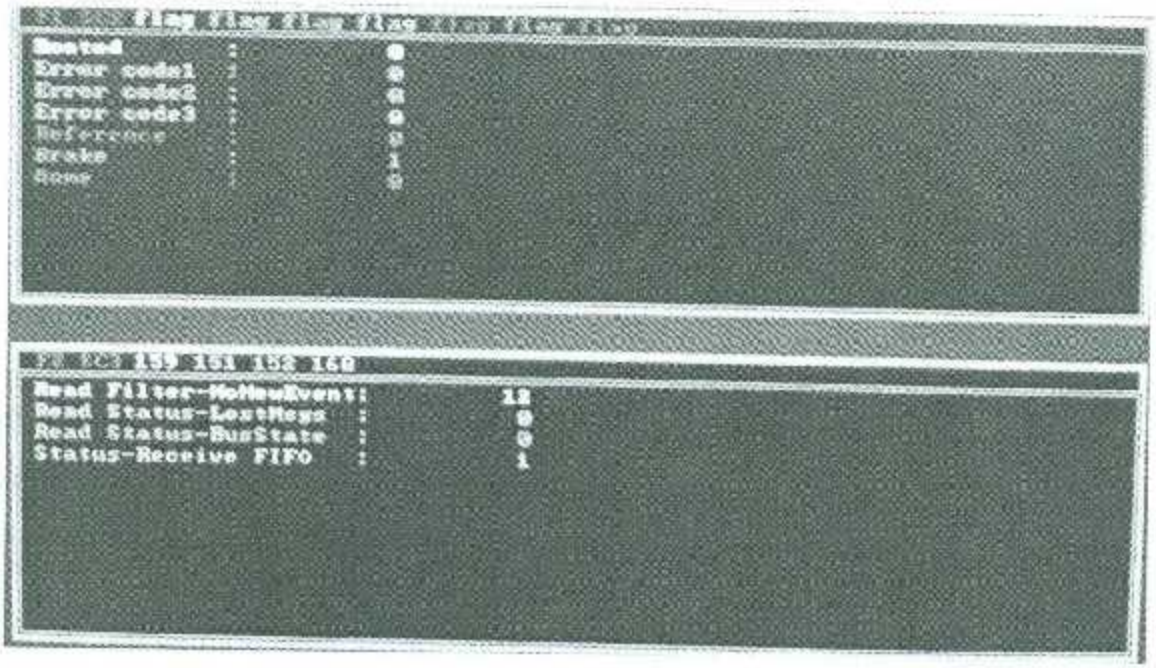


Figura 6.5: Información del estado de la muñeca y del bus de comunicación

La descripción de las variables que se visualizan en el monitor conectado a la compu-

tadora de control se presenta en la tabla 6.7. Para mayor información sobre la descripción del código de error presentado en el monitor se puede revisar la descripción de los módulos PowerCube en el capítulo 2.

La variable "Referenciado" toma el valor de '1' en el mismo instante que la muñeca se coloca en la posición de casa. La diferencia entre la variable "Casa" y la variable "Referenciado", es que esta última conserva el valor de '1' después de que la muñeca sale de la posición de casa. La variable "Referenciado" indica únicamente que los servomotores de la muñeca han ubicado su posición cero (de origen) a partir de la cual se establece la referencia para realizar cualquier tipo de movimiento (posición, velocidad o corriente).

6.5. Operación

En las subsecciones siguientes se describen los pasos necesarios para operar la muñeca MASKARA con la ayuda de la interfaz gráfica desarrollada en la computadora de desarrollo.

6.5.1. Iniciar el programa

Desde la computadora de desarrollo se inicia el programa de control al abrir la GUI con el nombre '*WristControlGui*' desde la ventana de comandos de Matlab con simplemente teclear el nombre de la GUI (con el directorio de Matlab ubicado en la ruta de acceso) y después dar Enter, aparece en seguida la GUI de Matlab mostrada en la figura 6.3 y en segundo plano aparece también el programa de Simulink que el usuario descargará a la computadora de control vía protocolo TCP/IP con ayuda de la GUI.

Tabla 6.7: Variables de estado de la muñeca y del bus de comunicación

Variable	Descripción
Booted	Indica si las UICs de la muñeca están listos (Booted=1) o no (Booted=0) para establecer comunicación con la computadora de control.
Error code 1	Almacena el código (en formato decimal) de un error presente en el servomotor 1 de la muñeca.
Error code 2	Almacena el código (en formato decimal) de un error presente en el servomotor 2 de la muñeca.
Error code 3	Almacena el código (en formato decimal) de un error presente en el servomotor 3 de la muñeca.
Referenciado	Indica si los servomotores de la muñeca están listos (Reference=1) o no (Reference=0) para ejecutar un movimiento.
Freno	Indica si la muñeca esta en estado de reposo (Freno=1) o movimiento (Freno=0).
Casa	Indica si la muñeca esta en la posición de casa (Casa=1) o no (Casa=0).
Read Filter-NoNewEvent	Indica el número de mensajes vacíos presentes en el buffer de recepción de la tarjeta CAN.
Read Status-LostMsgs	Indica el valor actual del contador de mensajes perdidos.
Read Status-BusState	Indica el estado actual del bus CAN. Valores posibles son 0 para error activo (funcionamiento adecuado del bus), 1 para error pasivo (bus con posibles errores) y 2 para bus off (bus con graves errores de transmisión y recepción).
Status-Receive FIFO	Indica el número actual de mensajes CAN almacenados en el buffer de recepción de la tarjeta CAN.

6.5.2. Antes de iniciar comunicación

Antes de iniciar comunicación con la muñeca MASKARA para empezar a controlarlo desde la interfaz gráfica, es necesario revisar los siguientes puntos:

- La computadora de control debe estar iniciada en modo kernel.
- La conexión del cable Ethernet entre la computadora de control y la computadora de desarrollo.
- La fuente de alimentación de los servos debe estar activa.

El estado de la computadora de control debe ser en modo kernel a través del disco de arranque de xPC Target, tal y como se explicó en la subsección 6.1.3. La figura 6.6 muestra la imagen que debe aparecer en el monitor de la computadora de control una vez que se encuentra en modo kernel; la pantalla del monitor muestra del lado derecho la configuración del disco de arranque descrita en el apéndice C y del lado izquierdo 8 parámetros descritos en la tabla 6.8.

Figura 6.6: Computadora de control en modo kernel de xPC Target.

Una vez revisados los puntos anteriores, se procede a seleccionar las características presentes en la GUI de Matlab con las que se desea controlar la muñeca. Después de que se inicia la comunicación no es posible cambiar estas características.

- **Modo de operación.** Una vez que se inicia la GUI de Matlab, es necesario seleccionar el tipo de consignas con las que se va a manejar la muñeca (velocidad o corriente). En la GUI por omisión aparece seleccionado el modo velocidad, aunque el usuario es libre de cambiar el modo de operación que se muestra en la figura 6.7(a).

Tabla 6.8: Parámetros en modo kernel de xPC Target

Parámetro	*Descripción*
Loaded App	Nombre de la aplicación cargada actualmente
Memory	Espacio de memoria disponible para la aplicación
Mode	Modo de operación actual de la computadora de control
Logging	Registro de variables por xPC Target
StopTime	Instante de tiempo en que se detendrá la aplicación
SampleTime	Periodo de muestreo de la aplicación
AverageTET	Tiempo de ejecución promedio de la aplicación
Execution	Tiempo de ejecución de la aplicación

(a) (b) (c)

Figura 6.7: Opciones de la GUI de Matlab

- **Selección del controlador.** El controlador a ejecutar se selecciona de la lista desplegable mostrada en la figura ??(b). En la GUI por omisión aparece seleccionado el controlador proporcional de posición con saturación (tanh) de velocidad, aunque el usuario es libre de cambiar el tipo de controlador antes de iniciar comunicación con la muñeca.

- **Generar archivos.** Al presionar el botón de la figura ??(c) se generan los archivos en código C correspondientes al programa de Simulink diseñado en la computadora de desarrollo. Una vez generados estos archivos, son descargados automáticamente dentro de la computadora de control vía protocolo TCP/IP para su posterior ejecución. El proceso de generación de archivos será necesario cada vez que se seleccione un controlador diferente o se cambie del tipo de controlador seleccionado después del proceso de generación de archivos.

6.5.3. Iniciar/detener comunicación

Una vez que la GUI de Matlab está activa (figura 6.3) y se seleccionaron las características con las que se va a controlar la muñeca, se inicia o detiene la comunicación con la muñeca presionando el botón conmutador de 'Establecer comunicación' o 'Terminar comunicación', respectivamente (figura 6.8(a)).

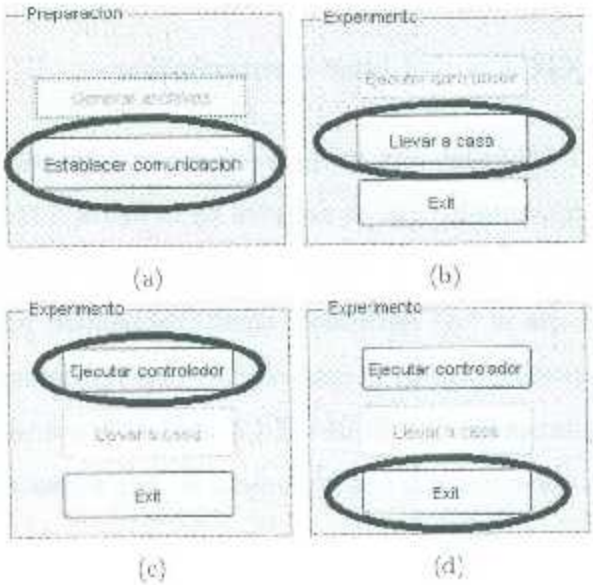


Figura 6.8: Botones de la GUI de Matlab

6.5.4. Llevar a casa

Una vez que se inició correctamente la comunicación con la muñeca, y antes de ejecutar algún tipo de controlador, es necesario llevar a casa la muñeca, presionando el botón que se muestra en la figura 6.8(b). En caso de que la ejecución de algún controlador se esté llevando a cabo en la muñeca, por medio de este botón es posible detener la ejecución del control y posteriormente enviar a casa la muñeca.

6.5.5. Ejecutar controlador

Para poder probar algún controlador es necesario seleccionarlo antes de iniciar la comunicación con el robot. Una vez que se inició correctamente la comunicación y se llevó a casa la muñeca, se puede ejecutar el controlador presionando el botón que se muestra en la figura 6.8(c).

6.5.6. Salir de GUI o detener controlador

Una vez que la GUI de Matlab está activa, y sin aun estar ejecutando algún controlador, por medio del botón conmutador que se muestra en la figura 6.8(d) es posible salir de la GUI y detener al mismo tiempo la comunicación con la muñeca (en caso de que sea necesario). En caso de que se esté ejecutando algún controlador, por medio de este mismo botón conmutador es posible detener el controlador. Una vez detenido el controlador, este mismo botón presenta la opción de salir de la GUI de Matlab con toda seguridad, al mismo tiempo que detiene la comunicación con la muñeca de una forma segura.

6.6. Errores más comunes

Algunas de las principales fallas al implementar el control de la muñeca MASKARA a través de la GUI de Matlab son:

- Los principales errores ocurren en la etapa antes de iniciar comunicación que se trató en la subsección 6.5.2; y es precisamente en el enlace de comunicación vía TCP/IP entre la computadora de desarrollo y la computadora de control que puede surgir un error de conexión al momento de generar los archivos (en código C) con la ayuda de la GUI de Matlab para descargarlos dentro de la computadora de control. Este error conocido como "*No target found*" (*Computadora de control no se encontró o no está conectada*) no permite la descarga de archivos dentro de la computadora

de control debido a una conexión defectuosa entre la computadora de desarrollo y la computadora de control. Para solucionar este problema se recomienda revisar el apéndice D y deshabilitar la protección Firewall de Windows para el puerto de red Ethernet de la computadora de desarrollo.

