



INSTITUTO POLITÉCNICO NACIONAL

Centro de Investigación en Computación

Laboratorio de Robótica y Mecatrónica

**Desarrollo de un robot de dos ruedas
dinámicamente estable**

TESIS

Para obtener el grado de:

Maestría en Ciencias de la Computación

Presenta:

Ing. José Ángel Martínez Navarro

Directores de tesis:

Dr. Juan Humberto Sossa Azuela

Dr. Elsa Rubio Espino



Ciudad de México

Enero 2018



INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 17:00 horas del día 30 del mes de noviembre de 2017 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Desarrollo de un robot de dos ruedas dinámicamente estable"

Presentada por el alumno:

MARTÍNEZ

Apellido paterno

NAVARRO

Apellido materno

JOSÉ ÁNGEL

Nombre(s)

Con registro:

B	1	5	1	1	7	4
---	---	---	---	---	---	---

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de Tesis

Dr. Juan Humberto Sossa Azuela

Dra. Elsa Rubio Espino

Dr. Sergio Suárez Guerra

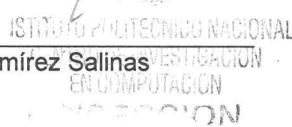
Dr. Herón Molina Lozano

Dr. Jesús Yaljá Montiel Pérez

Dr. Víctor Hugo Ponce Ponce

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Marco Antonio Ramírez Salinas





INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 6 del mes Diciembre del año 2017, el (la) que suscribe José Ángel Martínez Navarro alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B151174 , adscrito al Centro de Investigación en Computación , manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del Dr Juan Humberto Sossa Azuela y Dr. Elsa Rubio Espino y cede los derechos del trabajo intitulado Desarrollo de un robot de dos ruedas dinámicamente estable, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección josekun13@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.


José Ángel Martínez Navarro
Nombre y firma

Resumen

El rápido desarrollo e integración en el hogar que en la última década se ha presentado en el campo de la robótica resalta la necesidad de contar con plataformas que permitan desarrollar aplicaciones enfocadas a la interacción con humanos mediante robots de servicios.

En este trabajo se realizó el desarrollo de un prototipo robótico planeado como plataforma de desarrollo para aplicaciones que involucren interacción con humanos. La clase de robot seleccionada para este fin fue un péndulo invertido sobre dos ruedas debido a que es el modelo mas sencillo dentro de los conocidos como dinámicamente estables.

En este documento se explican los conceptos necesarios para entender el funcionamiento del prototipo así como el diseño de los componentes que lo integran.

Finalmente se realiza la descripción matématica del modelo dinámico del prototipo y se procede a la programación, utilizando como base un sistema embebido y el Robotic Operating System.

Los experimentos comprobaron que el robot es capaz de mantener el equilibrio ante perturbaciones dentro de un rango específico.

Finalmente se realizan recomendaciones para mejorar la funcionalidad e integrar mas tareas al robot. Lo anterior con base en la experiencia obtenida durante el desarrollo de este trabajo.

Abstract

The fast development and integration in home that in the last decade has presented in robotics highlights the need to have platforms that allow developing applications focused on human interaction through service robots.

In this thesis a robotic prototype was developed to be used as a development platform for applications that involve interaction with humans.

The class of robot selected for this purpose was an inverted pendulum on two wheels because it is the simplest model among those known as dynamically stable.

This document explains the necessary concepts to understand the operation of the prototype also the design of the components that make it up.

Finally the mathematical description of the dynamic model of the prototype is made and the programming is explained, using as base an embedded system and the Robotic Operating System.

The experiments proved that the robot is capable of maintaining equilibrium with disturbances in a specific range.

Finally recommendations are made to improve the functionality and integrate more tasks to the robot using the experience gained during the development of this work.

Agradecimientos

A a mi familia, especialmente a mi madre y a mis hermanos por siempre apoyarme, y a pesar de la distancia les mando un fuerte abrazo.

A mis directores el Dr. Juan Humberto Sossa Azuela y la Dr. Elsa Rubio Espino por guiarme durante estos dos años y medio de aventuras que ha sido la maestría.

A mis amigos dentro y fuera del CIC que han sido mi familia lejos de casa, por los consejos el apoyo y sobre todo por los buenos momentos.

Al Dr. Luis Yoichi Morales Saiki por la oportunidad de realizar una estancia internacional y a todos mis amigos en Nagoya por la increíble experiencia.

A los talleres de ESIME Zacatenco por el apoyo con la fabricación de las piezas y las clases de torno.

Al Instituto Politécnico Nacional (IPN) y la Secretaría de Investigación y Posgrado (SIP) por el apoyo económico brindado para llevar a cabo esta investigación.

Este trabajo fue apoyado económicamente por SIP-IPN (números 20171548 y 20170693) y CONACYT (número 155014[Investigación Básica] y número 65[Fronteras de la Ciencia]).

*Dedicado a todas las personas que mediante
la robótica forjan un futuro brillante
para la humanidad*

Índice general

Lista de figuras	X
Lista de tablas	XIII
Glosario de siglas	XV
Nomenclatura	XVII
1. Introducción	1
1.1. Justificación	1
1.2. Planteamiento del problema	1
1.3. Objetivos	2
1.3.1. Objetivo General	2
1.3.2. Objetivos Particulares	2
1.4. Contribuciones	2
1.5. Organización del documento	3
2. Estado del Arte	5
2.1. JOE	5
2.2. SEGWAY	5
2.3. UMBRA	6
2.4. NXTway-GS	7
3. Marco Teórico	9
3.1. Control Automático	9
3.1.1. El controlador PID	9
3.1.2. Modelado	10
3.2. Sensores	10
3.2.1. I.M.U.	10
3.2.2. Encoders	12
3.3. Motores	13
3.3.1. PWM	14
3.3.2. Etapa de potencia	14
3.4. Computadora	15
3.4.1. Raspbian	15
3.5. Robotic Operating System	15
4. Desarrollo	17
4.0.1. Funcionamiento general	17
4.1. Partes mecánicas	18
4.1.1. Estructura	18

4.1.2. Chumaceras	19
4.1.3. Ejes	20
4.2. Electrónica	21
4.2.1. Alimentación	21
4.2.2. Sensores	22
4.2.3. Etapa de potencia	24
4.3. Control	26
4.3.1. Modelo	26
4.3.2. Simulación	29
4.3.3. Controlador	30
4.4. Software	36
4.4.1. Nodos de ROS	36
5. Presentación y Discusión de Resultados	39
5.0.1. Aproximaciones previas	39
5.0.2. Metodología	39
5.0.3. Experimentos	40
5.0.4. Resultados	42
6. Conclusiones, recomendaciones y trabajo futuro	43
6.1. Conclusiones	43
6.2. Recomendaciones	44
6.3. Trabajo futuro	44
A. Programas	47
A.1. Modelo en espacio de estados	47
A.2. Nodo de Control (control2node)	48
A.3. Nodo que lee la IMU (imunode)	50
A.4. Nodo que genera la señal para los motores (motornode)	53
A.5. Nodo que procesa la señal de los encoders(encodernode)	54
B. Diagramas	57
Bibliografía	59
contenidos	

Índice de figuras

2.1.	Robot JOE	5
2.2.	SEGWAY HT	6
2.3.	Robot Loomo	6
2.4.	Robot UMBRA	6
2.5.	Robot NXTway-GS	7
3.1.	Diagrama de un controlador PID	9
3.2.	I.M.U.	11
3.3.	Acelerómetro Capacitivo	11
3.4.	Giroscopio	11
3.5.	Diagrama de un sistema I^2C	12
3.6.	Encoder	13
3.7.	Motor	13
3.8.	Etapa de potencia VNH5019	14
3.9.	Raspberry Pi	15
4.1.	Representación básica del funcionamiento del robot	17
4.2.	Forma básica del robot	18
4.3.	Estructura del robot	19
4.4.	Acercamiento al eje de las ruedas	19
4.5.	Detalles de la chumacera	20
4.6.	Detalles del eje	20
4.7.	Diagrama de bloques del sistema eléctronico	21
4.8.	Alimentación del sistema eléctronico	22
4.9.	Salida del <i>encoder</i>	23
4.10.	Conexión del <i>driver</i> VHN5019	24
4.11.	Esquema de un optoacoplador	25
4.12.	PCB central	25
4.13.	Desplazamiento angular con respecto a la vertical	26
4.14.	Desplazamiento angular de las ruedas	27
4.15.	Dirección del robot	27
4.16.	Esquema del modelo dinámico	29
4.17.	Esquema del servocontrol	30
4.18.	Esquema del control PID	31
4.19.	Ángulo de inclinación contra tiempo con una perturbación de 10 grados o 0.175 radianes.	32
4.20.	Ángulo de las ruedas contra tiempo con una perturbación de 10 grados.	32
4.21.	Ángulo de inclinación contra ángulo de las ruedas con una perturbación de 10 grados o 0.175 radianes.	33
4.22.	Ángulo de inclinación contra tiempo con una perturbación de 30 grados o 0.52 radianes. .	34

4.23. Ángulo de las ruedas contra tiempo con una perturbación de 30 grados.	34
4.24. Ángulo de inclinación contra ángulo de las ruedas con una perturbación de 30 grados o 0.52 radianes.	35
4.25. Diagrama de nodos de ROS.	36
B.1. Plano de la estructura.	57
B.2. Plano del eje.	58
B.3. Plano de la chumacera.	58
B.4. Esquema del PCB.	59

Índice de cuadros

3.1. Código Gray del 0 al 3	13
3.2. Características del motor Pololu 37D	14
3.3. Características del driver VHN5019	15
3.4. Características de la Raspberry Pi 2 B+	16
4.1. Requerimientos en el diseño del prototipo	18
4.2. Características del filtro pasa bajas	23
5.1. Experimentos realizados Pt1	40
5.2. Experimentos realizados Pt2	41

Glosario de siglas

- IMU Unidad de Medición Inercial por sus siglas en inglés.
- MEMS Sistemas Microelectromecánicos por sus siglas en inglés.
- SDA Línea Serial de Datos por sus siglas en inglés.
- SCL Línea de Reloj Serial por sus siglas en inglés.
- RPM Revoluciones por minuto.
- PCB Placa de Circuito Impreso por sus siglas en inglés.
- PID Proporcional Integrativo y Derivativo.
- PD+I Proporcional Derivativo mas Integrativo.
- LQR Regulador Linear Cuadrático por sus siglas en inglés.

Nomenclatura

- $g \text{ m/s}^2$ aceleración debida a la gravedad
- $m \text{ Kg}$ masa de la rueda
- $R \text{ m}$ radio de la rueda
- $jw \text{ Kgm}^2$ momento de inercia de la rueda
- $M \text{ Kg}$ peso del cuerpo
- $W \text{ m}$ ancho del cuerpo
- $D \text{ m}$ largo del cuerpo
- $H \text{ m}$ alto del cuerpo
- $L \text{ m}$ distancia del eje de la rueda al centro de masa
- $j\psi \text{ Kgm}^2$ momento de inercia del cuerpo (pitch)
- $j\phi \text{ Kgm}^2$ momento de inercia del cuerpo (yaw)
- $jm \text{ Kgm}^2$ momento de inercia del motor (propuesto)
- $R_m \Omega$ resistencia del motor
- $k_b \text{ Vseg/rad}$ constante contra electromotriz del motor (back EMF)
- $k_t \text{ Nm/A}$ constante de torque del motor
- n reducción del motor
- fm coeficiente de fricción entre el motor y el cuerpo(propuesto)
- fw coeficiente fricción entre las ruedas y el piso (propuesto)

Capítulo 1

Introducción

En este capítulo se explicarán los motivos que llevaron al desarrollo de este trabajo. Así como los objetivos alcanzados y la organización del documento.

1.1. Justificación

En la última década la robótica ha tenido grandes avances tanto en el área de investigación como en la industria. Actualmente podemos ver los primeros sistemas robóticos disponibles para su uso cotidiano en el hogar, sumado a lo anterior hay que tomar en cuenta el gran auge que ha tenido la inteligencia artificial, considerada por varios como la cuarta revolución industrial. Hoy en día en varias universidades y empresas se están desarrollando robots para realizar tareas cada vez mas complejas, pensando su aplicación en el hogar y la oficina. Lo anterior destaca la necesidad de contar con plataformas capaces de interactuar en ambientes con humanos.

El desarrollo de un robot dinámicamente estable involucra una serie de retos en el diseño de la estructura y del controlador, ha su vez este tipo de plataformas presentan cualidades muy deseables que le permiten realizar una amplia variedad de tareas en ambientes humanos desordenados [1].

El robot debe de ser fácil de utilizar y modificar, para que futuros usuarios se enfoquen en la tarea que quieren que realice sin preocuparse por la estabilidad del mismo, con el propósito de que sea utilizado para fines académicos y de investigación.

El prototipo diseñado en esta tesis fue pensando para aplicaciones que involucran interacción con humanos, como mensajero o mesero por dar unos ejemplos y la modularidad es clave para que esta plataforma pueda ser reutilizada.

1.2. Planteamiento del problema

El problema a resolver es el diseño y construcción de una plataforma robótica de desarrollo basada en el modelo del péndulo invertido sobre dos ruedas. La plataforma debe estar planeada tomando en cuenta el hecho de que interactuará con humanos. Debe de contar con un software que permita realizar modificaciones de una manera sencilla y debe ser lo suficiente mente versátil como para permitir que se hagan desarrollos utilizándola como base.

Se eligió este modelo de robot debido a que es el mas sencillo de los dinámicamente estables (el termino dinámicamente estable se refiere a que no cuenta con un punto de estable dentro de su rango de movimiento).

1.3. Objetivos

Los objetivos de este trabajo fueron planteados considerando que el prototipo será modificado y adaptado para diferentes tareas que mas adelante sean propuestas para continuar el desarrollo de este trabajo.

1.3.1. Objetivo General

Desarrollar una plataforma robótica dinámicamente estable que pueda ser utilizada posteriormente para implementar diferentes tareas enfocadas a la robótica de servicio.

1.3.2. Objetivos Particulares

1. Diseñar en C.A.D. una estructura robótica que sea robusta, ligera, económica y cuyas dimensiones sean adecuadas para la interacción con humanos.
2. Diseñar un sistema eléctronico embebido que permita el desplazamiento de manera autónoma del robot, y que a su vez permita un procesamiento rápido y eficiente para realizar las tareas encomendadas. Este sistema debe tener la característica de contar con dos partes aisladas, una para realizar el cómputo y mediciones, y la otra para proporcionarle la potencia necesaria a los actuadores.
3. Describir a través de un modelo matemático la dinámica del prototipo robótico diseñado.
4. Diseñar un controlador capaz de mantener en posición vertical el prototipo ha pesar de pequeñas perturbaciones externas.
5. Implementar los puntos anteriormente mencionados en un prototipo funcional.

1.4. Contribuciones

Los aportes de este trabajo son los siguientes.

- Robot funcional planeado para ser utilizado como plataforma de desarrollo.
- El robot puede ser utilizado para realizar pruebas y comparaciones entre diferentes modelos de controladores.
- El software del prototipo es modular y fácil de modificar.
- Una descripción de los sistemas necesarios para el funcionamiento del robot.
- El diseño de la estructura, chumaceras y ejes.
- El sistema eléctronico del prototipo así como una descripción del mismo

1.5. Organización del documento

A continuación se muestra una breve descripción de los temas ha tratar en los siguientes capítulos.

Capítulo 2 Estado del Arte

Se habla de los avances en el campo de la robótica que han servido como referencia para la realización del proyecto, empezando con el robot JOE, pasando por el SEGWAY, el robot UMBRA, etcétera.

Capítulo 3 Marco Teórico

Donde serán explicados los conceptos y herramientas que fueron empleadas para llevar acabo la construcción del prototipo.

