UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

EQUILIBRIO DINÁMICO EN TIEMPO REAL DE UN ROBOT TIPO BALLBOT

Que para obtener el título de
INGENIERO MECATRÓNICO

presentan

Gutiérrez Gaspar José Humberto y
Espinosa Mendoza José Luis

Director de Tesis M. I. Yukihiro Minami Koyama

Ciudad Universitaria
Febrero 2014

Quiero agradecer:

A mis padres, por confiar en mí y brindarme su apoyo y amor incondicional

A mi hermano, por todas las veces que me brindó su apoyo y consejo

A Sazy, por estar siempre en el momento correcto y decir las palabras adecuadas

A mis profesores y amigos, por brindarme su tiempo y su amistad, por su paciencia, tolerancia y ayuda

A mi universidad, la UNAM, por darme la oportunidad de crecer como persona

A mi facultad, por enseñarme la más linda de las profesiones: la ingeniería

Sin lugar a duda, este objetivo no se hubiera podido cumplir sin ustedes.

¡Gracias!

Por mi raza hablará el espíritu

Humberto.

Le quiero agradecer el apoyo incondicional a mi apá y a mi amá durante mi crecimiento como estudiante y como persona. A Nan por aguantar tantas cosas y seguir a mi lado. A los fénixes por tantas cosas que hemos vivido en estos años. A todos los profesores que me enseñaron cosas que se aplican afuera del aula, especialmente a Yuki. A todo el personal administrativo y docente de esta y de otras facultades que me ayudaron a salir adelante con la carrera. Y finalmente al Barto, que sin él la tesis hubiera sido más fácil.

Atte: José Luis Espinosa aka. Pepe

Investigación realizada gracias al Programa UNAMDGAPA-PAPIME PE104212 “Mejoramiento de la calidad educativa en Ciencias Básicas a través de la Robótica”.

ÍNDICE DE CONTENIDO

PREFACIO

El desarrollo de proyectos es una tarea compleja que requiere de mucha investigación y de un entendimiento claro y conciso de los problemas que se busca resolver, en particular lo relacionado con el desarrollo de prototipos robóticos, por lo que fue primordial utilizar toda la información al alcance para la realización de este proyecto. Uno de los recursos que más se encuentran a la mano es la asesoría de profesores de la Facultad de Ingeniería, ya sean de nivel licenciatura o de posgrado, pues proporcionan información y puntos de vista muy valiosos gracias a su experiencia y conocimientos en diversas áreas. Otro recurso de igual importancia son los trabajos anteriores que se han realizado en esta y otras universidades alrededor del mundo, muchos de los cuales se encuentran ampliamente documentados y son asequibles mediante Internet o consultando a las personas involucradas en estos trabajos.

Particularmente este proyecto utilizó como antecedente principal la tesis *Diseño, construcción y control de estabilidad de un robot que se balancea sobre una esfera* realizada en el año 2012 [1], esto con la finalidad de evitar recorrer dos veces el mismo camino. A lo largo de este proyecto se hará referencia al producto de la tesis antes mencionada como *primer prototipo*.

Se analizó la tesis anterior para poder generar un plan de trabajo que tuvo como objetivo rediseñar cada uno de los subsistemas que componen al Ballbot para lograr que funcionaran de la mejor manera posible. Posteriormente, fue de gran importancia obtener la mayor cantidad de parámetros significativos de cada uno de los subsistemas rediseñados, con

la finalidad de poder realizar al final un análisis del funcionamiento del sistema completo, basándolo en el análisis de sus subsistemas y en su funcionamiento en conjunto.

De esta manera, se logró obtener información que ayudó a realizar un diagnóstico completo del robot y con ello se realizó un plan de trabajo a futuro que aseguraría un mejor funcionamiento del producto de este proyecto.

Asimismo, se evitó, cuando fue posible, poner elementos técnicos en el cuerpo principal de este trabajo, por lo que se podrán encontrar diagramas simplificados de componentes electrónicos, diagramas de flujo, pseudocódigo, o descripciones simples de los diversos elementos que conforman este trabajo. Si el lector desea una mayor profundización en algunos de estos temas, información más detallada se colocará en el apéndice de este documento.

CAPÍTULO 1
INTRODUCCIÓN

La robótica es un campo que ha crecido de manera exponencial durante las últimas décadas. Dentro de ésta, hay una línea de investigación que en los últimos años ha sobresalido gracias a sus grandes avances, la robótica de servicio, que se enfoca a dispositivos robóticos capaces de interactuar con los seres humanos.

Para que un robot pueda interactuar adecuadamente con las personas, es necesario que éste se encuentre a su alcance de una manera cómoda, sin entorpecer la manera en que el ser humano se desarrolla en su ambiente, por lo que el tema de movilidad resulta de gran importancia cuando se habla de robots de servicio. Se han creado diversas configuraciones para que un robot móvil sea capaz de navegar en entornos humanos. De entre todas las posibles configuraciones, existe un tipo que sobresale gracias a su agilidad en entornos reducidos, su facilidad de navegación y que además es capaz de alcanzar alturas humanas. A este tipo de robot se le denomina ballbot.

1.1 ¿Qué es un ballbot?

Un ballbot es un robot móvil dinámicamente estable, diseñado para balancearse sobre una llanta esférica que solamente cuenta un punto de contacto con el suelo. Este tipo de robot se puede mover en cualquier dirección sobre un plano horizontal, es decir, es omnidireccional, lo cual junto con el hecho de que es dinámicamente estable lo vuelve muy ágil, lo

que le permite navegar en entornos que tienen cambios constantes y con diversos obstáculos.

Gracias a su agilidad para navegar, este tipo de robots se está estudiando para aplicarlo como robot de servicio dentro de hogares y oficinas e inclusive la industria. Algunos proyectos se enfocan en utilizarlos como medio de transporte personal, sin embargo, tiene un uso limitado debido a que solamente puede ser utilizado en superficies regulares, pues las imperfecciones en el suelo puede causar fallas en su funcionamiento.

1.2 Últimos avances

El primer robot de este tipo se realizó en el año 2006 en la Universidad de Carnegie Mellon [2], ubicado en Pittsburgh, PA, en los EEUU. Dicho robot se movía impulsado por un mecanismo similar al ratón de computadora, sólo que invertido. Se compone de dos rodillos motrices impulsados por motores de corriente directa, los cuales mueven al robot sobre la esfera en dos ejes ortogonales. La altura del robot era equiparable con la de un humano aunque su movimiento era poco ágil. Este ballbot contaba también con un dispositivo que lo mantenía en equilibrio estable en el suelo mientras estuviese apagado.

Posteriormente, se hizo una versión llamada BallP en la Universidad de Tohoku Gakuin [3], ubicado en Sendai, Japón. El proyecto fue dirigido por estudiantes de maestría y doctorado, y consistió básicamente en robots tipo ballbot impulsados por motores paso a paso que pueden cargar bloques de hasta 10 kg. A diferencia del ballbot de la Universidad de Carnegie Mellon, este robot tiene una altura mucho menor, se mueve sobre una bola de boliche recubierta con un polímero y tiene tres ruedas, con lo cual puede girar sobre su propio eje también. El movimiento de este robot es mucho más suave que su antecesor y tiene características de movimiento muy diferentes, como seguimiento de trayectorias. La limitante esencial del proyecto es que el intervalo válido en el cuál puede recuperar la estabilidad dinámica ante perturbaciones es para ángulos no mayores a 5°.

También se realizó la construcción de un robot de este tipo en la Universidad de Adelaide [4], ubicado en Australia. En este caso primero se hizo una iteración controlando un robot tipo ballbot con LEGO, para después construir un robot de altura similar al de una persona promedio.

Sin lugar a dudas, el robot tipo ballbot más exitoso es el “Rezero” creado por la Universidad Poitécnica de Zurich [5], de Suiza. Dicho robot supera por mucho el desempeño de los otros robots de su tipo. La inclinación máxima del robot es de 20°. El robot utiliza tres motores sin escobillas con reducción, baterías de litio polímero, un sensor láser para medir distancias, un microcontrolador ARM y una unidad de mediciones inerciales, o IMU. Sin embargo, el costo del robot se incrementó considerablemente debido a la calidad de sus actuadores.

1.3 Antecedentes

Dentro de la Facultad de Ingeniería de la UNAM, se han realizado dos proyectos similares a este. El primer antecedente es una tesis de licenciatura para obtener el grado de Ingeniero Eléctrico Electrónico [6]. Esta tesis se basó en utilizar un *Lego MINDSTORMS* para construir un ballbot de pequeñas dimensiones y realizar el control del mismo utilizando *MATLAB,* con una interfaz directa al dispositivo de control. Este proyecto logró estabilizar al robot sobre una esfera.

El segundo proyecto es una tesis de licenciatura que buscó construir un ballbot desde su diseño mecánico. Este proyecto es el antecedente más directo al presente trabajo [1]. Utilizando como principal antecedente el proyecto BallP, se logró construir una base de robot móvil, la cual era capaz de realizar movimientos en el plano horizontal con cierto grado de precisión, pero no logró estabilizarse sobre la esfera.

En dicho trabajo se concluyó que la principal razón por la cual no se lograba la estabilización eran los retrasos en las lecturas de la posición angular del robot. A la par de este diagnóstico, se comentó que se tuvieron muchos problemas con el desempeño de los diversos componentes del robot, como ruido en los componentes electrónicos, falsos contactos, desconexiones, entre otros.

En el trabajo a futuro del trabajo anterior, se dice que el principal factor a mejorar es el retraso en la adquisición de los datos de posición angular, ya que la medida obtenida fue de 410 ms con el cual resulta imposible estabilizar al robot correctamente. Además de esto, se propone que con el fin de comparar su desempeño, se utilicen otro tipo de motores. Finalmente se comenta que, una vez alcanzada la estabilización, se implementen en el

robot algoritmos de navegación y se empiece a utilizar como robot de servicio.

En la Figura 1.1 se puede ver una imagen del diseño conceptual del primer prototipo y el prototipo físico utilizado para las pruebas de funcionamiento y estabilidad.

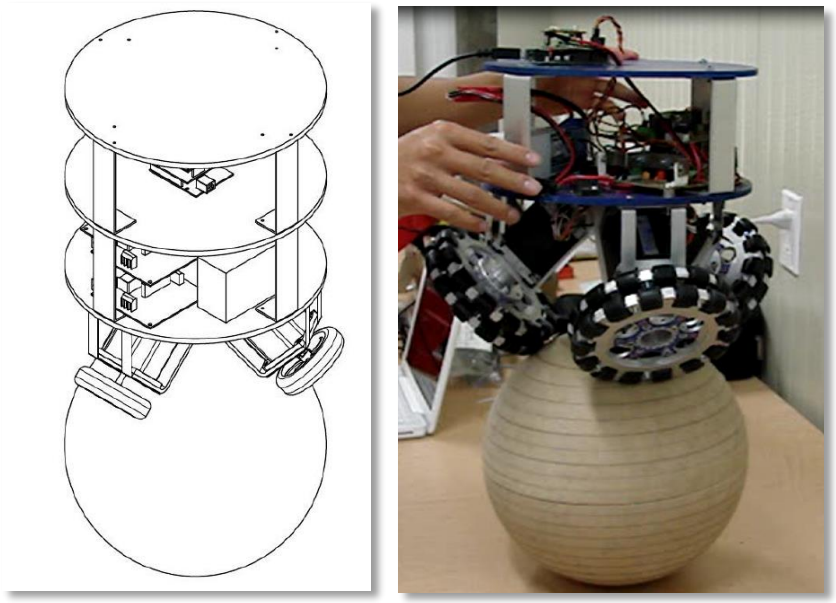


Figura 1.1 Diseño conceptual y prototipo final del primer prototipo.

1.4 Objetivo

Con base en el trabajo realizado en el primer prototipo y tomando en cuenta el trabajo a futuro propuesto, este proyecto **tiene como objetivo diseñar y construir un robot tipo ballbot que sea capaz de estabilizarse dinámicamente y que pueda servir para probar diferentes tipos de control de estabilización y navegación.** Para lograr el objetivo anterior se realizara un rediseño de cada uno de los subsistemas que conforman el primer prototipo.

1.5 Justificación

La construcción de un robot de este tipo y lograr su estabilización abre las puertas para que los alumnos de esta Facultad puedan experimentar con diversos algoritmos de control para su estabilización y para su navegación,

tanto autónoma como remota. Este tipo de actividades de investigación y experimentación ayudan a mejorar la formación personal y profesional de los estudiantes. Asimismo, el construir una plataforma de este tipo contribuye a la investigación en el área de robots móviles y de servicio.

1.6 Resumen del trabajo

Este trabajo se divide en siete capítulos. En el capítulo de análisis del problema se estudia a detalle todos los factores involucrados en este proyecto y se plasma una metodología para lograr el objetivo de este proyecto. El trabajo de diseño se inicia con el capítulo de Rediseño mecánico, en el cual se realiza un breve análisis del sistema mecánico del primer prototipo, posteriormente se realiza el nuevo diseño conceptual y diseño de detalle. Asimismo, en este capítulo se detalla el proceso de manufactura y se muestra el producto de este proceso de diseño. En el capítulo Rediseño del control de los motores se analiza la electrónica utilizada para realizar la locomoción del primer prototipo, posteriormente se proponen nuevas formas de realizarla para este proyecto y finalmente se muestra el algoritmo utilizado. Una vez que el sistema mecánico y el control de movimiento de los motores están resueltos, se aborda el tema del control de estabilización, este proceso se detalla en el capítulo de Modelado y control, en el cual se muestran simulaciones del control y de los filtros que se aplicaron al procesamiento de la posición angular.

Posterior a los capítulos dedicados a los procesos de diseño de sistemas mecánicos y de control se detallan las pruebas de funcionamiento de los diversos sistemas. Entre las pruebas que se realizaron se encuentran pruebas de movimiento de los motores, de trayectorias en el plano, de imperfecciones, de lecturas de la IMU y finalmente pruebas de estabilización. Todas estas pruebas se documentaron en el capítulo de Pruebas y resultados.

En el capítulo de Conclusiones y trabajo a futuro se puede encontrar el análisis del producto final de este proyecto dividido en los subsistemas que lo conforman. Asimismo, en este capítulo se propone trabajo a futuro para la siguiente iteración de este proyecto, con el fin de lograr una estabilización más robusta y que pueda mantenerse durante periodos prolongados. También, se propone trabajo a futuro en cuestión de navegación autónoma con el fin de utilizar este tipo de robots como robot de servicio.

Finalmente, en el apéndice se pueden encontrar los detalles más técnicos del desarrollo de este proyecto, como por ejemplo los planos de fabricación del robot, la deducción de las ecuaciones de movimiento, el diseño de los controladores y los filtros utilizados así como el código del programa del Ballbot.

CAPÍTULO 2
ANÁLISIS DEL PROBLEMA

Las bondades de los robots móviles tipo ballbot no han sido exploradas a fondo en la Facultad de Ingeniería de la UNAM, debido a que no se tienen robots con estructuras robustas capaces de servir como plataformas de servicio, o que sean flexibles a cambios simples en la programación de los mismos. Desafortunadamente, la tesis anterior en la que se basó este trabajo no consiguió el objetivo de estabilización, sin embargo, dejó una base importante de conocimiento al respecto.

El desarrollo del proyecto de investigación que se presenta, tuvo como base las cinco etapas del proceso de diseño en ingeniería, que consisten en la formulación del problema, el análisis del problema, la búsqueda de diversas soluciones, la etapa de decisión y la especificación completa de la solución propuesta al problema identificado. En la etapa de la formulación del problema se identificó la necesidad de continuar con el proyecto anterior y lograr estabilizar un ballbot. Posteriormente en el análisis del problema se detallaron a fondo las características del problema a resolver, para dar paso a la búsqueda de las diversas alternativas que se podían emplear para dar solución al problema planteado. Finalmente, después de una búsqueda y selección de componentes se decidió utilizar una serie de piezas de hardware y software específicos que resolverían el problema de forma adecuada, con lo cual se obtuvo la suficiente claridad para especificar la solución y reducirla a una sola alternativa, que finalmente es la que se

construyó y analizó a lo largo de este trabajo. A continuación se describe un poco de la metodología que se siguió.

El objetivo de este proyecto surgió como producto del análisis del trabajo a futuro del primer prototipo de un robot tipo ballbot, para el cual se desea conseguir su estabilidad y que sirva como una plataforma de pruebas para diversas estrategias de control, esto con el fin de enriquecer la formación de estudiantes de ingeniería a nivel de licenciatura y posgrado, y que adicionalmente pueda contribuir a la investigación en el área de robótica móvil dentro de la Facultad de Ingeniería.

También se busca que este robot sea lo suficientemente robusto para que, con un adecuado control, pueda servir como un agente móvil funcional y con ello se le pueda emplear en robótica de investigación.

Partiendo del análisis del primer prototipo, se observó que el robot construido tenía varios detalles en su funcionamiento que podían ser mejorados, por lo cual era plausible rediseñar varios subsistemas del mismo con la finalidad de corregir los problemas que impedían que pudiera estabilizarse.

Por otro lado, se pensó en la posibilidad de cambiar el esquema de un robot con motores paso a paso a uno que tuviera motores de corriente directa, debido a las bondades de los últimos y a que la mayoría de los desarrollos actuales de este tipo de robots emplean dichos actuadores.

Era posible también que con un rediseño completo se cambiara la configuración del robot en cuando a sus ruedas. Un problema de las mismas es que por momentos no tienen un solo punto de contacto con la esfera, sino que se deslizan sobre la misma en pequeños tramos. Esto tal vez pudiera resolverse cambiándolas por otras que tengan mejores características, ya sea adquiriéndolas o a través del rediseño y manufactura de las mismas. Otra alternativa similar era cambiar la configuración de tres a cuatro ruedas, con lo cual se incluiría un nuevo motor, pero se perdería la posibilidad de girar sobre el eje vertical.

Una opción adicional era comprar un robot ya hecho, sin embargo aún no existen dispositivos semejantes que se comercialicen, ni siquiera uno que sea parcialmente parecido.

Evaluando las alternativas propuestas, se tomó la decisión de que se realizaría un rediseño de varios subsistemas, para lo cual se consideraría el uso de los componentes e información empleada en el proyecto anterior. La decisión se basó en que se disponía de un robot casi funcional. Adicionalmente, se consideró el aspecto económico, pues para las demás alternativas se requería de materiales y dispositivos con los que no se contaba en ese momento, por lo que habría la necesidad de adquirirlos.

La información del trabajo previo fue de gran utilidad, particularmente los códigos de programación, el modelado del sistema físico y el diseño de un controlador. Los dispositivos utilizados en el primer prototipo que se conservaron fueron los motores paso a paso, sus controladores, la IMU, las ruedas omnidireccionales, una batería de litio polímero y la estructura mecánica del robot.

Finalmente, se propusieron como posibles actividades a desarrollar las siguientes:

- Desarrollo de planos de manufactura del ballbot
- Estabilización del robot en un rango de 0 a 5° alrededor de la vertical
- Consecución de una autonomía de al menos 15 min
- Búsqueda de la facilidad de adaptación a diferentes esferas
- Desarrollo de programas que puedan modificarse con facilidad, tanto para el control de los motores como el propio algoritmo de control.

Como parte del trabajo anterior, se contó con el análisis de diversas alternativas, tanto en el aspecto de programación como el de selección de componentes mecánicos; sin embargo, existen alternativas nuevas que no habían sido exploradas. A continuación se describirán brevemente algunas de éstas, y se propone un plan de trabajo para lograr los objetivos de este proyecto.

2.1 Unidad de procesamiento

En el proyecto anterior, para programar el controlador se evaluaron diversas plataformas de hardware, como el empleo de un PIC, un DSP o un FPGA, entre otros. Finalmente, se optó por emplear una tarjeta Arduino Uno que resultaba ser una alternativa viable para ese proyecto.

Después de un proceso de búsqueda en Internet, se encontraron nuevas opciones interesantes que se podrían explorar.

Se verificó que un problema en este tipo de sistemas es lograr que funcionen con desempeño muy cercano al tiempo real; es por ello que es crucial considerar procesadores capaces de ejecutar tareas en un tiempo muy corto. Dentro de la gama de micro controladores y tarjetas de desarrollo de prototipos, se encuentran alternativas superiores en capacidad de procesamiento al Arduino Uno, como son la tarjeta Stellaris fabricada por Texas Instruments. Dicha tarjeta tiene embebido un procesador ARM Cortex–M4 que tiene una velocidad de 80 MHz, tal como el que se muestra en la Figura 2.1. Similar a esta tarjeta de desarrollo, se encontró una alternativa con mayor capacidad de procesamiento, la tarjeta Arduino Due, que tiene un micro controlador ARM Cortex–M3 con un reloj de 84 MHz, como el que se observa en la Figura 2.2.

Otra opción interesante que se exploró fue la plataforma Raspberry, que es básicamente una computadora de dimensiones muy pequeñas que puede alcanzar hasta 1 GHz en su frecuencia de procesamiento. Se muestra dicha tarjeta en la Figura 2.3.

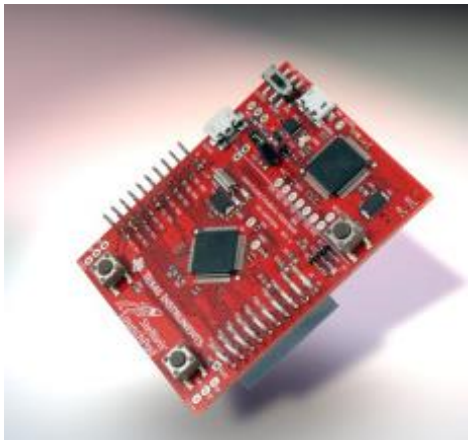


Figura 2.1 Tarjeta de desarrollo Stellaris launchpad.

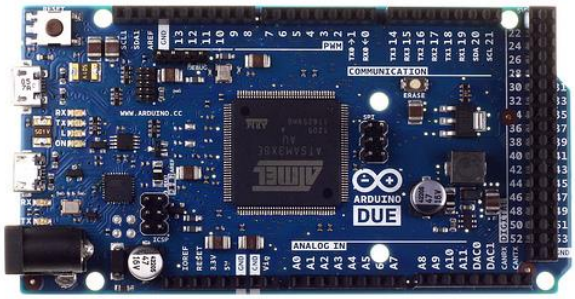


Figura 2.2 Tarjeta de desarrollo Arduino Due.

Figura 2.3 Tarjeta Raspberry Pi Mod B.

2.2 Control

En el trabajo anterior se exploraron alternativas de control diversas, desde las que se basan en modelos hasta las que son catalogadas como de control inteligente. Como parte de la búsqueda de soluciones y tras varias sugerencias de personas con experiencia en el tema, se decidió que primero se intentarían controles sencillos que estuvieran probados, para una vez que se lograra algún avance en el funcionamiento se fuera escalando en el nivel de complejidad, hasta culminar con un control que fuera lo suficientemente bueno para el correcto desempeño del robot.

2.3 Mecánica

Para la realización de este trabajo, se decidió recurrir a gran parte de los componentes utilizados en el primer prototipo, por lo que este proyecto cuenta con los mismos motores paso a paso y las mismas ruedas omnidireccionales. En cuanto a la estructura donde se colocarán los componentes que conforman al proyecto, se decidió realizar un rediseño que facilite el ensamble, sea robusto ante las vibraciones mecánicas provocadas por los actuadores y permita realizar modificaciones futuras para mejorar su funcionamiento.

2.4 Plan de trabajo

Teniendo en cuenta el objetivo planteado en este proyecto, los elementos con los que se cuenta del trabajo previo y las posibles alternativas de solución, se decidió realizar la planeación de las actividades. El plan de trabajo propuesto para el proyecto actual fue el siguiente:

- Realizar un análisis de cada subsistema del primer prototipo
- Rediseñar el subsistema mecánico y el de control de los motores
- Verificar el modelado y el algoritmo de control, así como la obtención de los datos de orientación del robot
- Realizar pruebas de funcionamiento de cada subsistema y analizar los resultados
- Efectuar pruebas de estabilización y examinar sus resultados
- Generar una conclusión acerca del funcionamiento de cada subsistema y del funcionamiento del conjunto de todos ellos
- Proponer el trabajo a futuro, con objeto de mejorar su desempeño.

CAPÍTULO 3
REDISEÑO DEL SUBSISTEMA MECÁNICO

En el trabajo a futuro del primer prototipo no se mencionan las posibles mejoras en el sistema mecánico. Debido a esto, se decidió realizar un pequeño resumen y análisis de las características de este sistema, que buscara elementos susceptibles de mejoras con la finalidad de crear un robot más robusto desde el punto de vista mecánico.

Se empezó analizando el material y la manufactura del primer prototipo. Este fue construido con el empleo de materiales como acrílico de 3 mm y lámina de aluminio para acoplar los motores y para fabricar los separadores de las diferentes etapas; para la base de prototipo y los pisos subsecuentes se empleó PVC espumado (policloruro de vinilo). En la forma en que se aplicaron estos materiales, no proporcionaban suficiente rigidez a las estructuras ni a los componentes. Para acoplar las diferentes estructuras entre sí se utilizaron tuercas convencionales, por lo que algunos elementos debían de apretarse varias veces durante las pruebas de funcionamiento a fin de evitar que las tuercas se desprendieran y cayeran sobre algún elemento al cual pudieran dañar. Asimismo, las ruedas omnidireccionales no estaban debidamente acopladas y tendían a deslizarse y desprenderse de los ejes de los motores mientras se encontraban en operación. La fabricación de éste no tenia la mayor precisión posible, lo cual no aseguraba que los dispositivos se encontraran en la posición óptima de

funcionamiento, tales como el de los motores y de las ruedas omnidireccionales con respecto a la esfera.

En cuanto a la colocación de los componentes, en algunos casos no se encontraban fijos de manera que se evitaran deslizamientos durante la operación, como el caso de la batería, que además de no estar bien fijada, se encontraba muy lejana del centro geométrico del robot, por lo que el centro de masa del prototipo no correspondía a su centro geométrico.

Finalmente, el primer prototipo era muy complicado de ensamblar y desensamblar con facilidad. Los elementos que sujetaban a los motores estaban en riesgo de ser dañados cada vez que se necesitaba desacoplar los motores del cuerpo del robot. Asimismo, para poder retirar ciertos elementos era necesario desarmar varias piezas del robot para alcanzarlos.

Los detalles antes mencionados fueron tomados en consideración junto con otros más, para realizar un rediseño que permitiera cumplir con los objetivos planteados en este proyecto.

3.1 Elementos a considerar

Se rediseñó la plataforma del ballbot en la que van acoplados los actuadores, la electrónica, la instrumentación y los demás elementos que conforman al robot, de una manera segura, eficiente y robusta, tomando en cuenta sus características físicas, así como las necesidades para su correcta operación. Se buscó que la manufactura de las piezas fuera la más sencilla y precisa posible y, que en el caso de una futura modificación, se pudieran reemplazar piezas sin tener que modificar todo el subsistema.

Los rubros que se tomaron en cuenta para el rediseño mecánico del ballbot fueron:

- Manufactura y materiales
- Colocación y funcionamiento de los componentes electrónicos y mecánicos.

A continuación se describirán detalladamente las características deseables para cada uno de los rubros anteriores.

3.1.1 Manufactura y materiales

Un factor muy importante dentro del diseño de cualquier sistema mecánico es su manufactura. En el caso del ballbot, se buscó que fuera de bajo costo y que las piezas tuvieran tolerancias precisas, para que los ensambles fueran menos susceptibles a fallas mecánicas provocadas por el juego de las piezas y las vibraciones. Asimismo, se consideró que la manufactura de nuevas piezas o el reemplazo de piezas fuera rápido y sencillo.

Con base en las características de manufactura, la elección de materiales se limitó a los que pudieran ser maquinados con corte láser. Es por eso que los dos materiales en los que se pensó en un inicio fueron lamina o placa de aluminio (dependiendo del espesor), placas de acrílico de 3, 4, ó 6 mm de espesor y PVC espumado de 3 ó 6 mm. Estos materiales son fáciles de maquinar y relativamente de bajo costo, cuando se comparan con materiales compuestos como Kevlar o fibra de carbono. También son menos pesados que la lámina de acero del mismo calibre.

Pese a que el Ballbot no se diseñó para operar en condiciones extremas (temperaturas mayores a los 40°C o menores a los 0° C), los motores paso a paso pueden llegar a calentarse después de ser utilizados durante un lapso mayor a veinte minutos a alrededor de 50°C, por lo que el material seleccionado debe soportar la temperatura de los motores sin sufrir deterioro. Igualmente, el material seleccionado debe asegurar que es capaz de soportar el peso del robot en funcionamiento, sin sufrir ninguna deformación que comprometa su desempeño.

Se decidió utilizar como material principal para la estructura y sujeción de los componentes mecánicos y electrónicos placa de acrílico de 3 mm y de 6 mm, dependiendo del tipo de pieza y las necesidades de su ensamble. Para acoplar los diferentes sistemas se optó por utilizar lámina de aluminio.

3.1.2 Colocación de los componentes electrónicos y mecánicos

Otro factor que se consideró para el rediseño fueron las características de los componentes mecánicos y electrónicos para que el Ballbot funcionara de manera óptima.

Se comenzó por considerar a los motores paso a paso que deben estar colocados a 120° uno del otro y estos se orientaron de manera que las ruedas omnidireccionales, además de formar un ángulo de 45° con la

horizontal, hicieran contacto perpendicularmente al plano tangente con la esfera.

La etapa de potencia para los motores paso a paso ya se encontraba integrada en su tarjeta controladora, por lo que sólo se consideró que la batería de alimentación se debe sujetar al cuerpo del ballbot de manera que su centro geométrico coincida con el del robot, con objeto de facilitar el control de su estabilidad al moverse sobre la esfera.

Respecto a los componentes eléctricos (la tarjeta microcontroladora *Arduino Uno, la Arduino Due y* la tarjeta controladora de los motores paso a paso *Quadstepper driver*), se distribuyeron a 120° uno del otro respecto al eje Z del Ballbot, con el mismo fin que el del acomodo de la batería. Asimismo, se procuró que quedaran acoplados rígidamente para evitar que sus conexiones fallaran, provocando falsos contactos o cortos circuitos. Se procuró que la IMU *Sensor Stick* [8] se ubicara sobre el eje Z lo más lejos posible de los motores, con objeto de reducir el ruido causado por estos sobre los sensores magnéticos.

Finalmente el rediseño contempló dejar abierta la posibilidad de poder alojar otros de sensores, con la finalidad de darle mayor versatilidad a las aplicaciones del Ballbot.

3.2 Diseño conceptual

Antes de realizar el modelado de las piezas utilizando la computadora, se decidió realizar bocetos a mano libre en los cuales se pudieran visualizar diversas soluciones para acoplar los motores, la colocación de las piezas y las posibilidades de expansión.

En las Figuras 3.1 y 3.2 se muestran algunos de los bocetos realizados.

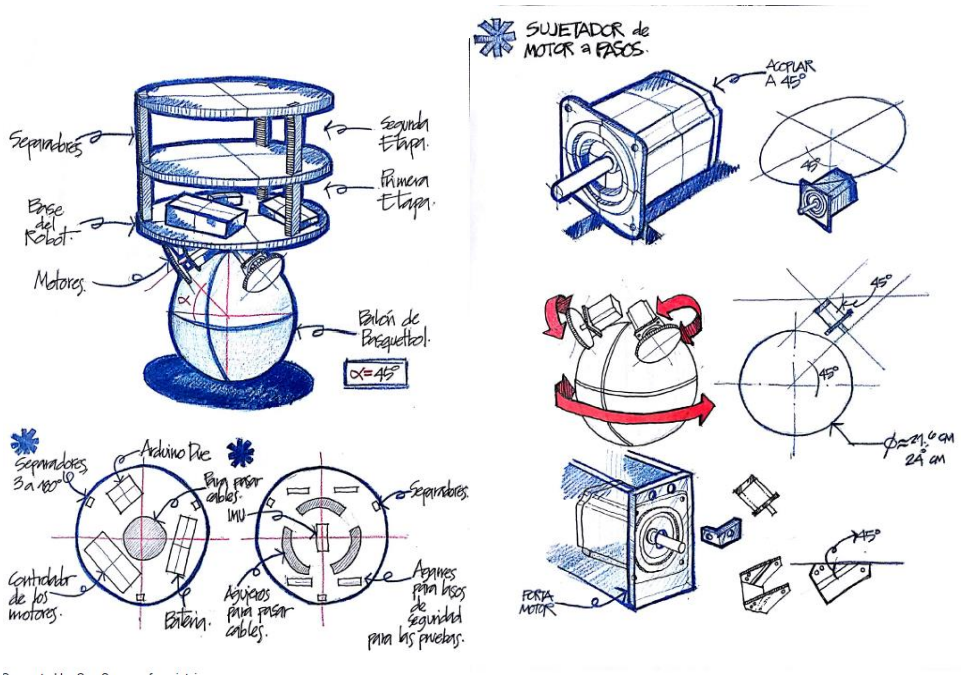


Figura 3.1 Bocetos del sistema mecánico.

