UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO, CONSTRUCCIÓN Y CONTROL DE ESTABILIDAD DE UN ROBOT QUE SE BALANCEA SOBRE UNA ESFERA

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

E

INGENIERO MECATRÓNICO

P R E S E N T A N

TOMÁS BARTOLOMÉ GARCÍA NATHAN

Y

YUKITOSHI MINAMI SHIGUEMATSU

DIRECTOR DE TESIS

M. I. Yukihiro Minami Koyama

Ciudad Universitaria 2012

Índice general

Índice de figuras

Glosario

ADC	Convertidor Analógico Digital
ARM	Maquina RISC Avanzada
CD	Corriente Directa
DFT	Transformada Discreta de Fourier
DSP	Procesador Digital de Señales
EEPROM	ROM Programable y Borrable Eléctricamente
FFT	Transformada Rápida de Fourier
FIR	Respuesta Finita al Impulso
FPGA	Arreglo de Compuertas Programables
I^2C	Comunicacin Inter-Integrados
IMU	Unidad de Mediciones Inerciales
LCD	Display de Cristal Líquido
LQR	Regulador Cuadr]'atico Lineal
LSB	Bit Menos Significativo
MCP	Módulo de Control de motor a Pasos
MDF	Tablero de Fibra de Densidad Media
MIC	Módulo de Comunicación Interna
MSB	Bit Más Significativo
PCB	Tarjeta de Circuito Impreso
PD	Control Proporcional y Derivativo
PIC	Controlador de Interfaz Periférica

PID	Control Proporcional, Integral y Derivativo
PLL	Lazo de Seguimiento de Fase
PWM	Modulación por Ancho de Pulso
RISC	Computadora con Juego de Instrucciones Reducido
RS-232	Protocolo de Comunicación Estándar Recomendado 232
TTL	Lógica de Transistor Transistor
UART	Receptor/Transmisor Asíncrono Universal
USB	Bus Serial Universal
VCO	Oscilador Controlado por Voltaje

Capítulo 1

Introducción

1.1. ANTECEDENTES

Un robot móvil es una máquina autónoma capaz de moverse en un ambiente específico. Existen robots móviles aéreos, terrestres y acuáticos. Los robots móviles surgen en la Segunda Guerra Mundial con la invención de misiles con sistemas de guía y sencillos modos de piloto automático. Posteriormente se desarrollaron robots móviles terrestres y acuáticos. Se encontraron muchas aplicaciones para los robots terrestres por lo cual esta área de los robots móviles ha sido la más desarrollada, mejorando a cada momento la velocidad, la agilidad y el control del movimiento. Mientras se desarrollaban mejores modelos de robot móvil terrestre, también se encontraron nuevas áreas de oportunidad donde éstos podrían ser usados, lo que permitió que los nuevos diseños fueran más complejos y estuvieran mejor adaptados a necesidades particulares del ambiente en donde se movería el robot.

Uno de estos ambientes que cada vez va tomando mayor importancia es el ambiente humano, para el cual ha sido necesario la reducción del tamaño y del área de la base de los robots. Esta reducción ha sido un proceso largo y que ha pasado por varias etapas, y su finalidad ha sido crear un robot móvil con una base pequeña que permita un movimiento libre y ágil, y que esté mejor adaptado a los angostos y cambiantes espacios que se encuentran en dichos ambientes. Este proceso inició quitando dos llantas en el típico modelo de cuatro llantas, generando así un robot cuya base tenía solamente dos llantas. Un ejemplo es el robot desarrollado por el proyecto YAMABIKO de la Universidad de Tsukuba en Japón, el cual cuenta con solamente dos

llantas y tiene un control basado en un modelo de un péndulo invertido. Para este proyecto se usó un control por retroalimentación de estados, que permitiera no sólo mantener el equilibrio, sino también desplazarse.

El siguiente paso fue construir un robot sobre una sola rueda motriz, es decir, un monociclo. El proyecto YAMABIKO también desarrolló un robot al cual llamó Ichiro. Este robot es un monociclo y se diseñó con la intención de que fuera un robot autónomo, por lo cual cuenta con sensores, actuadores y un módulo de control que le permite mantener el equilibrio y moverse de forma autónoma.

El robot Ichiro, cuenta con un monociclo construido con una rueda con forma elipsoidal semejante a la de un balón de rugby. El cuerpo del robot está separado en dos secciones, las cuales tienen una unión tipo bisagra que permite un mejor control del equilibrio. La rueda se utiliza para el control en la dirección del movimiento, que podría llamarse hacia adelante y hacia atrás, mientras que la estabilidad hacia los lados se logra gracias al movimiento lateral entre las dos secciones del cuerpo del robot. Este robot tiene un control que le permite no sólo mantener el equilibrio, sino también moverse en trayectorias definidas, ya sean rectas o curvas. Esto fue un gran avance para los robots de este tipo. Sin embargo, el robot en monociclo no puede moverse de forma lateral, lo cual es un problema si se requirieran hacer movimientos precisos en poco espacio, como en ambientes humanos interiores, como casas, oficinas, museos y hospitales, entre otros.

Hoy en día la solución de utilizar dos ruedas o un cilindro y modelar el sistema como un péndulo invertido unidimensional es muy utilizada tanto en la industria como en la investigación. Un ejemplo de esto en la industria es el conocido Segway, el cual tiene dos ruedas y mantiene el equilibrio moviéndose hacia adelante o hacia atrás, según sea necesario, manteniendo el centro de gravedad dentro de un intervalo de estabilidad. La reducción del espacio y el uso de sólo dos motores fue un gran avance, tanto económico como en la posibilidad de movimiento en ambientes humanos. Sin embargo, solamente puede moverse hacia adelante o hacia atrás, y al no contar con movimientos laterales, depende de girar en torno a su propio eje para poder moverse en otra dirección y dar vueltas.

Una solución a los problemas anteriormente mencionados, es decir, del uso de un área muy grande para la base de movimiento y la dificultad de moverse en cualquier dirección, es utilizar una sola rueda que permita el movimiento omnidireccional, es decir, una esfera. Si la base de movimiento se encontrara sobre una esfera, el área de la base sería apenas más grande que la ocupada por la máxima área transversal de la esfera. Además, una esfera puede girar hacia cualquier dirección, contando el giro sobre sí misma para cambiar la orientación del robot sin modificar su posición. El problema con la esfera es que el control se dificulta, ya que se debe encontrar el equilibrio dinámico por lo menos en dos dimensiones, y el sistema para dar locomoción a una esfera es un poco más complejo que tener un par de ruedas.

Sin embargo, es precisamente esta complejidad la que convierte a la solu-

ción del movimiento omnidireccional mediante el uso de una esfera en una opción muy interesante y con mucho potencial, por lo cual en el presente trabajo se eligió esta opción para entenderla mejor y poder diseñar y construir un robot móvil sobre una esfera.

1.2. ESTADO DEL ARTE

La idea de un robot que se balancea sobre una esfera ha sido investigada y puesta a prueba en diferentes universidades. El primer modelo fue desarrollado en el año 2006 en la Universidad de Carnegie Mellon, en los Estados Unidos, por los profesores Lauwers, Hollis y Kantor [1] y su equipo. Este modelo fue llamado "ballbot", término que más tarde sería utilizado para designar de forma genérica a los robots con este tipo de locomoción. Ellos usaron mecánica de Lagrange para obtener una ecuación dinámica simplificada del robot considerando solamente la componente viscosa de la fricción y despreciando la componente seca de la misma, tomando como salidas el par de los motores de corriente directa (o de CD) para transmitir el movimiento a la esfera, e hicieron uso de una ley de control LQR (Regulador Lineal Cuadrático, por sus siglas en inglés) para lograr la estabilización y un control básico sobre la posición del robot moviéndose en línea recta. Sin embargo, este tipo de control no pudo resolver la fricción resultante en la superficie de contacto entre la esfera y el terreno. Además, inicialmente tampoco contaban con un control para el giro alrededor de un eje perpendicular al piso (*yaw*).

También en el 2006, en la Universidad de Tohoku Gakuin, en Japón, Kumagai y Ochiai [2] propusieron un modelo orientado al transporte de carga y funciones cooperativas. Éste cuenta con tres ruedas omnidireccionales, cada una sobre un motor a pasos para trasmitir el par a la esfera. El uso de las ruedas omnidireccionales les permitió controlar también el *yaw* sin la necesidad de otro actuador. El uso de los motores a pasos redujo costos e hizo más sencillo el control en el software, el cual se hizo con base en ecuaciones centradas en comandos de aceleración, lo cual puede hacer más robusto al control con respecto a uno basado en el par aplicado. La inclinación del robot se obtuvo mediante un arreglo de acelerómetros y giroscopios, cuyas señales fueron combinadas mediante un filtro digital de primer orden para obtener una señal de entrada para el algoritmo de control. Para la esfera se utilizó inicialmente un balón de basquetbol pero tuvo fallas debido a que se comprimía, por lo que al final se utilizó una bola de boliche. Las constantes del control se obtuvieron experimentalmente y el robot mostró estabilidad con una carga adicional de hasta 10 kg.

También en Japón, en la Universidad de Tokio, se desarrolló un ballbot que pretendía servir como transporte personal utilizando un balón de básquetbol como rueda omnidireccional llamado B. B. Rider (BasketBall Rider) [3]. Para este ballbot se diseñaron y construyeron un asiento , controladores y mecanismos de transmisión de par especiales para los motores

de CD de tal forma que fueran capaces de soportar y trasladar a una persona promedio. Además, llevaron a cabo el análisis para la obtención de la velocidad angular y el par necesario en los motores por medio de un sensor de fuerza-par de seis ejes. Sin embargo, no llegaron a proponer algún tipo de control.

En la Universidad Nacional de Ching Hsing, en China, Liao, Tsai, Li y Chan [4] crearon un nuevo modelo del robot con dos motores sin escobillas y con un control por modos deslizantes para estabilidad y posicionamiento, con la intención de resolver los problemas con la fricción. El modelado del sistema lo realizaron con base en las ecuaciones de Euler-Lagrange considerando al robot como un cilindro de 1.5 m de altura con 300 mm de diámetro y 45 kg, sobre una esfera de 220 mm de diámetro y 7.264 kg. Además, hicieron uso de las transformaciones de ángulos de Euler para las variaciones entre las coordenadas del cilindro y las de la esfera. Despreciaron tanto la fricción como el deslizamiento entre la esfera y el cilindro y el suelo. Como ya se mencionó, diseñaron un control por modos deslizantes jerárquico, con un lazo interno para el control angular y de posición, y un lazo externo para el control del par.

Nagarajan, Mampetta, Kantor y Hollis [5] propusieron después un modelo con cuatro servomotores de CD para la transmisión del par a la esfera, debido a que con sólo dos se generaban fuerzas verticales. Además, utilizaron un servomotor más para el *yaw*. Cabe mencionar que los servomotores contaban con un *encoder* absoluto. Su modelado también se basaba en las ecuaciones de Euler-Lagrange, pero ellos sí modelaron la fricción tanto de Coulomb como la viscosa, y obtuvieron los coeficientes de forma experimental, así como los momentos de inercia. Asimismo, se le colocaron al robot tres patas retráctiles para mantenerse totalmente quieto. El control del balance se llevó a cabo mediante un control proporcional, integral y derivativo, o PID por sus siglas en inglés, en cada dirección que controlaba el ángulo de inclinación obtenido de la IMU (Unidad de Medición Inercial, por sus siglas en inglés). Por otra parte el control para mantenerse en el mismo lugar mientras se balancea se llevó a cabo con un control proporcional derivativo, o PD, con la ayuda de los *encoders* absolutos.

En el 2009 fue desarrollado un ballbot como proyecto estudiantil en la Universidad de Adelaide, en Australia [6], en donde primero se construyó un modelo funcional utilizando LEGO© Mindstorms© NXT y posteriormente, utilizando la teoría aplicada a dicho modelo, se construyó un ballbot en tamaño real. Para éste, utilizaron inicialmente el "ladrillo programable" de LEGO© para controlar al robot, cambiando a electrónica de potencia y utilizando servomotores de CD con escobillas para tener el par necesario para mover el modelo en tamaño real. Posteriormente, al no poder el "ladrillo programable" comunicarse con dispositivos seriales, cambiaron al uso de un microprocesador para el procesamiento de los datos obtenidos de una IMU y de *encoders* en los motores, así como para implementar el control del robot, cuyo método fue un LQR con una parte integral para corregir el error en

estado estable.

El ballbot más reciente fue desarrollado en la Escuela Politécnica Federal de Zurich, en Suiza, y fue llamado Rezero. Este último ballbot tuvo como finalidad ser más ágil, teniendo mayor velocidad y precisión de movimiento que los anteriores.

En la Tabla 1 se presentan las características de los diferentes proyectos mencionados.

Tabla 1.1: Características de los Ballbots existentes

Universidad	**Carnegie Mellon**	**Tohoku Gakuin**	**Tokio**	**Chung-Hsing**	**Adelaide**	**Politécnico Federal de Zurich**
Año	2006∼2010	2006∼2008	2006	2008∼2010	2009	2009∼2010
Masa, en kg	54	12.5	25.5	48.6	34.5	14.5
Altura, en m	1.5	0.5	0.63	1.5	1.6	∼1
Velocidad máxima, en m/s	0.1	1.1	-	-	0.1	3.5
Inclinación máxima	$< 1°$	$< 5°$	-	$< 2°$	$< 1°$	$< 20°$
Motores	Servomotores de CD	A pasos	CD	Sin escobillas	CD	Sin escobillas
Control	LQR/PID	PD	-	Modos Deslizantes, PI/LQR	LQR+I	LQR

Se puede observar que en todos los proyectos mencionados el sistema es modelado como un péndulo invertido bidimensional, con base en el cual se diseñaron los diferentes tipos de controladores que al final se aplicaron en dos ejes perpendiculares para obtener el control tridimensional del sistema.

Cabe mencionar que en la mayoría de estos proyectos, se utilizaron controladores que tenían como salida el par de los motores, siendo el único diferente el de la Universidad de Tohoku Gakuin, el cual utilizaba como salida la aceleración de los motores.

1.3. OBJETIVO

En los últimos años se ha visto una creciente tendencia de crear robots capaces de interactuar con el ser humano y esta convivencia ha requerido la adaptación de los robots a moverse en ambientes transitados por personas. Es por esto que es de gran relevancia crear un sistema base de movimiento para robots, diseñado específicamente para el movimiento e interacción en

espacios humanos. Por todo lo anterior, se decidió que el modelo de robot móvil ballbot es el mejor adaptado a las necesidades de movimiento en este tipo de espacios, por lo que **el objetivo de la presente tesis será el diseño, construcción y control de estabilidad de un robot que se balancee sobre una esfera que pueda servir en un futuro como una base móvil adaptable para diversas aplicaciones.**

1.4. RESUMEN DEL TRABAJO

El trabajo se divide en seis capítulos. En el capítulo de *Análisis* se describen grosso modo todos los posibles dispositivos a utilizar para el funcionamiento del sistema, tanto de la parte mecánica como de la parte electrónica y de control. Luego, en el capítulo de *Modelado*, se realiza el análisis dinámico del sistema con base en el método de Euler-Lagrange. En *Primer Prototipo* se describe al primer robot construido, desde los dispositivos seleccionados, hasta las pruebas que se realizaron con éste y los resultados obtenidos, así como y los problemas surgidos con este primer robot. Posteriormente, en *Segundo Prototipo* se describe cómo se resolvieron los problemas del primer prototipo, las pruebas reaizadas con este segundo robot y los resultados que se otuvieron con los cambios realizados. Finalmente, en *Conclusiones y Trabajo a Futuro* se habla de los resultados generales obtenidos, así como un poco acerca de la experiencia adquirida con este proyecto y los posibles cambios y mejoras que se pueden realizar para un mejor desempeño del robot.

Capítulo 2

Análisis del Problema

El Ballbot es una combinación de varios sistemas trabajando en conjunto para su óptimo funcionamiento, por lo que fue necesario analizar cada sistema tanto individualmente como trabajando en conjunto con el objeto de poder tomar las mejores decisiones posibles para el diseño del robot.

2.1. SISTEMA DE MEDICIÓN

Para controlar al ballbot es necesario conocer su inclinación con respecto a la vertical. Además podría ser útil, aunque no necesario, el conocer la inclinación de la esfera, o dicho de otro modo, la posición espacial del ballbot. Existen varias formas de obtener esta información.

2.1.1. Acelerómetro

La medición de la aceleración es sin duda de gran importancia para un sinnúmero de aplicaciones electrónicas, es un elemento central de los sistemas guiados por inercia. Este sensor tiene un amplio rango de uso tanto en la industria como en aplicaciones comerciales, desde la detección de choques en vehículos para el accionamiento de bolsas de aire, hasta la provisión de señales de retroalimentación para mantener fija una imagen en una grabadora de vídeo en modo manual. El hecho de que la medición de la aceleración sea una problemática común ha generado una amplia variedad de soluciones para llevar a cabo dicha medición en diversas circunstancias, como son mediante la medición en el cambio de capacitancia, masas con cristales *piezoeléctricos* que al aplicarles una fuerza causan cambios de voltaje o *piezoresistivos* que cambian su resistencia, la medición del cambio del campo magnético mediante el efecto Hall, la ubicación de una masa caliente mediante sensores de temperatura, entre otros.

2.1.2. Giroscopio

Éste es un dispositivo capaz de medir la aceleración rotacional de un cuerpo, con lo cual se puede obtener también la velocidad angular e incluso la inclinación con respecto a algún eje del mismo. Existen varios tipos de giroscopios, como son los mecánicos que consisten en una rueda giratoria montada sobre una suspensión de Cardán, los ópticos como el de fibra óptica o el de anillo láser que funcionan gracias al efecto de Sagnac, los de "momento London" que se basan en un fenómeno de mecánica cuántica, los de estructuras vibratorias que se basan en un oscilador que genera un momento angular o lineal constante acoplado a un sensor de aceleración de Coriolis, entre otros.

2.1.3. *Encoder*

El *encoder* es un dispositivo que convierte movimiento ya sea lineal o rotatorio, en una señal digital. El más común suele ser óptico, que utiliza sensores infrarrojos y un disco giratorio o tira móvil con secciones que dejan pasar el haz de luz y secciones que impiden el paso de éste.

El *encoder* óptico se clasifica en incremental y absoluto. El primero da información acerca de la velocidad angular o lineal de un objeto, mas no su posición exacta ni su dirección. Para obtener esta información es necesario realizar algunos ajustes, como el tener una ubicación de inicio o *home* para estimar la posición, la colocación de más de un sensor óptico para la dirección, etc. El *encoder* absoluto, en cambio, puede dar información tanto de la velocidad como de la posición y la dirección del objeto en cuestión, aunque se necesitan más sensores ópticos y suelen ser más grandes.

Además del *encoder* óptico, existe también el que utiliza el efecto Hall para su funcionamiento. Éste tiene la ventaja de ser más barato y poder ser utilizado en ambientes más extremos, ya que al funcionar con campos magnéticos, no importa si hay polvo, suciedad, etc., que en el caso del óptico sí causaría problemas.

2.1.4. Cámara de vídeo

Otra posible solución es el uso de una cámara de vídeo que pudiera grabar desde diferentes ángulos al robot, de manera que se pudieran obtener los datos de posición del robot mediante procesamiento de imágenes. Este método es problemático debido a que dicho procesamiento de imágenes requiere muchos recursos computacionales, y obtener una buena frecuencia de muestreo sería difícil, además de que el robot sólo podría funcionar dentro del espacio cubierto por dicha cámara.

2.2. SISTEMA DE PROCESAMIENTO

Obtenida la información de la inclinación y posición del robot, es necesario procesar esta información en tiempo real para que se logre controlar al robot. Para esto existen diversos dispositivos que podrían emplearse dependiendo del tipo de datos utilizados, velocidad o funciones necesarias, etc.

2.2.1. Microcontrolador

Es un circuito integrado programable, capaz de ejecutar órdenes almacenadas en memoria. Un *microcontrolador* es un circuito integrado que ya cuenta con un CPU, memoria y unidades de entrada/salida, todo dentro del mismo chip. Dicho de otra forma, es casi una computadora dentro de un chip. Se puede encontrar con un gran variedad de tamaños de memoria, velocidad de procesamiento, uso de palabras de distinta longitud de bits, etc. En la actualidad el *microcontrolador* es extensamente utilizado en los campos de la ingeniería electrónica y es utilizado para diversas aplicaciones.

2.2.2. PIC

El PIC, Controlador de Interfaz Periférica, por sus siglas en inglés, es una familia de *microcontroladores* tipo RISC (del inglés Reduced Instruction Set Computer, Computadora de Conjunto Reducido de Instrucciones), fabricados por la compañía Microchip Technology Inc. La gran variedad de microcontroladores PIC en precio y características, le ha dado una gran difusión y control del mercado. Actualmente el PIC cuenta con adaptaciones para ser utilizado con comunicación Ethernet, USB (por sus siglas en inglés de Universal Serial Bus), I^2C (del inglés Inter-Integrated Circuit), SPI (del inglés Serial Peripheral Interface), entre otras, además de tener archivos de biblioteca para pantallas de cristal líquido o LCD (del inglés Liquid Crystal Display), módulos de modulación de ancho de pulso o PWM (por sus siglas en inglés de Pulse Width Modulation), convertidores analógico a digital, entre otros.

Actualmente este *microcontrolador* está cambiando el uso de memoria: de utilizar EEPROM anteriormente, ahora se empieza a utilizar tecnología Flash. La compañía está haciendo desarrollos de PIC para entrar a los mercados de la tecnología inalámbrica y de procesamiento de señales. El nuevo dsPIC cuenta con todas las funciones de un PIC más algunas propias de los DSP, como son el *barrel shifting* (desplazamiento circular de los bits de una palabra), *bit reverse* (reversión de bits), multiplicación de dos números de 16 bits, entre otras.

2.2.3. Arduino

El Arduino es una plataforma de hardware libre basado en *microcontroladores* Atmel, que cuenta con su propio entorno de desarrollo programado mediante un lengaje propio de alto nivel llamado *Processing*, aunque es posible utilizar otros lenguajes como Java, C++, Python, entre otros. Su mayor ventaja es la facilidad de uso y su versatilidad, ya que cuenta con una interfaz muy interactiva e intuitiva y una misma tarjeta puede manejar datos analógicos, digitales, salidas PWM, etc., pero dependiendo del modelo puede llegar a tener algunas restricciones como menor número de entradas/salidas, velocidades de procesamiento más baja, entre otras.

2.2.4. FPGA

Sus siglas vienen del inglés *Field Programmable Gate Array*, es un dispositivo semiconductor que tiene la posibilidad de ser programado de manera que los bloques lógicos que tiene pueden variar su interconexión y funcionamiento. Gracias a esta posibilidad de cambiar sus interconexiones, se puede programar en el FPGA desde un circuito de lógica *combinacional* sencillo hasta complejos sistemas en un solo circuito integrado. La posibilidad de ser reprogramado hace que el tiempo de diseño e implementación sea mucho menor en un FPGA que en la construcción de un circuito similar haciendo uso de bloques lógicos en circuitos integrados, por lo cual es una excelente opción para sistemas de desarrollo.

Actualmente se están desarrollando chips FPGA que también incluyen un núcleo de procesador de *microcontrolador* embebido dentro de la lógica FPGA, logrando crear sistemas programables en un integrado. Gracias a esto se pueden modificar partes del circuito sin modificar el procesador, es decir, se intenta llegar a los ideales de la computación reconfigurable. La programación de los FPGA suele utilizar sistemas HDL de las siglas en ingles de *Hardware Description Language*; entre los más utilizados se encuentran VHDL, Verilog y ABEL.

2.2.5. DSP

El DSP, Procesador de Señales Digitales por sus siglas en inglés, es un dispositivo que tiene un procesador o microprocesador con un hardware y software diseñados de tal modo que puede llevar acabo operaciones matemáticas a muy altas velocidades. Algunas veces cuenta con convertidores analógico a digital dado que las señales se procesan de forma digital pero pueden entrar en forma analógica. Gracias a su gran velocidad, se pueden tomar un gran número de muestras y éstas pueden ser procesadas en tiempo real.

El DSP suele ser utilizado principalmente para aplicaciones de audio y vídeo. Es común ver implementado en él filtros de diversos tipos. Gracias a características como el *barrel shifting*, el *bit reverse* y el *multiply-accumulate* o *MAC* que es la posibilidad de multiplicar dos acumuladores y sumarlos en un solo ciclo de reloj, se pueden llevar a cabo procesamientos complejos en tiempo real.

2.3. SISTEMA DE CONTROL

Una vez procesados los datos de los sensores, se puede utilizar la información obtenida para controlar al robot. Para esto, existen diversos tipos de métodos de control para que el robot haga lo que se desea, en este caso, mantener el balance sobre la esfera.

2.3.1. PID

Este método de control denominado Proporcional Integral Derivativo, consiste en retroalimentar una señal para corregir el error existente entre ella y un valor deseado. La señal correctora consta de tres partes: la parte proporcional, que genera una reacción al error, la parte derivativa que aumenta o disminuye la velocidad de la reacción al error, y la parte integral, que elimina el llamado error en estado estacionario. La parte proporcional resulta de la multiplicación del error por una constante K_P, la parte derivativa de la multiplicación de la derivada del error por una constante K_D y finalmente la parte integral resulta de la multiplicación de la integral del error por una constante K_I. No siempre es necesario utilizar las tres partes, puede haber sistemas en los cuales sea suficiente un control PD, PI, entre otros.

2.3.2. Retroalimentación de estados

Este método se basa en retroalimentar los estados de un sistema, obtenidos mediante sensores u observadores, y generar una señal correctora basada en dichos estados multiplicados por constantes, las cuales se obtienen teéricamente, con base en una respuesta deseada del sistema, y lo que hace el control es mantener a todos los estados en el valor de referencia deseado.

Para lo anterior es necesario expresar al sistema en variables de estado y ya sea que se trabaje con un sistema lineal, o en caso de ser no lineal es necesario entonces *linealizar* al sistema, para de ahí determinar si el sistema es controlable y en caso afirmativo obtener las constantes necesarias.

2.3.3. LQR

El control LQR, Regulador Cuadrático Lineal, por sus siglas en inglés, es un método de control que optimiza una función de costo del sistema, la cual es una función que penaliza el "mal" desempeño de éste, aumentando de valor cuando el sistema se aleja de las condiciones deseadas y disminuyendo en caso contrario. Esta función de costo está basada en las matrices de peso, Q y R, las cuales proporcionan la información del costo de los estados y las salidas.

En general es un método similar al de retroalimentación de estados, por lo que también está basado en un análisis del sistema en variables de estado, sólo que las constantes necesarias para generar la señal correctora se obtienen mediante la resolución de la ecuación diferencial de Riccati. Al ser un método óptimo, el funcionamiento del sistema resulta muy bueno, pero la dificultad en el uso de este método radica en la determinación de las matrices de costo, las cuales dependen de lo que se desee y de la experiencia en el uso de este control.

2.3.4. Modos deslizantes

Este es un tipo de control no lineal cuya señal de control es discontinua y cambia deliberadamente de una estructura continua a otra durante el proceso de control, dependiendo del estado del sistema. A este tipo de sistemas se les llama de estructura variable, y los modos deslizantes son uno de los métodos que permiten el cambio de la ley de control entre dichas estructuras.

Este tipo de control tiene ventajas como insensibilidad a perturbaciones externas y robustez, pero para su aplicación es necesario el conocimiento de bases matemáticas específicas para este método.

2.3.5. Lógica difusa

Este es un tipo de lógica *multivariable* que hace uso de conjuntos difusos, los cuales se refieren a conjuntos con fronteras no tan definidas, como por ejemplo la temperatura de algo caliente, la velocidad de un objeto lento, etc. Estos conjuntos se expresan matemáticamente mediante las llamadas "funciones de membresía", las cuales asignan un valor entre 0 y 1 a las diferentes variables según qué tanto entren dentro de un conjunto dado.

Para un sistema de control con lógica difusa, se mapean tanto los valores de entrada como los de salida a conjuntos difusos y se diseñan reglas de inferencia de la forma "SI x Y/O y, ENTONCES z", las cuales son la base del control. El sistema, entonces, al recibir una entrada, la convierte a un valor

difuso, y dependiendo del las reglas obtiene un valor de salida difuso, el cual se convierte en un valor utilizable antes de entregarla a la salida.

2.3.6. Inteligencia artificial

En lugar de un método común de control también podría utilizarse algún algoritmo de inteligencia artificial, como redes neuronales, cadenas de Markov, redes bayesianas, aprendizaje Q, etc. El detalle con varios de estos métodos es que al ser de aprendizaje, es necesario primero "entrena" al robot a actuar de forma deseada, y poco a poco éste irá mejorando sus decisiones para llegar al ideal deseado. Pero para "entrenar" al robot sería necesario poder controlarlo a control remoto y tener a una persona muy hábil que logre las acciones deseadas, lo cual en el presente caso resulta bastante complicado.

2.4. SISTEMA DE LOCOMOCIÓN

Finalmente, procesada la información y obtenida la salida del control, es necesario mandar las señales adecuadas a los actuadores para que el sistema físico funcione correctamente.

Una de las características del ballbot es que tiene movimiento omnidireccional dado que se traslada sobre una esfera, pero existen varias formas de conseguir que el robot tenga este movimiento sobre la esfera.

2.4.1. Motor de corriente directa

El motor de corriente directa, o motor de CD, es el más común; al pasar la corriente a través de una bobina se hace girar al rotor. La dirección del giro del rotor depende de la dirección de la corriente que pasa por la bobina, por lo cual se puede controlar la dirección del giro si se controla la dirección de la corriente utilizando un circuito como un puente H.

La velocidad de giro de este tipo de motores se puede controlar mediante el uso de una señal de PWM. El control de la velocidad mediante PWM para un motor de CD consiste en controlar la energía que se le suministra al motor, sin embargo existe un límite en la velocidad mínima alcanzable, ya que si se reduce más la energía aportada al motor, éste no sería capaz de vencer su propia inercia rotacional. Además se pueden utilizar juegos de engranes o moto-reductores para modificar el intervalo de velocidades que el motor puede alcanzar, ya sea para disminuir o aumentar dicho intervalo, sin embargo este método introduce un *backlash*, también llamado juego o huelgo, el cual es la distancia mxima que se puede mover un diente de

un engrane antes de mover al diente del engrane al cual está acoplado. Esto es una desventaja ya que resta precisión al movimiento y es difícil de modelar.

2.4.2. Motor sin escobillas

Es un motor que usa corriente directa para mover al rotor, pero a diferencia del motor de CD no hace uso de conmutación mecánica sino que ésta se lleva acabo de forma electrónica, es decir, no usa las escobillas que utilizan los motores de CD y de ahí su nombre. Entre las ventajas de no hacer uso de conmutación mecánica mediante escobillas se encuentran generar más par con respecto al peso del robot, tener mayor eficiencia, generar menor ruido, mayor tiempo de vida, generar menor interferencia electromagnética, entre otros. El costo extra es la necesidad de circuitos electrónicos más complejos para ponerlo en operación.

A diferencia de los motores de CD, los motores sin escobillas son mucho más difíciles de controlar, dado que requieren de circuitos más complejos. Para determinar la corriente necesaria en cada fase, es importante conocer la posición del eje del motor. Para esto muchos controladores de motores sin escobillas hacen uso de sensores magnéticos de efecto Hall, aunque otros usan mediciones de la fuerza contra-electromotriz para conocer la posición del eje sin necesidad de sensores extra. Los controladores más complejos hacen uso de microcontroladores, y permiten controlar los límites de voltaje de funcionamiento, la aceleración, el frenado y los cambios de dirección.

2.4.3. Motor a pasos

El motor a pasos cuenta con una conmutación electrónica similar a la del motores brushless, sin embargo no está diseñado para girar libremente, sino con la intención de que el rotor se pueda detener en puntos preestablecidos. En un motor de pasos se puede controlar el movimiento entre puntos preestablecidos, con lo que se puede girar el rotor sólo en ángulos deseados. El control de la posición final es relativa a la posición inicial del rotor y no absoluta.

La posibilidad de mover el rotor en ángulos conocidos permite un control para el motor a pasos de lazo abierto. La característica de poder detenerse en puntos preestablecidos le permite al motor a pasos la posibilidad de girar con velocidades angulares muy bajas sin tener los problemas que tienen el motor de CD o el motor sin escobillas, es decir, el motor a pasos puede girar tan lentamente como sea necesario, sin limitaciones de velocidad mínima. Su inconveniente es que la electrónica necesaria para controlarlo es un poco más compleja que la del motores de CD o sin escobillas.

2.4.4. Servomotor

El servomotor es un motor que cuenta con un control interno de posición absoluto del rotor, de manera que se puede controlar a qué posición final se quiere colocar al rotor sin la necesidad de conocer la posición actual. El rango de movimiento de un servomotor es cuando más de una revolución y está diseñado para aplicaciones en las que se requiere precisión y posiciones establecidas. La señal de control utilizada en el servomotor es una PWM, en la cual dependiendo del ciclo de trabajo de la señal es la posición a la que irá el rotor. De esta manera, modificando el ancho de pulso se puede controlar la posición del servomotor de manera absoluta.

La velocidad en la que el servomotor responde y se mueve entre un punto y otro es siempre la misma. Sin embargo, se puede trabajar con perfiles de velocidad con la intención de tener el control de movimiento del servomotor. Este motor puede ser modificado de manera que pueda girar libremente y no sólo moverse a posiciones establecidas. Esta modificación es común dado que el servomotor tiene un buen par con respecto a su tamaño y precio.

2.4.5. Rueda normal

Además del motor, otro componente de suma importancia para la locomoción es la rueda. Ésta, dependiendo de cómo se transmita el movimiento omnidireccional a la esfera, es decir, dependiendo de cómo estén acomodados los motores, puede ser de diferentes tipos. La rueda que se puede denominar normal o común suele tener el perímetro recubierto con polímeros que tienen un alto coeficiente de fricción. Las variaciones en este tipo de ruedas radican en los materiales, el tamaño y el tipo de orificio para el eje con el que cuentan.