- Algunas veces, la función del botón "Generar archivos" de la interfaz gráfica (GUI) es interrumpida al momento de intentar descargar los archivos generados dentro de la computadora de control, por lo que al final del proceso de generación de archivos aparece un error proveniente del software Real-Time Workshop etiquetado como "*RTW error*". Las causas exactas de este error se desconocen hasta el momento, pero es posible solucionarlo cerrando y volviendo abrir la interfaz gráfica y el software de Matlab.

- Algunas veces, al momento de oprimir el botón de "*Establecer comunicación*" de la interfaz gráfica, uno de los tres servomotores del robot no está preparado para iniciar comunicación con la computadora de control, por lo que la variable '*Booted*' que aparece en el monitor conectado a la computadora de control tiene el valor de cero. Para que la variable '*Booted*' conmute a '1' (indicando que los tres servomotores están listos para iniciar comunicación con la computadora de control), es necesario detener la comunicación con ayuda de la GUI de Matlab, apagar y encender la fuente de alimentación de los manejadores de los servos, y posteriormente volver a iniciar la comunicación desde la GUI.

- El programa desarrollado en Simulink protege la muñeca limitando el valor máximo de las consignas de corriente y velocidad; además, la muñeca cuenta también con protecciones para cuando se exceden los límites articulares de la misma, cuando esto sucede, los servos son apagados automáticamente. Cuando se activa la protección de límites articulares, se genera un error correspondiente al módulo PowerCube (véase capítulo 2) que puede ser visible en el monitor conectado a la computadora de control, el cual, por el momento sólo puede ser depurado utilizando el software de programación estándar del fabricante (MTS) para sacar la muñeca de sus límites

[Schunk, 2012]

- El programa de Simulink desarrollado ha sido probado bajo experimentos en la muñeca MASKARA con un tiempo de muestreo mínimo de 2.5ms sin que se exceda el tiempo de respuesta de los manejadores del robot y sin que la comunicación falle. En caso de que un error de comunicación ocurra por perdida de datos durante el experimento, la muñeca se protege automáticamente deteniendo el movimiento rápidamente y activando los frenos de los servos; cuando esto sucede aparece un error de paro de emergencia que se puede visualizar en el monitor de la computadora de control.

Capítulo 7

Evaluación experimental

Empleando el sistema de control desarrollado, se han hecho diversos experimentos en las dos configuraciones, vertical y horizontal, de la muñeca esférica (ver sección B.3 del apéndice B). En cada una de las configuraciones de la muñeca, se han evaluado controladores tanto en modo corriente como en modo velocidad, ambos para los problemas de regulación y seguimiento. Las trayectorias deseadas en el espacio articular que fueron seleccionadas para evaluar experimentalmente los controladores de seguimiento se obtuvieron de [Kelly et al., 2005] y tienen la siguiente forma general

$$q_d(t) = b[1 - e^{-2.0t^3}] + c[1 - e^{-2.0t^3}]\sin(wt) \tag{7.1}$$

y que al ser derivada con respecto al tiempo resulta en

$$\dot{q}_d(t) - 6bt^2e^{-2.0t^3} + 6ct^2e^{-2.0t^3}\sin(wt) + [c - ce^{-2.0t^3}]\cos(wt)w \tag{7.2}$$

los parámetros b, c y ω fueron escogidos con el debido cuidado para evitar, tanto saturaciones de corriente en los actuadores como saturaciones de velocidad angular de los mismos actuadores. La tabla 7.1 muestra los valores de los correspondientes parámetros para las trayectorias deseadas de cada articulación. La figura 7.1 muestra el perfil de la trayectoria

deseada para las tres articulaciones de la muñeca.

Tabla 7.1: Parámetros para las trayectorias de los controladores de seguimiento

Articulación	b	c	ω
1	65	35	3
2	70	40	3.25
3	60	40	3

El periodo de muestreo para todos los experimentos fue de 2.5 ms.

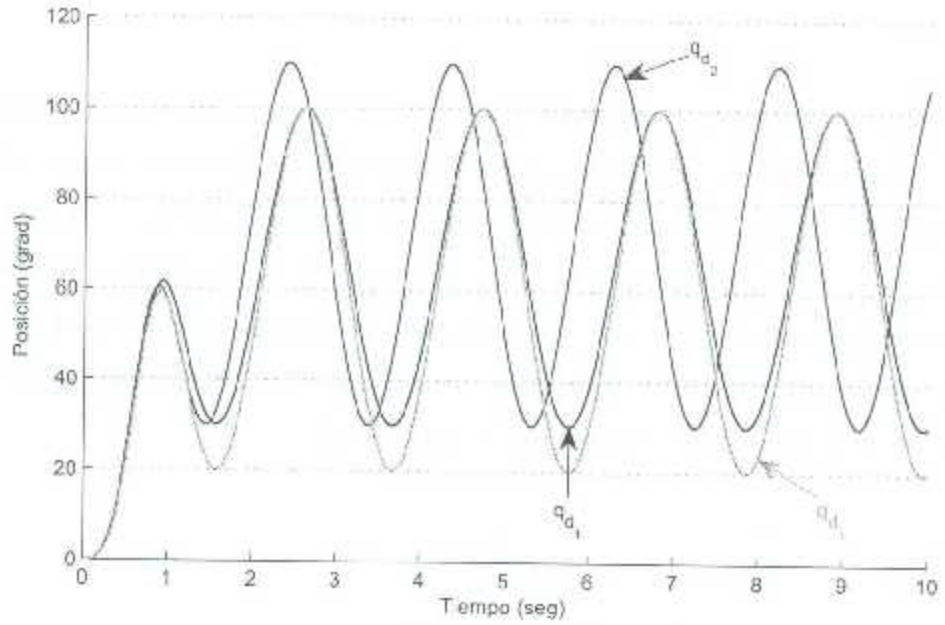


Figura 7.1: Trayectorias de posición deseada para los controladores de seguimiento

En este capítulo se describen los controladores probados experimentalmente para ambas configuraciones de la muñeca esférica, vertical y horizontal, y se presentan los resultados obtenidos.

7.1. Controladores en modo velocidad

A continuación se muestran los resultados obtenidos con los controladores en modo velocidad.

7.1.1. Control P con velocidades saturadas (regulación)

Primero se considera un controlador no lineal con saturación (tangente hiperbólica) descrito por la expresión,

$$\boldsymbol{v}_d = K_p \mathbf{tanh}(\boldsymbol{q}_d - \boldsymbol{q})$$

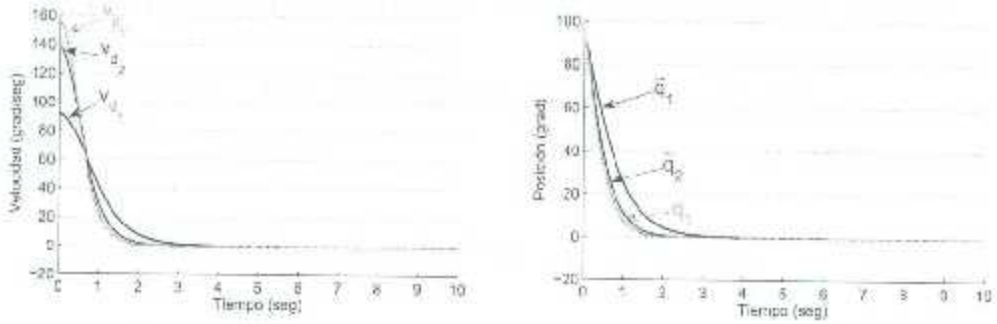
cuya función tangente hiperbólica es un vector de tres elementos representado por

$$\mathbf{tanh}(\boldsymbol{q}) = \begin{bmatrix} \tanh(q_1) \\ \tanh(q_2) \\ \tanh(q_3) \end{bmatrix}$$

donde $\boldsymbol{q} \in \mathrm{R}^3$ es el vector de posiciones articulares, $\boldsymbol{q}_d \in \mathrm{R}^3$ es el vector de posiciones articulares deseadas (que en este caso es constante), $\boldsymbol{v}_d \in \mathrm{R}^3$ es el vector de velocidades articulares aplicadas a la muñeca; $K_p \in \mathrm{R}^{3\times3}$ es una matriz diagonal de ganancias de control. Las ganancias de control utilizadas en este experimento han sido sintonizadas mediante un procedimiento heurístico que consiste en sintonizar tres ganancias (correspondientes a las ganancias proporcionales) para las tres articulaciones de la muneca empezando con valores pequeños y respetando los límites de velocidad de los actuadores de la muñeca. Las ganancias de control empleadas se muestran en la tabla 7.2. En las figuras 7.2 y 7.3 se tiene la evolución temporal de las velocidades entregadas por el controlador y de los errores de posición para las configuraciones vertical y horizontal de la muñeca, respectivamente. Se considera $\boldsymbol{q}_d = [90° \; 90° \; 90°]^T$ y la configuración de casa $\boldsymbol{q} = [0 \; 0 \; 0]^T$.

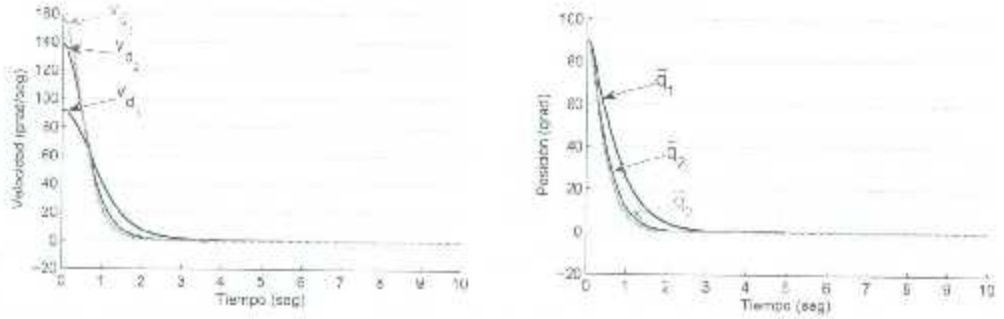
Tabla 7.2: Ganancias del controlador P con velocidades saturadas (regulación)

Articulación	$K_p[grados/s]$
1	100
2	150
3	170



(a) (b)

Figura 7.2: Resultados del experimento de regulación en modo velocidad y configuración vertical. (a) Consigna de velocidad deseada y (b) error de posición.



(a) (b)

Figura 7.3: Resultados del experimento de regulación en modo velocidad y configuración horizontal. (a) Consigna de velocidad deseada y (b) error de posición.