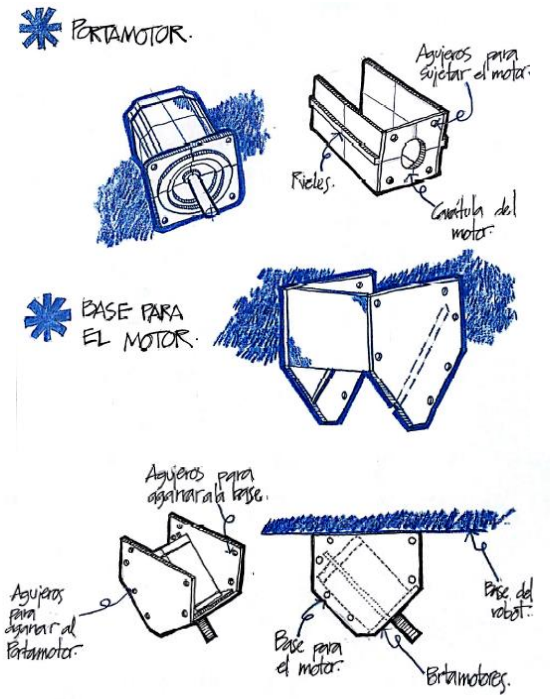


Figura 3.2 Bocetos del acoplador del motor.

Algo que se puede observar a partir de los bocetos, es que gran parte del rediseño se basó en modificar la manera en que se acoplan los motores paso

a paso a la estructura del robot. Se buscó que al ser acoplados se asegure que se encuentren en la posición correcta respecto a la esfera, por lo que se propuso la fabricación de una camisa o porta motor, que se acoplara directamente al motor y que tuviera un riel que, ayudado por tornillos, la acoplara con una pieza que sujetaría al motor con la base del Ballbot. La base del robot se ensamblaría con las bases de los motores en lugares definidos que aseguraran una posición radial fija respecto al eje vertical del Ballbot y una separación de 120° entre ellos. También se puede observar en los bocetos que, para facilitar la adición de etapas al Ballbot, se decidió facilitar la adición de módulos con el uso de separadores, de esta manera la configuración mecánica se podría expandir sin sufrir con restricciones de espacio.

3.3 Diseño de detalle

Una vez que se definió la configuración para el *Ballbot*, se procedió a diseñar cada una de las piezas con las medidas necesarias para su funcionamiento correcto. Para esta tarea se decidió trabajar en *Autodesk Inventor* en el cual se dibujaron las piezas de la manera más exacta posible. Asimismo, este programa permitió realizar los ensambles de las piezas, visualizaciones del prototipo final y realizar los planos para manufactura de una manera fácil y rápida.

Se comentó anteriormente que se utilizaría lámina de acrílico para la mayor parte de las piezas del robot, y dependiendo de la pieza, se consideraría el espesor del acrílico. Se eligió como material para los porta motores y las bases de los motores, acrílico de 3mm. Sin embargo para la base del robot y para las etapas subsecuentes se decidió utilizar acrílico de 6 mm, ya que estas piezas estarán sujetos a mayores esfuerzos mecánicos debido a que se le acoplarán los motores y los cables de seguridad que se utilizarán durante las pruebas de estabilidad. También se utilizarán ángulos de aluminio para acoplar los porta motores a las bases de los motores, y éstas a la base del robot; para estas piezas se utilizará lámina de aluminio de 1/16” y será el mismo material para los separadores de las etapas superiores del Ballbot.

3.3.1 Diseño del montaje de los motores

Como se visualizó en los bocetos mostrados anteriormente en la Figura 4.2, uno de los propósitos del rediseño fue asegurar que los motores se

encontraran acoplados en la posición correcta. Para lograr este objetivo, se decidió crear un porta motor que se acoplara mediante un riel que asegure el ángulo deseado respecto a la base del robot. Esta pieza se acoplará al motor mediante cuatro tornillos con tuercas de seguridad que evitarán que el motor tienda a aflojarse durante su operación. En la Figura 3.3 se puede ver el ensamble del porta motor en el que se resaltan las piezas que lo conforman.

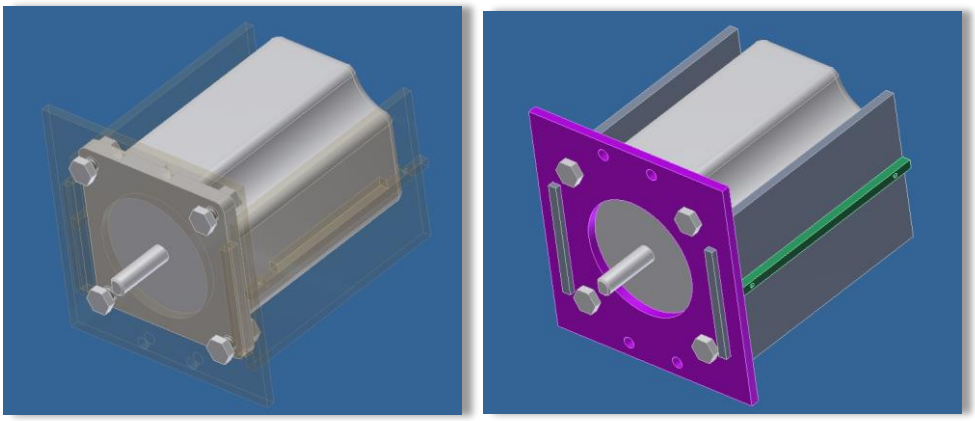


Figura 3.3 Porta motor (izquierda) y piezas que lo componen (derecha).

De igual forma, se diseñó un elemento intermedio para acoplar el porta motor con la base del robot; dicho elemento es la base del motor, la cual es una pieza que se acopla al porta motor en la posición deseada con ayuda del riel, tornillos y tuercas. Esta pieza se fijará a la base del robot mediante ángulos, los cuales asegurarán que los tres motores se encontrarán a 120° entre sí y a la distancia deseada del centro del robot. La geometría de la base del motor fue diseñada para permitir el paso de los cables de los motores paso a paso. En la Figura 3.4 se puede observar la base del motor y el porta motor acoplado a la base del motor junto con la rueda omnidireccional.

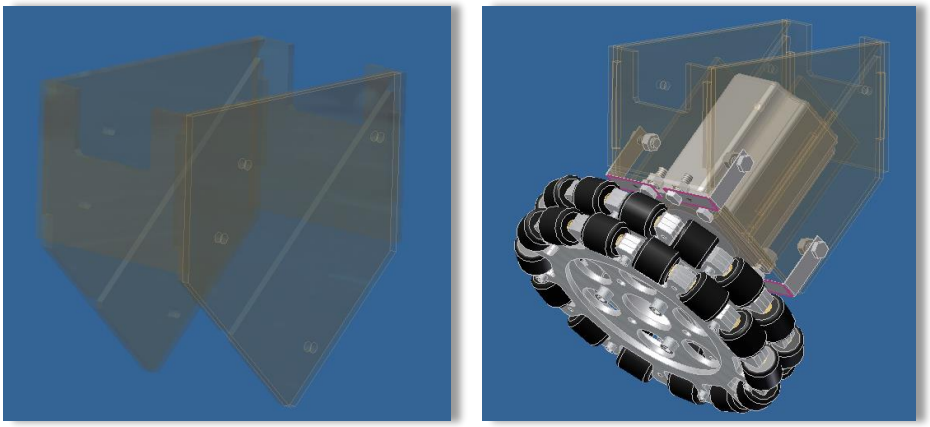


Figura 3.4 Porta motor y su acoplamiento al motor.

3.3.2 Base del robot

La pieza esencial del Ballbot es su base, ya que en ésta se acoplarán los motores, los componentes electrónicos, los sensores y servirá para alojar futuras expansiones del proyecto. Una vez resuelta la manera de colocar los motores paso a paso en la posición deseada, se diseñó la pieza donde se colocarían estos y que además servirá como soporte para los dispositivos electrónicos y las demás partes del robot: su base.

Para diseñar la base se comenzó por analizar a qué distancia debería de colocarse el motor. Para eso se tomaron en cuenta el ángulo de contacto deseado y las dimensiones de la rueda omnidireccional acoplada al motor, tal como se muestra en la Figura 3.5.

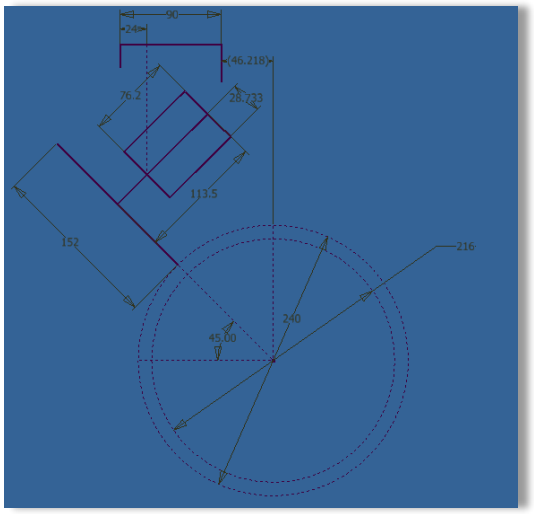


Figura 3.5 Análisis de la colocación del motor, considerando distancia y ángulo de contacto con la esfera.

Con base en la posición final de los motores, se colocaron los barrenos de sujeción para las bases de éstos. Las bases mencionadas se acoplarán mediante ángulos a la base del robot. Para poder asegurar que las piezas diseñadas fueran funcionales, se decidió realizar el ensamble de los tres motores con la base del robot y corroborar que su ensamble se vea dificultado debido a sus dimensiones. En la Figura 3.6 se muestra la base del robot sin ningún elemento colocado, en la Figura 4.7 se muestra un motor acoplado en proyección frontal y lateral. La colocación de los tres motores en su posición final se muestra en las Figuras 3.8 y 3.9 en sus proyecciones superior y lateral.

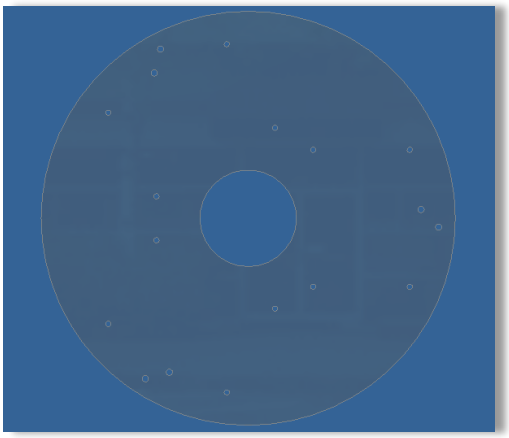


Figura 3.6 Base del Ballbot.

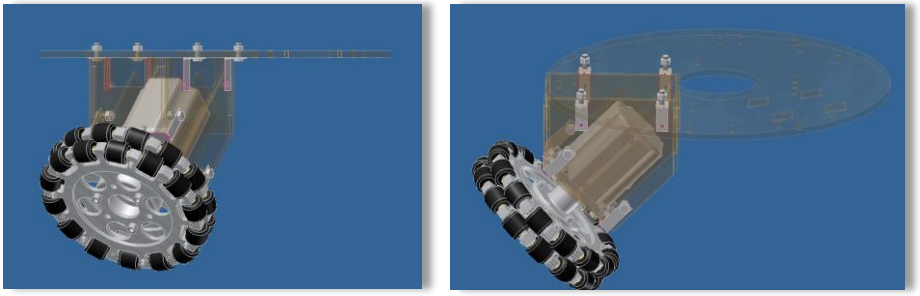


Figura 3.7 Colocación del porta motor en la base del robot. Proyecciones lateral y frontal.

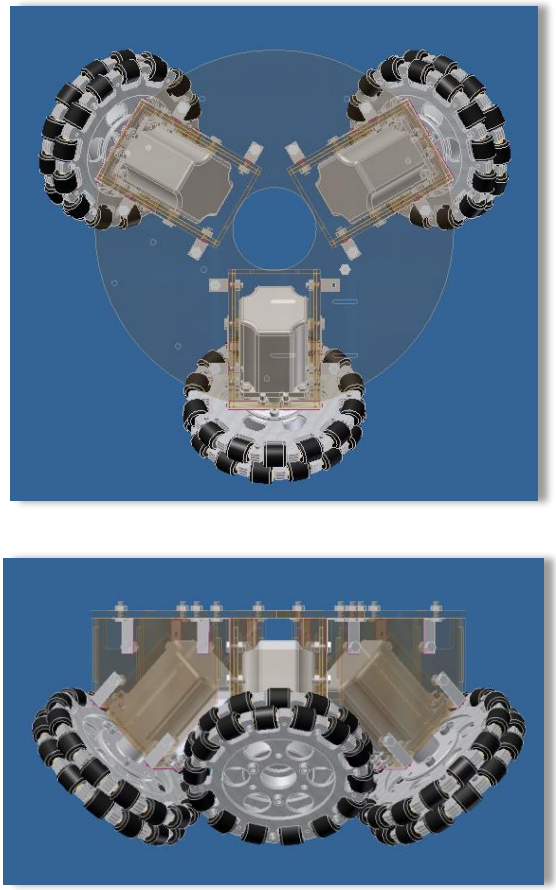


Figura 3.8 y 3.9 Colocación de los tres motores con porta motor en la base del Ballbot. Vista superior y lateral.

Gracias al ensamble de las piezas, se pudo observar que los motores se encontraban en la posición deseada y no interferían entre sí. Asimismo, el ensamble de los motores con la base demostró que el armado será sencillo de realizar con el prototipo físico y permitiría retirar uno o más motores sin comprometer la integridad de las piezas que conforman al subsistema mecánico.

Una vez que se realizó el ensamble, se analizó la distribución y colocación de los demás dispositivos que conforman al Ballbot. Fueron necesarias varias iteraciones para lograr colocar todos los dispositivos electrónicos en el lugar deseado.

3.3.3 Sujeción de los componentes electrónicos y las baterías

Se buscó que los dispositivos encargados de realizar las tareas de locomoción del Ballbot estuvieran acoplados directamente a su base. Se

consideró la posición deseada para la batería y a partir de ésta se colocaron los demás componentes. En la Figura 3.10 se muestra la base del robot con los motores, representados por cuadros amarillos, el controlador de los motores, cuadro rojo, el Arduino Uno, cuadro café y el Arduino Due, cuadro naranja; también se muestran los separadores de aluminio representados por cuadros negros.

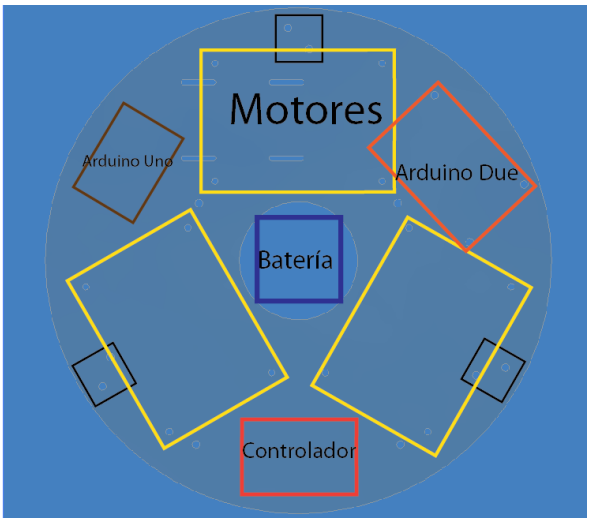


Figura 3.10 Disposición de los dispositivos del robot sobre la base.

Pese a que pareciera que algunos elementos se sobreponen, esto no presenta ninguna dificultad ya que están en diferentes planos y su ensamble no se ve obstruido por algún otro componente.

En la configuración se puede observar que, pese a que aún existen posibles espacios para ser utilizados en la base del robot, no está disponible la zona donde se pretendía fijar la IMU, por lo que su colocación en el piso superior fue la mejor opción.

3.3.4 Colocación y sujeción de la IMU

Como se comentó anteriormente, se buscó colocar la IMU lo más cercana al eje de rotación del robot y, como se mencionó en la sección anterior, la base del robot no permitió colocar la IMU donde se deseaba, por lo que se decidió colocarla en el centro del piso superior.

La ubicación y dirección de la IMU son muy importantes, así que se procuró que ésta sólo pudiera ser colocada en un solo sentido, evitando que se orientara de manera incorrecta, para no modificar el sistema de referencia del Ballbot por error. Asimismo, se decidió colocar símbolos como referencia

visual del eje de las abscisas y de las ordenadas, para facilitar las pruebas de movimiento, estabilización y navegación. Debido a que no se planeó agregar un segundo piso porque no se requería en ese momento, estas referencias visuales podrían ser observadas desde un plano superior sin ningún problema durante la realización de las pruebas. En la Figura 3.11 se muestra el piso superior con el espacio asignado para la IMU resaltado en color rojo; se pueden observar las referencias visuales de los ejes coordenados.

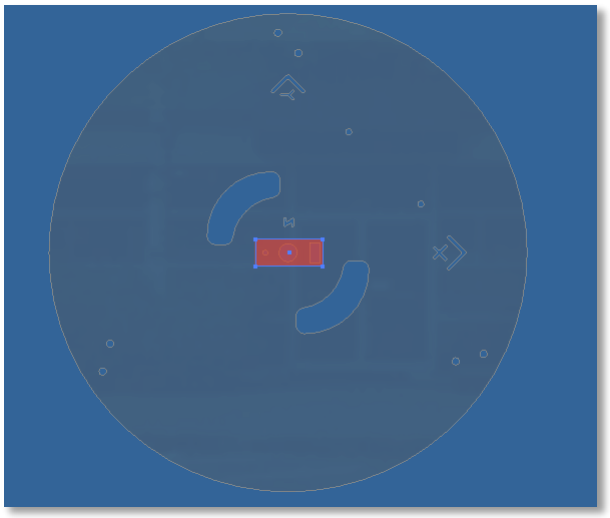


Figura 3.11 Ubicación de la IMU en el piso superior.

3.3.5 Opciones de expansión

Debido a que era posible que en futuras implementaciones de este proyecto se necesitara acoplar un mayor número de dispositivos eléctricos como sensores o computadoras portátiles, se decidió dejar abierta la posibilidad de agregar pisos o módulos superiores. Para agregarlos, será necesario fabricar un siguiente piso utilizando las medidas del primero como referencia. En la Figura 3.12 se muestra una imagen del Ballbot con un segundo piso; se podrán seguir agregando pisos conforme el proyecto lo requiera, sin necesidad de modificar los pisos inferiores.

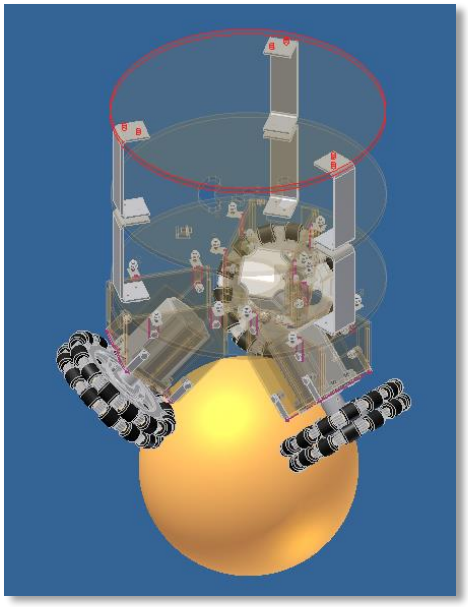


Figura 3.12 Ballbot con pisos superiores.

3.3.6 Ensamble completo en CAD

Después de diseñar las diferentes piezas necesarias para el robot, se ensamblaron todas las piezas diseñadas y se colocó al Ballbot sobre una esfera de las dimensiones de un balón de básquetbol, con el fin de tener una idea general de cómo se vería en condiciones de funcionamiento. Esto fue importante para verificar si las consideraciones geométricas planteadas al inicio se cumplieron satisfactoriamente. Esta verificación también fue útil para medir algunos parámetros necesarios en las ecuaciones del modelo matemático que se utilizaría en el control del robot.

En la Figura 3.13 se pueden observar las proyecciones horizontal y frontal del robot sobre la esfera. En la Figura 3.14 se observa una perspectiva del sobre la esfera y un detalle del contacto entre la rueda omnidireccional y la esfera de básquetbol.

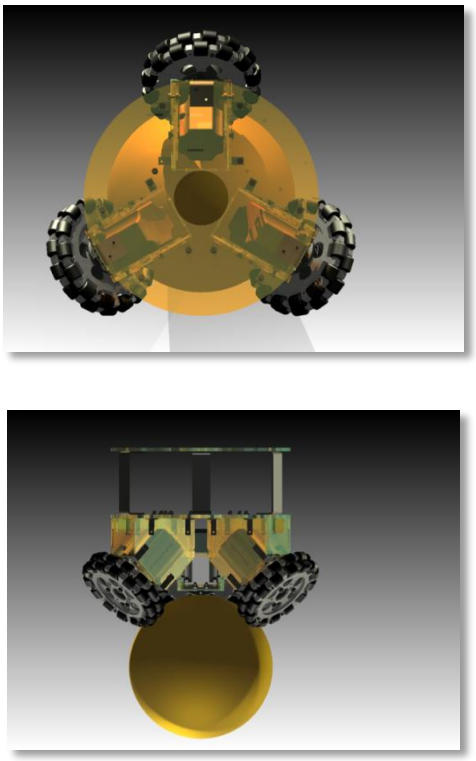


Figura 3.13 Proyecciones horizontal y frontal del robot sobre la esfera.

Figura 3.14 Ilustración del robot sobre la esfera y detalle del contacto con la esfera.

3.4 Manufactura

El diseño del Ballbot fue pensado para que se pudiera manufacturar mediante corte láser de acrílico, además de fabricar algunas piezas en lámina de aluminio doblada, por lo que se realizaron los planos correspondientes para el corte láser. En las Figuras 3.15, 3.16 y 3.17 se muestran los planos de corte láser para la base del robot en acrílico de 6

mm, un porta motor y la base del motor en acrílico de 3 mm y el piso superior en acrílico de 6 mm.

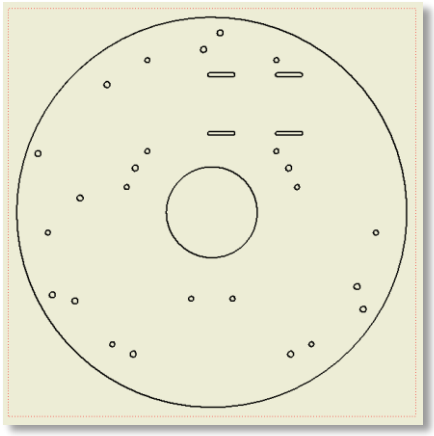


Figura 3.15 Plano para corte láser de la base del robot.

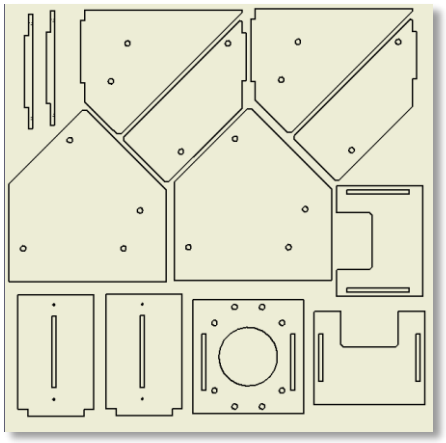


Figura 3.16 Plano para corte láser del porta motor.

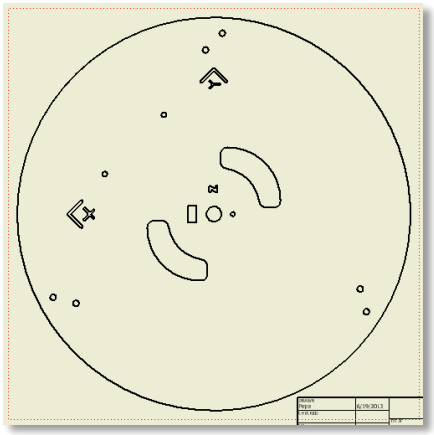


Figura 3.17 Plano para corte láser del piso superior.

Una vez realizado el corte láser, las piezas se unieron mediante pegamento especial para acrílico, tornillos con tuercas de seguridad y láminas dobladas de aluminio.

Después de realizar el ensamble de las piezas cortadas en acrílico y de manufacturar las piezas necesarias tales como los separadores y las camisas para los porta motores, se acoplaron los componentes eléctricos, electrónicos y las baterías para tener el prototipo listo para realizar las pruebas de movimiento y de estabilidad.

A continuación se muestran las imágenes del prototipo final con el cual se realizaron las pruebas de movimiento y control. En la Figura 3.18 se muestran fotografías del robot visto por arriba y de frente, respectivamente. En la Figura 3.19 se muestran dos perspectivas del prototipo final sobre balón de básquetbol.

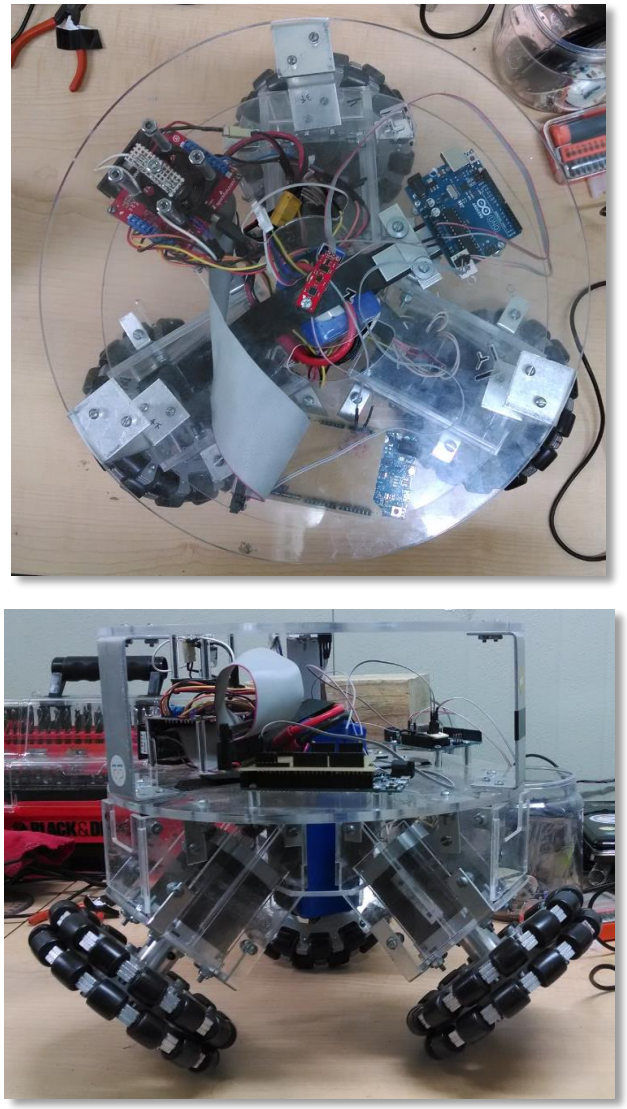


Figura 3.18 Fotografías del robot visto por arriba y de frente.

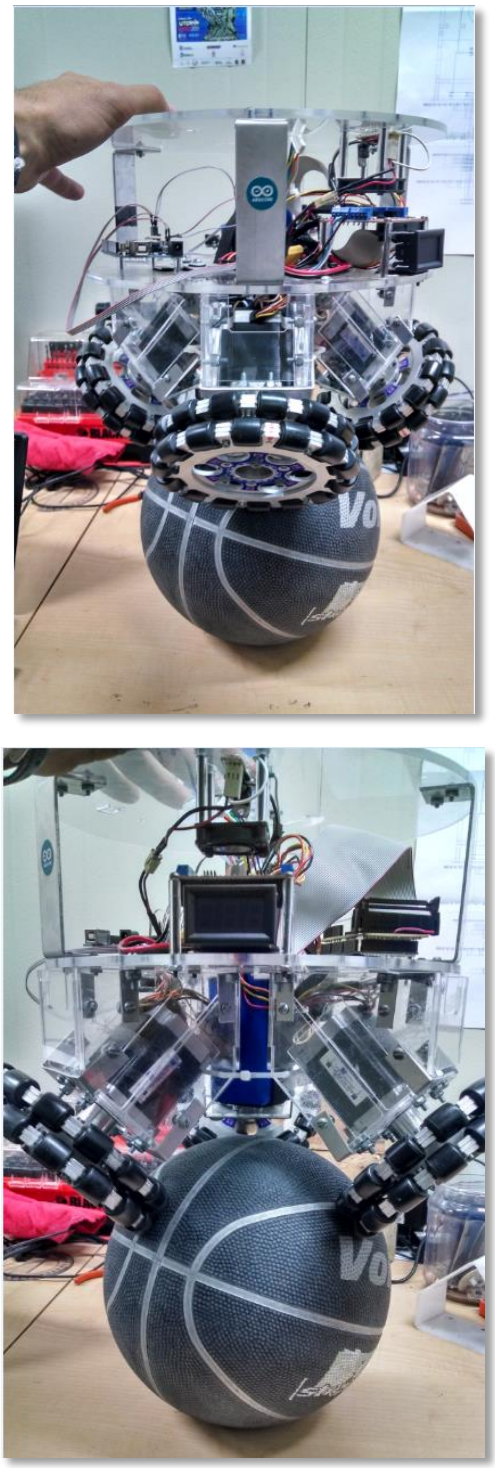


Figura 3.19 Fotografía del Ballbot sobre el balón de básquetbol.

CAPÍTULO 4
REDISEÑO DEL SUBSISTEMA DE CONTROL DE LOS MOTORES

Los motores que se utilizaron en este proyecto son motores paso a paso. A diferencia de los motores de corriente directa, o CD, los cuales se pueden controlar mediante voltaje, y existen un sinfín de dispositivos electrónicos para facilitar su control, los motores paso a paso necesitan de un tren de pulsos específico para dar un paso en una dirección determinada. Estos motores son ampliamente utilizados en la industria, por lo que también se pueden encontrar en el mercado diversas tarjetas controladoras. Sin embargo, no son tan utilizados para proyectos de control similares al presente proyecto, por lo que la selección de dispositivos electrónicos que existen para controlar los motores paso a paso no es tan amplia.

En este proyecto se requirió controlar tanto la velocidad como la aceleración de los motores paso a paso, por lo que se realizó un *microstepping* (dividir cada paso del motor en más pasos intermedios) de tres motores paso a paso simultáneamente. Para lograrlo, se tuvo que buscar un dispositivo que funcionara de manera adecuada, y que fue de vital importancia para el correcto funcionamiento del Ballbot.

4.1 Rediseño del módulo de control de motores

Para realizar el movimiento de los motores en el primer prototipo, se generaban tres señales PWM, una para cada motor paso a paso, y tres señales de dirección en un A*rduino Uno*; estas eran recibidas por tres tarjetas llamadas MCP (Modulo de control del motor paso a paso). Cada tarjeta recibía la señal PWM y mediante un filtro RC que la transformaba en una señal analógica, y a continuación esta nueva señal era recibida por un oscilador que generaba un tren de pulsos de frecuencia proporcional al voltaje de entrada de la señal analógica. Este tren de pulsos funcionaba como señal de control de pasos en un circuito integrado TA8435H que transformaba el tren de pulsos en pasos del motor. La señal de dirección era alimentada al TA8435H para controlar la dirección del paso a realizarse. En la Figura 4.1 se muestran las tarjetas MCP del primer prototipo vistas de lado y de frente.