Capítulo 4 Desarrollo

Describe el proceso de diseño y todo los que involucro él mismo, como es el diseño de las partes mecánicas, el sistema electrónico, el modelado, la programación, etc...

Capítulo 5 Precentación y Discusión de Resultados

Se muestran los resultados y se explica a detalle la metodología de los experimentos desarrollados para probar las calibrar el sistema de control y probar sus capacidades.

Capítulo 6 Conclusiones, Recomendaciones y Trabajo Futuro

Son explicados los resultados obtenidos en los experimentos y se plantean cuales son las futuras mejoras que se le pudieran hacer al prototipo.

Capítulo 2

Estado del Arte

En el presente capítulo se describirán varios de los robots mas sobresalientes construidos utilizando el modelo del péndulo invertido, los cuales han servido como referencia para este trabajo, así como también las aportaciones de cada uno de ellos al área de control automático y la robótica en general.

2.1. JOE

El robot JOE desarrollado por laboratorio de electrónica industrial del instituto federal Suizo de tecnología Lausanne E.P.F.L. como un experimento para probar si un péndulo invertido sobre dos ruedas era controlable, utilizaron un D.S.P. (procesador digital de señales) como circuito de control y como entradas del sistema un giroscopio junto con *encoders* en los motores probaron poder controlar el sistema utilizando un controlador lineal en espacio de estados [2].



Figura 2.1: Robot JOE

2.2. SEGWAY

El SEGWAY HT fue el primer robot de esta clase en comercializarse con éxito como un medio de transporte personal en ambientes urbanos, su sistema de control utiliza varios sensores para medir la inclinación entre ellos cinco giroscopios, dos de los cuales no son estrictamente necesarios pero por motivos de seguridad se incluyeron.



Figura 2.2: SEGWAY HT

Actualmente la empresa que desarrolla el SEGWAY HT ofrece otros productos que siguen este mismo modelo como el Segway Advanced Personal Robot conocido como Loomo, pensado como una plataforma de desarrollo para asistentes personales [3].



Figura 2.3: Robot Loomo

2.3. UMBRA

UMBRA es un robot creado utilizando el modelo del péndulo invertido sobre dos ruedas, fue desarrollado por el M. en C. Mauricio Ontiveros Rodríguez [4] como trabajo de tesis en el laboratorio de robótica y mecatrónica del Centro de Investigación en Computación del Instituto Politécnico Nacional.



Figura 2.4: Robot UMBRA

Este robot se destaca por utilizar un microcontrolador con un núcleo ARM Cortex M3 como unidad de procesamiento embebido y el uso de una IMU (unidad de medición inercial por sus siglas en Ingles)

para obtener el ángulo de inclinación del robot.

Las mayores contribuciones de este trabajo son la comparación de los métodos de control para este tipo de sistemas (sistemas no lineales), entre un control clásico PID y un control inteligente difuso, y el prototipo con la documentación correspondiente.

2.4. NXTway-GS

Fue robot desarrollado por Yorihisa Yamamoto [5] utilizando el Nxt y piezas de LEGO. La metodología que se utilizada en el diseño, así como el modelo matemático y el sistema de control propuestos para este robot se destacan por lo bien detallados que están.



Figura 2.5: Robot NXTway-GS

El autor proporciona una explicación excelente de como replicar su trabajo en una plataforma Ntx y los programas necesarios para hacerlo desde un principio e implementarlo si así se desea.

La principal aportación a este trabajo es el método de control y el modelo matemático que sirvieron como base para los desarrollados en este trabajo.

Capítulo 3

Marco Teórico

A continuación se explican las herramientas que fueron necesarias para la elaboración de este trabajo, entre las que se encuentran los sistemas de control, el modelado, los componentes electrónicos y el software.

3.1. Control Automático

El control automático es la disciplina de la ingeniería encargada de regular un sistema de manera autónoma. Existe una gran variedad de sistemas de control desde el gobernador centrífugo utilizado en las máquinas a vapor llegando a los programados en microcontroladores que hoy en día están por todas partes.

3.1.1. El controlador PID

Las siglas PID vienen de Proporcional, Integrativo y Derivativo, estos son los tres bloques fundamentales que conforman este controlador. Utilizando el error como señal de entrada para estos bloques se genera la señal de control, la cual reduce el error.

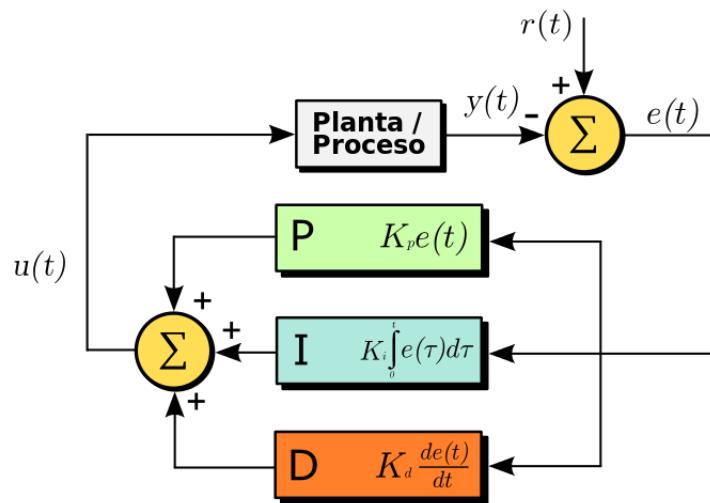


Figura 3.1: Diagrama de un controlador PID

El bloque proporcional aproxima la señal de control al valor deseado. El bloque derivativo obtiene la pendiente de la función que describe el error para suavizar la señal de control y por último el integrativo

que acelera los cambios en la señal de control.

Estos tres bloques contienen constantes que definen que tan significativos van a ser cada uno en el sistema de control, estas constantes se ajustan dependiendo el sistema que queramos controlar y pueden ser obtenidas manualmente o de manera analítica.

La utilidad de estos sistemas de control esta en que pueden ser aplicados a casi todos los sistemas de control. Especialmente cuando no se conoce el modelo matemático que lo represente [6].

3.1.2. Modelado

Una de las tareas mas importantes para poder controlar un sistema es crear una representación matemática de la dinámica que describe al sistema. Esta representación nos permite conocer mas a fondo como las leyes de la física lo rigen y ayuda para poder crear el modelo de control [6]. Esta representación debe ser lo suficiente aproximada a la realidad como para poder generar datos útiles al momento de diseñar el controlador y lo suficientemente sencilla como para que sea práctico trabajar con ella.

Ecuaciones de dinámicas de Lagrange

Las ecuaciones dinámicas de Lagrange son una herramienta que sirve para crear una descripción del sistema, nos permiten estimar la posición del robot en nuestro caso con base en las fuerzas aplicadas a este.

Para poder plantear el modelo utilizando esta herramienta es necesario contar con la descripción de las propiedades del sistema que deseamos modelar, estas son:

- Masa
- Fuerzas
- La geometría
- Los grados de libertad

3.2. Sensores

3.2.1. I.M.U.

La I.M.U. o Unidad de Medicion Inercial, es un sensor que mide la velocidad y orientación, esta compuesto de al menos un acelerómetro de tres ejes y un giroscopios igualmente de tres ejes.



Figura 3.2: I.M.U.

Su principal función es medir el ángulo de inclinación. El modelo que se utilizó en el prototipo fue una MPU-6050.

Acelerómetro

Los acelerómetros son sensores que miden la aceleración en una dirección, la IMU que se utilizó tiene un acelerómetro MEMS (Sistemas Microelectromecánicos), utilizan la variación de la capacitancia entre dos placas y una masa sísmica para obtener una medición.

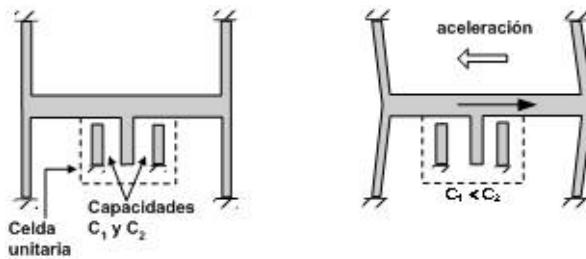


Figura 3.3: Acelerómetro Capacitivo

Giroscopio

El giroscopio mide en cambio en la orientación, en este caso del I.M.U. también es un MEMS, su funcionamiento consiste en medir el desplazamiento de una pequeña masa que cambia de posición con el giro del sensor.

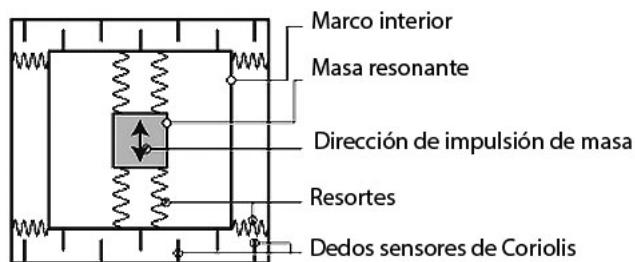


Figura 3.4: Giroscopio

Los giroscopios acumulan un error a lo largo del tiempo, con el fin de reducirlo se agrega un filtro que utiliza las mediciones de los acelerómetros para corregir la orientación obtenida por el sensor.

I^2C

I^2C es un protocolo de comunicación serial desarrollado por Philips en 1982, comúnmente utilizado para comunicación interna de circuitos integrados.

Su funcionamiento consiste en un bus el cual conecta a un maestro, generalmente un microcontrolador y varios esclavos mediante un par de cables, el maestro manda instrucciones a un esclavo utilizando una dirección para identificarlo y el esclavo responde al maestro, es posible tener varios maestros.

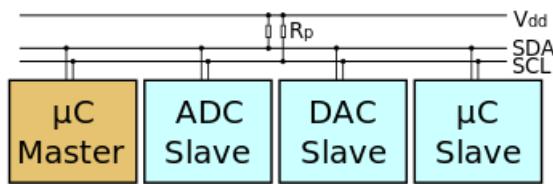


Figura 3.5: Diagrama de un sistema I^2C

Las principales características de este protocolo [7] son:

1. Velocidad estándar de 100Kbit/s, con un modo de alta velocidad *HS-mode* que alcanza los 3.4Mbit/s.
2. Voltaje típico de operación entre 1.2V a 5.5V.
3. Requiere de solamente dos líneas para establecer comunicación el SDA (*Serial Data*) y el SCL(*Serial Clock*).
4. La SDA se encarga de recibir y enviar la información entre los diferentes dispositivos
5. La SCL transmite una señal de reloj entre los dispositivos conectados con el fin de sincronizar el envío y recepción de información.
6. En la implementación estándar se pueden conectar hasta 112 dispositivos mediante este protocolo

Este es el protocolo utilizado por la I.M.U. para comunicarse con la computadora del robot.

3.2.2. Encoders

Los sensores que se utilizaron para medir la velocidad angular en el eje de un motor reciben el nombre de *encoder* (codificadores en Español). Porque la señal que generan de salida esta en código gray.

Los *encoders* que utiliza el robot generan códigos del cero al tres. Estos datos nos permiten estimar la posición y sentido de giro de manera indirecta utilizando una sumatoria.

Decimal	Gray
0	00
1	01
2	11
3	10

Cuadro 3.1: Código Gray del 0 al 3



Figura 3.6: Encoder

El circuito eléctrico del encoder esta aislado del motor para evitar interferencias y puede ser alimentado a diferentes voltajes.

En *encoder* obtiene las lecturas de posición midiendo la variación del campo magnético en un disco adherido a la flecha del motor.

3.3. Motores

Los motores que se utilizaron son modelo 37D marca Pololu de corriente directa. Se eligió este tipo de motor ya que es fácil controlar su velocidad y tienen una gran eficiencia energética.



Figura 3.7: Motor

Sus características son las siguientes [8].

Tamaño	37D por 70L mm
Peso	225 g
Diametro del eje	6 mm
Reducción	50:1
Velocidad sin carga a 6 V	100 RPM
Corriente sin carga a 6 V	250 mA
Corriente de operación a 6 V	2500 mA
Torque de operación a 6 V	0.60 Nm
Velocidad sin carga a 12 V	200 RPM
Corriente sin carga a 12 V	300 mA
Corriente de operación a 12 V	5000 mA
Torque de operación	1.20 Nm

Cuadro 3.2: Características del motor Pololu 37D

3.3.1. PWM

El PWM (modulación por ancho de pulso) es una técnica utilizada entre otras cosas, para controlar la velocidad de los motores de corriente directa.

Consiste en alimentar al motor con un tren de pulsos de la misma amplitud y frecuencia con un ancho variable, al reducir el ancho de los pulsos el valor efectivo del voltaje que entra al motor se reduce, lo que causa la disminución en su velocidad, lo inverso ocurre al incrementar el ancho de los pulsos.

3.3.2. Etapa de potencia

La computadora del robot no puede suministrar por si sola la potencia eléctrica necesaria a los motores. De ahí la necesidad de una etapa de potencia, la que consiste en un arreglo de transistores que amplifican la señal que es enviada al motor, la configuración más comúnmente utilizada se conoce como puente H. Otra de las funciones que se le da es proteger al circuito de control de alguna descarga o cambio drástico de corriente causado por el motor. Actualmente existen circuitos integrados especialmente diseñados como etapas de potencia algunos incluyen la capacidad de trabajar con protocolos de comunicación I^2C y SPI.

En el prototipo se utilizaron etapas de potencia Pololu VNH5019, tienen la capacidad de manejar 12 Ampres de salida continua en un rango de 5.5 a 24 Voltios.



Figura 3.8: Etapa de potencia VNH5019

Las características de esta etapa de potencia son las siguientes [9].

Canales	1
Voltaje mínimo de operación	5.5 V
Voltaje máximo de operación	24 V
Salida continua de corriente	12 A
Pico en corriente de salida	30 A
Frecuencia máxima de PWM	20 kHz
Protección contra voltaje de reversa	Si

Cuadro 3.3: Características del driver VHN5019

3.4. Computadora

La computadora elegida para controlar al robot es una Raspberry Pi. Esta computadora en miniatura fue desarrollada con el propósito de facilitar el desarrollo de prototipos, así como permitir el acceso a Internet con muy bajo costo, esto con el fin de apoyar la educación en países con escasos recursos. En la actualidad es utilizada por muchas universidades así como por inventores para realizar prototipos y enseñar programación [10].

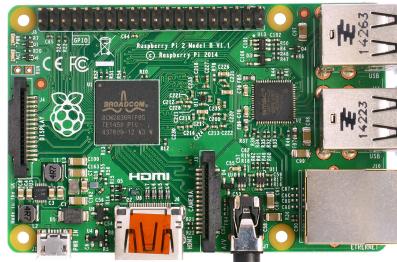


Figura 3.9: Raspberry Pi

El modelo que se utilizó es el 2 B+.

3.4.1. Raspbian

El sistema operativo recomendado por los fabricantes de la Raspberry Pi se llama Raspbian, que es una versión compacta de GNU/Linux Debian. Raspbian fue especialmente creado para esta computadora, la interfaz es prácticamente igual a la de Debian y soporta la gran mayoría del software compatible con esta familia de distribuciones.

3.5. Robotic Operating System

El Robotic Operating System conocido comúnmente como ROS empezó siendo un proyecto de la universidad de Stanford en el año 2007. Actualmente es utilizado alrededor del mundo por universidades

CPU	ARM11 ARMv6 700 MHz
GPU	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0
RAM	512 MB LPDDR SDRAM 400 MHz
USB 2.0	4
Salidas de video	HDMI 1.4 @ 1920x1200 píxeles
Almacenamiento	microSD
Ethernet	10/100 Mbps
Tamaño	85,60x56,5 mm
Peso	45 g

Cuadro 3.4: Características de la Raspberry Pi 2 B+

y empresas para desarrollar robots.