7.1.2. Control P con velocidades saturadas (seguimiento)

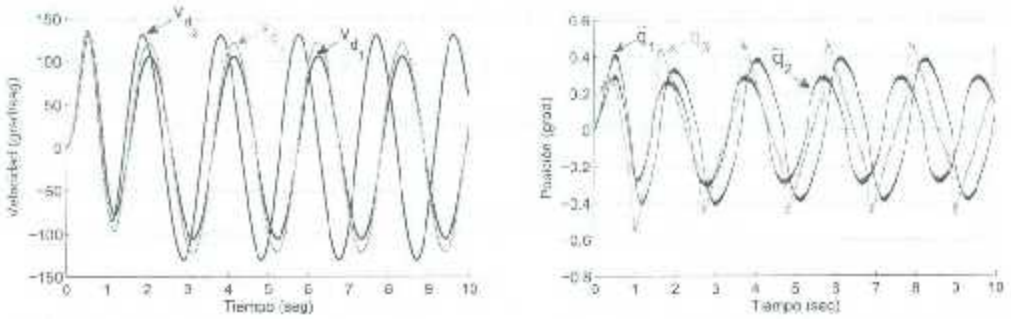
Este controlador es similar al anterior, también genera consignas de velocidad pero ahora aplicadas al problema de seguimiento. La ley de control es:

$$v_d = K_p \tanh(q_d - q) + \dot{q}_d$$

donde $\boldsymbol{q}$, $\boldsymbol{q}_a$, $\boldsymbol{v}_d$ y K_p tienen la misma función que en el caso de regulación, y sólo se ha agregado $\dot{\boldsymbol{q}}_d \in \mathrm{R}^3$ que es el vector de velocidades articulares deseadas. Las ganancias de control fueron sintonizadas utilizando el mismo método que en el caso de regulación. Las ganancias de control se muestran en la tabla 7.3. En las figuras 7.4 y 7.5 se tiene la evolución temporal de las velocidades entregadas por el controlador y de los errores de posición para las configuraciones vertical y horizontal de la muñeca, respectivamente, considerando la trayectoria mostrada en la figura 7.1.

Tabla 7.3: Ganancias del controlador P con velocidades saturadas (seguimiento)

Articulación	$K_p[grados/s]$
1	80
2	60
3	800

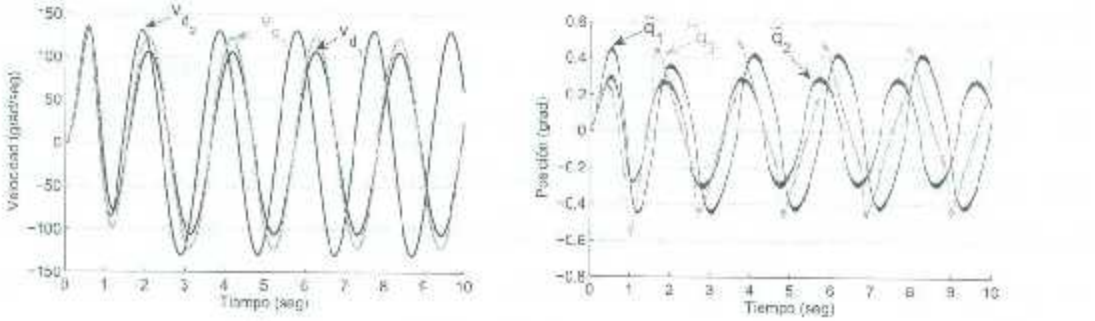


(a) (b)

Figura 7.4: Resultados del experimento de seguimiento en modo velocidad y configuración vertical. (a) Consigna de velocidad deseada y (b) error de posición.

7.2. Controladores en modo corriente

A continuación se muestran los resultados obtenidos en la muñeca esférica MASKARA con los controladores en modo corriente.



(a) (b)

Figura 7.5: Resultados del experimento de seguimiento en modo velocidad y configuración horizontal. (a) Consigna de velocidad deseada y (b) error de posición.

7.2.1. PID en regulación

El objetivo de control de posición se puede cumplir para robots sin efectos del vector de gravedad con un controlador PD, pero el robot MASKARA si es afectado por el vector de gravedad en ambas configuraciones de la muñeca esférica, es por esto que se tiene que agregar la acción integral para poder reducir el error a cero. La ley de control PID puede expresarse de la siguiente manera para el problema de regulación:

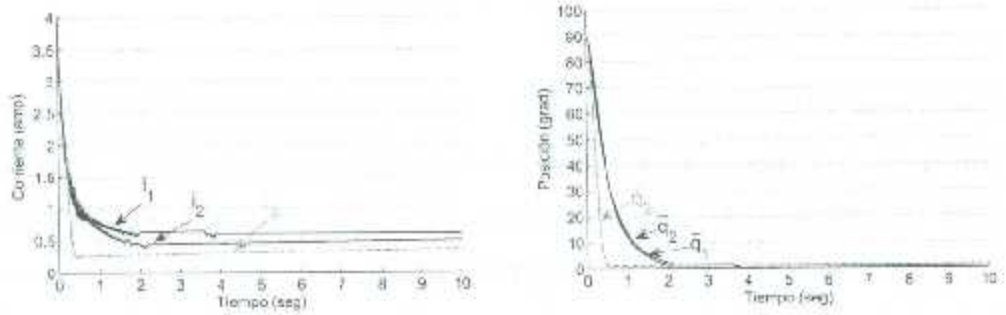
$$\boldsymbol{i}_d = K_p\tilde{\boldsymbol{q}} - K_v\dot{\boldsymbol{q}} + K_i\int_0^t \tilde{\boldsymbol{q}}(\sigma)d\sigma$$

donde las matrices de ganancias K_p, K_v, $K_i \in \mathbb{R}^{3x3}$, son matrices simétricas y definidas positivas (generalmente diagonales) cuyos elementos corresponden a las ganancias proporcional, derivativa e integral, respectivamente, y donde $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d - \boldsymbol{q}$ denota el error de posición ($\tilde{\boldsymbol{q}}, \boldsymbol{q}_d, \boldsymbol{q} \in \mathbb{R}^3$). Las ganancias de control fueron sintonizadas utilizando un método heurístico que consiste en sintonizar tres ganancias (correspondientes al controlador PID) por cada articulación de la muñeca empezando por establecer ganancias pequeñas para la acción proporcional (P) y definiendo valores de cero para las acciones derivativa (D) e integral (I). La acción proporcional del controlador PID se incrementa poco a poco en cada iteración hasta obtener un pequeño sobreimpulso de la señal medida de posición ($\boldsymbol{q}$) con respecto a la señal de posición deseada ($\boldsymbol{q}_d$). Posteriormente se repite el mismo

método (pequeños incrementos) para sintonizar la acción derivativa con el fin de amortiguar el sobreimpulso generado por la acción proporcional. Por último se sintoniza la acción integral del controlador PID estableciendo pequeños incrementos en cada iteración para disminuir el error de posición en estado estacionario. Las ganancias de control empleadas se muestran en la tabla 7.4. En las figuras 7.6 y 7.7 se muestra la evolución temporal de las consignas de corriente entregadas por el controlador y de los errores de posición para las configuraciones vertical y horizontal de la muñeca, respectivamente. Se considera $q_d = [90° \ 90° \ 90°]^T$, partiendo de la configuración de casa

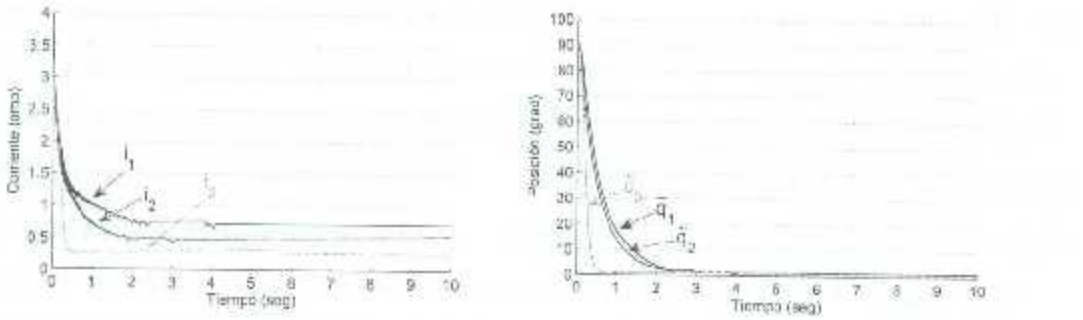
Tabla 7.4: Ganancias del controlador PID para regulación en modo corriente

Articulación	$K_p[A/grados]$	$K_i[A/(grados \cdot s)]$	$K_v[A/(grados/s)]$
1	0.04	0.012	0.01
2	0.044	0.008	0.01
3	0.055	0.01	0.003



(a) (b)

Figura 7.6: Resultados del experimento de regulación en modo corriente y configuración vertical. (a) Consigna de corriente deseada y (b) error de posición.



(a) (b)

Figura 7.7: Resultados del experimento de regulación en modo corriente y configuración horizontal. (a) Consigna de corriente deseada y (b) error de posición

7.2.2. PID en seguimiento

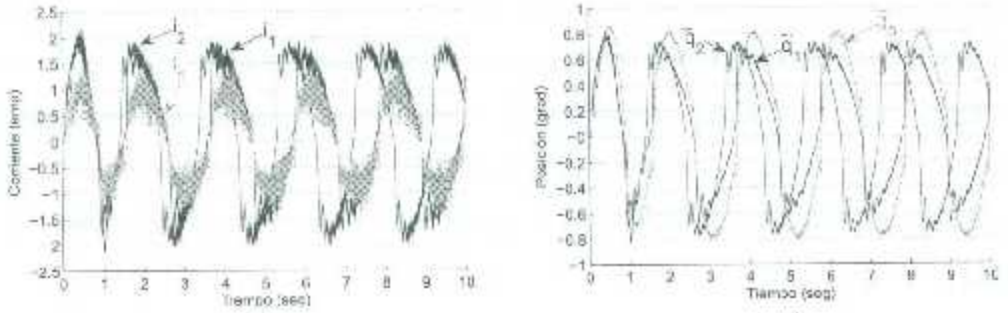
La ley de control PID, pero ahora para el problema de seguimiento, puede expresarse de la siguiente manera:

$$i_d = K_p\tilde{q} + K_v\dot{\tilde{q}} + K_i\int_0^t \tilde{q}(\sigma)d\sigma$$

donde únicamente se ha cambiado el segundo término del lado derecho ($\dot{\tilde{q}} = \dot{q}_d - \dot{q}$). Las ganancias de control fueron sintonizadas utilizando el mismo método que en el caso de regulación. Las ganancias de control empleadas se muestran en la tabla 7.5. En las figuras 7.8 y 7.9 se muestra la evolución temporal de las consignas de corriente entregadas por el controlador y de los errores de posición para las configuraciones vertical y horizontal de la muñeca, respectivamente, considerando las trayectorias mostradas en la figura 7.1.

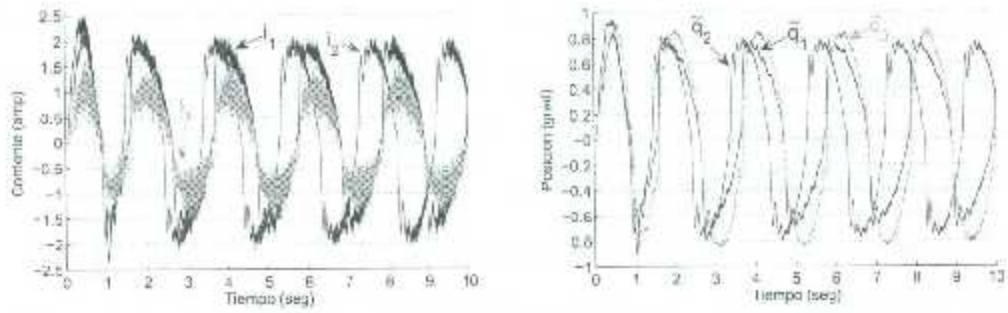
Tabla 7.5: Ganancias del controlador PID para seguimiento en modo corriente

Articulación	$K_p[A/grados]$	$K_i[A/(grados \cdot s)]$	$K_v[A/(grados/s)]$
1	2.5	0.008	0.02
2	2.5	0.008	0.01
3	1.2	0.0025	0.04



(a) (b)

Figura 7.8: Resultados del experimento de seguimiento en modo corriente y configuración vertical. (a) Consigna de corriente deseada y (b) error de posición.



(a) (b)

Figura 7.9: Resultados del experimento de seguimiento en modo corriente y configuración horizontal. (a) Consigna de corriente deseada y (b) error de posición.