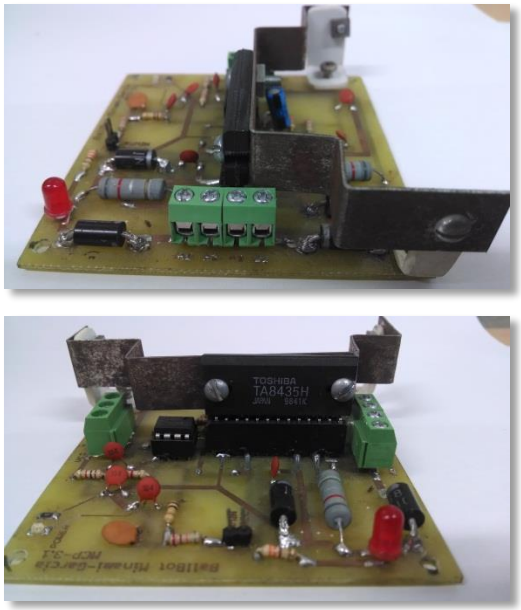


Figura 4.1 Módulo de control del motor paso a paso del primer prototipo.

Esta tarjeta presentó fallas como ruido y falsos contactos, presumiblemente debido a errores en la manufactura y en las conexiones de los cables. Es por ello que se decidió diseñar una nueva tarjeta basada en el mismo circuito integrado, pero sin emplear el oscilador con la finalidad de reducir la complejidad del circuito y fabricar la tarjeta de circuito impreso en una sola cara; ésta se manufacturó utilizando una máquina de control numérico para

reducir los falsos contactos. Este nuevo MCP recibió el tren de pulsos y la dirección directamente del *Arduino Uno*, utilizando una biblioteca para generar un tren de pulsos de frecuencia variable. En la Figura 4.2 se al nuevo MCP visto de frente y de lado.

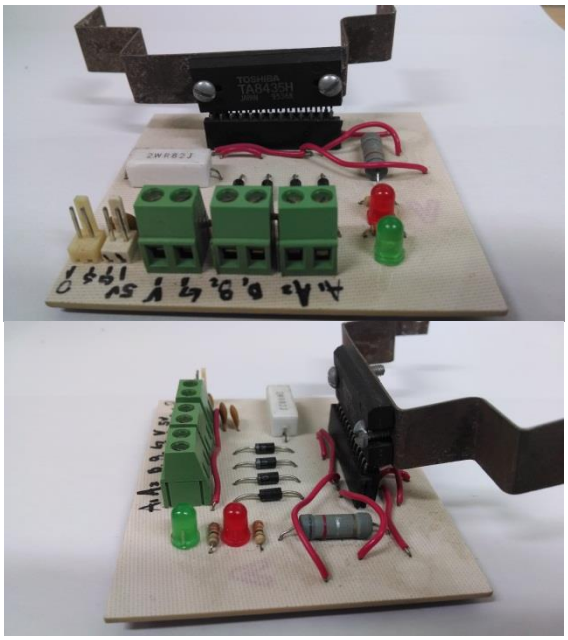


Figura 4.2 Módulo de control de motores rediseñado.

Sin embargo, pese a las consideraciones tomadas para la realización de la tarjeta, persistieron algunos problemas de ruido, por lo que se propuso buscar una alternativa que pudiera encontrarse en el mercado que utilizara circuitos integrados similares a los propuestos para los MCP anteriores.

Se decidió realizar una búsqueda en diversos medios para saber de qué manera se realiza el control de este tipo de motores en otras aplicaciones. Se encontró que los sistemas de manufactura asistida por computadora utilizan tarjetas controladoras que emplean circuitos integrados similares al de este proyecto para controlar motores paso a paso, por lo que se buscaron controladores diseñados para sistemas de control numérico que fueran capaces de proporcionar la corriente necesaria para los motores con los que se contaba.

Se encontraron diversas opciones de controladores para motores paso a paso. Inicialmente se decidió adquirir una tarjeta llamada *TB6560HQV3-T3,*

la cual utiliza tres circuitos integrados TB6560HQ que son la mejora del TA8435H. En la Figura 4.3 se puede observar la citada tarjeta.

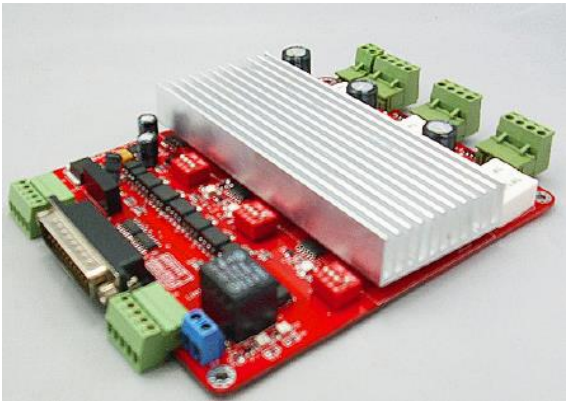


Figura 4.3 Tarjeta controladora de motores paso a paso de 3 ejes.

Esta tarjeta tiene una función muy similar a la del segundo MCP, necesita como entradas de control un tren de pulsos y una dirección de movimiento para cada uno de los tres motores paso a paso, con la característica de que las entradas de control están opto-acopladas con la etapa de potencia, lo cual reduce el ruido que recibe el micro controlador encargado del control del Ballbot y que se genera debido al funcionamiento de los motores.

Al realizar pruebas de respuesta y funcionamiento de la tarjeta al operar los tres motores paso a paso, se corroboró que ésta funcionaba correctamente, logrando el objetivo de optimar el funcionamiento del controlador. Sin embargo, al realizar pruebas de movimiento del Ballbot, la tarjeta falló y dejo de funcionar correctamente. Se presume que fue debido a fallas de fábrica, ya que no lograba proporcionar la corriente necesaria para que los motores se movieran correctamente.

Se decidió reemplazar esta tarjeta por la *Quadstepper Motor Driver* (ROB-10507), la cual utiliza otro circuito integrado con matrícula A4983 y es capaz de controlar cuatro motores paso a paso de hasta 1 A por fase, sin la necesidad de disipador, o 2 A con disipador. Los motores se controlaban de la misma manera que con el *TB6560HQ*, por lo que la modificación del programa de control de los motores del Ballbot no presentó ningún problema.

Después de realizar una serie de pruebas de funcionamiento y someter a la tarjeta a tiempos de trabajo prolongado y altas cargas mecánicas, la tarjeta no presentó ningún problema, por lo que se procedió a trabajar en los algoritmos de control de velocidad y aceleración del Ballbot. La tarjeta *Quadstepper Motor Driver* se puede muestra en la Figura 4.4.

Figura 4.4 Tarjeta controladora de motores paso a paso Quadstepper Motor Driver.

4.2 Algoritmo de movimiento del Ballbot

Como ya se ha comentado anteriormente, el movimiento de los motores paso a paso por medio de la tarjeta controladora se realizó mediante un tren de pulsos. Para controlar la velocidad, el tren de pulsos debe ser de frecuencia variable, específicamente para la configuración del *microstepping* del controlador, estaba configurado por software a 1/16, es decir, cada paso es dividido en 16 pasos intermedios. La relación entre la frecuencia del tren de pulsos y la velocidad angular del motor paso a paso es la siguiente:

$$f = \omega * ms * 200$$

Donde f es la velocidad en pasos por segundo deseada, ω es la velocidad angular en revoluciones por segundo, y ms es el valor del *microstepping* configurado, y la constante *200* es el número de pasos que tiene el motor. El número de pasos por revolución es igual al producto del número de pasos que tiene el motor sin realizar el *microstepping* por la configuración del controlador, en este caso 16. Así pues, el problema de controlar la velocidad de los motores se redujo a generar un tren de pulsos de frecuencia variable.

Para generar dicha frecuencia variable se propuso inicialmente programar un contador dentro de una interrupción que se llamara al doble de la frecuencia que correspondía a la velocidad máxima deseada para el motor, en el caso particular del proyecto 10 KHz, dentro de la interrupción se generó un tren de pulsos que era proporcional al valor del contador. Sin embargo, este algoritmo generaba un comportamiento exponencial, pues la resolución a medias y altas frecuencias en el rango de 3200 Hz- 5000 Hz no era suficiente, solamente a frecuencias menores a 500 Hz se obtenía una resolución adecuada. Por este motivo se propuso un segundo algoritmo que generó una interrupción de frecuencia variable. Éste resolvió el problema de la resolución a diferentes frecuencias, sin embargo, cuando la frecuencia deseada del tren de pulsos era menor que la frecuencia en la que se actualizaba la variable de la interrupción, el algoritmo no generaba la interrupción. Para corregirlo se propuso una tercera solución, basada en un programa que contara el tiempo que transcurrió entre un paso y otro, y decidiera mediante una comparación con la frecuencia deseada, si era momento o no de dar el siguiente paso. Debido a que el desarrollo de un programa que ejecutara esta tarea de manera confiable podría tomar mucho tiempo, se decidió buscar en la Internet programas que realizaran una tarea similar. Se logró encontrar una biblioteca capaz de controlar la velocidad y la aceleración de motores paso a paso llamada *AccelStepper,* creada por Mike McCauley [15], la cual realiza un procedimiento muy similar al comentado anteriormente. Posteriormente, se decidió realizar pruebas de funcionamiento a la biblioteca con un motor paso a paso y después con los otros dos motores funcionando en paralelo.

Al realizar las pruebas, se verificó que la biblioteca funciona correctamente, sin embargo al poner a funcionar en paralelo a los tres motores, el algoritmo de aceleración generó colisiones de interrupciones en el microcontrolador lentificándolo y provocando errores en la velocidad y aceleración de los motores; no obstante, al solamente controlar la velocidad de los tres motores, estos funcionaron correctamente, por lo que se decidió utilizar la biblioteca para controlar únicamente las velocidades.

Debido la incapacidad de la citada biblioteca para controlar la aceleración de los tres motores simultáneamente, se decidió agregar un algoritmo para el control de la aceleración que actuará conjuntamente con la biblioteca para controlar la velocidad. Este algoritmo simplemente consistió en llamar a la acción de control de velocidad dentro de una interrupción de frecuencia fija,

y multiplicar la aceleración deseada por el tiempo entre la salida de la interrupción y la obtención de un nuevo valor de aceleración. Se verificó que este algoritmo fue robusto para controlar el movimiento de los motores en el prototipo final.

Finalmente, el algoritmo de control del movimiento de los motores recibió los datos de velocidad deseada por medio de comunicación por I2C con el módulo de control, con una frecuencia de 200 Hz.

En la Figura 4.5 se muestran los diagramas de flujo del algoritmo que controla el movimiento de los motores paso a paso. El código completo se encuentra en el Apéndice A.6.1.

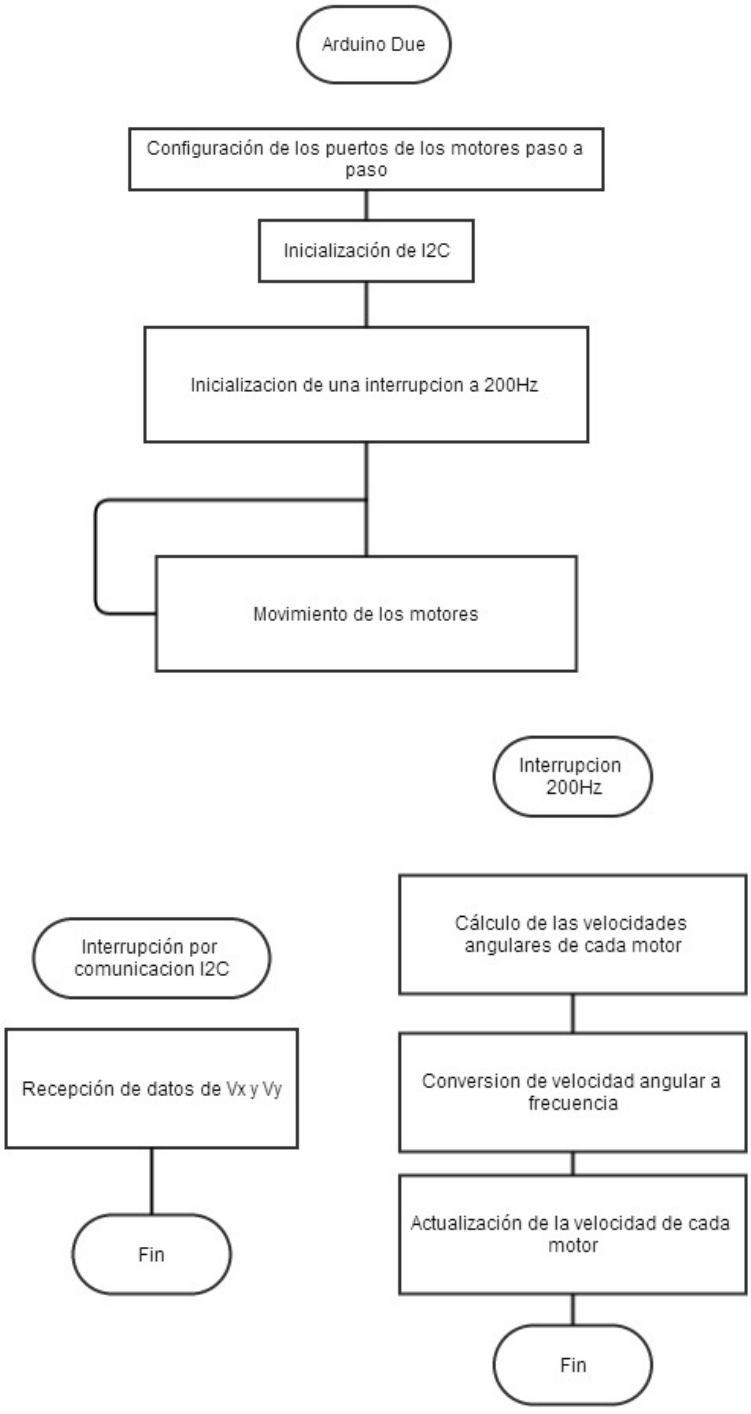


Figura 4.5 Diagramas de flujo del programa de control de motores.

CAPÍTULO 5
MODELADO Y CONTROL DE ESTABILIDAD

5.1 Objetivos

En el trabajo a futuro del primer prototipo se comentó la posibilidad de utilizar diferentes tipos de control, una vez que el retraso en la obtención de las señales de la IMU fuera eliminado; sin embargo, se realizó un análisis a fondo del sistema de control a partir del cual se definieron los siguientes objetivos:

- Verificar las ecuaciones del modelo obtenido.
- Analizar diferentes estrategias de control y evaluar la factibilidad de uso de cada una de ellas.
- Resolver el problema de la obtención de la orientación del robot.
- Implementar el control de estabilidad del robot en una plataforma de hardware que permita una mayor versatilidad.
- Estabilizar al robot.

5.2 Modelado

El robot tipo ballbot que se analizó se puede considerar como un péndulo invertido que actúa en múltiples planos que contienen al eje vertical que pasa por el centro del robot. Con propósitos de simplificación, se analizó el

sistema solamente en dos planos ortogonales que contienen a los ejes cartesianos x y y del sistema de referencia global propuesto, asumiendo que el sistema está desacoplado, es decir, que el movimiento se puede considerar independiente uno del otro en los respectivos planos de análisis.

Se realizó el modelado del sistema utilizando la metodología de Euler-Lagrange, que se basa en el análisis de la energía del sistema para encontrar las expresiones matemáticas que describen su dinámica. Se asumió que no existe deslizamiento entre las ruedas y la esfera, así como tampoco entre la esfera y la superficie sobre la que se desplaza.

La Figura 5.1 describe gráficamente los parámetros del robot y las variables del sistema en uno de los planos de análisis, donde:

m_p: masa del péndulo
I_p: momento de inercia del péndulo
m_e: masa de la esfera
I_e: momento de inercia de la esfera
d: distancia del centro de giro al centro de masa
G: centro de masa del péndulo
r_e: radio de la esfera
θ: posición angular del péndulo
Ψ: posición angular de la esfera
τ: par aplicado a la esfera
x: posición lineal sobre el eje x
v: velocidad lineal
a: aceleración lineal

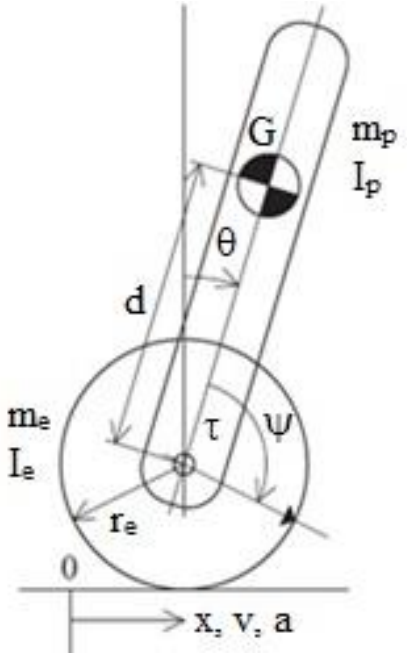


Figura 5.1 Descripción gráfica del plano de análisis del robot.

Sean $q_1 = \theta$ y $q_2 = \psi$ las coordenadas generalizadas del sistema.

La energía cinética total del sistema, T, es la suma de la energía cinética del péndulo, T_p, y la energía cinética de la esfera, T_e:

$$T_p = \frac{1}{2} m_p((\dot{q}_1 + \dot{q}_2) r_e + \dot{q}_1 d \cos q_1)^2 + \frac{1}{2} I_p \dot{q}_1{}^2$$

$$T_e = \frac{1}{2} m_e((\dot{q}_1 + \dot{q}_2) r_e)^2 + \frac{1}{2} I_e (\dot{q}_1 + \dot{q}_2)^2$$

$$T = T_p + T_e = \frac{1}{2} m_p((\dot{q}_1 + \dot{q}_2) r_e + \dot{q}_1 d \cos q_1)^2 + \frac{1}{2} m_e((\dot{q}_1 + \dot{q}_2) r_e)^2 + \frac{1}{2} I_p \dot{q}_1{}^2 + \frac{1}{2} I_e (\dot{q}_1 + \dot{q}_2)^2$$

La energía potencial del sistema está determinada por la componente del peso del robot sobre la esfera y es función del ángulo de inclinación del mismo:

$$V = m_p g d \cos q_1$$

Por último, se consideró que el movimiento del robot sobre la esfera será suave y lento cerca del punto de estabilidad, además de considerar despreciable la fricción de la esfera con la superficie sobre la que se desplaza. Con estas suposiciones fue posible despreciar la energía de

pérdidas por disipación que son función del cambio de posición y de la velocidad del robot:

$$D = 0$$

Una vez definidas las expresiones matemáticas de la energía del sistema se procede con el análisis, siguiendo la metodología de Euler-Lagrange.

El lagrangiano del sistema depende de la energía cinética y la energía potencial; se expresa como:

$$L = T - V$$

Se sustituyeron las ecuaciones de energía del sistema con el fin de obtener el lagrangiano como una función de las coordenadas generalizadas del sistema:

$$L = \frac{1}{2} m_p((\dot{q}_1 + \dot{q}_2) r_e + \dot{q}_1 d \cos q_1)^2 + \frac{1}{2} m_e((\dot{q}_1 + \dot{q}_2) r_e)^2 + \frac{1}{2} I_p \dot{q}_1{}^2 + \frac{1}{2} I_e(\dot{q}_1 + \dot{q}_2)^2 - m_p g d \cos q_1$$

Posteriormente se obtuvieron las ecuaciones de Euler-Lagrange para cada coordenada generalizada, esto es:

para q_1:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_1} - \frac{\partial L}{\partial q_1} + \frac{\partial D}{\partial \dot{q}_1} = 0$$

para q_2:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_2} - \frac{\partial L}{\partial q_2} + \frac{\partial D}{\partial \dot{q}_2} = \tau$$

El modelo matemático final del sistema quedó determinado por ambas ecuaciones, las cuales pueden ser representadas matricialmente como:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = E$$

M es conocida como la matriz de inercias del sistema, en tanto que C es la matriz que contiene los términos de fricción y de Coriolis y G es la matriz de los términos relacionados con la energía potencial. E es la matriz de entradas del sistema.

Para el caso del sistema analizado, las matrices resultantes fueron:

$$M(q) = \begin{bmatrix} \alpha & \beta \\ \beta & I_e + m_e {r_e}^2 + m_p {r_e}^2 \end{bmatrix}$$

$$\alpha = I_p + I_e + m_e {r_e}^2 + m_p r_e (r_e + d\cos q_1) + m_p d\cos q_1 \, (r_e + d\cos q_1)$$

$$\beta = I_e + m_e {r_e}^2 + m_p r_e (r_e + d\cos q_1)$$

$$C(q, \dot{q}) = \begin{bmatrix} -m_p r_e d \sin q_1 \, \dot{q}_1 - m_p d^2 \cos q_1 \sin q_1 \, \dot{q}_1 \\ -m_p r_e - d \sin q_1 \, \dot{q}_1 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} -m_p g d \sin q_1 \\ 0 \end{bmatrix}$$

Es importante destacar que este análisis es válido para ambos planos, dado que se asume independencia entre ambos.

Para expresar las ecuaciones que modelan al sistema en variables de estado, se establecieron las siguientes variables:

$$x_1 = \theta$$

$$x_2 = \dot{\theta}$$

$$x_3 = \psi$$

$$x_4 = \dot{\psi}$$

y sus derivadas:

$$\dot{x_1} = \dot{\theta} = x_2$$

$$\dot{x_2} = \ddot{\theta}$$

$$\dot{x_3} = \dot{\psi} = x_4$$

$$\dot{x_4} = \ddot{\psi}$$

Dado que $q_1 = \theta$ y $q_2 = \psi$, entonces:

$$\ddot{q} = \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix}$$

$$\ddot{q} = M(q)^{-1}[E - C(q, \dot{q})\dot{q} - G(q)]$$

Cabe mencionar que las ecuaciones diferenciales obtenidas resultaron ser no lineales.

5.3 Control de estabilidad

Cómo una primera aproximación al control del robot, se planteó la resolución del problema de estabilización utilizando estrategias de control clásico; dichas estrategias requieren que el modelo matemático de la planta esté determinado por ecuaciones diferenciales lineales invariantes en el tiempo, por lo cual fue necesario convertir a ecuaciones diferenciales lineales el modelo previamente obtenido. Para efectos prácticos se realizó el proceso mencionado cerca de uno de los puntos de equilibrio del sistema; al modelar a éste como un péndulo invertido, fue posible identificar múltiples puntos de equilibrio, espaciados cada $n\pi$ donde $n = 0,1,2, \dots, n$. Si se define el sistema de referencia comenzando en el eje vertical, es posible identificar el punto de equilibrio deseado que se encuentra cuando $n = 0$, es decir para $\theta = 0,\ \psi = 0,\ \dot{\theta} = 0, \dot{\psi} = 0,\ \tau = 0$. Dada la naturaleza del sistema, el punto de equilibrio mencionado es inestable, por lo cual uno de los objetivos del diseño del control fue la estabilización.

Entonces, se tiene que las ecuaciones lineales resultantes de la conversión son:

$$\ddot{\theta}_{lin} = \ddot{\theta}_{P.eq} + \left.\frac{\partial \ddot{\theta}}{\partial \theta}\right|_{P.eq} (\theta - \theta_{P.eq}) + \left.\frac{\partial \ddot{\theta}}{\partial \dot{\theta}}\right|_{P.eq} (\dot{\theta} - \dot{\theta}_{P.eq}) + \left.\frac{\partial \ddot{\theta}}{\partial \psi}\right|_{P.eq} (\psi - \psi_{P.eq}) + \left.\frac{\partial \ddot{\theta}}{\partial \dot{\psi}}\right|_{P.eq} (\dot{\psi} - \dot{\psi}_{P.eq}) + \left.\frac{\partial \ddot{\theta}}{\partial \tau}\right|_{P.eq} (\tau - \tau_{P.eq})$$

$$\ddot{\psi}_{lin} = \ddot{\psi}_{P.eq} + \left.\frac{\partial \ddot{\psi}}{\partial \theta}\right|_{P.eq} (\theta - \theta_{P.eq}) + \left.\frac{\partial \ddot{\psi}}{\partial \dot{\theta}}\right|_{P.eq} (\dot{\theta} - \dot{\theta}_{P.eq}) + \left.\frac{\partial \ddot{\psi}}{\partial \psi}\right|_{P.eq} (\psi - \psi_{P.eq}) + \left.\frac{\partial \ddot{\psi}}{\partial \dot{\psi}}\right|_{P.eq} (\dot{\psi} - \dot{\psi}_{P.eq}) + \left.\frac{\partial \ddot{\psi}}{\partial \tau}\right|_{P.eq} (\tau - \tau_{P.eq})$$

Ahora, es posible expresar el sistema en la forma convencional de variables de estado como:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

donde x es el vector de estados, y es la salida deseada y u es la entrada de control, en este caso el par, τ.

Con el modelo resultante, se obtuvieron las matrices A y B, en tanto que C y D son matrices que se proponen por conveniencia.

En este caso:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \left.\frac{\partial \ddot{\theta}}{\partial \theta}\right|_{P.eq} & \left.\frac{\partial \ddot{\theta}}{\partial \dot{\theta}}\right|_{P.eq} & \left.\frac{\partial \ddot{\theta}}{\partial \psi}\right|_{P.eq} & \left.\frac{\partial \ddot{\theta}}{\partial \dot{\psi}}\right|_{P.eq} \\ 0 & 0 & 0 & 1 \\ \left.\frac{\partial \ddot{\psi}}{\partial \theta}\right|_{P.eq} & \left.\frac{\partial \ddot{\psi}}{\partial \dot{\theta}}\right|_{P.eq} & \left.\frac{\partial \ddot{\psi}}{\partial \psi}\right|_{P.eq} & \left.\frac{\partial \ddot{\psi}}{\partial \dot{\psi}}\right|_{P.eq} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \left.\frac{\partial \ddot{\theta}}{\partial \tau}\right|_{P.eq} \\ 0 \\ \left.\frac{\partial \ddot{\psi}}{\partial \tau}\right|_{P.eq} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad D = [0]$$

Con las ecuaciones establecidas, fue posible implementar técnicas de control convencional para estabilizar al robot.

Aunque en el modelo matemático obtenido se tiene el par del motor como entrada de control al sistema, en la construcción del robot se emplearon motores de pulsos cuyo par es difícil de controlar dada la naturaleza de su operación. Es por ello que se buscaron distintas alternativas para realizar el control. En la universidad de Tohoku Gakuin se propuso utilizar un método basado en la aceleración en lugar del par [3], el cual fue empleado en el trabajo anterior y que también es el empleado en el presente trabajo.

Básicamente el método consiste en manipular las funciones de transferencia de forma que el par sea sustituido por la aceleración como entrada de control al sistema. Con esta modificación es posible controlar efectivamente los motores de pulsos para estabilizar al robot.

Con base en la representación en variables de estado y dado que el modelo simplificado del sistema en el plano es de cuarto orden, fue posible obtener dos funciones de transferencia de segundo orden cada una:

$$G(s) = C(sI - A)^{-1}B + D = \begin{bmatrix} G_1(s) \\ G_2(s) \end{bmatrix}$$

Donde $G_1(s)$ y $G_2(s)$ expresan la relación que existe entre la entrada al sistema, en este caso el par, y las salidas seleccionadas en la matriz C de la representación en variables de estado.

Por lo tanto:

$$G_1(s) = \Theta(\mathrm{s}) = \frac{\theta(s)}{\tau(s)}$$

$$G_2(s) = \Psi(\mathrm{s}) = \frac{\psi(s)}{\tau(s)}$$

Ahora es posible manipular algebraicamente las funciones de transferencia con el fin de prescindir del uso del par como entrada al sistema. Al dividir las funciones de transferencia, se obtuvo que:

$$\frac{\Theta(\mathrm{s})}{\Psi(\mathrm{s})} = \frac{\frac{\theta(s)}{\tau(s)}}{\frac{\psi(s)}{\tau(s)}} = \frac{\theta(s)}{\psi(s)}$$

La función de transferencia obtenida es una relación entre el ángulo existente entre el eje del robot y el eje vertical (salida) y el ángulo que se desplaza la esfera respecto del eje vertical del robot (entrada).

En el domino de Laplace, la operación derivada se expresa como la multiplicación por un término s, esto es:

$$\mathcal{L}\left\{\frac{d}{dt}\right\} = s$$

Entonces, es posible operar el ángulo de entrada como:

$$\ddot{\psi} = s^2\psi(s)$$

Luego entonces, la función de transferencia buscada es:

$$T(s) = \frac{\theta(s)}{s^2\psi(s)} = -\frac{I_e + r_e(m_p d + r_e(m_e + m_p))}{-m_p g d + (I_p + I_e + d^2 m_p + 2 d m_p r_e + {r_e}^2(m_e + m_p))s^2}$$

la cual relaciona la posición del robot con respecto al eje vertical y la aceleración angular de la esfera.

Los valores de los parámetros del robot se muestran en la Tabla 5.1.

TABLA 5.1 Valores de los parámetros del robot.

Parámetros del modelo		
Símbolo	***Valor***	***Unidades***
m_p	8	kg
I_p	0.08	kg·m^2
m_e	0.6	kg
I_e	0.005	kg·m^2
d	0.21	m
r_e	0.125	m

Luego de considerar el valor de la aceleración de la gravedad como g = 9.78 m/s^2 y sustituir los valores en la expresión matemática anterior, se obtuvo la función de transferencia del robot:

$$T(s) = \frac{0.349375}{16.4304 - 0.992175s^2}$$

Con base en esta función de transferencia, se diseñaron distintos tipos de controladores que posteriormente serán implementados.

Para analizar mejor los controladores diseñados, se utilizó el software MATLAB; más específicamente, las simulaciones realizadas se hicieron empleando la herramienta Simulink.

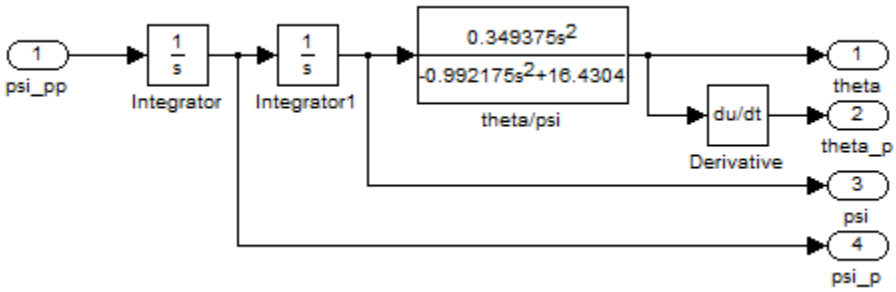


Figura 5.2 Modelo del sistema realizado en Simulink de MATLAB.

Para realizar el análisis y la simulación de los diferentes controladores, se construyó un modelo del sistema tal como el que se muestra en la Figura 5.2. La entrada al sistema es la aceleración angular de la esfera, en tanto la salida es el ángulo entre el cuerpo del robot y la vertical. Se añadieron también los otros tres estados del sistema para análisis adicionales.

El diseño completo de los controladores que se presentan en este trabajo puede consultarse en el apéndice A.8.

5.3.1 Control proporcional (P)

Como es sabido un control proporcional se emplea principalmente para estabilizar un sistema inestable. Dado que el sistema a controlar tiene dicha característica, se procedió a diseñar un control proporcional y evaluar la factibilidad de su uso.

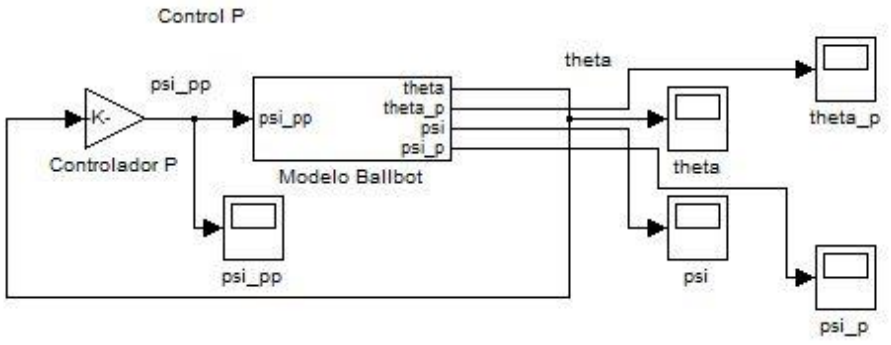


Figura 5.3 Modelo de Simulink para el controlador P.