2.4.6. Rueda omnidireccional

Estas ruedas consisten en una rueda común con rodillos colocados de forma perpendicular al movimiento normal de la rueda, de tal forma que al existir movimientos laterales, estos rodillos giran, evitando la fricción que surgiría si fuera una rueda común y corriente. Por lo tanto, este tipo de ruedas pueden moverse tanto en el sentido normal de giro de la rueda, como en sentido perpendicular a éste.

2.4.7. Rueda sueca

Llamadas así por su creador de origen sueco Bengt Ilon, y también llamada ruedas mecanum, por la compañía donde trabajaba su creador, Mecanum AB, es una rueda omnidireccional compuesta por una rueda común con una

Figura 2.1: Rueda Omnidireccional.

serie de rodillos acoplados a su circunferencia, los cuales tienen un eje de rotación a 45° con respecto al plano de la rueda.

Figura 2.2: Rueda Omnidireccional Sueca.

2.5. MECÁNICA DEL ROBOT

En los párrafos anteriores se describieron a grandes rasgos los posibles dispositivos a utilizar para poder implementar las diferentes funciones del robot. Sin embargo, es necesario saber cómo se combinarán los diversos posibles dispositivos, tanto espacial como funcionalmente, para el mejor funcionamiento posible del sistema.

Por la parte del acomodo espacial, primero se analizarán los modelos de posibles cuerpos del robot que optimizarán el acomodo de los diferentes dispositivos para que tengan un tamaño apropiado para moverse en ambientes humanos y además soporten el peso de otros mecanismos que se pongan encima para darle más funciones.

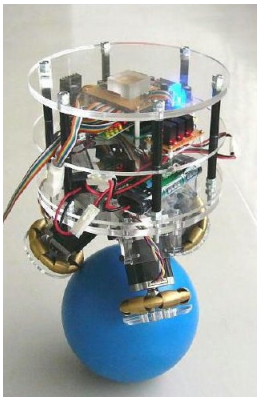
2.5.1. Configuración del Cuerpo

El cuerpo es la pieza central del robot que contiene toda la electrónica y a la que los motores se montan. Además, es la parte más grande de todo

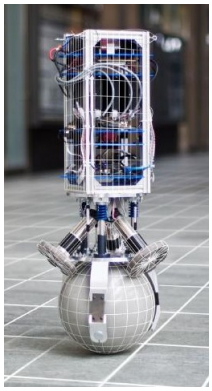
el robot, por lo que es la parte que establece la pauta de la configuración mecánica de toda la estructura, y por ende, es la primer parte que se requiere analizar.

Con la intención de que el cuerpo del robot sea alto pero con una base de área reducida para asegurar una buena movilidad y agilidad dentro de espacios humanos, se tienen básicamente dos posibles formas de cuerpo: cilíndrico y prismático de base cuadrada.

Ambos diseños son altamente simétricos, lo cual es de gran ayuda para la determinación de parámetros como momentos de inercia y centros de gravedad, y además es posible hacer un análisis bidimensional y posteriormente aplicarlo independientemente en dos planos verticales perpendiculares entre sí para obtener el análisis tridimensional del sistema.



(a)



(b)

Tabla 2.1: Formas del cuerpo: (a)cilíndrico y (b)rectangular

En cuanto a la parte funcional del cuerpo del robot, su principal función es alojar a los diferentes sistemas del robot, como son la electrónica, los sensores, la batería y los circuitos de manejo de los motores, además de servir de soporte para los motores. Para esto, se tienen dos formas usuales de acomodar los elementos de los sistemas mencionados: efectuarlo en pisos separando así la electrónica en partes, o bien en las paredes suponiendo un cuerpo cerrado, o pegados a un eje central en el cuerpo del robot. El acomodo que hace uso de pisos requiere también una estructura que los mantenga en su lugar, para lo cual se pueden usar espárragos, o bien, columnas entre pisos en forma de "C". Esto deja cuatro posibles configuraciones de cuerpo, variando la forma de la base y el sistema de separación de pisos en caso de usarlo.

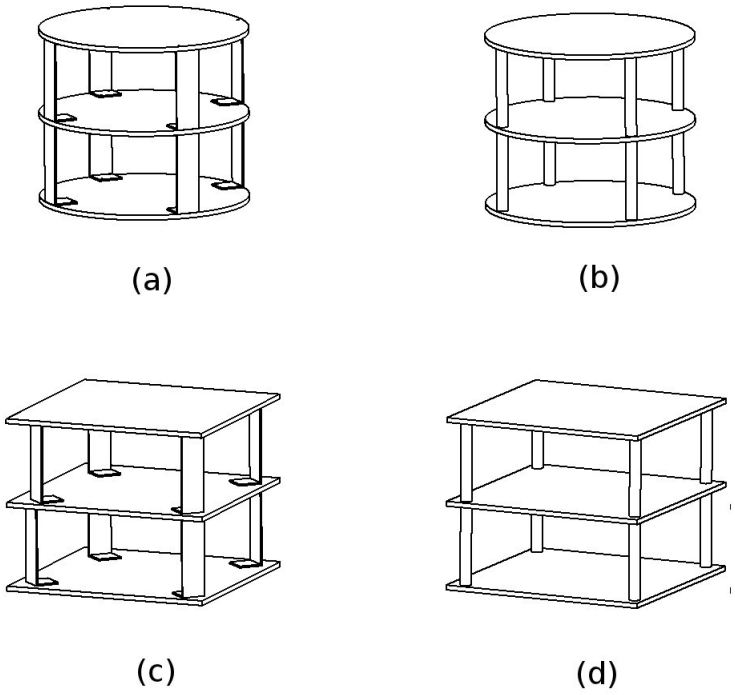


Figura 2.3: Las cuatro posibles estructuras de cuerpo: (a) circular con columnas en forma de "C", (b) circular con espárragos, (c) cuadrada con columnas en forma de "C", (d) cuadrada con espárragos.

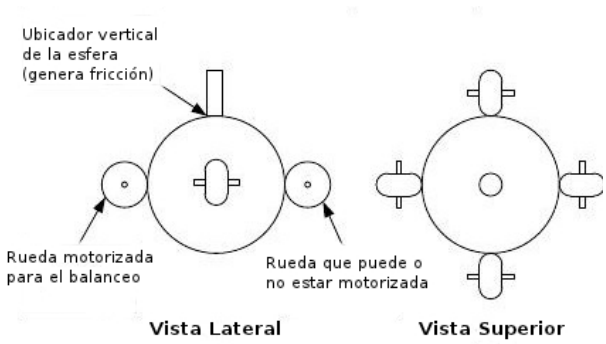
2.5.2. Configuración de los motores y ruedas

Con base en las configuraciones de cuerpo anteriores, el siguiente paso es analizar las posibles formas de transmitir el movimiento necesario a la esfera mediante motores. Esta configuración se puede estudiar en tres partes: el número de motores y ruedas a usar, el acomodo de los motores y las ruedas o los rodillos y el uso de transmisión o acople directo de las ruedas al eje según sea necesario y posible. Con base en sistemas ya existentes, se tienen las opciones mostradas en la Figura 2.2.

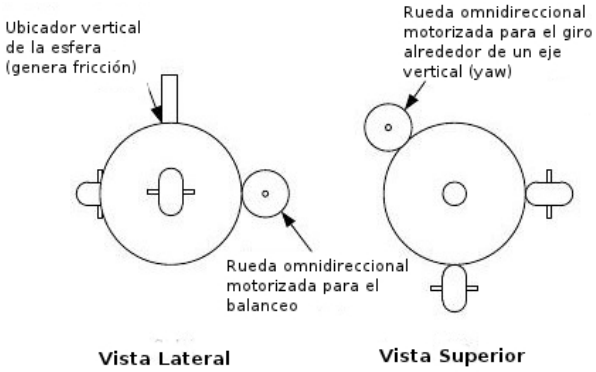
Cabe destacar que en todos los casos se pueden usar ya sean ruedas o rodillos, estén éstos acoplados directamente al motor o por medio de una transmisión para enviar la potencia de los motores colocados en lugares más alejados.

2.6. ACCESORIOS

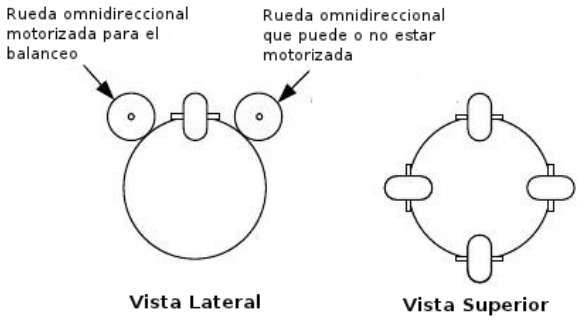
También es posible agregar otros subsistemas que podrían ser útiles, a pesar de no ser cruciales para el correcto funcionamiento del robot. A con-



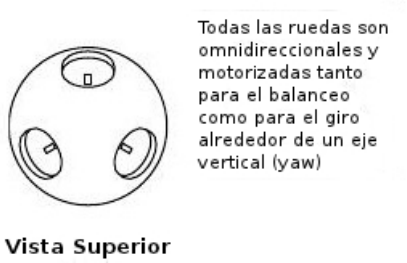
(1) Ruedas ortogonales fijas al eje central de la esfera, adaptado de Lauwers (2006).



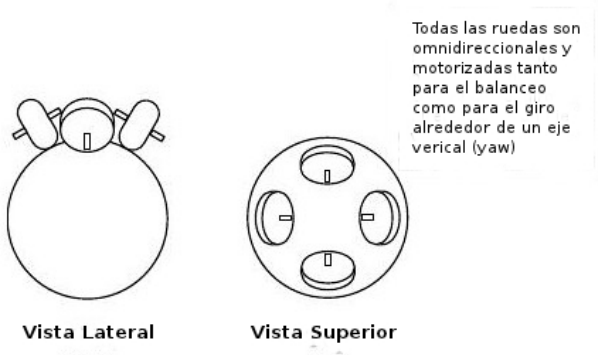
(2) Ruedas omnidireccionales ortogonales sobre el eje central de la esfera.



(3) Ruedas omnidireccionales ortogonales por encima del eje central de la esfera, adaptado de Wu y Hwang (2008).



(4) Tres ruedas omnidireccionales por encima del eje central d la esfera, adaptado de Kumagai y Ochiai (2008).



(5) Pares ortogonales de ruedas por encima del eje central de la esfera.

Tabla 2.2: Opciones de acomodo de motores y ruedas.

tinuación se presentan algunas opciones.

2.6.1. Patas de soporte

Estas patas se podrían colocar para el caso en que se requiera que el robot se mantenga totalmente estático en un lugar por un periodo más o menos largo de tiempo, evitando procesamiento y control innecesarios y ahorrando así energía.
En la Figura 2.4 se puede observar un ejemplo de estas patas ya aplicadas a un modelo real.

Figura 2.4: Ballbot con patas de soporte para equilibrio estático.

2.6.2. Porta-esfera

Para asegurar que el robot esté siempre en contacto con la esfera se podría utilizar una porta-esfera, dispositivo encargado de enjaular de alguna forma a la esfera y mantenerla siempre en contacto ya sea con los rodillos o ruedas que le transmiten la potencia.
En la Figura 2.5 se puede observar un ejemplo de una porta-esfera ya aplicada a un modelo real.

Figura 2.5: Esfera del Rezero con porta-esfera.

Capítulo 3

Modelado

3.1. INTRODUCCIÓN

El primer paso para hacer que funcione el robot es el modelado dinámico del mismo, ya que con éste es posible obtener la relación entre las señales de entrada y las salidas del sistema y posteriormente, utilizando esta información, se puede diseñar un control para que el sistema actúe de forma deseada. También es muy útil contar con el modelo del sistema para realizar simulaciones de su control en computadora, en lugar de estar probándolo en el sistema real cada vez que se desee realizar algún cambio.

Para este robot, el sistema se analizó como un caso particular de un péndulo invertido, siendo el péndulo el cuerpo del robot, y el eje de rotación el centro de la esfera. Para modelarlo, se consideró al cuerpo del robot como un cilindro rígido montado sobre una esfera igualmente rígida. Además, se hicieron las siguientes suposiciones:

- No existe deslizamiento de la esfera y la superficie donde se encuentre el robot.
- No existe deslizamiento de las ruedas del robot y la esfera.
- No existen deformaciones en la esfera.
- El movimiento del robot en dos planos verticales perpendiculares está desacoplado.

Esta última suposición permitió llevar a cabo un modelado bidimensional del robot y diseñar un control para mantener la posición vertical del robot, el cual se aplicó en dos ejes perpendiculares para obtener un control tridimensional del robot.

Para la obtención de las ecuaciones dinámicas del sistema, se decidió utilizar el método de Euler-Lagrange, el cual se basa en la conservación de la

energía para obtener las ecuaciones dinámicas. Para esto, es necesario seguir los siguientes pasos:

1. Determinar los parámetros y las coordenadas generalizadas del sistema.
2. Obtener las expresiones de las energías cinética (T) y potencial (V) de todos los cuerpos en términos de las coordenadas generalizadas.
3. Expresar las energías disipadas en términos de las coordenadas generalizadas.
4. Obtener el Lagrangiano $\mathcal{L}$ del sistema.
5. Sustituir $\mathcal{L}$ en la ecuación de Euler-Lagrange.
6. Despejar la ecuación de Euler-Lagrange para la segunda derivada de cada coordenada generalizada.

3.2. PARÁMETROS Y COORDENADAS GENERALIZADAS DEL SISTEMA

En la Figura 3.1 se muestra un diagrama simplificado del sistema a modelar.

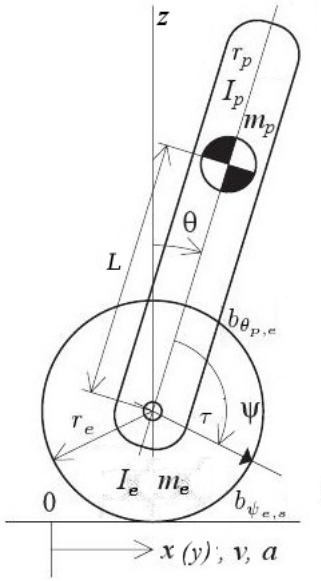


Figura 3.1: Diagrama del sistema a modelar.

Para el diagrama anterior

Tabla 3.1: Parámetros del sistema

Variable	**Significado**
m_e	Masa de la esfera
m_p	Masa del péndulo
r_e	Radio de la esfera
r_p	Radio del péndulo
I_e	Momento de inercia de la esfera
I_p	Momento de inercia del péndulo en los planos xz y yz
L	Distancia entre el centro de gravedad del péndulo y de la esfera
$b_{\psi_{e,s}}$	Coeficiente de fricción agular entre esfera y suelo (en kg·m^2)
$b_{\theta_{p,e}}$	Coeficiente de fricción angular entre péndulo y esfera (en kg·m^2)
g	Aceleración gravitacional

De la Figura 3.1 se pueden también determinar las coordenadas generalizadas utilizadas para describir al sistema, la cuales son θ_x, θ_y, que representan la inclinación del cuerpo del robot con respecto a la vertical y alrededor de los ejes x y y respectivamente, y ψ_x, ψ_y, que representan la orientación de la esfera con respecto al eje del cuerpo. Esta segunda coordenada se eligió de esa forma ya que así, ψ representa el ángulo que gira la esfera al hacer girar las ruedas del cuerpo del robot. Además, como ya se mencionó, al suponer que el movimiento del robot en dos planos verticales perpendiculares, en lo planos xz y yz, está desacoplado, se puede llevar a cabo un análisis bidimiensional del robot y suponer que se tendrán los mismos resultados en ambos planos, por lo que las coordenadas generalizadas para el análisis son:

$$q = \begin{bmatrix} \psi \\ \theta \end{bmatrix} \tag{3.1}$$

3.3. ENERGÍAS CINÉTICA, POTENCIAL Y DISIPADAS DEL SISTEMA

Para la esfera, la expresión de la energía cinética resulta de la suma de la parte traslacional más la parte rotacional, quedando de la siguiente forma:

$$T_e = \frac{1}{2} m_e r_e^2 (\dot{\theta} + \dot{\psi})^2 + \frac{1}{2} I_e (\dot{\theta} + \dot{\psi})^2 \tag{3.2}$$

La energía potencial de la esfera, con respecto a un plano horizontal que pasa por el centro de ésta, resulta ser nula.

$$V_e = 0$$

Finalmente la energía disipada por la esfera solamente es la causada por la posible fricción con el suelo, por lo que queda de la siguiente forma:

$$D_e = \frac{1}{2} b_{\psi_{e,s}} \dot{\psi}^2 \tag{3.3}$$

Para el cuerpo del robot, la expresión de la energía cinética también resulta de la suma de la parte traslacional más la parte rotacional, pero aparecen términos que resultan de la interacción entre el cuerpo y la esfera, por lo que dicha expresión queda de la siguiente manera:

$$T_p = \frac{1}{2} m_p \left[r_e^2 (\dot{\theta} + \dot{\psi})^2 + L^2 \dot{\theta}^2 + 2 r_e L \dot{\theta} (\dot{\theta} + \dot{\psi}) \cos\theta \right] + \frac{1}{2} I_p \dot{\theta}^2 \tag{3.4}$$

La energía potencial del cuerpo con respecto a un plano horizontal que pasa por el centro de la esfera sólo depende del peso del cuerpo que no es soportado por la esfera al estar el cuerpo inclinado, por lo que la expresión resulta:

$$V_p = m_p L g \cos\theta \tag{3.5}$$

Por último, la energía disipada por el cuerpo sólo es la ocasionada por la posible fricción entre el cuerpo y la esfera, o mejor dicho, entre las ruedas del cuerpo y la esfera, quedando la expresión como se muestra a continuación:

$$D_c = \frac{1}{2} b_{\theta_{p,e}} \dot{\theta}^2 \tag{3.6}$$

3.4. ECUACIONES DINÁMICAS DEL SISTEMA

Para obener las ecuaciones dinámicas del sistema, se procede a obtener el *Lagrangiano* $\mathcal{L}$ del mismo, que no es más que el resultado de restar algebráicamente las energías potenciales de las energías cinéticas del sistema:

$$\mathcal{L} = T - V$$

$$\mathcal{L} = (T_e + T_p) - (V_e + V_p) \tag{3.7}$$

Obtenido el *Lagrangiano*, se procede a resolver la ecuación de Lagrange para cada coordenada generalizada q_i del sistema:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i} \right) - \frac{\partial \mathcal{L}}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = E \tag{3.8}$$

donde D se refiere a la suma de las energías disipadas y E a las entradas generalizadas del sistema, siendo la única entrada el par generado por los motores para mover la esfera, afectando directamente la coordenada ψ, por lo que:

$$D = D_e + D_p \tag{3.9}$$

$$E = \begin{bmatrix} \tau \\ 0 \end{bmatrix} \tag{3.10}$$

Resolviendo la ecuación de Lagrange para cada coordenada generalizada, es posible llegar a la forma de Euler-Lagrange, la cual es una forma matricial de las soluciones que se puede expresar de la siguiente manera:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + K(q) = E \tag{3.11}$$

donde $M(q)$ representa la llamada matriz de inercia, $C(q,\dot{q})$ los términos de fricción y de Coriolis y $K(q)$ los términos debidos a la energía potencial. Estas matrices, para el presente sistema resultan de la siguiente forma:

$$\begin{aligned} M(q) &= \begin{bmatrix} \alpha & \alpha + \gamma\cos\theta \\ \alpha + \gamma\cos\theta & \alpha + \beta + 2\gamma\cos\theta \end{bmatrix} \\ C(q,\dot{q}) &= \begin{bmatrix} -\gamma\dot{\theta}\sin\theta & b_{\psi_{e,s}} \\ -\gamma\dot{\theta}\sin\theta + b_{\theta_{p,e}} & 0 \end{bmatrix} \\ K(q) &= \begin{bmatrix} 0 \\ -m_p L g\sin\theta \end{bmatrix} \end{aligned} \tag{3.12}$$

donde:

$$\begin{aligned} \alpha &= m_e r_e^2 + I_e + m_p r_e^2 \\ \beta &= m_p L^2 + I_p \\ \gamma &= m_p r_e L \end{aligned} \tag{3.13}$$

Para encontrar la solución para cada coordenada, es necesario despejarlas, quedando la ecuación general:

$$\ddot{q} = M(q)^{-1}E - M(q)^{-1}C(q,\dot{q})\dot{q} - M(q)^{-1}K(q) \tag{3.14}$$

Finalmente la solución para cada coordenada queda de la siguiente forma:

$$\ddot{\psi} = \frac{(\alpha+\beta+2\gamma\cos\theta)(\tau - b_{\psi_{e,s}}\dot{\psi}) + (\beta+\gamma\cos\theta)(\gamma\dot{\theta}^2\sin\theta) + (\alpha+\gamma\cos\theta)(b_{\theta_{p,e}}\dot{\theta} - m_p L g\sin\theta)}{\alpha\beta - \gamma^2\cos^2\theta}$$

$$\ddot{\theta} = \frac{(\alpha+\gamma\cos\theta)(b_{\psi_{e,s}}\dot{\psi} - \tau) - \gamma^2\dot{\theta}^2\sin\theta\cos\theta + \alpha(m_p L g\sin\theta - b_{\theta_{p,e}}\dot{\theta})}{\alpha\beta - \gamma^2\cos^2\theta} \tag{3.15}$$

$$\begin{bmatrix} \ddot{\psi} \\ \ddot{\theta} \end{bmatrix} = M(q)^{-1}\begin{bmatrix} \tau \\ 0 \end{bmatrix} - M(q)^{-1}C(q,\dot{q})\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \end{bmatrix} - M(q)^{-1}K(q)$$

Con el método de Euler-Lagrange fue posible obtener un modelo del sistema cuyas entradas son los pares necesarios, por lo que se requerirían motores cuyo par sea controlable. Sin embargo, se pretende utilizar motores a pasos, los cuales son difíciles de controlar por medio del par, mas son fáciles de

controlar por velocidad, por lo que fue necesario convertir el modelo obtenido a otro donde la entrada del sistema fuera la velocidad o la aceleración de los motores, siendo necesaria, en el caso de control por aceleración, una integración para obtener la velocidad, la cual ya es una entrada utilizable en los motores a pasos.

Para llevar a cabo esta conversión, fue necesario primero *linealizar* el modelo obtenido mediante una expansión en la serie de Taylor de las ecuaciones dinámicas en el punto de operación. Para el presente sistema, dado un punto de operación:

$$\bar{a} = (\bar{\psi}, \dot{\bar{\psi}}, \bar{\theta}, \dot{\bar{\theta}}, \bar{\tau}) \tag{3.16}$$

esta expansión queda de la siguiente forma:

$$\ddot{\psi}_{lin} = \ddot{\psi}(\bar{a}) + \left.\frac{\partial \ddot{\psi}}{\partial \psi}\right|_{\bar{a}} (\psi - \bar{\psi}) + \left.\frac{\partial \ddot{\psi}}{\partial \dot{\psi}}\right|_{\bar{a}} (\dot{\psi} - \dot{\bar{\psi}}) + \left.\frac{\partial \ddot{\psi}}{\partial \theta}\right|_{\bar{a}} (\theta - \bar{\theta}) + \left.\frac{\partial \ddot{\psi}}{\partial \dot{\theta}}\right|_{\bar{a}} (\dot{\theta} - \dot{\bar{\theta}}) + \left.\frac{\partial \ddot{\psi}}{\partial \tau}\right|_{\bar{a}} (\tau - \bar{\tau})$$

$$\ddot{\theta}_{lin} = \ddot{\theta}(\bar{a}) + \left.\frac{\partial \ddot{\theta}}{\partial \psi}\right|_{\bar{a}} (\psi - \bar{\psi}) + \left.\frac{\partial \ddot{\theta}}{\partial \dot{\psi}}\right|_{\bar{a}} (\dot{\psi} - \dot{\bar{\psi}}) + \left.\frac{\partial \ddot{\theta}}{\partial \theta}\right|_{\bar{a}} (\theta - \bar{\theta}) + \left.\frac{\partial \ddot{\theta}}{\partial \dot{\theta}}\right|_{\bar{a}} (\dot{\theta} - \dot{\bar{\theta}}) + \left.\frac{\partial \ddot{\theta}}{\partial \tau}\right|_{\bar{a}} (\tau - \bar{\tau}) \tag{3.17}$$

En este caso particular, el punto de equilibrio está en $\bar{a} = (0, 0, 0, 0, 0)$, el cual se puede sustituir en las ecuaciones *linealizadas*, y de ahí es posible obtener la representación del sistema en variables de estado de la forma:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$

$$\vec{y} = C\vec{x} + D\vec{u} \tag{3.18}$$

donde $\vec{x} = \left[\psi, \dot{\psi}, \theta, \dot{\theta}\right]^T$, $\vec{u} = \tau$ y $\vec{y} = [\theta, \dot{\theta}]^T$ debido a que sólo es posible detectar la inclinación y velocidad angular del cuerpo del robot. Por lo tanto, las matrices A, B, C y D quedan de la siguiente forma:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \left.\frac{\partial \ddot{\psi}}{\partial \psi}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\psi}}{\partial \dot{\psi}}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\psi}}{\partial \theta}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\psi}}{\partial \dot{\theta}}\right|_{\bar{a}} \\ 0 & 0 & 0 & 1 \\ \left.\frac{\partial \ddot{\theta}}{\partial \psi}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\theta}}{\partial \dot{\psi}}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\theta}}{\partial \theta}\right|_{\bar{a}} & \left.\frac{\partial \ddot{\theta}}{\partial \dot{\theta}}\right|_{\bar{a}} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \left.\frac{\partial \ddot{\psi}}{\partial \tau}\right|_{\bar{a}} \\ 0 \\ \left.\frac{\partial \ddot{\theta}}{\partial \tau}\right|_{\bar{a}} \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.19}$$

Teniendo la representación en variables de estado del sistema es posible, a partir de ésta, obtener la función de transferencia, la cual está dada por la expresión:

$$G(s) = C(sI - A)^{-1}B + D \tag{3.20}$$

Esto puede realizarse fácilmente en Matlab con el comando `[num,den]=ss2tf(sys)`, donde `sys` es el sistema en variables de estado, y `num` y `den` son arreglos con los coeficientes de las potencias de ***s*** en orden decreciente del numerador y el denominador de la funcióne de transferencia resultantes. Así se obtiene:

$$G_1 = \frac{\Psi(s)}{T(s)}$$

$$G_2 = \frac{\Theta(s)}{T(s)} \tag{3.21}$$

donde $T(s)$ es la transformada de Laplace del par de entrada.

Dividiendo las funciones anteriores es posible obtener:

$$\frac{G_2}{G_1} = \frac{\Theta(s)}{\Psi(s)} \tag{3.22}$$

Finalmente, como se mencionó, es necesario que la entrada del sistema sea velocidad o aceleración. Además se sabe que en el dominio de Laplace, la derivada de una función f se obtiene como se muestra a continuación:

$$\mathcal{L}\left\{f^{(n)}(t)\right\} = s^n F(s) - \sum_{i=1}^{n} s^{n-i} f^{i-1}(0) \tag{3.23}$$

Para el caso particular de $f(t) = \psi$:

$$\mathcal{L}\left\{\ddot{\psi}(t)\right\} = s^2\Psi(s) - s\bar{\psi} - \dot{\bar{\psi}} = s^2\Psi(s) \tag{3.24}$$

De esa forma, la función de transferencia del ángulo del robot, θ, con respecto a la aceleración de la esfera, $\ddot{\psi}$, resulta:

$$\mathcal{L}\left\{\frac{\theta(t)}{\ddot{\psi}(t)}\right\} = \frac{\Theta(s)}{s^2\Psi(s)} \tag{3.25}$$

Y para obtener la función de transferencia del ángulo de la esfera, ψ, con respecto a su aceleración:

$$\mathcal{L}\left\{\frac{\psi(t)}{\ddot{\psi}(t)}\right\} = \frac{\Psi(s)}{s^2\Psi(s)} = \frac{1}{s^2} \tag{3.26}$$

La razón por la que se obtuvieron las funciones de transferencia con la aceleración de la esfera, $\ddot{\psi}$, en lugar de su velocidad, $\dot{\psi}$, como entrada, fue debido a que con la velocidad como entrada, la matriz de controlabilidad del sistema no era de rango completo, por lo que era no controlable.

Capítulo 4

Primer Prototipo

4.1. SELECCIÓN

Una vez analizados tanto los posibles dispositivos a utilizar como las diferentes configuraciones del sistema, hubo que seleccionar las mejores opciones para el mejor funcionamiento posible del robot.

4.1.1. Sensores

En la gran mayoría de los trabajos anteriores se observa que los métodos de control utilizados se basan en evitar la inclinación del robot, es decir, controlan el ángulo del cuerpo del robot. Los mejores dispositivos para obtener esta información del robot son los sensores inerciales, como acelerómetros y giroscopios, cuya información se puede procesar y combinar para obtener mejores datos. Para esto, existen unas unidades llamadas IMU (Unidad de Mediciones Inerciales, por sus siglas en inglés), las cuales por lo general cuentan con acelerómetros y giroscopios en tres ejes ortogonales, y proporcionan la información inercial completa de un objeto.

En este caso particular, se seleccionó a la Ultimate IMU del fabricante Sparkfun para satisfacer las necesidades de mediciones inerciales. Además, se observó que tiene el protocolo de comunicación que se adapta mejor al diseño propuesto. Este dispositivo cuenta con los siguientes componentes:

- Acelerómetro ADXL345, el cual es de tres ejes, con posibilidad de medición de ±2 g, ±4 g, ±8 g hasta ±16 g, bajo consumo de potencia y tamaño reducido. Cuenta con un convertidor analógico a digital integrado, el cual tiene una resolución máxima de 13 bits. El integrado se encuentra en un empaquetado de montura superficial.

- Giroscopio ITG3200, el cual se encuentra en un solo circuito integrado de empaquetado para montura superficial. Realiza la medición en tres ejes haciendo uso de tecnología MEMS, cuenta con tres convertidores analógico a digital de 16 bits, filtros internos seleccionables, y

un módulo de comunicación serial I²C. Es de bajo consumo de potencia, cuenta además con un sensor de temperatura y su sensibilidad de rango completo es de ±2000° por segundo.

- Magnetómetro HMC5843, el cual realiza mediciones de campo magnético en tres ejes y cuenta con un empaquetamiento para montura superficial. Incluye un convertidor analógico a digital de 12 bits, además de un módulo para comunicación serial I²C.

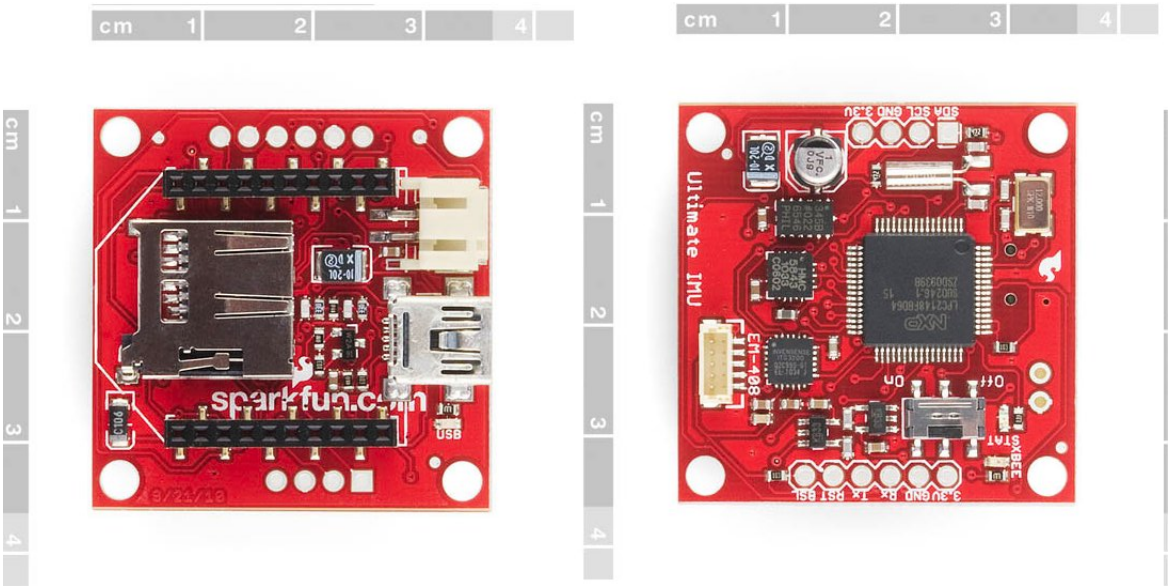


Figura 4.1: Fotografía frontal (izq.) y trasera (der.) de la Ultimate IMU con sus medidas indicadas. Las imágenes son propiedad de la compañía creadora SparkFun Electronics.

4.1.2. Procesamiento

La Ultimate IMU cuenta con un *microcontolador* LPC2148 basado en una arquitectura ARM7DMI-S, el cual tiene 32 kB de memoria RAM interna, 14 canales de convertidores analógico a digital de 10 bits y un convertidor digital a analógico. A máxima frecuencia del reloj externo alcanza una velocidad de 60 MIPS. Cuenta con módulos de comunicación USB, SPI, SSP e I²C. Cuenta con canales de PWM y 45 pines de entrada/salida digital de alta velocidad. Lamentablemente la gran mayoría de los pines de entrada/salida así como los de PWM no se encuentran conectados en la tarjeta.

Además cuenta con dos módulos UART (Receptor/Transmisor Asíncrono Universal, por sus siglas en inglés: *Universal Asynchronous Receiver/Transmitter*), los cuales pueden alcanzar velocidades de transmisión de hasta 115200 baudios. La programación del *microcontrolador* se realizó a través del puerto

serial de una computadora, para lo cual se utilizó un cable de la marca FTDI, que acopla la señal bipolar de Serial RS-232 que usa la computadora a la señal de directa de 3.3 V de TTL Serial que utiliza el *microcontrolador*. Se diseñó un pequeño circuito que permitiera tener los valores de entrada necesarios para ingresar al modo de programación, así como un botón de reinicio para la IMU.

El procesador de la IMU tiene la capacidad de llevar a cabo todo el procesamiento de la posición y velocidad angular además del control, sin embargo, el no tener sus salidas digitales y de PWM conectadas a la tarjeta dificultaba la conexión del procesador con los controladores de los motores, así como con los demás dispositivos que puedan utilizarse a futuro como sensores, otros procesadores, etc., para agregar funciones al robot.