ROS es un marco de trabajo flexible compuesto por herramientas, librerías y convenciones [11], el cual permite crear software de una manera modular por medio de nodos que realizan tareas específicas.

El utilizar ROS permite la rápida implementación de cambios en los prototipos, así como la capacidad de reutilizar y compartir código. Hoy en día existen repositorios de ROS en línea con una gran variedad de funciones, desde control de servomotores hasta cartografía en tres dimensiones por medio de láser.

Capítulo 4

Desarrollo

En este capítulo se habla del diseño, construcción y puesta en marcha del robot. Así también de discutirán las decisiones que llevaron a este prototipo y puntos clave en su construcción.

4.0.1. Funcionamiento general

El funcionamiento general del prototipo se puede observar en el siguiente diagrama de flujo.

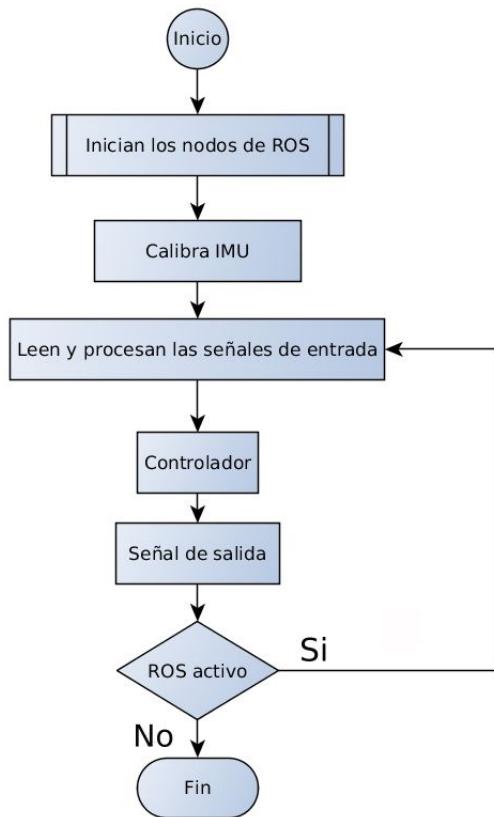


Figura 4.1: Representación básica del funcionamiento del robot

Básicamente los sensores miden la desviación del punto de equilibrio ubicado en la vertical y el controlador compensa esas desviaciones. Mientras el sistema siga activo este ciclo se repite.

4.1. Partes mecánicas

Lo primero que se consideró para la construcción del robot fue la forma. Debido a las tareas que se espera que realice y el ambiente en el que va a trabajar, se decidió la forma básica que tendrá así como la altura y peso aproximado.

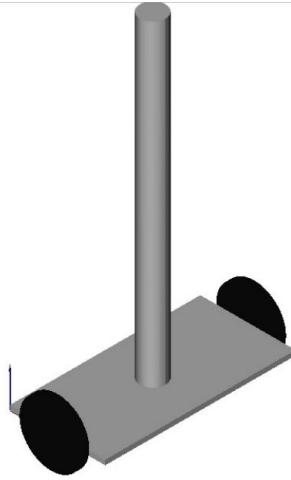


Figura 4.2: Forma básica del robot

Características	Valores
Peso	4kg 10kg
Altura	1m 1.5m
Material	Aluminio
Motores	Corriente directa
Alimentación	Batería de LiPO

Cuadro 4.1: Requerimientos en el diseño del prototipo

Para llegar a estos requerimientos se consideró lo siguiente. La altura debe ser la aproximada a la de una mesa o estante para poder colocarle objetos encima, debe ser ligero para que sea fácil de transportar y reducir el consumo de corriente en los motores, el material con el que se construya debe ser fácil de maquinar y la forma del prototipo tiene que ser sencilla para facilitar el ensamblaje y modelado.

Los requerimientos anteriores son un criterio de diseño tomado por mi parte con base en las necesidades que pueden surgir en ambientes con humanos.

4.1.1. Estructura

La estructura consiste de dos partes principales. La placa base y la columna. La placa base cumple la función de sostener todos los componentes eléctricos del robot así como a la columna y los ejes. Está colocada abajo de los ejes para que el robot tenga un centro de gravedad bajo, lo que le brinda mayor estabilidad. Esta hecha de aluminio de 1/4 de pulgada de espesor, mide 40 por 60 cm y tiene una masa de 2.056 Kg.

La columna es un perfil estándar de aluminio comúnmente conocido como IPS [12], su función principal es brindar mas estabilidad a la estructura y sostener los aditamentos que posteriormente se le incorporen. Los perfiles estándar de aluminio tienen ranuras que facilitan el adherirles otros objetos, este perfil mide 2 por 2 cm y 1 m de largo, su masa es de 1.6 Kg.

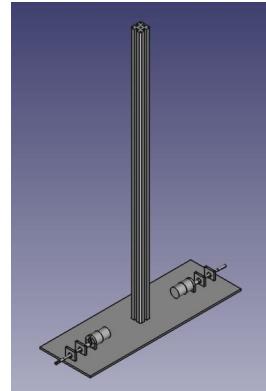


Figura 4.3: Estructura del robot

Cuando se diseño la estructura se tomaron en consideración los motores requeridos para mover al robot, por esta razón se agregó un eje que conecta la flecha del motor con las ruedas y dos chumaceras para cada eje. Las chumaceras transfieren el peso del robot directamente a los ejes y al utilizar dos por eje se restringen los movimientos que estos pudieran tener, esto con el motivo de no generar esfuerzos adicionales en el motor y proteger el sistema de engranajes del motorreductor. Finalmente para reducir las vibraciones y aumentar la tolerancia de diseño en las chumaceras se utilizaron coples flexibles para unir los ejes al motor.



Figura 4.4: Acercamiento al eje de las ruedas

4.1.2. Chumaceras

Las chumaceras tienen que resistir el peso del robot y las fuerzas que se generen en los ejes. La pieza es sencilla y rápida de fabricar, están hechas de dos partes, la primera es un rodamiento que permite el giro del eje y la segunda es en si la estructura de la chumacera.

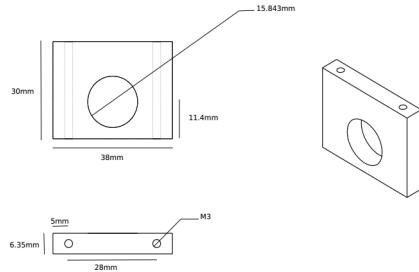


Figura 4.5: Detalles de la chumacera

La estructura fue elaborada mediante impresión 3D con plástico PLA, esto por la rapidez del proceso. La resistencia del material se puso a prueba con una pieza la cual soportó 95 Kg. La impresión está hecha con 75 % de densidad y un patrón de relleno hexagonal.

El diámetro interior del rodamiento es de 1/4 de pulgada y el exterior es de 5/8 de pulgada, para que el rodamiento se mantenga adherido a la estructura el diámetro interno de esta es 2 milésimas partes más pequeño que el exterior del rodamiento.

4.1.3. Ejes

Los ejes deben soportar la mayor tensión en todo el prototipo, su ancho está determinado por la flecha del motor en un extremo y el cople de la rueda en el otro. Y deben ser lo suficientemente largos para atravesar las dos chumaceras.

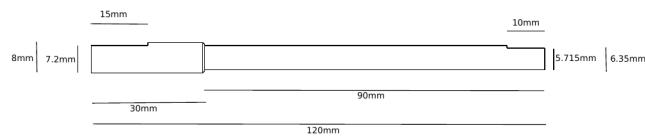


Figura 4.6: Detalles del eje

Tomando en cuenta el ancho de la pieza, la rigidez con la que debe contar y pensando en que no sufra deterioro por estar en contacto con el aire se decidió utilizar acero inoxidable para fabricarlos. Este material tiene las cualidades que se requieren pero es difícil de trabajar debido a su dureza.

La pieza fue realizada en torno con el apoyo de los talleres de la sección de posgrados de la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Zacatenco. El proceso se realizó en tres etapas. La primera consistió en rebajar una varilla de acero inoxidable hasta aproximadamente el diámetro interno del cople de las ruedas, la segunda consistió en separar la varilla en los dos ejes y la ultima en ajustar un extremo al diámetro exterior de la flecha del motor y ajustar todas las medidas para que calzaran en sus respectivos lugares.

4.2. Electrónica

La electrónica del robot consiste en tres partes, la alimentación, la etapa de potencia y el circuito de control.

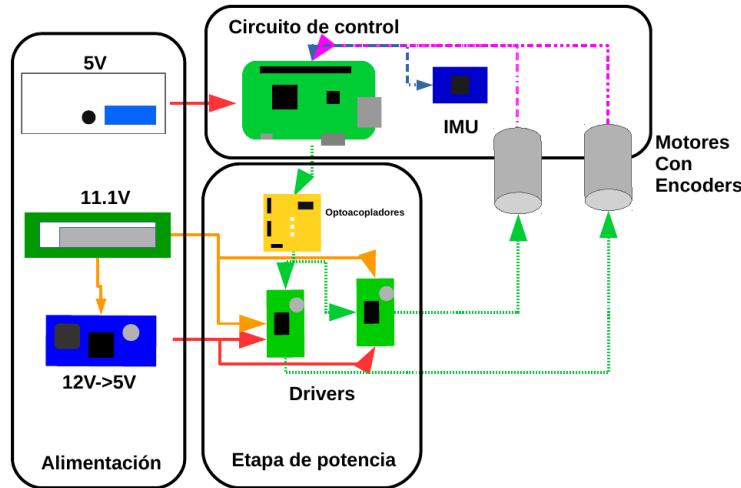


Figura 4.7: Diagrama de bloques del sistema eléctrico

En las siguientes secciones se profundizará en cada una de las partes que componen el sistema eléctrico

4.2.1. Alimentación

La alimentación del sistema es mediante dos baterías de litio polímero. La primera es una de 5V con una capacidad de salida de 2A, este tipo de batería es utilizada para cargar teléfonos móviles y su terminal de carga así como su puerto de salida son USB, se eligió esta por su duración y porque suministra suficiente corriente a la Raspberry Pi.

La segunda es una batería de 3 celdas y 11.1V con una carga de 5200mAh, esta batería tiene una capacidad de descarga máxima recomendada de 52A.

El sistema de alimentación es complementado por convertidor DC-DC que reduce el voltaje de 11.1V a 5V para poder alimentar los circuitos lógicos de los Drivers con el fin de mantener aislada eléctricamente la etapa de potencia y los motores del circuito de control. El convertidor es una fuente conmutada de tipo BUCK [13].

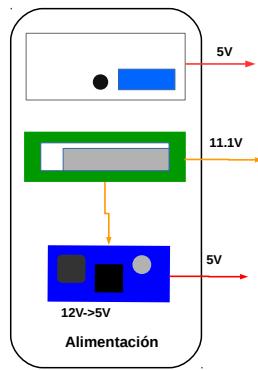


Figura 4.8: Alimentación del sistema eléctronico

Se utilizan dos baterías con el fin de mantener aisladas la parte de potencia y de control, esto con el fin de evitar interferencias y proteger a la computadora de variaciones en la corriente.

4.2.2. Sensores

La estabilidad del robot depende en gran medida de una rápida y precisa medición del ángulo de inclinación y la posición de las ruedas. Por lo cual es fundamental considerar los factores que inducen ruido en las mediciones y reducirlos lo mas posible.

I.M.U.

La *I.M.U.* que se utilizó en este prototipo es una ITG/MPU 6050 [14], con una velocidad de muestreo de 8Khz y comunicación mediante I^2C . Debido a la manera en que están construidos el acelerómetro y giroscopio del sensor este se vuelve sensible a las perturbaciones y el error que estas generan se acumula.

El acelerómetro es sensible a las vibraciones y los pequeños movimientos, intensificando de manera desproporcionada las lecturas, para compensar esto el sensor tiene un filtro pasa bajas digital incluido en el hardware el cual se puede configurar, en el prototipo se delimitó a un rango de $\pm 4g$.

A la vez que cuenta con un filtro pasa altas que es aplicado al acelerómetro y giroscopio, el cual ayuda a delimitar la velocidad de los movimientos que se desean medir. Pero conlleva un retraso en la velocidad de lectura, la cual se muestra en la siguiente tabla [15].

Para obtener la aceleración angular del sensor se utilizan las mediciones de los tres ejes conforme a la siguiente ecuación.

$$\omega_{acel} = \tan^{-1} \frac{Ax}{\sqrt{Ay^2 + Az^2}} \quad (4.1)$$

En este caso solo utilizamos la aceleración angular sobre el eje x .

La medición que proporciona el giroscopio de la variación en el ángulo de inclinación es estable y aproximada a la realidad. A esta se le aplica una operación integral para obtener la inclinación. Los cambios rápidos en la inclinación producen un error en las lecturas del giroscopio ya que este no regresa de manera instantánea a la posición real, a este error se le añaden las desviaciones que se acumulan en el tiempo debido a la integración.

Acelerómetro	Acelerómetro	Giroscopio	Giroscopio
Ancho de banda (Hz)	Retraso (ms)	Ancho de banda (Hz)	Retraso (ms)
260	0	256	0.98
184	2	188	1.9
94	3	98	2.8
44	4.9	42	4.8
21	8.5	20	8.3
10	13.8	10	13.4
5	19	5	18.6

Cuadro 4.2: Características del filtro pasa bajas

Para reducir obtener una lectura mas precisa de la inclinación se utiliza un filtro de la siguiente forma.

$$\psi = \alpha * (\psi_{-1} + \omega_{gyro} * \Delta t) + (1 - \alpha) * (\omega_{acel}) \quad (4.2)$$

$$\alpha = \frac{\tau}{\tau + \Delta t} \quad (4.3)$$

Donde τ es una constante de tiempo mucho mayor al tiempo del ruido en el acelerómetro y Δt es el intervalo de tiempo entre las mediciones [16].

Encoders

Los *encoders* que se utilizan en el prototipo vienen integrados al motor en la parte trasera y consisten de dos partes. Un disco magnético fijo en el eje del motor y un sensor de efecto Hall que detecta las variaciones en el campo magnético.

El *encoder* genera 64 pulsos por revolución del motor.

Figura 4.9: Salida del *encoder*

La salida tiene dos canales, esto nos permite obtener la dirección del giro en el eje del motor.

Hay que tomar en cuenta la reducción que tiene la flecha del motor para obtener la posición en la

que se encuentra. En el prototipo se utilizan motores la reducción de 50:1.

Lo que significa que genera 3200 pulsos por revolución de la flecha.

$$64 * 50 = 3200 \quad (4.4)$$

Y por lo que existe una relación de grados por pulso que es de 0.1125.

$$\frac{360}{3200} = 0,1125 \quad (4.5)$$

Tenemos que considerar también que la velocidad máxima del motor es de 200 RPM por lo que la computadora debe ser capaz de procesar 10666.6666667 lecturas por segundo.

$$\frac{3200 * 200}{60} = 10666,6666667 \quad (4.6)$$

Se utiliza la lectura actual y la anterior para poder determinar la dirección de giro y en base a eso se incrementa o decrementa un acumulador. El cual es multiplicado por la relación de grados por pulso.

La velocidad con la que es procesada la información que generan los *encoders* es un factor clave para evitar errores al momento de realizar el control del prototipo, debido a que estos se presentarían en los momentos cuando la velocidad del motor sea elevada.

4.2.3. Etapa de potencia

En el prototipo se utilizaron dos *drivers* para motores VHN5019 de la marca Pololu de un canal. Se seleccionó este modelo por la corriente de salida que puede soportar y que puede manejar un voltaje diferente para la lógica.

Los motores tiene una corriente de operación máxima sostenida de 5 A, para asegurar la vida útil del *driver* este tiene que utilizarse a un 80 % de su capacidad como máximo. Esto significa que el *driver* debe ser capaz de mantener una corriente de salida sostenida de 6.25 A.

$$\frac{5}{80} * 100 = 6,25 \quad (4.7)$$

Dentro de esta familia de *drivers* se seleccionó el modelo utilizando como criterio la corriente de salida sostenida de 6 A y que el costo de los dispositivos es prácticamente el mismo.