7.3. Discusión de resultados

Se concluye que para ambos casos de configuración de la muñeca esférica (posición vertical y posición horizontal), las gráficas de los experimentos con los controladores en modo velocidad son más suaves y tienen menos ruido que las gráficas de los experimentos con los controladores en modo corriente. Por otro lado, se puede observar que en el primer experimento correspondiente al controlador proporcional con velocidades saturadas para el problema de regulación, no existe (prácticamente) variación alguna en las gráficas de las figuras 7.2 y 7.3, con referencia a las configuraciones vertical y horizontal de la muñeca esférica. Sin embargo, en el siguiente experimento del controlador proporcional con

velocidades saturadas aplicado al problema de seguimiento, se observa una pequeña variación al inicio de la señal del error de posición para ambas configuraciones de la muñeca en las figuras 7.4 y 7.5. Estas variaciones de respuesta de los controladores para ambas configuraciones de posición de la muñeca, son más fáciles de observar en los experimentos de controladores en modo corriente, ya que en este caso se presenta el efecto del vector de gravedad.

Otro punto importante de resaltar, es que para ambas configuraciones de la muñeca, el controlador para regulación en modo velocidad obtuvo una respuesta menos rápida con un error mínimo en estado estacionario en comparación con el controlador para regulación en modo corriente que obtuvo una respuesta más rápida pero con un error en estado estacionario mayor. Por otro lado, el controlador que obtuvo un mejor desempeño para el problema de seguimiento fue el saturado (tanh) en modo velocidad, debido a que se obtuvo una respuesta más suave con un error de posición menor a 0.6 grados.

Por lo general, la típica respuesta ruidosa que se obtiene con controladores en modo corriente es atribuida a fenómenos físicos no modelados, como la fricción y la transmisión de los servomotores, así como también a perturbaciones relacionadas con la dinámica eléctrica de los actuadores. Por último, es importante mencionar que los controladores aquí seleccionados no fueron sintonizados a la perfección.

Capítulo 8

Conclusiones

El diseño de una muñeca esférica utilizando componentes mecatrónicos inteligentes conocidos como módulos PowerCube, ha dado como resultado una plataforma robótica experimental de arquitectura abierta muy útil para implementar y evaluar diferentes algoritmos de control en tiempo real. El sistema resultante ha sido denominado MASKARA de "Module-Assembled Spherical-Kinematics Articulated Robot Arm". En este trabajo se han presentado los componentes que constituyen las interfaces de hardware y software que han sido utilizadas para operar esta muñeca tanto en modo corriente como en modo velocidad. Los experimentos de control llevados a cabo con este sistema muestran buenos resultados; las contribuciones más relevantes de este trabajo son:

- La construcción de la muñeca esférica.
- La obtención del modelo cinemático y dinámico que caracteriza a la muñeca esférica en función de sus parámetros cinemáticos y dinámicos.
- El desarrollo de un sistema de control para la implementación y evaluación de algoritmos de control en modo velocidad y corriente.

Es importante mencionar que aunque el sistema de control desarrollado emplea dos computadoras también es posible implementar este mismo sistema de control con una sola

computadora y utilizando el software xPC Target Embedded pero sin los beneficios que da la interfaz del software de Matlab.

Como trabajo a futuro, se propone mejorar el sistema de control desarrollado en el aspecto de almacenar y graficar datos de una manera más cómoda y práctica para el usuario. También se propone el desarrollo de una animación en 3D que permita visualizar el comportamiento de la muñeca cuando se este ejecutando algún experimento o simulación.

Por otro lado, en el aspecto de modelado, es indispensable la identificación de parámetros dinámicos de los eslabones que conforman el robot, como momentos de inercia, masas y centros de masa, con el fin de obtener un modelo dinámico completo de la muñeca esférica. Además, también podría ser deseable la identificación de los parámetros de fricción del robot para su inclusión en el modelo dinámico general del robot.

Bibliografía

[Amtec, 2004] Amtec Robotics, "Programmers guide for PowerCube: Developing PC Programs for PowerCube Modules", versión 1.2, 2004. (Disponible en `http://www.mx.schunk.com/schunk/schunk_websites/service/`).

[Barrientos et al., 1997] Barrientos A., Peñín L., Balaguer C. y Aracil R., "Fundamentos de Robótica", McGraw-Hill, 1997.

[Blomdell et al., 2005] Blomdell A., Bolmsjo G., Brogardh T., Cederberg P., Isaksson M., Johansson R., Haage M., Nilsson K., Olsson M., Olsson T., Robertsson A. and Wang J., "Extending an industrial robot controller: Implementation and applications of a fast open sensor interface", IEEE Robotics and Automation Magazine, Vol.12, no. 3, pp. 85-94, September 2005.

[Bosch, 1991] Bosch R., "CAN Specification Versión 2.0", Bosch, 1991. (Disponible en `http://esd.cs.ucr.edu/webres/can20.pdf`).

[Campa et al., 2008] Campa R., Torres E., Salas F. y Santibáñez V., "On modeling and parameter estimation of brushless DC servoactuators for position control tasks", Proceedings of the 17th IFAC World Congress, Seoul, Korea, July 2008.

[Craig, 1989] Craig J., "Introduction to Robotics: Mechanics and Control", Addison Wesley, 1989.

[Di Natale et al., 2012] Di Natale M., Zeng H., Giusto P. y Ghosal A., "Understanding and Using the Controller Area Network Communication Protocol", Springer, 2012.

[Domingo et al., 2003] Domingo J., Gámiz J., Grau A. y Martínez H., "Comunicaciones en el Entorno Industrial", Editorial UOC, 2003.

[ESD, 2012] ESD Electronic System Design Gmbh, "CAN-Wiring: Notes on the Wiring of CAN-Bus Systems and Cable Selection", Technical document No. C.1300.02/Rev. 4.0, 2012. (Disponible en `http://www.esd-electronics-usa.com/Shared/Library/CAN-Wiring-3.pdf`).

[Fu et al., 1988] Fu K., González R. y Lee C., "Robótica: Control, Detección, Visión e Inteligencia", McGraw-Hill, 1988.

[Kelly et al., 2005] Kelly R., Santibáñez V. y Loría A.,"Control of Robot Manipulators in Joint Space", Springer, 2005.

[Kelly y Santibáñez, 2003] Kelly R. y Santibáñez V.,"Control de Movimiento de Robots Manipuladores", Pearson Education, 2003.

[Koren et al., 1999] Koren Y., Heisil U., Jovane F., Moriwaki T., Pritschow G., Ulsoy G. y Van Brussel H., "Reconfigurable manufacturing systems", CIRP Annals-Manufacturing Technology, Vol.48, No. 2, pp. 527-540, February 1999.

[Krause et al., 2002] Krause, P. C., O. Wasynczuk, y S. D. Sudhoff, "Analysis of Electric Machinery and Drive Systems", Wiley-Interscience, 2002.

[Lui and Wu, 2001] Lui J. and Wu J., "Multiagent Robotic Systems", CRC Press, 2001.

[MathWorks, 2002] MathWorks, "xPC Target for Use with Real-Time Workshop: I/O Reference Guide", version 2, 2002. (Disponible en `http://www.mathworks.com/support/product/XP/productnews/xpc_target_io_ref_2001_July_02.pdf`).

[MathWorks, 2010a] MathWorks, Inc., "xPC Target 4: Getting Started Guide", version 4.3, 2010. (Disponible en `www.mathworks.com/support/`).

[MathWorks, 2010b] MathWorks, Inc., "Supported Ethernet Chipsets for xPC Target", version 4.3, 2010. (Disponible en `http://www.mathworks.com/products/`

simulink-real-time/supported/xpc-target-supported-ethernet-chipsets.pdf).

[Mayr et al., 2011a] Mayr J., Gattringer H. y Bremer H., "Online walking gait generation with predefined variable height of the center of mass". Proceedings of the International Conference on Intelligent Robotics and Applications, Aachen, Germany, December 2011.

[Mayr et al., 2011b] Mayr J., Gattringer H. y Bremer H., "On the suitability of different online gait generation methods for pre-defined footsteps", Proceedings of the 82nd Annual Meeting of the International Association of Applied Mathematics and Mechanics, Graz, Austria, December 2011.

[Monroy et al., 2001] Monroy C., Campa R. y Kelly R., "An application of real-time control systems to robotics", Robótica, Vol. 19, pp. 323-329, September 2001.

[Niku, 2001] Niku S., "Introduction to Robotics: Analysis, Systems, Applications", Prentice Hall, 2001.

[Ramírez, 2008] Ramírez C., "Modelado dinámico y control en modo par del robot Mitsubishi PA10-7CE", tesis de maestría, Instituto Tecnológico de la Laguna, Torreón, Coahuila, México, 2008.

[Schunk, 2008] Schunk, "Rotatory Modules: Automation", Technical document No. 9948231-6M-04/2008/en, 2008. (Disponible en http://www.mx.schunk.com/schunk/schunk_websites/service).

[Schunk, 2011a] Schunk, "Servo Electric Swivel Unit Type PR 70-110: Assembly and Operating Manual", Technical document No. 04/PR/en/2011-08-01/JH-CW, 2011. (Disponible en http://www.mx.schunk.com/schunk/schunk_websites/service).

[Schunk, 2011b] Schunk, "Servo Electric Rotary Pan-Tilt Actuator Type PW 70-90: Assembly and Operating Manual", Technical document No. 02/PW/en/2011-09-08/JH-

CW, 2011. (Disponible en http://www.mx.schunk.com/schunk/schunk_websites/service).

[Schunk, 2012] Schunk, "Motion Tool Schunk: Software Manual", Technical document No. 1.56/Motion Tool Schunk/12-06-19/en, 2012. (Disponible en http://www.mx.schunk.com/schunk/schunk_websites/service).

[Sciavicco y Siciliano, 2000] Sciavicco L. y Siciliano B., "Modelling and Control of Robots Manipulators", Springer, 2000.

[Smith y Christensen, 2009] Smith C. y Christensen H., "Constructing a high performance robot from commercially available parts", IEEE Robotics and Automation Magazine, Vol.16, pp. 75-83, December 2009.

[Softing, 2004] Softing Industrial Automation, "CAN-ACx-PCI: Hardware User Manual", versión 1.02.01, 2004. (Disponible en http://industrial.softing.com/en/downloads.html)

[Spong y Vidyasagar, 1989] Spong M. y Vidyasagar M., "Robot Dynamics and Control", Wiley, 1989.

[Strasser et al., 2008] Strasser T., Rocker M. y Ebenhofer G., "Distributed control concept for a 6-DOF reconfigurable robot arm", Proceedings of the 4th Intelligent Production Machines and Systems Virtual International Conference, Steyr-Gleink, Austria, July 2008.

[Tsai, 1999] Tsai L., "Robot Analysis: The Mechanics of Serial and Parallel Manipulators", John Wiley & Sons, 1999.

[Wang et al., 2010] Wang J., Li Y. y Zhao X., "Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm", International Journal of Advanced Robotic Systems, Vol.7, pp. 1-9, 2010.

Apéndice A

Interfaz para monitoreo y configuración de los módulos

En la ventana principal de la interfaz para monitoreo y configuración de los módulos PowerCube, conocida como MTS (Motion Tool Schunk) y representada en la figura A.1, es posible administrar la conexión de los módulos al bus de comunicación de la computadora de control, visualizar el estado del bus de comunicación, y procesar la configuración referente a los módulos PowerCube.