Para resolverlo se decidió utilizar otra tarjeta de desarrollo externa, la cual facilitó la comunicación entre los sensores y el resto del robot y agregó posibilidades a la implementación del control. Además se tuvieron que cubrir tanto la necesidad de contar con salidas PWM para el control de los motores, como la de tener salidas digitales para el control de la dirección de los mismos y para conectar indicadores ópticos que facilitaran la depuración de lo implementado en el *microcontrolador*. Finalmente, por lo general las tarjetas de desarrollo cuentan con entradas digitales y analógicas, las cuales podrían ser utilizadas para futuras implementaciones de funcionalidades para el robot, como la posibilidad de moverlo mediante un *joystick*, lectura de otros sensores como sonares o lásers para evadir obstáculos, generación de trayectorias, etc.

Por todo lo anterior se decidió utilizar una tarjeta de desarrollo Arduino. Estas tarjetas cuentan con todas las salidas y entradas requeridas, además de que el uso de sus bibliotecas ya existentes para la solución de infinidad de problemas como comunicación serial, control de motores, controladores PID, etc., facilitó notablemente su implementación.

El Arduino es una plataforma de hardware libre basado en *microcontroladores* Atmel, que cuenta con su propio entorno de desarrollo de programación mediante un lenguaje propio de alto nivel llamado *Processing*, aunque es posible utilizar otros lenguajes como Java, C++, Python, entre otros. El dispositivo utilizado es el Arduino Duemilanove, con un microprocesador Atmega328, el cual cuenta con las siguientes características:

- Voltaje de operación: 5 V
- Pines digitales: 14 en total, que pueden servir como entradas o como salidas. Seis pueden proporcionar una señal de PWM como salida y dos pueden utilizarse para interrupciones externas
- Pines analógicos: Seis que funcionan únicamente como entradas.
- Memoria Flash: 32KB, de los cuales 2KB son utilizados para el *bootloader*
- Memoria Estática de Acceso Aleatorio (SRAM): 2KB

- Memoria EEPROM: 1KB
- Velocidad del reloj: 1 MHz
- Comunicación serial:
  - RX y TX para comunicación UART
  - SDA y SCL para comunicación I^2C
  - SCLK, MOSI, MISO y SS para comunicación SPI

El diseño del Arduino está pensado para ser utilizado como una tarjeta de desarrollo más que para una aplicación final, por lo que los conectores que utiliza y el arreglo de los mismos están diseñados para ser de fácil acceso pero poco robustos. Dado que en este proyecto el Arduino funciona como el procesador central, era necesario que pudiera conectarse con todos los demás módulos, es decir, con la IMU y con el controlador de los motores.

Para comunicar a la IMU con el Arduino, se diseñó e implementó una tarjeta de comunicación a la que se denominó Módulo Interno de Comunicación, o MIC. Éste tiene por objetivo realizar la conexión entre el Arduino y los motores, la IMU y demás conexiones necesarias de forma robusta. También cuenta con indicadores ópticos conectados a las salidas digitales del Arduino, de manera que se puede facilitar la depuración de los algoritmos gracias al uso de banderas e indicadores, implementados en los leds del MIC, que permiten conocer el estado del algoritmo.

4.1.3. Cuerpo del robot

Se decidió seleccionar un cilindro como la mejor forma para el cuerpo, debido a que optimiza el volumen con respecto a la superficie total del cuerpo, además de que su simetría axial resulta ser muy útil a la hora del modelado del sistema y la implementación del control, ya que permite el análisis bidimensional y la posterior generalización en planos verticales perpendiculares para obtener el análisis tridimensional.

Además, pensando en que el diseño fuera una estructura fácilmente modificable y expandible, se buscó que la forma no tuviese esquinas o protuberancias para que facilitaran el diseño futuro de una posible carcasa, por lo cual la forma circular resultó la más conveniente.

También el uso de un cilindro para el cuerpo ayuda a cumplir el objetivo de que el robot se pueda mover libremente en espacios humanos, dado que facilita el movimiento y el giro libre sin chocar con otros objetos.

Para el acomodo de los dispositivos se decidió que lo más sencillo era utilizar un cuerpo con varios pisos, pudiendo acomodar así la parte de potencia más cerca de los motores en pisos más bajos y la parte de *sensado* más arriba, o sea en pisos más altos. Además, al usar diferentes pisos resultó más fácil la fijación de los dispositivos a ellos, en lugar de tener que buscar la forma de fijarlos a un pilar central o a las paredes interiores de un cilindro cerrado.

Finalmente, para la separación entre pisos se decidió por la utilización de columnas en forma de "C", esto con la finalidad de poder tratar a cada piso como un módulo independiente en caso de necesitarse cambios o reparaciones, facilitando el desarmado del robot.

4.1.4. Locomoción

Para el sistema de locomoción, primero fue necesario elegir el tipo de motor a utilizar. En la Tabla 4.1 se presenta la matriz en la que se basó su selección.

Como se puede observar en la Tabla 4.1, se decidió por el motor a pasos debido a que en principio se requería que pudiera girar en un rango amplio de velocidades. El control de la velocidad de un motor a pasos consiste en modificar los tiempos de espera en los que el motor permanece en un mismo punto o paso, y como este tiempo no tiene un límite inferior, el motor a pasos se puede mover tan lentamente como sea necesario y aún a velocidades bajas es posible asegurar el control del movimiento.

Aparte de lo anterior, en el motor de corriente directa así como en el motor sin escobillas se suele utilizar juegos de engranes para obtener velocidades más bajas, pero esto introduce un juego entre los engranes lo cual puede causar problemas en el control dado que es difícil de modelar, por lo cual se consideró que el utilizar engranes no era conveniente para este caso. Las características de control de lazo abierto del motor a pasos reducen los costos de los circuitos y del software de control.

Los motores seleccionados son de la serie 23Y de la compañía Anaheim Automation, de 200 pasos por revolución, con 1.85 N·m de par máximo.

Una vez elegido el motor, hubo que seleccionar la forma en la que se transmitiría el movimiento a la esfera, o dicho de otro modo, la forma en la que se acomodarían los motores con sus respectivas ruedas o rodillos para hacer girar a la esfera. En la Tabla 4.2 se presenta una matriz con el análisis de las distintas posibilidades de acomodo de los motores basado en las opciones presentadas en la Figura 2.2 del capítulo 2.

Con los resultados obtenidos en la matriz de decisión, las dos mejores soluciones resultaron ser la opción 2 y la opción 4. Se decidió utilizar esta última para el diseño final, no sólo por tener la mayor puntuación, sino también debido a que se consideró que era un reto académico más interesante además de que era estéticamente más agradable. Una vez elegido el acomodo, se tuvieron que utilizar ruedas omnidireccionales requeridas por el tipo de acomodo de los motores. Las ruedas seleccionadas fueron las de la compañía VEX con un diámetro de 7 cm y con doce rodillos.

Para el acoplamiento entre el eje del motor y la rueda fue necesario maquinar un cople dado que el diámetro del eje del motor era de 0.25 in, y el tamaño del orificio de la rueda era de forma cuadrada con 0.125 in por lado.

Tabla 4.1: Matriz de decisión de los motores.

Concepto	**Rango de velocidad (/10)**	**Control y costo de implementación (/10)**	**Señal de control de velocidad (/10)**	**Controlabilidad a velocidades bajas (/10)**	**Resolución (/10)**	**Costo (/10)**	**Otros**	**Total**
Motor CD	Normal 5	Lazo cerrado 5	PWM 8	Baja, requiere reducción 2	Depende del *encoder* 8	Bajo 8	Muy simple en general +2	38
Motor a pasos	Amplio 8	Lazo abierto 10	Tiempo entre pasos 5	Buena 8	Requiere *microstepping* 6	Medio 5	Más modos de manejo: *microstepping* +5	47
Motor sin escobillas	Normal 5	Lazo cerrado 5	Tiempo entre fases 5	Baja, requiere reducción 2	Depende del *encoder* 8	Alto 2	Mejor eficiencia +5	32
Servomotor	Amplio 8	Lazo abierto 10	Perfiles de velocidad 3	Buena 8	Resolución máxima dada por el fabricante 6	Alto 2	Control de posición absoluta +3 Requiere modificaciones -2	38

El diseño de esta pieza quedó como se observa en la Figura 4.2. Consta de una sección donde se inserta el eje del motor y se fija mediante un tornillo prisionero, y otra sección que es la que se inserta en el orificio de la rueda y termina en una sección con cuerda para poder colocar una tuerca como tope para que no se salga la rueda.

Figura 4.2: Acoplamiento entre el eje del motor y la rueda omnidireccional.

Para lograr el acomodo seleccionado fue necesaria la construcción de un soporte que fijara los motores al cuerpo y que lograra que las ruedas estuvieran en contacto con la esfera a un ángulo de 45° con respecto a la horizontal. Se eligió el ángulo de 45° por ser un punto intermedio entre colocar las ruedas totalmente horizontales o totalmente verticales.

Este soporte consistió en una estructura de tamaño similar al motor, con una altura justa para que el motor entrara sin dejar espacios libres, con la intención de que fuera una estructura más resistente y que el ángulo deseado no se modificara por el peso ejercido por el robot. Se dejaron los lados libres con la intención de que fuera fácil introducir y sacar el motor del soporte, siendo de esta manera más fácil modificar el acomodo de los motores o remplazarlos en caso necesario. Esto también facilitó el paso de los cables de los motores hacia los respectivos dispositivos montados en el cuerpo. La forma del soporte se puede observar claramente en la Figura 4.3.

El material seleccionado fue acrílico de 4 mm de espesor y de color blanco, el cual se dobló por medio de uso de calor mediante un escantillón con los ángulos deseados para los dobleces.

4.2. CONTROLADOR PARA EL MOTOR A PASOS

Una vez elegidos los motores a pasos, hubo que diseñar un controlador de velocidad, ya que a diferencia de los otros motores, para el motor a pasos es necesario cambiar el tiempo entre pasos para controlar la velocidad del motor. Además, haciendo pruebas con el motor, se observó que a velocidades bajas el movimiento era algo tosco, característica indeseada para el sistema, por

Tabla 4.2: Matriz de decisión del acomodo de los motores.

Opción	**Tipo de ruedas necesarias (/10)**	**Dificultad de manufactura (/10)**	**Transmisión de potencia del motor a las ruedas (/10)**	**Giro alrededor de un eje vertical (*yaw*) (/3)**	**Espacio que ocupa (/10)**	**Transmisión de potencia a la esfera (/10)**	**Número de ruedas (/10)**	**Total**
1	Comunes 10	Fácil 10	Requiere transmisión para reducir espacio 6	No 2	Mucho 6	Puede ser total 8	Cuatro 6	48
2	Omnidireccionales 8	Fácil 10	Requiere transmisión para reducir espacio 6	Sí 3	Medio 8	Parcial 6	Tres 10	51
3	Omnidireccionales 8	Un poco difícil 8	Requiere transmisión para reducir espacio 6	No 2	Mucho 6	Puede ser total 8	Cuatro 6	44
4	Omnidireccionales 8	Difícil, soportes de los motores con formas complicadas 6	Directa 10	Sí 3	Poco 10	Parcial 6	Tres 10	52
5	Omnidireccionales 8	Difícil, soportes de los motores con formas complicadas 6	Directa 10	Sí 3	Poco 10	Parcial 6	Cuatro 6	49

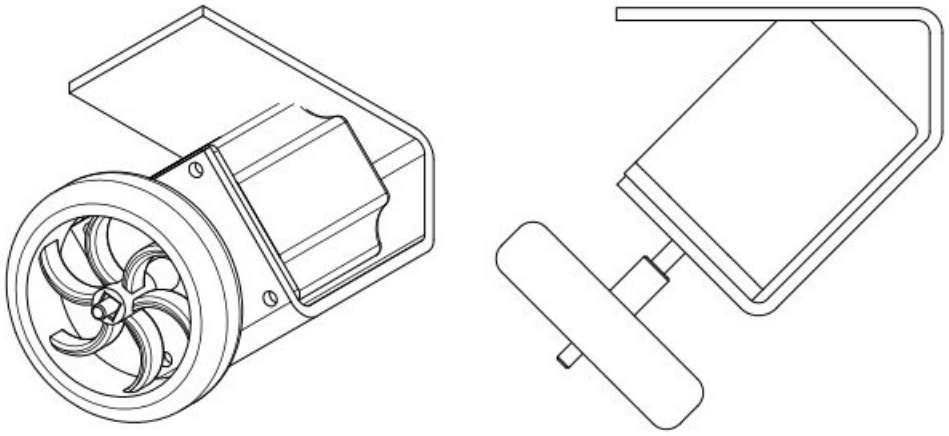


Figura 4.3: Motor con la rueda ya montado sobre el soporte.

lo que se buscó una forma para suavizarlo. Para esto, se utilizó un método llamado *microstepping*, el cual consiste en introducir al motor una forma de onda sinusoidal en lugar de cuadrada, lo cual permite que el rotor se desplace suavemente entre distintas posiciones antes de llegar a uno de los polos o pasos determinados. De esta manera el rotor gira de forma continua manteniendo constante tanto el par generado como el movimiento del rotor. Si el voltaje de entrada simula una forma sinusoidal, el comportamiento del motor es similar al de un motor de corriente directa.

Para esto se utilizó el circuito TA8435H, el cual es un *PWM Chopper Driver* para motores a pasos bipolares, el cual puede dividir cada paso del motor en hasta ocho pasos, haciendo que el movimiento del mismo sea más suave, y se tenga una mayor resolución en su control. El TA8435H utiliza una señal digital para el control de la dirección de giro, y una señal digital cuadrada de frecuencia variable para el control de la velocidad. Esta última es la más importante, pero debido a que su implementación en el microcontrolador podría utilizar muchos recursos, se decidió diseñar un circuito electrónico que la generara para que el microcontrolador sólo se encargara del control.

Así pues, para la señal digital cuadrada de frecuencia variable se decidió utilizar un oscilador controlado por voltaje (VCO por las siglas en inglés de Voltage Controlled Oscillator), el cual es un convertidor de voltaje a frecuencia con el cual se puede tener como salida una señal cuadrada de frecuencia variable. Con esto, haciendo la electrónica más compleja se logra un procesamiento más sencillo, permitiendo usar toda la velocidad del microprocesador en el control del robot.

El control de la frecuencia que otorga el VCO está dado por una señal analógica de voltaje variable. Dado que no todos los microprocesadores utilizados para el control de motores tienen esta función, se utilizó un filtro para convertir una señal de PWM en una señal analógica de voltaje variable.

Específicamente la señal de PWM que se utilizó es la que genera el Ar-

duino, la cual es una señal con una frecuencia de 430 Hz. Se utilizó un simple filtro RC de primer orden para convertir la señal cuadrada de ciclo de trabajo variable en una señal de CD con amplitud variable. Este filtro se puede observar en la Figura 4.4 y se diseñó con una frecuencia de corte de 70 Hz.

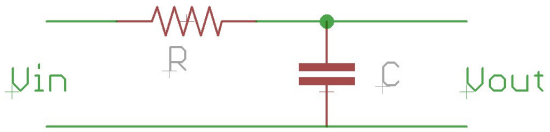


Figura 4.4: Filtro para convertir una señal de PWM en un voltaje analógico.

De esta forma se implementó un controlador que recibe una señal de PWM del *microcontrolador* y la convierte en una velocidad para el motor a pasos.

Se diseñó una tarjeta con el circuito previamente descrito, con la intención de que se contara con un módulo de control del motor a pasos. Este módulo fue nombrado MCP y fue diseñado con la intención de que fuera robusto y portable. El módulo cuenta con un disipador que se atornilla directamente al disipador del circuito integrado TA8435H para asegurar una buena disipación del calor y mantener en funcionamiento al módulo por periodos largos. El MCP recibe una señal de PWM y de dirección de giro y genera la velocidad deseada en el motor a pasos mediante *microstepping*, dividendo los pasos originales del motor entre ocho, generando así una mayor resolución y un movimiento mucho más suave.

Finalmente se obtuvo un controlador para los motores a pasos cuyo diagrama de operación se muestra en la Figura 4.5.

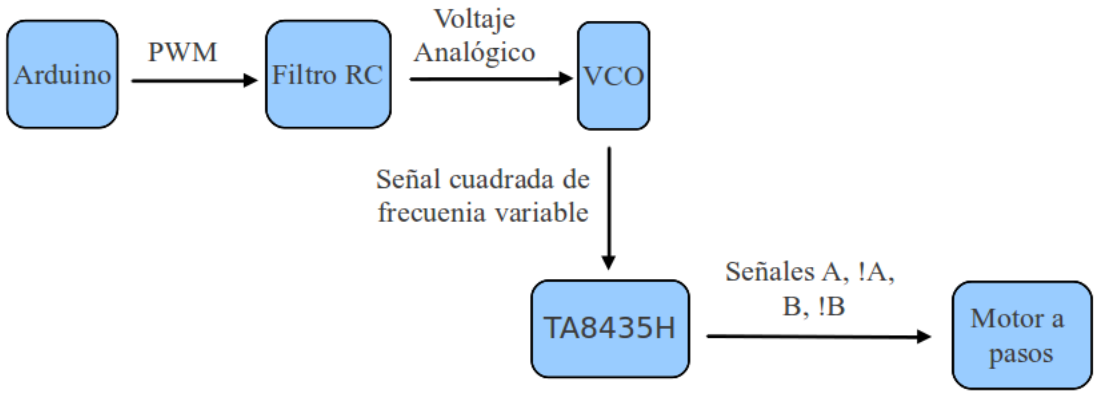


Figura 4.5: Diagrama de operación del módulo de control del motor a pasos.

4.3. ESFERA

Para la elección de la esfera, se pensó en aquéllas de uso común, siendo pelotas y balones los que cumplen de mejor manera este requisito. Con base en la consideración de que se necesitaba cierta rigidez en la superficie del balón o pelota, se pensó en utilizar un balón de básquetbol, que tiene alrededor de 24 cm de diámetro. En trabajos anteriores, sin embargo, mencionan que este balón funcionaba bien para la estabilidad pero presentaba problemas de vibración al acelerar los motores debido a la falta de rigidez en la superficie [2], por lo que al final utilizaron una bola de boliche de 22 cm de diámetro. Esto dio un parámetro para elegir el tamaño de la esfera, que al final por restricciones debido al tamaño de los motores, se decidió que su diámetro fuera de aproximadamente 26 cm.

Para su construcción se tomó como base una esfera de unicel de 25 cm de diámetro, la cual primero se recubrió con pegamento líquido para rigidizar la superficie. Posteriormente fue recubierta con tres capas de *rellenador* plástico Dupont NovoLite© para darle más rigidez todavía, y finalmente fue rociada con cuatro capas de hule líquido de la marca PlastiDip©. El resultado fue una esfera de aproximadamente 25.5 cm de diámetro, ligera, de aproximadamente 1 kg, con un exterior rígido y con una cobertura que asegura una alta fricción con otros materiales.

4.4. CONTROL

Como primer propuesta de control se decidió utilizar un control proporcional y derivativo, o PD, ya que de los sensores es posible obtener directamente el ángulo de inclinación del cuerpo del robot y su velocidad angular, que resulta ser la derivada del ángulo de inclinación, por lo que para el control sólo era necesario retroalimentar estos valores, compararlos con el valor deseado y multiplicarlos por una ganancia para obtener la señal de control.

Así, con base en el modelado realizado previamente, la señal de control que es el valor de la aceleración necesaria de la esfera para balancear al cuerpo, quedó de la siguiente forma:

$$a_x = K_{\theta x}\theta_{xz} + K_{\dot{\theta} x}\dot{\theta}_{xz} \tag{4.1}$$

$$a_y = K_{\theta y}\theta_{yz} + K_{\dot{\theta} y}\dot{\theta}_{yz} \tag{4.2}$$

Para el diseño de este control, primero fue necesario obtener los valores de los parámetros del sistema, los cuales se muestran en la Tabla **??**.

Cabe destacar que los coeficientes de fricción angular se eligieron a partir de los encontrados en las diferentes fuentes bibliográficas de los sistemas ya existentes.[5]

Tabla 4.3: Valores de los parámetros físicos del sistema.

Variable	**Significado**	**Valor**
m_e	Masa de la esfera	0.8 kg
m_p	Masa del péndulo	5 kg
r_e	Radio de la esfera	0.255 m
r_p	Radio del péndulo	0.27 m
I_e	Momento de inercia de la esfera	0.005 kg·m^2
I_p	Momento de inercia del péndulo en los planos xz y yz	0.111 kg·m^2
L	Distancia entre el centro de gravedad del péndulo y de la esfera	0.13 m
$b_{\psi_{e,s}}$	Coeficiente de fricción agular entre esfera y suelo (en kg·m^2)	0.01 N·m·s/rad
$b_{\theta_{p,e}}$	Coeficiente de fricción angular entre péndulo y esfera (en kg·m^2)	0.1 N·m·s/rad
g	Aceleración gravitacional	9.81 m/s^2

A partir de los parámetros, se diseñó un control por retroalimentación de estados para tener una primera aproximación de las constantes dada una respuesta deseada. Para esto se utilizó una respuesta con tiempo de asentamiento $t_s = 0.5$ s y sobrepaso $s_p = 8\,\%$. Además, dado que sólo se utilizaron las señales de posición y velocidad angular del cuerpo, primero se obtuvo un sistema reducido sólo con dichas variables, sin tomar en cuenta la posición y velocidad angulares de la esfera.

Las matrices A, B, C y D de este sistema reducido resultaron:

$$A = \begin{bmatrix} -0.156 & 14.927 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & -0.339 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

Con estas matrices, primero se calculó la matriz de controlabilidad, $\mathbf{C}$, para asegurar que el sistema fuera *estabilizable*. Dicha matriz resultó ser la siguiente:

$$\mathbf{C} = \begin{bmatrix} 1 & -0.156 \\ 0 & 1 \end{bmatrix}$$

Se puede comprobar fácilmente que es de rango completo, por lo que el sistema resulta ser controlable y por tanto *estabilizable*.

De ahí se obtuvo la matriz $\mathbf{K}$ de las constantes para cada estado:

$$\mathbf{K} = \begin{bmatrix} K_D & K_P \end{bmatrix} = \begin{bmatrix} 18.264 & 139.236 \end{bmatrix}$$

Estas constantes se simularon en Matlab sobre el modelo del sistema. Con objeto de mejorar el desempeño, las constantes obtenidas mediante el método de retroalimentación de estados se fueron sintonizando manualmente.

Este primer control se implementó en el Arduino, el cual se programó para recibir los datos de posición y velocidad angular de la IMU, y con base en ellos producir una señal PWM para cada motor.

Una desventaja de esta propuesta fue que al no controlar la posición del robot, a pesar de que éste mantenga el equilibrio, era posible que no se mantuviera fijo en un punto, sino que se moviera indefinidamente para mantener el equilibrio.

4.5. PRUEBAS DE MOVIMIENTO EN PLANO

Para probar que se tuviera un control adecuado sobre las ruedas y que las ecuaciones de movimiento del motor fueran correctas, se realizaron pruebas de movimiento del robot sin la esfera sobre una superficie plana, enviando velocidades deseadas a los motores durante un tiempo establecido y comparando la distancia teórica, resultante de la velocidad enviada multiplicada por el tiempo, con la distancia real recorrida por el robot. Para esto, se fijó a la base del robot un marcador para obtener las mediciones de las distancias recorridas. Se realizaron pruebas de movimiento lineal para conocer la precisión del control de velocidad con la que se contaba, así como pruebas de cambio de dirección para conocer la precisión de la dirección de los movimientos del robot. A partir de estas pruebas hubo que agregar y ajustar unas constantes para mejorar el desempeño del robot.

Se logró tener una precisión en el control de la velocidad con un error menor al 10 %. Los movimientos en línea recta tuvieron un desviación de ± 5 % del recorrido realizado, de manera que se puede hablar de la existencia de ángulos de desviación de aproximadamente $\pm 3°$. Estos errores se tuvieron en la dirección de los movimientos que son perpendiculares a alguna de las ruedas omnidireccionales. En los demás movimientos los errores fueron menores al ± 1 %. Un ejemplo de esta prueba se puede observar en la Figura 4.6

Las pruebas de cambio de dirección se llevaron a cabo realizando cambios de dirección con el robot, de tal forma que se describieran ángulos para que su medición fuera más sencilla. Los errores en los ángulos descritos varían entre $0°$ y $\pm 5°$. El error es más grande cuando en el ángulo deseado una de las velocidades (velocidad en la componente x o en la y) es pequeña. El error es mínimo cuando ambas velocidades tienen valores grandes, como en el ángulo de $45°$. Estas pruebas fueron realizadas en todos los cuadrantes y se obtuvieron resultados similares, y se puede obsevar un ejemplo en la Figura 4.7. Los errores encontrados en el movimiento del robot en plano no son constantes y son causados por imperfecciones tanto en las ruedas como en la superficie de movimiento. La superficie utilizada fue una hoja de papel de 70x94 cm pegada sobre una mesa.

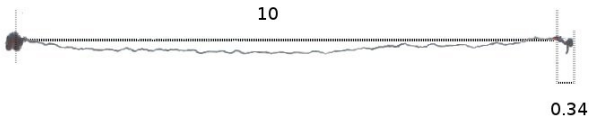


Figura 4.6: Trayectoria seguida por el robot (línea continua) y trayectoria deseada (línea punteada). La distancia deseada era de 10 cm, y se tuvo un error de 0.34 cm extra y una pequeña curvatura en la trayectoria.

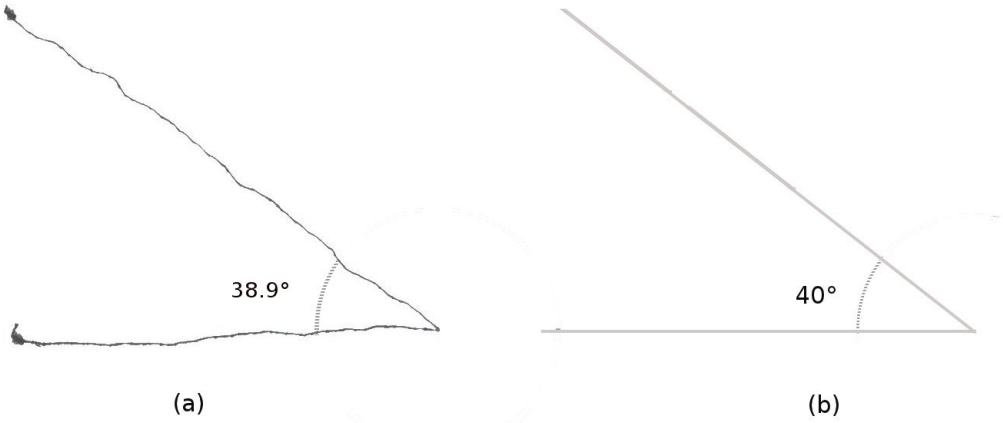


Figura 4.7: (a) Trayectoria realizada por el robot, (b) trayectoria deseada. El error es de 1.1° en 40°, es decir, de 2.75 %.

4.6. RESULTADOS

Se logró controlar a los motores a pasos con los MCP, obteniendo una velocidad máxima promedio de 140 rpm en las ruedas, lo que se traduce en una velocidad lineal de 0.513 m/s en el robot. La velocidad de respuesta del conjunto fue de 8 ms desde que el Arduino manda la señal de PWM hasta que el rotor empieza a girar. Se logró tener control simultáneo de los 3 motores, además de poder visualizar con los leds del MIC el PWM enviado a cada uno de los motores mediante su intensidad luminosa. También se logró controlar el movimiento del robot de forma externa mediante un *joystick* haciendo uso de las entradas analógicas del Arduino.

4.6.1. Problemas con la esfera y las ruedas

Hubo problemas con las ruedas debido a que había puntos de éstas en donde los rodillos no tocaban la esfera, rayando la esfera y removiendo la capa de hule, provocando al final que en las zonas sin hule las ruedas se deslizaran completamente y el robot fallara. Se trataron de realizar pruebas de estabilidad, sin embargo, el problema arriba mencionado impedía el movimiento correcto de las ruedas y no se lograron realizar pruebas satisfactorias. La esfera perdió la mayoría de su recubrimiento de PlastiDip e incluso se generaron fisuras y hoyos en el recubrimiento de *rellenador* plástico.

4.6.2. Ruido en las mediciones

En las pruebas de funcionamiento se encontró que las mediciones tanto de posición angular como de velocidad angular obtenidas de la IMU tenían ruido posiblemente generado por el movimiento del robot o las vibraciones de los motores. Este ruido indeseable podía provocar el mal funcionamiento del control, por lo que se decidió hacer algo al respecto.

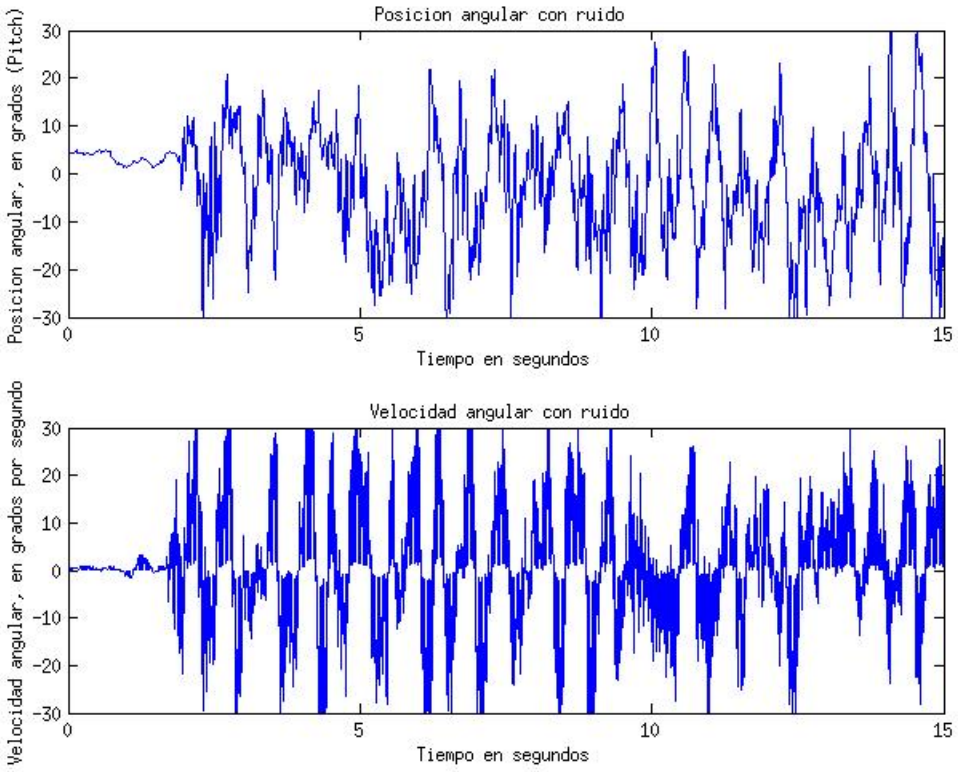


Figura 4.8: Gráficas de las señales de posición angular y velocidad angular obtenidas por la IMU.

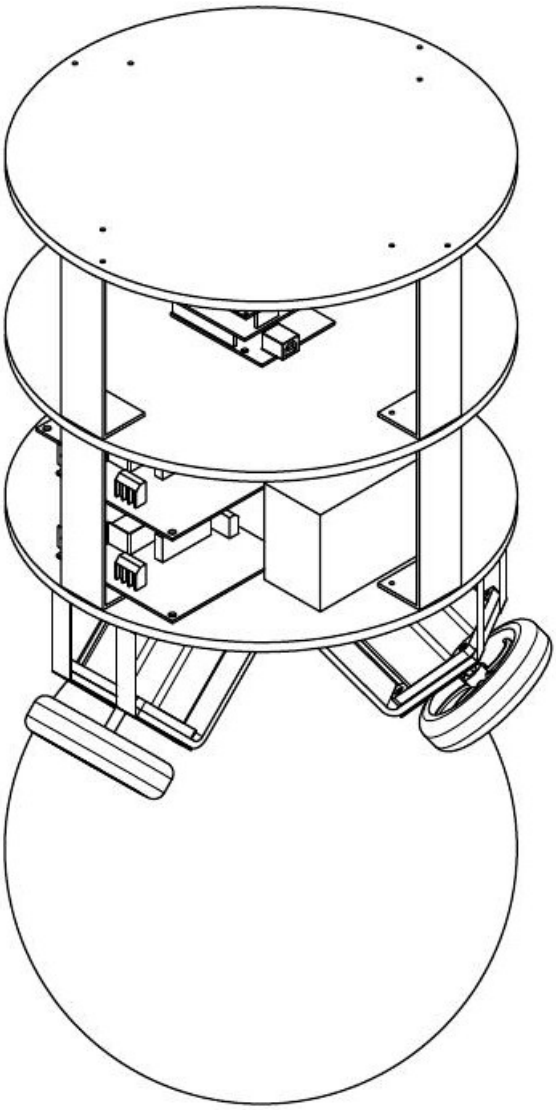


Figura 4.9: Primer prototipo.

Capítulo 5

Segundo Prototipo

De los resultados del primer prototipo surgieron varios puntos a resolver. Uno de ellos fueron las ruedas seleccionadas y la esfera, ya que las ruedas dañaron a esta última al punto que no se pudieron continuar las pruebas. Otro punto fue que las señales obtenidas de la IMU tenían ruido y una baja frecuencia de muestreo, lo cual no permitía un funcionamiento correcto del robot. Además, no era claro si la velocidad de respuesta del robot en conjunto era los suficientemente rápida para que se lograra la estabilidad sobre la esfera. Para arreglar estas problemáticas fue necesario realizar algunos cambios para que el robot tuviese un mejor desempeño, los cuales se muestran a continuación.

5.1. MECÁNICA

Un problema importante que se presentó en el primer prototipo fue el hecho de que las ruedas omnidireccionales maltrataban la esfera sobre la que se montaba el robot debido a que el espacio entre rodillos era muy grande. Esto provocaba que algunas veces, la rueda estuviera apoyada sobre la esfera en uno de estos puntos entre rodillos, el cual al ser rígido, causaba la pérdida de omnidireccionalidad de las ruedas, raspando y dañando la esfera. Esto, además de no permitir un buen funcionamiento del sistema, fue deteriorando a la esfera, o más específicamente, fue removiendo la capa de hule, haciendo que en donde ya no había hule las ruedas se deslizasen y el robot fallara.