Este dispositivo tiene la capacidad de manejar diferente voltaje lógico y de salida. Lo que representa una gran ventaja debido a que los motores trabajan a 12 V y las señales de control que envía la computadora son a 5 V. Aunque presenta la desventaja de que la referencia de voltaje es la misma.

La conexión es como se muestra en la siguiente imagen.

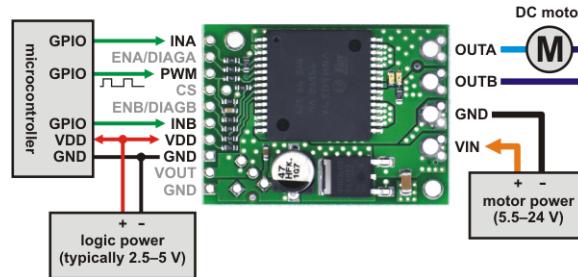


Figura 4.10: Conexión del *driver* VHN5019

La velocidad del motor es controlada por PWM, las entrada A y B determinan el sentido del giro. Cuando ambas son iguales el motor se frena. Este freno no fue utilizado en el prototipo debido a que en todo momento debe ser capaz de corregir el ángulo de inclinación para mantenerse vertical.

Aislamiento eléctrico

A pesar de que los *drivers* proporcionan protección a la computadora contra las variaciones de corriente, los motores generaban gran cantidad de ruido en las mediciones de la IMU por lo que se decidió aislar eléctricamente el circuito de control y los motores.

Se utilizaron optoacopladores los cuales consisten de un led y un fototransistor encapsulados, en el momento que el led emite luz el fototransistor cierra el circuito proporcionando así una transferencia de información sin necesidad de tener fuentes de alimentación comunes entre los circuitos.

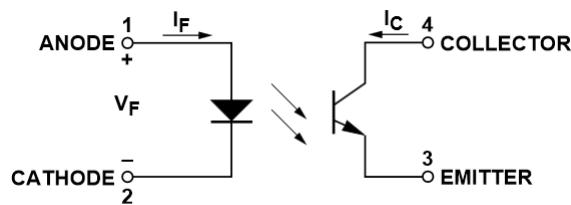


Figura 4.11: Esquema de un optoacoplador

Estos fueron montados en un PCB especialmente diseñado para el robot. El cual también es utilizado como punto de conexión central entre los circuitos que integran el prototipo, con el fin de facilitar el manejo, reducir la cantidad de cableado necesario, las posibles fallas, facilitar la inspección y la modificación de los circuitos.

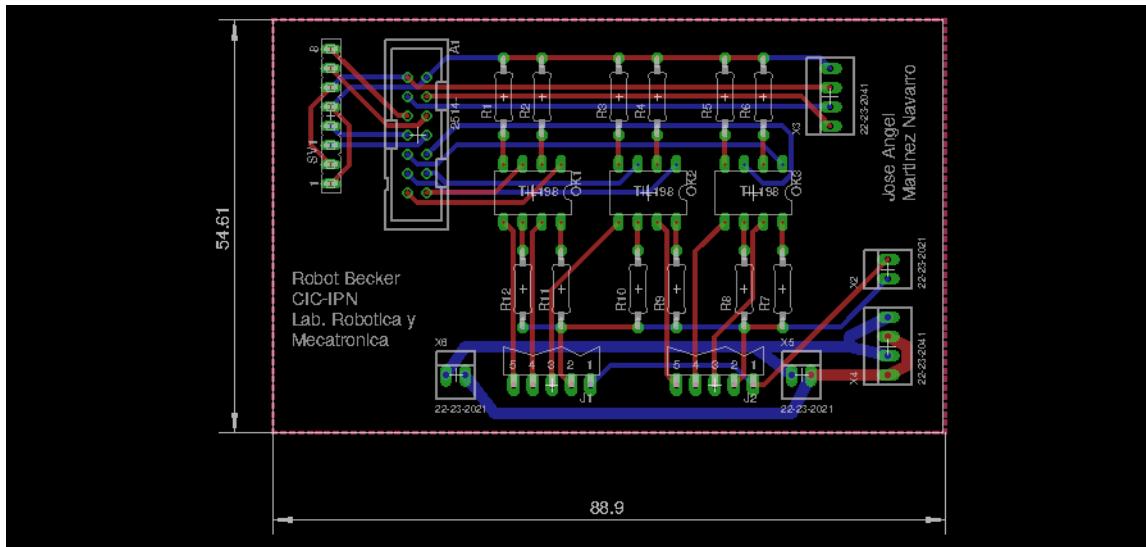


Figura 4.12: PCB central

El PCB fue diseñado utilizando la paquetería de Eagle, los esquemáticos se encuentran en el apéndice B.

4.3. Control

Para el control del prototipo se decidió utilizar un enfoque basado en el control clásico inspirado por el trabajo de Yorihisa Yamamoto [5].

El tamaño y las fuerzas que actúan sobre el robot hacen de este un sistema que reacciona de manera muy diferente a otro péndulos invertidos de menor escala. Por lo que es fundamental contar con un modelo dinámico del sistema para poder proponer un esquema de control eficaz.

4.3.1. Modelo

Para modelar el sistema se partió de la suposición que un péndulo invertido en un caso ideal tiene uno de sus puntos de equilibrio en la posición vertical, en la realidad esto no sucede debido a que la menor perturbación en el sistema lo saca de su punto de equilibrio.

Debido a que entre mas cerca se encuentre el péndulo de su punto de equilibrio requiere una menor cantidad de energía para mantener la vertical y que la inercia de la estructura es un factor fundamental a considerar para acercarse a este punto de equilibrio se decidió utilizar ecuaciones de Lagrange para crear el modelo dinámico del sistema, esto nos permite representar de manera sencilla el movimiento del prototipo.

Los grados de libertad del modelo dinámico son 3 y están representados por las letras griegas ψ , θ y ϕ .

La primera representa el desplazamiento angular del centro de masa con respecto a la vertical.

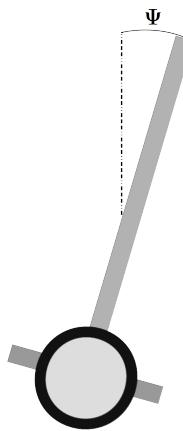


Figura 4.13: Desplazamiento angular con respecto a la vertical

La segunda representa el desplazamiento angular de las ruedas con respecto a la estructura.

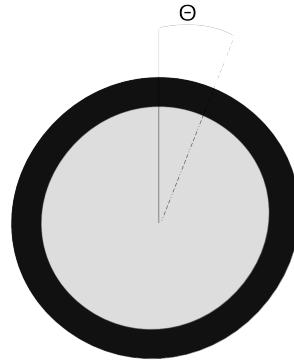


Figura 4.14: Desplazamiento angular de las ruedas

La Tercera representa el la dirección de giro del robot.

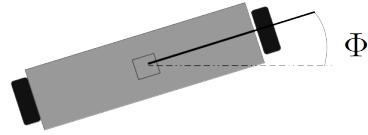


Figura 4.15: Dirección del robot

Fuerzas que actúan en el modelo

Las fuerzas que actúan en nuestro modelo se definen con las siguientes ecuaciones (El significado de las variables y sus unidades se pueden encontrar en la nomenclatura).

$$(F_\theta, F_\psi, F_\phi) = (F_l + F_r, F_\psi, \frac{\omega}{2 * R} (F_r - F_l)) \quad (4.8)$$

Fuerza en la rueda izquierda

$$F_l = nk_t i_l + fm(\dot{\psi} - \dot{\theta}_l) - fw\dot{\theta}_l \quad (4.9)$$

Fuerza en la rueda derecha

$$F_r = nk_t i_r + fm(\dot{\psi} - \dot{\theta}_r) - fw\dot{\theta}_r \quad (4.10)$$

Fuerza en el centro de masa del péndulo

$$F_\psi = -nk_t i_l - nk_t i_r - fm(\dot{\psi} - \dot{\theta}_l) - fm(\dot{\psi} - \dot{\theta}_r) \quad (4.11)$$

Utilizando las características brindadas por el fabricante de los motores se realiza una equivalencia entre el voltaje de entrada en los motores y el torque a la salida.

$$L_m \dot{i}_{l,r} = v_{l,r} + k_b(\dot{\psi} - \dot{\theta}_{l,r}) - R_m \dot{i}_{l,r} \quad (4.12)$$

Considerando que la inductancia es muy pequeña simplificamos la ecuación de la siguiente manera.

$$\dot{i}_{l,r} = \frac{v_{l,r} + k_b(\dot{\psi} - \dot{\theta}_{l,r})}{R_m} \quad (4.13)$$

Como resultado obtenemos las nuevas ecuaciones en función del voltaje.

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + fw)\dot{\theta} + 2\beta\dot{\psi} \quad (4.14)$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \quad (4.15)$$

$$F_\phi = \frac{\omega}{2R}\alpha(v_r - v_l) - \frac{\omega^2}{2R^2}(\beta + fw)\dot{\phi} \quad (4.16)$$

$$\alpha = \frac{nk_t}{R_m} \quad (4.17)$$

$$\beta = \frac{nk_t k_b}{R_m} + fm \quad (4.18)$$

Velocidades angulares

Se realizó un análisis por espacio de estados del sistema para obtener las derivadas que representan las velocidades angulares en el modelo.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (4.19)$$

$$y(t) = C(t)x(t) + D(t)u(t) \quad (4.20)$$

Forma general de la reprecentación por espacio de estados

Este análisis permite saber si el sistema es controlable y observable como lo es en este caso, Este fue realizado utilizando Matlab y el código se encuentra en el anexo de este documento.

Las velocidades angulares del sistema quedan expresadas de la siguiente manera.

Velocidad angular promedio de las ruedas

$$\dot{\theta} = A_1(3, 2)\psi + A_1(3, 3)\dot{\theta} + A_1(3, 4)\dot{\psi} + b_1(3)v_l + b_1(3)v_r \quad (4.21)$$

Velocidad angular de la inclinación del péndulo

$$\dot{\psi} = A_1(4, 2)\psi + A_1(4, 3)\dot{\theta} + A_1(4, 4)\dot{\psi} + b_1(4)v_l + b_1(4)v_r \quad (4.22)$$

Velocidad angular de rotación (dirección del giro)

$$\dot{\phi} = \frac{-J}{I}\dot{\phi} + \frac{-k}{I}v_l + \frac{k}{I}v_r \quad (4.23)$$

Valor de las constantes.

$$A_1(3, 2) = \frac{-gMLE(1, 2)}{\det(E)} \quad (4.24)$$

$$A_1(3, 3) = \frac{-2[(\beta + fw)E(2, 2) + \beta E(1, 2)]}{\det(E)} \quad (4.25)$$

$$A_1(3, 4) = \frac{2\beta[E(2, 2) + E(1, 2)]}{\det(E)} \quad (4.26)$$

$$b_1(3) = \frac{\alpha[E(2, 2) + E(1, 2)]}{\det(E)} \quad (4.27)$$

$$b_1(4) = \frac{\alpha[E(1, 1) + E(1, 2)]}{\det(E)} \quad (4.28)$$

$$E = \begin{bmatrix} (2m + M)R^2 + 2jw + 2n^2jm & MLR - 2n^2jm \\ MLR - 2n^2jm & ML^2 + j_\psi + 2n^2jm \end{bmatrix} \quad (4.29)$$

4.3.2. Simulación

El modelo se simuló utilizando el paquete Simulink de Matlab

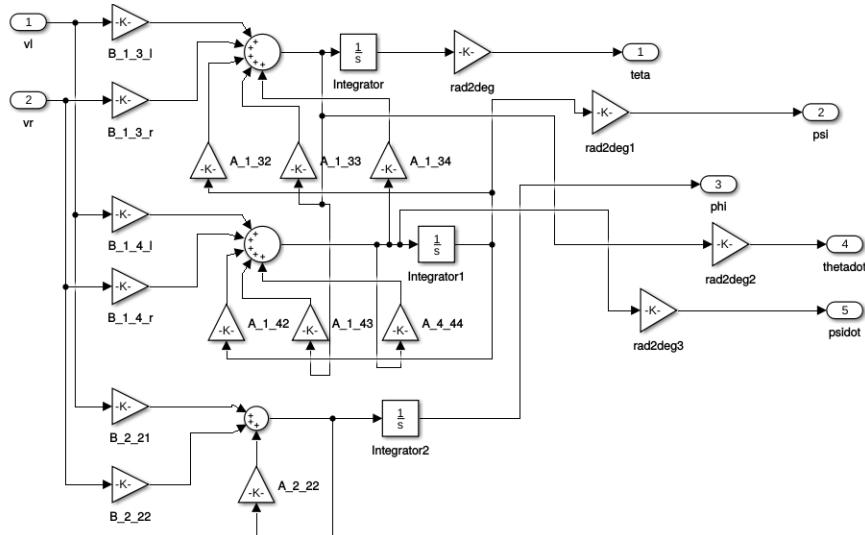


Figura 4.16: Esquema del modelo dinámico

Las entradas del sistema son.

- V_l es el voltaje de entrada en el motor izquierdo.
- V_r es el voltaje de entrada en el motor derecho.

Comúnmente en esta clase de modelos la entrada al sistema es el torque proporcionado por los motores.

Pero ya que contamos con la relación del voltaje al torque en el motor y que en realidad la señal de control proporcionada al sistema es el voltaje de entrada de a los motores se consideró que los parámetros del controlador se aproximarán mas en la simulación a los del prototipo utilizando como entrada al modelo los voltajes de alimentación de cada motor.

Las salidas del sistema son las siguientes.

- teta θ ángulo promedio entre la estructura y las ruedas.
- psi ψ ángulo de inclinación de la estructura.
- phi ϕ ángulo de giro del robot.
- teta punto $\dot{\theta}$ velocidad angular promedio entre la estructura y las ruedas.
- psi punto $\dot{\psi}$ velocidad angular de inclinación de la estructura.

El modelo tiene como salida no solamente las mediciones de los ángulos, las cuales en la práctica son lo que obtendríamos de los sensores. También tenemos las velocidades angulares que en el prototipo calculan utilizando el método numérico de la derivada sucia [17]. Se hizo de esta manera para poder obtener mas fácilmente las lecturas en el modelo simulado y debido a que los módulos de derivada pueden acarrear varios problemas en Matlab al momento de simular.

4.3.3. Controlador

El controlador utilizado es una variación del PID llamada PD+I. En esta configuración se toma parte de la señal de entrada y se pasa por un integrador.

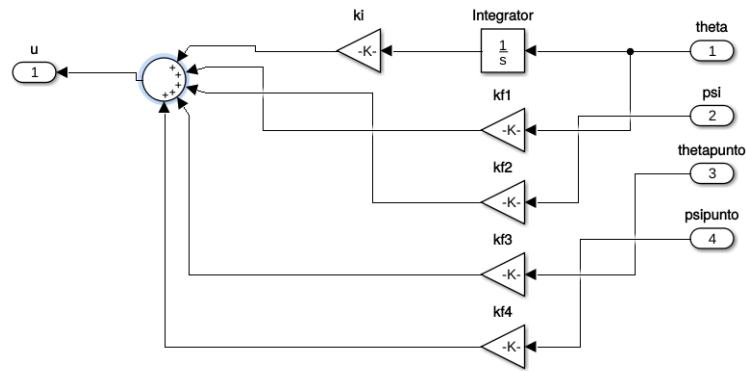


Figura 4.17: Esquema del servocontrol

En La simulación se tomaron las derivadas directamente del modelo. En la práctica son calculadas dentro del módulo de control, se retomará el tema con mayor detalle en la siguiente sección de este documento.