La ventana principal consiste de:

- Barra de menús. Proporciona las siguientes opciones:
  - File. Permite al usuario cargar, guardar o imprimir un archivo de configuración de la EEPROM correspondiente al módulo de PowerCube. También permite cerrar la aplicación MTS.
  - View. Permite desplegar información referente al bus de comunicación, marcar información y borrar mensajes de la ventana de salida.
  - Module. Permite activar, desactivar y buscar el o los módulos conectados al bus de comunicación de la computadora de control. También permite realizar

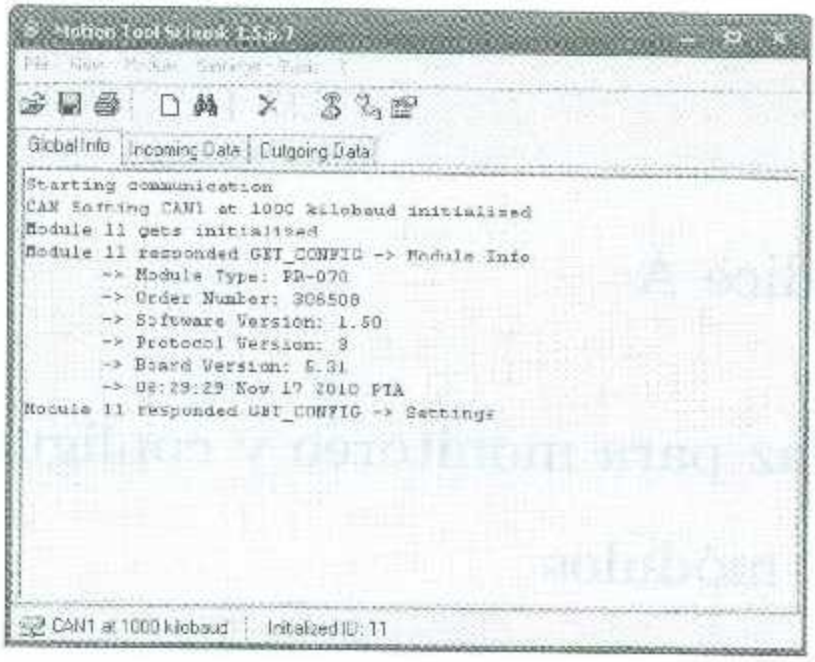


Figura A.1: Ventana principal de MTS

un diagnóstico de problemas; además de ofrecer la opción de enviar un paro de emergencia general a todos los módulos conectados.

- Settings. Permite seleccionar y abrir la interfaz de comunicación que utilizará la computadora de control para comunicarse con los módulos. También ofrece la opción de cambiar el idioma de la aplicación así como la opción de configurar aspectos básicos de inicialización de los módulos.
- Tools. Ofrece una herramienta para calcular el código de detección de error o CRC (Cyclic Redundancy Check). También ofrece una herramienta para convertir números de punto flotante a su representación en código hexadecimal (de 32 bits) y viceversa; además, proporciona una herramienta para crear código de programación para los módulos.
- ?. Proporciona información y documentación de ayuda respecto al software MTS.

- Barra de herramientas.

Tabla A.1: Barra de herramientas

Función	Descripción
	Carga información de EEPROM desde un archivo.
	La información contenida en la EEPROM de todos los módulos activos será guardada.
	La información contenida en la EEPROM de todos los módulos activos será impresa.
	Un módulo con número de identificación conocido puede ser activado.
	La interface de comunicación seleccionada realizará una búsqueda para identificar módulos conectados al bus de comunicación de la computadora de control.
	Un comando de paro de emergencia será enviado a todos los módulos conectados al bus.
	Abre o cierra la interface de comunicación configurada.
	Realiza una busqueda para configurar interfaces de comunicación disponibles en la computadora de control.
	Permite modificar ciertas configuraciones básicas de la aplicación para activar los módulos.

- Ventana de salida. Se divide en tres pestañas etiquetadas como "*Global Info*", "*Incoming Data*" y "*Outgoing Data*". La pestaña de *Global Info* presenta toda la información en texto simple relacionada a los módulos conectados al bus de comunicación de la computadora de control. La pestaña de *Incoming Data* muestra los paquetes de datos entrantes a la computadora de control (datos enviados por los módulos) y la pestaña de *Outgoing Data* muestra los paquetes de datos salientes de la computadora de control (datos enviados al bus de comunicación).

- Barra de estado. Muestra las propiedades actuales del bus de comunicación así como los números de identificación de los módulos activados.

Por otra parte, por medio de la barra de herramientas de la ventana principal se puede activar un módulo (con número de identificación conocido) conectado al bus de comunicación de la computadora de control. Esta opción proporciona una ventana de

programación del módulo correspondiente como se muestra en la figura A.2.

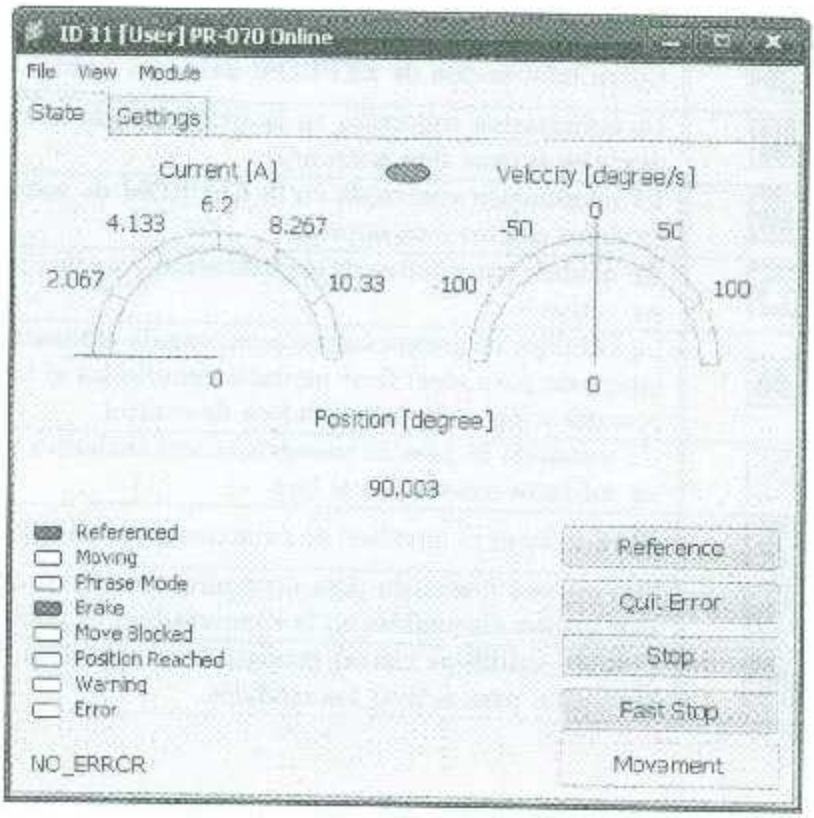


Figura A.2: Ventana de programación del módulo

La ventana de programación del módulo consiste de:

- Barra de menús. Proporciona las siguientes opciones:
  - File. Permite al usuario cargar, guardar o imprimir un archivo de configuración de la EEPROM correspondiente al módulo de PowerCube. También permite cerrar la ventana de programación del módulo.
  - View. Permite actualizar valores del estado del módulo, graficar los estados de corriente, velocidad y posición, y pausar las gráficas.
  - Module. Permite al usuario seleccionar entre varias funciones básicas de control del módulo, como por ejemplo programación de movimientos, pruebas de

comunicación, lectura y escritura de la EEPROM, entre otras. Es posible que para utilizar algunas de las funciones de control aquí mostradas sea necesario introducir una contraseña llamada "Password Flash", la cual es "Start!" (sin comillas). También se presenta la opción de seleccionar el cambio de usuario ("Change User") para modificar parámetros de configuración del módulo en la pestaña de "*Settings*" de la ventana de operación del módulo. El cambio de usuario establece el grado de autorización para modificar parámetros críticos del módulo.

- Ventana de operación del módulo. Se divide en dos pestañas etiquetadas como "*State*" y "*Settings*". La pestaña de "*State*" presenta al usuario la información actual de los estados de corriente, velocidad y posición, además de mostrar al usuario los controles básicos de operación del módulo como *Reference* (tomar posición de origen), *Quit Error* (limpiar error de módulo), *Stop* (detener módulo), *Fast Stop* (paro de emergencia) y *Movement* (para ejecutar movimientos en modo posición, velocidad o corriente). En la pestaña de "*State*" también se puede visualizar por medio de leds indicadores si el módulo esta en movimiento, frenado, estado de error, etcétera. La pestaña de "*Settings*" presenta al usuario información referente a la configuración del módulo como por ejemplo parámetros del método de comunicación, parámetros del servomotor, parámetros de los lazos de control internos, entre otros. Esta pestaña también permite al usuario la modificación de ciertos parámetros dependiendo del tipo de usuario seleccionado en "Module" ("Change User") ubicado en la barra de menú.

Para la modificación de parámetros desde la pestaña "*Settings*", existe la siguiente clasificación de usuarios con sus respectivas contraseñas:

- Profi. Usuario con contraseña de acceso para un amplio rango de ajuste de parámetros del módulo. La contraseña es "Schunk" (sin comillas).

- Diag. Usuario con contraseña de acceso para ejecutar un diagnóstico del módulo. La contraseña es "Diag" (sin comillas).
- Advanced. Usuario con contraseña de acceso para un rango de ajuste de parámetros con más derechos que Profi. La contraseña es "?SCHUNK!" (sin comillas).

Para mayor información de la interfaz para monitoreo y configuración llamada MTS se puede consultar [Schunk, 2012].

Apéndice B

Construcción de la muñeca esférica

El diseño mecánico de la muñeca esférica incluye maquinar tres piezas adicionales de unión (porque se usan para unir o acoplar los módulos) y un marco de coordenadas que simule la orientación del órgano terminal. Las características que han sido tomadas en cuenta para diseñar estas piezas adicionales de unión fueron las siguientes:

- El diseño de las piezas adicionales de unión se hará conforme a las dimensiones de las piezas originales de unión proporcionadas por el fabricante de los módulos PowerCube tipo PR-70 y PW-70 (Schunk). Estas piezas originales de unión se pueden consultar en [Schunk, 2008].
- El material a utilizar para el maquinado de las piezas de unión se hará conforme al material de las piezas originales de unión; aunque es posible utilizar otro material siempre y cuando no se afecte el rendimiento de las articulaciones del robot.
- No deben existir colisiones entre los elementos propios de la muñeca (las posibles colisiones con el ambiente son independientes a este planteamiento).
- Un armado sencillo y rápido.

B.1. Diseño de piezas adicionales

El software que se utilizó para diseñar las piezas de unión fue SolidWorks. En él se diseñaron los modelos 3D y los correspondiente planos de cada uno de los elementos de unión, así como del marco de coordenadas. Posteriormente estos planos fueron enviados a un taller de torno para su maquinado. Es importante mencionar que las dimensiones expresadas en los planos están dadas en milímetros (si no están encerradas en corchetes) o en pulgadas (si están encerradas en corchetes).

B.1.1. Base escuadra de acero

La base escuadra de acero une la brida cuadrada de acero con el pilar de acero que esta empotrado en el suelo. Además esta base escuadra permite colocar la muñeca esférica en posición vertical y horizontal con respecto al suelo. Las dimensiones de la base escuadra de acero están especificadas en la figura B.1.