En la Figura 5.3 se observa el esquema de Simulink para el control propuesto.

Para que el sistema sea estable:

$$Kp \geq 47.028$$

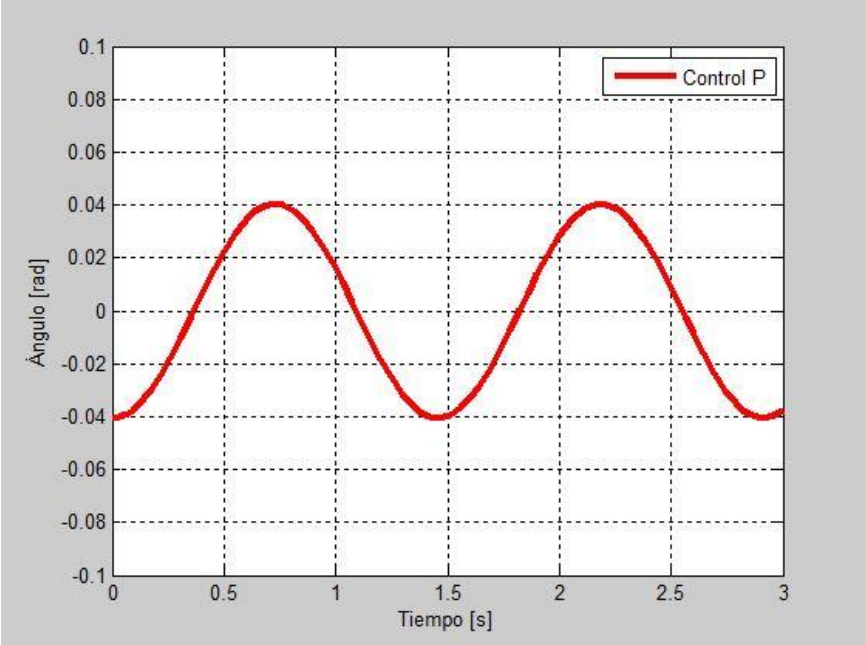


Figura 5.4 Respuesta del controlador P en simulación.

En la Figura 5.4 se puede observar la respuesta del sistema en lazo cerrado para una constante $Kp = 100$. Como se puede apreciar, el sistema con control proporcional es estabilizado marginalmente, oscilando entre -2 y 2 grados aproximadamente.

5.3.2 Control proporcional-derivativo (PD)

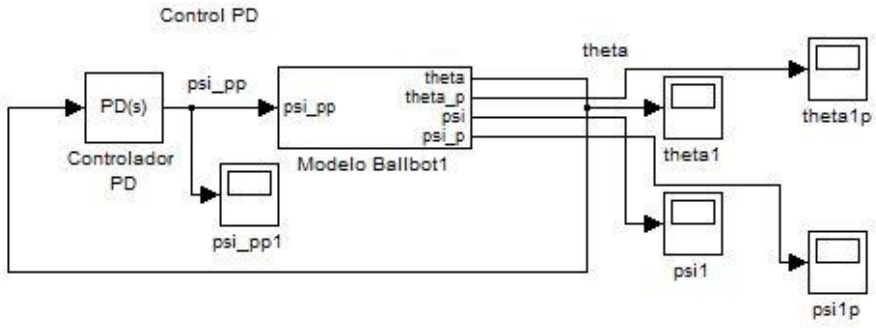


Figura 5.5 Modelo de Simulink para el controlador PD.

De forma similar al control proporcional, se muestra en la Figura 5.5 un esquema de Simulink con el control proporcional-derivativo. Se eligieron como parámetros de diseño un tiempo de asentamiento $ts = 0.5\ s$ y un

porcentaje de sobrepaso $\%sp = 20\%$. Con ello se buscó que la respuesta del sistema tuviera un sobrepaso pequeño y que alcanzara la estabilidad cuando más en 500 ms.

Considerando los parámetros de diseño, se obtuvieron los siguientes valores de constantes:

$$Kp = 1413.06$$

$$Kd = 56.797$$

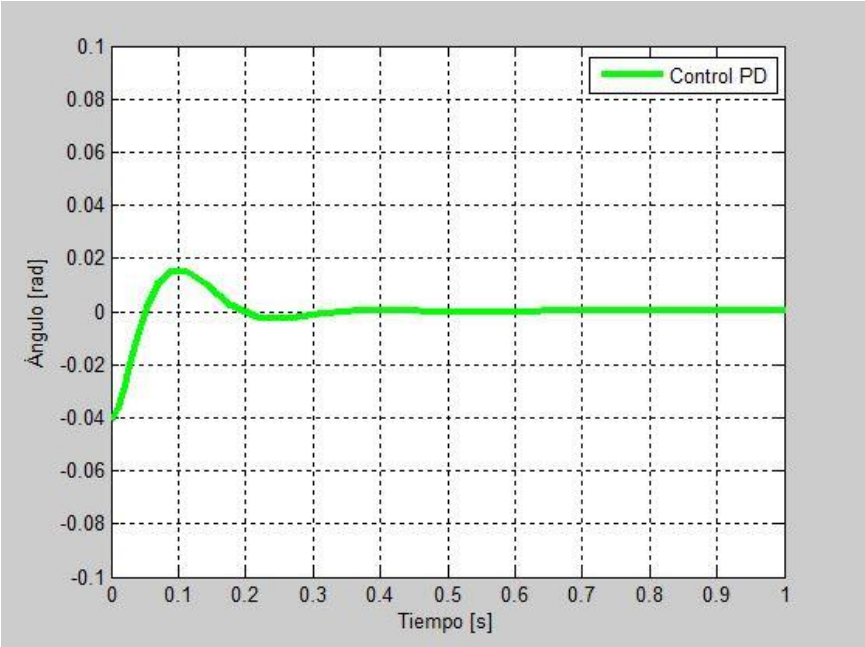


Figura 5.6 Respuesta del controlador PD en simulación.

Se observa en la Figura 5.6 que la respuesta del sistema en simulación cumplió con los parámetros de diseño.

5.3.3 Control por realimentación de estados

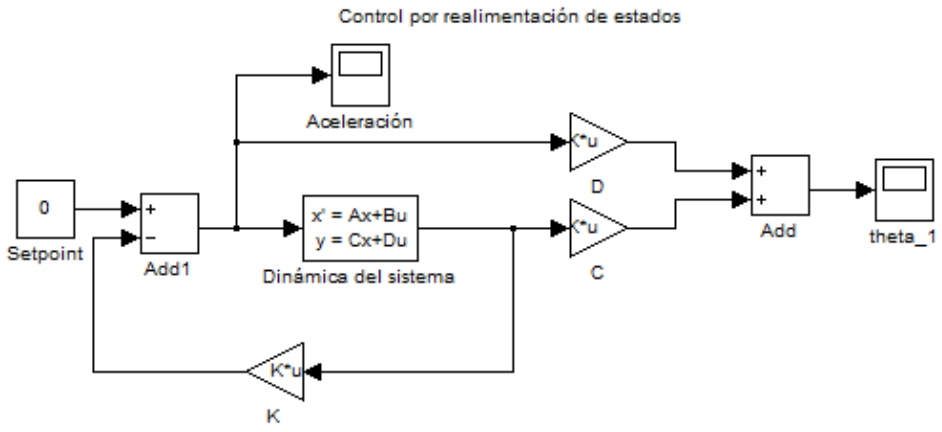


Figura 5.7 Modelo de Simulink para el controlador por realimentación de estados

Con base en un proceso similar al empleado en la manipulación de las funciones para establecer la aceleración como entrada al sistema en lugar de par, se obtuvo una representación de estados del sistema empleando el comando "tf2ss" en MATLAB y un modelo de Simulink como el que se muestra en la Figura 5.7. Posteriormente, con el empleo del método de asignación de polos con los parámetros de diseño propuestos anteriormente, se obtuvo la matriz de constantes K del algoritmo de control:

$$K = [20 \quad 497.58]$$

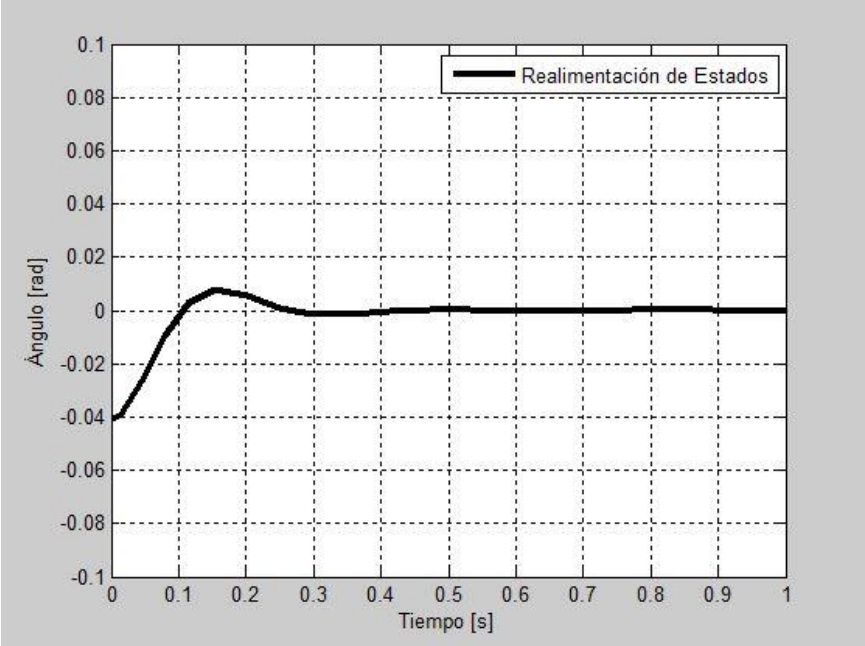


Figura 5.8 Respuesta de la simulación del controlador por realimentación de estados.

Se observa en la Figura 5.8, que la respuesta del controlador diseñado cumple con las especificaciones de diseño propuestas.

5.3.4 Control difuso

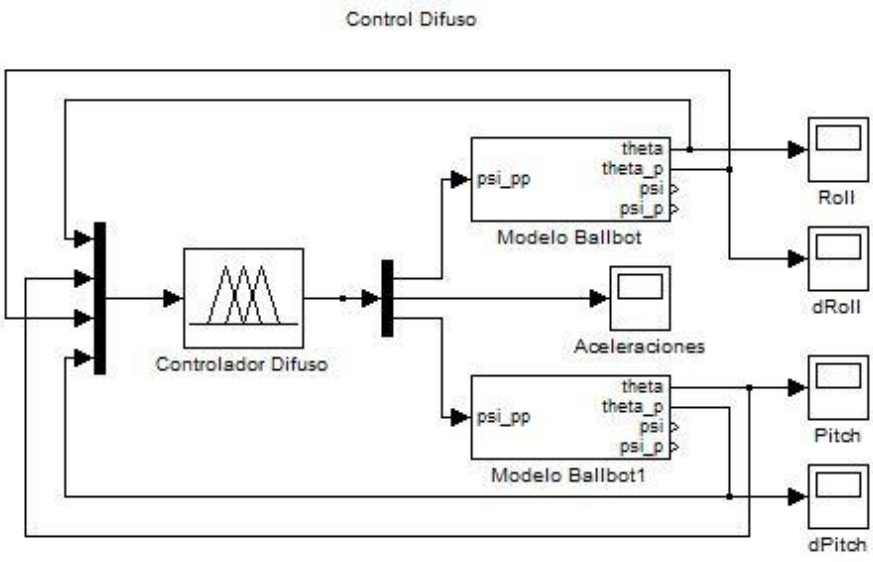


Figura 5.9 Modelo del controlador difuso.

Se optó también por diseñar un controlador difuso para estabilizar al robot, cuyo modelo se observa en la Figura 5.9. Dado que un controlador difuso

puede ser de tipo MIMO, o Multiple-Input Multiple-Output, por sus siglas en inglés, es posible diseñar un solo controlador para todo el robot, esto es, para ambos planos de actuación del sistema simplificado propuesto.

Como se observa en la Figura 5.10, el diseño se realizó con la caja de herramientas, o *toolbox*, para sistemas difusos implementado en MATLAB. Se eligieron cuatro diferentes conjuntos difusos, uno para cada una de las variables de estado del sistema. Los conjuntos difusos establecidos contienen cinco funciones de pertenencia cada uno. El conjunto de reglas de la máquina de inferencia se compone de 30 elementos. Los conjuntos de salida son las aceleraciones demandadas por el sistema en cada uno de los planos y, al igual que los conjuntos de entrada, se componen de cinco funciones de pertenencia cada uno.

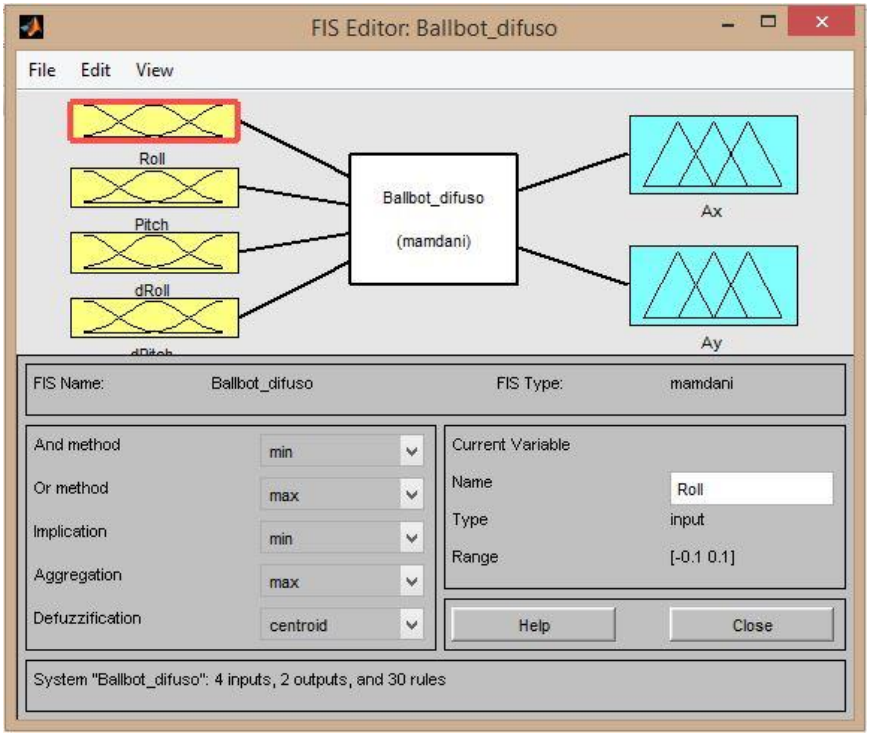


Figura 5.10 Caja de herramientas de sistemas difusos en MATLAB.

Al ser un controlador que no se basa en el modelo matemático del sistema, no se requiere ningún tipo de manipulación de las ecuaciones que modelan al sistema, por ende no hay constantes que calcular.

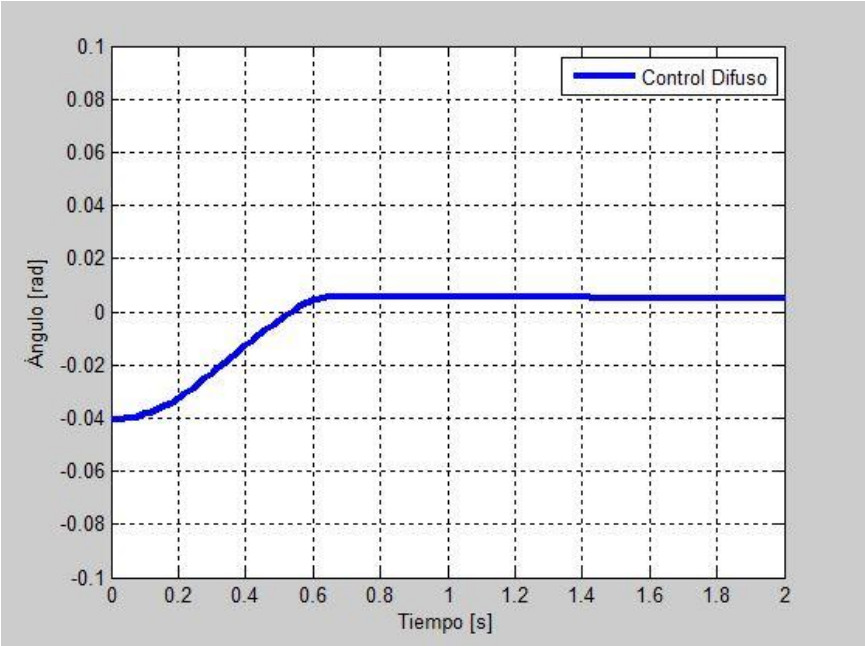


Figura 5.11 Respuesta de la simulación del controlador difuso.

Se observa en la Figura 5.11 que el control estabiliza al robot pero no consigue llevarlo al punto de equilibrio deseado, por lo cual no cumple las especificaciones deseadas.

5.3.5 Análisis de resultados

Al realizar el análisis comparativo de las estrategias de control propuestas, se obtuvieron los resultados que se comentan a continuación.

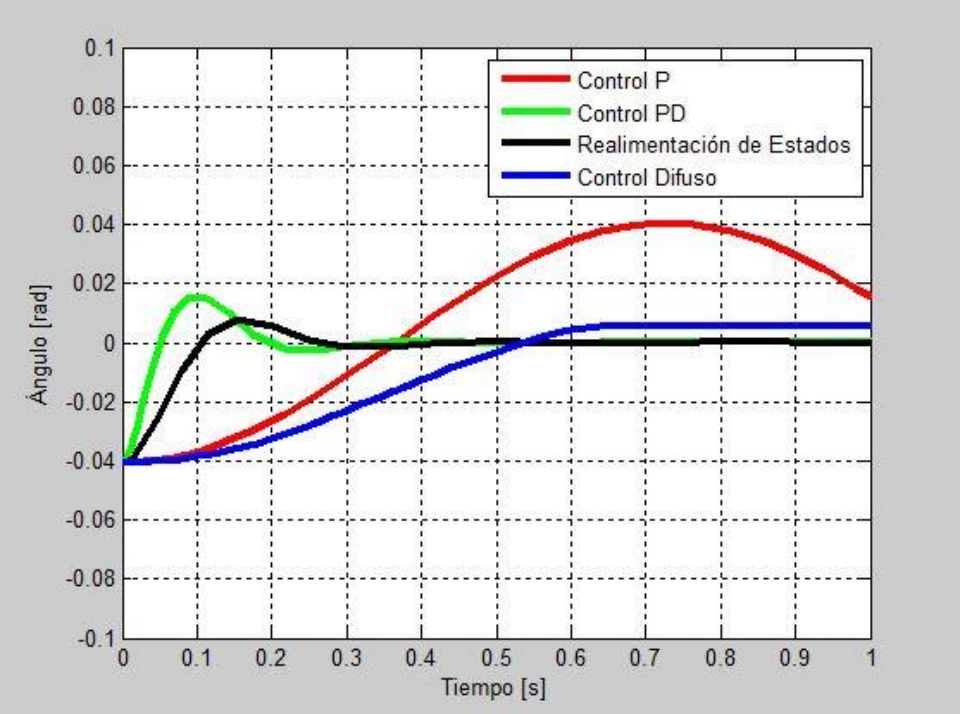


Figura 5.12 Comparación de las respuestas de los controladores.

En la Figura 5.12 se observa el desempeño de la simulación de los cuatro controles diseñados para el sistema. Es claro que el que mejores cualidades presentó es el control Proporcional-Derivativo, pues estabilizó al robot antes de los 500 ms propuestos como objetivo de diseño; adicionalmente, el sobrepaso fue menor al 20%. Por otra parte, el control por realimentación de estados tuvo un menor sobrepaso que el anterior, sin embargo, el tiempo de asentamiento fue un poco mayor. Adicionalmente se observó que el control Proporcional estabilizó marginalmente al sistema, lo que implica que el robot estará oscilando cerca del punto de equilibrio periódicamente, además de que no es posible controlar el desempeño del sistema, por lo cual no es conveniente emplearlo. Por último, el control Difuso no estabilizó al robot en el punto de equilibrio deseado, por lo cual no cumple con el objetivo primordial; esto no implica que dicho controlador sea malo o no deba usarse, sino que es un indicio de que se requiere mayor experiencia en el funcionamiento del sistema para poder diseñar unos mejores conjuntos difusos y con ello proponer un mejor conjunto de reglas.

Con base en lo que se discutió anteriormente, se decidió continuar la estrategia utilizada por el proyecto precedente, utilizando un control PD para alcanzar la estabilidad del robot.

5.4 Implementación

El prototipo anterior empleó el mismo módulo de procesamiento de la unidad de mediciones inerciales para obtener las señales de control que se enviaron a los motores para luego comunicarse con un microcontrolador ATmega328. Dado que en el presente trabajo se empleó un sensor diferente, se tuvieron que hacer modificaciones con respecto al hardware empleado.

5.4.1 Sistema de sensado

Para determinar el ángulo de orientación del Ballbot así como su velocidad angular, en el primer prototipo se utilizó la unidad de mediciones inerciales Ultimate IMU junto con el algoritmo propuesto en la página electrónica de Sparkfun. Se concluyó que el desempeño de la IMU que se utilizó no fue el adecuado para una aplicación de este tipo, dadas las condiciones de velocidad de obtención de datos logradas, que resultaron ser demasiado lentas para poder controlar al robot.

Posteriormente, los integrantes del equipo involucrado en el desarrollo del primer prototipo realizaron una evaluación comparativa de diferentes sensores de estas características [7], y llegaron a la conclusión de que el sensor denominado Sensor Stick en su versión 10724 de la página de Sparkfun es el más adecuado para la aplicación debido a que se tenía en existencia y soporta un algoritmo de obtención de posición que está muy bien documentado y tiene soporte continuo.

Figura 5.13 IMU Sensor Stick.

Dicho sensor inercial que se muestra en la Figura 5.13; tiene embebidos un acelerómetro, un giroscopio y un magnetómetro, cada uno con tres ejes, por

lo cual es posible conocer la orientación y la velocidad del robot en todo momento.

Se decidió entonces emplear este sensor utilizando el algoritmo recomendado por la página del fabricante. Dicho algoritmo es un código que se implementa en la plataforma de desarrollo Arduino y que puede ser portable a otras plataformas, debido a que está escrito en lenguaje C. Para estimar el ángulo se utiliza el algoritmo DCM, o Matriz de Cosenos Directores, por sus siglas en inglés, con lo cual se obtuvo un buen desempeño en las estimaciones. El algoritmo está hecho para funcionar con una frecuencia de muestreo de 50 Hz y es posible obtener los tres ángulos de Euler así como las velocidades angulares del dispositivo en todo momento.

Ángulos de Euler

Para conocer la orientación de un objeto en el espacio es necesario conocer al menos tres ángulos que describan la rotación del eje de referencia móvil del sistema con respecto a un eje de referencia estático. Estos ángulos reciben el nombre de ángulos de Euler y, en terminología náutica, corresponden a *roll* o alabeo, *pitch* o cabeceo y *yaw* o guiño.

El alabeo corresponde al ángulo de rotación sobre el eje x del sistema de referencia global, el cabeceo corresponde al ángulo de rotación sobre el eje y del sistema de referencia global y finalmente el guiño corresponde al ángulo de giro sobre el eje z.

Particularmente para el caso del Ballbot, los ángulos empleados para controlar al robot fueron el alabeo y el cabeceo, debido a la ubicación del sistema de referencia de la IMU y del robot.

5.4.2 Sistema de procesamiento

Como una primera aproximación se decidió emplear el microcontrolador ATmega328 que emplea la tarjeta de desarrollo Arduino en su versión Uno. Se observó que la memoria disponible restante no fue suficiente para implementar junto con el sistema de sensado el controlador, por lo cual se empleó posteriormente un Arduino Due. Finalmente, después de diversas pruebas que se detallaron en el Capítulo 4, se llegó a la conclusión de que era conveniente emplear dos microcontroladores comunicados por I^2C.

5.4.3 Implementación del controlador

Las ecuaciones del control PD en función del tiempo son:

$$u(t) = \ddot{\psi} = Kp * \theta + Kd * \dot{\theta}$$

Donde $\ddot{\psi}$ es la aceleración angular de la esfera, θ es el ángulo del robot con la vertical, *Kp* es la constante proporcional y *Kd* es la constante derivativa.

Previamente, se estableció que se modelaría al robot en un par de planos ortogonales entre sí, por lo cual el control diseñado sólo controló al robot en un plano. Por consiguiente, las ecuaciones de la aceleración angular en cada uno de los planos verticales con base en los ángulos de Euler son:

$$\ddot{\psi}_{alabeo} = Kp * \theta_{alabeo} + Kd * \dot{\theta}_{alabeo}$$

$$\ddot{\psi}_{cabeceo} = Kp * \theta_{cabeceo} + Kd * \dot{\theta}_{cabeceo}$$

Dado que los parámetros del robot fueron muy semejantes en cada uno de los planos, se consideró que los valores de *Kp* y *Kd* son válidos para ambos casos. Luego de un proceso de sintonización de constantes, los valores que mejor funcionaron en el robot fueron:

$$Kp = 800, \;\; Kd = 200$$

Físicamente, el algoritmo de control se implementó en una tarjeta Arduino Uno que se comunicó por I^2C con una tarjeta Arduino Due que controla los motores, como se describió previamente en el Capítulo 4.

Dado que se pudieron obtener datos a una frecuencia de 143 Hz, se consideró pertinente utilizar directamente las ecuaciones obtenidas en función del tiempo, evitando con esto el proceso de discretización del sistema.

El módulo de control de los motores recibe una velocidad lineal por cada plano para mover al robot, es decir, un total de dos velocidades para controlar al sistema completo. Luego entonces fue necesario obtener las componentes cartesianas de velocidad de la acción de control para enviarlas al módulo de control de los motores.

Para ello, se empleó la aceleración angular obtenida en cada uno de los planos verticales para determinar las componentes de la aceleración tangencial requerida.

$$A_y = (Kp * \theta_{alabeo} + Kd * \dot{\theta}_{alabeo})/r_{esfera}$$

$$A_x = (Kp * \theta_{cabeceo} + Kd * \dot{\theta}_{cabeceo})/r_{esfera}$$

Posteriormente, dado que el ciclo de actualización del algoritmo de control es de 7 ms, fue posible integrar la aceleración para obtener las componentes del vector velocidad. Una consideración importante es que, dado el uso del *microstepping*, el par máximo que pueden producir los motores sólo permitió estabilizar al robot dentro de un intervalo muy cercano al punto de equilibrio, esto es alrededor de 5°. Debido a lo anterior, se puede considerar que la aceleración deseada es aproximadamente la aceleración instantánea absoluta, por lo que el algoritmo implementado resultó ser el siguiente:

$$V_y = A_y * \Delta t$$

$$V_x = A_x * \Delta t$$

Donde Δt = 0.007 s.

El diagrama de flujo del algoritmo empleado en la tarjeta Arduino se muestra en la Figura 5.14.

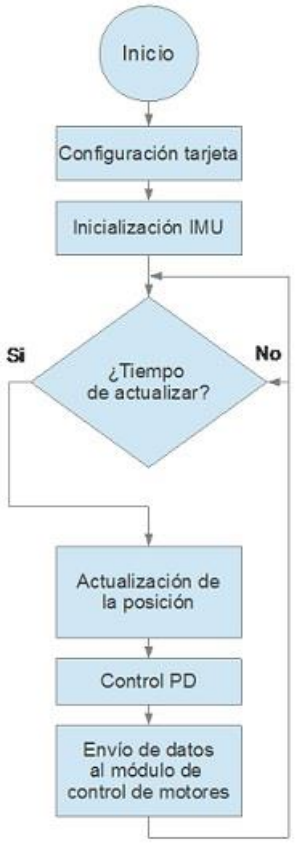


Figura 5.14 Diagrama de flujo del módulo de control y sensado.

Finalmente en la Figura 5.15 se observa un esquema general de la implementación del control en el robot.

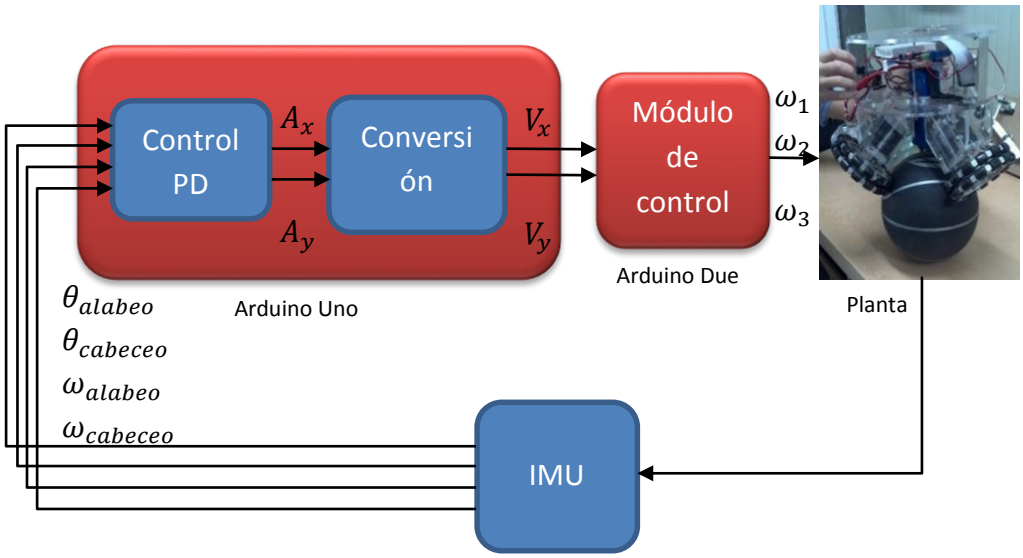


Figura 5.15 Implementación del control en el Ballbot.

5.4.3 Filtro pasa bajas

A pesar de las buenas características de funcionamiento de la IMU, una vez que se enlazaron el módulo de control implementado en el Arduino Uno y el módulo de control de los motores implementado en el Arduino Due, se observó una señal con ruido en las mediciones obtenidas del sensor. Dado que las señales obtenidas de la IMU eran los valores de las variables de estado del sistema y que de ellos dependía el control de estabilidad del robot, fue crucial atacar el problema con un filtro.

La implementación de un filtro analógico no era posible debido a que el sensor es completamente digital, por lo que no fue posible acceder a una señal analógica. Por consiguiente, se diseñó un filtro digital para implementarlo en el mismo microcontrolador que recibía los datos de orientación y calculaba la acción de control.

Al observar la naturaleza de las señales sin filtrar, se observó que la frecuencia del ruido oscilaba entre 2 y 3 Hz, por lo cual se diseñaron diversas aproximaciones para abordar el problema.