La parte proporcional y derivativa aplicadas a la señal de error proveniente de la IMU (ψ) tienen la función de reducir las perturbaciones en el ángulo de inclinación de la estructura y compensar la inercia de la misma.

En cuanto al control aplicado sobre el ángulo de las ruedas con respecto a la estructura (θ) es el de reducir las vibraciones y eliminar posibles sobre impulsos.

Para ajustar el valor de las constantes se utilizó un método conocido como LQR (Linear-Quadratic Regulator). Este método se utiliza en la teoría de control óptimo para poder obtener las constantes de control que requieren un menor costo. El LQR se utiliza en sistemas de ecuaciones diferenciales lineales.

Para calcular las constantes mediante el método LQR se utilizaron funciones predefinidas en Matlab, el

código se encuentra en el Anexo de este documento.

Las constantes obtenidas para el modelo fueron las siguientes.

- $K_i = 0.7071$

- $K_{f1} = 1.3933$

- $K_{f2} = -74.268$

- $K_{f3} = 1.6141$

- $K_{f4} = -1.3836$

El modelo con el controlador quedaron de la siguiente forma.

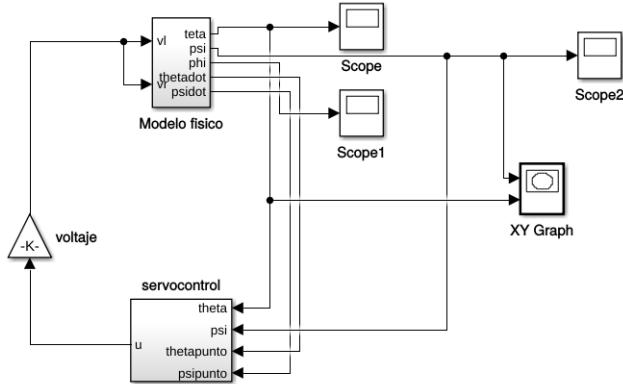


Figura 4.18: Esquema del control PID

Podemos observar una constante de voltaje antes de la entrada el modelo, esta constante tiene la función de escalar el voltaje de entrada a los motores ya que en la práctica el voltaje máximo suministrado a cada uno es de 11.1V, lo que es equivalente al 100 % del PWM.

Se realizaron pruebas con la simulación. Se introdujo una perturbación inicial en el primer caso de 10 grados, en el segundo de 30 grados.

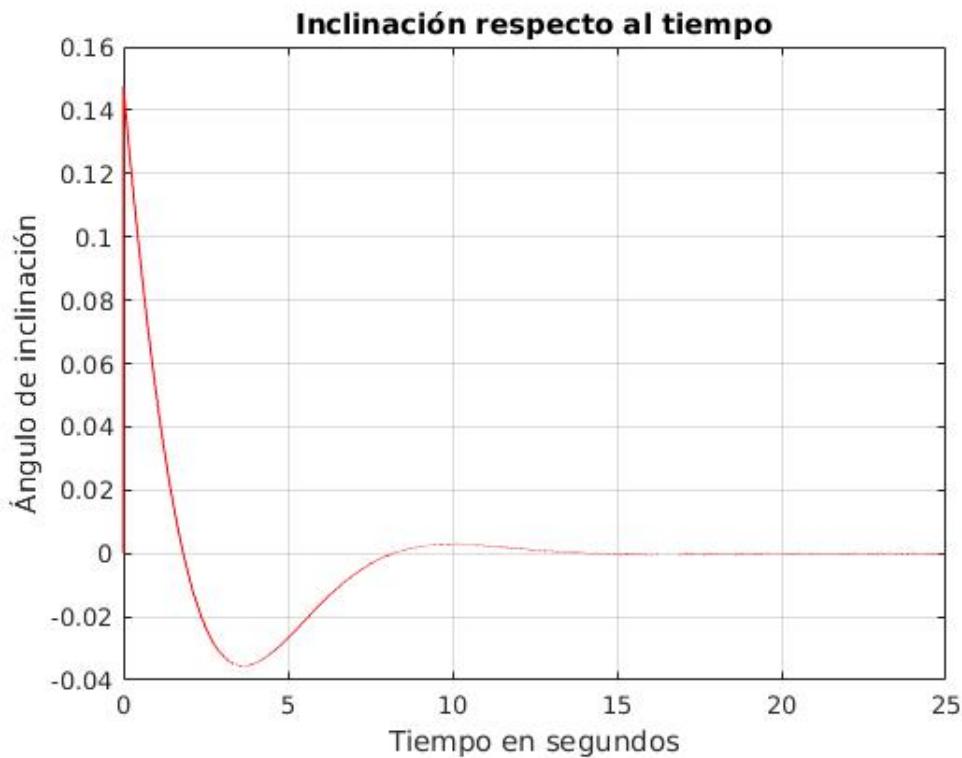


Figura 4.19: Ángulo de inclinación contra tiempo con una perturbación de 10 grados o 0.175 radianes.

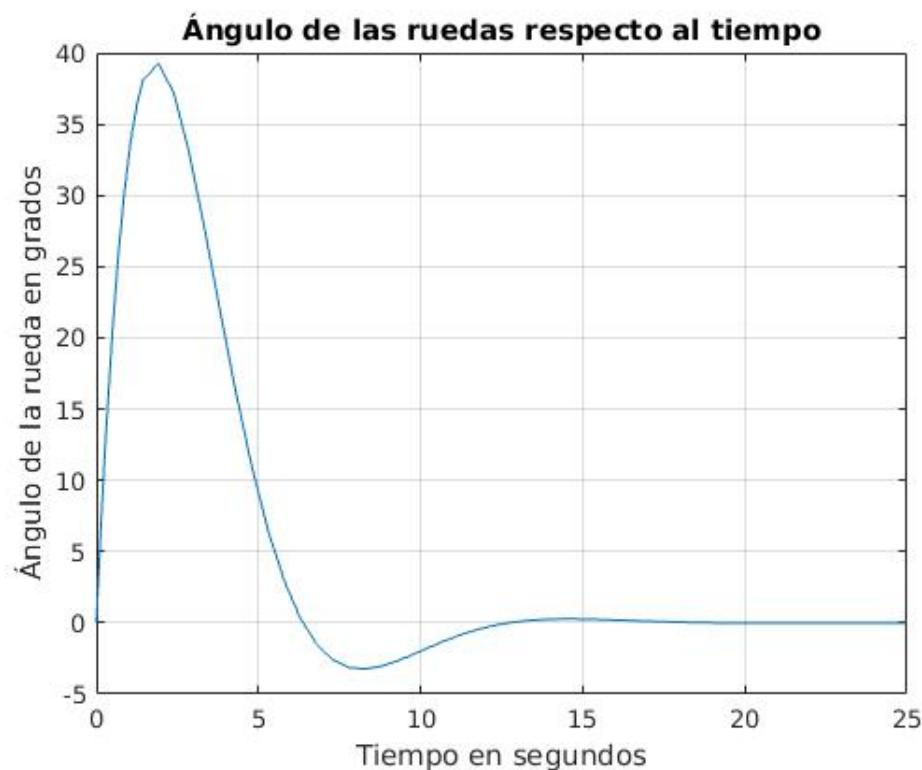


Figura 4.20: Ángulo de las ruedas contra tiempo con una perturbación de 10 grados.

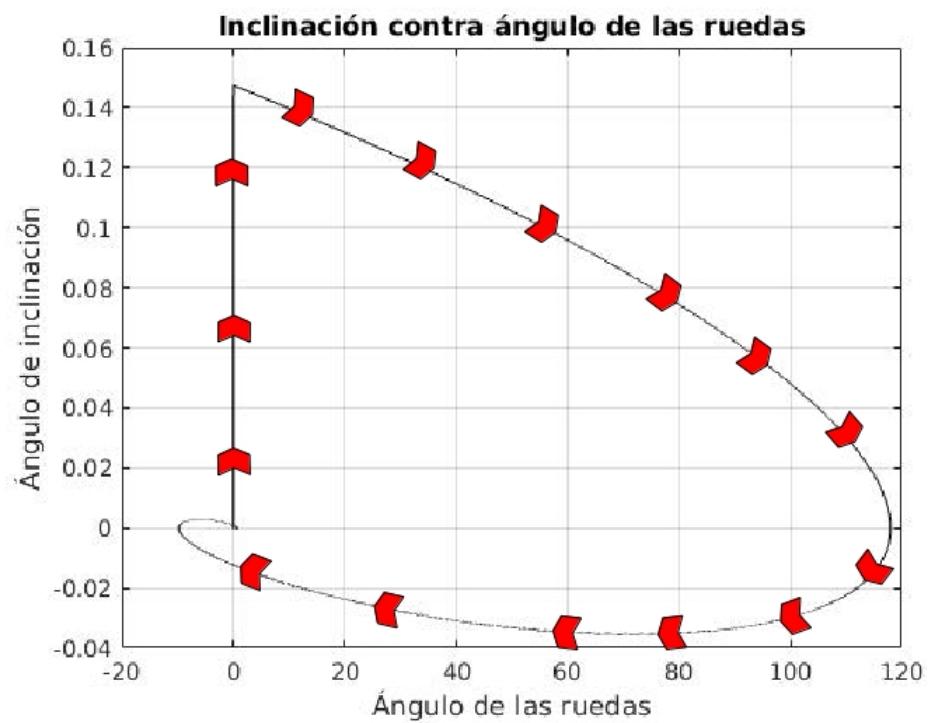


Figura 4.21: Ángulo de inclinación contra ángulo de las ruedas con una perturbación de 10 grados o 0.175 radianes.

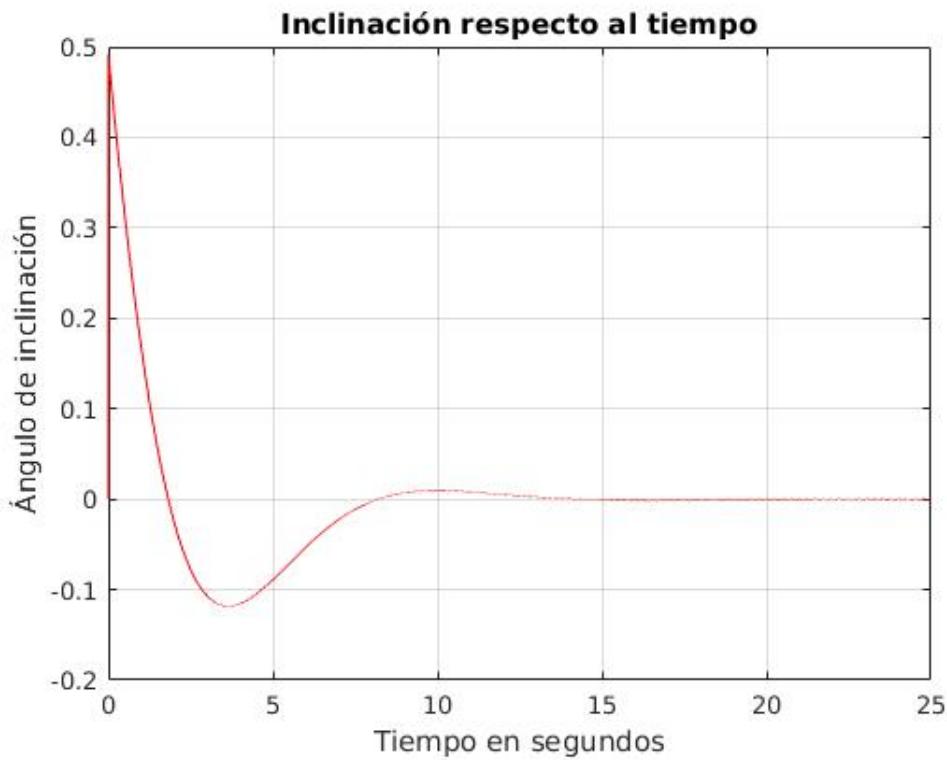


Figura 4.22: Ángulo de inclinación contra tiempo con una perturbación de 30 grados o 0.52 radianes.

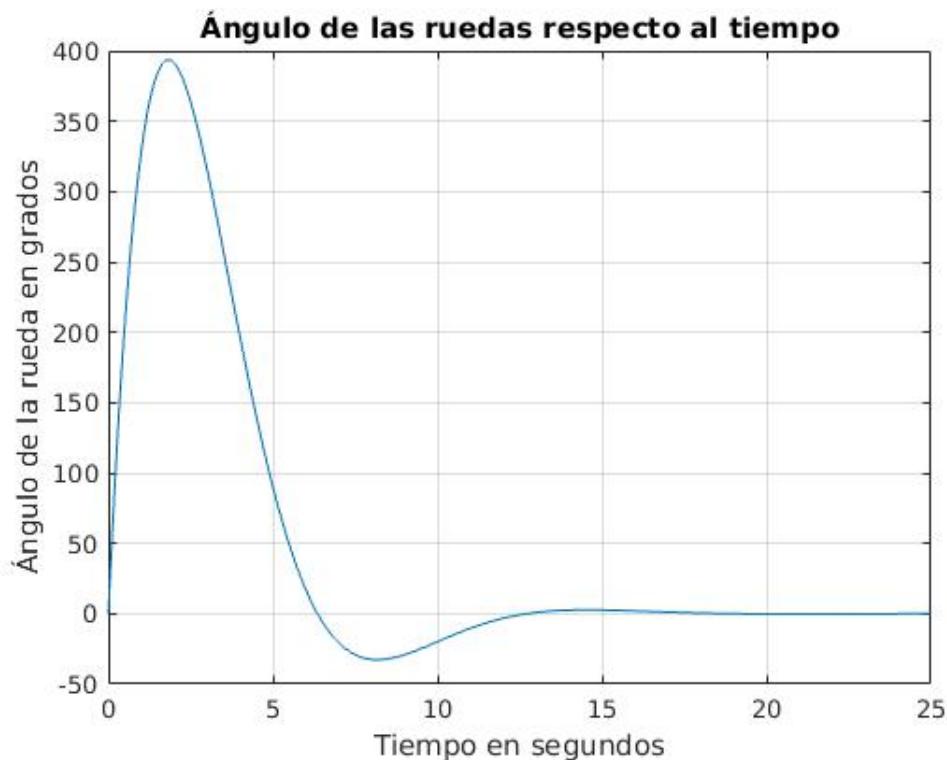


Figura 4.23: Ángulo de las ruedas contra tiempo con una perturbación de 30 grados.

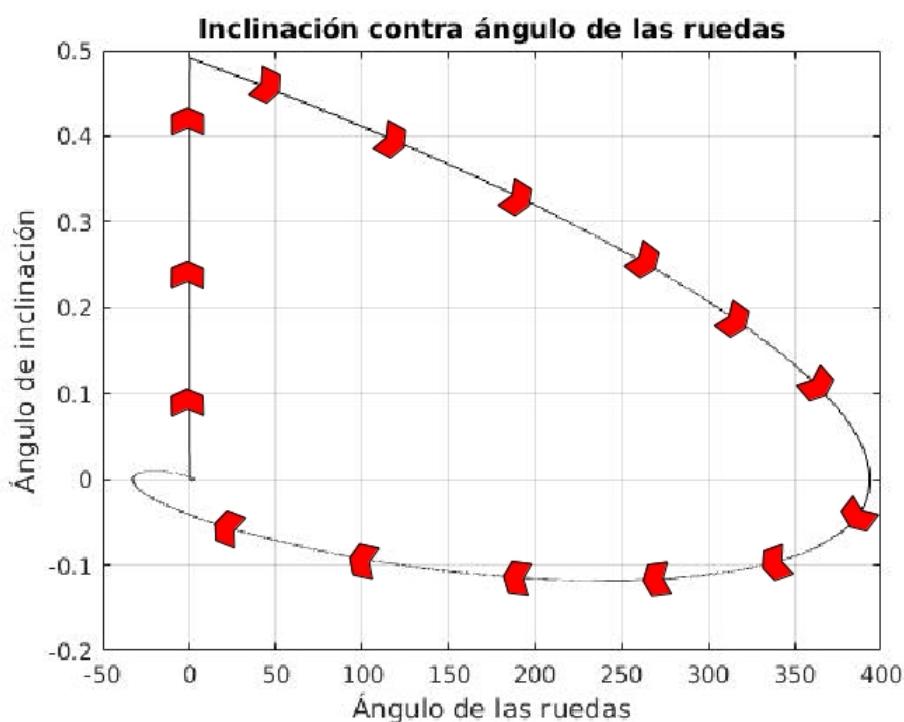


Figura 4.24: Ángulo de inclinación contra ángulo de las ruedas con una perturbación de 30 grados o 0.52 radianes.