B.1.2. Brida cuadrada de acero

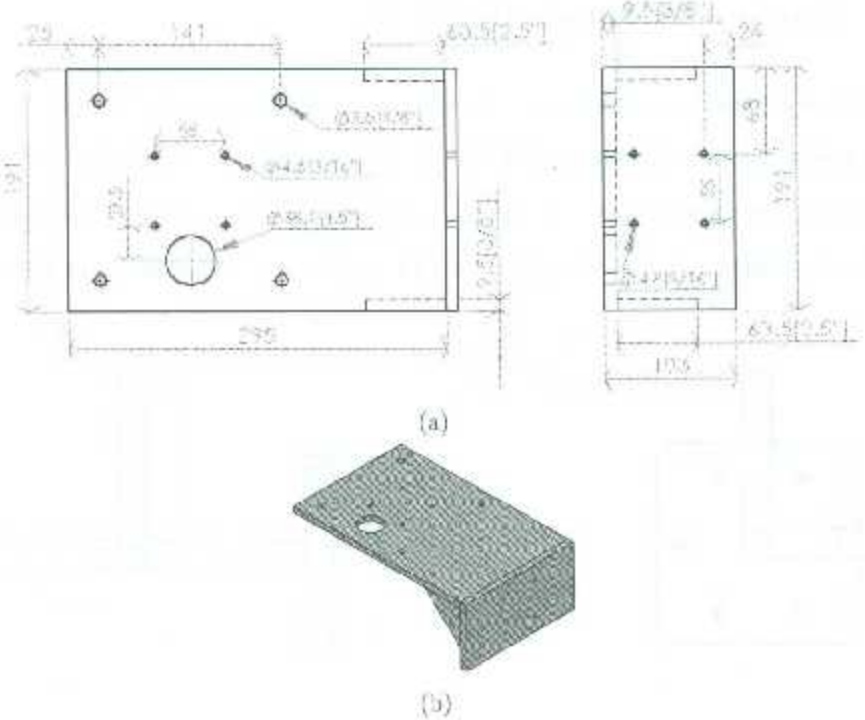
La brida cuadrada de acero une la base escuadra de acero con la primera articulación del robot. Físicamente, permite unir la parte fija del módulo PowerCube PR-70 (estator) con la base escuadra de acero.

Las dimensiones de la brida cuadrada de acero están especificadas en la figura B.2.

B.1.3. Brida cuadrada de aluminio

La brida cuadrada de aluminio une la primera articulación con la segunda articulación del robot. Físicamente, permite unir la parte móvil del módulo PowerCube tipo PR-70 (rotor) con la la parte fija del módulo PowerCube tipo PW-70 (estator).

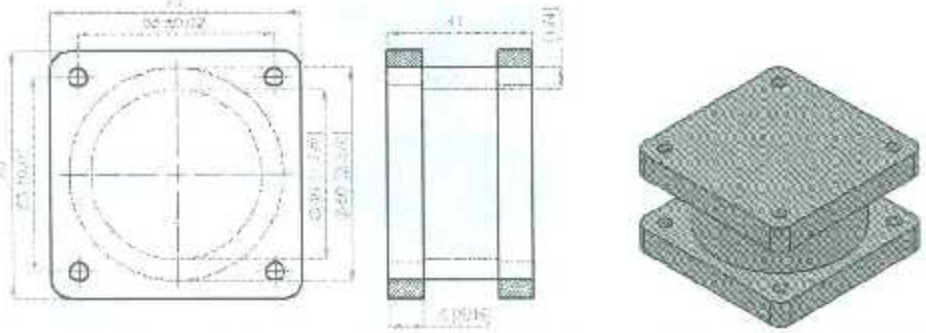
Las dimensiones de la brida cuadrada de aluminio son las mismas que las presentadas



(a)

(b)

Figura B.1: Base escuadra: a) principales dimensiones, b) representación en 3D



(a) (b)

Figura B.2: Brida cuadrada: a) principales dimensiones, b) representación en 3D

en la figura B.2 para la brida cuadrada de acero; la única diferencia esta en el material de fabricación.

B.1.4. Marco coordenado de PVC

El marco coordenado se compone de tres ejes ortogonales (X, Y y Z) y de una base cuadrada; todos hechos de material PVC. Este marco de coordenadas se conecta en la última articulación permitiendo visualizar la orientación del extremo final del robot. Las dimensiones del marco de coordenadas de PVC están especificadas en la figura B.3.

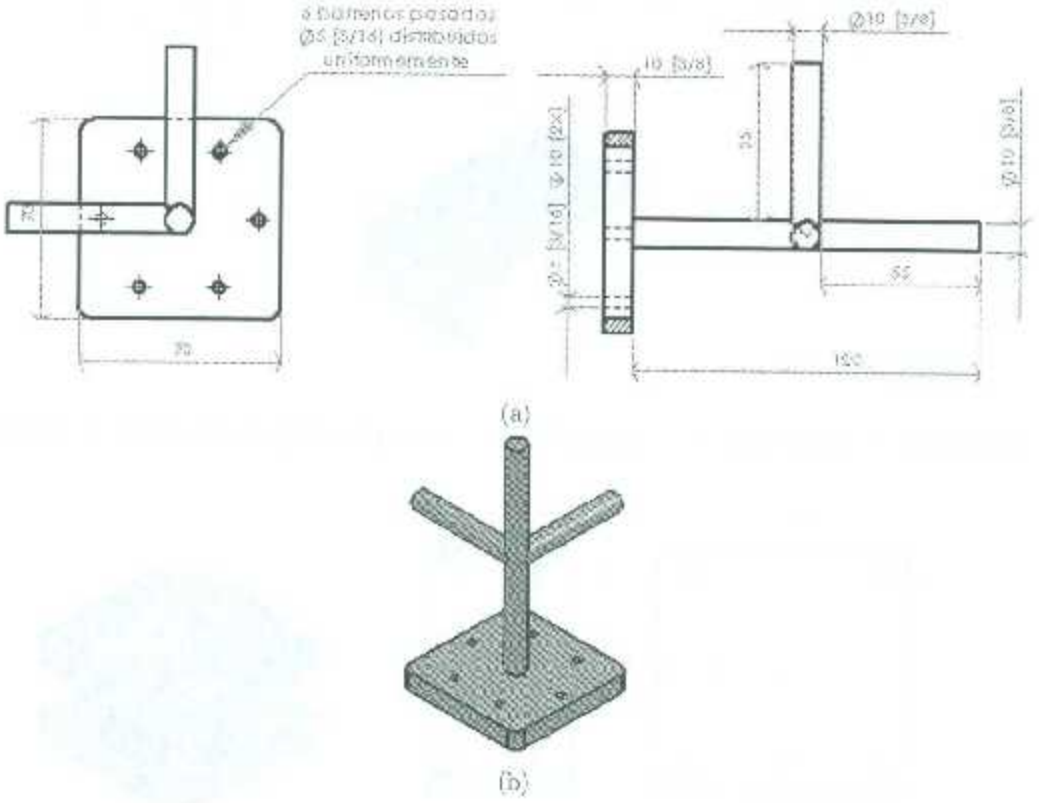


Figura B.3: Marco coordenado: a) principales dimensiones, b) representación en 3D

B.2. Ensamblado de la muñeca esférica

El proceso de ensamblado de la muñeca esférica se lleva a cabo en el interior del laboratorio de Mecatrónica y Control del Instituto Tecnológico de La Laguna, sobre un pilar de acero que esta empotrado en el piso del laboratorio. Para ensamblar los elementos

que componen la estructura mecánica de la muñeca esférica se deben de seguir los siguientes pasos:

1. Unir la brida cuadrada de acero con la parte fija (estator) del módulo PowerCube tipo PR-70 utilizando los tornillos de rosca métrica $M4 \times 1''$ y agregando una arandela plana y una de presión en cada tornillo.

2. Unir la brida cuadrada de acero con la base escuadra de acero (en cualquiera de sus dos ángulos) utilizando los tornillos de rosca estándar de $3/16'' \times 1''$ y colocando una tuerca con una arandela plana y una de presión en cada tornillo. La tuerca con su arandela plana debe quedar del lado de la brida cuadrada de acero y la arandela de presión del lado de la base escuadra.

3. Unir la base escuadra de acero con el pilar de acero que esta empotrado en el suelo utilizando los tornillos de rosca estándar de $3/8'' \times 1''$ y agregando 2 arandelas (una plana y otra de presión) con una tuerca en cada tornillo. La arandela plana queda del lado de la escuadra y de la cabeza del tornillo, y la arandela de presión del lado de la tuerca.

4. Unir la brida cuadrada de aluminio con la parte móvil (rotor) del módulo PowerCube tipo PR 70 utilizando los tornillos de rosca métrica $M4 \times 1''$ y agregando una arandela de presión en cada tornillo.

5. Unir la brida cuadrada de aluminio con la parte fija (estator) del módulo PowerCube tipo PW-70 utilizando los tornillos de rosca métrica $M4 \times 3/4''$ y agregando una arandela de presión en cada tornillo.

6. Unir el marco de coordenadas de PVC con el segundo eje del módulo PowerCube tipo PW-70 utilizando los tornillos $M4 \times 3/4''$ y agregando una arandela plana en cada tornillo.

B.3. Instalación de la muñeca esférica

Una vez realizado el ensamblado de la muñeca esférica descrito en el apéndice B.2 es posible operar el robot en dos configuraciones de postura, configuración vertical y configuración horizontal; como se muestra en las figuras B.4 y B.5, respectivamente.

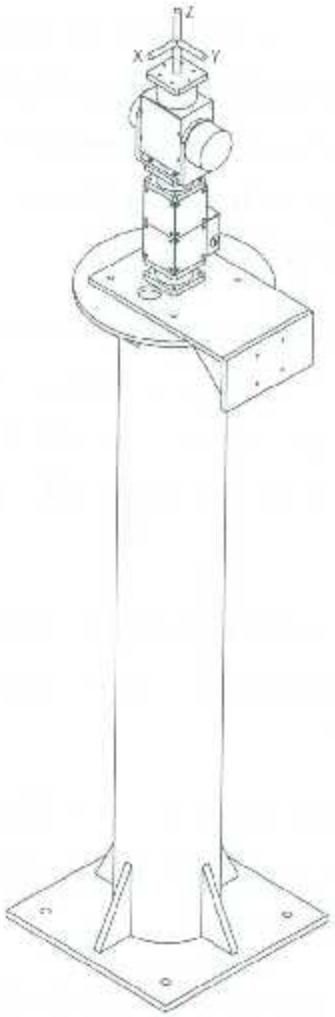


Figura B.4: Muñeca esférica instalada de forma vertical

Durante la etapa de instalación de la muñeca, se deben prevenir colisiones con el ambiente considerando que el espacio de trabajo de la muñeca abarca una semiesfera con

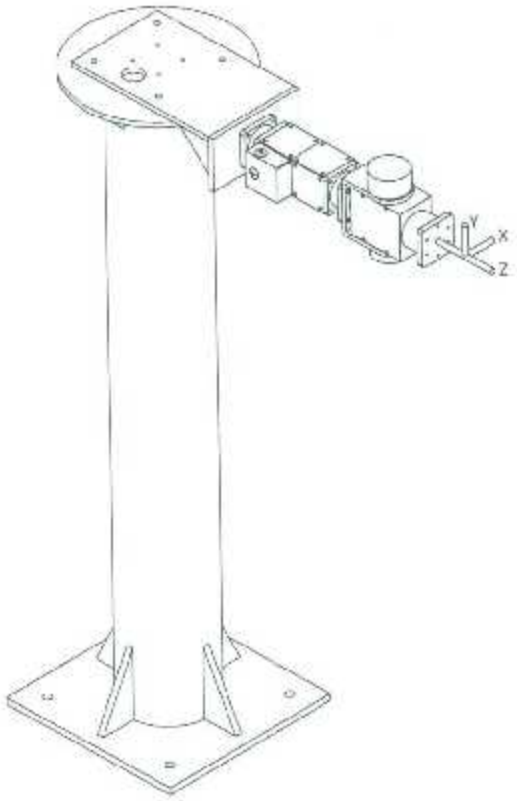


Figura B.5: Muñeca esférica instalada de forma horizontal

radio de 22.51 cm y centro en la intersección de los 3 ejes articulares, como se muestra en la figura B.6.