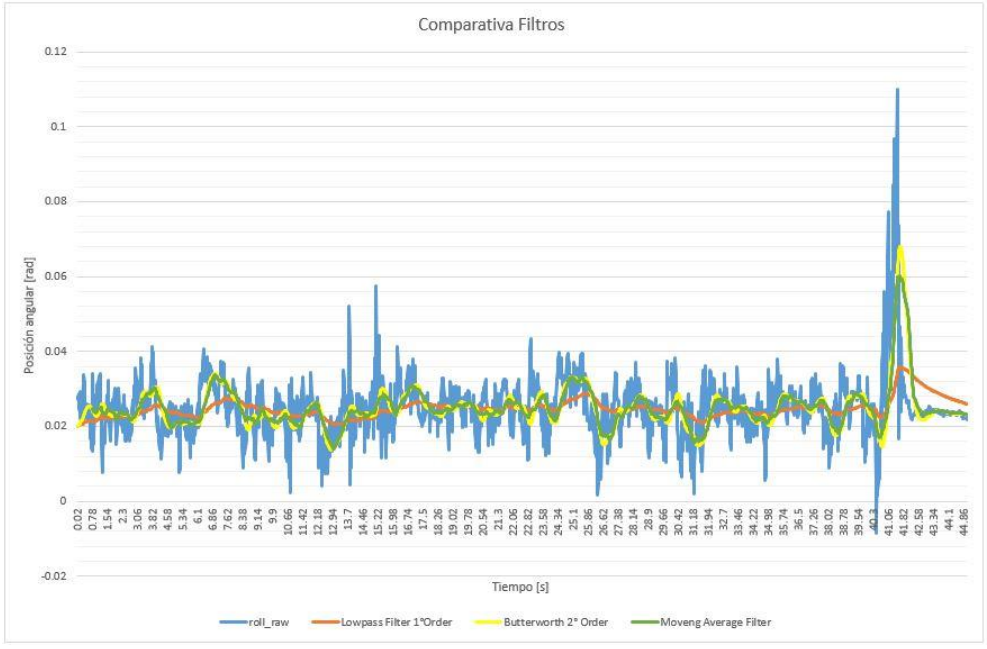


Figura 5.16 Comparación de los diversos algoritmos empleados para el filtro requerido.

En la Figura 5.16 se muestra la respuesta de diversos algoritmos que se probaron para el filtro requerido para la señal de la IMU. De la comparación se puede observar que el filtro media móvil y el de Butterworth de 2° orden

se comportan de manera similar, reduciendo el ruido con muy poco retraso en la señal. El filtro que mejor atenuó la señal resultó ser el filtro pasa bajas de 1° orden, el cual tuvo la gran desventaja de generar un retraso considerable en las mediciones filtradas respecto a la posición instantánea real, que fue función de la magnitud del pico de ruido instantáneo.

Con base en las pruebas de funcionamiento se decidió implementar el filtro pasa bajas de 1° orden con una frecuencia de corte de 2.52 Hz, que redujo de forma considerable el ruido durante la operación.

Finalmente, luego de incluir este último filtro en la operación del robot, se observó una mejoría en el control de su estabilización, aunque aun así fue insuficiente para mantener al robot por más tiempo sobre la esfera.

CAPÍTULO 6
PRUEBAS Y RESULTADOS

6.1 Pruebas de movimiento de los motores

Estas pruebas tuvieron el objetivo principal de averiguar si el algoritmo para controlar la velocidad y la aceleración de los motores funcionaba correctamente. Asimismo, como segundo objetivo, se realizaron pruebas de aceleración para conocer el valor máximo al que el Ballbot puede operar en situaciones similares a las de que se presentaran durante su funcionamiento.

Las pruebas de velocidad fueron las primeras en realizarse, las cuales consistieron en solicitar que los motores paso a paso adquirieran una velocidad estable y cotejar la velocidad obtenida con la solicitada por medio de un programa de cómputo. Para realizar esta comparación, se realizó una grabación de video a 120 cuadros por segundo y se contrastó el tiempo que tardó en volver a la misma posición un punto específico de la rueda. Para realizar este análisis, se reprodujo el video a una cuarta parte de su velocidad, 30 cuadros por segundo, por lo que la velocidad que aparece de color amarillo no es la velocidad en tiempo real del video, sino el tiempo de la reproducción, el cual es 4 veces más.

Se programó que cada uno de los motores girara a distintas velocidades, a un décimo, media y una revolución por segundo. A cada rueda se le marco un punto de color rojo en su parte externa. Después, se grabó cada rueda por un periodo suficiente para que realizara más de tres revoluciones. Este

vídeo se analizó en un programa que permite ver cuadro por cuadro y se verificó el tiempo en que el punto marcado regresa a la posición inicial.

Las imágenes más representativas de las pruebas realizadas se muestran a continuación junto con el análisis de su error.

Figura 6.1 Prueba de un décimo de revolución por segundo.

En la Figura 6.1 se muestra el tiempo inicial, 51.10 segundos y el tiempo final, 1:31.11 segundos, que se necesitó para que la rueda realizara una revolución completa. En esta prueba se programó al controlador para que el motor se moviera a un décimo de revolución por segundo, por lo que para completar una revolución se necesitan diez segundos en tiempo real y cuarenta segundos en la reproducción a 30 cuadros por segundo. Al realizar el cálculo de la velocidad real de la llanta por medio de los tiempos

mostrados en el análisis del vídeo, se verificó que el error de velocidad en esta prueba fue del 0.025% .

Figura 6.2 Prueba de media revolución por segundo.

La Figura 6.2 muestra la prueba de movimiento a media revolución por segundo, luego de que la rueda dio cuatro revoluciones completas; esta prueba dio como resultado, después de realizar un análisis similar al comentado anteriormente, que a esta velocidad el programa tuvo un error del 0.5%.

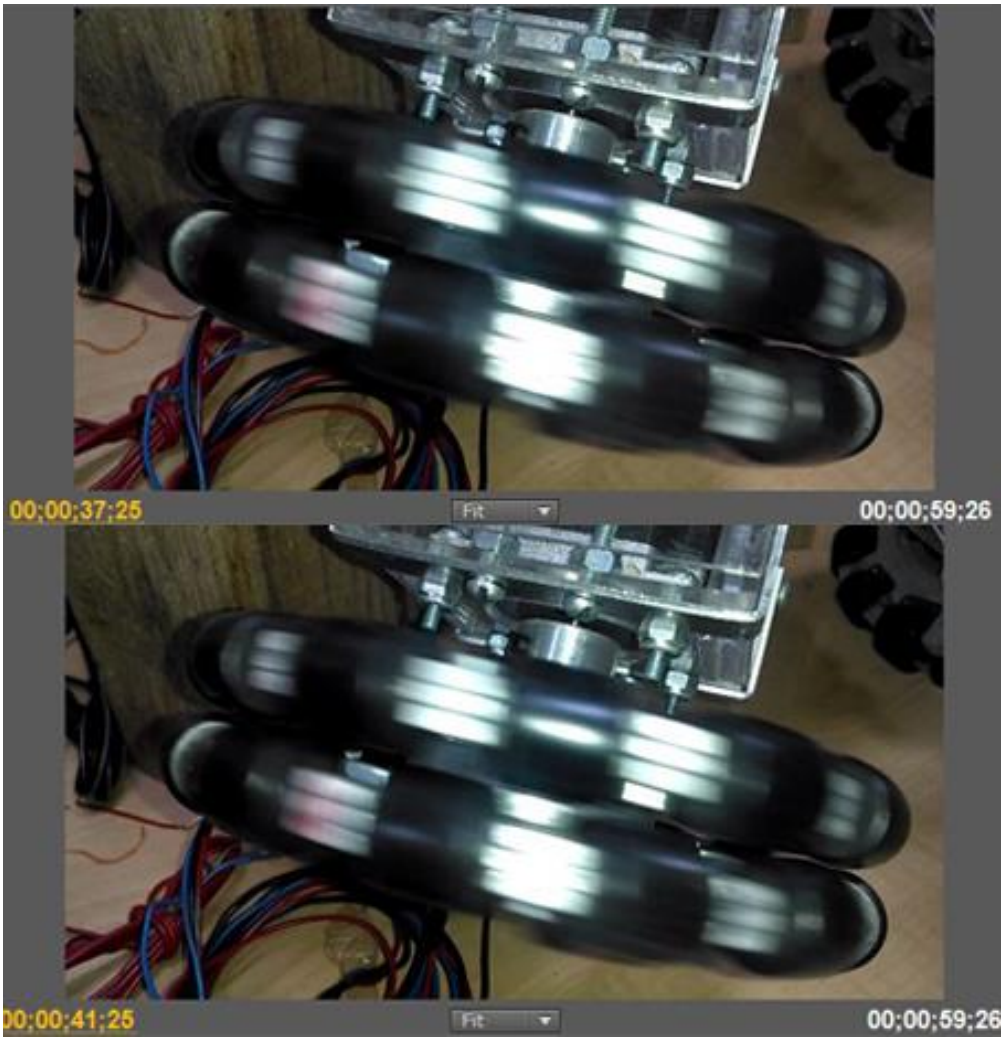


Figura 6.3 Prueba de una revolución por segundo.

El punto marcado en la rueda para la prueba de una revolución por segundo no se visualizó tan claro como en las pruebas anteriores, debido a su mayor velocidad, sin embargo, fue posible detectarlo, tal como se puede ver en la Figura 6.3, por lo que fue posible determinar la velocidad real del motor paso a paso. El análisis de este experimento arrojó que no existía error medible entre la velocidad deseada y la velocidad real.

Después de realizar las pruebas de velocidad, se continuó con la ejecución de las pruebas de aceleración. Estas pruebas buscaron someter a los motores a cambios de velocidad similares a los que se necesitarían para mantenerlo equilibrado sobre la esfera.

Se decidió realizar pruebas de aceleración desde el reposo, tanto en una superficie horizontal como en un plano inclinado, desde una velocidad inicial

baja y tanto en el mismo sentido como en sentido contrario a la aceleración. Estas pruebas fueron importantes debido a que son situaciones que se presentaran durante el proceso de estabilización del Ballbot, y el conocer la magnitud de la aceleración máxima resulta de vital importancia para determinar los puntos donde el sistema no será capaz de responder correctamente. Con lo anterior en mente, las pruebas de aceleración en plano inclinado se realizaron con un ángulo de 5° respecto de la horizontal; esta inclinación es representativa del que el Ballbot será capaz de recuperarse durante el proceso de estabilización. Finalmente, la metodología para determinar la máxima aceleración posible del robot fue iniciar con una aceleración baja e incrementar paulatinamente esta aceleración hasta el punto de falla. La aceleración máxima en cada prueba será la última que el Ballbot sea capaz de moverse sin un fallo considerable. La velocidad máxima que se le permitió al robot fue de 70 cm por segundo.

TABLA 6.1 Resultados de las pruebas de aceleración.

	Plano horizontal	**Plano inclinado (5°)**	**Unidades**
Estático	33	25	$\frac{cm}{s^2}$
Velocidad positiva	37	30	$\frac{cm}{s^2}$
Velocidad negativa	36	11	$\frac{cm}{s^2}$

En la Tabla 6.1 se pueden revisar los resultados obtenidos en las pruebas de aceleración. El término velocidad positiva se refiere a que el robot se está moviendo en el mismo sentido en el que se aplica la aceleración, y velocidad negativa cuando la aceleración se aplica en sentido contrario. Tomando como referencias las pruebas en el plano horizontal, se puede observar que la fricción estática que actúa en las ruedas del Ballbot en reposo disminuye considerablemente su capacidad de acelerar. Asimismo, se nota en la prueba de plano inclinado que la aceleración, cuando se tiene una velocidad inicial contraria, se reduce al 30% de la aceleración máxima que el Ballbot puede realizar.

6.2 Pruebas de movimiento en un plano horizontal

Una vez que se verificó el funcionamiento del algoritmo de control de la velocidad y aceleración de los motores, se procedió a realizar pruebas de movimiento del robot en un plano horizontal. Con estas pruebas se pudo verificar que el Ballbot era capaz de recibir como entrada los componentes de la velocidad deseada y generar el movimiento necesario de cada motor para lograr la trayectoria programada. Las ecuaciones utilizadas para el movimiento del robot fueron similares a las ecuaciones para mover otro tipo de robots omnidireccionales con los actuadores colocados a 120° uno del otro. El desarrollo de estas ecuaciones se puede encontrar en el Apéndice A.1.

Las pruebas que se realizaron consistieron en mover al Ballbot en trayectorias que describieran dos líneas rectas a 90°, 45° y 135°, además de realizar un recorrido que forma un cuadrado. Estas pruebas se realizaron a una velocidad constante.

Debido a que se necesitaba analizar la trayectoria realizada, inicialmente se acopló un marcador de textos a una de las bases de los motores, con el fin de trazar la trayectoria realizada sobre una cartulina.

Las Figuras 6.4, 6.5, 6.6, muestran imágenes de las pruebas efectuadas, en las cuales se puede observar al Ballbot realizando las trayectorias deseadas y a un lado la medición del ángulo mediante el uso de un transportador. En orden de aparición, los recorridos mostrados en las figuras son: 45°, 90° y 135°

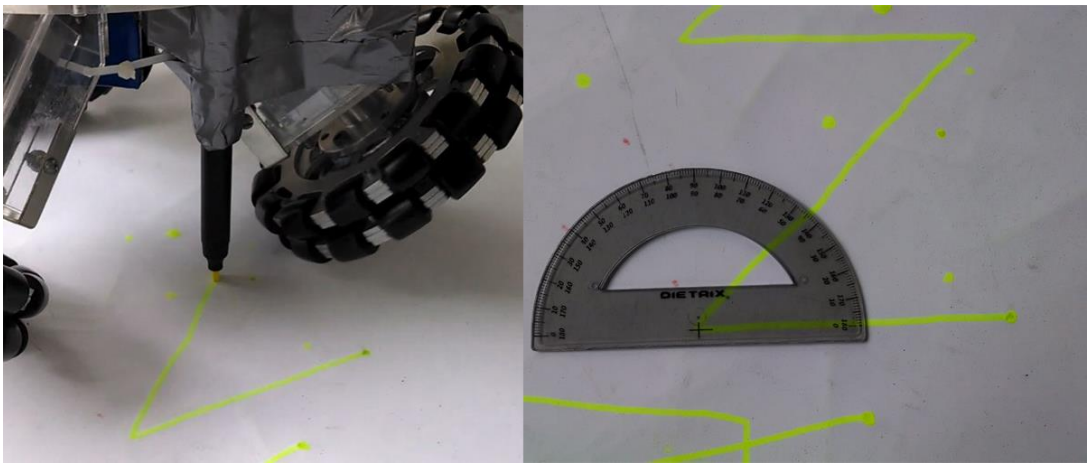


Figura 6.4 Trayectoria de 45°.

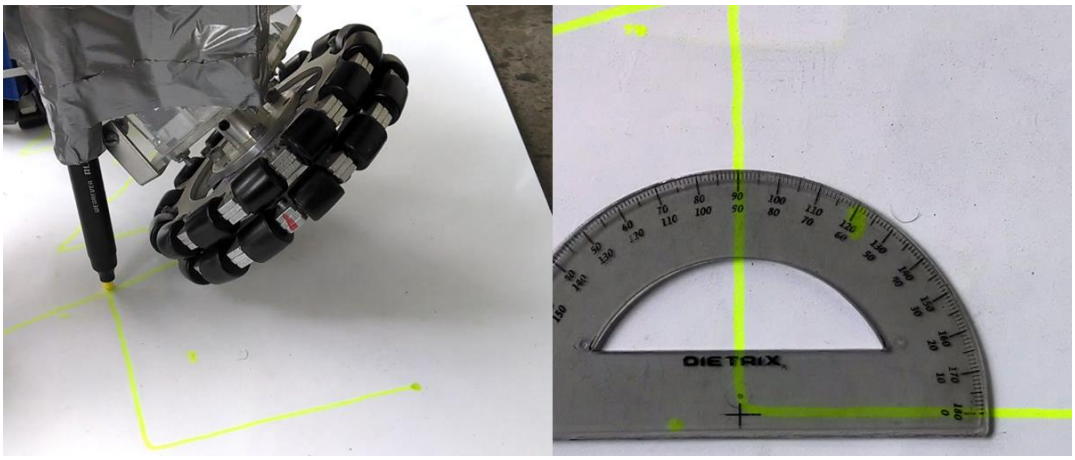


Figura 6.5 Trayectoria de 90°.

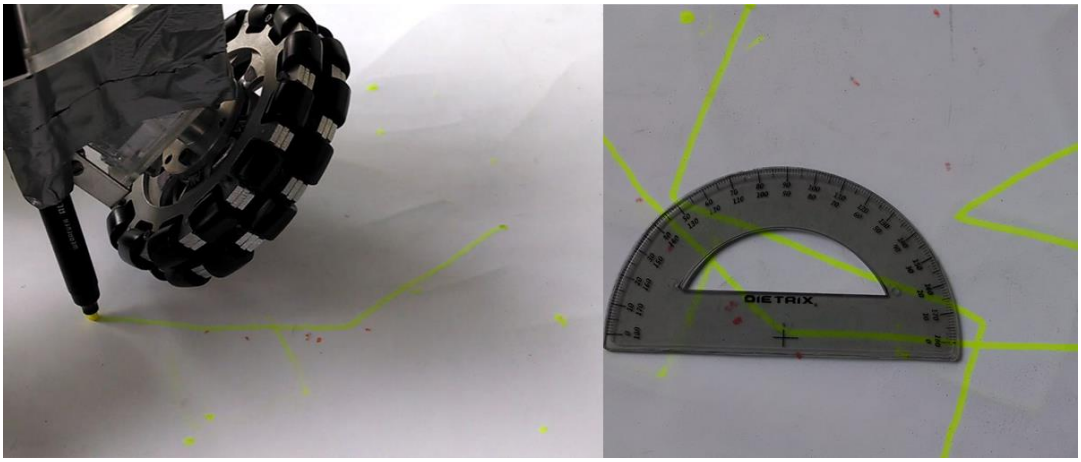
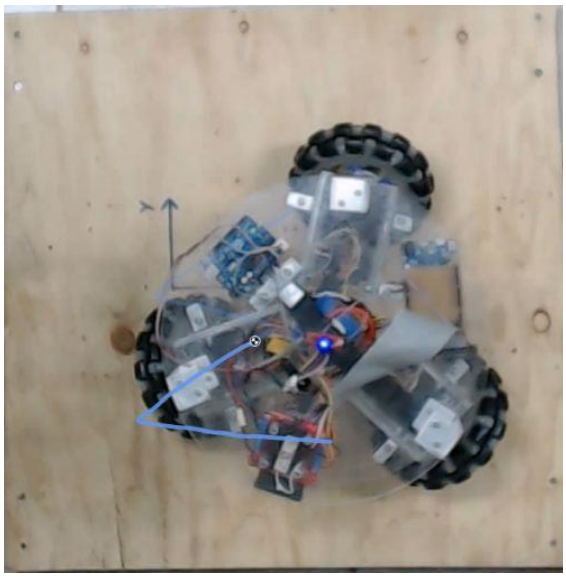


Figura 6.6 Trayectoria de 135°.

Posteriormente se decidió utilizar un método para analizar las trayectorias realizadas sin incluir algún elemento que afectara su desempeño, ya que en las pruebas anteriores el marcador modificaba los recorridos realizados. Por esta razón se volvieron a realizar algunas pruebas, pero ahora grabando en vídeo desde arriba las trayectorias y analizándolas con un programa dedicado a ello. El resultado de las pruebas se puede ver en las Figuras 6.7 y 6.8, en las que se muestran las trayectorias de 45° y la cuadrada, respectivamente.



Figuras 6.7 45° respectivamente.

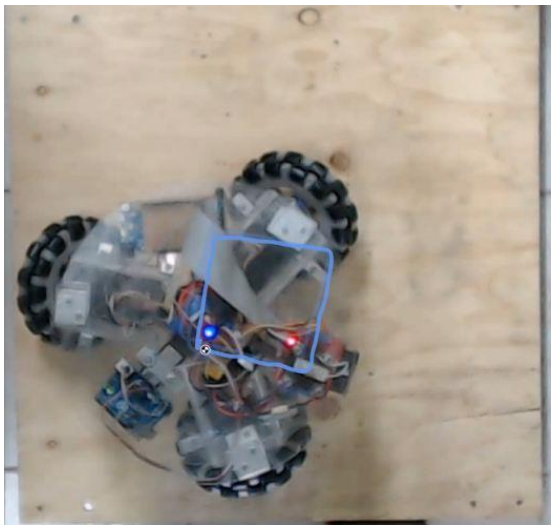


Figura 6.8 Trayectoria cuadrada.

Al analizar las pruebas de movimiento del Ballbot en un plano horizontal, se observó que las trayectorias que realiza se aproximan bastante a las deseadas. Lo anterior indica que las ecuaciones de movimiento utilizadas en el control de velocidad son correctas. Se cree que el hecho de que las trayectorias no sean perfectas se debe a diversos factores, como los que se comentan a continuación:

- No se utilizaron perfiles de velocidad, por lo que los cambios de dirección se podrían ver afectados por la inercia del robot

- El control de los motores se realizó en serie, lo cual no puede garantizar un control en tiempo real del movimiento de las ruedas
- Existen imperfecciones geométricas en las ruedas que podrían modificar la trayectoria del robot.

6.3 Pruebas de lectura de la IMU

En paralelo a las pruebas de locomoción del Ballbot, se realizaron pruebas para conocer la confiabilidad de las lecturas de la IMU y su desempeño. Se realizaron pruebas para medir el tiempo de retraso en la obtención de la posición angular ante una perturbación en tiempo real, con y sin filtro, la caracterización de la incertidumbre en estado estático, la caracterización del ruido provocado por las vibraciones de los motores paso a paso y la modificación de la posición angular debida a las imperfecciones geométricas de las ruedas omnidireccionales. En la mayoría de las pruebas, se compararon las lecturas sin filtrar del sensor contra los datos obtenidos después de la etapa de filtrado.

La primera prueba que se realizó fue la medición del tiempo de retraso de las lecturas del sensor. Para realizar esta prueba, se fijó al sensor perpendicularmente en el eje del motor y se realizaron movimientos oscilatorios con éste. Se comparó el tiempo entre el movimiento real del motor paso a paso y la lectura del sensor, para obtener el tiempo de respuesta. Para esta prueba, se probaron dos algoritmos diferentes: uno, con una frecuencia de muestreo de 50 Hz y el otro, con una frecuencia de 200 Hz.

Las pruebas arrojaron que el algoritmo de la IMU, sin la aplicación de ningún filtro, tiene un retraso de 0.204 segundos en la obtención de la posición angular, a una frecuencia de muestreo de 50 Hz, y un retraso de 0.0068 segundos con la configuración de 200 muestras por segundo. Los resultados del experimento se muestran en la Tabla 6.2, y la configuración física del experimento se puede ver en la Figura 6.9.

TABLA 6.2 Pruebas de respuesta de la IMU.

	200 Hz	**50 Hz**	**Tiempo motor 90°**
Prueba 1	0.81081	1.1615	0.806365
Prueba 2	0.81774	0.9253	0.806365
Prueba 3	0.81081	0.9449	0.806365

Tiempo promedio	0.81312	1.0106	0.806365
Retraso	0.00675	0.2042	

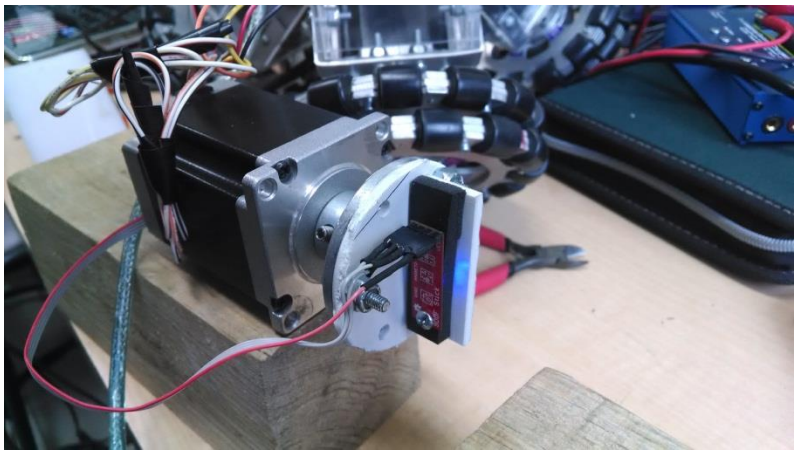


Figura 6.9 Configuración del experimento de respuesta de la IMU.

Una vez obtenido el tiempo de retraso en la medición de la posición angular sin filtro, se realizó una prueba similar pero utilizando el filtro diseñado en el capítulo anterior. Se obtuvo que una vez aplicado el filtro, la obtención de la posición angular tiene una demora de 0.056 segundos. En la Figura 7.11 se muestra una gráfica de la posición angular sin filtrar contra la posición angular filtrada. Cada una de las marcas en el eje X corresponde al tiempo de una muestra, con una frecuencia de muestreo de 143 Hz. En el eje de las ordenadas se muestra el valor de la posición, en grados sexagesimales, y en el de las abscisas el número de muestra.

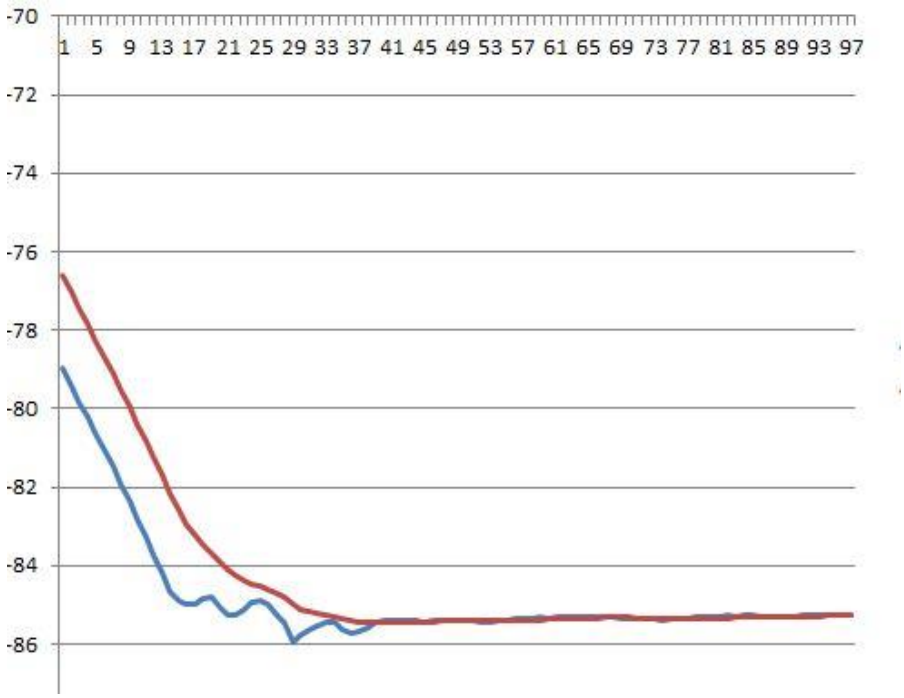


Figura 6.10 Posición angular sin filtro (línea azul) contra posición angular filtrada (línea roja).

Posteriormente, se continuó con las pruebas de ruido en las mediciones del sensor. La siguiente prueba que se realizó tuvo como objetivo conocer la incertidumbre de las medidas en estado estático. Consistió en colocar al sensor en una posición fija y realizar medidas. Una gráfica en la que se observan las medidas obtenidas por el sensor se muestra en la Figura 6.11.

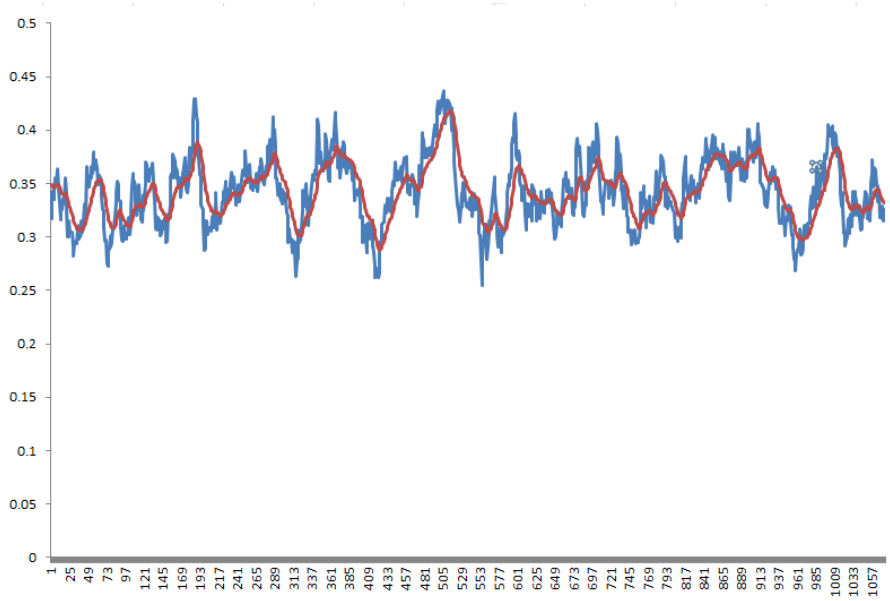


Figura 6.11 Lecturas de la IMU en estado estático.

Como se puede verificar, el sensor en estado estático tiene una incertidumbre de ±0.075°, una vez filtrada la señal.

Para realizar la prueba siguiente, se montó al Ballbot en unos polines de madera para evitar que las ruedas hicieran contacto con el suelo, y se pusieron a funcionar a los motores a una velocidad constante, al mismo tiempo, se tomaron medidas con el sensor de posición angular y se dibujó su gráfica.

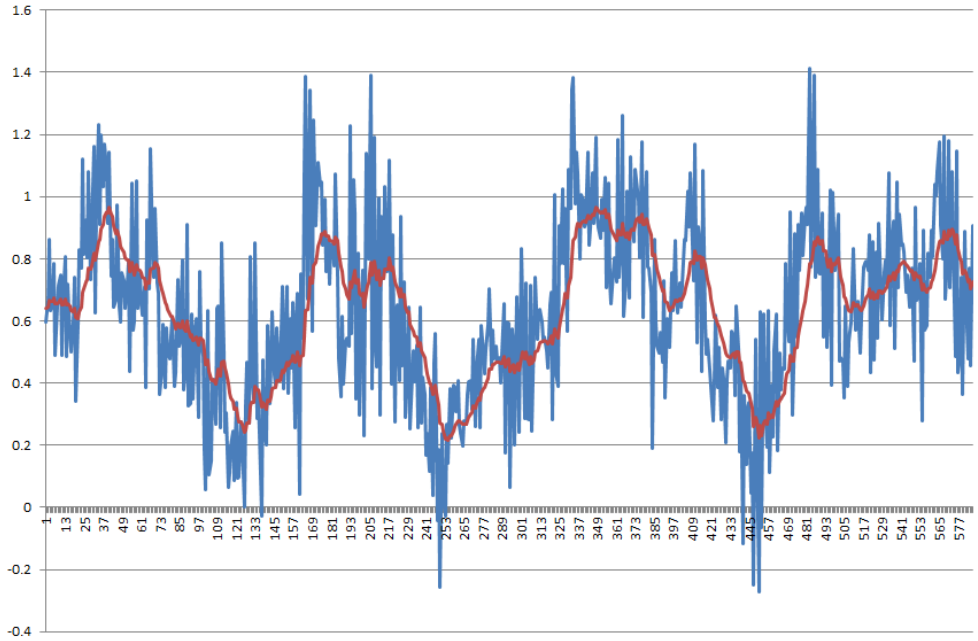


Figura 6.12 Ruido generado en la IMU debido al movimiento de los motores.

Se pueden observar los resultados de esta prueba en la Figura 6.12, en la cual se puede notar que una vez que los motores están funcionando, la incertidumbre en las mediciones aumenta a ± 0.37°. A continuación, se apagaron los motores y se obtuvieron lecturas de la posición angular mientras se desplazaba al robot sobre una superficie horizontal en dirección del eje X e Y. Estas pruebas sirvieron para caracterizar la incertidumbre en el cálculo de la posición angular debido a las imperfecciones geométricas de las ruedas omnidireccionales. En la Figura 6.13 se muestra una gráfica de las lecturas de la IMU mientras se realizó esta prueba. Se puede observar que existe una incertidumbre de ±2.75°.