Para arreglarlo se decidió cambiar las ruedas por otras de aluminio, más grandes, de 6in de diámetro, pero además dobles como las que se muestran en la Figura 5.1, creando un movimiento de giro continuo y evitando por completo el posible contacto de uno de los puntos entre dos rodillos con la esfera.

Además, como se mencionó, las ruedas anteriores deterioraron notablemente a la esfera, por lo que fue necesario conseguir una nueva. Ésta se hizo de tablero de fibra de densidad media, o MDF por sus siglas en inglés de Medium Density Fiberboard, y resultó más pesada, siendo de 6 kg, y ligera-

Figura 5.1: Nuevas ruedas omnidireccionales.

mente más grande, con 27 cm de diámetro.

Con las nuevas ruedas el peso que sostenían los soportes de los motores aumentó considerablemente, haciendo que éstos tuvieran movimientos indeseados y que posiblemente pudieran afectar su estructura o incluso hasta romperlos, por lo que se diseñaron unos refuerzos para dar mayor robustez y ayudar con la carga del peso de los motores y las ruedas. Estos refuerzos, que se muestran en la Figura 5.2, se construyeron de aluminio y se sujetaron a la estructura de los soportes de los motores con los mismos tornillos que sujetan al motor y al cuerpo del robot.

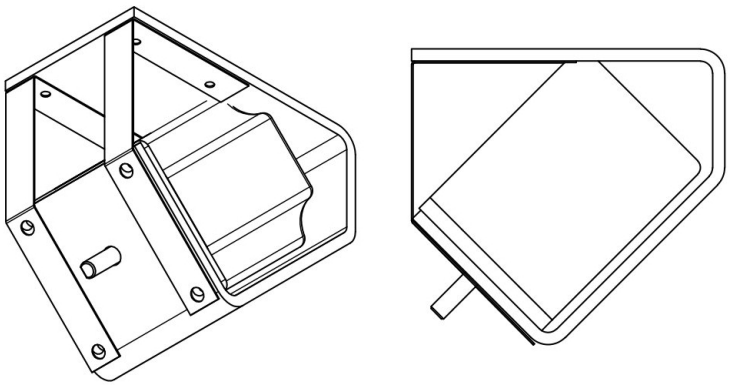


Figura 5.2: Refuerzos para los soportes de los motores.

5.2. ACONDICIONAMIENTO DE LAS SEÑALES DE LA IMU

5.2.1. Frecuencia de muestreo

Entre las características que pudieron generar que el primer prototipo no pudiera equilibrarse está el tener una frecuencia de muestreo muy baja. Por esto se decidió aumentarla tanto para la posición como para la velocidad angulares. Esta frecuencia estaba limitada por la velocidad de comunicación entre el Arduino y la IMU. Se aumentó la velocidad de comunicación entre los

dispositivos y se modificó la información transmitida por comunicación serial de la IMU al Arduino, de manera que se pudieran utilizar menos caracteres y por lo tanto menos paquetes que enviar, obteniendo una mayor velocidad de muestreo. Gracias a esta modificación, la frecuencia de muestreo aumentó de 66 Hz a 122 Hz.

5.2.2. Filtrado

Un resultado que arrojaron las pruebas con el primer prototipo fue que las mediciones de la velocidad y la posición angulares obtenidas con la IMU tenían bastante ruido, causado por fuentes desconocidas pero que no permitían un funcionamiento óptimo del algoritmo de control. Para arreglar este problema se decidió diseñar e implementar un filtro digital.

Para el diseño del filtro se utilizaron los datos de velocidad y posición angulares proporcionados por la IMU en una prueba realizada con el primer prototipo. Mediante la transformada rápida de Fourier se hizo un análisis en frecuencia de los datos que corroboró la existencia del ruido. En la gráfica mostrada en la Figura 5.3 se puede observar la señal de velocidad angular, tanto en el dominio del tiempo como en el de la frecuencia. Se puede verificar que las magnitudes más altas, es decir las que corresponden a la señal fundamental, se encuentran en frecuencias de hasta aproximadamente 5 Hz y a mayores frecuencias solamente se tiene ruido. Por lo anterior, se decidió diseñar un filtro pasa bajas.

Se decidió utilizar un filtro tipo FIR (Respuesta Finita al Impulso), dado que aunque se requiere que dicho filtro sea de un orden elevado para obtener el resultado esperado, tiene la ventaja de ser estable, la cual es una característica deseable. Para el diseño del filtro se utilizó la herramienta de procesamiento de señales (Signal Processing Toolbox) del programa Matlab. El filtro se diseñó como un filtro de ventanas de orden 35, y los parámetros de diseño fueron una frecuencia de corte de 5 Hz, obtenida a partir del análisis en frecuencia de la señal y una frecuencia de muestreo de 122 Hz.

Una vez diseñado el filtro y simulado con datos obtenidos de los sensores, se probó el filtro físicamente. La Figura 5.4 muestra los resultados de las pruebas en tiempo real, donde se puede comparar tanto en tiempo como en frecuencia la diferencia entre la señal original y la filtrada. Como se puede observar en el análisis de frecuencia de la señal ya filtrada, en la banda de corte existe una gran atenuación, que resultó ser de 30 db, suficiente en este caso para eliminar el ruido. La metodología de diseño del filtro se muestra a detalle en el apéndice A.6.

El filtro diseñado se implementó en el microprocesador de la IMU, de manera que la señal se pudo introducir filtrada al ciclo de control. El ruido anteriormente existente era de tal magnitud que dificultaba, si no es que imposibilitaba el funcionamiento del control. Se implementó un filtro tanto a la señal de velocidad angular como a la de posición angular. Dado que el patrón de ruido y la frecuencia de corte son iguales para las dos señales,

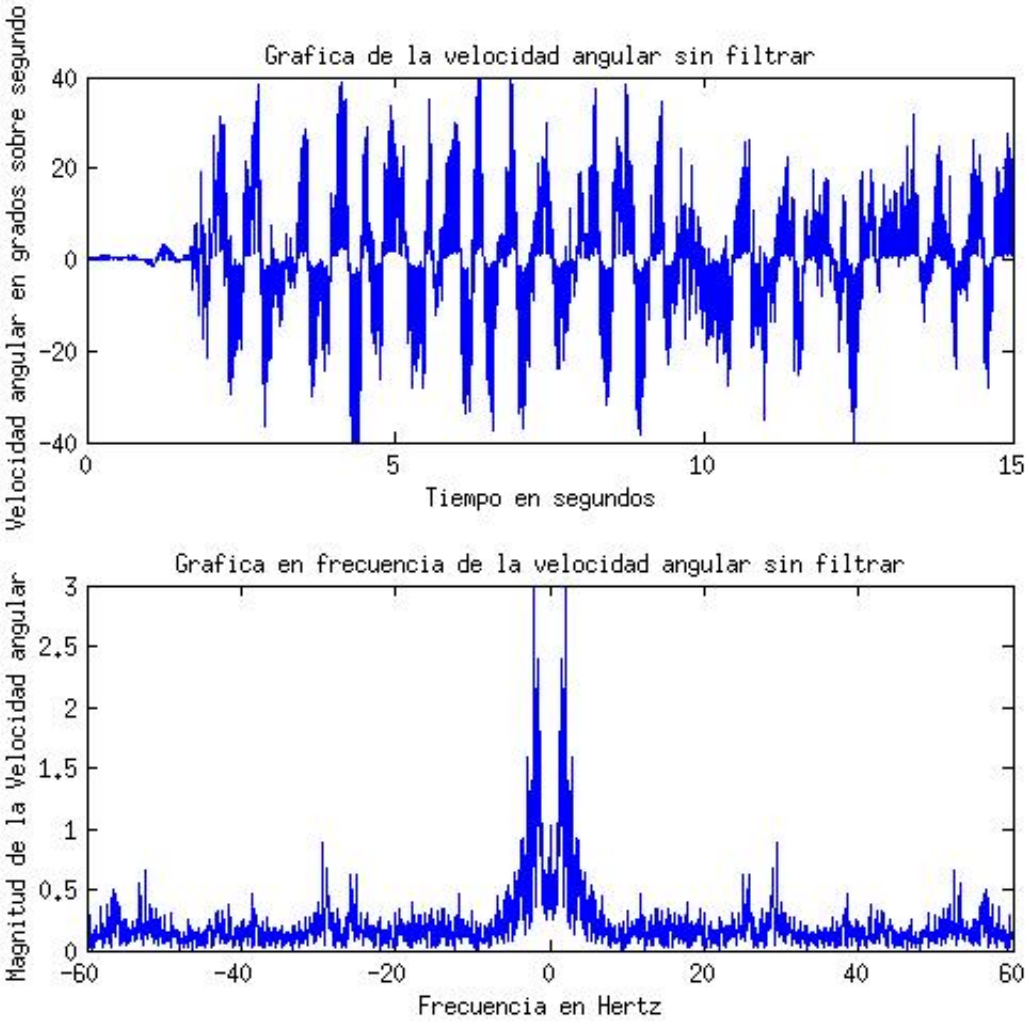


Figura 5.3: Gráfica de la señal de velocidad angular obtenida de la IMU sin filtrar en el dominio del tiempo (arriba) y en el de la frecuencia (abajo).

se empleó el mismo tipo de filtro, haciendo de esta forma que se filtraran simultáneamente ambas velocidades angulares y posiciones angulares.

5.3. CONTROL

Para aumentar la velocidad de muestreo reduciendo los caracteres enviados de la IMU al Arduino, se decidió trasladar el programa del algoritmo de control al *microcontrolador* de la IMU. De esta forma, en lugar de enviar todos los datos de posiciones y velocidades angulares al Arduino para que éste produjera las señales PWM para los motores, se modificó de tal forma que la IMU enviara las direcciones y los valores analógicos para generar los PWM de los motores al Arduino, y este último sólo se utilizó para generar

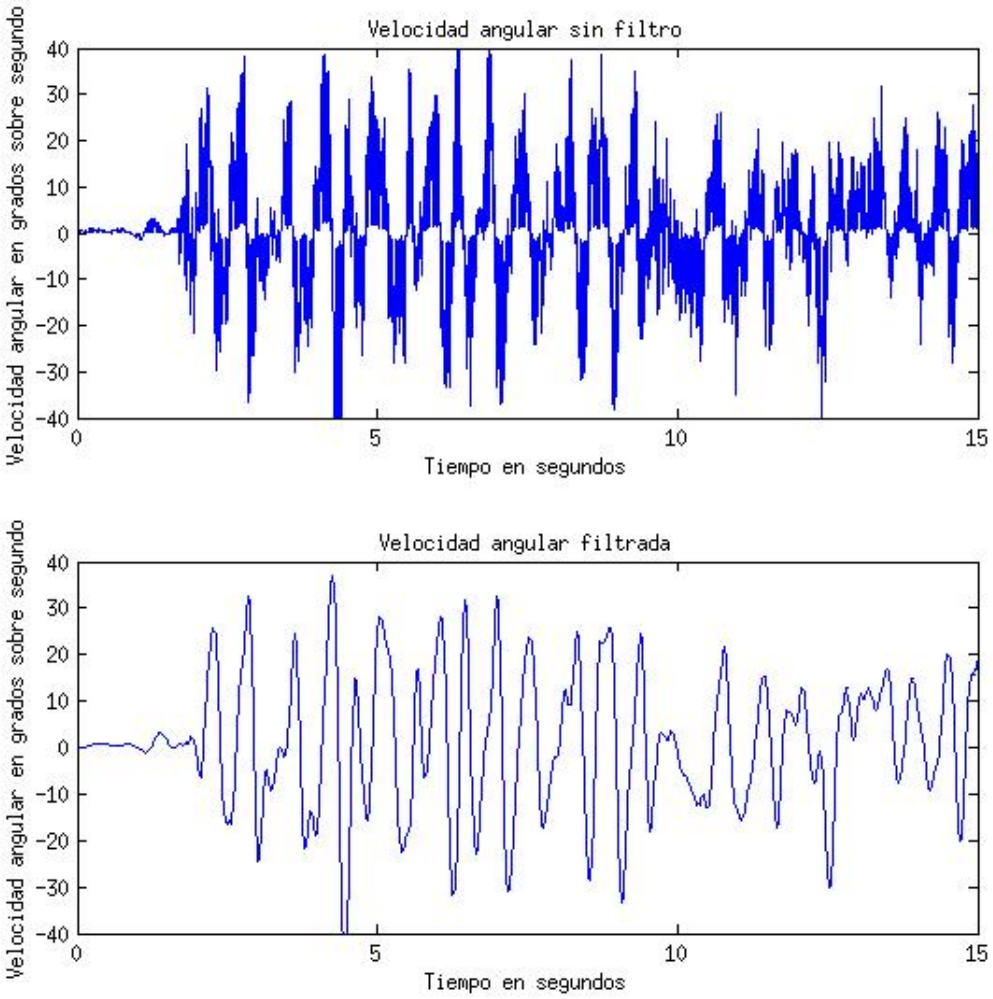


Figura 5.4: Comparación de la señal de velocidad angular obtenida de la IMU sin filtrar (arriba) y filtrada (abajo).

las salidas digitales y de PWM para los motores. Todo lo anterior se realizó con el propósito de aumentar la velocidad de procesamiento, dado que el *microcontrolador* de la IMU es más rápido que el del Arduino, siendo el *microcontrolador* de la IMU capaz de procesar a hasta 60 millones de instrucciones por segundo (Mips) y el del Arduino solamente a 16 Mips. El *microcontrolador* Atmega328 utilizado en el Arduino puede manejar hasta 20 Mips con un oscilador de 20 MHz, sin embargo, la tarjeta cuenta sólo con un oscilador de 16 MHz.

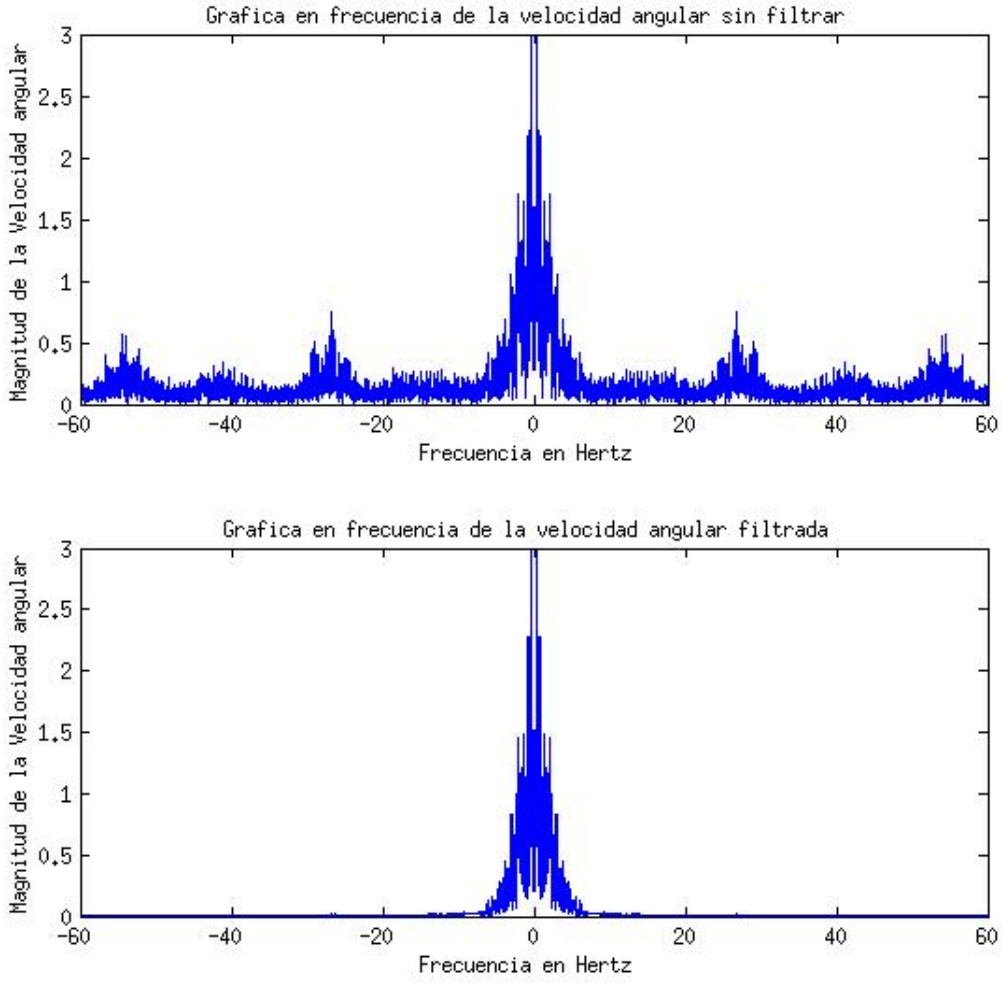


Figura 5.5: Comparación de la señal de velocidad angular obtenida de la IMU en el dominio de la frecuencia sin filtrar (arriba) y filtrada (abajo).

5.4. VELOCIDAD DE RESPUESTA DEL ROBOT

Un parámetro de gran importancia para el funcionamiento del robot es la velocidad de respuesta de todo el sistema. Para que el robot pueda equilibrarse sobre la esfera, éste debe ser capaz de responder en cuestión de milisegundos a los cambios de su inclinación. Para asegurar esto fue necesario analizar la velocidad de respuesta de los diferentes componentes del robot.

5.4.1. Ciclo de control

El tiempo de cómputo del ciclo de control resultó ser relativamente bajo, menor a 1 ms. Este tiempo se obtuvo del hecho de que el programa completo desde la adquisición de los datos de los sensores hasta el algoritmo de control resultó ser de unas 500 líneas, y el *microcontrolador* de la IMU tiene una velocidad de procesamiento de 60 Mips. Sin embargo, para obtener las salidas finales de dirección y PWM de los motores es necesaria la comunicación entre la IMU y el Arduino, siendo este un proceso que toma aproximadamente 8.2 ms debido a la frecuencia de muestreo de 122 Hz, por lo que al final la comunicación establece el límite de tener un máximo de 122 ciclos de control por segundo.

5.4.2. Respuesta de los motores y el MCP

Para conocer la velocidad de respuesta de los motores se utilizaron *encoders* magnéticos de efecto Hall, específicamente el integrado A013MU de la compañía Austria Microsystems. El tiempo de respuesta de los motores junto con los MCP resultó ser de aproximadamente 7 ms para que el eje del motor empezara a moverse desde el reposo . Los MCP cuentan con un filtro analógico a la entrada para transformar el PWM en una señal analógica de CD, el cual agrega un retardo de 18.2 ms. Agregando este retardo al tiempo de respuesta del integrado utilizado como controlador del motor a pasos y el tiempo de respuesta del motor mismo, se obtuvo que los motores tardan aproximadamente 90 ms en llegar a la velocidad deseada.

Para conocer la velocidad de respuesta de los MCP se hizo uso de un osciloscopio digital y señales escalón como entrada a los circuitos. La gráfica mostrada en la Figura 5.6 permite observar la velocidad de respuesta del motor junto con el MCP. El retraso entre la velocidad deseada y la real contempla los retrasos de la MCP más los del motor. De esta manera se puede conocer el tiempo de retraso desde que llega la señal de la velocidad deseada hasta que el motor la alcanza. Las pequeñas oscilaciones que tiene la velocidad real son generadas por la electrónica de control y por la baja amplitud que tienen no son consideradas un problema para la implementación del control.

5.4.3. Velocidad de adquisición de los datos de los sensores

Se llevaron a cabo pruebas para determinar el error existente entre el ángulo obtenido por la combinación de los datos de los sensores y el ángulo real de inclinación. Se encontró que el error era mínimo, pero desafortunadamente se descubrió la existencia de un retraso en la obtención del ángulo de inclinación de 410 ms. Esto se puede verificar claramente en la Figura 5.7, en donde se encuentran trazadas la velocidad angular, la inclinación obtenida

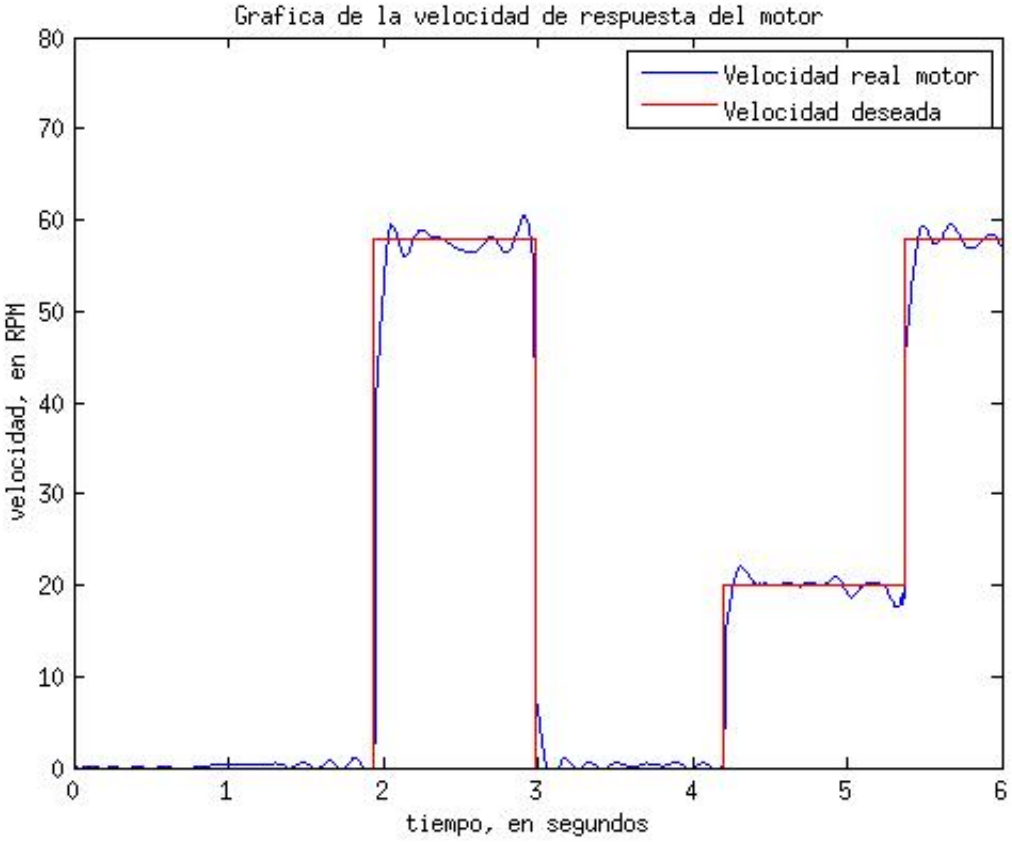


Figura 5.6: Gráfica de la señal de entrada al MCP junto con la respuesta de velocidad del motor.

de la IMU y la inclinación obtenida de la integración directa de la velocidad angular.

Se pueden observar en la Figura 5.7 las señales de la velocidad angular y la posición angular de la IMU simultáneamente. Para medir el retraso se realizó la integral directa de la señal de velocidad angular, la cual también se encuentra graficada. Se puede observar que cuando la velocidad angular decrece, la integral directa también empieza a decrecer, lo cual es lo esperado en la señal de posición angular de la IMU. Sin embargo lo que se obtuvo es que la señal de posición angular se mantiene en su mismo valor. Más adelante la señal de posición angular aumenta negativamente de forma muy drástica y alcanza a la integral directa. La diferencia de tiempo entre el cambio de valor de la integral directa y la señal de posición angular de la IMU, es un retraso dado aparentemente por el algoritmo de combinación de los datos de aceleración y velocidad angular para la obtención de la inlicnación en el microprocesador de la IMU.

Este retraso tan grande dificulta el correcto funcionamiento del algoritmo de control. Para tratar de arreglar esto, se buscaron alternativas para la

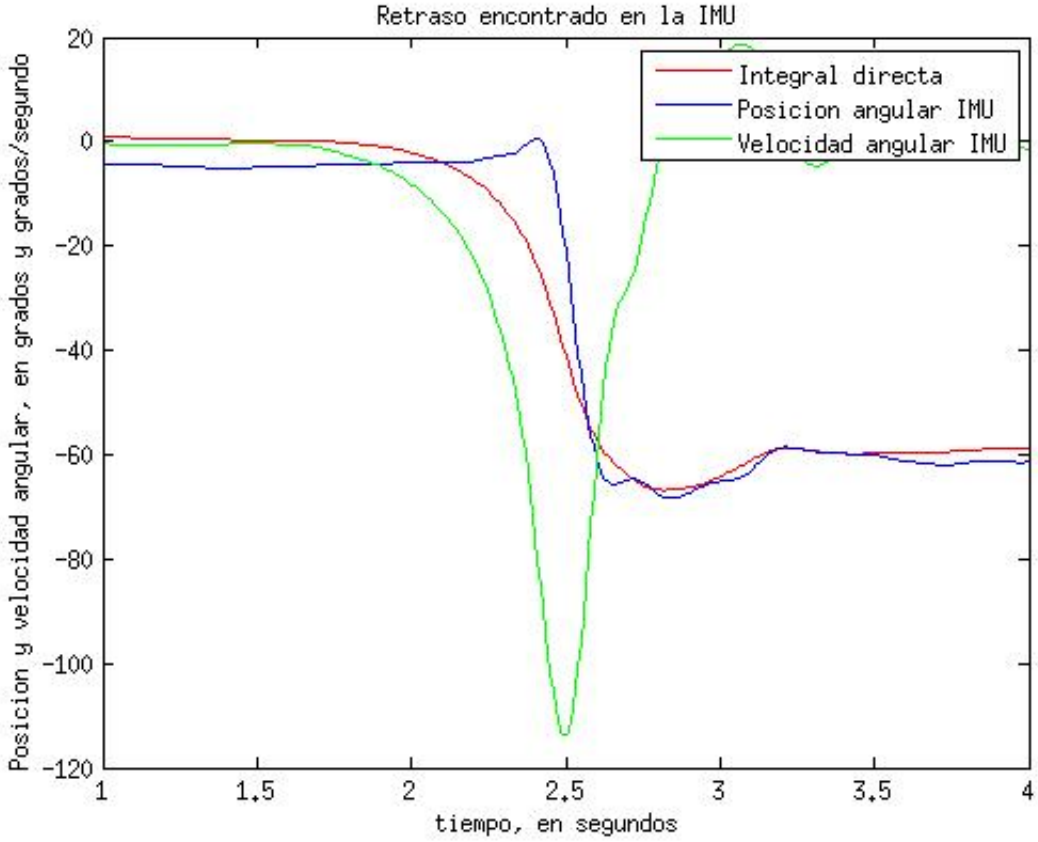


Figura 5.7: Gafica de la inclinación y velocidad angular del cuerpo del robot.

obtención del ángulo de inclinación a partir de los datos de aceleración y velocidad angular. Una de las opciones encontradas fue un algoritmo basado en *cuaterniones* para la obtención de dichos ángulos. Se trató de implementar este algoritmo, el cual mejoró un poco el retraso de la obtención de los datos de inclinación, pero introdujo un offset variable a la referencia del sistema, causando que el sistema no tuviera una referencia estable.

Capítulo 6

Conclusiones y Trabajo a Futuro

6.1. CONCLUSIONES

Se logró construir un robot móvil robusto que puede servir como base para otros sistemas, y sobre el cual se pueden seguir haciendo pruebas para mejorar su desempeño, siendo relativamente fácil su programación para cambiar su funcionamiento en caso de requerirse. El equilibrio logrado sobre la esfera fue dentro de un pequeño intervalo angular y por cortos periodos. Este desempeño fue menor al esperado, por lo que para que el robot pudiera mantener el equilibrado por más tiempo y recuperarse de inclinaciones mayores, el robot requería de ayuda.

Lo anterior se pudo deber a diferentes motivos:

- Retraso en la obtención de los datos de inclinación del robot, los cuales a su vez provocaban un retraso en las acciones de control e impedían el correcto balanceo del robot sobre la esfera.
- Disminución del par de los motores debido al uso de *microstepping*.

Por otro lado, se logró una buena precisión de movimientos sobre superficies planas sin la esfera, con lo cual, teniendo un buen algoritmo de control y arreglados los posibles problemas mencionados, debería ser posible el mejorar el balanceo sobre la esfera.

Otro problema que se tuvo en general fue el hecho de que no fue sino hasta el final, una vez que todo estaba construido, que empezaron a surgir varios pequeños errores como fallas en componentes electrónicos y conexiones, falsos contactos, ruido electrnico etc., que no eran evidentes durante la fase de pruebas por módulos o partes del robot, pero sí afectaban en el funcionamiento del robot completo con todos los módulos conectados, lo cual retrasó bastante el trabajo por el hecho de tener que volver a desarmar el robot, detectar exactamente el problema, arreglarlo y volver a probarlo sobre el sistema completo para comprobar su debido funcionamiento.

Por todo lo anterior, se concluyó que para desarrollar un robot como el propuesto y lograr el balance deseado se requiere un mayor tiempo de trabajo y la obtención de elementos que cumplan con las especificaciones necesarias, como por ejemplo en nuestro caso, sensores inerciales cuyo algoritmo de obtención de la posición angular no tenga retrasos tan grandes.

6.2. TRABAJO A FUTURO

El presente sistema podría ser una buena alternativa para brindar movimiento omnidireccional a diversos tipos de robots, sobre todo, como ya se mencionó, a aquellos que trabajan en ambientes humanos, por lo que como trabajo a futuro quedan los siguientes puntos.

1. Dado que aparentemente el principal factor por el cual no funcionó el robot fue el retraso en el sensor, el primer trabajo importante es la implementación de un algoritmo de obtención de la inclinación del robot en tiempo real o con el menor retraso posible o cambiar los sensores y comparar desempeños.

2. Una vez que sea posible obtener los datos sin retraso, se pueden probar otros algoritmos de control como ubicación de polos, LQR, modos deslizantes, entre otros, para optimizar el problema del balanceo sobre la esfera.

3. Conseguido el balanceo sobre la esfera, se puede ahora empezar a trabajar sobre el seguimiento de trayectorias.

4. También se puede utilizar al robot como un ayudante para el transporte de cargas, dado que se requiere de muy poca fuerza para desplazar al robot, y éste se mantiene equilibrado por sí solo. De esa forma, el robot puede ser empleado para ayudar a mover objetos pesados y de proporciones grandes, para lo cual es necesario diseñar un controlador lo suficientemente robusto para lograr dichos objetivos.

5. Paralelamente a los puntos anteriores, es posible intentar cambiar los motores a pasos por otro tipo de motores, con el fin de comparar el desempeño de los robots con diferentes tipos de motores.

6. Utilizar el robot construido como una base móvil para diversas aplicaciones. Se puede hacer uso de las entradas analógicas en el MIC para tener un control externo del funcionamiento de la base móvil. Se tienen tres entradas analógicas disponibles por lo que, por ejemplo, las entradas podrían ser dos señales analógicas referentes a velocidad en las coordenadas x e y. Esto facilita mucho el convertir al robot no sólo en una base que se balancea y pueda seguir trayectorias, además de poder tener una altura similar a la de una persona y una base estrecha, sino convertirlo en un robot capaz de interactuar en espacios humanos.

7. Otro trabajo interesante sería ponerle inteligencia al robot, de tal forma que fuera capaz de ubicarse espacialmente por sí mismo, y con base en esto agregarle la capacidad de diseñar sus propias trayectorias dependiendo de las funciones que requiera realizar.

Apéndice A

A.1. FUNCIONAMIENTO DE LOS DISPOSITIVOS

A.1.1. Acelerómetro

Existen un gran número de soluciones para la medición de la aceleración, como son mediante el cambio de capacitancia, masas con cristales piezoeléctricos que al aplicarles una fuerza causan cambios de voltaje o *piezoresistivos* que cambian su resistencia, el cambio del campo magnético mediante el efecto Hall, la ubicación de una masa caliente mediante sensores de temperatura, entre otros.

El sistema básico del acelerómetro consta de una masa de prueba que se encuentra sujeta de forma elástica a un marco rígido. En el momento en el que este marco rígido experimenta una aceleración, entonces la masa por acción de la inercia tendrá un movimiento relativo al marco rígido, haciendo que se deformen sus soportes, ya sea doblando o comprimiendo los mismos. La detección de la aceleración se logra ya sea mediante la observación directa del cambio de la posición de la masa de prueba respecto al marco rígido, o de forma indirecta mediante la medición de las deformaciones de los soportes elásticos de la masa. Por ejemplo, se puede hacer una medición indirecta haciendo uso del principio de la *piezoresistencia* que consiste en medir el cambio de la resistencia que sufre la estructura por la deformación de los soportes. En el mercado existen acelerómetros que utilizan tanto mediciones directas como indirectas. La mayoría de los acelerómetros en el mercado utilizan un sistema de lazo abierto, aunque también existen los de lazo cerrado[11].

Para los acelerómetros de medición directa existen varios métodos para la medición de la posición, entre los que se encuentran mediciones de cambios de capacitancia, cambios de inductancia, métodos ópticos y puntas de escaneo-prueba. En los *microacelerómetros*, el cambio de capacitancia es el método más común, dado que la lectura no es compleja y la precisión es grande. Las mediciones con cambio de inductancia son muy comunes en *macrosis-*

temas, pero en *microsistemas* aún no. Los sistemas de mediciones ópticas son relativamente nuevos, pero es posible que en el futuro tengan mayor éxito.

Existen varios tipos de condensadores eléctricos utilizados en el método de medición de cambios de capacitancia; a continuación se muestran algunos ejemplos.

Figura A.1: Se ilustra una variedad de configuraciones de condensadores eléctricos que pueden ser utilizadas para la detección del cambio de posición.

La configuración más utilizada es la de placas paralelas, en la cual la capacitancia puede variar con el cambio de la posición vertical de alguna de las placas, modificando así la distancia entre las mismas, o mediante el movimiento transversal de alguna de las placas, modificando de esta manera el área efectiva del condensador.