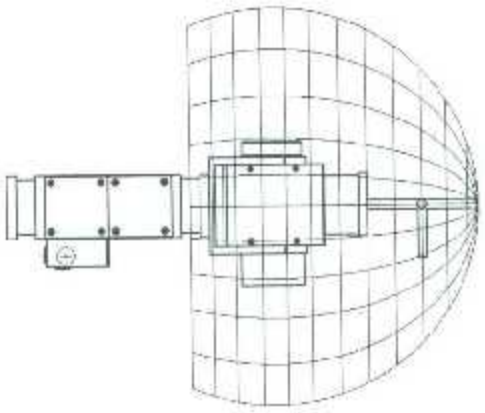


Figura B.6: Espacio de trabajo de la muñeca esférica

Apéndice C

Creación de un disco de arranque para la computadora de control

La computadora de control requiere de un disco de arranque para inicializar el modo de funcionamiento en modo kernel. Además, en este mismo disco se especifican los parámetros de configuración del protocolo de comunicación TCP/IP a utilizar por la computadora de control para comunicarse con la computadora de desarrollo, así como la configuración de otros parámetros del software de MathWorks xPC Target para disponer de los recursos de hardware de la computadora de control.

Para crear un disco de arranque en modo kernel para el CPU de la computadora de control del nivel 3 de la arquitectura de la muñeca MASKARA, se deben seguir los siguientes pasos en el software de MathWorks Matlab y xPC Target instalados en la computadora de desarrollo:

1. Insertar un CD-ROM vacío en la unidad grabable de CD-ROM de la computadora de desarrollo.

2. En la ventana de comandos de Matlab, teclear

   >> getxpcenv

3. Asegurarse que las propiedades del software de xPC Target estén configuradas como se muestra en la tabla C.1.

Tabla C.1: Parámetros de xPC Target

Version	: 4.3
CCompiler	: VisualC
CompilerPath	: C:\Microsoft Visual Studio 9.0
Name	: TargetPC1
TargetRAMSizeMB	: Auto
MaxModelSize	: 1MB
SecondaryIDE	: off
NonPentiumSupport	: off
MulticoreSupport	: on
HostTargetComm	: TcpIp
RS232HostPort	: COM1
RS232Baudrate	: 115200
TcpIpTargetAddress	: 192.168.0.10
TcpIpTargetPort	: 22222
TcpIpSubNetMask	: 255.255.255.0
TcpIpGateway	: 255.255.255.255
TcpIpTargetDriver	: Auto
TcpIpTargetBusType	: PCI
TcpIpTargetISAMemPort	: 0x300
TcpIpTargetISAIRQ	: 5
TargetScope	: Enabled
xPC Target Embedded Option	
EmbeddedOption	: Enabled
TargetBoot	: CDBoot
TargetMACAddress	:
BootFloppyLocation	: a:
CDBootImageLocation	: C:\work\xPC\cdimage
DOSLoaderLocation	: C:\work\xPC\cdimage

4. Si las propiedades anteriores no están correctamente configuradas, usar la función *setxpcenv* para corregirlas. Por ejemplo:

   >> setxpcenv('TargetBoot','CDBoot')

   >> setxpcenv('CDBootImageLocation','C:\work\xPC\cdimage') >> updatexpcenv

5. En la ventana de comandos de Matlab, teclear

   >> xpcbootdisk

   El software de xPC Target muestra el siguiente mensaje y crea la imagen ISO del disco de arranque.

   Current boot mode: CDBoot

   CD boot image is successfully created

6. Realizar una de las siguientes acciones, dependiendo del software instalado:

   - Si se cuenta con Microsoft Windows XP (SP2 o SP3), Vista o 7 (seven) con la herramienta Image Mastering API v2.0 (IMAPIv2.0), la función *xpcbootdisk* solicita al usuario insertar un disco vacío en alguna unidad grabable de CD-ROM disponible.

     Insert an empty CD/DVD. Available drives:

     [1] *D:*

     [0] *Cancel Burn*

     Seleccione la unidad apropiada, inserte un CD o DVD, luego presione la tecla Enter.

   - Si no se cuenta con Microsoft Windows XP (SP2 o SP3), Vista o 7 (seven), y con la herramienta Image Mastering API v2.0 (IMAPIv2.0), se recomienda usar algún software externo para escribir la imagen cdboot.iso en un CD/DVD vacío con la característica de autoarrancable.

7. Cuando la unidad grabable de CD-ROM se detenga, retire el disco de arranque.

8. Insertar el disco de arranque dentro de la unidad lectora de CD-ROM de la computadora de control y reiniciela.

Cabe mencionar que algunas de las propiedades de configuración del software xPC Target vistas en el paso 3, dependen de la versión del software, del nombre y la ruta del compilador, de rutas de almacenamiento para los archivos creados, así como de los recursos

físicos disponibles en el CPU de la computadora de control. Para mayor información sobre la correcta configuración de estos parámetros puede consultarse [MathWorks, 2010a].

Apéndice D

Configuración de las computadoras de desarrollo y de control

Para poder comunicar las computadoras de desarrollo y de control, correspondientes a los niveles 4 y 3 de la arquitectura de control de la muñeca MASKARA respectivamente, es necesario realizar algunas configuraciones básicas de hardware y de software en ambas computadoras. En el caso de la computadora de desarrollo, se deben de configurar algunos parámetros del puerto de red Ethernet referentes al protocolo TCP/IP, además de configurar también el software de Real-Time Workshop instalado en la computadora de desarrollo. Para el caso de la computadora de control, se deben especificar también algunos parámetros del puerto de red Ethernet y, realizar una configuración de la BIOS del CPU.

D.1. Configuración de la computadora de desarrollo

La computadora de desarrollo necesita tener una dirección IP fija y una máscara de subred conocidas para establecer comunicación vía protocolo TCP/IP con la computadora de control; para esto es necesario realizar una pequeña configuración de red TCP/IP dentro del sistema operativo que se esté utilizando. Por ejemplo, en el caso de la computadora

de desarrollo utilizada en el nivel 4 de la muñeca, se dispone de un sistema operativo Windows XP (32 bits) para el cual se describen los siguientes pasos para configurar la red TCP/IP:

1. Seleccionar el menú Inicio>Configuración>Panel de Control, luego dar doble clic en Conexiones de red.

2. Dar clic derecho en Conexiones de Área Local, luego seleccionar Propiedades.

3. Seleccionar Protocolo de Internet (TCP/IP), luego dar clic en Propiedades.

La figura D.1 muestra un cuadro de diálogo al que se tendría acceso si se llevan a cabo los pasos anteriores. Se recomienda definir una dirección IP fija para la computadora de desarrollo que sea subsecuente a la establecida en la computadora de control, y que la mascara de subred sea la misma en ambas computadoras. Para conocer la dirección IP y la mascara de subred de la computadora de control, se puede consultar la tabla C.1 del apéndice C.

Configuración de Real-Time Workshop

Para poder utilizar la herramienta xPC Target proporcionada por MathWorks Simulink, es necesario configurar algunos parámetros del software de Real-Time Workshop dentro del ambiente de Simulink; configuración referente a la generación de código C/C++ a partir de los modelos construidos con los bloques de Simulink, para su posterior descarga en la computadora de control. Además, es necesario llevar a cabo esta configuración para poder operar de manera correcta el nivel 3 de la arquitectura de la muñeca. Para la configuración de los parámetros de Real-Time Workshop se deben de seguir los siguientes pasos:

1. En la ventana principal del modelo de Simulink, y dentro del menú *Simulation*, dar clic en *Configuration Parameters*. Luego se despliega una ventana de diálogo para la configuración de parámetros.

Figura D.1: Propiedades de conexión de red de la computadora de desarrollo

2. Dar clic en el nodo de Real-Time Workshop para abrir el panel principal de Real-Time Workshop.

3. Buscar el parámetro "System target file" y seleccionar el archivo *xpctarget.tlc* con su configuración por default como se muestra en la figura D.2 y dar clic en *Apply* y luego en *OK*.

4. Un subnodo llamado *xPC Target Options* aparece del lado izquierdo del panel, dar clic en el subnodo y revisar que la configuración por default sea la que se muestra en la figura D.3.

D.2. Configuración de la computadora de control

Al igual que la computadora de desarrollo, la computadora de control necesita establecer los parámetros de dirección IP y la mascara de subred para manejar el protocolo