4.4. Software

El prototipo fue programado en C++ utilizando ROS para que funcione en forma modular. Permitiendo que el prototipo pueda ser fácilmente modificado convirtiéndolo en una plataforma ideal para implementar diversas aplicaciones y probar métodos de control para sistemas no lineales.

4.4.1. Nodos de ROS

El ROS esta diseñado para facilitar el desarrollo en el área de la robótica permitiendo a los ingenieros compartir código entre diferentes prototipos ahorrando tiempo en la construcción.

El código es compartido en forma de nodos que se comunican mediante mensajes, en el caso del prototipo los nodos se conectan de la siguiente manera.

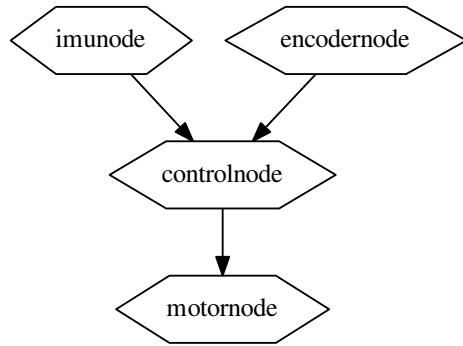


Figura 4.25: Diagrama de nodos de ROS.

A continuación hablaremos mas a detalle de como funciona cada uno de ellos.

imunode

El nodo imunode es el encargado de comunicarse con la IMU, una vez establecida la comunicación obtiene un promedio de todas las lecturas que requiere para calcular el ángulo, esto con el fin de obtener un *setpoint* con un valor cero, es necesario debido a la manera en que funcionan los MEMS y que al ser reiniciados pueden no regresar al valor cero.

Después de obtener el *setpoint* entra en un ciclo infinito que se repite 42 veces por segundo. En este ciclo toma las mediciones de la IMU, calcula la aceleración angular en el eje x y el ángulo del giroscopio también en el eje x.

Finalmente se hace una fusión de sensores utilizando el filtro complementario del que se habló previamente en este documento, se obtiene el ángulo de inclinación ψ de la estructura del prototipo. Este ángulo es enviado al controlnode.

encodernode

El nodo encodernode entra en un ciclo infinito el cual se repite 10700 veces por segundo, esta frecuencia esta ligada directamente con la velocidad de muestreo necesaria para no pasar por alto los pulsos enviados por el *encoder*.

Dentro del ciclo la primera acción del nodo es leer las señales generadas por los *encoders* y hacer una comparación con el estado anterior, con el fin de determinar si se realizó un desplazamiento y en que sentido.

El resultado de esta comparación es guardado en un acumulador. Y finalmente se toma el valor del acumulador, se multiplica por 0.1125 para convertirlo a grados y se envía al controlnode.

El nodo realiza este proceso de manera simultanea para ambos *encoders*.

motornode

El nodo motor node al igual que los anteriores entra en un ciclo infinito que se repite en este caso 1000 veces por segundo. El nodo recibe un mensaje del controlnode con cuatro dígitos, dos por cada motor, uno de estos dígitos indica el sentido y el otro la velocidad.

Estos valores son utilizados para generar las señales de salida que van a los *drivers*.

control2node

El control2node es el nodo central del prototipo, este nodo contiene el controlador del robot.

Este nodo tiene funciones que reciben de manera asíncrona los mensajes enviados por el imunode y el encodernode, en el caso del imunode guarda en angulo ψ en una variable global, calcula la derivada que también es guardada en una variable global.

Para el encodernode, la función que lee el mensaje obtiene primero el ángulo promedio θ y posteriormente lo guarda en una variable global, finalmente calcula el la derivada e integral las cuales también se guardan en variables globales.

Lo anterior se tiene el propósito de no retrasar las mediciones y realizar los cálculos de una manera mas rápida.

Después de inicializar estas funciones el nodo entra en un ciclo infinito en el cual se leen las variables globales y las multiplica por las correspondientes constantes de control. Para finalmente enviar la señal de control al motornode.

Capítulo 5

Presentación y Discusión de Resultados

Durante el desarrollo de este trabajo se realizaron varios cambios en el *hardware* y *software*. El modelo del control se cambió y así también el lenguaje en el que fue escrito el programa, la última versión y con la que se obtuvieron mejores resultados es la que se reporta en esta tesis. En este capítulo se recapitula brevemente los cambios mas significativos del prototipo así también se muestran las pruebas realizadas para obtener la calibración final utilizada en el robot y los resultados obtenidos.

5.0.1. Aproximaciones previas

El prototipo paso por cuatro etapas principales de pruebas, cada una de ellas utilizando diferentes aproximaciones.

En la primera etapa el robot no tenia cubierta, el software estaba hecho en Python para apresurar el desarrollo, el modelo era en base a un carro péndulo y no tomaba en consideración la medición de los *encoders*.

En la segunda etapa se hizo un cubierta para el prototipo mediante impreción 3D, se cambió el modelo a una versión basada en el robot JOE que contempla el avance y giro del robot.

La tercera etapa consistió en cambiar el lenguaje de programación a C++ para aumentar la velocidad de procesamiento, también se cambiaron los filtros de la IMU y se integró el PCB.

En la última etapa se cambió el modelo al utilizado actualmente, se integraron las mediciones de los *encoders*, se cambió también la cubierta por una mas rígida y finalmente se cambió la manera en que el nodo de control procesa las señales que recibe de los sensores.

5.0.2. Metodología

El método que se utilizó para probar los parámetros del controlador consistió en realizar una serie de pruebas ajustando los parámetros de manera empírica partiendo de los obtenidos mediante LQR, primero se buscó reducir el sobre impulso y las vibraciones en la estructura, posteriormente encontrar una serie valores en los cuales el prototipo se mantuviera estable y finalmente buscar un grupo de parámetros que mejoraran la robustez del modelo.

5.0.3. Experimentos

En esta sección se muestra una tabla que contiene los valores de las constantes del controlador utilizadas durante las pruebas de ajuste del mismo. Se denominan como estables las pruebas en las que el robót pudo mantener un ángulo vertical sin oscilaciones perceptibles.

A continuación se muestran los experimentos realizados.

#	K_i	K_{f1}	K_{f2}	K_{f3}	K_{f4}	Notas
1	0.7071	1.3933	-74.268	1.6141	-1.3836	Primer experimento
2	0.7071	1.3933	-74.268	1.6141	-0.065	
3	0.7071	1.3933	-74.268	1.6141	-0.26	
4	0.7071	1.3933	-39	1.6141	-0.14	
5	0.7071	1.3933	-39	1.6141	-0.28	
6	0.7071	1.3933	-39	1.6141	-0.56	
7	0.70	1.3933	-39	1.6141	-0.56	
8	0.0105	1.3933	-39	1.6141	-0.56	
9	0.007	1.3933	-39	3.2	-0.56	
10	0.007	1.3933	-39	0.8	-0.56	
11	0.007	0.65	-39	0.8	-0.56	
12	0.007	2.6	-39	0.8	-0.56	
13	0.007	4	-39	0.4	-0.56	
14	0.007	8	-39	0.4	-0.56	
15	0.007	8	-39	0.8	-0.56	
16	0.007	8	-80	0.8	-0.56	
17	0.007	8	-60	0.8	-0.56	
18	0.007	8	-60	0.8	-0.75	
19	0.007	8	-60	0.6	-0.75	
20	0.007	8	-60	0.4	-0.75	
21	0.007	8	-60	1	-1.5	
22	0.007	8	-60	0.8	-1.5	
23	0.007	8	-60	0.8	-3	
24	0.007	6	-60	0.8	-3	
25	0.007	3	-60	1	-3	
26	0.0007	3	-60	1	-3	
27	0.0007	5	-60	1	-5	
28	0.0007	5	-60	1	-15	
29	0.0007	5	-60	1	-17	
30	0.0007	5	-80	1	-15	
31	0.0007	5	-80	1	-10	
32	0.0007	5	-80	1	-3	
33	0.0007	5	-80	1	-5	
34	0.0007	5	-90	1	-3	

Cuadro 5.1: Experimentos realizados Pt1

#	K_i	K_{f1}	K_{f2}	K_{f3}	K_{f4}	Notas
35	0.0007	5	-80	1	-1	
36	0.0007	10	-60	1	-1	
37	0.0007	10	-50	1.6	-0.5	
38	0.0007	15	-40	1.6	-0.2	
39	0	15	-40	1	-0.8	
40	0	5	-40	0.5	-0.8	
41	0	2	-8	0.1	0	
42	0	2	-5	0.1	0	Estable
43	0	0.5	-15	0.01	0	
44	0	0.8	-8	0.001	0	
45	0	0.8	-8	0.005	-0.002	
46	0	1.5	-8	0.005	-0.002	
47	0	1.5	-8	0.005	-0.01	
48	0	2	-5	0.1	-0.01	Estable
49	0	2	-5	0	-0.5	Estable
50	0.001	2	-0	0.1	-0.01	
51	0.0005	0.1	-5	0.03	-0.05	Estable
52	0.06	0.1	-5	0.03	-0.05	
53	0.0005	0.1	-5	0.03	-0.02	Estable
54	0.0005	0.1	-15	0.03	-0.02	
55	0.0005	0.1	-4	0.03	-0.02	Estable
56	0.0005	0.1	-4	0.01	-0.02	Estable
57	0.0005	0.1	-4	0.01	-0.5	
58	0.0005	0.1	-4	0.15	-0.008	Estable
59	0.0005	0.1	-4	0.05	-0.008	Estable
60	0.0005	0.01	-4	0.005	-0.008	Estable
61	0.0005	0.03	-10	0.02	-0.08	
62	0	0.2	-5.5	0.005	0	Estable
63	0.0	1.5	-5.5	0.03	-0.005	Estable
64	0.0	0.8	-5.5	0.03	-0.005	Estable
65	0.0	0.8	-6	0.03	-0.005	Estable
66	0.0	0.8	-6	0.003	-0.005	Estable
67	0.0	0.8	-10	0.003	-0.005	
68	0.0	0.06	-10	0.003	-0.005	
69	0.0	2	-10	0.003	-0.005	
70	0.0	0.4	-10	0.003	-0.005	
71	0.0	0.2	-10	0.003	-0.005	
72	0.0	0.1	-10	0.03	-0.06	
73	0.0	1.39	-7	0.1	-0.2	
74	0.0	1.39	-6.5	0.1	-0.2	Estable
75	0.0	1.1	-6	0.05	-0.03	Estable
76	0.0	1.5	-5.5	0.05	-0.03	Estable

Cuadro 5.2: Experimentos realizados Pt2

Condiciones de prueba

El prototipo fue probado en un ambiente cerrado, se coloco en posición vertical de tal manera que la computadora pudiera obtener el ángulo cero y posteriormente se observó si podía mantener la vertical por si mismo, en caso de que lo lograra se consideró que era estable. Finalmente se introdujeron perturbaciones en el sistema inclinando el robot con la mano y se observó como reaccionaba.

5.0.4. Resultados

En la tabla de la sección anterior se observan los experimentos realizados con diferentes valores en las constantes del controlador, siendo la mas estable la prueba número 63.

De las pruebas se obtuvo que las constantes que controlan el ángulo θ ayudan a disminuir el sobre impulso, pero a la vez limitan la reacción del robot ante perturbaciones grandes, se trató de probar con constantes proporcionales K_{f2} mas grandes, para tratar de compensar este efecto pero causó histéresis en el sistema de control.

También se observó que los valores obtenidos por el método LQR varían en gran medida con los encontrados experimentalmente, especialmente en las constantes que involucran el ángulo ψ . Esto se atribuye a que los sensores trabajan de manera asíncrona y a que los filtros aplicados en la IMU reducen en gran medida la velocidad de muestreo.

Capítulo 6

Conclusiones, recomendaciones y trabajo futuro

6.1. Conclusiones

En este trabajo se realizó el C.A.D. de un prototipo de un robot de estructura vertical que esta planeado para realizar tareas en ambientes humanos desordenados; que según el articulo [1] consiste en entornos donde no existen referencias planeadas para que el robot pueda navegar. El peso total de este prototipo es de 4.6 kg y soporta una carga en la parte superior de hasta 0.5 kg manteniendo óptimo su funcionamiento. Mantener el diseño sencillo y utilizar como columna un IPS de aluminio de medidas estandarizadas reduce significativamente el costo en comparación a utilizar partes manufaturadas a medida.

Para el robot se realizó el diseño y construcción de un sistema electrónico autocontenido que permite el funcionamiento del robot sin necesidad de conexiones externas. El sistema se planeo tomando en cuenta el aprovechamiento de las baterías. Cuenta con dos fuentes de alimentación aisladas con el fin de proteger a la computadora de las variaciones en la corriente causadas por los motores, a su vez esta configuración evita que se pueda introducir ruido en los sensores a través de su línea de alimentación y permite alimentar los actuadores con un nivel de potencia mayor al de la computadora.

Se describió de manera matemática el modelo dinámico del prototipo robótico. Esta descripción permite entender de una manera general el comportamiento del robot.

Se diseño un controlador que es capaz de mantener en posición vertical la estructura del prototipo robótico pese a pequeñas perturbaciones externas en el rango de los 4 grados de inclinación.

Este trabajo comprobó que el prototipo puede mantener la vertical de manera estable ante pequeñas perturbaciones, que los motores y la computadora tienen la capacidad necesaria para llevar a cabo el control de un sistema dinámicamente estable a pesar que el ROS disminuye el tiempo de procesamiento, quedando como precedente para futuros trabajos que involucren sistemas embebidos como la Raspberry Pi trabajando con ROS.

6.2. Recomendaciones

A continuación se encuentra una lista de recomendaciones permitirían un mejor desempeño del prototipo como una plataforma de desarrollo.

- Añadir una tarjeta de desarrollo Arduino [18].
- Cambiar la computadora por una Raspberry pi 2 b+ por una Raspberry pi 3 u otra mas reciente.
- Cambiar la implementación del PWM.
- Fabricación de un PCB con mascara anti-soldante y elementos de montaje. superficial.
- Cambio en el cableado.
- Incluir un circuito que permita monitorear el voltaje de las baterías.
- Utilizar un sistema de control difuso.

Estas mejoras se sugieren por las siguientes razones.

El añadir una tarjeta de desarrollo Arduino nos permite tomar muestras a mas alta velocidad que los puertos GPIO de la Raspberry Pi y también tiene mas puertos con la capacidad de generar señales PWM lo que permite eliminar el PWM por software que utiliza el prototipo actualmente, esto evita que se dañen los puertos GPIO de la computadora y reduce la cantidad de procesos que requiere manejar permitiendo que esta pueda enfocarse en tareas mas complejas como el control.

Cuando se comenzó la construcción del prototipo se seleccióno la Raspberry Pi 2 B+ como computadora debido a sus capacidades y al software disponible para esta, pero en el lapso en que se construyó el prototipo surgieron opciones con mejores cualidades como la Raspberry Pi 3 que cuenta con módulos de wifi y bluetooth integrados o la ODROID-XU4 [19] que tiene una capacidad de cómputo mucho mayor en un espacio similar.