Existen varios métodos y circuitos para medir los cambios de la capacitancia, los cuales varían su complejidad puesto que algunos hacen uso de voltajes de corriente continua y otros de voltajes de corriente alterna para la comparación y medición. Entre los métodos utilizados para la medición de la capacitancia en acelerómetros se encuentran el método de amplificador de *transimpedancia* para capturar la corriente del condensador, inversor conmutado con condensadores, uso de seguidores de voltaje para medir el voltaje de salida de un condensador diferencial, etc.

Sin importar qué método de medición de los cambios de capacitancia se utilice, una vez conocido este cambio se puede determinar directamente cuál es la posición de la masa de prueba. Si es posible en todo momento conocer la posición de la masa de prueba en relación con el marco rígido además de sus características físicas y las constantes de la estructura que la sostienen, se puede conocer su aceleración mediante el uso de relaciones matemáticas, y por lo tanto la aceleración del sensor en general. Este es el principio básico del funcionamiento de un acelerómetro.

A.1.2. Giroscopio

El giroscopio es un dispositivo capaz de medir la aceleración rotacional. Existen varios tipos, como son los mecánicos que consisten en una rueda giratoria montada sobre una suspensión de cardán, los ópticos como el de fibra óptica o el de anillo láser que funcionan gracias al efecto de Sagnac,

los de *momento London* que se basan en un fenómeno de mecánica cuántica, los de estructuras vibratorias que se basan en un oscilador que genera un momento angular o lineal constante acoplado a un sensor de aceleración de Coriolis, entre otros. Estos últimos son los que serán de interés para este trabajo.

El principio utilizado en los giroscopios de estructuras vibratorias es el efecto de Coriolis, descrito por primera vez en 1836 por el científico francés Gaspard-Gustave Coriolis. El efecto de Coriolis surge debido a la fuerza inercial o ficticia de Coriolis, la cual aparentemente actúa sobre un objeto cuando éste se encuentra en movimiento en un sistema de referencia rotatorio y por tanto no inercial, y es observado desde este mismo sistema de referencia. El efecto consiste en la aparición de una aceleración sobre el cuerpo en movimiento en dicho marco de referencia rotatorio, la cual es siempre perpendicular al eje de rotación del sistema y a la dirección de la velocidad del cuerpo.

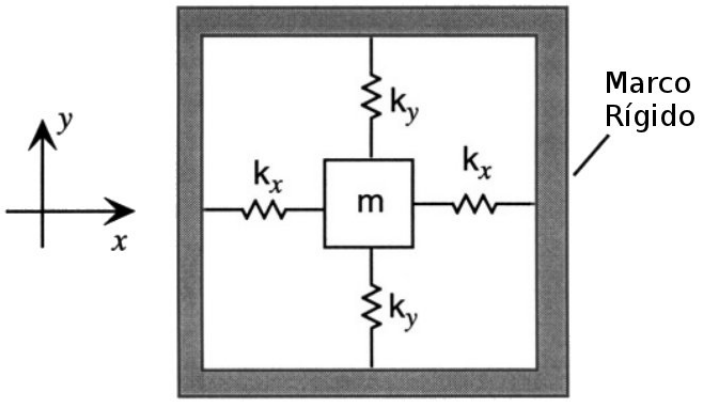


Figura A.2: Modelo simplificado del funcionamiento de un giroscopio. Se considera que el marco rota alrededor del eje z.

En el giroscopio, al igual que en el acelerómetro, se hace uso de una masa de prueba, la cual se encuentra sujeta a un marco rígido mediante eslabones elásticos, los cuales pueden ser modelados como resortes con una constante elástica conocida dependiendo del material. La principal diferencia es que en el giroscopio la masa de prueba se hace vibrar por medio de un oscilador. Se puede observar un diagrama representativo en la Figura A.2. De la misma forma que en el acelerómetro, el método que se utiliza es el de medir la posición de la masa de prueba, la cual cambia con respecto a un marco rígido cuando éste experimenta una aceleración. Como se puede conocer la posición y movimiento de la masa de prueba, se busca una manera en la que este movimiento brinde una descripción del movimiento rotatorio que experimenta el marco rígido, para lo cual se utiliza la fuerza de Coriolis. Dado que es necesario que el cuerpo, en este caso la masa de prueba, se encuentre en movimiento respecto al sistema de referencia para que exista una fuerza de Coriolis, la técnica que se utiliza es hacer vibrar la masa de prueba con

una amplitud fija y de forma perpendicular a el eje de rotación por medio de un oscilador, de manera que la fuerza de Coriolis inducirá movimiento en una dirección perpendicular tanto al eje de rotación como al eje del movimiento constante.

Como se puede observar en la Figura A.2, la masa m está suspendida por tres resortes. Considerando que el movimiento de oscilación en el plano sea de amplitudes pequeñas, se pueden considerar ambos movimientos como independientes, acoplados solamente a través de la fuerza de Coriolis y de la aceleración angular.

No existe una necesidad fundamental de que el giroscopio haga uso de diferentes masas y resortes para cada eje. Cualquier modo vibrador para la masa puede ser utilizado tanto para medición como medio de generación de movimiento en el giroscopio. Lo único necesario es que esté diseñado de tal manera que la fuerza de Coriolis puede acoplar ambos movimientos y que este acoplamiento genere un término proporcional a la rotación.

Siguiendo alguno de los métodos explicados en el acelerómetro sobre la medición de la posición de la masa de prueba o cualquier otro método existente, se puede conocer el término proporcional a la aceleración angular. Este es el método más comúnmente utilizado en giroscopios comerciales para conocer la aceleración angular.

A.1.3. Magnetómetro

Existen dos tipos de magnetómetros:

- Magnetómetros escalares. Son aquéllos que miden la fuerza total del campo magnético al que están sujetos, pero no es posible medir la dirección de dicho campo. Ejemplos de este tipo son los de precesión protónica, los de efecto Overhauser y los de absorción atómica que utilizan vapores alcalinos como cesio, potasio o helio.

- Magnetómetros vectoriales.- Miden la componente del campo magnético en una dirección específica relativa a la orientación del sensor. Ejemplos son el Dispositivo Superconductor de Interferencia Cuántica (SQUID por sus siglas en inglés), el de núcleo saturado o fluxgates, el *magnetoresistivo*, entre otros.

Los utilizados en el presente proyecto son los magnetómetros *megnetoresistivos*, los cuales utilizan un material que cambia su resistencia eléctrica dependiendo del campo magnético al que estén sujetos. En específico utilizan un material que presenta una *magneto-resistencia anisotrópica* (AMR por las siglas en inglés de Anisotropic Magneto-Resistance), es decir, un material cuya resistencia eléctrica varía dependiendo de la dirección de la corriente eléctrica y la del campo magnético. La resistencia se maximiza cuando las direcciones tanto de la corriente como del campo magnético aplicado son paralelas. La *magnetoresistencia* depende en gran medida de una característica

de los semiconductores llamada movilidad eléctrica, la cual describe qué tan rápido se mueven los portadores de carga a través de un metal o un semiconductor sujeto a un campo, en este caso electromagnético. Al aplicar dicho campo, los portadores de carga se mueven con una velocidad promedio, llamada velocidad de deriva, vd. En el caso de la aplicación de solamente un campo eléctrico, se tiene una relación como la siguiente:

$$\mu = \frac{v_d}{E}$$

Pero en un caso *magnetoresistivo* sencillo, la velocidad es igual a:

$$v = \frac{\mu}{1 + (\mu B)^2} \left(E + \mu E \times B\right)$$

donde: v = velocidad del portador de carga
μ = movilidad eléctrica
E = campo eléctrico
B = campo magnético

A.2. PROCESO DE DIGITALIZACIÓN

Los sensores utilizados, tanto el acelerómetro como el giroscopio, dan como salida una señal analógica. Entiéndase como señal analógica una señal continua definida para cualquier valor del dominio de los números reales y que sus valores se encuentran en un intervalo continuo. Lamentablemente este tipo de señales no puede ser procesado por un microprocesador, por lo que se requiere transformarla en una señal digital.Entiéndase por señal discreta una señal definida sólo para ciertos puntos del dominio de los reales y comúnmente con valores en el dominio de los enteros, donde estos puntos no necesitan ser equidistantes. Una señal digital es una señal discreta que solo puede tener un conjunto discreto de valores, normalmente proviene de una señal discreta cuyos valores se han aproximado a los valores que la señal digital puede tomar. Dado que se requiere una señal digital para su procesamiento, la señal analógica sufrirá una conversión a una señal digital. En las Figuras A.3 y A.4 se muestra una señal en su forma analógica y su correspondiente forma digital.

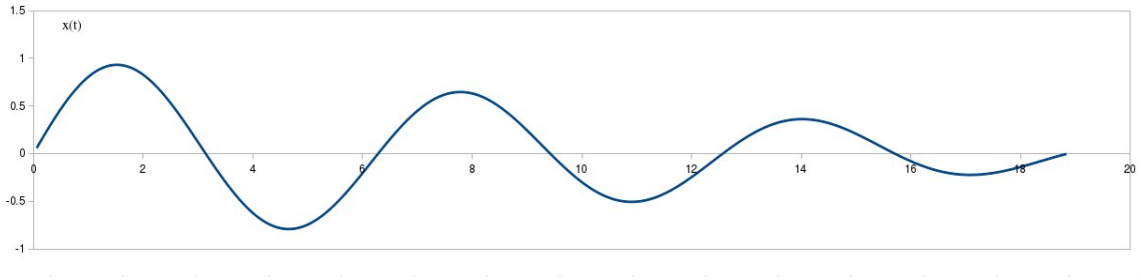


Figura A.3: Señal en su forma analógica.

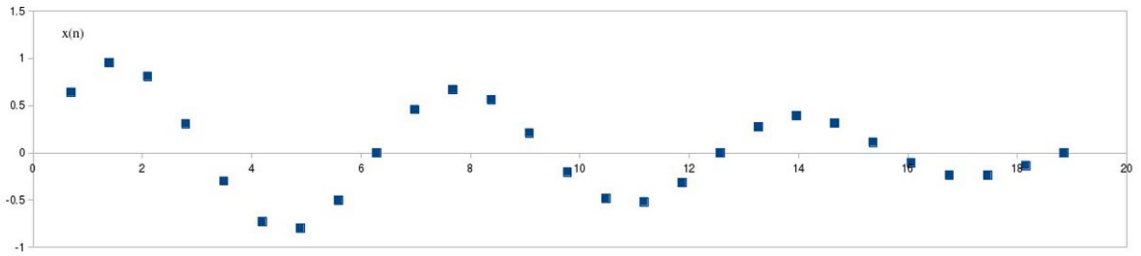


Figura A.4: Señal anterior en su forma discreta.

Para llevar a cabo la conversión se utiliza un proceso llamado muestreo, el cual es el proceso de adquisición de muestras a intervalos específicos de tiempo de una señal analógica. Las muestras obtenidas tienen una magnitud diferente y son *cuantizadas* a valores digitales, los cuales están basados en una palabra de L bits, donde L es la longitud de la palabra. La *cuantización*

es el proceso mediante el cual a cada muestra discretizada se le otorga un valor digital de L bits. El proceso de *cuantización* involucra un error, el cual depende del número de bits empleados en la palabra digital. Para llevar a cabo la *cuantización*, hay que hacer una aproximación a los valores analógicos correspondientes, que puede ser realizada por truncamiento o por redondeo.

Como se mencionó anteriormente, la señal digital tiene valores solamente para ciertos puntos del dominio, la distancia entre estos puntos está dada por la frecuencia de muestreo, f_s. Esta frecuencia no puede ser elegida arbitrariamente, sino que se encuentra en función de la frecuencia de la señal analógica que será muestreada. De acuerdo con el Teorema de Nyquist o de Shannon, para que a partir de una señal digital se pueda reconstruir la señal analógica original es necesario que la frecuencia de muestreo sea por lo menos mayor al doble de la frecuencia máxima de la señal analógica. Si esto no se lleva a cabo, la señal digitalizada no representará correctamente a la señal original. En las Figuras A.5, A.6, A.7 y A.8 se muestra una señal analógica, su digitalización con una frecuencia de muestreo menor al doble de la frecuencia de la señal analógica, un poco mayor al doble, y mucho mayor al doble de la señal analógica.

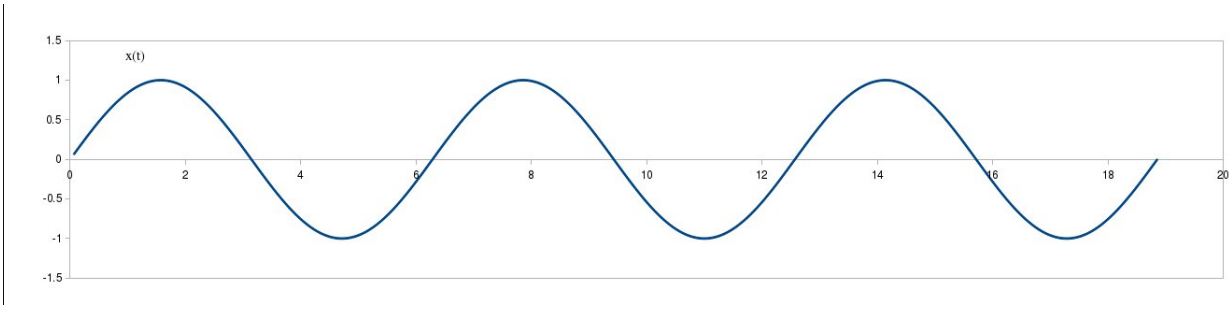


Figura A.5: Señal analógica original.

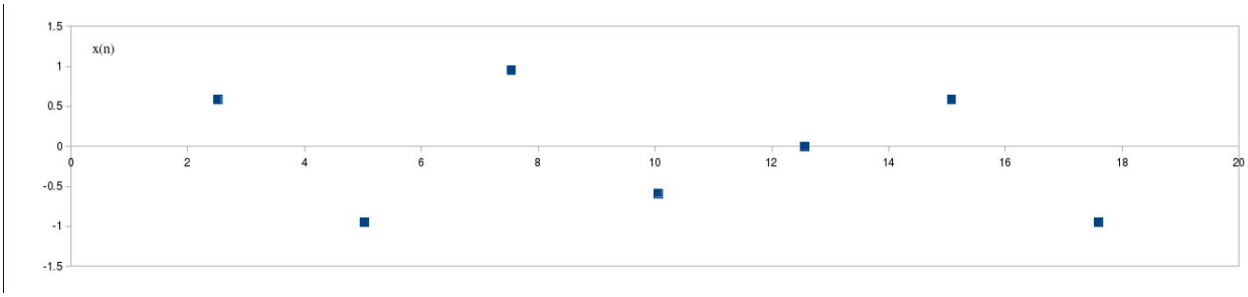


Figura A.6: Señal discreta donde $f_s < 2f$.

Como se puede observar en las imágenes anteriores, existen problemas cuando la frecuencia de muestreo no es la correcta. Cuando ésta es menor a la exigida por el Teorema de Nyquist, se produce el fenómeno conocido como *aliasing* [9], el cual es la superposición espectral de las repeticiones consecu-

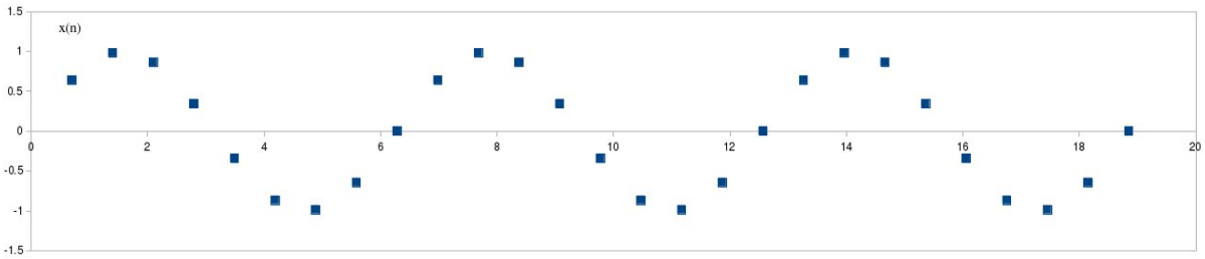


Figura A.7: Señal discreta donde $f_s > 2f$.

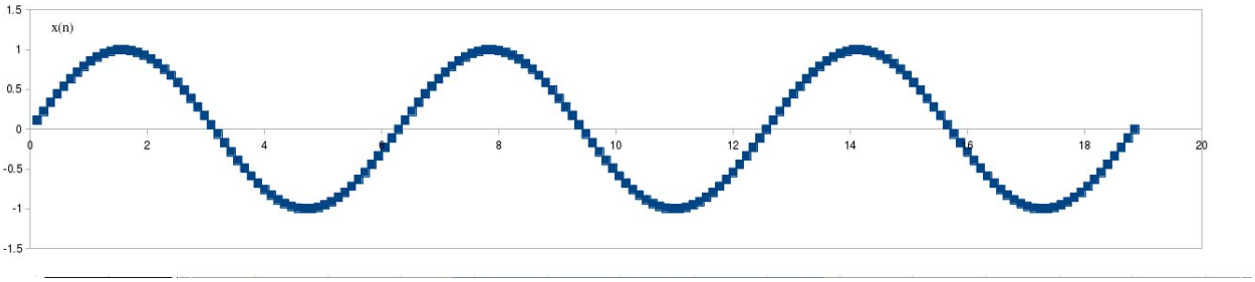


Figura A.8: Señal digital donde $f_s >> 2f$.

tivas en el espectro de la señal digital. Este espectro puede ser identificado a partir de una transformada de Fourier de la señal muestreada.

En una señal digital en la cual la frecuencia de muestreo es mucho mayor al doble de la frecuencia máxima de la señal analógica, se puede ver que se obtiene prácticamente la señal original, casi no existe pérdida de datos y sí es posible reconstruir la señal original analógica, sin embargo, esto genera una cantidad excesiva de muestras ocasionando que un sistema digital sea insuficiente para almacenarlas y procesarlas en tiempo real. A este error de digitalización se le conoce como *sobremuestreo*.

La amplitud de la señal una vez muestreada, o *discretizada*, pertenece al dominio de los reales; el proceso de *cuantización* consiste en representar estas amplitudes en un conjunto de valores no continuos, es decir, ya no en el dominio de los reales sino en el dominio de los enteros. Dado que se trasladan valores de una escala infinita a una escala finita, se producen errores en la *cuantización*. La mayoría de las veces estos valores se representan en forma binaria; también se puede decir que son codificadas. Se puede conocer la longitud de la palabra binaria si se conoce el número de niveles que se requieren para la *cuantización*, de manera que si el número de niveles N es una potencia exacta de dos, la longitud de la palabra puede ser obtenida mediante:

$$L = log_2 N$$

y si no es una potencia exacta de dos:

$$L = Int(log_2(N)) + 1.$$

A.3. OBTENCIÓN Y FILTRADO DE LOS DATOS DE LA IMU

Como ya se mencionó, se utilizó la Ultimate IMU para obtener los datos de inclinación del robot respecto a la vertical mediante un acelerómetro y un giroscopio, así como los datos de la orientación del robot mediante un magnetómetro.

La IMU ya está programada para obtener directamente los datos del acelerómetro en *fuerzas g* (1 g = 9.81 m/s2), los datos del giroscopio en °/s y los datos del magnetómetro en gauss (G), todos los datos anteriores para los tres ejes coordenados x, y y z. A continuación se presenta a grandes rasgos la transformación de la señal digital a un dato conocido ya sea de aceleración, velocidad angular o intensidad de campo magnético.

Para el acelerómetro:

1. Se establece el rango del acelerómetro, que puede ser de ±2, ±4, ±8 ó ±16 g; para el presente caso se utilizó ±4 g.

2. El sensor entrega un valor digital de 16 bits en formato complemento a dos por medio de un protocolo serial I^2C.

3. Se toman mediciones estando el sensor estático para obtener el error estático del sensor, que en este caso puede ser de ±0.04 g para los ejes x y y, y de ±0.08 g para el eje z. En especial, para el eje z, es necesario calibrar el valor de la aceleración estática en 1 g debido a la acción de la gravedad.

4. Al dato obtenido se le resta el error estático, se convierte a entero y se le aplica el factor de sensibilidad, que para el rango de ±4 g es de 128 LSB/g, por lo que se divide el valor entero entre la sensibilidad para obtener el valor en *fuerzas g*, para cada eje.

Para el giroscopio:

1. Primero es necesario establecer algunos parámetros como la frecuencia del reloj interno, la velocidad de muestreo, las interrupciones y la frecuencia de comunicación entre el sensor y el *microcontrolador*.

2. Hecho lo anterior, se obtienen los datos en crudo de temperatura y velocidad angular en cada eje del sensor por medio de una comunicación I^2C. Cada dato es de 16 bits en formato complemento a dos.

3. En este caso no se hace uso del dato de la temperatura.

4. Se toman mediciones estando el sensor estático para obtener el error estático del sensor, que en este caso puede ser de ±40 °/s.

5. Al dato en binario de las velocidades angulares se le resta el error estático, es convertido a un entero, y dado que la sensibilidad del giroscopio es de 14.375 LSB/ °/s, sólo se divide el dato obtenido de la resta entre la sensibilidad, obteniendo así la información en °/s para cada eje.

Para el magnetómetro:

1. Se establece la velocidad de actualización de datos así como el rango del sensor que puede ser ±0.7, ±1, ±1.5, ±2, ±3.2, ±3.8, ±4.5 y ±6.5 G; para el robot se utilizó ±1 G.

2. Se obtiene un dato digital de 16 bits en formato complemento a dos, el cual se transforma en un dato de tipo entero.

3. Finalmente, para el rango establecido se tiene una sensibilidad de 1300 LSB/G, por lo que sólo es necesario dividir el entero entre la sensibilidad para obtener el valor en G, para los tres ejes.

Dado que los datos obtenidos desde los sensores son de aceleración, velocidad angular e intensidad de campo magnético, es necesario procesar los datos para obtener la información de la inclinación del robot, que es de mayor interés más que su aceleración o velocidad angular, así como para obtener su orientación con respecto al norte magnético en lugar de la intensidad del campo magnético en cada eje.

Para la obtención de la inclinación del robot, se lleva a cabo una combinación de los datos del acelerómetro con los del giroscopio mediante un filtro basado en el filtro de Kalman pero muy simplificado, que utiliza una ganancia constante en lugar de depender del ruido de medición, y no utiliza un modelo del sistema.

El proceso para la obtención de la inclinación del robot es el siguiente:

1. Se obtienen los datos del acelerómetro en los tres ejes, los cuales son las componentes en x, y y z del vector de fuerza inercial del sistema del acelerómetro Racc = [RxAcc, RyAcc, RzAcc], que en el caso en el que el robot esté en reposo, resultaría ser el vector de la fuerza de gravedad.

2. Se normaliza el vector obtenido para asegurar que su módulo sea de uno, facilitando los cálculos.

3. Se crea un vector Rest = [RxEst, RyEst, RzEst,], que es el que tendrá la información de la estimación de los ángulos de inclinación del robot. Éste se inicializa con los valores iniciales de Racc.

4. Del giroscopio se obtienen las velocidades angulares en °/s, alrededor de los tres ejes.

5. El vector de fuerza inercial del giroscopio Rgyro = [RxGyro, RyGyro, RzGyro] se inicializa con los valores de Racc.

6. Con los datos de Rest se calcula el ángulo anterior, mediante las siguientes expresiones:

$$A_{planoXZ}(n-1) = AXZ(n-1) = arctan(RxEst(n-1)/RzEst(n-1))$$

$$A_{planoYZ}(n-1) = AYZ(n-1) = arctan(RyEst(n-1)/RzEst(n-1))$$

donde n representa el número de ciclo, A_{XZ} representa al ángulo alrededor del eje y (pitch) y A_{YZ} representa al ángulo alrededor del eje x (roll). Para facilitar la obtención de los ángulos reales, se puede hacer uso de la función atan2(a,b), la cual es una función que devuelve el valor del ángulo contemplando el cuadrante en el que se encuentran las componentes a y b. El ángulo alrededor del eje z (yaw) no es necesario ya que se calculará posteriormente con los magnetómetros.

7. Se calcula la diferencia angular entre la medición pasada y la actual mediante los datos de velocidad angular, simplemente multiplicando la velocidad angular por el intervalo de tiempo transcurrido.

8. Se calcula el ángulo actual sumando los ángulos obtenidos en el punto 6. con los obtenidos en el punto 7, con lo cual se obtiene A_{YZ}(n) y A_{XZ}(n).

9. Se calculan las componentes de Rgyro mediante las siguientes expresiones:

$$RxGyro = \frac{1}{\sqrt{1 + cot^2(A_{XZ}(n)) * sec^2(A_{YZ}(n))}}$$

$$RyGyro = \frac{1}{\sqrt{1 + cot^2(A_{YZ}(n)) * sec^2(A_{XZ}(n))}}$$

$$RzGyro = Signo(RzGyro) * \sqrt{1RxGyro^2RyGyro^2}$$

donde Signo(RzGyro) = Signo(RzEst(n-1)). Hay que tener cuidado cuando RzEst(n-1) tenga valores muy cercanos a 0, ya que como se utiliza como referencia en el cálculo de A_{XZ} y A_{YZ}, pueden resultar errores grandes ocasionando cálculos incorrectos al evaluar funciones trigonméticas.

10. Luego, se combinan Racc y Rgyro para obtener Rest actual, mediante la siguiente expresión:

$$Rest(n) = \frac{(Racc + Rgyro * wGyro)}{(1 + wGyro)}$$

donde wGyro es el peso que se le quiera dar a los datos del giroscopio sobre los del acelerómetro, o dicho de otra forma, es la ganancia de lo datos del giroscopio, mientras que los datos del acelerómetro tienen una ganancia unitaria.

11. Finalmente se normaliza Rest, que dará los datos del tiempo anterior para el próximo ciclo.

Posteriormente, teniendo los datos de inclinación del robot, sólo falta obtener su orientación, la cual se obtiene a partir de los datos del magnetómetro de la siguiente forma:

1. Se obtienen los datos de intensidad de campo magnético del magnetómetro para los tres ejes bx, by y bz.

2. Dependiendo de los rangos de valores en los diferentes ejes, puede ser necesario mover la referencia de los datos para que tengan una media de cero, ya que dado que el cálculo del ángulo se basa en la función atan2(a,b), es necesario conocer el cuadrante en el que se encuentran los valores, y para ello es necesario tener valores tanto positivos como negativos.

3. Dependiendo de la ubicación geográfica, se calcula el ángulo declinación magnética, que es el ángulo entre el norte magnético local y el norte geográfico del planeta.

4. Si el sensor se encuentra inclinado, la lectura de los datos cambia, por lo que también cambiaría la orientación. Para arreglar esto es necesario compensar la orientación con respecto a la inclinación del sensor, es decir, con base en los ángulos A_{XZ} (pitch) y A_{YZ} (roll) calculados con anterioridad. Las expresiones para los valores en x e y compensados están dadas por las siguientes expresiones:

$$X_{compensado} = bx*cos(A_{XZ}) - by*sen(A_{XZ})*sen(A_{YZ}) + bz*cos(A_{YZ})*sen(A)XZ)$$

$$Y_{compensado} = by * cos(A_{YZ}) + bz * sen(A_{YZ})$$

5. Finalmente se calcula el ángulo con respecto al norte verdadero o azimut mediante la expresión:

$$Azimut = atan2(Y_{compensado}, X_{compensado}) + angulo\ de\ declinacion\ magnetica$$

obteniendo así la orientación del robot.[17]

La compañía que diseñó y manufacturó la Ultimate IMU provee una programación base para el funcionamiento de la IMU, la cual se modificó ligeramente con la intención de que la tarjeta funcionara específicamente para las necesidades del sistema. Se aumentaron indicadores para facilitar la separación de datos en el Arduino, y se removió el parámetro de tiempo, dado que es un dato que no es de utilidad para el procesamiento del control.

El sistema de comunicación de la IMU utilizada con el *microcontrolador* fue el serial, para lo cual se utilizó un cable FTDI cambiador de serial RS232

a TTL serial de 3.3 V para poder comunicar la computadora con el *microcontrolador*, ya que el protocolo RS-232 utilizado por la computadora hace uso de voltajes bipolares y el protocolo TTL utilizado por el *microcontrolador* usa un voltaje unipolar. Se diseñó un pequeño circuito que permitiera tener los valores de entrada necesarios para ingresar al modo de programación, así como un botón de Reset para la Ultimate IMU.

A.4. CONTROL DEL MOTOR DE PASOS

Se utilizaron motores de pasos para implementar el movimiento de la esfera en el robot, sin embargo el uso de éstos tiene sus problemáticas e inconvenientes. El circuito de manejo de los motores de pasos es más complejo que el circuito de manejo de los motores de CD, debido a que el motor de pasos tiene más bobinas y se deben de controlar mayor número de voltajes. Además, su movimiento es discontinuo debido a que existen pasos o posiciones establecidas en las que se puede encontrar el rotor. Las distintas velocidades generables en un motor de pasos están dadas por el tiempo de espera que hay entre estos pasos. Este movimiento discontinuo hace que el sistema no gire de forma suave, generando vibración y dificultando el control.

El método más común para controlar un motor de pasos es el de usar voltajes máximos o mínimos en las terminales de sus bobinas, es decir, alimentar con el voltaje del motor o con tierra, haciendo de esta manera que la intensidad de la corriente sea siempre la misma y que lo único que varíe sea la dirección de ésta en las bobinas. Esto permite que el motor pueda girar en una dirección deseada y a una velocidad controlada por el tiempo en el que se modifican los sentidos de las corrientes. Este método ocasiona que el movimiento no sea constante, dado que el rotor salta de una posición o paso hacia otro de manera no continua. Para obtener un movimiento más suave se hizo uso del método de *microstepping* que se explica más adelante. Dado que el movimiento deseado es un giro constante, fue necesario cambiar la forma en que se controlan las bobinas del motor de pasos, por lo que para el presente caso, en vez de utilizar corrientes máximas, se utilizó un rango variable de corrientes que permitieran al rotor girar de forma casi continua.

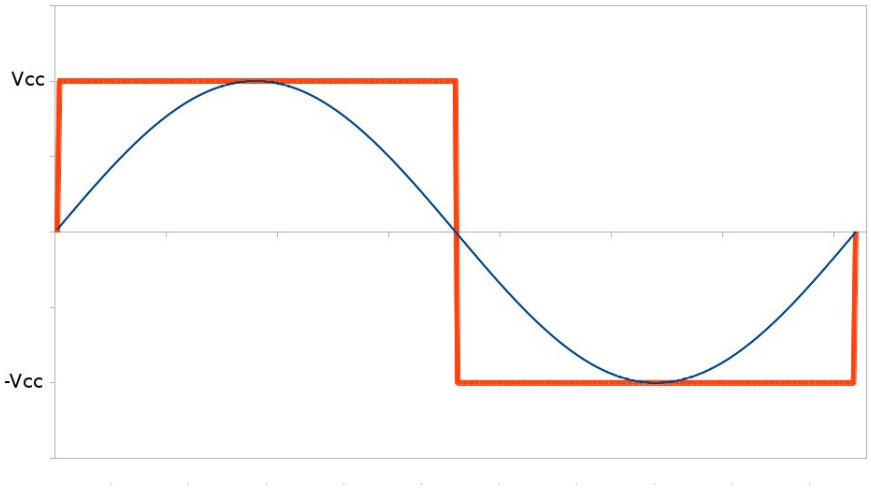


Figura A.9: Métodos de control de un motor de pasos. La señal cuadrada corresponde al método común, y la sinusoidal al de *microstepping*.

En la Figura A.9 se puede observar una gráfica del voltaje contra el tiempo

en las bobinas del motor de pasos para el método común de control, en la que se utiliza una señal cuadrada. Se puede observar también la forma de onda para el método de *microstepping*, el cual describe una onda sinusoidal. El uso de formas de onda sinusoidales en lugar de las cuadradas, permite que el rotor se desplace suavemente entre distintas posiciones antes de llegar a uno de los polos o pasos determinados. De esta manera el rotor gira de forma continua manteniendo el par generado constante. Es posible generar un movimiento constante debido que se cuenta con puntos intermedios y no solamente con los polos o pasos definidos. Si el voltaje de entrada simula una forma sinusoidal el comportamiento del motor es similar al de un motor de corriente directa.

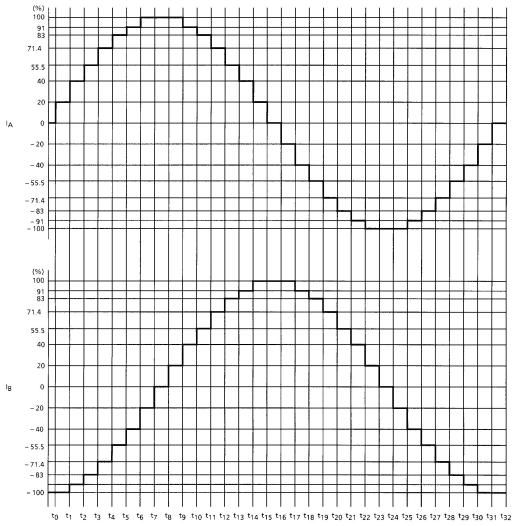


Figura A.10: Gráfica del voltaje contra el tiempo en las bobinas del motor de pasos.

En la Figura A.10 se muestran las señales sinusoidales que entran a las bobinas del motor; éstas se han escalonado en valores discretos que simulan la sinusoidal. Utilizar una sinusoidal no escalonada representaría un circuito más complejo, además de que los escalones otorgan una opción más, que consiste en que si el ciclo de la sinusoidal se interrumpe en un valor que no sea el máximo o mínimo y se mantiene el voltaje y la corriente fijos en el valor correspondiente, el rotor se detiene en un punto que no es el polo o paso determinado. Esto permite generar pasos intermedios a los pasos originales, con lo cual se puede aumentar la resolución del motor y se obtiene un movimiento más suave. Es por esto que el método es conocido como microstepping, porque se pueden generar pasos más pequeños, intermedios a los originales.