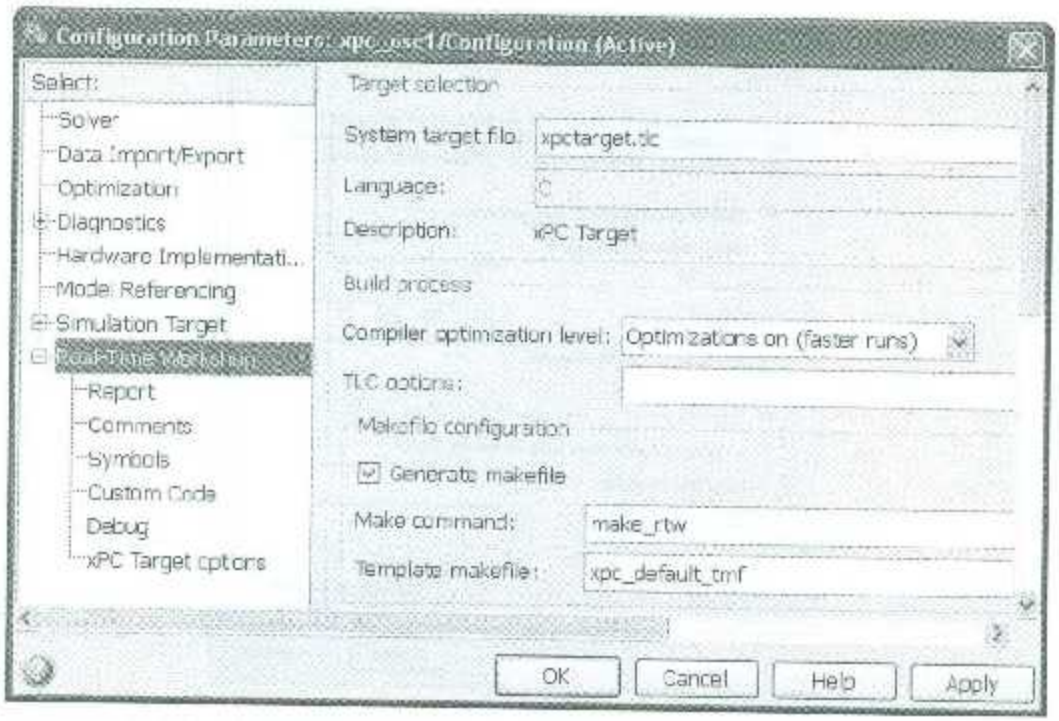


Figura D.2: Panel principal de Real-Time Workshop

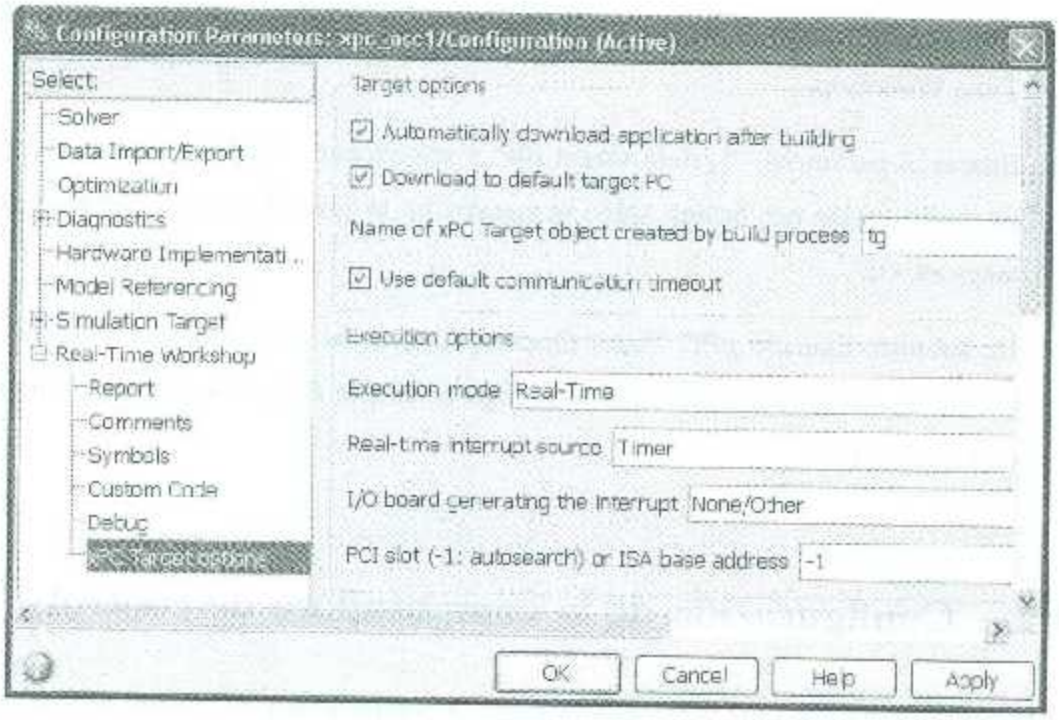


Figura D.3: Configuración de xPC Target

de red TCP/IP; la especificación de estos parámetros se hace directamente en la creación del disco de arranque para modo kernel del CPU de la computadora de control como se menciona en el apéndice C. Además de la configuración de los parámetros de red TCP/IP antes mencionados, la computadora de control necesita configurar la BIOS de su CPU para establecer la secuencia de dispositivos de arranque y para deshabilitar algunas funciones del CPU con el fin de optimizar el funcionamiento de la computadora de control durante la transferencia de datos con el nivel 2 de la muñeca. A continuación se muestra una secuencia general de pasos para llevar a cabo la configuración de la BIOS del CPU de la computadora de control:

1. Encender el CPU y el monitor de la computadora de control. Antes de que el sistema operativo de la computadora de control se inicialice, presionar la tecla <F2>del teclado para entrar a configuración de la BIOS.

2. En el menú superior, dar clic a la pestaña de "Advanced" y después a la opción de "USB Configuration", para seleccionar deshabilitar los puertos USB ("USB Ports [Disable]").

3. Después ir a la pestaña de "Boot" del menú superior para seleccionar la unidad lectora de CD-ROM como primera prioridad en la secuencia de arranque ("Boot Device Priority [CD/DVD-ROM Drive]")

Para mayor información sobre la configuración correcta de la BIOS del CPU de la computadora de control se puede consultar [MathWorks, 2010a].

Apéndice E

Agregando nuevos controladores

Para modificar un controlador existente, es necesario entrar al programa de Simulink, abrir la carpeta de controladores con la ruta correspondiente (por ejemplo, en este apéndice se supone que la ruta es: "D:\Mis Documentos\Pedro ZuFlo\MASKARA\Controladores mdl\") y seleccionar el modelo que almacena el código del controlador que queremos modificar; en el caso de que se desee agregar un controlador nuevo, es necesario crear un nuevo modelo de Simulink que almacene el código del controlador y guardarlo en la carpeta de controladores. Cada controlador corresponde a un modelo de Simulink que está compuesto de un solo bloque de Función Embebida de Matlab; dicho modelo esta dado de alta en la lista desplegable de controladores disponibles en la GUI del programa de control, en el archivo "WristControlGui.m" localizado en "D:\Mis Documentos\Pedro ZuFlo\MASKARA\" y en la carpeta de controladores antes mencionada. Para agregar un nuevo controlador hay que seguir una serie de pasos que se describen a continuación:

Crear un nuevo controlador

1. Abrir Simulink.
2. Ir al menú Archivo, seleccionar Abrir y buscar en "D:\Mis Documentos\Pedro ZuFlo \MASKARA\Controladores mdl\" el archivo "Plantilla_Velocidad.mdl" o "Planti-

lla_Corriente.mdl", para seleccionar abrir uno de ellos dependiendo del modo de operación que se desee utilizar en la muñeca; selecciónese el archivo "Plantilla_Velocidad .mdl" si se desea crear un controlador de consignas de velocidad o selecciónese el archivo "Plantilla_Corriente.mdl" si se desea crear un controlador de consignas de corriente.

3. Guardar el archivo abierto en el paso anterior con otro nombre que represente el nuevo controlador a crear. La ruta en donde se guardará este nuevo controlador es dentro de la carpeta de controladores del paso 1.
   Nota: Es importante no hacer modificaciones al archivo "Plantilla_Velocidad.mdl" o "Plantilla_Corriente.mdl" ya que ambos son los archivos base para crear nuevos controladores.

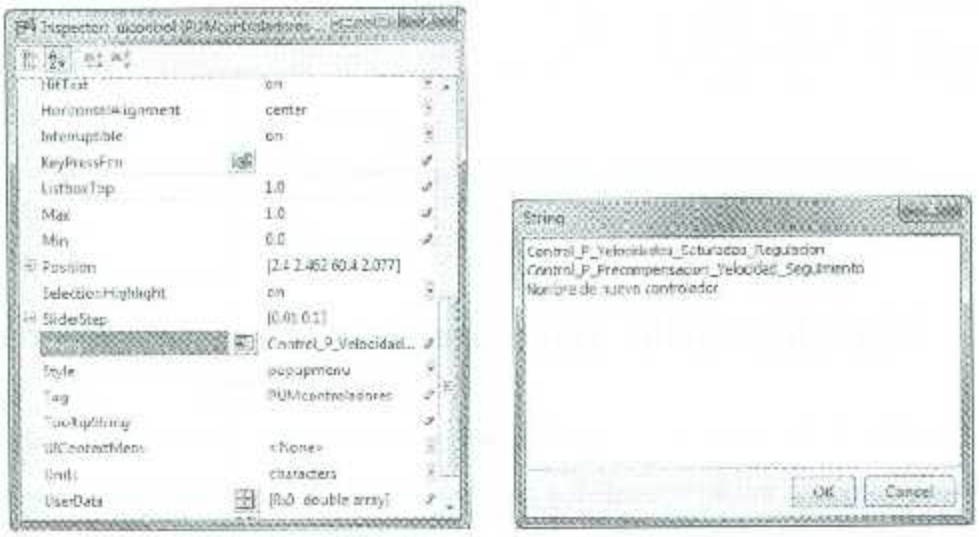
4. Una vez creado el archivo modelo que almacenará el nuevo controlador con el nombre que se asignó en el paso anterior, es posible ahora diseñar la programación del algoritmo de control dentro del bloque de Función Embebida llamado "Ley de control" del archivo modelo del nuevo controlador; si se revisa el código dentro del bloque "Ley de control" se pueden encontrar tres secciones, la primera sección referente a información técnica del controlador, la segunda sección correspondiente a la declaración de variables del controlador y la última sección referente al código del algoritmo de control a utilizar. La primera sección es de especial interés al usuario porque en élla se especifica (a manera de comentario) la descripción de variables contenidas en el código y los límites máximos permitidos para algunas de esas variables propias de cada servomotor de la muñeca. Nota: Es importante que al momento de escribir la programación del nuevo controlador, se tenga cuidado de no modificar las partes de código con la leyenda "*no modificable*". Estas partes de código corresponden a protecciones de límites de variables especificadas en la primera sección del código.

Agregar el controlador al programa de control

Para agregar un nuevo controlador al programa de control, es necesario darlo de alta dentro de la GUI de Matlab del programa de control y dentro del archivo "WristControlGui.m" localizado en "D:\Mis Documentos\Pedro ZuFlo\MASKARA\" como se muestra en los siguientes pasos:

1. Para agregar un controlador en modo corriente pase al paso 3, de lo contrario continue hasta terminar todos los pasos. Abrir el archivo localizado en "D:\Mis Documentos\Pedro ZuFlo\MASKARA\WristControlGui.fig" con la herramienta GUIDE de Matlab.

2. Seleccione el objeto con el nombre "*PUMcontroladores*" y haga doble click izquierdo sobre él; enseguida aparecerá una ventana con el nombre del objeto que muestra las propiedades del mismo (figura E.1a), busque la propiedad llamada "*String*" y agregue dentro de élla el nombre del nuevo controlador en modo velocidad (figura E.1b). Guarde los cambios de la GUI.

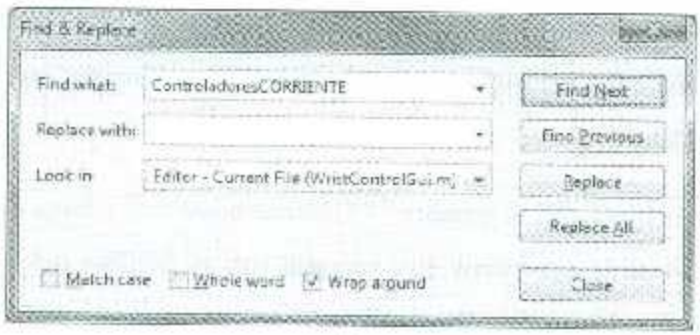


(a) (b)

Figura E.1: Agregando controlador a la GUI de Matlab

3. Abrir el archivo localizado en "D:\Mis Documentos\Pedro ZuFlo\MASKARA\WristControlGui.m" desde Matlab, y a través de la ventana "Buscar y reemplazar" tex-

to (Ctrl+F) buscar la palabra "ControladoresVELOCIDAD" o "ControladoresCORRIENTE" dependiendo del tipo de controlador que se quiera agregar y se presiona "Buscar siguiente" (figura E.2a). En el código de "WristControlGui.m" se resalta la palabra que se buscó, y en esa parte de código es donde se agrega el nuevo controlador (figura E.2b).



(a)

```
% Code for when RBmode_controlador is selected.
%ControladoresCORRIENTE
set(handles.PUMcontroladores,'String',{'PID_Regulacion';'PID_Seguimiento';
% Get the default selection for the controller from the pop-up menu
NewController = get(handles.PUMcontroladores,'Value');
string_list = get(handles.PUMcontroladores,'String');
string_list = cellstr(string_list);%Convierte a cell array
selected_string = string_list(NewController); % Convert from cell array to string
```

(b)

Figura E.2: Agregando controlador al código de la GUI de Matlab

E.1. Variables útiles para programación del controlador

En la tabla E.1 se muestran las principales variables que se usan en los programas de los controladores, en modo velocidad o en modo par, cada una con sus correspondientes dimensiones y unidades.

Para estandarizar las variables más comunes de los controladores, en la tabla E.2 se muestran las variables auxiliares que ya están declaradas en el programa, con sus respectivas dimensiones y unidades.

Tabla E.1: Variables del controlador

Variable de salida del controlador	
Variable	*Descripción*
sC	Vector de 3 señales de control a la muñeca para: consignas de corriente a los servomotores (amperios), o consignas de velocidad a los servomotores (grados/s)
Variables de entrada del controlador	
Variable	*Descripción*
q	Vector de 3 variables articulares de la muñeca (grados)
qp	Vector de 3 velocidades articulares de la muñeca (grados/s)
curr	Vector de 3 corrientes de la muñeca (amperios)

Tabla E.2: Variables auxiliares del controlador

Variable	*Descripción*
qd	Vector de 3 variables articulares deseadas (grados)
qtilde	Vector de 3 errores de posición (grados)
qpd	Vector de 3 velocidades articulares deseadas (grados/s)
qtildep	Vector de 3 errores de velocidad (grados/s)