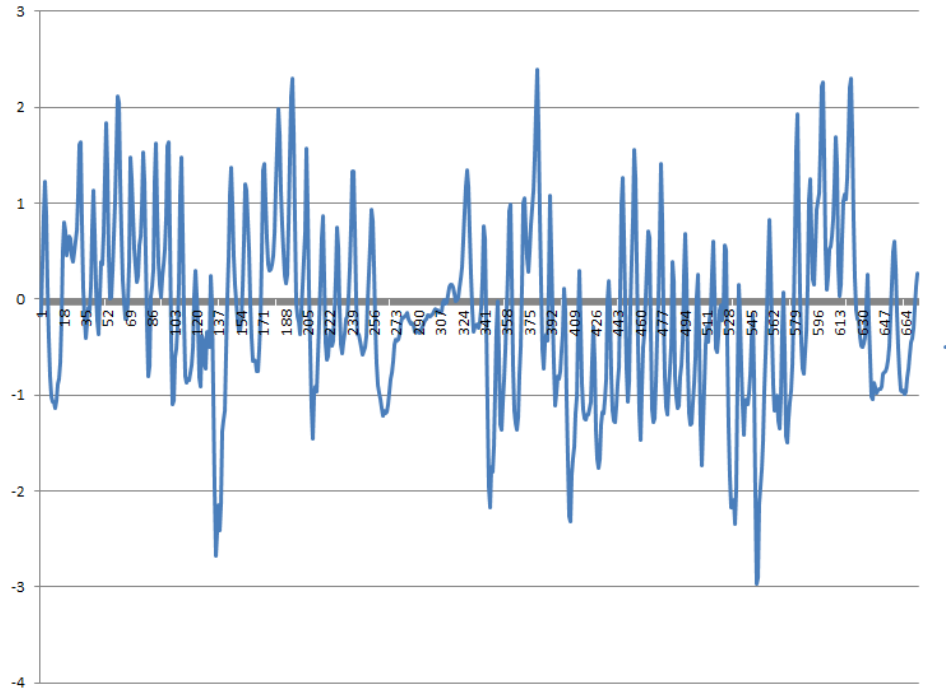


Figura 6.13 Prueba de incertidumbre de la posición angular debida a las imperfecciones geométricas de las ruedas.

Una vez realizadas todas las pruebas necesarias para conocer el error que se podía generar en la obtención de la posición angular debido a diferentes factores, se concluyó que prácticamente en todo momento se tendría una incertidumbre de ± 3°, aproximadamente. Se puede notar que el ruido en estado estático no colabora en gran medida a la incertidumbre total de la posición angular; esta fluctuación, en gran parte, es generada por la geometría de las ruedas omnidireccionales, más específicamente, por la configuración de los rodillos de éstas, ya que las ruedas tienen una superficie de contacto que no es continua, lo que modifica el ángulo del robot con cada cambio de posición de las mismas.

6.4 Pruebas de estabilización sobre la esfera

Las pruebas de estabilización se realizaron con el fin de determinar si el Ballbot era capaz de balancearse sobre la esfera y por cuánto tiempo. Asimismo, estas pruebas sirvieron para sintonizar las constantes de control con el fin de mejorar el desempeño del robot.

Durante las pruebas, se notaron fallas al momento de utilizar todos los sistemas en conjunto; la más notoria fue una falla de la comunicación I^2C

entre el micro controlador que se encargaba del cálculo de la velocidad y el que la aplicaba a los motores. La falla consistía en que aleatoriamente el dispositivo dejaba de funcionar, dejando a los motores sin una actualización de velocidad, por lo que éstos se mantenían a una velocidad constante y provocaban la pérdida de equilibro. Este error no se pudo corregir por completo, sin embargo, se logró minimizarlo al mejorar las conexiones entre los dos dispositivos.

Como se comentó anteriormente, estas pruebas sirvieron para encontrar las constantes que lograrían mejorar el desempeño en la estabilización del robot. Se tomó como punto de partida las constantes obtenidas mediante el desarrollo teórico, y se realizaron pruebas hasta encontrar valores que aumentaran el tiempo y el desempeño de la estabilización. Estas pruebas se hicieron colocando manualmente al Ballbot cerca de su punto de equilibrio y después se le soltaba; para evitar que el robot impactara contra el suelo y pudiera sufrir alguna falla permanente, se le detuvo una vez que se determinaba que no era capaz de recuperar su posición horizontal. Asimismo, durante la realización de estas pruebas, se modificaron parámetros considerados en el modelado, con el fin de verificar si éstos mejoraban el desempeño. En algunas pruebas se le agregó más peso al Ballbot mediante la adición de polines de madera, y en otros casos se intercambió la esfera por un balón de futbol o por una esfera de boliche. En todos los casos el desempeño no mejoró; para el caso de la adición de peso, los motores fallaron más cerca del punto de equilibrio que sin peso extra, y para el caso de las diferentes esferas de menos diámetro y de superficie más lisa, las llantas tuvieron deslizamientos. Por esta razón, se decidió continuar con las pruebas de estabilización con los parámetros originales. En la Figura 6.14 se muestra el Ballbot con los polines de madera acoplados a la primera etapa.

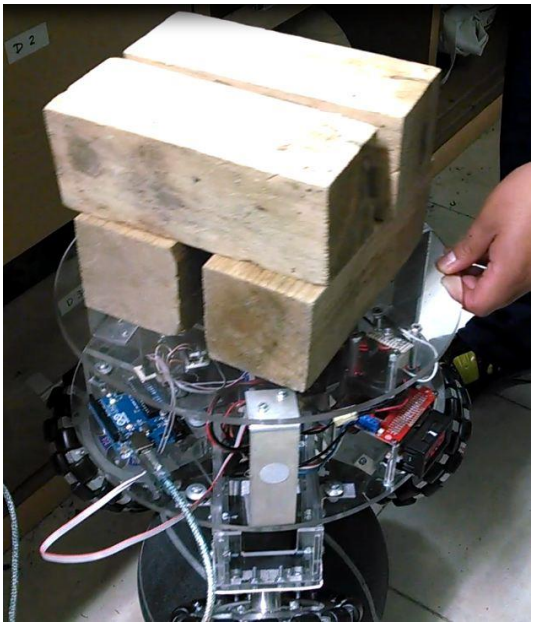


Figura 6.14 Ballbot con polines de madera para aumentar su peso.

Después de realizar muchas pruebas, se obtuvieron los valores para las constantes de control que maximizaran el tiempo de estabilidad. Se logró mantener al Ballbot balanceado sobre la esfera por lapsos de 4 a 6 segundos. Estos periodos de estabilidad se documentaron en vídeo, y en algunos casos mediante lecturas de la posición angular del robot. En las Figuras 6.15, 6.16, y 6.17 se muestran gráficas de la posición angular del robot durante diferentes pruebas. Estas gráficas se escogieron debido a que son representativas de los mejores desempeños durante la estabilización del robot, tomaron con base en una tasa de 143 muestras por segundo; en el eje de las abscisas se muestra el número de muestra y en el de las ordenadas el valor de la posición angular, en grados sexagesimales.

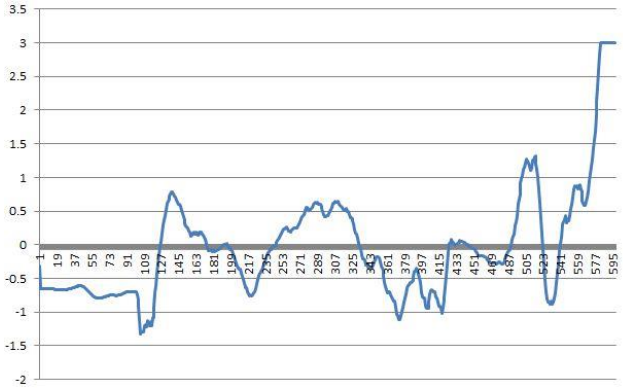


Figura 6.15 16 Posición angular durante periodos de estabilización, prueba 1

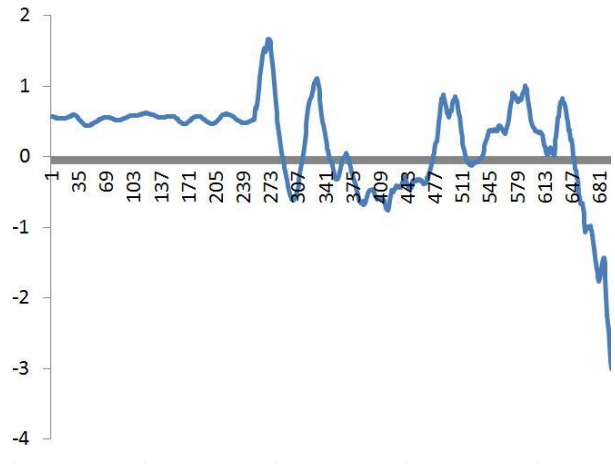


Figura 6.16 Posición angular durante periodos de estabilización, prueba 2.

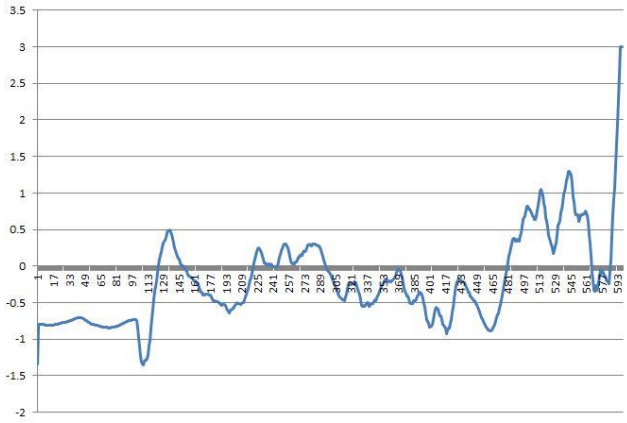


Figura 6.17 posición angular durante el periodo estabilización, prueba 3.

Analizando los vídeos y gráficas obtenidas, se puede notar que mientras el Ballbot se mantiene dentro del rango de ±0.8°, logra largos periodos de estabilización; sin embargo, una vez que cruza ese valor, muy pocas veces logra mantener la estabilidad, pues una vez que se sale de dicho rango, la amplitud de las oscilaciones se incrementa, lo que provoca que se caiga de la esfera. Una vez superados los ± 2.5°, el robot ya no es capaz de regresar, a su punto de equilibrio. Asimismo, durante el análisis de los vídeos se observó que cerca de los valores límite los motores fallan, perdiendo toda posibilidad de regresar al punto de estabilidad.

En la Figura 6.18 se muestra una imagen del Ballbot durante su estabilización; esta imagen fue obtenida a partir de una de las grabaciones que muestran la estabilidad del robot.

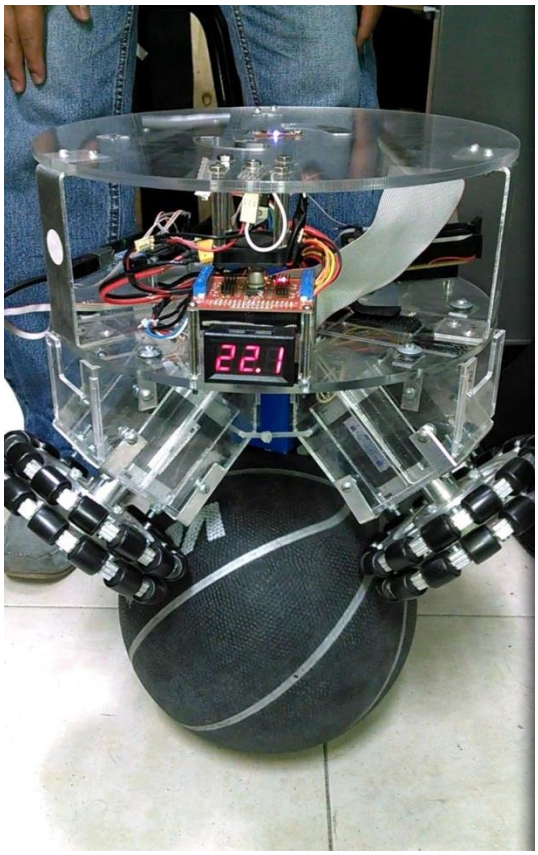


Figura 6.18 Ballbot en equilibrio dinámico.

CAPÍTULO 7
CONCLUSIONES Y TRABAJO A FUTURO

7.1 Diseño mecánico

Después de realizar el ensamble del Ballbot y realizar pruebas con diferentes pesos, tanto sobre piso plano como sobre el balón, se puede concluir que cumple con las características establecidas para rediseño mecánico. Éste resultó ser de bajo costo, su diseño de detalle resultó sencillo, la manufactura no presentó ningún problema y, lo más importante, se logró fijar las ruedas en la posición deseada y los dispositivos eléctricos se pudieron sujetar sólidamente a la base del robot.

Sin embargo, como trabajo a futuro queda la necesidad de rediseñar la manera en que se sujetan los motores paso a paso, con la finalidad de permitir montar diferentes tipos de motores y permitir montarlos a diferentes ángulos respecto a la horizontal, ya que en el prototipo actual sólo se pueden colocar con un ángulo de 45°. Este cambio sería bastante útil para poder analizar si, para este tipo de sistemas, existe un actuador y/o una posición que sea óptima para su funcionamiento.

Asimismo, con el fin de mejorar las imperfecciones geométricas, se recomienda adquirir otro tipo de ruedas omnidireccionales, específicamente unas de menor diámetro y que la superficie de contacto sea continua, ya que las discontinuidades de las utilizadas en el proyecto afectaron las lecturas de la posición y velocidad angular así como la realización de las

trayectorias de movimiento. De no ser así, se recomienda cambiar los motores por unos motores con mayor par, ya sean paso a paso o, como ya se mencionó anteriormente, por motores de corriente directa con o sin escobillas.

7.2 Control de los motores

Las pruebas realizadas a los motores paso a paso con el sistema de control fueron satisfactorias y se observa que el mismo no presenta problemas mayores que deban ser corregidos.

No obstante, es conveniente considerar la opción de emplear diferentes tipos de motores, entre ellos motores de corriente directa con o sin escobillas. Esto podría ampliar de manera notable el rango de reacción del robot en condiciones de movimiento, y por ende facilitaría el rechazo de perturbaciones.

Si es el caso, como trabajo a futuro se requerirá cambiar la estrategia de control de los motores en función del motor elegido, y de igual manera se necesitará adaptar una nueva etapa de potencia para energizar el sistema motriz.

7.3 Sistema de control

Es evidente que el controlador funciona de forma adecuada en rangos con perturbaciones pequeñas, ya que se consigue la estabilidad del robot dentro del mismo; sin embargo, esto no es suficiente para estabilizar de manera robusta al Ballbot. La estrategia de control elegida fue una estrategia basada en el modelo del sistema, y depende mucho de la calidad de las mediciones de la IMU, debido a que es lo único que se emplea como entrada de control.

Como trabajo a futuro se propone mejorar la eficiencia del filtro pasa bajas implementado, ya sea mejorando código, o bien, diseñando un filtro inteligente, y verificar si el algoritmo propuesto es el adecuado para este tipo de control de estabilidad. En caso contrario, se sugiere que se busque una alternativa diferente en cuanto a estrategia de control, como puede ser un controlador robusto o un controlador no basado en modelos. Una primera aproximación interesante de control podría ser implementar el algoritmo del control difuso diseñado en este trabajo con una mejor

elección de los conjuntos difusos y su universo de discurso. De otra forma se recomendaría utilizar técnicas de control robusto que no sean susceptibles al ruido de las mediciones.

Una vez que se estabilice robustamente al robot, se sugiere implementar un control para seguimiento de trayectorias, que podría ser muy simple, una vez que se logre la condición de estabilidad. Los autores no recomiendan intentar diseñar este control sin primero lograr la condición de estabilidad necesaria, dado que el robot puede sufrir averías en su funcionamiento, si no se tiene la precaución debida.

7.4 Reflexiones finales

Después de realizar un análisis exhaustivo de las pruebas efectuadas, y tomando en cuenta los objetivos planteados al principio de este trabajo, se puede afirmar que se lograron importantes avances tanto en la construcción de este tipo de robots como en su funcionamiento.

Queda pendiente mejorar algunos aspectos de los demás sistemas que componen al Ballbot, pero es un trabajo que puede ser realizado por estudiantes de posgrado o con mucha experiencia en el procesamiento de señales, control, diseño, entre otras áreas involucradas en el presente proyecto.

La plataforma móvil que se creó es fácil de modificar en cada una de sus partes, lo que asegura que sea un proyecto que pueda continuarse fácilmente y llevarse a la práctica sin demasiado trabajo previo.

En conclusión, el proyecto realizado cumplió el objetivo propuesto; sin embargo, queda mucho trabajo a futuro que se puede realizar para mejorar su desempeño.

APÉNDICE

A.1 Ecuaciones de movimiento del ballbot

Debido a que el sistema de locomoción del *Ballbot* es un sistema omnidireccional, fue necesario realizar un análisis cinemático para obtener una función que nos permita controlar el desplazamiento del *ballbot* en la dirección deseada para mantener el equilibrio.

Para poder realizar este análisis, se necesitó establecer un sistema de referencia para el desplazamiento del robot, los vectores de movimiento de cada una de las ruedas (R0, R1, R2, son vectores unitarios), un vector genérico de desplazamiento del robot (V) y finalmente una velocidad angular para el ballbot (W).

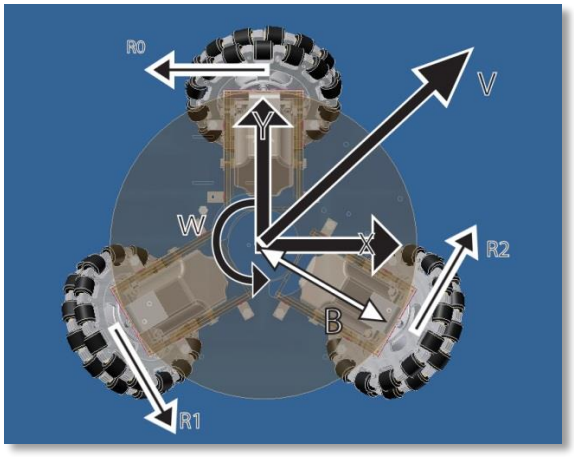


Figura A.1 Vectores de dirección y de movimiento sobrepuestos a la vista superior del ballbot

Una vez establecido un sistema de referencia y los vectores involucrados, se comenzó el análisis del movimiento del ballbot. Nos interesó establecer un vector deseado para el movimiento del robot (V) junto con una velocidad angular deseada, y que mediante el análisis, obtengamos la velocidad angular requerida de cada rueda omnidireccional.

En la siguiente tabla podemos ver las variables que nos ayudaron a definir una ecuación de movimiento:

TABLA A.1 Variables para el movimiento omnidireccional

Variable	**Descripción**
r	Radio de las ruedas omnidireccionales
R0,R1,R2	Vectores unitarios en dirección del movimiento de las ruedas
V	Velocidad deseada del ballbot
W	Velocidad angular deseada del ballbot
B	Distancia del centro del ballbot al centro de las ruedas
w0, w1, w2	Velocidades angulares de las ruedas omnidireccionales
v0, v1, v2	Velocidades lineales de las ruedas omnidireccionales
n	Numero de rueda omnidireccional
Vn	Velocidad del ballbot en una rueda especifica
R0	[-1,0]
R1	[1/2,-sqrt(3)/2]
R2	[1/2, sqrt(3)/2]

Cada rueda deberá de tener una velocidad especifica la cual depende completamente del V (vector de velocidad) y de W (velocidad angular) del ballbot, la cual es la suma de la velocidad debida al movimiento del cuerpo del ballbot y su rotación.

$$Vn = V + (B \bullet W) * Rn$$

$$Vn \bullet Rn = \text{velocidad de la rueda}$$

Por lo tanto

$$\text{Velocidad de la rueda} = V \bullet Rn + B*W$$

Y

$$r*wn = V \bullet Rn + B*W$$

Por lo tanto

$$w0 = (V \bullet R0 + B*W)/r$$

$$w1 = (V \bullet R1 + B*W)/r$$

$$w2 = (V \bullet R2 + B*W)/r$$

Cabe resaltar que las unidades de todas las variables se encuentran en sistema internacional, por lo que las velocidades angulares de cada rueda se encuentran en radianes por segundo.

A.2 Tarjeta de comunicación entre el Quadstepper y el Arduino Due

Con el fin de conectar el *Arduino Due*, que es el encargado de controlar los motores paso a paso, con la tarjeta utilizada para facilitar el control, la *Quadstepper driver*, se realizó un circuito en placa de cobre desde al cual saldría un cable plano de 32 hilos. El diseño del circuito y el resultado de la manufactura se muestran en la Figura A.2. Esta tarjeta se maquinó mediante una máquina de control numérico.

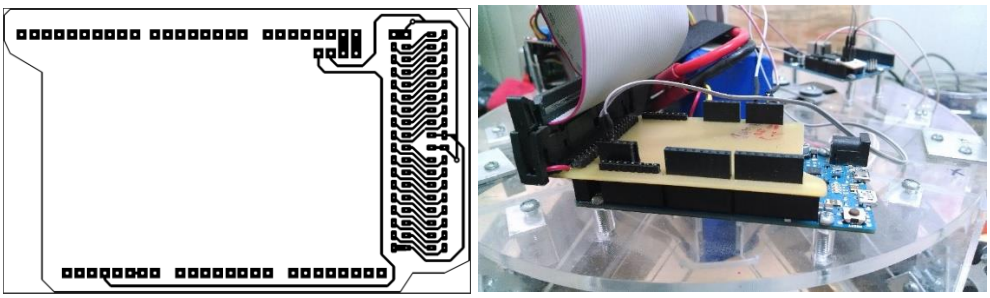


Figura A.2 Tarjeta de comunicación entre el Arduino Due y el *Quadstepper driver*

A.3 Comparación de la respuesta de la IMU conectada a 3.3V y a 5V

Durante la realización del proyecto nos dimos cuenta que a la IMU se le debía de alimentar con 5V (según su hoja de datos), sin embargo, esta nos estaba dando lecturas con mucho ruido y con grandes picos. Después de investigar en diversos foros de internet, encontramos que en algunos casos el sensor funcionaba mejor cuando se le alimentaba con 3.3V, por lo que decidimos intentar realizar ese ajuste y las lecturas del sensor mejoraron de gran manera. Durante el resto del proyecto se mantuvo alimentada la IMU con 3.3V

A.4 Notas de carga de la batería Li-Po

El utilizar este tipo de baterías requiere de gran cuidado, ya que si se descarga más de lo que debe, es probable que esta deje de funcionar correctamente, asimismo, si esta se descarga a mayor velocidad para la cual está diseñada, esta puede fallar y hasta inflarse y explotar, asimismo, cuando se carga de más

Durante las pruebas de funcionamiento de los subsistemas donde la batería se descargaba, esta se puso a cargar cuando su voltaje bajo de los 21.1V y se le cargo hasta los 23.9V haciendo una carga balanceada cada 7-10 cargas no balanceadas. La batería se cargó con una corriente de 1.5A. Realizando este procedimiento la batería no se dañó en ningún momento, por lo que a reserva de tener un mejor método para la carga y descarga de la batería, se recomienda que se utilice el que se citó anteriormente.

A.5 Selección de componentes

La selección de los componentes es una parte esencial para el éxito del proyecto. El seleccionar erróneamente alguno de ellos puede hacer que todo el proyecto falle o su funcionamiento no sea el correcto. La selección de los componentes es resultado de una selección previa realizada por la primera iteración del *Ballbot,* de la cual ya se platicó en los antecedentes. Sin embargo en el lapso entre la conclusión del proyecto anterior y este se adquirieron nuevos dispositivos entre los cuales podemos escoger. También

existe la posibilidad de adquirir otros. La selección de los componentes la dividiremos en las siguientes categorías:

- Sensores
- Actuadores
- Ruedas omnidireccionales
- Electrónica
- Baterías

A continuación abordaremos cada categoría y enunciaremos las posibles opciones para cada una de ellas. Tomaremos como parámetros a considerar, la facilidad de uso, el costo, la facilidad de adquisición, y las capacidades propias del elemento.

A.5.1 Sensores

En este apartado se ha considerado que se utilizara una *IMU* para la obtención de los *ángulos de Euler*. Sin embargo tenemos cuatro posibles modelos de *IMUs*.

- Ultimate IMU
- Mongoose
- Sensor Stick

Con base en el artículo *Aplicación de una unidad de medición inercial en tiempo real para el control de un ballbot*, en el cual se realiza un comparativo de los sensores inerciales antes mencionados, se obtiene que el que tiene un mejor desempeño es la *IMU Sensor Stick,* la cual ya había sido adquirida en el laboratorio donde se realizó el proyecto.

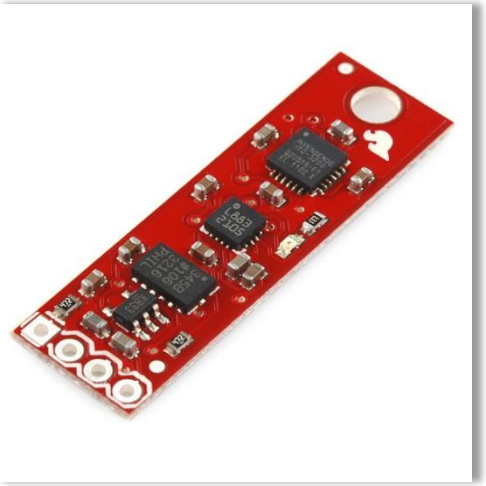


Figura A.3 9 Degrees of Freedom –Sensor Stick

TABLA A.2 Especificaciones del Sensor Stick

Nombre	Rango	Ejes	Requerimientos de energía	Características extras
9 Degrees of Freedom – Sensor Stick	Accel: ±2,4,8,16g Gyro: ±2000°/s	Accel: 3 Gyro: 3	3.3 – 16VDC	Tiene un magnetómetro de 3 ejes

A.5.2 Actuadores

Como parte de la primera iteración del proyecto, ya se contaba con los motores paso a paso y con un respaldo teórico de cómo funcionan y el antecedente de que se desempeñó en proyectos similares ha sido bueno. Así mismo los motores tienen las características mecánicas necesarias para el proyecto.

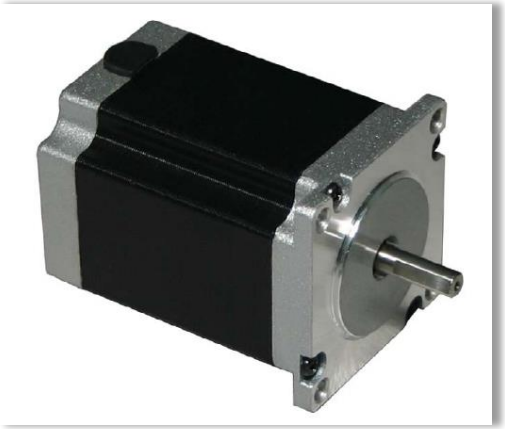


Figura A.4 Motor de pulsos modelo 23Y202S-LW8

TABLA A.3 Caracteristicas del motor 23Y202s-LW8

Nombre	Tamaño NEMA	Par (oz-in)	Corriente (A)	Voltaje RMS (V)	# de cables	Peso (lbs)	Longitud (in)
23Y202D-LW8	23	262	0.7	10.5	8	2.21	2.99

A.5.3 Ruedas omnidireccionales

En el caso de las ruedas omnidireccionales que se utilizaron en el proyecto, son las de mejores características y desempeño que se pueden conseguir de manera comercial, las *6" Aluminum Dualie Omni Wheel (am-0432)* [9], creadas por *AndyMark*. Ya que el tipo de rueda que sería ideal para el proyecto, tendría que mandarse a manufacturar a un centro de manufactura avanzada, para conseguir las tolerancias requeridas, y el costo del proyecto se incrementaría sensiblemente.

Figura A.5 6" Aluminum Dualie Omni Wheel (am-0432)

A.5.4 Electrónica

A.5.4.1 Tarjeta controladora de los motores

Como parte de los antecedentes del proyecto se tenían unas tarjetas que funcionaban como controladores de los motores paso a paso, las cuales tenían la función de realizar *microstepping*. Sin embargo estas tarjetas tenían un rango limitado de operación lo cual afectaba de gran manera el desempeño del sistema. Como solución a este problema se propuso adquirir una tarjeta comercial para controlar este tipo de motores con un mejor desempeño. Esta tarjeta contiene cuatro circuitos integrados *A4983* los cuales se encargan de realizar un *Microstepping* para poder tener más resolución de movimiento de los motores.

Figura A.6 Tarjeta controladora Quadstepper Motor Driver Board [11]

A.5.4.2 Arduino UNO

Al inicio del proyecto se propuso implementar el control del *Ballbot* en un *Arduino UNO* cuyas características principales son.

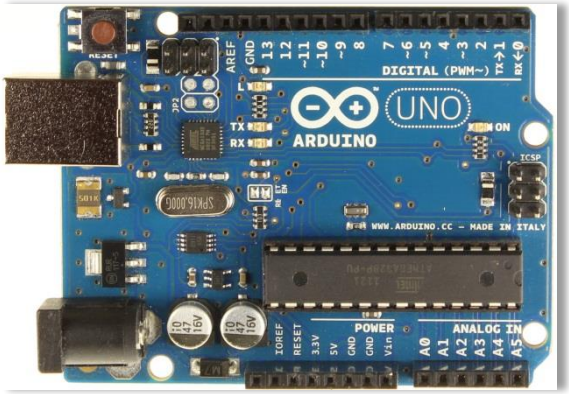


Figura A.7 Arduino UNO [12]

TABLA A.4 Características del Arduino Uno

Microcontrolador	ATmega328
Voltaje de operación	5V
Pines Digitales I/O	14 de las cuales 6 con PWM
Pines Analógicos	6
Memoria Flash	32 KB (ATmega328) del cual 0.5 KB es usado por el bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Reloj	16 MHz

Sin embargo esta tarjeta presenta importantes limitantes en su memoria y velocidad de procesamiento, por lo cual se decidió que esta tarjeta se encargaría únicamente de recibir y procesar los datos de la IMU.

A.5.4.3 Arduino Due

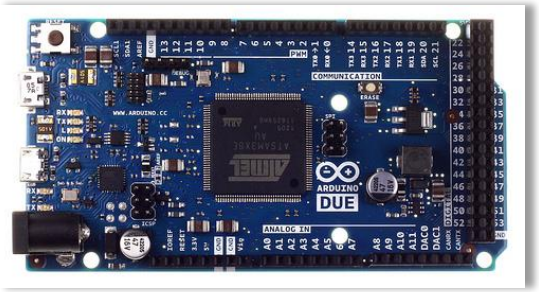


Figura A.8 Arduino Due [13]

TABLA A.5 Características del Arduino Due

Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3V
Digital I/O Pins	54 de los cuales 12 tienen PWM
Analog Input Pins	12
Analog Outputs Pins	2 (DAC)
Flash Memory	512 KB
SRAM	96 KB (dos bancos: 64KB y 32KB)
Clock Speed	84 MHz

El *Arduino Due* es una versión mejorada del *Arduino UNO*, la cual contiene un reloj de mucha mayor velocidad (84 MHz), más memoria para programar, y un procesador mucho más poderoso (procesador ARM de 32 bits, *Atmel SAM3X8E ARM Cortex-M3*). Todo esto manteniendo la facilidad de uso que el caracteriza al *Arduino.* Sumadas a estas características podemos tomar en cuenta que este dispositivo tiene el respaldo de una comunidad de desarrolladores muy grande, que está generando nuevo código día a día. Por lo que, pese a que actualmente muchas bibliotecas del *Arduino UNO* no son compatibles, en poco tiempo existirán sus versiones para el *Arduino Due*.

Finalmente se tomó la decisión de adquirir un *Arduino Due* y utilizarlo junto con el *Arduino UNO,* basándonos en que existe una gran comunidad de desarrolladores que facilitan la creación e implementación de nuevas bibliotecas que podrían ser utilizadas en nuestro proyecto.

Así mismo el número de pines de entrada y/o salida del *Arduino DUE* presenta una ventaja para poder utilizarlo como controlador para la tarjeta de los motores paso a paso. Esto nos permitió utilizar la tarjeta como un controlador de los motores, el cual recibirá las aceleraciones que deberá llevar cada eje del Ballbot y el controlador las transformara a velocidad de cada rueda omnidireccional.

A.5.5 Baterías

Como se adquirió para el proyecto una batería Li-Po de 5000mAh y con una constante de descarga de 25c de 6 celdas con un total de 22.2V, la cual tiene un desempeño suficiente para la parte del movimiento de los motores.