En la Figura A.11 se muestra una gráfica de los pasos dependiendo de las corrientes en cada bobina; las corrientes se encuentran en porcentajes. Cuando la corriente en una bobina se encuentra al 100 % en la otra bobina

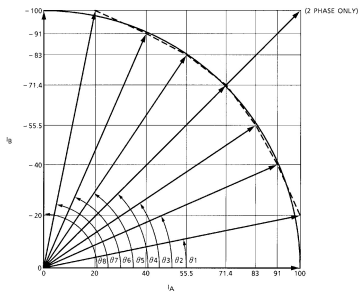


Figura A.11: Gráfica de los pasos dependiendo de las corrientes en cada bobina, las corrientes se encuentran en porcentajes.

está a 0 %. Estos puntos son los pasos originales o polos del motor. Todas las demás combinaciones de corrientes entre las dos bobinas generan pasos intermedios como se puede observar en la Figura. La magnitud del vector creado por las dos corrientes corresponde al par generado por el motor, y como se puede observar, si se hace uso de *microstepping* el par se mantiene constante a lo largo de los pasos del motor. Si ambas corrientes están a su máximo, el rotor irá a un punto intermedio entre los dos polos, pero con un par mayor.

El circuito con el que se puede implementar este método es un circuito *chopper*, o de cortado, el cual mediante el uso de un transistor y la bobina del motor, puede generar un control de corrientes y voltajes, alcanzando voltajes mínimo, máximo e intermedio. Un circuito básico de este modelo es el que se muestra en la Figura A.12. El control de la corriente está dado mediante el apagado y prendido del transistor, de manera que la corriente aumenta o disminuye. Se cuenta con una resistencia de medición de voltaje, la cual cierra el ciclo de control de corriente. Cuando el voltaje medido en la resistencia es mayor al de referencia, se interrumpe la corriente apagando el transistor; cuando el voltaje en la resistencia de medición es menor al de referencia, se enciende el transistor y se permite el paso de la corriente. De esta manera la corriente se mantendrá con oscilaciones pequeñas alrededor de la intensidad deseada.

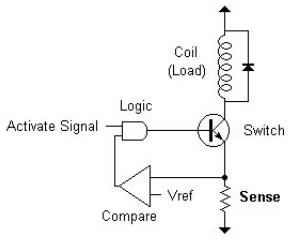


Figura A.12: Un circuito básico de control de corriente *chopper*.

En la Figura A.13 se muestra un circuito que funciona con la base de *chopping* o cortado de corriente; sin embargo este circuito tiene un arreglo

de cuatro transistores con configuración de puente H, de manera que se puede controlar el sentido de la corriente además de su intensidad. Haciendo uso de un convertidor digital a analógico se pueden producir las corrientes necesarias para generar la sinusoidal escalonada requerida para el *microstepping*.

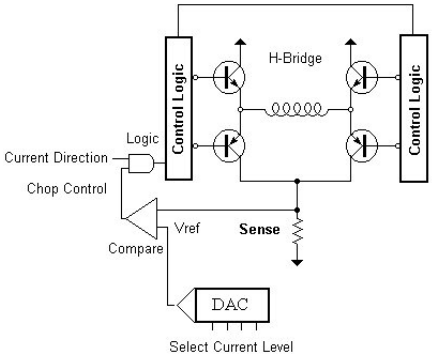


Figura A.13: Un circuito de control de corriente *chopper* y puente H.

El circuito integrado que se utilizó es el TA8435H como controlador del motor de pasos, el cual es un circuito de control por medio de PWM chopper. Este integrado puede manejar una corriente promedio máxima de 1.5 A y una corriente pico máxima de 2.5 A, utiliza *microstepping* como método de control del motor de pasos y puede dividir cada paso del motor en 2, 4 o hasta 8 pasos. El integrado tiene como entrada dos señales de reloj, aunque se puede hacer funcionar con sólo una, utilizado para controlar el tiempo entre pasos, y una señal que dictará la dirección del giro del motor.

A.5. OSCILADOR CONTROLADO POR VOLTAJE

Los osciladores controlados por voltaje, también conocidos como convertidores de frecuencia a voltaje o VCO, son muy comunes y usados con mucha frecuencia, por ejemplo, en el método de lazos de seguimiento de fase, PLL por las siglas en inglés de Phase-Locked Loop. Los VCO en circuito integrado suelen estar diseñados con multivibradores RC, en los cuales la corriente de carga en el condensador varía en respuesta a la entrada de control. A continuación se muestra un ejemplo de un VCO. Primero se considera al multivibrador de emisores acoplados que se muestra en la Figura A.14.

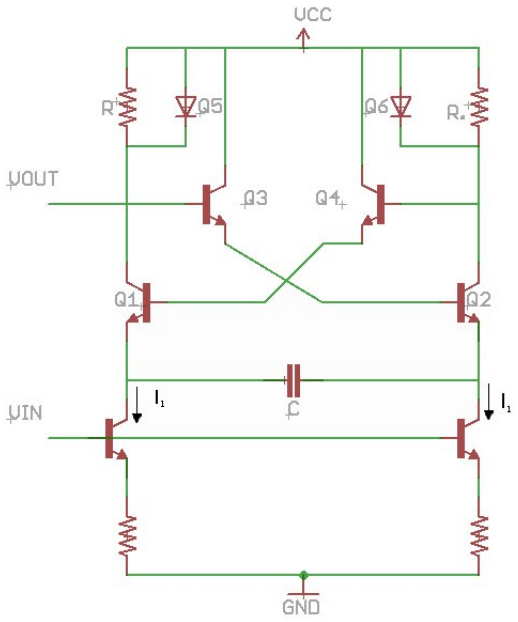


Figura A.14: Circuito básico de un VCO.

Para calcular el periodo se supone que al inicio, Q1 se encuentra apagado y Q2 prendido. Se asume que la corriente I es suficientemente grande como para encender el diodo Q6. De esta manera la base del transistor Q4 se encuentra a un voltaje Vcc menos el voltaje de encendido del diodo, V_D, el emisor a un voltaje Vcc - $2V_D$, al igual que la base del transistor Q1. La base del transistor Q3 se encuentra en un voltaje igual a Vcc y su emisor se encuentra a un voltaje Vcc - V_D, por lo que el emisor del transistor Q2 se encuentra a un voltaje Vcc - V_D. Dado que el transistor Q1 se encuentra apagado, la corriente I_1 está cargando al condensador de manera que el emisor del transistor se vuelve más negativo. El transistor Q1 se prenderá cuando el voltaje en su emisor sea igual a un voltaje Vcc - $3V_D$. Cuando esto suceda, la corriente de su colector activará al diodo Q5, de manera que la base del

transistor Q3 bajará su voltaje en VD al igual que la base del transistor Q2. Por lo anterior el transistor Q2 se apagará, haciendo que la base del transistor Q1 aumente su voltaje en VD al apagarse el diodo Q6. Ahora la corriente I_1 fluirá en el sentido contrario y cambiará la polaridad del voltaje almacenado el condensador, hasta que el voltaje del emisor del transistor Q2 sea $2V_D$ menor y entonces el circuito regresará al estado inicial. Dado que el circuito es simétrico, el medio periodo está dado por la siguiente expresión:

$$\frac{T}{2} = \frac{Q}{I_1}$$

en donde $Q = C\Delta V$, $Q = 2CV_{BE}(on)$ es la carga del condensador. De esta manera la frecuencia de oscilación es:

$$f = \frac{1}{T} = \frac{I_1}{4CV_{BE}}$$

Este circuito de emisor acoplado es no saturado y usa solamente transistores NPN. Se puede observar cómo la frecuencia de operación está en función del valor V_{BE}(on), de manera que ésta es la entrada de voltaje de control. Este circuito es solamente una muestra del funcionamiento de un oscilador controlado por voltaje; los diseños utilizados en los circuitos integrados típicos son más complejos, ya que tienen un acondicionamiento para la salida de la señal y una corrección para el deslizamiento de la frecuencia central debido a las variaciones de temperatura.[16]

El circuito integrado utilizado como VCO fue el KA331, el cual es un convertidor de voltaje a frecuencia, aunque también puede ser usado como convertidor de frecuencia a voltaje y en otras aplicaciones. La salida obtenida con este integrado es una señal cuadrada de valores lógicos, con una frecuencia variable. Se hizo uso de uno de los circuitos propuestos por el fabricante, que está mostrado en la Figura A.15.

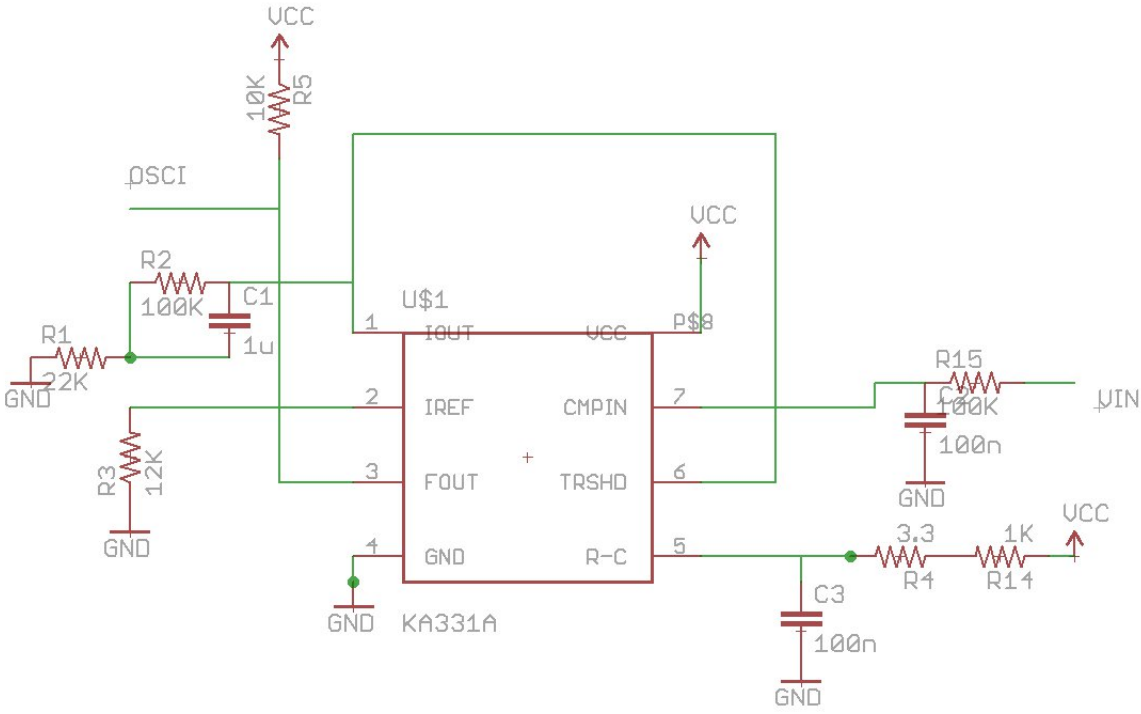


Figura A.15: Circuito para el VCO.

A.6. DISEÑO DEL FILTRO RC EN SERIE

Para conocer las características de un filtro, es necesario proponer una ecuación que permita relacionar el voltaje de salida con el voltaje de entrada. En este caso, en la Figura A.16 se puede considerar al circuito como un divisor de voltaje, en donde el voltaje de salida Vout es igual al voltaje en el condensador, por lo cual se puede llegar a la siguiente expresión en el dominio de Laplace:

$$V_{out}(\mathbf{s}) = \frac{V_{in}(\mathbf{s})}{(1 + RC\mathbf{s})}$$

De la expresión anterior se deduce que se tiene un sólo polo, el cual se encuentra en:

$$\mathbf{s} = \frac{1}{RC}$$

de donde se puede obtener que la frecuencia de corte que es:

$$f_c = \frac{1}{2\pi RC} Hz$$

Dado que el objetivo es tener una señal de CD con amplitud variable, la frecuencia de corte debe de ser baja, por lo cual se propone una de 70 Hz. Conocida esta frecuencia, se propone un condensador de 100 nF. Despejando de la ecuación anterior, se llega a:

$$R = \frac{1}{2\pi f_c C} \Omega$$

Sustituyendo valores:

$$R = \frac{1}{2\pi 70 \times 10^{-7}}$$

$$R = 22.736 \times 10^3 \Omega$$

Por lo anterior, el filtro RC utilizado para pasar de una señal de PWM de 430 Hz dada por el Arduino, a una señal de CD, se necesitará que R = 22 kΩ y C = 100nF.

A.7. MÓDULO DE CONTROL DEL MOTOR DE PASOS

La electrónica de potencia del robot es manejada por el Módulo de Control de Pasos (MCP), el cual está diseñado para controlar un sólo motor de pasos, teniendo dos entradas, una para el control de la velocidad y otra para controlar la dirección del giro. La intención de que utilice solamente dos entradas es reducir el número de salidas de un microprocesador, para el caso en el que un diseño utilice simultáneamente más de un MCP además de otras posibles salidas.

Dado que gran cantidad de los microprocesadores comúnmente utilizados para el control de motores incluyen salidas analógicas de PWM, el MCP tiene como entrada controladora de velocidad una señal de PWM, de modo que la variación del ciclo de trabajo de la señal es la que controla la velocidad del motor. De esta manera se logró tener control sobre la velocidad haciendo uso de una sola línea de comunicación y utilizando sólo una salida del microprocesador.

El control de la dirección de giro del motor está dado por una señal digital, ya que se tienen solamente dos direcciones de giro y dos estados en la lógica digital. De esta manera el control sobre la dirección de giro también se hace utilizando una sola línea de comunicación. Así, el MCP solamente requiere de dos entradas para su funcionamiento, una analógica de PWM y una digital. Gracias a que estas dos son comunes en microprocesadores para el control de motores, se cumplió con un diseño versátil, portable y que exige poco procesamiento por parte del microprocesador.

Se requería que diseño físico del MCP fuera lo más pequeño posible, además de que tuviera una buen método de disipación de calor. Esto último es importante si el módulo quiere utilizarse en aplicaciones que estén en constante funcionamiento. También se buscó que su uso fuera fácil para gente no familiarizada con su diseño, por lo que el MCP está bien señalizado y tiene indicadores, tanto de energización como de la actividad del mismo.

De esta manera, el circuito final constituido por todas las partes anteriormente mencionadas, además de un led indicador de encendido de la tarjeta se puede observar en la Figura A.16.

A.7.1. Diseño de la PCB del MCP

Retomando los objetivos anteriormente mencionados de todo el MCP, se requería que fuera un módulo portátil y versátil, que pudiera ser utilizado para otros proyectos, solucionando de esta manera el control de un motor de pasos. El tamaño era un factor importante, dado que el espacio que se tiene dentro del robot es reducido, y se tenía que hacer el mejor uso del mismo, además de que para cumplir el objetivo de ser portátil, es importante utilizar solamente el espacio necesario. Aparte de lo anterior, el MCP debía de poder

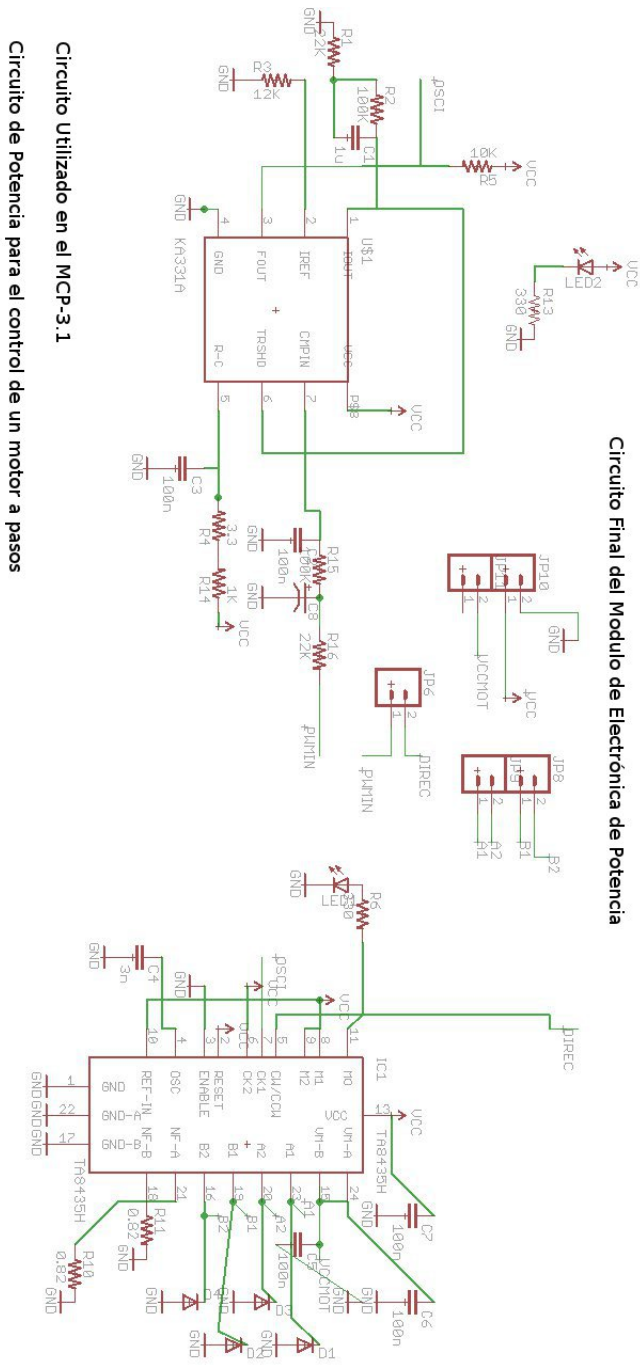


Figura A.16: Circuito Final del MCP.

funcionar constantemente por un largo periodo de tiempo sin tener problemas de sobrecalentamiento, por lo que fue necesario el diseño de un disipador.

Para cumplir con el objetivo de portabilidad y que el diseño pueda ser utilizado en otras aplicaciones, se diseñó la PCB con cuatro perforaciones en sus esquinas, de manera que pudiera ser fijada a diversas superficies en cualquier orientación y ésta se mantuviera fija. Se utilizaron terminales Molex

con tornillo con la intención de que las conexiones fueran cambiables, pero siempre estables, reduciendo así posible ruido electrónico. Los circuitos integrados y demás elementos se encuentran alejados de las orillas de la placa, con la intención de que no haya problemas al acomodar la placa en su lugar.

Para cumplir el objetivo del tamaño reducido, se diseñó la tarjeta de manera que los elementos se encontraran lo más cerca posible, para un mejor aprovechamiento del espacio. Se diseñó en una tarjeta de doble cara, dado que la problemática del espacio reside en las conexiones necesarias. Al tener doble capa de cobre se reducen tanto el problema de las conexiones como el espacio utilizado. Se utilizó un led indicador de montura superficial, para así ahorrar espacio y tener un menor gasto energético.

A.7.2. Versiones del MCP

El diseño final del MCP es el 3.1. A continuación se muestran las versiones y sus problemáticas. En las versiones subsiguientes se han resuelto las anomalías de las versiones anteriores.

1.0 Conexiones erróneas.

1.1 El tamaño de la placa excesivo.

1.2 Prototipo: espacio entre elementos erróneo, los elementos no caben. Uso erróneo de Footprints, algunos elementos no pueden ser soldados a la placa.

2.0 Uso erróneo de Footprints para led superficial.

2.1 Espacio insuficiente para el acomodo de las perforaciones de fijación.

2.2 Espacio insuficiente para el disipador.

2.3 Footprints incorrectos para resistencias de potencia.

2.4 Prototipo: circuito más complejo de lo necesario, existía ruido en la placa, dada la proximidad de las pistas. Pistas muy delgadas, que se calentaban cuando el motor requería de corrientes mayores.

3.0 Pistas aún no lo suficientemente gruesas, leyendas de tamaño incorrecto.

3.1 Prototipo: diseño final.

A.7.3. Diseño final del MCP

Las Figuras A.17 y A.18 muestran el diseño final de las dos caras del circuito impreso para el MCP.

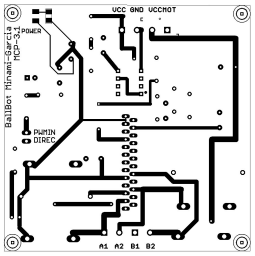


Figura A.17: Cara anterior del circuito impreso del MCP-3.1.

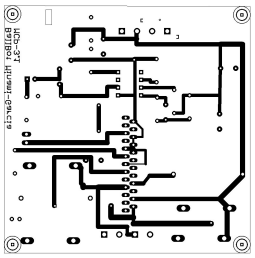


Figura A.18: Cara posterior del circuito impreso del MCP-3.1.

A.7.4. Disipador del MCP

El circuito de potencia hace uso de altos niveles de energía, es decir, altos niveles de voltajes y corrientes en relación con la electrónica del resto del robot, por lo que el circuito se calienta. El integrado TA8435H es el encargado de manejar el motor de pasos, y también maneja las altas corrientes y voltajes, por lo cual es el que más se calienta. El integrado tiene una placa metálica en su parte posterior, la cual ayuda a disipar el aumento de temperatura interna. El TA8435H está diseñado para tener una temperatura interna de operación máxima de 85°C [15], y el sobrepasar esta temperatura haría que el integrado detuviera su funcionamiento y, en el peor de los casos, podría dañar al mismo.

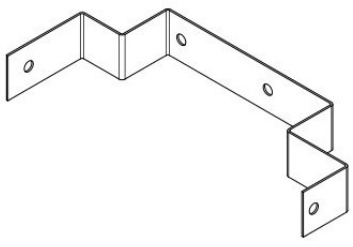


Figura A.19: Disipador sobre el MCP.

Con el fin de asegurar la integridad del TA8435H y de asegurar su continuo funcionamiento, se diseñó un disipador que ayudara a reducir el aumento de la temperatura. El uso de un disipador obliga a la temperatura a estabilizarse

en una temperatura menor al máximo de funcionamiento, de forma que el circuito pueda estar funcionando constantemente sin tener problemas por exceso de temperatura.

Como disipador se utilizó una placa metálica de Aluminio que está en contacto con la placa posterior del integrado para una máxima transferencia de calor. Para los parámetros de diseño se utilizó una placa con una altura igual a la del integrado, esto es de 1.4 cm, la longitud de la placa es tal que asegura la disipación de la temperatura, además de que pueda sostenerse con los tornillos ubicados en los extremos de la placa, de tal manera que no se ejerza un esfuerzo mecánico sobre la conexión del integrado. La longitud utilizada fue de 15.8 cm, y se hicieron seis dobleces para que la placa tuviera la forma necesaria y pudiera ser atornillada tanto al integrado como a los tornillos de la placa. Esto se puede observar más claramente en la Figura A.19.

Se usaron soportes hechos de plástico, con la intención de asegurar que no haya contactos eléctricos entre el disipador y el circuito del MCP. Esto permitió darles una mayor área para fijarlos sobre la placa sin preocuparse por el contacto con conexiones o pistas eléctricas, asegurando que no haya un esfuerzo mecánico ejercido sobre el integrado.

Las pruebas realizadas dieron resultados positivos, como se puede ver en la gráfica de la Figura A.20. La medición se llevó a cabo manteniendo constante una velocidad baja del motor en la cual consume más corriente y se tomaron los datos de temperatura a lo largo de media hora. Se puede observar que aunque la temperatura aumentó en comparación a la temperatura inicial, ésta se estabilizó en un valor por debajo del máximo de trabajo, por lo cual la disipación térmica es la suficiente para asegurar el buen funcionamiento del circuito.

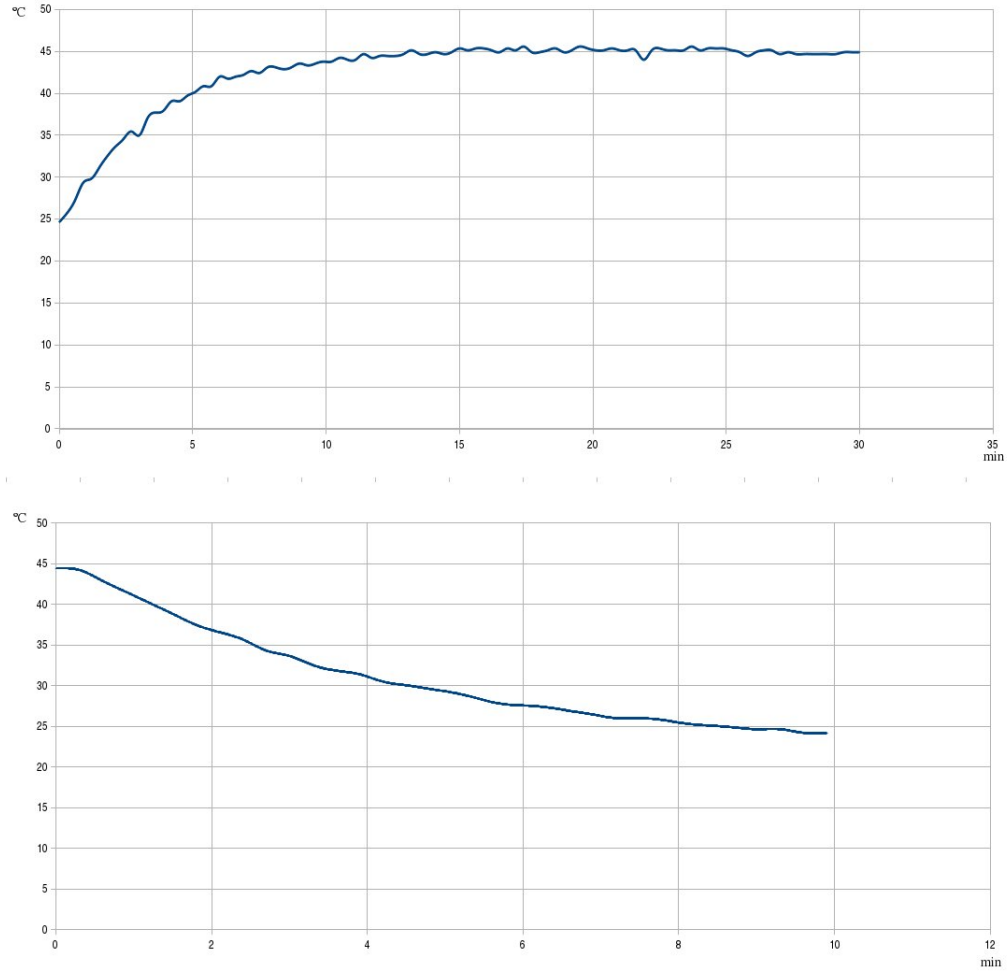


Figura A.20: Gráficas de calentamiento (arriba) y enfriamiento (abajo) del MCP.

A.8. MÓDULO INTERNO DE COMUNICACIÓN

Este módulo se diseñó con el propósito de poder conectar al Arduino con todos los demás módulos, como son la IMU y los controladores de los motores, de manera robusta y de fácil acceso.

El Módulo Interno de Comunicación (MIC) fue diseñado especialmente para conectarse con la tarjeta Arduino Duemilanove y con la Ultimate IMU, de manera que lograra establecer correctamente la conexión necesaria para la comunicación entre ambas tarjetas, así como alimentarlas y asegurar que entraran al modo funcionamiento y no al modo programación. De la misma manera propone un nuevo arreglo de los pines de las salidas PWM y digitales del Arduino con la intención de facilitar la conexión y arreglo de los cables en el robot para comunicar el Arduino con los controladores de los motores.

Dado que el objetivo es que este módulo sea capaz de proveer facilidades y mejoras en el proceso de diseño y construcción, para la etapa de prueba del algoritmo de control es muy práctico tener una forma visual de conocer el estado del algoritmo sin tener que estar conectándose constantemente a la computadora, por lo que se agregaron leds indicadores, que fueron asignados a variables o banderas, de tal manera que se pudiera visualizar en el circuito el estado del algoritmo de control. También el tener la posibilidad de programar la IMU desde el ambiente de programación del Arduino otorga una depuración del control más fácil, además de que puede ser útil si en el futuro se quieren escalar las capacidades del robot.

Se deseaba que el tamaño físico del MIC fuera el mínimo posible. El hecho de que se manejara casi exclusivamente señales digitales además de algunas de bajo voltaje (3.3 V), y que no existiera ningún requerimiento de altas corrientes, hicieron que fuera más sencillo reducir el espacio y que no fuera necesario hacer uso de disipadores. Siguiendo esta idea, se diseñó el circuito con elementos de montura superficial, logrando una manufactura más sencilla, al mismo tiempo que se redujo el tamaño; se hizo menor uso de potencia y no fue necesario el uso de disipadores. Se propuso que este módulo fuera portátil para otros proyectos, de manera que pudiera ser considerado dentro de los circuitos diseñados especialmente para el Arduino (*Arduino shields*), por lo cual era deseable que estuviera lo más claramente diseñado, que su uso fuera fácil e intuitivo y que estuviera correctamente documentado.

A.8.1. Diseño del circuito

La gran mayoría de líneas fueron empleadas para conectar los pines del Arduino con *headers* acomodados específicamente para que se le pudiera conectar la Ultimate IMU. Aparte de eso se colocaron varios leds indicadores para conocer visualmente el estado del algoritmo de control. La comunicación entre la Ultimate IMU y el Arduino se controla por medio de un *jumper*, de forma que al colocarlo se cierra el circuito de comunicación entre las dos

tarjetas y al quitarlo se corta la conexión y se entra al modo de programación.

Dado que se contaba con entradas analógicas no utilizadas del Arduino, éstas fueron conectadas a un *header* extra, con la intención de que esta entrada pudiera servir en un futuro como conexión con un módulo de comunicación inalámbrica, o con algún otro sensor como un sonar. Como ya se mencionó, se tenía la intención de que el módulo fuera portátil y expansible, por lo que se buscó hacer uso de la mayoría de las opciones que tiene el Arduino, debido a que una vez puesta la tarjeta MIC no es posible acceder a los *headers* del Arduino. De la misma manera el MIC tiene un botón pulsador con la función de reiniciar el Arduino sin quitar el módulo MIC.

El circuito final constituido por todas las partes anteriormente mencionadas, además de un led indicador de encendido de la tarjeta, quedó como se puede observar en la Figura A.21.

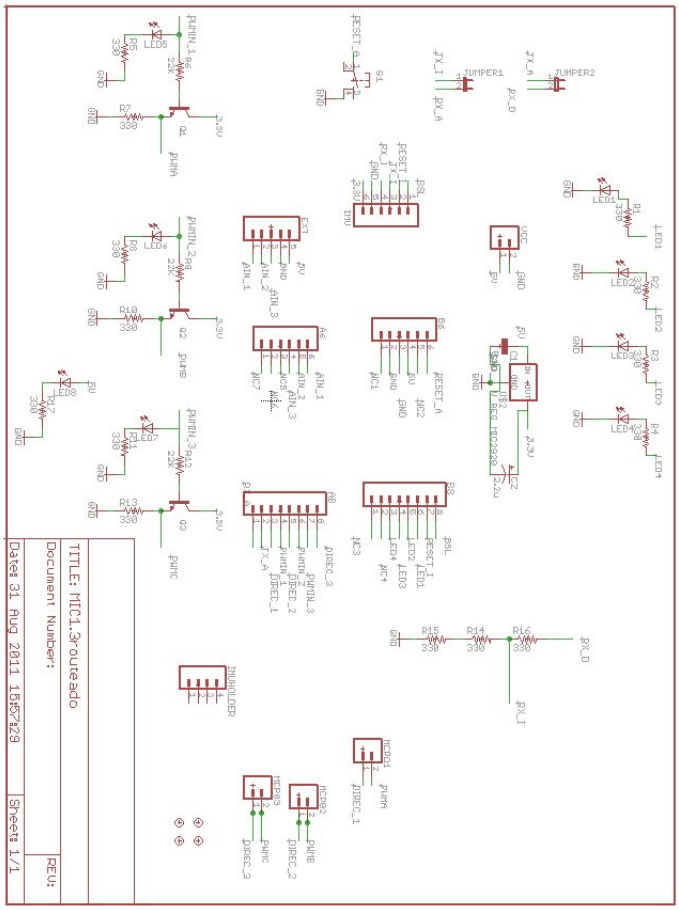


Figura A.21: Circuito de comunicación del Arduino con el resto de los Módulos.

A.8.2. Diseño de la PCB del MIC

Retomando los objetivos anteriormente mencionados del MIC, se requería que fuera un módulo portátil y versátil, que pudiera ser utilizado para otros proyectos, solucionando de esta manera la comunicación entre el Arduino y la Ultimate IMU, así como reorganizando el acomodo de las conexiones del Arduino. El tamaño era un factor importante, ya que el espacio que se tenía dentro del robot era reducido, y era necesario hacer el mejor uso del mismo, además de que para cumplir el objetivo de portabilidad, era importante que se utilizara solamente el espacio necesario.

Para cumplir con los objetivos de portabilidad, de que el diseño pudiera ser utilizado en otras aplicaciones y de que fuera totalmente compatible con el Arduino Duemilanove, el circuito impreso se diseñó de manera que el acomodo de los *headers* fuera idéntico al del Arduino para que embonen a la perfección. Se hizo lo mismo para la IMU, de forma que se obtuvo una conexión no sólo funcional sino también robusta, asegurando que la conexión no tenga problemas por el movimiento o las vibraciones que pudieran presentarse en el robot. La conexión de alimentación se realizó mediante un *molex* de terminal de tornillo para darle robustez.

Para cumplir el objetivo del tamaño reducido, se utilizó un circuito de doble cara, el cual permite mover las pistas con mayor facilidad. Al tener doble capa de cobre, tanto el problema de las conexiones como el espacio se redujeron considerablemente.

A.8.3. Versiones de la MIC

El diseño final del MIC fue la versión 1.3. A continuación se muestran las versiones y sus problemáticas. En las versiones subsiguientes se resolvieron las anomalías de las versiones anteriores.

1.0 Tamaño de las pistas muy pequeño.

1.1 Footprints erróneas para algunos de los integrados con encapsulado para soldadura superficial.

1.2 No se encontraba funcionando la opción de seleccionar modo de funcionamiento o modo de programación para la Ultimate IMU.

1.3 Prototipo: leds de visualización para las salidas de PWM hacia los MCP acomodados erróneamente, voltajes para modos de programación o funcionamiento de la Ultimate IMU incorrectos, errores con el circuito de cambio de voltaje para los PWM.

A.8.4. Diseño final del MIC

En las Figuras A.22 y A.23 se muestran las dos caras del circuito impreso del MIC.

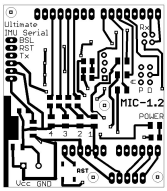


Figura A.22: Cara superior del MIC.