El PCB diseñado para el prototipo se pensó para que las conexiones a los puertos fueran fáciles de modificar. Pero en este momento se cuenta con un arquitectura definida, por lo que sería conveniente cambiar el PCB por un diseño que utilice cable plano en lugar de *headers* y que tenga la capa de esmalte anti-soldante para protegerla del oxido.

El incluir un circuito que le permita a la computadora monitorear la carga de las baterías proporciona varias ventajas. En primer lugar permite saber cuando es momento de cargar las baterías, además que el voltaje de la batería que alimenta los motores determina cual es la velocidad máxima de los mismos, este factor puede incluirse en el modelo de control para mejorarlo.

Finalmente sería interesante cambiar el controlador clásico por uno difuso o neuro-difuso para aumentar la robustez del prototipo en cuanto a mantenerse en un punto estable, esto debido al resultado obtenido en trabajos anteriores utilizando estos métodos.

6.3. Trabajo futuro

Con el fin de mejorar el desempeño del prototipo como plataforma de desarrollo para aplicaciones en ambientes humanos complejos es deseable que cuente con la capacidad de navegar de manera

autonoma, reconocer objetos y acciones, y poder interactuar de una manera mas intuitiva con el usuario.

Con lo anterior en mente se sugieren las siguientes mejoras al prototipo.

- Implementar sensores de proximidad o un LIDAR.
- Implementar algoritmos de visión por computadora.
- Implementar un sistema de reconocimiento de voz.
- Utilizar el prototipo como un asistente controlado por voz.

El implementar un sistema que le permita al robot obtener información de su entorno y procesarla para tomar decisiones puede aportarle un sin fin de aplicaciones al prototipo, como la de mesero o mensajero, este sistema debe de contar con la capacidad de detectar obstáculos por lo que se sugiere utilizar sensores de proximidad o si es posible un LIDAR, el reconocer personas, objetos y puntos de referencia. También serían clave para que intereactue con su entorno. Esto se puede conseguir con algoritmos de visión por computadora y *machine learning*.

Y por último una de las aplicaciones finales para las que se pensó este robot es la de mensajero pero también podría ser utilizado como mesero o asistente en alguna tarea específica, para lo que sería muy útil implementar algoritmos de reconocimiento de voz para esta clase de tarea.

Si en un futuro se planea utilizar el robot para llevar cargas mas mayores a las reportadas en este trabajo, se requerirá de un nuevo diseño de controlador así como actuadores con un mayor torque.

Apéndice A

Programas

A.1. Modelo en espacio de estados

```
g=9.81;
m=0.18;
R=0.0625;
j_w=(m*R^2)/2;
M=4.14;
W=0.6;
D=0.2;
H=1;
L=0.0386;
j_psi=(M*L^3)/3;
j_phi=(M*(W^2+D^2))/12;
j_m=1*10^(-5);
R_m=2.4;
K_b=0.57295;
K_t=0.24;
n=1;
f_m=0.0022;
f_w=0;
alpha=(n*K_t)/R_m;
beta=(n*K_t*K_b)/R_m+f_m;
A_1=zeros(4,4);
A_2=zeros(2,2);
B_1=zeros(4,2);
B_2=zeros(2,2);
D_1=zeros(4,2);
D_2=zeros(2,2);
C_1=eye(4);
C_2=eye(2);
E=zeros(2,2);
E(1,1)=((2*m+M)*R^2)+(2*j_w)+(2*n^2*j_m);
E(1,2)=(M*L*R)-(2*n^2*j_m);
E(2,1)=E(1,2);
E(2,2)=(M*L^2)+j_psi+(2*n^2*j_m);
dete=det(E);
A_1(3,2)=-(g*m*L*E(1,2))/dete;
A_1(4,2)=(g*M*L*E(1,1))/dete;
A_1(3,3)=(-2*((beta+f_w)*E(2,2)+beta*E(1,2)))/dete;
A_1(4,3)=(2*((beta+f_w)*E(1,2)+beta*E(1,1)))/dete;
A_1(3,4)=(2*beta*(E(2,2)+E(1,2)))/dete;
A_1(4,4)=(-2*beta*(E(1,1)+E(1,2)))/dete;
```

```

A_1(1,3)=1;
A_1(2,4)=1;
B_1(3,1)=(alpha*(E(2,2)+E(1,2)))/dete ;
B_1(3,2)=B_1(3,1);
B_1(4,1)=(-alpha*(E(1,1)+E(1,2)))/dete ;
B_1(4,2)=B_1(4,1);
I=(1/2)*m*W^2+j_phi+((W^2)/(2*R^2))*(j_w+n^2*j_m) ;
J=((W^2)/(2*R^2))*(beta+f_w) ;
K=((W)/(2*R))*alpha ;
A_2(1,2)=1;
A_2(2,2)=-J/I;
B_2(2,1)=-K/I;
B_2(2,2)=-B_2(2,1);
[num, den]=ss2tf(A_1, B_1, C_1, D_1, 2);
%%
Co = ctrb(A_1, B_1);
unco = length(A_1) - rank(Co);
Ob = obsv(A_1, C_1);
unob = length(A_1)-rank(Ob);
%%
Co2 = ctrb(A_2, B_2);
unco2 = length(A_2) - rank(Co2);
Ob2 = obsv(A_2, C_2);
unob2 = length(A_2)-rank(Ob2);
%
Q_1=[1,0,0,0,0
      0,1e6,0,0,0
      0,0,1,0,0
      0,0,0,1,0
      0,0,0,0,1e-8];
R_1=1e2*eye(2);
A_bar=[A_1, zeros(4,1); C_1(1,:), 0];
B_bar=[B_1; 0, 0];
K4222=lqr(A_bar, B_bar, Q_1, R_1);

```

A.2. Nodo de Control (control2node)

```

/*
 * control2node.cpp
 *
 * Copyright 2017 <Jose Angel Martinez Navarro>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.

```

```

/*
*/
#include "ros/ros.h"                                //libreria de ROS
#include "std_msgs/String.h"                         //leer y enviar mensajes String en ROS
#include <iostream>
#include <sstream>
#include <math.h>
//----- variables globales
float val[3]={0.0,0.0,0.0};                      //entradas
int sal[4]={0,0,0,0};                             //salidas
float ki, kf[4], angpro;                          //constantes y promedio del
                                                 //angulo
float derpsi, dertheta, psiac[2]={0.0,0.0}, thetac[2]={0.0,0.0}, integ=0.0,u; //acumuladore,
                                                 //integrales y salida

//----- funcion que recibe los datos de la IMU
void imudata(const std_msgs::String::ConstPtr& msg)
{
    std::stringstream ss;
    ss<<msg->data.c_str();
    ss>>val[0];
    psiac[0]=val[0];                                 //derivadas sucias
    derpsi=(psiac[0]-psiac[1])/0.047;
    psiac[1]=psiac[0];
}
//----- funcion que recibe los datos de los Encoders ////
// 2=un robot debe obedecer las ordenes dadas por los seres humanos, excepto si estas
// ordenes entran en conflicto con la primera ley
void encoderdata(const std_msgs::String::ConstPtr& msg){
    std::stringstream sa;
    sa<<msg->data.c_str();
    sa>>val[1];
    sa>>val[2];
    angpro=(val[1]+val[2])/2; //obtiene el promedio de los encoders
    thetac[0]=angpro;
    dertheta=(thetac[0]-thetac[1])/0.01;
    thetac[1]=thetac[0];
    integ=integ+angpro;   //integral discreta
}
//----- funcion donde se realiza el algoritmo de
// control
void control()
{
    u=ki*(-integ)+kf[0]*(-angpro)+kf[1]*(-val[0])+kf[2]*(-dertheta)+kf[3]*(-derpsi); // calculo del SERVOCONTROL (PD-I)
    if (u>0)                                            //
        direccion de las ruedas
    {
        sal[0]=1;
        sal[2]=1;
    }
    else
    {
        sal[0]=0;
        sal[2]=0;
        u=-u;
    }
}

```

```

}
if(u>100)           //tope
{
    u=100;
}
sal[1]=u;
sal[3]=u;
}
//—————funcion principal
int main(int argc, char **argv)
{
    ros::init(argc, argv, "controllnode"); //inicializa el nodo
//—————constantes de control
ki=-0.0;
kf[0]=1.5;
kf[1]=-5.5; //negativo
kf[2]=0.03;
kf[3]=-0.005;
ros::NodeHandle n;
ros::Subscriber sub = n.subscribe("imumsg", 10, imudata); //se declaran los
                datos de entrada y se inicializan los hilos
ros::Subscriber sub2 = n.subscribe("wheelpos", 1000, encoderdata);
ros::Publisher control_pub=n.advertise<std_msgs::String>("controlmv", 1000); ////
                se declara el tipo de mensaje
ros::Rate loop_rate(50); //la cantidad de veces que se repite por segundo
while(ros::ok()){
    std::stringstream sb;
    control(); // el control
    sb << sal[0] << " " << sal[1] << " " << sal[2]<< " " << sal[3];
    std_msgs::String mss;
    mss.data=sb.str();
    control_pub.publish(mss); //la funcion que publica en ROS
    ros::spinOnce(); //revisa los hilos
    loop_rate.sleep();
}
return 0;
}

```

A.3. Nodo que lee la IMU (imunode)

```

//la IMU es una MPU 6050
// Librerias

#define <wiringPiI2C.h> //i2c
#define <wiringPi.h>//puertos
#define "ros/ros.h"
#define "std_msgs/String.h"
#define <iostream>
#define <sstream>
#define <math.h>

#define MPU6050_ADDRESS (0x68) //Direccion del dispositivo
#define MPU6050_CLOCK (0x6b) //Reloj interno 8MHz
#define MPU6050_PASABAJAS (0x1a) //Filtro pasa bajas
#define MPU6050_ACCELEROMETRO (0x1c) //Rango del acelerometro

```

```

//funciones
float x_pos(float xa, float gx, float x);
float setoffAcell(int fd);
float setoffGyro(int fd);
float x_accel(int ax, int ay, int az); //0=un robot no puede hacer dano a la humanidad o,
                                         por inaccion, permitir que la humanidad sufra dano
int readword(int fd,int dir);
//programa principal
int main(int argc, char **argv)
{
    ros::init(argc, argv, "imunode"); //inicializa el nodo
    ros::NodeHandle n;
    int fd;
    float ante=0.0;
    float setg ,seta ,gx ,accel ,ang ,sal ;
    int ax ,ay ,az ,i ;
    if (wiringPiI2CSetup(0x68) == -1)
    {
        printf("Error _X.x");
        return 0;
    }
    fd=wiringPiI2CSetup(MPU6050_ADDRESS); //comunicacion i2c
    wiringPiI2CWriteReg8(fd ,MPU6050_CLOCK,0); //reloj interno 8MHz
    wiringPiI2CWriteReg8(fd ,MPU6050_ACCELEROMETRO,4); //Rango del acelerometro +-4g
    wiringPiI2CWriteReg8(fd ,MPU6050_PASABAJAS,4); //filtro pasa bajas accl y gyro
    setg=setoffGyro(fd); //obtiene el 0 en el gyroskopio
    seta=setoffAcell(fd); //obtiene el 0 en el acelerometro
    ros::Publisher imumsg_pub=n.advertise<std_msgs::String>("imumsg" , 1);
    ros::Rate loop_rate(47); // veces que se ejecuta por segundo
    while(ros::ok())
    {
        gx=(readword(fd ,0x43)/131-setg); //lee el gyroskopio
        ax=readword(fd ,0x3b); //lee el accel en x
        ay=readword(fd ,0x3d); //lee el accel en y
        az=readword(fd ,0x3f); //lee el accel en z
        accel=x_accel(ax,ay,az)-seta; //obtiene la velocidad angular en x
        ang=x_pos(ante,gx,accel); //obtiene la posicion en x
        loop_rate.sleep();
        ante=ang;
        sal=ang;
        sal=((float)((int)(sal*10)))/10;
        std_msgs::String msg;
        if(1.5>sal&&sal>-1.5)
        {
            sal=0.0;
        }
        std::stringstream ss;
        ss << sal;
        msg.data=ss.str();
        imumsg_pub.publish(msg); //publica los mensajes
        ros::spinOnce();
    }

    return 0;
}
//lee la imu
int readword(int fd, int dir)
{
    int msb, lsb ,sal ;

```

```

msb=wiringPiI2CReadReg8(fd , dir );
lsb=wiringPiI2CReadReg8(fd , dir+1);
sal= msb << 8 | lsb ;
if( sal>0x8000){
    sal=sal-0x10000;
}
return sal;
}
//obtiene la velocidad angular en x
float x_accel(int x, int y , int z)
{
    int m;
    if(y*y+z*z==0)
    {
        m=0.0000001;
    }
    else{
        m=y*y+z*z ;
    }
    if(x/sqrt(m)>1)
    {
        return 0.0;
    }
    else if(x/sqrt(m)<-1)
    {
        return 180.0;
    }
    else
    {
        return (180/M.PI)*acos(x/sqrt(m));
    }
}

//obtiene el cero en el acelerometro
float setoffAcell(int fd)
{
    float Xprom=0.0;
    float x,y,z;
    for(int i=0; i<100; i++)
    {
        x=readword(fd ,0x3b);
        y=readword(fd ,0x3d);
        z=readword(fd ,0x3f);
        Xprom=Xprom+x_accel(x,y,z);
        delay(5);
    }
    return (Xprom/100);
}
//obtiene el cero en el gyroskopio
float setoffGyro(int fd)
{
    float Gprom=0.0;
    float gx;
    for(int i=0; i<100; i++)
    {
        gx=(readword(fd ,0x43)/131);
        Gprom=Gprom+gx;
        delay(5);
    }
}

```

```

        }
        return (Gprom/100);
    }

//obtiene el angulo en x
float x-pos(float xa, float gx, float x)
{
    return (0.9*(xa+gx*0.021)+0.1*x);//filtro complementario
}

```

A.4. Nodo que genera la señal para los motores (motornode)

```

/*
 * motornode.cpp
 *
 * Copyright 2017 <Jose Angel Martinez Navarro>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 *
 */
// Librerias

#include <iostream>
#include <sstream>
#include <wiringPi.h>
#include <softPwm.h>
#include <unistd.h>
#include <stdio.h>
#include "ros/ros.h"
#include "std_msgs/String.h"

int val[4]; //variables globales
int i;
std::stringstream ss;
int a1=12,b1=16,a2=27,b2=22,pwm1=26,pwm2=25; //pines

//funcion que genera los pwm y mensajes
void genPwmPos(const std_msgs::String ::ConstPtr& msg)
{
    //2=un robot debe proteger su propia existencia en la
    medida que esta proteccion no entre en conflicto con la 1ra o la 2da Ley.
    ss<<msg->data.c_str(); // lee los datos
}

```

```

for( i=0;i<4;i++)
{
    ss>>val[ i ];
}
if( val[0]==1) //obtiene la direccion
{
    digitalWrite(a1,1);
    digitalWrite(b1,0);
}
else{
    digitalWrite(a1,0);
    digitalWrite(b1,1);
}
if( val[2]==1)
{
    digitalWrite(a2,0);
    digitalWrite(b2,1);
}
else{
    digitalWrite(a2,1);
    digitalWrite(b2,0);
}
softPwmWrite(pwm1, val[ 1 ]); //genera el pwm
softPwmWrite(pwm2, val[ 3 ]);
ss . clear();
}

int main(int argc , char **argv)
{
    wiringPiSetupGpio(); //se declaran los nodos
    pinMode(a1,OUTPUT);
    pinMode(b1,OUTPUT);
    pinMode(a2,OUTPUT);
    pinMode(b2,OUTPUT);
    softPwmCreate(pwm1,0,100);
    softPwmCreate(pwm2,0,100);
    ros::init(argc , argv , "motornode"); //inicializa el nodo
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("controlmv",1000,genPwmPos); //lee el mensaje
    ros::spin();
    return 0;
}