Figura A.9 Turnigy 5000mAh 3S 25C Lipo Pack [10]

A.5.6 Esfera

Para que el *Ballbot* se pueda mover, se requiere de una esfera. Del proyecto anterior se contaba con una esfera realizada de *MDF* la cual presentaba problemas mecánicos debido a su irregularidad geométrica. Es por eso que se decidió utilizar un balón de basquetbol que por geometría y características mecánicas se adapta mejor a nuestras necesidades.

Figura A.10 Balón de Basquetbol VOIT

Diámetro: 24 cm

Peso: 600g

A.6 Código de control de los motores

A continuación se muestra el código utilizado para mover a los motores con una entrada de velocidad compuesta por una cantidad en X y otra en Y.

```
#include <DueTimer.h>//libreria para poder realizar interrupciones facilmente
#include <AccelStepper.h>//libreria para controlar los motores paso a paso
#include <Wire.h>/libreria para utilizar el protocolo I2C
#include <I2C_Anything.h>//Libreria que facilita el protocolo I2C para valores flotantes
#include <math.h>//librer[ia para operaciones matematicas
void configMotor();
/*motor uno*/ //Se define cada una de los pines para utilizar a los motores
#define ENC1 23
#define SLP1 25
#define RST1 27
#define MS11 29
```

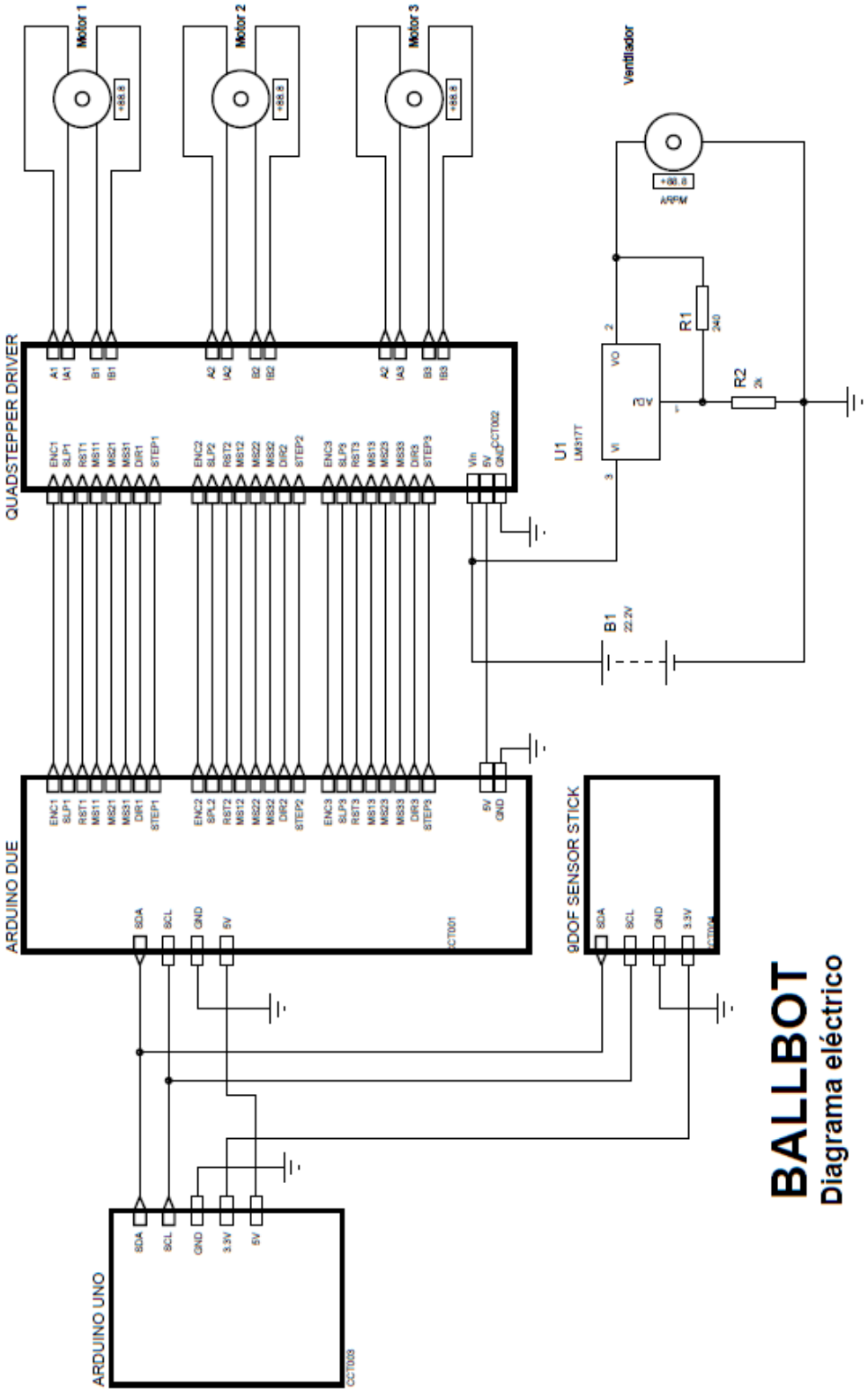
```
#define MS21 31
#define MS31 33
#define DIR1 35
#define STEP1 37
/*motor dos*/
#define ENC2 22
#define SLP2 24
#define RST2 26
#define MS12 28
#define MS22 30
#define MS32 32
#define DIR2 34
#define STEP2 36
/*motor tres*/
#define ENC3 52
#define SLP3 50
#define RST3 48
#define MS13 46
#define MS23 44
#define MS33 42
#define DIR3 40
#define STEP3 38
 ///parametros del ballbol
static float R0x=-1.00;  //vectores unitarios de movimiento de las ruedas
 static float R0y=0.00;
 static float R1x=(1.00/2.00);
 static float R1y=-(sqrt(3.00)/2.00);
 static float R2x=(1.00/2.00);
 static float R2y=sqrt(3.00)/2.00;
 static float resf=.12;//radio esfera
 ////////////////////////////////////////////
//velocidade iniciales y finales en X, Y y angula en m/s
float Vx=0.0;
float Vy=0.0;
//velocidades maximas permisibles en m/seg
float w=.0; //en rad/seg
float wmax=.5;
float alfa=0.00; //en rad/seg*seg
```

```
//velocidades angulares de las llantas
float wx=0.0;
float wy=0.0;
float wz=0.0;
//pasos por segundo de los motores
float fx=0.;
float fy=0.;
float fz=0.;
//frecuencia de interrupcion para la accion de control.
float frecint=200.0;
//Entradas controlador en m/s*s
float Ax=0.0;
float Ay=0.0;
int x=0;
// Define los pines a usarse para controlar los motores
AccelStepper MotX(1,STEP2,DIR2);// 1 es que se esta utilizando con un driver
AccelStepper MotY(1,STEP3,DIR3);
AccelStepper MotZ(1,STEP1,DIR1);
//declara la cantidad de pasos por revolucion dependiendo de MS1,MS2,MS3 de cada motor
   float pasos=3200;
void setup()
{
   configMotor();
   MotX.setMaxSpeed(6400.0); //velocidad maxima en pasos por segundo
   MotX.move(-64000);
   MotY.setMaxSpeed(6400.0); //velocidad maxima en pasos por segundo
   MotY.move(-64000);
   MotZ.setMaxSpeed(6400.0); //velocidad maxima en pasos por segundo
   MotZ.move(-64000);
  Wire.begin(4);              // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
 Timer6.attachInterrupt(Control).setFrequency(frecint).start();
}
void loop()
{
  MotX.runSpeed();//se encarga de dar el siguiente paso cuando es necesario
```

```
  MotY.runSpeed();//es necesario llamar esta funcion lo mas frecuente posible
  MotZ.runSpeed();
}
void receiveEvent(int howMany)
{
  I2C_readAnything(Vx);
  I2C_readAnything(Vy);
}
void configMotor(){
  /*motor1*/
    pinMode(ENC1,OUTPUT);//ENC es activa baja
    digitalWrite(ENC1,0);
    pinMode(SLP1,OUTPUT);//SPL es activa baja
    digitalWrite(SLP1,1);
    pinMode(RST1,OUTPUT);//RST es activa baja
    digitalWrite(RST1,1);
    pinMode(MS11,OUTPUT);//MS son activas bajas
    digitalWrite(MS11,1);//MS1=1 MS2=1 MS3=1 es paso/16
    pinMode(MS21,OUTPUT);//MS1=1 MS2=1 MS3=0 es paso/8
    digitalWrite(MS21,1);//MS1=0 MS2=1 MS3=0 es paso/4
    pinMode(MS31,OUTPUT);//MS1=1 MS2=0 MS3=0 es paso/2
    digitalWrite(MS31,1);//MS1=0 MS2=0 MS3=0 es un paso
    /*motor2*/
    pinMode(ENC2,OUTPUT);
    digitalWrite(ENC2,0);
    pinMode(SLP2,OUTPUT);
    digitalWrite(SLP2,1);
    pinMode(RST2,OUTPUT);
    digitalWrite(RST2,1);
    pinMode(MS12,OUTPUT);
    digitalWrite(MS12,1);
    pinMode(MS22,OUTPUT);
    digitalWrite(MS22,1);
    pinMode(MS32,OUTPUT);
    digitalWrite(MS32,1);
    /*motor3*/
    pinMode(ENC3,OUTPUT);
```

```
   digitalWrite(ENC3,0);
   pinMode(SLP3,OUTPUT);
   digitalWrite(SLP3,1);
   pinMode(RST3,OUTPUT);
   digitalWrite(RST3,1);
   pinMode(MS13,OUTPUT);
   digitalWrite(MS13,1);
   pinMode(MS23,OUTPUT);
   digitalWrite(MS23,1);
   pinMode(MS33,OUTPUT);
   digitalWrite(MS33,1);
}
void Control(){
  //calculo de las velocidades angulares de las ruedas
  wx=( (-Vx*R0x) + (Vy*R0y) + (b*w) ) / r;
  wy=( (-Vx*R1x) + (Vy*R1y) + (b*w) ) / r;
  wz=( (-Vx*R2x) + (Vy*R2y) + (b*w) ) / r;
  //calculo de la direccion del movimiento de los motores
  fx=-wx*pasos;
  fy=-wy*pasos;
  fz=-wz*pasos;
  MotX.setSpeed(fx);//le actualiza la velocidad a los motores
  MotY.setSpeed(fy);
  MotZ.setSpeed(fz);
}
```

A.7 Diagrama eléctrico



A.8 Control

El diseño de los controladores propuestos en este trabajo se desarrolló teóricamente empleando la herramienta de software Mathematica en su versión 7.

Para poder diseñar las diversas estrategias de control lo primero que se hizo fue obtener el valor numérico de la función de transferencia según los parámetros del robot y la función resultante del modelado. En la Figura A.11 se muestra el código implementado en Mathematica.

Función de transferencia

```
Clear["Global`*"]
(*Parámetros del ballbot*)
Ie = 0.005;
Ip = 0.08;
re = 0.125;
mp = 8;
me = 0.6;
L = 0.21;
g = 9.78;
(*Función de transferencia θ[s]/(s^2 ψ[s]) del ballbot*)
G = (Ie + (mp + me) * re^2 + mp * L * re) / (mp * L * g - (Ip + mp * (L + re) ^2 + Ie + me * re^2) * s^2)
```

$$\frac{0.349375}{16.4304 - 0.992175\, s^2}$$

Figura A.11 Código de la definición de la función de transferencia

A.8.1 Diseño del controlador Proporcional

El primer controlador diseñado fue el proporcional. Esto se hizo como una primera aproximación para estabilizar al robot una vez que se sabe que el sistema es inestable. Dependiendo del rendimiento, que se espera no sería el mejor, se podía analizar la factibilidad del mismo para ser implementado debido a la facilidad que esto implica. La Figura A.12 muestra el código empleado.

Control P

```
(*Función de transferencia del controlador*)
Co = Kp;
(*Función de transferencia del sistema en lazo cerrado*)
Tf = G*Co / (1 - G*Co) // FullSimplify
```

$$\frac{0.349375\,Kp}{16.4304 - 0.349375\,Kp - 0.992175\,s^2}$$

```
(*Es posible emplear el criterio de estabilidad de Newton. Por lo tanto
 todos los coeficientes del polinomio característico deben ser del mismo signo*)
Coefficient[Denominator[Tf], s^2]
Coefficient[Denominator[Tf], s, 0]
```

```
-0.992175
```

```
16.4304 - 0.349375 Kp
```

```
(*Para que el sistema sea estable*)
Ec1 = Coefficient[Denominator[Tf], s, 0] < 0
```

```
16.4304 - 0.349375 Kp < 0
```

```
Reduce[Ec1, Kp]
```

```
Reduce::ratnz: Reduce was unable to solve the system with inexact coefficients. The answer was obtained by solv
Kp > 47.028
```

Figura A.12 Diseño del control Proporcional

A.8.2 Diseño del controlador Proporcional-Derivativo

Posteriormente se diseñó un controlador PD para controlar el desempeño del robot y estabilizarlo al mismo tiempo. Los parámetros de diseño propuestos se obtuvieron con base en la experiencia práctica utilizando el robot. El diseño del controlador se realizó por ubicación de polos. A partir de los parámetros de diseño se obtuvo una ecuación característica deseada. Posteriormente, se compararon la ecuación característica deseada y la correspondiente a la función de transferencia del sistema en lazo cerrado, con ello se conocieron los valores de las constantes de control que idealmente harían funcionar al robot como se observaba en simulación. Sin embargo, debido a la incertidumbre paramétrica esto no fue posible. En la Figura A.13 se observa el código empleado.

Control PD

```
Clear[s, Tf, Kp, Kd]

(*Función de transferencia del controlador*)
Co = Kp + Kd * s;
Tf = G * Co / (1 - G * Co) // FullSimplify
```

$$\frac{0.349375\,Kp + 0.349375\,Kd\,s}{16.4304 - 0.349375\,Kp - 0.349375\,Kd\,s - 0.992175\,s^2}$$

```
Clear [psi, Dend, wn];
ts = 0.5;
sp = .2;
psir = FindRoot[sp == Exp[-Pi * psi / Sqrt[1 - psi^2]], {psi, 0}]
```

$$\{psi \to 0.45595\}$$

```
wnr = FindRoot[ts == 5 / (psi * wn) /. psir, {wn, 1}]
```

$$\{wn \to 21.9322\}$$

```
Dend = s^2 + 2 * psi * wn * s + wn^2 /. psir /. wnr
```

$$481.023 + 20.\,s + s^2$$

```
Denc = Denominator[Tf] / Coefficient[Denominator[Tf], s^2] // Simplify
```

$$-16.56 + 0.35213\,Kp + 0.35213\,Kd\,s + 1.\,s^2$$

```
Solve[Coefficient[Denc, s] == 2 * psi * wn, Kd] /. psir /. wnr
Solve[Coefficient[Denc, s, 0] == wn^2, Kp] /. psir /. wnr
```

$$\{\{Kd \to 56.7971\}\}$$

$$\{\{Kp \to 1413.06\}\}$$

```
root = Solve[Dend == 0, s]
```

$$\{\{s \to -10. - 19.5198\,i\}, \{s \to -10. + 19.5198\,i\}\}$$

Figura A.13 Diseño del control Proporcional-Derivativo

A.8.3 Diseño del control por realimentación de estados

La siguiente aproximación en el diseño de controladores fue un control por realimentación de estados. Este tipo de controlador se asemeja mucho a un controlador proporcional derivativo convencional, sin embargo, la metodología de diseño es distinta. En este caso, las ganancias del control

actúan en la realimentación del estado y no en el error respecto a la salida deseada.

La plataforma de software MATLAB tiene comandos especiales que facilitan el diseño de este tipo de controladores, por lo cual el diseño completo del mismo se hizo en dicho software.

El procedimiento fue, primero, declarar la función de transferencia del sistema en lazo cerrado, después encontrar su equivalente en el espacio de estados empleando el comando "TF2SS" de MATLAB y una vez obtenidas las matrices verificar la controlabilidad y observabilidad del sistema. Una vez que se comprobó que ambos parámetros del sistema permiten el diseño de un controlador, esto sucede sólo si el rango de las matrices de controlabilidad y observabilidad es igual al orden del sistema, se definieron los parámetros de diseño y se obtuvo la matriz de ganancias K del control empleando el comando "place" de MATLAB que recibe como parámetros las matrices A y B del sistema y los polos deseados.

A continuación se presenta el código de MATLAB empleado. En la Figura A.14 se observa una captura de pantalla de los resultados de la ventana de trabajo de MATLAB.

Código de MATLAB

```
%Script para el diseño del controlador por realimentación de
estados
clear
clc
%Parámetros del Ballbot
Ie = 0.005;
Ip = 0.08;
re = 0.125;
mp = 8;
me = 0.6;
L = 0.21;
g = 9.78;
%Función de transferencia final
Tf=tf([(Ie+re*(L*mp+(me+mp)*re))],[-
(Ie+Ip+L^2*mp+2*L*mp*re+(me+mp)*re^2) 0 g*L*mp])
[n,d] = tfdata(Tf,'v');
%Se encuentra el equivalente del sistema en el espacio de
estados
[A,B,C,D] = tf2ss(n,d);
%Se analiza si el sistema es controlable y observable
controlable = rank(ctrb(A,B))
observable = rank(obsv(A,C))
%Se definen los parámetros de diseño
psi=0.45595;
```

```
wn=21.9322;
polos=roots([1 2*psi*wn wn^2])
%Se obtiene el valor de las constantes del controlador
K = place(A,B,polos)
```

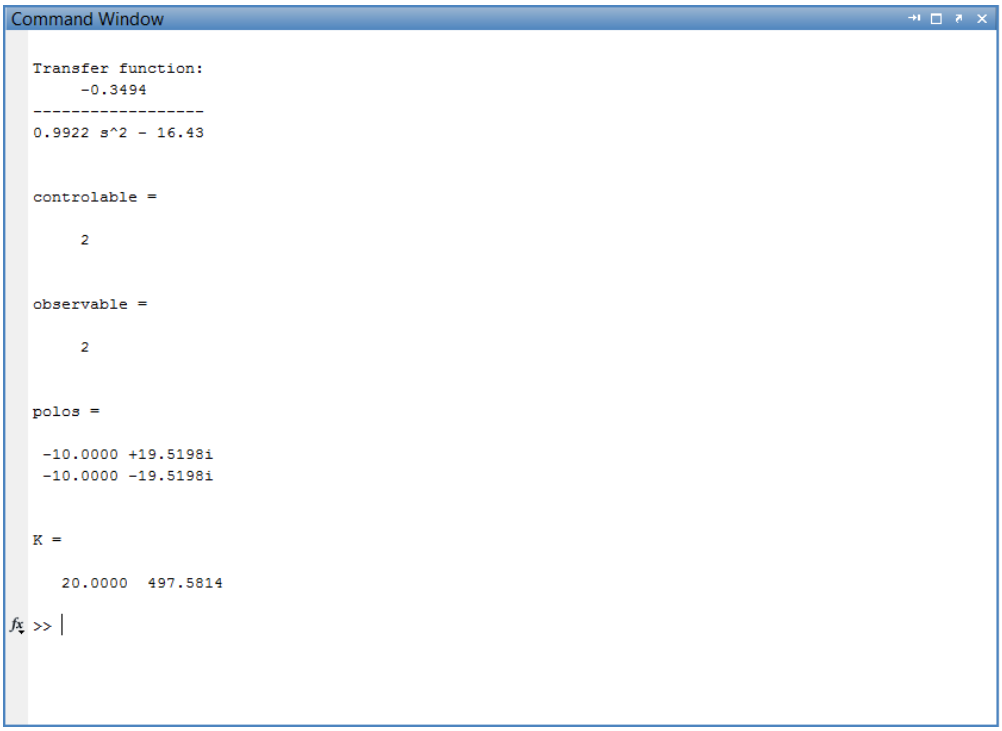


Figura A.14 Resultados de la consola de MATLAB después de emplear el código propuesto

A.8.4 Diseño del controlador difuso

En la búsqueda de una alternativa diferente para controlar al robot se pensó en utilizar un control no basado en modelos, por lo cual se diseñó un controlador empleando lógica difusa tipo mamdani.

El diseño de este tipo de controladores consiste básicamente en proponer valores diversos de los estados del sistema y definirles una función de propiedad o de membresía. El rango de valores seleccionado se denomina universo de discurso. Esto se realiza de igual forma para entradas y salidas. Posteriormente se propone un conjunto de reglas que definan cuál debe ser el comportamiento del sistema ante una entrada determinada respecto a la salida. Esto como podrá inferirse es un proceso que requiere de un amplio conocimiento del sistema y de cómo se desea modificar el comportamiento con el control.

Una vez realizado todo esto se debe implementar un algoritmo específico que realice un mapeo de las variables de entrada en valores difusos. A este

proceso se le conoce como “difusión”. Una vez que las variables de entrada tienen valores difusos se evalúan las reglas y se obtiene una salida resultante que es difusa. Esta parte del controlador se conoce como máquina de inferencia. Finalmente se desea obtener un valor numérico de la salida que será la señal de control enviada a los actuadores. A este proceso se le denomina “desdifusión”.

El proceso es iterativo y se ejecuta a cada momento, por lo cual el algoritmo que se diseñe para la implementación debe ser los suficientemente rápido si se desean controlar aplicaciones como un ballbot.

En la plataforma de MATLAB también se tiene un toolbox auxiliar que facilita el diseño de este tipo de controladores, el cual se empleó en este trabajo.

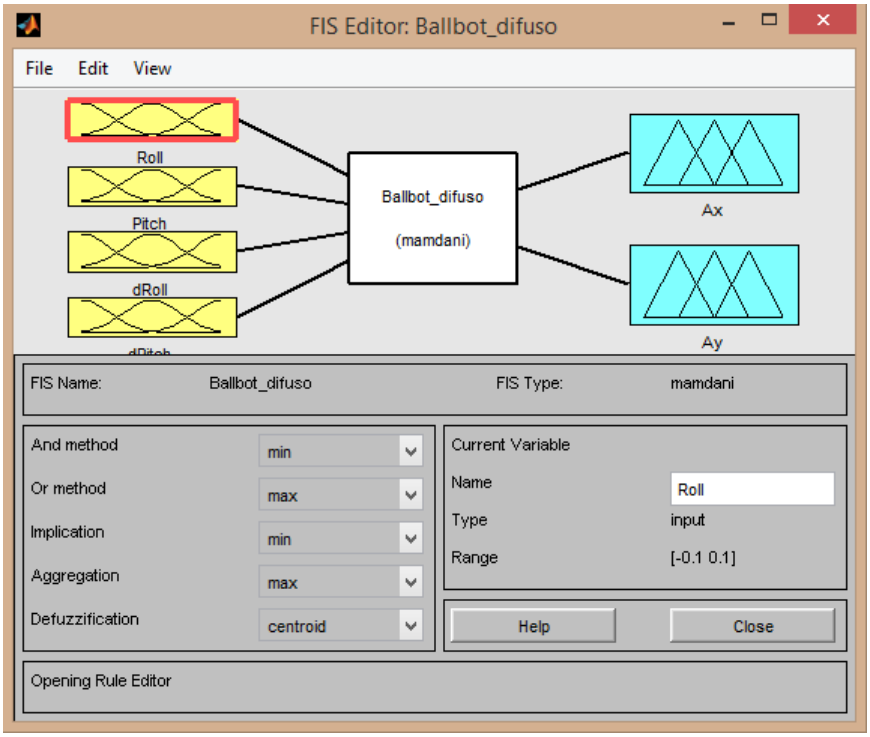


Figura A.15 Ventana de diseño del toolbox de MATLAB

En la Figura A.15 se observa la ventana de diseño del controlador difuso propuesto. En la misma se observan los diferentes menús que se ofrecen para el diseño así como las características del controlador diseñado, como los métodos de difusión, agregación, implicación, desdifusión y las variables que componen al controlador. En este caso se utilizan 4 entradas, es decir, cada uno de los estados de los dos planos de control del robot, por lo que también se tienen dos salidas, correspondientes cada una a su respectivo

plano de control. Desde esta ventana también se pueden seleccionar las reglas de la máquina de inferencia que se observa en el centro de la Figura 5 y se puede ver la superficie de control, que es útil revisar constantemente en el diseño para evitar tener cambios bruscos en la salida provocados por un mal diseño de reglas.

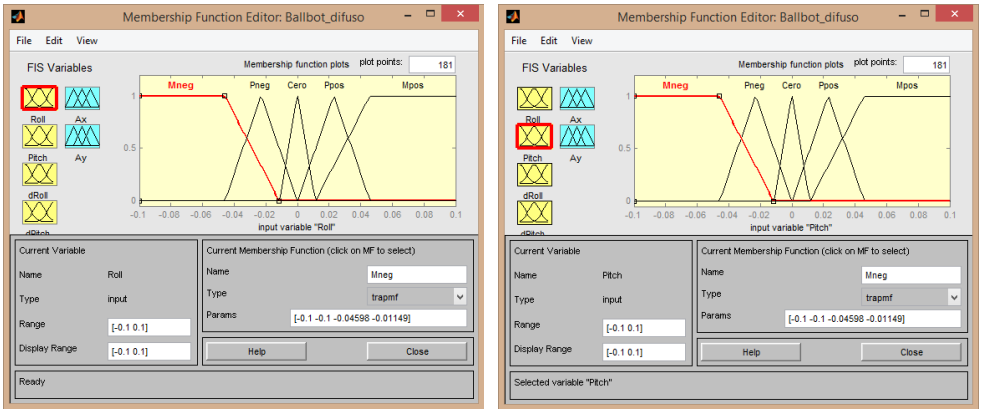


Figura A.16 Ventanas de diseño de las variables de entrada roll y pitch

En la Figura A.16 se observan las ventanas de diseño de las funciones de pertenencia de dos los conjuntos de entrada propuestos. El diseño de los mismos se realizó por inferencia, es decir, mediante la experiencia práctica con el robot y conociendo de forma somera su respuesta ante las condiciones del ambiente. Las funciones de pertenencia que componen a cada sistema son tres funciones triangulares, una función “s” y una función “z” denominadas así por su forma, las cuales se observan de igual forma en la Figura A.16. El universo de discurso va de -0.1 a 0.1 rad y el valor máximo de cada función es de 1.

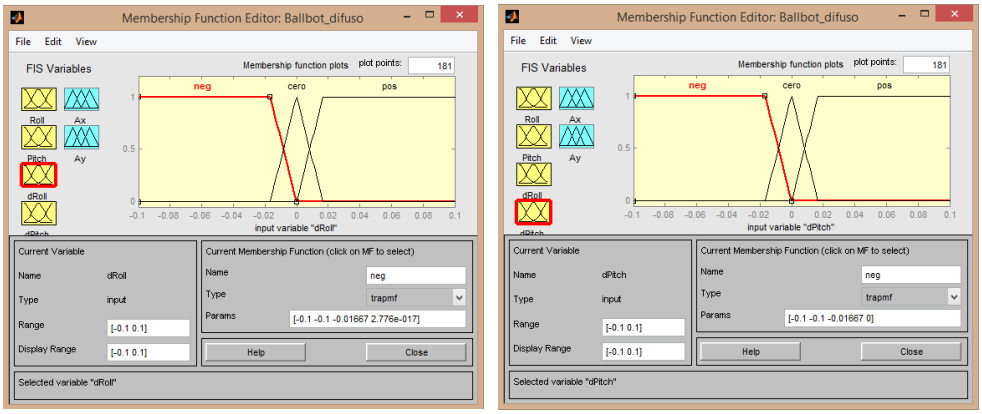


Figura A.17 Ventanas de diseño de las variables de entrada droll y dpitch

En la Figura A.17 se observan las funciones de pertenencia para los conjuntos de entrada correspondientes a las velocidades angulares en cada plano. En este caso las funciones de pertenencia para ambos se componen de una función “z”, una función “s” y una función triangular. El universo de discurso al igual que en los conjuntos anteriores se propueso de -0.1 a 0.1 rad/s y el valor máximo de cada función es de 1.

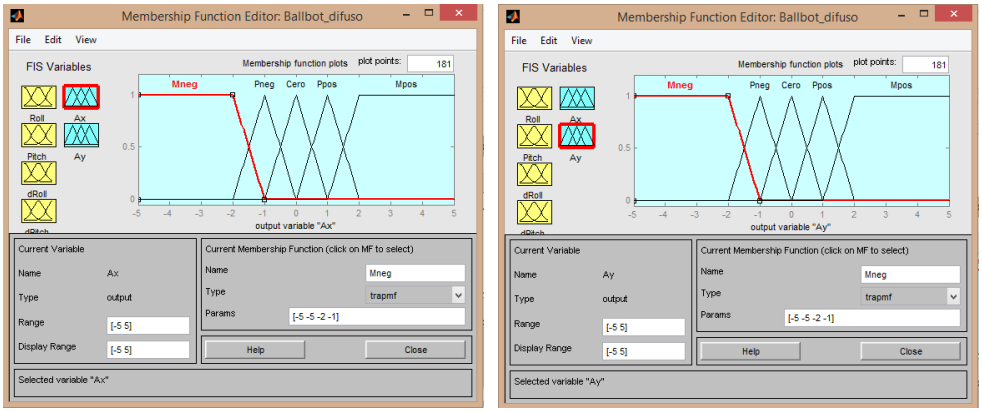


Figura A.18 Ventanas de diseño de las variables de entrada roll y pitch

En la Figura A.18 se pueden observar las ventanas de diseño de las salidas del controlador. Corresponden a los valores de las aceleraciones angulares en cada uno de los planos del robot y se componen de la misma cantidad de funciones de pertenencia que las entradas del controlador. En este caso el universo de discurso va de -5 a 5 rad/s^2.

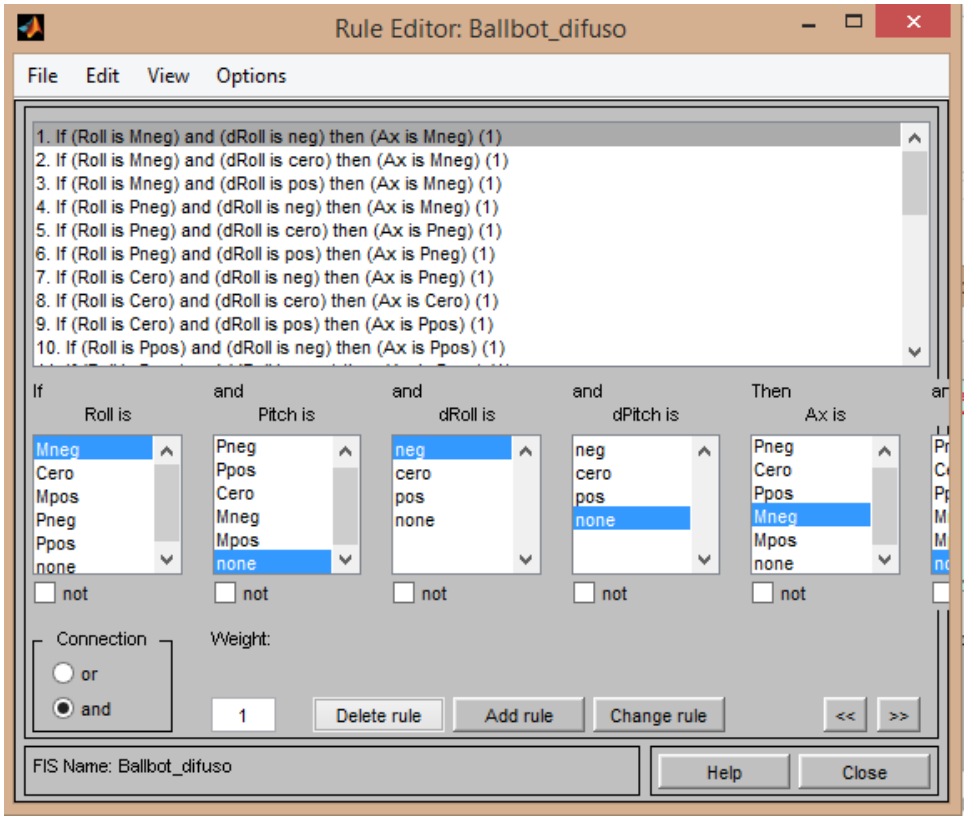


Figura A.19 Ventana de diseño de las reglas de la máquina de inferencia

Finalmente se diseñaron las reglas del controlador. Las mismas fueron propuestas por inferencia, es decir, intentando predecir cuál debería ser el comportamiento del sistema ante determinadas condiciones. La ventana de diseño de reglas se puede observar en la Figura A.19.