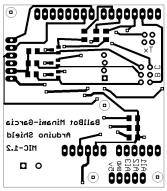


Figura A.23: Cara inferior del MIC.

A.9. DISEÑO DE LAS PCB

El diseño de los circuito impreso fue hecho en un programa llamado Eagle CadSoft. Este programa de desarrollo alemán cuenta con una versión gratuita para estudiantes, con la cual se realizaron los diferentes diseños. Las librerías utilizadas provienen de varias fuentes; algunas son las que el programa tiene predeterminadas, las de los elementos superficiales fueron desarrolla por la compañía Sparkfun, y algunas otras tantas fueron desarrolladas por nosotros.

En general se buscó que las diferentes PCB diseñadas pudieran ser utilizadas en otros proyectos por otras personas, por lo que era necesario que el empleo de las tarjetas fuera sencillo. Para esto se diseñaron con diversas leyendas, que permitieran el uso de las mismas a personas que desconocieran su diseño. En general, se nombraron todas las entradas y salidas necesarias de las tarjetas, así como algunos leds indicadores. También se incluyó el nombre de la tarjeta y su versión, de manera que no haya problemas de hubiera confusión entre las diversas versiones y de las tarjetas.

Específicamente, en la tarjeta del MCP se encuentran impresas las salidas hacia el motor de pasos indicando en cada una, las entradas tanto de tierra como de voltaje lógico y voltaje para los motores, las entradas de PWM y de dirección, así como el led indicador de energización de la tarjeta, y en la tarjeta del MIC, las salidas hacia los MCP, la tierra, el voltaje lógico y las entradas para la Ultimate IMU.

A.9.1. Metodología del nombramiento de las versiones

El método utilizado para nombrar las diferentes versiones de las tarjetas es el método utilizado por los desarrolladores de la compañía Sparkfun, el cual consiste en tener documentado cuál es la versión actual, la fecha de la misma y los cambios entre versiones. La identificación de las versiones es numérico, en donde se agrega una unidad por cada vez que se haya construido una versión, y se hayan hecho mejoras sobre el prototipo. Si las mejoras son realizadas antes de fabricar el prototipo, entonces se suma un décimo.

A.9.2. Manufactura de las tarjetas

Las tarjetas fueron hechas a mano en casi todos sus procesos. El diseño, como ya se mencionó, fue digital haciendo uso del programa Eagle CadSoft. Una vez realizado el diseño, se utilizó papel transfer para imprimir el circuito sobre la tarjeta de doble capa de cobre. Para la correcta alineación de ambas caras de la placa, se utilizaron marcas en la tarjeta en forma de perforaciones, obteniendo una alineación correcta superponiendo dichas marcas y haciéndolas coincidir. Para remover el cobre restante se utilizó cloruro férrico, haciendo el grabado de ambas capas de la placa simultáneamente. Las perforaciones se realizaron con una fresa para dentista y un taladro de

banco. La soldadura de todos los elementos, incluyendo la de los integrados de montaje superficial, fue realizada a mano.

A.10. CARACTERIZACIÓN DE LOS MOTORES

Se utilizó un encóder óptico incremental con una resolución de aproximadamente 6°, junto con un sensor infrarrojo, conectando la salida de este último al Arduino y obteniendo el tiempo entre pulsos en milisegundos para calcular la velocidad angular de los motores.

Se llevaron a cabo pruebas en los tres motores, en ambos sentidos, enviándoles una señal de PWM desde 10 hasta 150 en intervalos de 10. La librería de PWM del Arduino divide el 100 % de ciclo de trabajo en 255, de tal manera que un 0 es igual al 0 % de ciclo de trabajo y 255 es igual al 100 %.

Se tomaron 1000 muestras del tiempo entre pulsos y se sacó un promedio. Con el valor obtenido, y sabiendo que cada pulso equivale a 6°, se obtuvo el valor equivalente de la velocidad angular, en rpm, con la siguiente fórmula:

$$\omega = \frac{60 * 1000}{64t} rpm$$

donde t es el tiempo medido en milisegundos.

En las Figuras A.24 y A.25se pueden observar las gráficas obtenidas para cada motor en cada dirección. Con estas gráficas se pudo observar que prácticamente no existía diferencia en el comportamiento de lo motores girando de forma horaria o antihoraria. En donde sí se observó una diferencia fue en el comportamiento de cada uno de los motores, siendo muy similares los motores 1 y 2, y ligeramente distinto el motor 3.

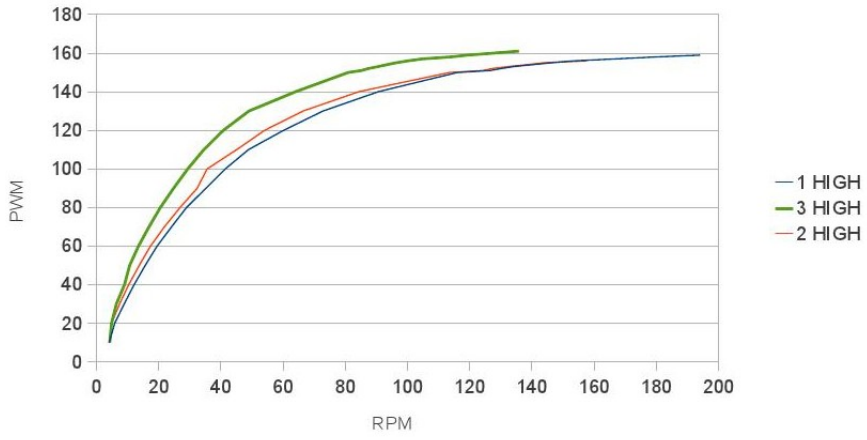


Figura A.24: Gráfica de la velocidad angular, en rpm, contra el valor de PWM en dirección antihoraria.

Finalmente, para obtener las ecuaciones para convertir la velocidad angular deseada en el valor de PWM necesario, sólo fue necesario buscar la curva de tendencia que más se acercara a la curva de cada motor. Para esto, se

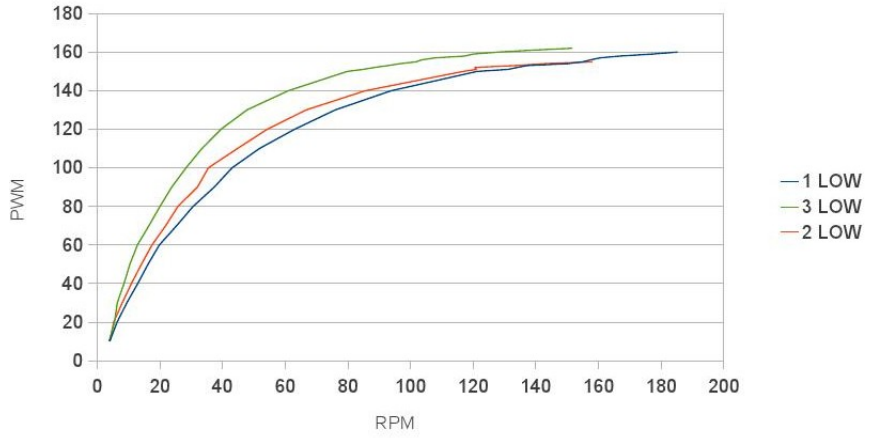


Figura A.25: Gráfica de la velocidad angular, en rpm, contra el valor de PWM en dirección horaria.

asumió que el comportamiento hacia ambos lados era igual, y sólo se obtuvo una ecuación para cada motor.

Estas ecuaciones quedaron de la siguiente forma:

$$PWM1 = 0.000000003\omega_5^1 - 0.000002\omega_4^1 + 0.0004\omega_3^1 - 0.0516\omega_2^1 + 4.0057\omega_1 - 3.8055$$

$$PWM2 = 0.000000006\omega_5^2 - 0.000003\omega_4^2 + 0.000\omega_3^2 - 0.0712\omega_2^2 + 4.6713\omega_2 - 3.1401$$

$$PWM3 = 0.00000001\omega_5^3 - 0.000007\omega_4^3 + 0.0013\omega_3^3 - 0.1251\omega_2^3 + 6.5774\omega_3 - 9.5947$$

donde $\text{PWM}_{1,2,3}$ son los valores de PWM a enviar a los controladores de cada motor, y $\omega_{1,2,3}$ son las velocidades angulares deseadas para cada motor, en rpm.

A.11. DISEÑO DEL FILTRO PARA VELOCIDAD Y POSICIÓN ANGULAR

Se decidió utilizar un filtro paso bajas dado que el ruido en los sensores se encontraba a frecuencias mayores a la frecuencia donde se encontraba la señal deseada. Se decidió hacer uso de un filtro de respuesta finita al impulso FIR (de sus siglas en inglés Finite Impulse Response) dado que su cualidad de estabilidad era algo deseado. Se decidió utilizar filtrado por ventana, dado que el hacer uso de ventanas permite tener un mayor control sobre la respuesta del filtro.

De los datos obtenidos de los sensores conocemos los siguientes parámetros para el filtro: Frecuencia de corte (f_c) 5 Hz, Frecuencia de muestreo (f_s) 122 Hz. El siguiente paso es elegir que tipo de ventana se va utilizar sobre un filtro paso bajas, se eligió utilizar la ventana tipo *Kaiser* , la cual depende de una parámetro . La forma de la ventana puede ser modificada modificando este parámetro.

Para obtener los pesos de una ventana tipo *Kaiser* se hace uso de la siguiente formula:

$$w(n) = \frac{I_0\left(\beta\sqrt{1-\left(\frac{2n}{M}-1\right)^2}\right)}{I_0(\beta)}$$

En donde $w(n)$ es el peso de la ventana para cada coeficiente del filtro, B es un parámetro para modificar la forma de la ventana, n es un coeficiente del filtro, M es el orden del filtro y I_0 es una función de Bessel modificada de primer tipo de orden cero. Donde I_0 se puede calcular de la siguiente forma :

$$\begin{aligned} I_0(x) &= \sum_{i=0}^{\infty} \frac{\left(\frac{x}{2}\right)^{2i}}{(i!)^2} \\ &= 1 + \frac{\left(\frac{x}{2}\right)^2}{(1!)^2} + \frac{\left(\frac{x}{2}\right)^4}{(2!)^2} + \frac{\left(\frac{x}{6}\right)^2}{(3!)^2} + \cdots \end{aligned}$$

Aunque es una función infinita el denominador crece rápidamente, por lo que se puede tener una buena aproximación en donde se utilice hasta i=20.

Se propuso utilizar un filtro de orden 30 y una B de 0.5. Con estos parámetros y los anteriormente mencionados, se obtuvieron los valores de los coeficientes del filtro paso bajas, los cuales fueron :

h(0)=-0.011807218907300191107490050512751622591
h(1)=-0.008636880670262829171046625731378298951
h(2)=-0.004287310817523732643652412122037276276
h(3)= 0.001176952246989978372385121652143880056
h(4)= 0.007633023295013000204811692839257375454

h(5)= 0.014903014402050428557910954907583800377
h(6)= 0.022760492620704706606682421465848165099
h(7)= 0.030939781153384297851527406919558416121
h(8)= 0.039147660049870261489424194678576895967
h(9)= 0.047076871899046149327894283942441688851
h(10)= 0.054420720577851917909573131737488438375
h(11)= 0.060887974666331820539078023557522101328
h(12)= 0.066217256931246179507688509602303383872
h(13)= 0.070190119741042958367849280421069124714
h(14)= 0.072642072760702128264576060701074311 51
h(15)= 0.073470940101706017077987098673474 97493
h(16)= 0.072642072760702128264576060701074311 51
h(17)= 0.070190119741042958367849280421069124714
h(18)= 0.066217256931246179507688509602303383872
h(19)= 0.060887974666331820539078023557522101328
h(20)= 0.054420720577851917909573131737488438375
h(21)= 0.047076871899046149327894283942441688851
h(22)= 0.039147660049870261489424194678576895967
h(23)= 0.030939781153384297851527406919558416121
h(24)= 0.022760492620704706606682421465848165099
h(25)= 0.014903014402050428557910954907583800377
h(26)= 0.007633023295013000204811692839257375454
h(27)= 0.001176952246989978372385121652143880056
h(28)=-0.004287310817523732643652412122037276276
h(29)=-0.008636880670262829171046625731378298951
h(30)=-0.011807218907300191107490050512751622591

La aplicación del filtro se puede observar en el código del microprocesador de la IMU.

A.12. PROGRAMA FINAL DE LA IMU

```
/*
Obtencin de posicin y velocidad angular y algoritmo de control
para un robot que se balancea sobre una esfera. Modificado del
algoritmo realizado por Ryan Owens para la Ultimade IMU de
SparkFun Electronics.

Minami, Garca
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "LPC214x.h"
#include "target.h"
#include <math.h>

//*********************************************************
//    Libreras de C
//*********************************************************

extern "C"{
#include "main_msc.h"
#include "serial.h"
#include "rprintf.h"
#include "timer0.h"
#include "timer0ISR.h"
#include "timer1.h"
#include "uart0.h"
#include "uart0ISR.h"
#include "uart1.h"
#include "uart1ISR.h"
}

//*********************************************************
//  Libreras de C++
//*********************************************************

#include "main.h"
#include "configuration.h"
#include "ADXL345.h"
#include "HMC5843.h"
#include "ITG3200.h"
#include "sensor.h"
#include "memory.h"
#include "EM408.h"

//*********************************************************
//     Funciones Principales
//*********************************************************
```

```
void bootUp(void);
void runTest(void);
void initPeripherals(void);

char Direction(float);
int Saturate(int);
double SaturateVel(double, double);

//*******************************************************
//        Variables Globales
//*******************************************************

//Version Information.
//KEEP THIS UPDATED!

char major_version=0;
char minor_version=1;

char sensors_updated=0;
char sensor_string[70]="";
char variable[70]="";

char sensor_log_string[255]="";

cMemory sensorData;
cMemory gpsData;
cMemory configData;


//RPM MAX
const int MAXw = 130;
const double Wmax = 130*PI/30;

//*******************************************************
//  Cdigo Principal
//*******************************************************

int main(void)
{
//Inicializa I/O del ARM
bootUp(); //Inic. Puertos de entrada/salida, protocolos de comunicacin e interrupciones

//Coeficientes para el Filtro
//122Hz
double filtro[31]=
{-0.011807218907300191107490050512751622591, -0.008636880670262829171046625731378298951,
-0.004287310817523732643652412122037276276, 0.001176952246989978372385121652143880056,
0.007633023295013000204811692839257375454, 0.014903014402050428557910954907583800377,
0.022760492620704706606682421465848165099, 0.030939781153384297851527406919558416121,
0.039147660049870261489424194678576895967, 0.047076871899046149327894283942441688851,
0.054420720577851917909573131737488438375, 0.060887974666331820539078023557522101328,
0.066217256931246179507688509602303383872, 0.070190119741042958367849280421069124714,
0.072642072760702128264576060701074311510, 0.073470940101706017077987098673474974930,
0.072642072760702128264576060701074311510, 0.070190119741042958367849280421069124714,
```

```
0.066217256931246179507688509602303383872, 0.060887974666331820539078023557522101328,
0.054420720577851917909573131737488438375, 0.047076871899046149327894283942441688851,
0.039147660049870261489424194678576895967, 0.030939781153384297851527406919558416121,
0.022760492620704706606682421465848165099, 0.014903014402050428557910954907583800377,
0.007633023295013000204811692839257375454, 0.001176952246989978372385121652143880056,
-0.004287310817523732643652412122037276276, -0.008636880670262829171046625731378298951,
-0.011807218907300191107490050512751622591};

double acumuladorx=0;
double acumuladory=0;

double tempx[30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double tempy[30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

double acumuladorr=0;
double acumuladorp=0;

double tempr[30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double tempp[30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

////Parmetros
//ngulo de las ruedas
const float phi = 30.7*PI/180;
const float phiR = 45*PI/180;

//Radio ruedas
const float r = 0.0762;

/////Control
int primeraVez = 1;

double ThEpx = 0;  //Velocidad angular alrededor del eje x
double ThEppx = 0;  //Aceleracin angular alrededor del eje x

double ThEpy = 0;  //Velocidad angular alrededor del eje y
double ThEppy = 0; //Aceleracin angular alrededor del eje x

double ThEpxAnt = 0; //Velocidad angular anterior alrededor del eje x
double ThEppxAnt = 0;  //Aceleracin angular anterior alrededor del eje x

double ThEpyAnt = 0;  //Velocidad angular anterior alrededor del eje x
double ThEppyAnt = 0;  //Aceleracin angular anterior alrededor del eje x

//Constantes del controlador
double K1 = -20.0*5;
double K2 = -11.69*3.75;
double K3=0;
double K4=-15;

float vs1, w1, vs2, w2, vs3, w3, vx, vy, Kz, wz; //vs en m/s, w en RPM
int PWM1, PWM2, PWM3;
char dir1, dir2, dir3;    //Direcciones de giro de los motores

LEDon();
```

```
//Inicializa perifricos
initPeripherals();

LEDoff();

double this_time = 0;
double last_time = 0;
double interval = 0;

while(true)
{
accelerometer.update();
gyro.update();
compass.update();

//Clculo de ngulos de Euler y orientacin compensada
filter.last_time=filter.this_time;
filter.this_time=T0TC; //Obtencin del tiempo actual en ms

//Calcula el intervalo de tiempo en ms
filter.interval=(filter.this_time-filter.last_time)*100/6102656;

//Llena el arreglo RwAcc
filter.fillRwAcc(accelerometer.getX(), accelerometer.getY(), accelerometer.getZ());

//Normaliza el vector gravedad del Acelermetro
filter.normalizeVector(filter.RwAcc);

//Las primeras mediciones del ngulo no funcionan debido a la falta de informacin anterior,
//para lo cual evitamos ls clculos la primera vez con este loop
if(filter.first_run)
{
double x_vel_rate_aux=0;
double y_vel_rate_aux=0;
double z_vel_rate_aux=0;


for(int w=0; w<3; w++)  filter.RwGyro[w] = filter.RwAcc[w];
filter.first_run=0;

for(int i=0; i<4; i++)
{
accelerometer.update();
accelerometer.getX();
x_vel_rate_aux += accelerometer.getX();
accelerometer.update();
accelerometer.getY();
y_vel_rate_aux += accelerometer.getY();
accelerometer.update();
accelerometer.getZ();
z_vel_rate_aux += accelerometer.getZ();
delay_ms(200);
}
```

```
filter.x_vel_rate_offset = (x_vel_rate_aux)/4;
filter.y_vel_rate_offset = (y_vel_rate_aux)/4;
filter.z_vel_rate_offset = (z_vel_rate_aux)/4;
}
else
{
//Si los valores estimados anteriores son muy pequeos, no calculamos nuevos
//ya que se pueden presentar errores muy grandese
if(filter.RwEst[2] < 0.1)
{
for(int w=0; w<3; w++) filter.RwGyro[w]=filter.RwEst[w];
}
//De otra forma, se calcula el ngulo mediante los giroscopios y combinamos dicha informacin
//con la obtenida mediante los acelermetros
else
{
//Obtencin de la velocidad angular del giroscopio en grad/s
filter.x_rate=gyro.getX();
filter.y_rate=gyro.getY();
filter.z_rate=gyro.getZ();
filter.z_rate=filter.z_rate+1.9;

//Hallamos la doferencia angular entre la lectura pasada y la actual
filter.x_angle=filter.x_rate*(filter.interval/1000.0); //grad/s * segundo ==grados
filter.y_angle=filter.y_rate*(filter.interval/1000.0);
filter.z_angle=filter.z_rate*(filter.interval/1000.0);

//Hallamos el ngulo actual basado en la medicin del ngulo anterior
filter.Axz = atan2(filter.RwEst[0], filter.RwEst[2])*180/PI;

//Obtenemos el ngulo anterior en grados
filter.Axz += filter.x_angle;

//Sumamos el ngulo anterior y el actual para obtener el ngulo total actual
filter.Ayz = atan2(filter.RwEst[1], filter.RwEst[2])*180/PI;
filter.Ayz += filter.y_angle;

if( abs2(filter.z_angle) > 0.05 ) filter.Axy += filter.z_angle;
if( filter.Axy > 360 ) filter.Axy = 0;
if( filter.Axy < -360 ) filter.Axy = 360;

//Hallamos la velocidad en los tres ejes: x, y y z
filter.x_vel_rate = (accelerometer.getX()-filter.x_vel_rate_offset);
filter.y_vel_rate = (accelerometer.getY()-filter.y_vel_rate_offset);
filter.z_vel_rate = (accelerometer.getZ()-filter.z_vel_rate_offset);

if( abs2(filter.x_vel_rate) > 0.0005 ) filter.x_vel += filter.x_vel_rate*(filter.interval/1000.0);
if( abs2(filter.y_vel_rate) > 0.0005 ) filter.y_vel += filter.y_vel_rate*(filter.interval/1000.0);
if( abs2(filter.z_vel_rate) > 0.0005 ) filter.z_vel += filter.z_vel_rate*(filter.interval/1000.0);
}

if(filter.RwAcc[2] >=0)  filter.signRzGyro=1;
else  filter.signRzGyro=-1;
```

```
//Usamos Axz para encontrar RxGyro = 1 / SQRT (1 + cot(Axz(n))^2 * sec(Ayz(n))^2)
//http://www.starlino.com/imu_guide.html
filter.RwGyro[0] = 1 / sqrt(1 + 1/(tan(filter.Axz*(PI/180))*tan(filter.Axz*(PI/180))*
cos(filter.Ayz*(PI/180))*cos(filter.Ayz*(PI/180))));

if ((filter.Axz > 180 & filter.Axz < 361)|(filter.Axz < 0 & filter.Axz > -181))
filter.RwGyro[0]*=-1;

//Usamos Ayz para encontrar RyGyro = 1 / SQRT (1 + cot(Ayz(n))^2 * sec(Axz(n))^2)
//http://www.starlino.com/imu_guide.html
filter.RwGyro[1] =
1 / sqrt(1 + 1/(tan(filter.Ayz*(PI/180))*tan(filter.Ayz*(PI/180))*cos(filter.Axz*(PI/180))*
cos(filter.Axz*(PI/180))));

if ((filter.Ayz > 180 && filter.Ayz < 361)|(filter.Ayz < 0 && filter.Ayz > -181))
filter.RwGyro[1]*=-1;

//RzGyro  =  Sign(RzGyro)*SQRT(1  RxGyro^2  RyGyro^2)
//http://www.starlino.com/imu_guide.html
filter.RwGyro[2] = filter.signRzGyro * sqrt(1-pow(filter.RwGyro[0],2)-pow(filter.RwGyro[1],2));
}

//Obtenidos el vector de gravedad tanto con el aceleretro como con el giroscopio,
//procedemos a combinar los datos para obtener
//RwEst =  (Racc + Rgyro * wGyro ) / (1 + wGyro), wGyro=gyro_weight=5 (en sensor.cpp)

for(int w=0; w<3; w++)
{
filter.RwEst[w] = (filter.RwAcc[w] + filter.RwGyro[w] * filter.gyro_weight)/(1+filter.gyro_weight);
}
filter.normalizeVector(filter.RwEst);

filter.EstAxz=atan2(filter.RwEst[0], filter.RwEst[2])*180/PI-3.4;
filter.EstAyz=atan2(filter.RwEst[1], filter.RwEst[2])*180/PI+5.2;

//Obtenidos los ngulos de Euler, calculamos la orientacin compensada
filter.RwMag[0] = compass.getX()+0.3;    //+0.3 para centrar las lecturas en cero
filter.RwMag[1] = compass.getY()-0.65;   //-0.65 para centrar las lecturas en cero
filter.RwMag[2] = compass.getZ();

//ngulo de declinacin magntica para la ciudad de Mxico = 5 19' E con cambios de 0 7' O/ao
//http://www.ngdc.noaa.gov/geomagmodels/struts/calcDeclination
double Declination = 5.3167*(PI/180);

//Encontramos los valores de orientacin en  X y Y utilizando la informacin de la inclinacin
//Ayz->roll->theta, Axz->pitch->phi
//Xh = bx * cos(phi) - by * sin(phi) * sin(theta) + bz * cos(theta) * sin(phi)
//Yh = by * cos(theta) + bz * sin(theta)
filter.x_h = filter.RwMag[0]*cos(filter.EstAxz*(PI/180))+
filter.RwMag[1]*sin(filter.EstAxz*(PI/180))*sin(filter.EstAyz*(PI/180))+
filter.RwMag[2]*cos(filter.EstAyz*(PI/180))*sin(filter.EstAxz*(PI/180));

filter.y_h=filter.RwMag[1]*cos(filter.EstAyz*(PI/180))-filter.RwMag[2]*sin(filter.EstAyz*(PI/180));
```

```
double Azimuth = (atan2(filter.y_h,filter.x_h)+Declination)*180/PI;

if (Azimuth<0) filter.heading = 360+Azimuth;
else filter.heading = Azimuth;

//Filtro
for(int b=29;b>=0;b--){
if(b==0){
tempx[b]=filter.x_rate;
tempy[b]=-filter.y_rate;
tempr[b]=filter.EstAyz;
tempp[b]=filter.EstAxz;
}
else{
tempx[b]=tempx[b-1];
tempy[b]=tempy[b-1];
tempr[b]=tempr[b-1];
tempp[b]=tempp[b-1];
}
acumuladorx=acumuladorx+(tempx[b]*filtro[b]);
acumuladory=acumuladory+(tempy[b]*filtro[b]);
acumuladorr=acumuladorr+(tempr[b]*filtro[b]);
acumuladorp=acumuladorp+(tempp[b]*filtro[b]);
}
filter.x_rate=acumuladorx-1.4; //Offset de ~1.4
acumuladorx=0;
filter.y_rate=acumuladory-1.6;  //Offset de ~1.6
acumuladory=0;
filter.EstAyz=acumuladorr;  //Offset de ~-5.2
acumuladorr=0;
filter.EstAxz=acumuladorp; //Offset de ~3.4
acumuladorp=0;

//Algoritmo de Control
if(primeraVez){
ThEppx = K1*filter.EstAyz*PI/180+K2*filter.x_rate*PI/180+K4*ThEpx;
ThEppy = K1*filter.EstAxz*PI/180+K2*filter.y_rate*PI/180+K4*ThEpy;
ThEppxAnt = ThEppx;
ThEppyAnt = ThEppy;
ThEpxAnt = ThEpx;
ThEpyAnt = ThEpy;
primeraVez = 0;
}
else {
ThEppx = K1*filter.EstAyz*PI/180+K2*filter.x_rate*PI/180 +K4*ThEpx;
ThEppy = K1*filter.EstAxz*PI/180+K2*filter.y_rate*PI/180 +K4*ThEpy;
ThEpx=ThEpxAnt+(ThEppx+ThEppxAnt)*(filter.interval/1000.0)/2;
ThEpxAnt=SaturateVel(ThEpx, Wmax);
ThEppxAnt = ThEppx;
ThEpy=ThEpyAnt+(ThEppy+ThEppyAnt)*(filter.interval/1000.0)/2;
ThEpyAnt=SaturateVel(ThEpy, Wmax);
ThEppyAnt = ThEppy;
}
```

```
//vx y vy en m/s
vy=ThEpx*r*2.5;   //Factor de velocidad en y de 2.5
vx=-ThEpy*r*3;      //Factor de velocidad en x de 3

//Para velocidad angular: v=w*r, w=v/r
Kz=-r*sin(phi);
wz=0;

//Conversin del movimiento en x y y al sistema de tres ruedas
vs3=-vy*cos(phi)+Kz*wz;
w3=fabs(vs3/r)*30/PI;               //RPM
vs2=(sqrt(3)*vx/2+vy/2)*cos(phi)+Kz*wz;
w2=fabs(vs2/r)*30/PI;               //RPM
vs1=(-sqrt(3)*vx/2+vy/2)*cos(phi)+Kz*wz;
w1=fabs(vs1/r)*30/PI;               //RPM

dir1=Direction(vs1);
dir2=Direction(vs2);
dir3=Direction(vs3);

 /*
Para la obtencin del valor de PWM segn la velocidad deseada
y=y'*PI/30 de RPM a rad/s, y=y'*30/PI de rad/s a RPM
y=PWM, x=RPM
y1 = 3E-09x5 - 2E-06x4 + 0.0004x3 - 0.0516x2 + 4.0057x - 3.8055
y2 = 6E-09x5 - 3E-06x4 + 0.0006x3 - 0.0712x2 + 4.6713x - 3.1401
y3 = 1E-08x5 - 7E-06x4 + 0.0013x3 - 0.1251x2 + 6.5774x - 9.5947
*/
PWM1=0.000000003*pow(w1,5)-0.000002*pow(w1,4)+0.0004*pow(w1,3)-0.0516*pow(w1,2)+4.0057*w1-3.8055;
PWM2=0.000000006*pow(w2,5)-0.000003*pow(w2,4)+0.0006*pow(w2,3)-0.0712*pow(w2,2)+4.6713*w2-3.1401;
PWM3=0.00000001*pow(w3,5)-0.000007*pow(w3,4)+0.0013*pow(w3,3)-0.1251*pow(w3,2)+6.5774*w3-9.5947;
PWM1=Saturate(PWM1);
PWM2=Saturate(PWM2);
PWM3=Saturate(PWM3);

//Envo de datos por serial
sprintf(sensor_string, "P%c%iX%c%iY%c%iF\n", dir1,PWM1,dir2, PWM2,dir3, PWM3);

rprintf(sensor_string);
}
return 0;
}

//Uso: bootUp();
//Entradas: Ninguna
//Esta funcin inicializa el puerto serial, la tarjeta SD, los pines de entrada/salida
//y las interrupciones
void bootUp(void)
{
//inicializa FCCLK--------------------
PLLCON = 1; //habilita el PLL
PLLCFG = (1 << 5) | 4; // P = 2, M = 5
PLLFEED = 0xAA;
PLLFEED = 0x55;
```

```
while ((PLLSTAT & (1 << 10)) == 0);
PLLCON = 3; //habilita y conecta
PLLFEED = 0xAA;
PLLFEED = 0x55;

MAMCR=0x02;
MAMTIM=0x03;

//Inicializa UART for RPRINTF
rprintf_devopen(putc_serial0); //Inic. rprintf
init_serial0(115200);

//Inicializa puertos de entrada/salida y perifricos
IODIR0 |= (LED| XBEE_EN);

//Configuracin de las interrupciones
//Habilita las interrupciones
VPBDIV=1; //Establece PCLK igual al reloj del sistema
VICIntSelect = ~(INT_TIMER0|INT_UART1|INT_UART0);
VICVectCntl0 = 0x20 | 4; //Timer 0 Interrupt
VICVectAddr0 = (unsigned int)ISR_Timer0;
VICVectCntl1 = (0x20 | 7);
VICVectAddr1 = (unsigned int)ISR_UART1; //UART 1 Interrupt
VICVectCntl2 = (0x20 | 6);
VICVectAddr2 = (unsigned int)ISR_UART0; //UART 0 Interrupt

T0TCR = 0x3;
T0IR = 0xff;
T0TC = 0;
T0TCR = 0x01;
}

//Uso: reset();
//Entradas: Ninguna
//Descripcin: Reinicia al LPC2148
void reset(void)
{
//Activamos intencionalmente al Watchdog para crear ua condicin de reinicio
WDMOD |= 3;
WDFEED = 0xAA;
WDFEED = 0x55;
WDFEED = 0xAA;
WDFEED = 0x00;
}

void initPeripherals(void)
{
//Configuracin de sensores
accelerometer.begin(configuration.range_accel);
gyro.begin();
compass.begin(configuration.range_compass);

//Establecemos los offsets de los sensores
```

```
accelerometer.setCalibrationValues(configuration.cal_ax, configuration.cal_ay, configuration.cal_az);
gyro.setCalibrationValues(configuration.cal_gx, configuration.cal_gy, configuration.cal_gz);

delay_ms(100);
}

void runTest(void)
{
char value;
int gpstest;

//Fijamos los pines de UART0 pins como entradas/salidas para pruebas iniciales del XBee
PINSEL0 &= ~((3<<0) | (3<<2)); //Set P0.0 and P0.1 to GPIO
IODIR0 |= (1<<0)|(1<<1); //Set P0.0 and P0.1 to outputs

for(int blink=0; blink < 10; blink++)
{
IOSET0 = (1<<0)|(1<<1); //Turn on P0.0 and P0.1
delay_ms(50);
IOCLR0 = (1<<0)|(1<<1);
delay_ms(50);
}

//Habilita el puerto serial
//Inicializa UART para RPRINTF
rprintf_devopen(putc_serial0); //Init rprintf
init_serial0(9600);

//Prueba del GPS
IODIR0 &= ~((1<<8)|(1<<9)|(1<<12)); //Fijamos P0.8, P0.9 y P0.12 como entradas
gpstest = IOPIN0;
gpstest = (gpstest>>8)&0x13;
if(gpstest != 0x11)
{
rprintf("GPS Failed.");
while(1);
}

//Prueba del Acelermetro
accelerometer.begin(1);
//Obtenemos el  ID del acelermetro
value = DEVID;
accelerometer.read(&value, 1);

if(value != 0xE5)
{
rprintf("Accel Failed to ping");
while(1);
}

//Prueba del  Giroscopio
gyro.begin();
//Obtenemos el ID del giro
value = WHO_AM_I;
```