```

A.5. Nodo que procesa la señal de los encoders(encodernode)

```

/*
 * encodernode.cpp
 *
 * Copyright 2017 <Jose Angel Martinez Navarro>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,

```

```

* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
* MA 02110-1301, USA.
*
*/
//Librerias

#include <wiringPi.h>
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <iostream>
#include <sstream>
//funcion principal
int main(int argc, char **argv)
{
    ros::init(argc, argv, "encodernode");//inicializa el nodo
    ros::NodeHandle n;
    int contador = 0;//contadores
    int contador2=0;
    int clk = 13;//pines
    int clk2=5;
    int dt=19;
    int dt2=6;
    wiringPiSetupGpio();//inicia wiring pi //1=un robot no puede hacer dano a un ser
    //humano o, por inaccion, permitir que un humano sufra dano
    pinMode(clk,INPUT);
    pinMode(dt,INPUT);
    pinMode(clk2,INPUT);
    pinMode(dt2,INPUT);
    bool clkLastState=digitalRead(clk);
    bool clkLastState2=digitalRead(clk2);
    bool dtState;
    bool dtState2;
    bool clkState;
    bool clkState2;
    ros::Publisher wheelpos_pub=n.advertise<std_msgs::String>("wheelpos", 1000);//mensaje
    ros::Rate loop_rate(15000); //veces que se ejecuta por segundo
    while(ros::ok())
    {
        for (int i=0; i<30; i++){ //contador de la rueda 2
            clkState=digitalRead(clk);
            dtState=digitalRead(dt);
            if (clkState!=clkLastState){
                if(dtState!=clkState){
                    contador++;
                }
            }
            else{
                contador--;
            }
        }
        clkLastState=clkState;
    }
}

```

```
//_____
clkState2=digitalRead(clk2); //contador de la rueda 1
dtState2=digitalRead(dt2);
if (clkState2!=clkLastState2){
    if(dtState2!=clkState2){
        contador2++;
    }
    else{
        contador2--;
    }
}
clkLastState2=clkState2;
loop_rate.sleep();
}

std_msgs::String msg;
std::stringstream ss;
ss << contador2*0.225 << "°" << contador*0.225; //conversion a grados y
publicacion
msg.data=ss.str();
wheelpos_pub.publish(msg);
ros::spinOnce();
}

return 0;
}
```

Apéndice B

Diagramas

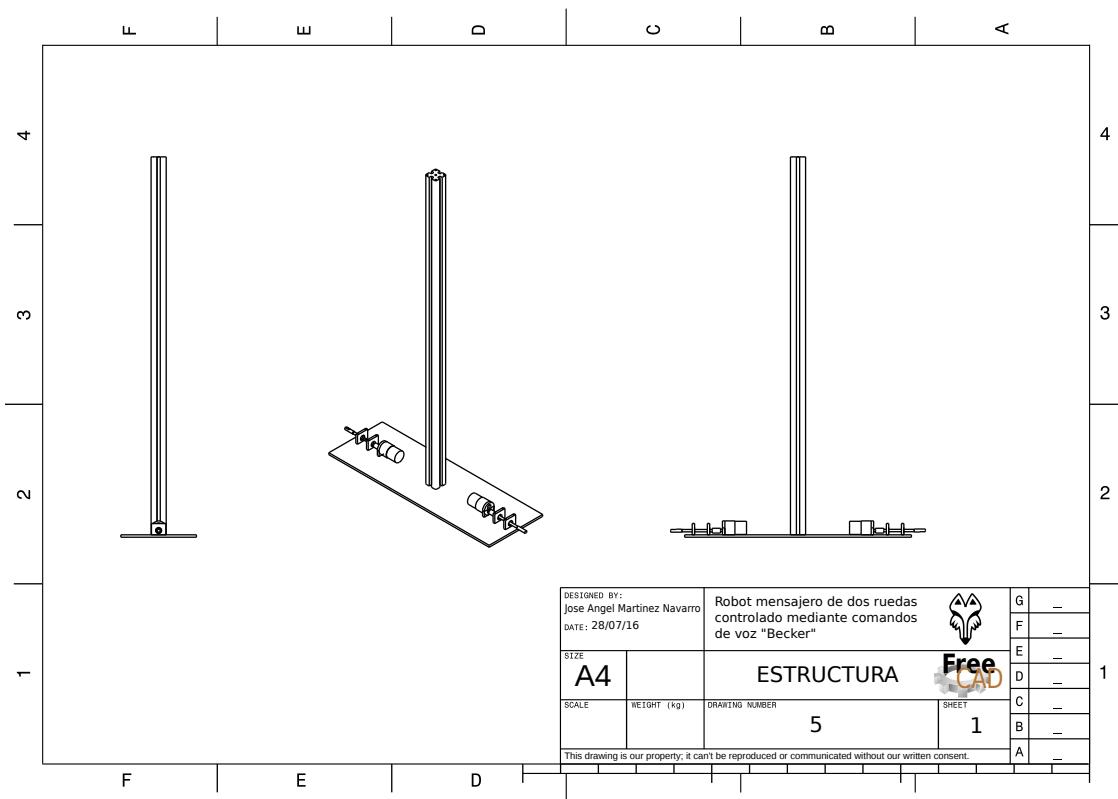


Figura B.1: Plano de la estructura.

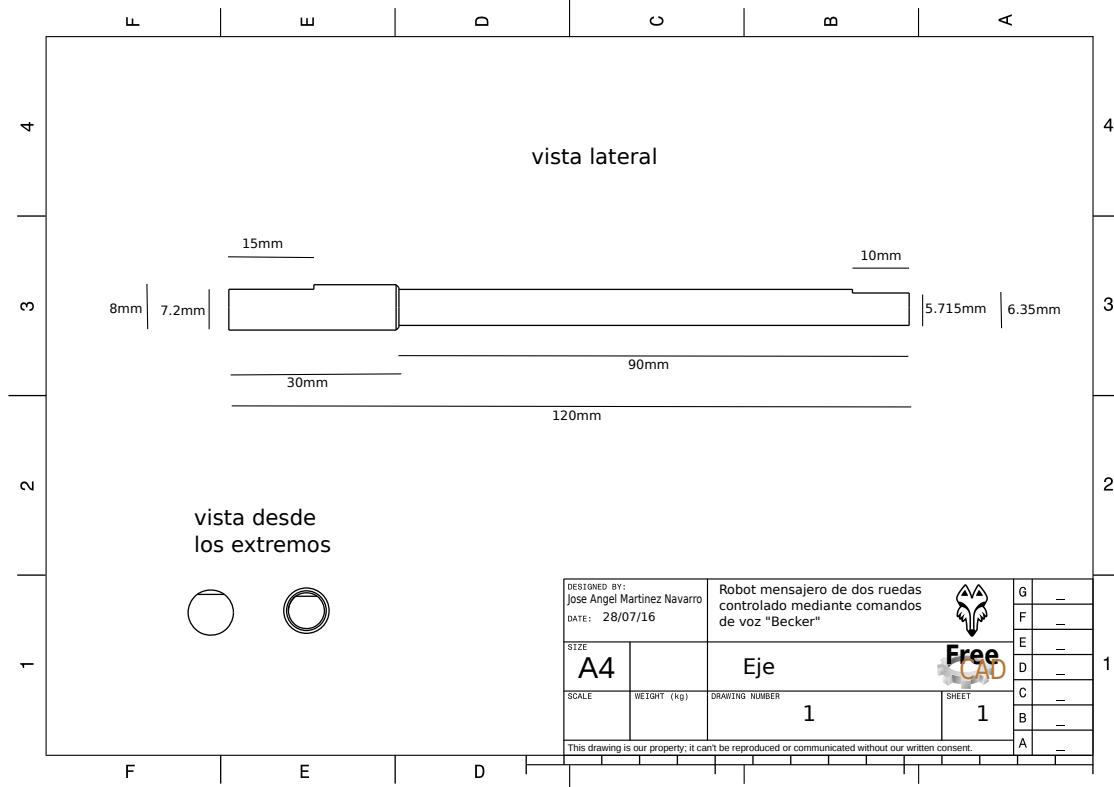


Figura B.2: Plano del eje.

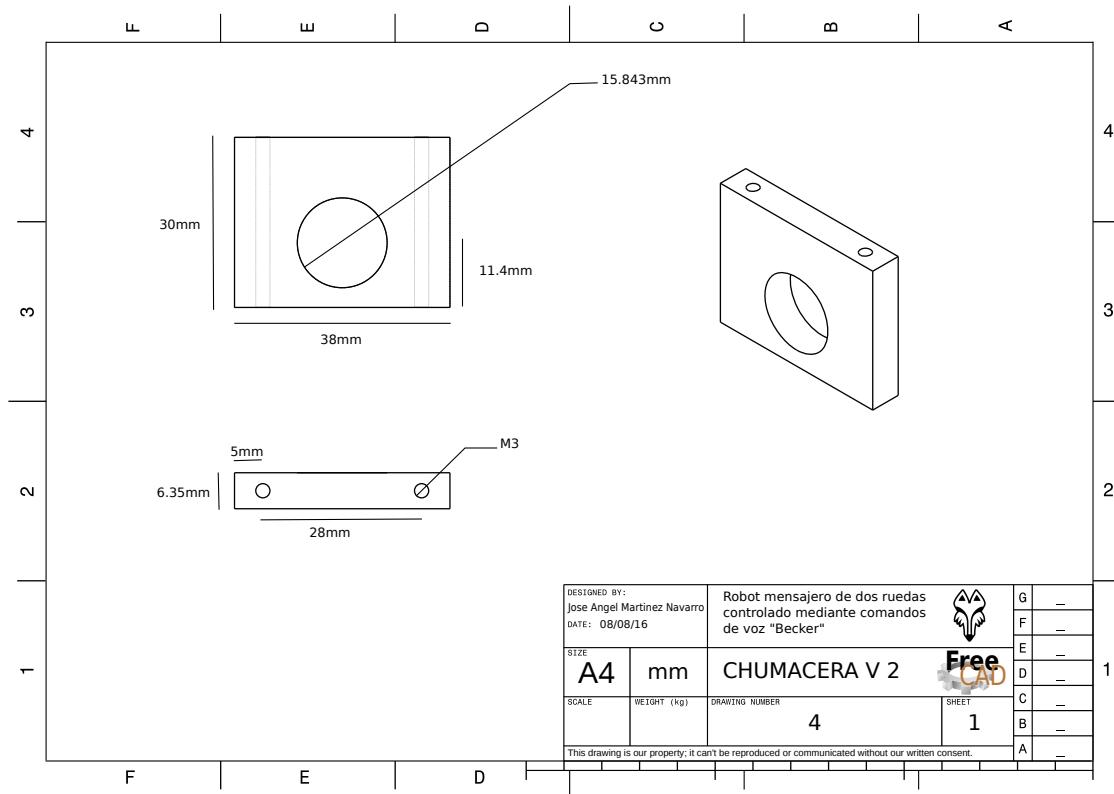


Figura B.3: Plano de la chumacera.

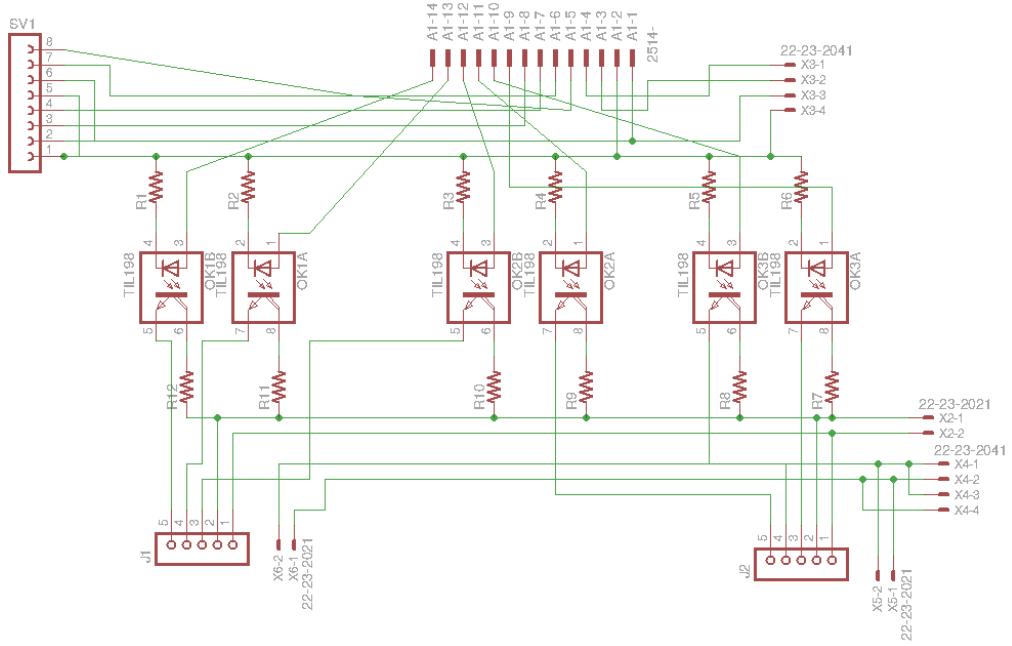


Figura B.4: Esquema del PCB.

Bibliografía

- [1] Umashankar Nagarajan and George Kantorand Ralph Hollis. The ballbot: An omnidirectional balancing mobile robot. *The International Journal of Robotics Research*, 33(6):917–930, may 2014.
- [2] Felix Grasser, Aldo D’ Arrigo, Silvio Colombi, and Alfred C. Rufer. Joe: A mobile, inverted pendulum. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 49(1), February 2002.
- [3] <http://www.segway.com/>.
- [4] Ing. Mauricio Ontiveros Rodriguez. Construcción y puesta en operación de un robot móvil basado en el principio del péndulo invertido. Master’s thesis, Centro de Investigación en Computación del Instituto Politécnico Nacional, 2014.
- [5] Yorihisa Yamamoto. Nxtway-gs model-based design - control of self-balancing two-wheeled robot built with lego mindstorms nxt -. *Applied Systems First Division CYBERNET SYSTEMS CO., LTD*, 2009.
- [6] Katsuhiko Ogata. *Ingeniería de control moderna*. PEARSON, 5 edition, 2010.
- [7] <http://www.i2c-bus.org/>.
- [8] <https://www.pololu.com/product/2824/specs>.
- [9] <https://www.pololu.com/product/1451/specs>.
- [10] Raspberry pi faq. <https://www.raspberrypi.org/help/faqs>.
- [11] Open Source Robotics Foundation. About ros. <http://www.ros.org/about-ros/>.
- [12] <http://profilesdealuminioranurado.com.mx/Fichas>
- [13] Ricardo J. Aguasca Colomo, José Cabrera Pe na, and Ibán Vega Ramírez. *Introducción a las fuentes de tensión conmutadas. Teoría y Práctica*. Universidad de las Palmas de Gran Canaria, 2001.
- [14] InvenSense Inc. Mpu-6000 and mpu-6050 product specification revision 3.4. Technical report, InvenSense Inc., 2013.
- [15] InvenSense Inc. Mpu-6000 and mpu-6050 register map and descriptions revision 4.2. Technical report, InvenSense Inc., 2013.
- [16] Ing. Roberto Vela Pe na. Sistema de detección de movimientos basado en sensores iniciales integrados. Master’s thesis, Centro de Investigación e Innovación Tecnológica, 2013.
- [17] Eric W Weisstein. Numerical differentiation.
<http://mathworld.wolfram.com/numericaldifferentiation.html>.

- [18] [https://www.arduino.cc/.](https://www.arduino.cc/)
- [19] [http://www.hardkernel.com/main.](http://www.hardkernel.com/main)