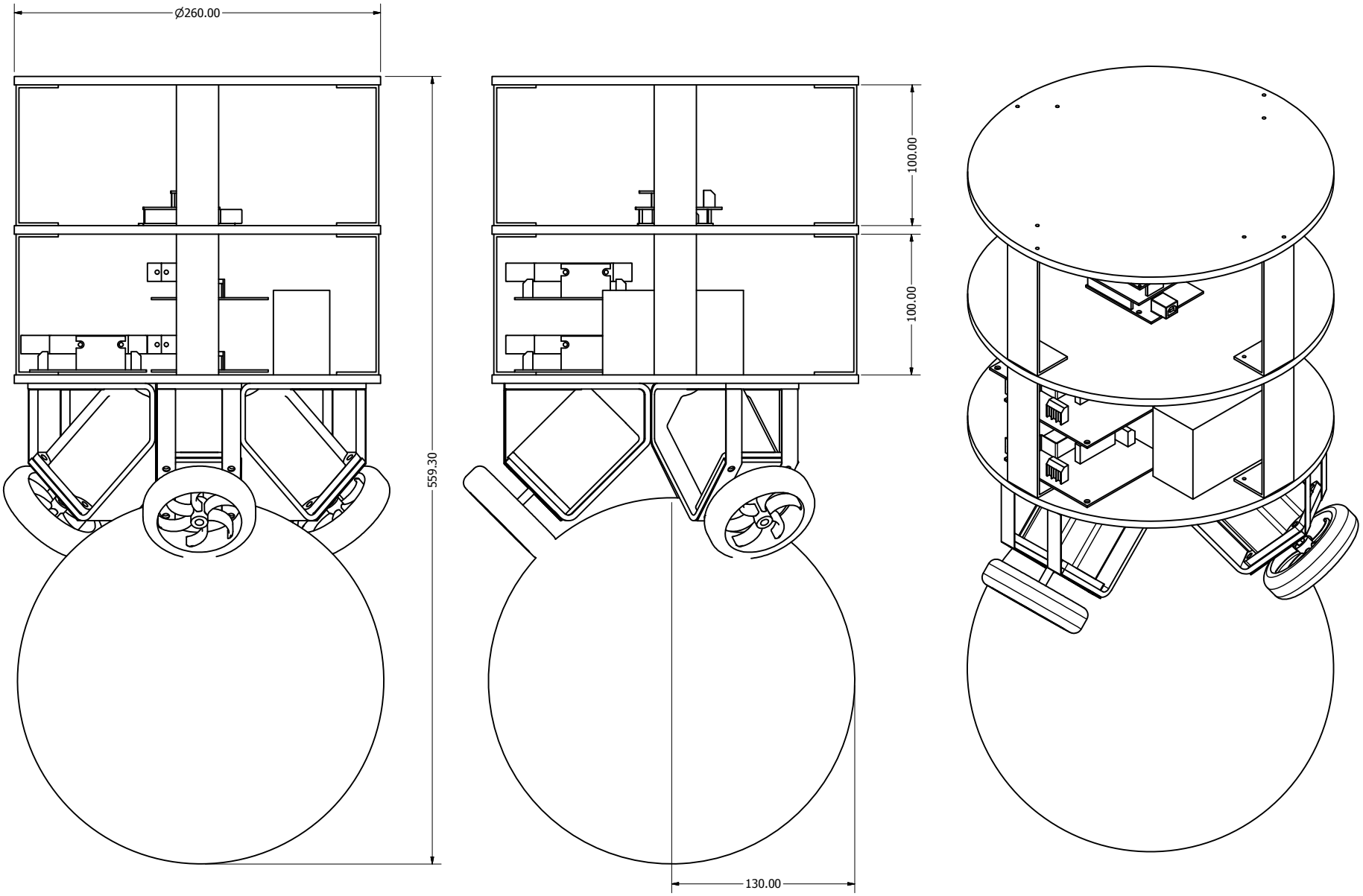
```
gyro.read(&value, 1);

if((value & 0x68) != 0x68)
{
rprintf("Gyro failed to ping");
while(1);
}

//Prueba de la brjula
compass.begin(1);
//Obtenemos el ID de la brjula
value = ID_REGA;
compass.read(&value, 1);

if(value != 0x48)
{
rprintf("Compass failed to ping");
while(1);
}

rprintf("Pass!");
LEDon();
memoryDelete("Test.txt");
}

double abs2(double a){

if(a < 0)
{
a = a*-1;
}
return a;
}

//Funcin para decidir el sentido de giro de los motores
char Direction(float vel){
if(vel>=0)  return 'p'; //Rotacin antihoraria
else   return 'n'; //Rotacin horaria
}

//Funcin para saturar el valor del PWM enviado a los motores
int Saturate(int pwm){
if(pwm>=MAXw) pwm=MAXw;
else if(pwm<0) pwm=0;
return pwm;
}

//Funcin para saturar la velocidad deseada hasta la velocidad mxima alcanzable
//de los motores
double SaturateVel(double W, double Wm){
if(W>=Wm) W=Wm;
else if (W<=-Wm) W=-Wm;
return W;
}
```

A.13. PROGRAMA FINAL DEL ARDUINO

```
#include <math.h>
#include <SoftwareSerial.h>

//Pines PWM = 3, 5, 6, 9, 10, 11
const int MCP1 =  3;
const int MCP2 =  5;
const int MCP3 =  6;

//Pines de direccin
const int DIR1 =  7;
const int DIR2 =  4;
const int DIR3 =  2;

//Obtencin de datos de IMU

//Cadenas para guardar los datos recibidos
char lineaP[32];
char lineaX[32];
char lineaY[32];

//ndices de las cadenas de los datos recibidos
int indexP, indexX, indexY;

//Banderas para la recepcin de datos
boolean startedP = false;
boolean endedP = false;
boolean startedX = false;
boolean endedX = false;
boolean startedY = false;
boolean endedY = false;

//Enteros convertidos de los datos recibidos
int inFloatP, inFloatX, inFloatY;

int pP=1, pX=1, pY=1;

//Funcin para enviar un HIGH o un LOW
//al pin de direccin de giro de los motores
//dependiendo de la direccin de giro
//deseada
void Direction(char vel, int dir){
  if(vel=='p') digitalWrite(dir,HIGH);
  else digitalWrite(dir,LOW);
}

void setup() {
  pinMode(MCP1, OUTPUT);
  pinMode(MCP2, OUTPUT);
  pinMode(MCP3, OUTPUT);
  pinMode(DIR1, OUTPUT);
  pinMode(DIR2, OUTPUT);
  pinMode(DIR3, OUTPUT);
  Serial.begin(115200);
}

//Programa principal
void loop()
{
//Lectura de datos de la IMU
//Se recibe una cadena con la forma:
//Ps##Xs##Ys##F
//donde P, X, Y y F son delimitadores
//s son los signos de los valores
//y ## son los valores numricos de los PWM
//a enviar a cada motor
  while(Serial.available() > 0)
  {
    char aChar = Serial.read();
    if(aChar == 'P')
    {
      pY=1;
      startedP = true;
      indexP = 0;
      lineaP[indexP] = '\0';
    }
    else if(aChar == 'X')
    {
      pP=1;
      endedP = true;
      startedX = true;
      indexX = 0;
      lineaX[indexX] = '\0';
    }
    else if(aChar == 'Y')
    {
      pX=1;
      endedX = true;
      startedY = true;
      indexY = 0;
      lineaY[indexY] = '\0';
    }
    else if(aChar == 'F')
    {
      endedY = true;
    }
    else if(startedP && !endedP)
    {
      if(pP)   {Direction(aChar,DIR1);
      pP=0;}
      else{
```

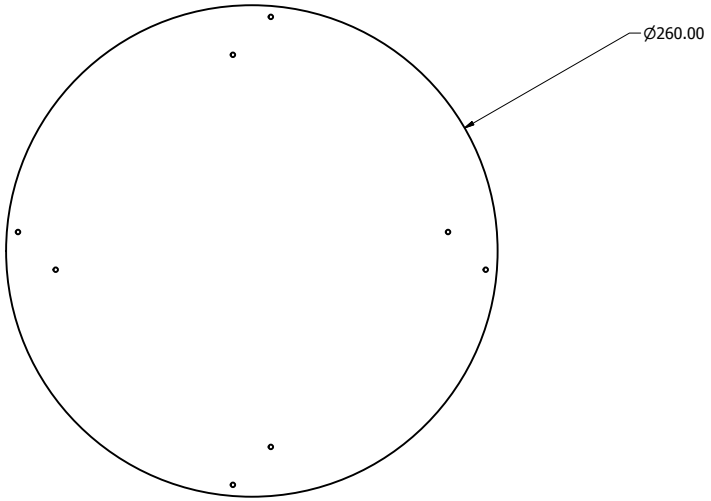
```
    lineaP[indexP] = aChar;
    indexP++;
    lineaP[indexP] = '\0';
    }
  }
  else if(startedX && !endedX)
  {
    if(pX)   {Direction(aChar,DIR2);
    pX=0;}
    else{
    lineaX[indexX] = aChar;
    indexX++;
    lineaX[indexX] = '\0';
    }
  }
  else if(startedY && !endedY)
  {
    if(pY)   {Direction(aChar,DIR3);
    pY=0;}
    else{
    lineaY[indexY] = aChar;
    indexY++;
    lineaY[indexY] = '\0';
    }
  }
}

if(startedP && endedP && startedX &&
    endedX && startedY && endedY)
{
  // Convierte los string a enteros
  inFloatP = atoi(lineaP);
  inFloatX = atoi(lineaX);
  inFloatY = atoi(lineaY);

  // Imprime los valores obtenidos
   Serial.print(inFloatP);
   Serial.print(',');
   Serial.print(inFloatX);
   Serial.print(',');
   Serial.println(inFloatY);
   //Serial.println(T*1000);

  // Reinicia las banderas para la siguiente lectura
  startedP = false;
  endedP = false;
  startedX = false;
  endedX = false;
  startedY = false;
  endedY = false;

  indexP = 0;
  lineaP[indexP] = '\0';
  indexX = 0;
  lineaX[indexX] = '\0';
  indexY = 0;
  lineaY[indexY] = '\0';
 }
 analogWrite(MCP1,inFloatP);
 analogWrite(MCP2,inFloatX);
 analogWrite(MCP3,inFloatY);
}
```

A.14. PLANOS DEL CUERPO DEL ROBOT

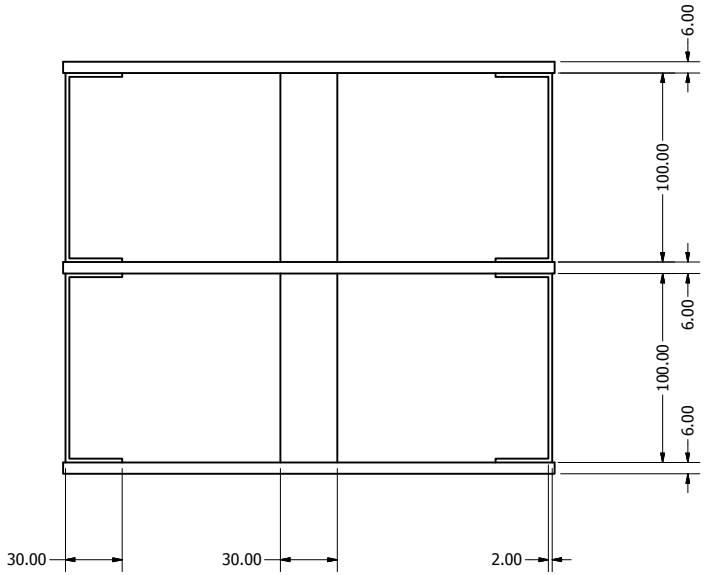


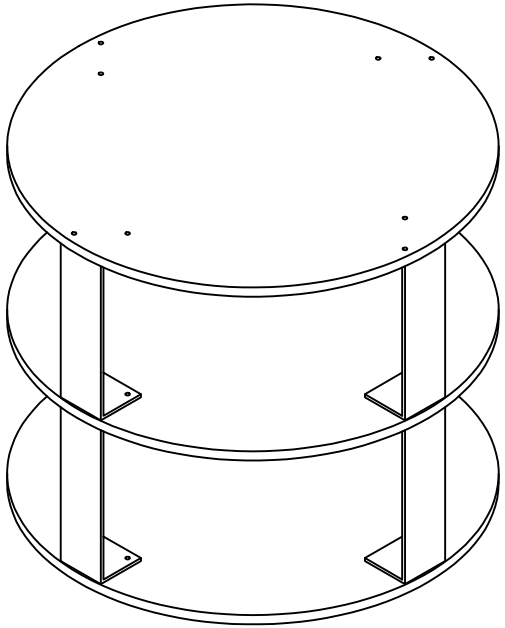
SCALE 1 / 2

PROYECTO:	
COTAS : MM	PLANO Planos
	HOJA 1 DE 8



SCALE 1 / 2



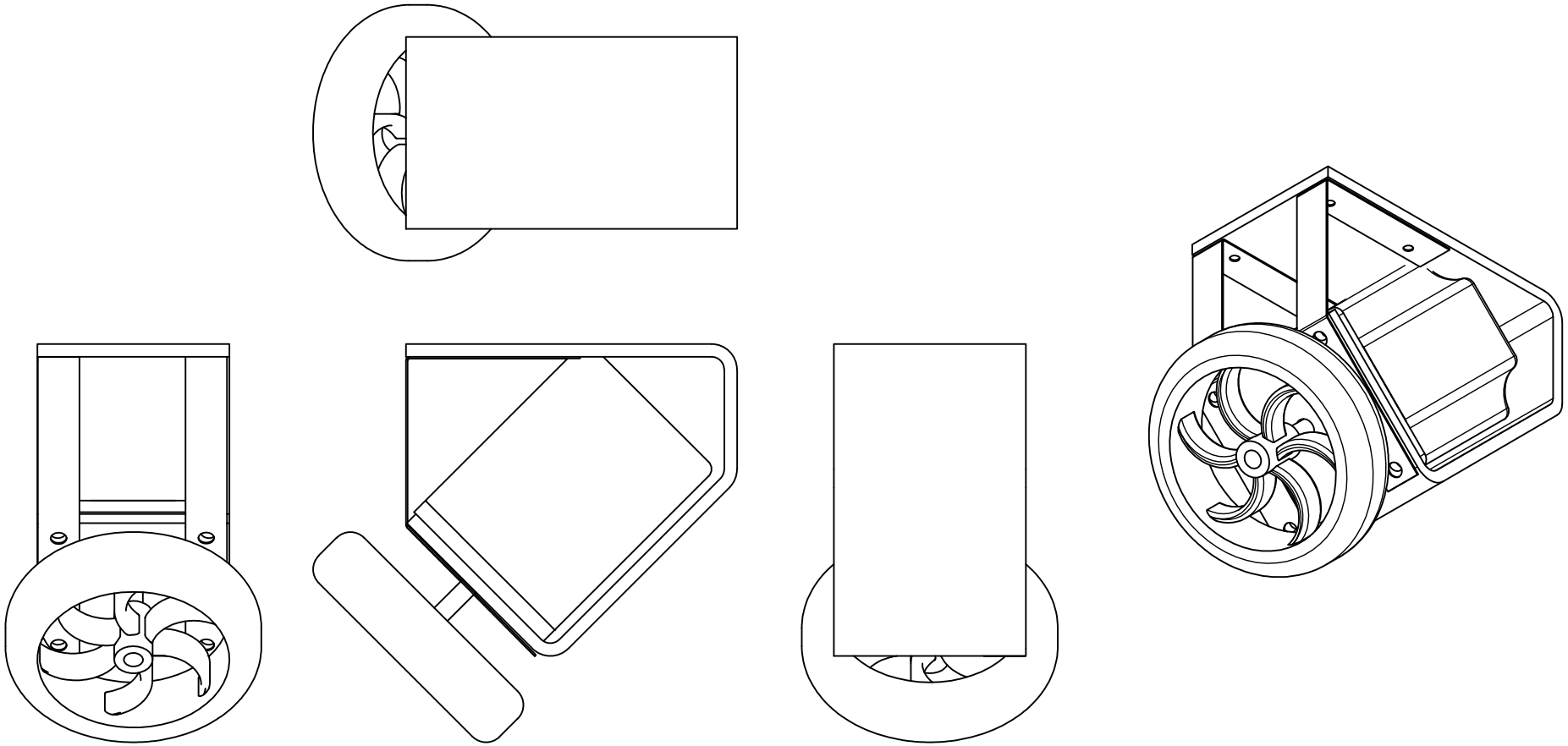


PROYECTO:

COTAS : MM

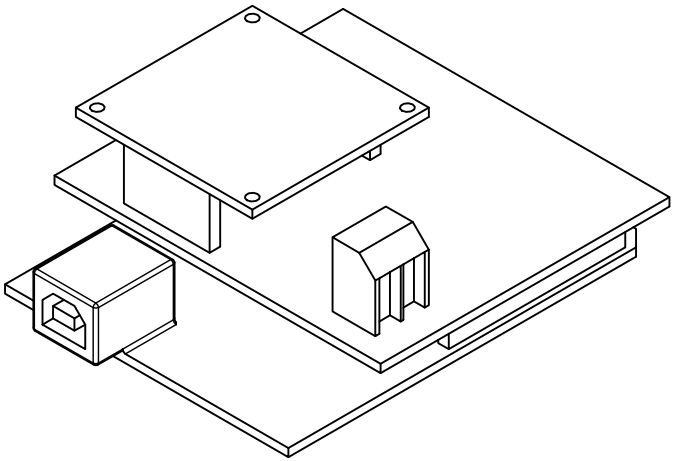
PLANO
Planos

HOJA 2 DE 8

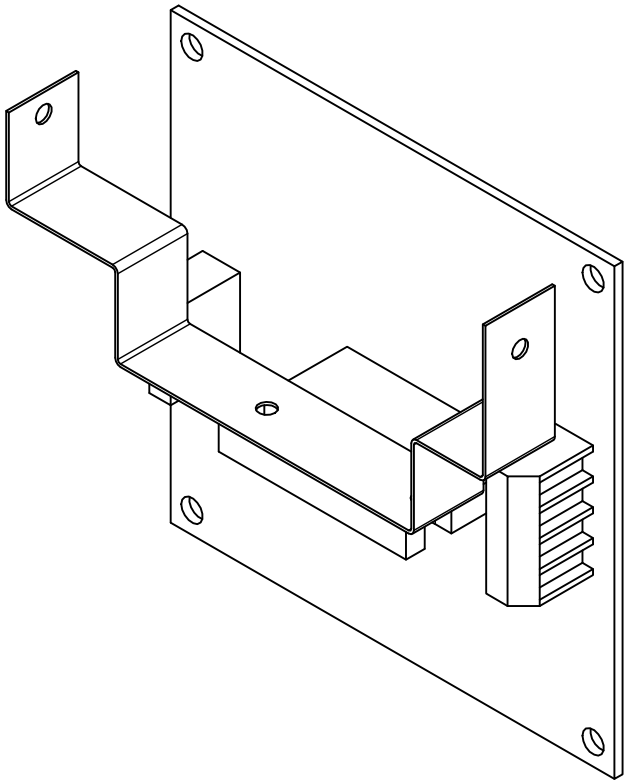


SCALE 1 : 1

PROYECTO:	
COTAS : MM	PLANO Planos
	HOJA 3 DE 8

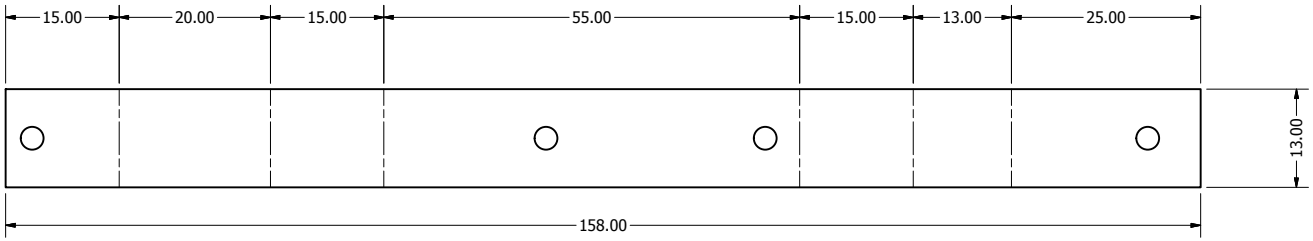


SCALE 2 : 1

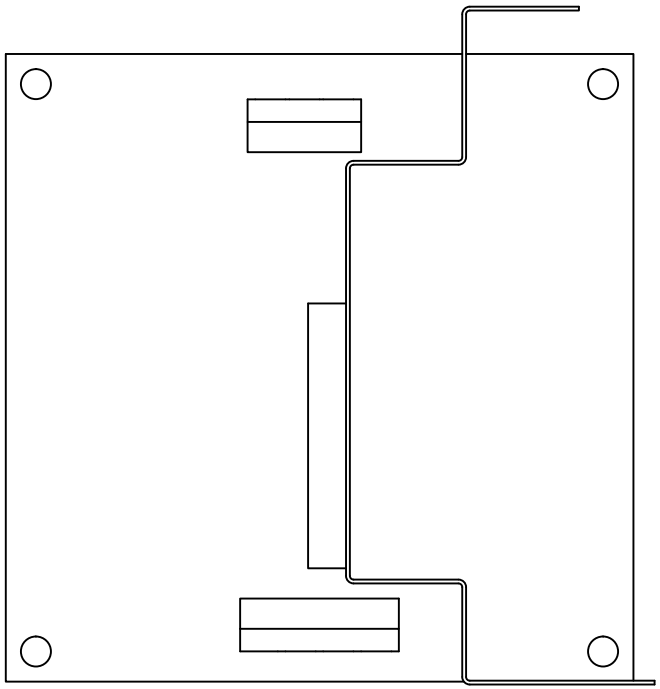


SCALE 2 : 1

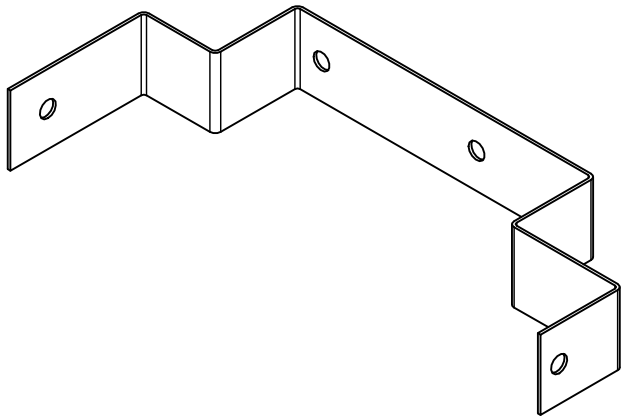
PROYECTO:	
COTAS : MM	PLANO: Planos
	HOJA 4 DE 8



SCALE 2 : 1



SCALE 2 : 1

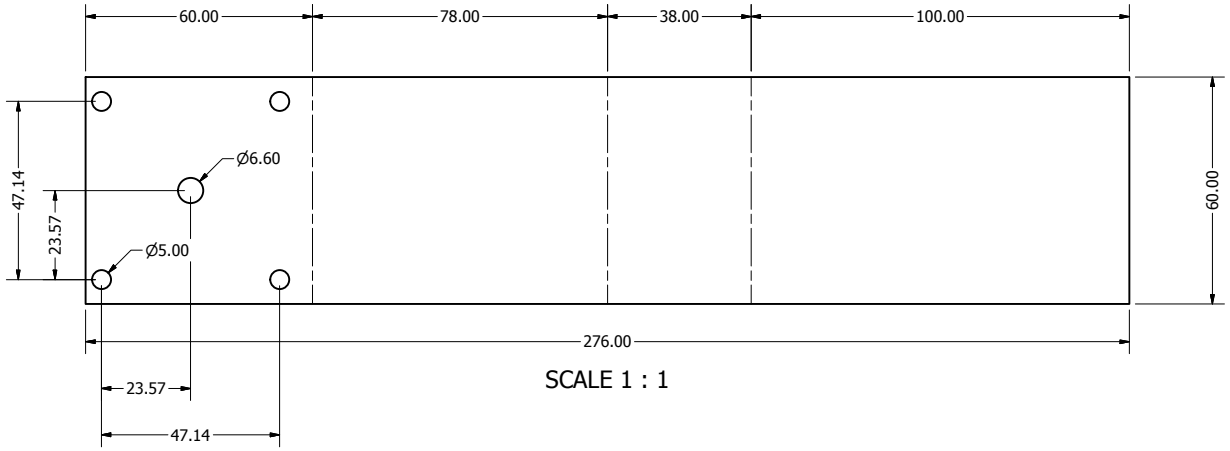


PROYECTO:

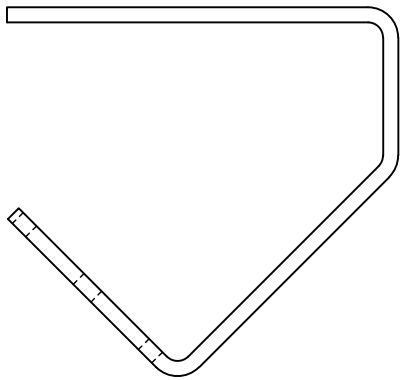
COTAS : MM

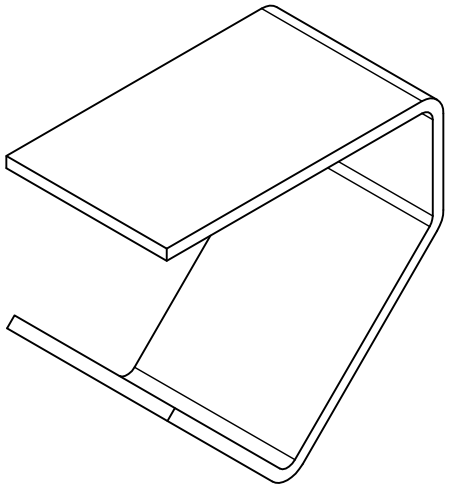
PLANO
Planos

HOJA 5 DE 8

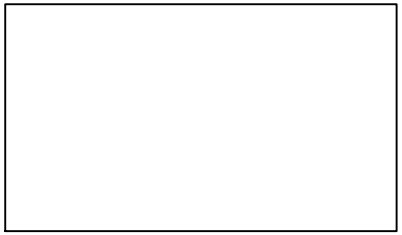


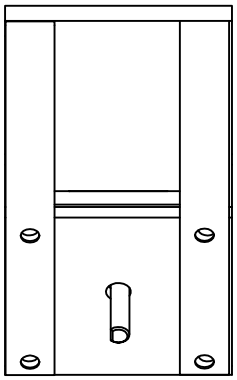
SCALE 1 : 1

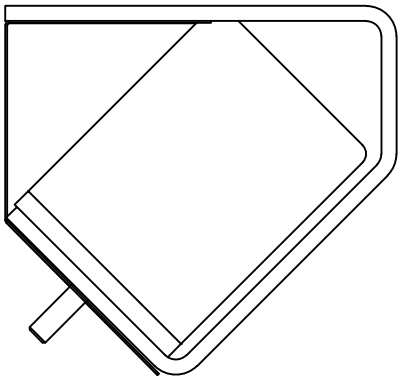


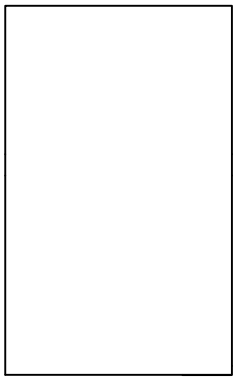


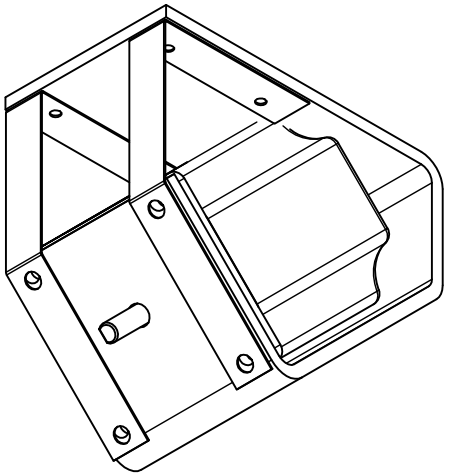
PROYECTO:	
COTAS : MM	PLANO Planos
	HOJA 6 DE 8

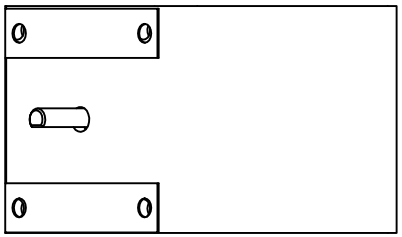










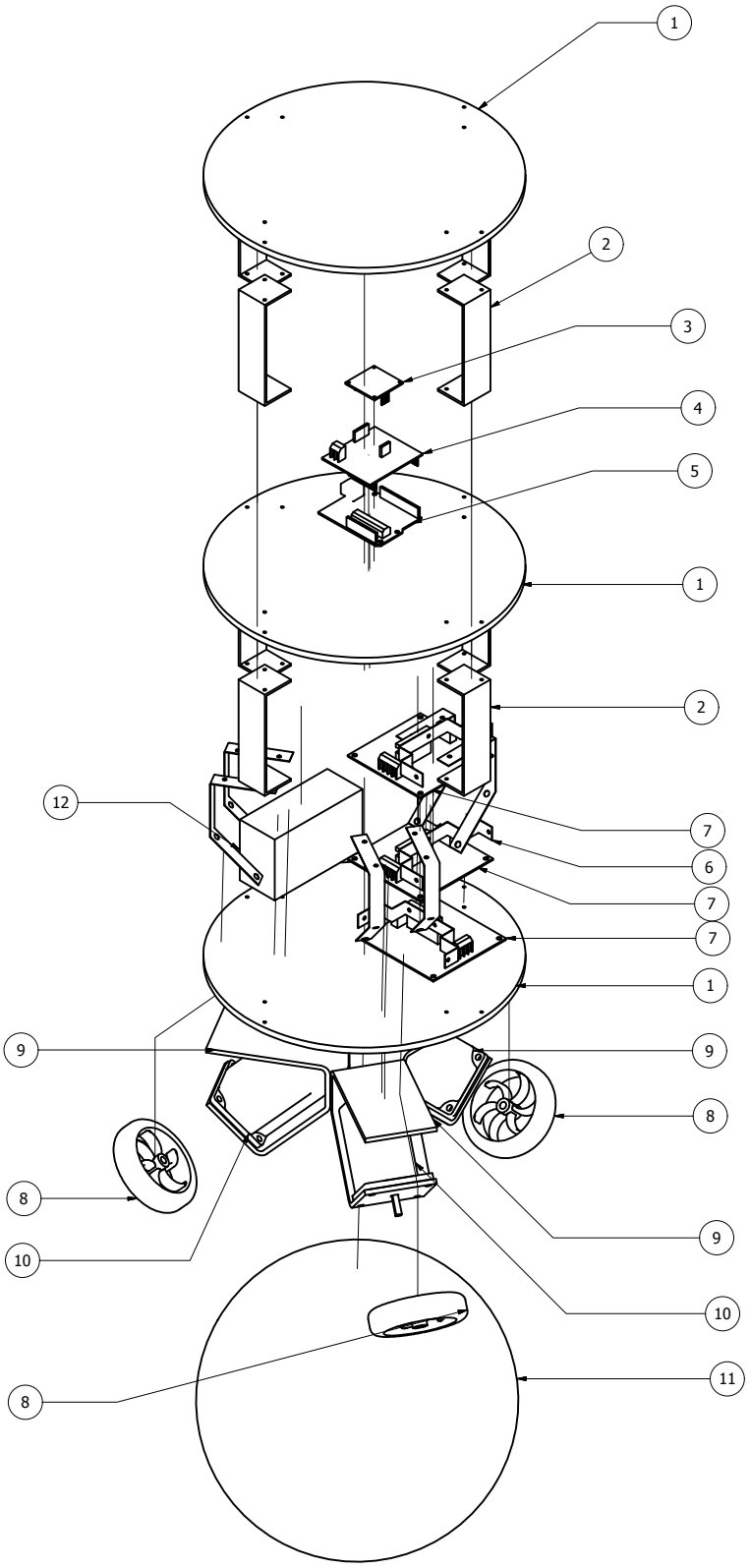


PROYECTO:

COTAS : MM

PLANO
Planos

HOJA 8 DE 8



Parts List			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	3	Tapa Cuerpo	
2	8	Soporte Cuerpo	
3	1	IMU	
4	1	MIC	
5	1	ARDUINO	
6	3	Disipador	
7	3	MCP	
8	3	Ruedas	
9	3	Soporte Motor	
10	3	Motor	
11	1	Esfera	
12	1	Bateria	

PROYECTO:

COTAS : MM

PLANO

Planos

HOJA 7 DE 8

Referencias

[1] Lauwers, T. B., Kantor, G. A., Hollis, R. L., *A dynamical stable single-wheeled mobile robot with inverse mouse-ball drive*, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 28842889, Orlando, Florida, EUA, May 2006.
[2] Kumagai, M., Ochiai, T., *Development of a robot balancing on a ball*, International Conference on Control, Automation and Systems, Seul, Corea del Sur, 2008.
[3] Endo, T., Nakamura, Y., *An omnidirectional vehicle on a basketball*, IEEE, 2005.
[4] Liao, C. W., Tsai, C. C., Li, Y. Y., Chan, C. K., *Dynamic Modeling and Sliding-Mode Control of a Ball Robot with Inverse Mouse-Ball Drive*, Proceedings of SICE 2008, Tokyo, Japan, pp. 29512955, Aug 2008.
[5] Nagarajan, U., Mampetta, A., Kantor, G., Hollis, R., *State Transition, Balancing, Station Keeping, and Yaw Control for a Dynamically Stable Single Spherical Wheel Mobile Robot*, Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, May 2009.
[6] Fong, J., Uppil, S., *Design and Build a Ballbot, Final Report*, The University of Adelaide, Faculty of Engineering, Computer and Mathematical Sciences, School of Mechanical Engineering, Adelaide, Australia, Oct 2009. (Tesis)
[7] Frankhauser, P., Gwerder, C.s, *Modelling and Control of a Ballbot*, Swiss Federal Institute of Technology Zurich, Zurich, Suiza, 2010. (Tesis)
[8] Wu, C. W., Hwang, C. K., *A Novel Spherical Wheel Driven By Omniwheels*, In Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, pp. 38003803, Kunming, China, Jul 2008.
[9] Escobar Salguero, L. H., *Conceptos Básicos de Procesamiento Digital de Señales*, Facultad de Ingeniería, UNAM, pp. 19, México, Feb 2009.
[10] Sedra, A., Smith, K. C., *Dispositivos Electrónicos y Amplificación de Señales*, Interamericana, México, 1987
[11] Senturia, S. D., *Microsystem Design*, Kluwer Academic Publishers, New York, EUA, 2002
[12] Hoja de datos para el ADXL345 .
[13] Hoja de datos para el ITG-3200
[14] Hoja de datos para el HMC5843
[15] Hoja de datos del TA8435H
[16] Gray, P. R., *Analysis and Design of Analog Integrated Circuits*, 2a ed., John Wiley and Sons, EUA, 1984
[17] Baluta, S., *Starlino Electronics*, http://www.starlino.com/