A.8.5 Script de gráficas de los resultados

Como parte del reporte escrito se tuvieron que graficar las simulaciones, a continuación se presenta el código empleado para esta tarea.

```
plot(ControlP(:,1),ControlP(:,2),'red','LineWidth',3)
legend('Control P');
xlabel('Tiempo [s]'); ylabel('Ángulo [rad]');
axis([0 1 -0.1 0.1]);
grid on;
hold on
%figure
plot(ControlPD(:,1),ControlPD(:,2),'green','LineWidth',3)
legend('Control PD');
xlabel('Tiempo [s]'); ylabel('Ángulo [rad]');
axis([0 1 -0.1 0.1]);
grid on;
hold on
%figure
```

```
plot(SS(:,1),SS(:,2),'black','LineWidth',3)
legend('Realimentación de Estados');
xlabel('Tiempo [s]'); ylabel('Ángulo [rad]');
axis([0 1 -0.1 0.1]);
grid on;
hold on
%figure
plot(Difuso(:,1),Difuso(:,2),'blue','LineWidth',3)
legend('Control     P','Control     PD','Realimentación     de
Estados','Control Difuso');
%legend('Control Difuso');
xlabel('Tiempo [s]'); ylabel('Ángulo [rad]');
axis([0 1 -0.1 0.1]);
grid on;
hold on
```

A.8.6 Código de control del Arduino

El código de control del robot se agregó al código con el que se realizó la estimación de la posición angular del robot. El código que se sugiere en la página de Sparkfun donde se puede adquirir la IMU es el Razor_AHRS. El código de obtención de la posición angular emplea un algoritmo DCM (Direction Cosine Matrix por sus siglas en inglés). Por defecto el algoritmo funciona con una frecuencia de 50 Hz, que depende del tiempo de refresco en el envío de datos de los sensores. Dicho parámetro es enviado a los sensores por medio del protocolo de comunicación I2C cuando se inicializa el dispositivo. En el código propuesto este parámetro se modificó a 200 Hz para que se pudieran tomar datos nuevos del sensor con esta frecuencia. Además de ello se modificó el tiempo de integración del giroscopio, que se emplea en el algoritmo.

Para conservar una mejor organización del código se buscó preservar el código de la estimación como se encontraba dividido en varios archivos de Arduino. Es por ello que se agregaron dos archivos nuevos que incluían el código del control y el código del filtro de las señales obtenidas. En la Figura A.20 se observa una captura de pantalla de la ventana de programación de Arduino.

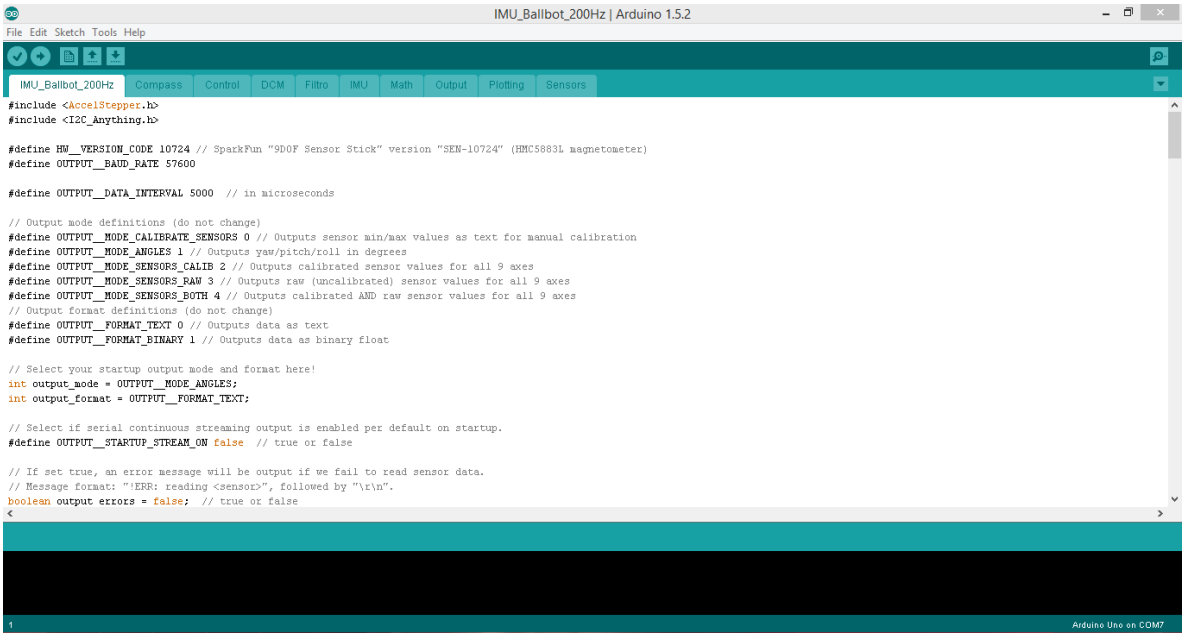


Figura A.20 Interfaz de Arduino con el programa empleado.

Finalmente se presentan algunos fragmentos de código que pueden servir para futuras implementaciones.

A.8.6.1 Sketch Control.ino

```
void Control(){

  //Algoritmo de Control PD
  Ax=-(Kp*roll_f + Kd*vroll_f)/resf;
  Ay=-(Kp*pitch_f + Kd*vpitch_f)/resf;

  //Actualizacion de las velocidades
  Vx=Ax*(0.007)/100;
  Vy=Ay*(0.007)/100;
  w=0;

  Wire.beginTransmission(4);
  I2C_writeAnything(-Vy);
  I2C_writeAnything(Vx);
  Wire.endTransmission();
}
```

A.8.6.2 Sketch Filtro.ino

```
float lpf_iir(float y_past, float x_now, float alpha)
{
 float value = x_now*(1-alpha)+alpha*y_past;
 return value;
}

void filtrado()
{
 vroll = GYRO_SCALED_RAD(gyro[0])+0.0024;
 vpitch = GYRO_SCALED_RAD(gyro[1])+0.011;

 //Filtro pasa bajas de 1° orden
 roll_f = lpf_iir(roll_f,roll,0.9);
 vroll_f = lpf_iir(vroll_f,vroll,0.9);
 pitch_f = lpf_iir(pitch_f,pitch,0.9);
 vpitch_f = lpf_iir(vpitch_f,vpitch,0.9);

 //Saturación de la salida del controlador
 if(roll_f > roll_sat)
  roll_f=roll_sat;
 if(pitch_f > pitch_sat)
  pitch_f=pitch_sat;
 if(vroll_f > vroll_sat)
  vroll_f=vroll_sat;
 if(vpitch_f > vpitch_sat)
  vpitch_f=vpitch_sat;
 if(roll_f < -roll_sat)
  roll_f=-roll_sat;
 if(pitch_f < -pitch_sat)
  pitch_f=-pitch_sat;
 if(vroll_f < -vroll_sat)
  vroll_f=-vroll_sat;
 if(vpitch_f < -vpitch_sat)
  vpitch_f=-vpitch_sat;
}
```

```
void imprimir()
{
  Serial.print(roll,5); Serial.print(",");
    Serial.print(roll_f,5); Serial.print(",");
  Serial.print(pitch,5);Serial.print(",");
    Serial.print(pitch_f,5); Serial.print(",");
  Serial.print(vroll,5); Serial.print(",");
    Serial.print(vroll_f,5); Serial.print(",");
  Serial.print(vpitch,5); Serial.print(",");
    Serial.print(vpitch_f,5);
  Serial.println();
}

void graficar()
{
  roll_imp=roll_f*100;
  pitch_imp=pitch_f*100;
  vroll_imp=vroll_f*100;
  vpitch_imp=vpitch_f*100;
  plot(roll_imp,roll*100,pitch_imp,pitch*100);
}
```

A.8.6.3 Sketch IMU_Ballbot_200Hz.ino (Principal)

```
#include <I2C_Anything.h>
#include <Wire.h>
#include <math.h>

//Varibles de configuración del programa Razor_ARHS.ino
.
.
.
.

// Datos de calibración del sensor
// Accelerometer
// "accel x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX  Z_MIN/Z_MAX"
#define ACCEL_X_MIN ((float) -271)
```

```
#define ACCEL_X_MAX ((float) 264)
#define ACCEL_Y_MIN ((float) -266)
#define ACCEL_Y_MAX ((float) 275)
#define ACCEL_Z_MIN ((float) -277)
#define ACCEL_Z_MAX ((float) 233)

// Magnetometer (extended calibration)
// Uncommend to use extended magnetometer calibration (compensates hard & soft iron errors)
#define CALIBRATION__MAGN_USE_EXTENDED true
const float magn_ellipsoid_center[3] = {59.9933, -39.2222, -848.900};
const float magn_ellipsoid_transform[3][3] = {{0.771573, -0.00651306, -0.0151888}, {-0.00651306, 0.791164, -0.0424512}, {-0.0151888, -0.0424512, 0.990528}};

// Gyroscope
// "gyro x,y,z (current/average) = .../OFFSET_X  .../OFFSET_Y  .../OFFSET_Z
#define GYRO_AVERAGE_OFFSET_X ((float) -11.99)
#define GYRO_AVERAGE_OFFSET_Y ((float) 50.4)
#define GYRO_AVERAGE_OFFSET_Z ((float) -12.75)
#define TO_RAD(x) (x * 0.01745329252)  // *pi/180
#define TO_DEG(x) (x * 57.2957795131)  // *180/pi

//Variables del programa de control
.
.
.
.

//Constantes del control
float Kp=800;
float Kd=200;

void setup()
{
 /*Función de configuración de la IMU.
Corresponde a la función setup del algoritmo
Razor_ARHS */
```

```
  setupIMU();
}

void loop()
{
/*Función principal de la IMU.
Corresponde a la función loop del algoritmo
Razor_ARHS */
  loopIMU();
}
```

A.8.6.4 Sketch IMU.ino

Contiene las funciones del algoritmo Razor_ARHS, en esta función se encuentra el loop principal, donde se ejecutan las funciones de control y filtrado, además de gráficas e impresión de ángulos.

```
void setupIMU()
{
  // Init serial output
  Serial.begin(OUTPUT__BAUD_RATE);

  // Init status LED
  pinMode (STATUS_LED_PIN, OUTPUT);
  digitalWrite(STATUS_LED_PIN, LOW);

  // Init sensors
  delay(50);  // Give sensors enough time to start
  I2C_Init();
  Accel_Init();
  Magn_Init();
  Gyro_Init();

  // Read sensors, init DCM algorithm
  delay(20);  // Give sensors enough time to collect data
  reset_sensor_fusion();
}
```

```
// Main loop
void loopIMU()
{
  if((micros() - timestamp) >= OUTPUT__DATA_INTERVAL)
  {
    timestamp_old = timestamp;
    timestamp = micros();
    if (timestamp > timestamp_old)
      G_Dt = (float) (timestamp - timestamp_old) / 1000000.0f; // Real time of
loop run. We use this on the DCM algorithm (gyro integration time)
    else G_Dt = 0;

    // Update sensor readings
    read_sensors();

      // Run DCM algorithm
      Compass_Heading(); // Calculate magnetic heading
      Matrix_update();
      Normalize();
      Drift_correction();
      Euler_angles();
      filtrado();
      Control();
      //imprimir();
      //graficar();
}
```

A.9 Diseño de los filtros pasa bajas

Dado que es fundamental la inclusión de un filtro para obtener una mejor estimación se diseñaron tres alternativas distintas para mitigar este problema. MATLAB cuenta con una caja de herramientas o *toolbox* que simplifica el diseño de filtros, por lo cual se empleó en este trabajo.

En la primer parte del código se diseñó un filtro pasa bajas de primer orden. En este caso al ser de primer orden sólo se tiene una constante que calcular, por ende la frecuencia de corte del filtro está directamente relacionada con

el valor de esta constante. Si la misma aumenta, la atenuación es mayor y la frecuencia de corte disminuye, lo mismo ocurre con el caso opuesto.

En la segunda parte del código se obtuvieron las constantes del filtro butterworth de segundo orden. Estas constantes son los coeficientes de la ecuación en diferencias del filtro. A continuación se presenta el código utilizado.

Código de MATLAB

```
clc
 clear
%%Diseño del filtro pasa bajas de primer orden
  disp('Constante del filtro pasa bajas de primer orden')
  disp('Filtro de la forma Y[k]=(alpha-1)*X[k]+alpha*Y[k-1]')
%Tiempo de muestreo
 Ts=0.007;
%Frecuencia de corte del filtro en Hz
 Fc=2.52;
%Constante de tiempo del filtro:
tau=1/(2*pi*Fc);
%Obteniendo alpha
alpha=tau/(tau+Ts)

%%Diseño del filtro butterworth pasa bajas
 clear
 disp('Filtro butterworth de segundo orden')
 disp('Filtro  de  la  forma  Y[k]=a1*X[k]+a2*X[k-1]+a3*X[k-
2]+a4*Y[k-1]+a5*Y[k-2]')
%Frecuencia de corte del filtro en Hz
 Fc=2;
%Frecuencia de muestreo en Hz
 Fs=142;
%Frecuencia de Nyquist
 Wn=Fc/(Fs/2);
%Orden del filtro diseñado
 Order=2;
%Obtención de la función de transferencia del filtro
 [z,p,k]=butter(Order,Wn);
%Función de transferencia final del filtro
 H = zpk(z,p,k,1/Fs)
 num=k*poly(z);
 den=poly(p);
 a1=num(1)
 a2=num(2)
 a3=num(3)
 a4=-den(2)
 a5=-den(3)
```

El resultado de la ejecución de este código se observa en las Figuras A.21 y A.22.

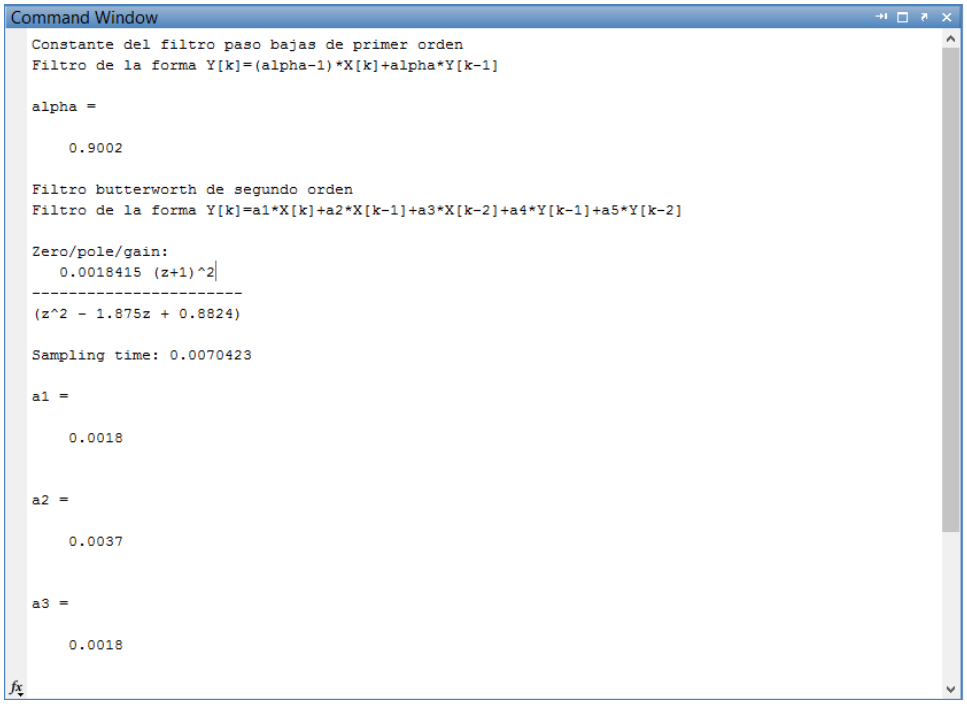


Figura A.21 Ventana de comandos después de la ejecución del código del diseño.

Figura A.22 Constantes obtenidas para la implementación del filtro butterworth

Finalmente con un análisis simple se optó por implementar un filtro de media móvil. El diseño del filtro es trivial de cierto modo, dado que sólo se debe elegir un umbral para estimar el valor promedio de la señal dentro de

esa ventana. El valor de la dimensión del arreglo empleado fue de 50 elementos.

Desempeño de los filtros

Dado que evaluar el desempeño en tiempo real de cada uno de los filtros resulta ser muy lento, se procedió a guardar una serie de mediciones ruidosas y con perturbaciones en un archivo de datos para posteriormente evaluar el desempeño del filtro diseñado implementando la ecuación en diferencias y graficándola junto con las mediciones.

A continuación se presentan los resultados gráficos de cada uno de los filtros diseñados.

En la Figura A.23 se observa el desempeño del filtro pasa bajas de primer orden con frecuencia de corte de 2.52 Hz correspondiente a un valor de alpha de 0.9. En la Figura A.24 se observa el funcionamiento del filtro butterworth de segundo orden con una frecuencia de corte de 2 Hz. Finalmente, en la Figura A.25 se muestra el resultado del filtro de media móvil con un arreglo de 50 elementos.

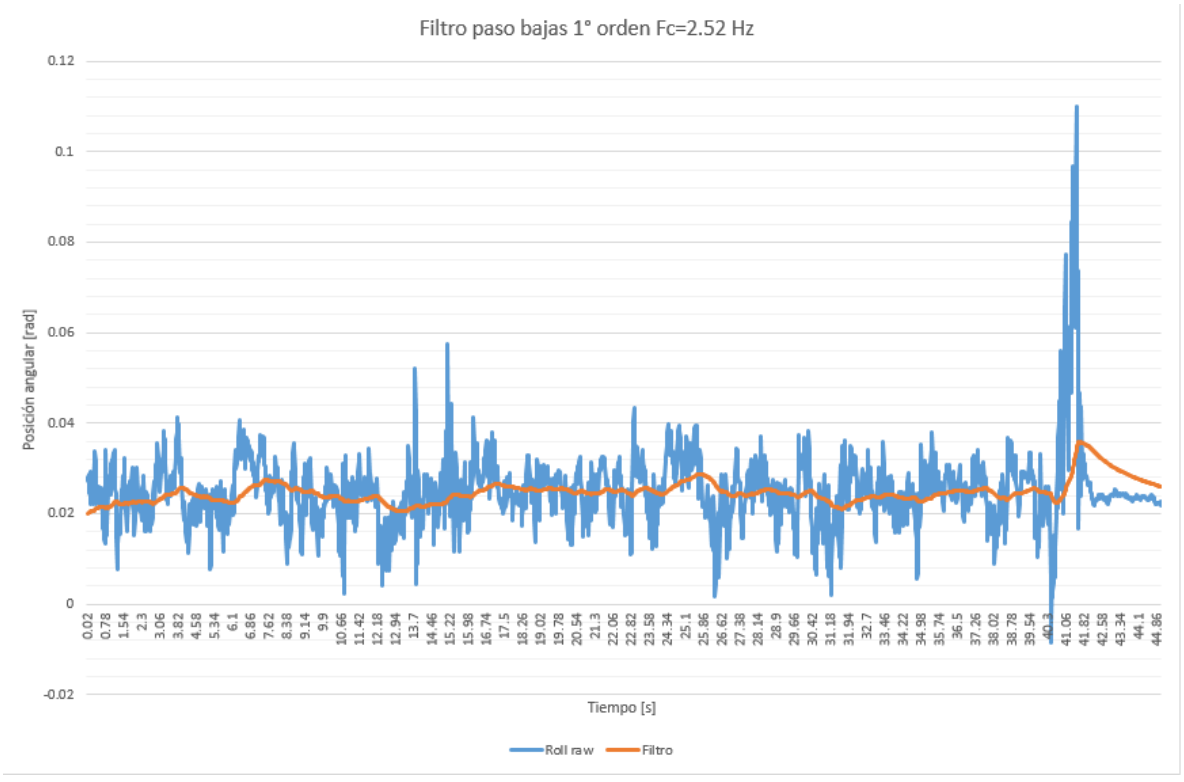


Figura A.23 Desempeño del filtro pasa bajas de primer orden

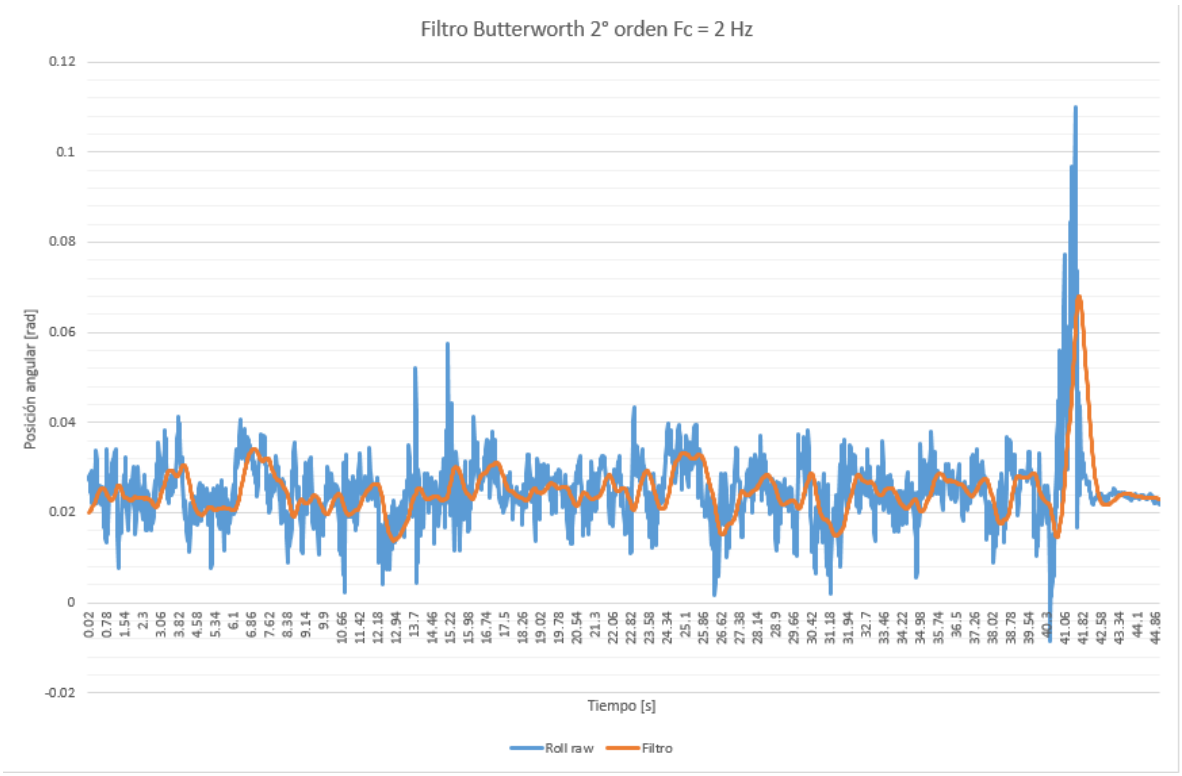


Figura A.24 Desempeño del filtro butterworth de segundo orden

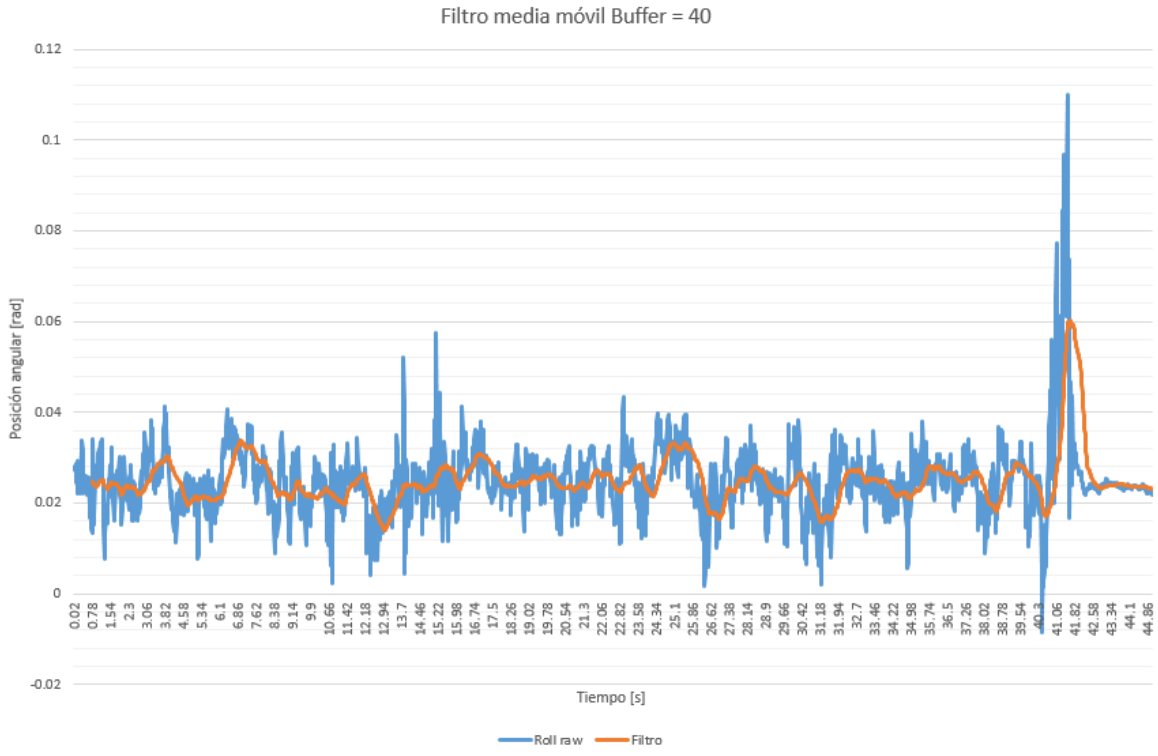
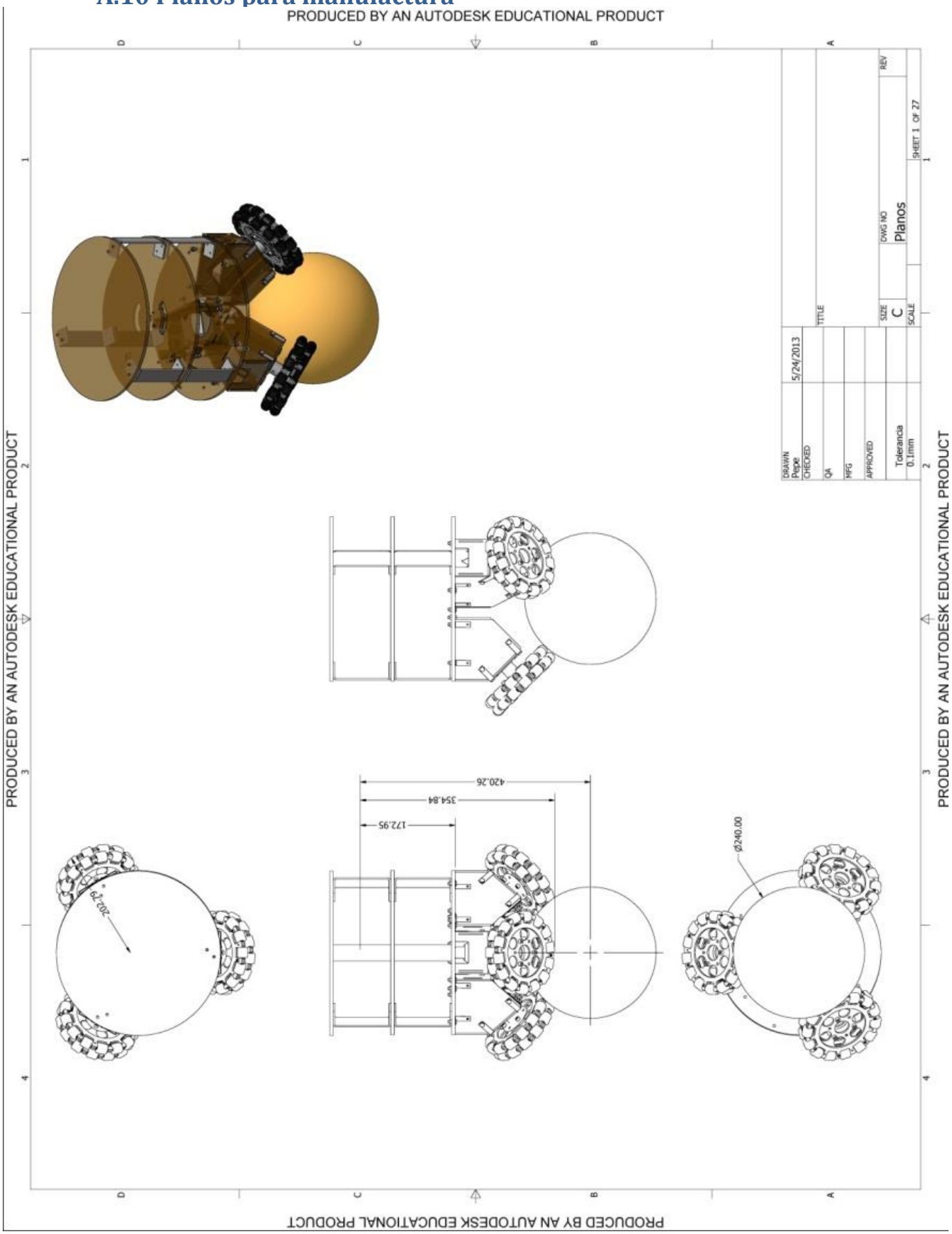
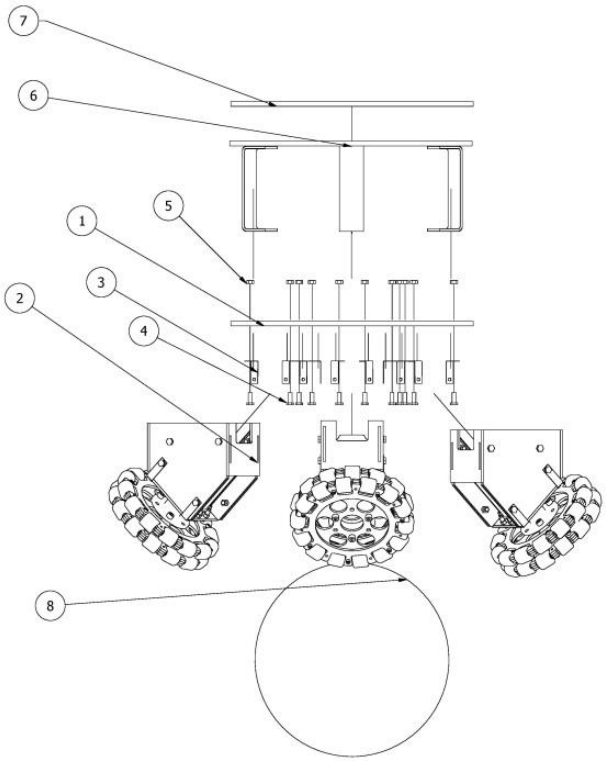


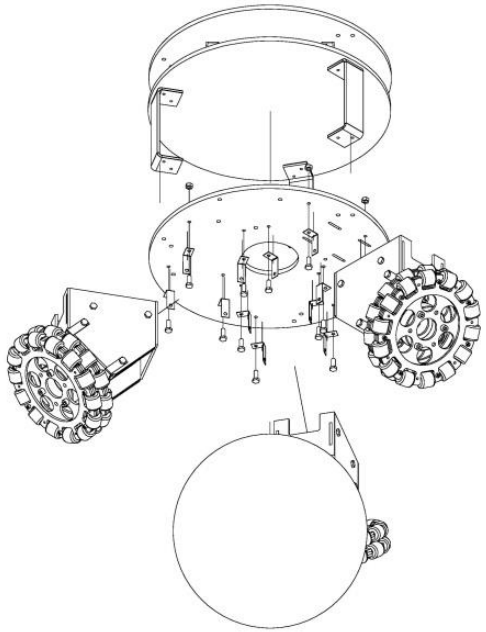
Figura A.25 Desempeño del filtro de media móvil

A.10 Planos para manufactura

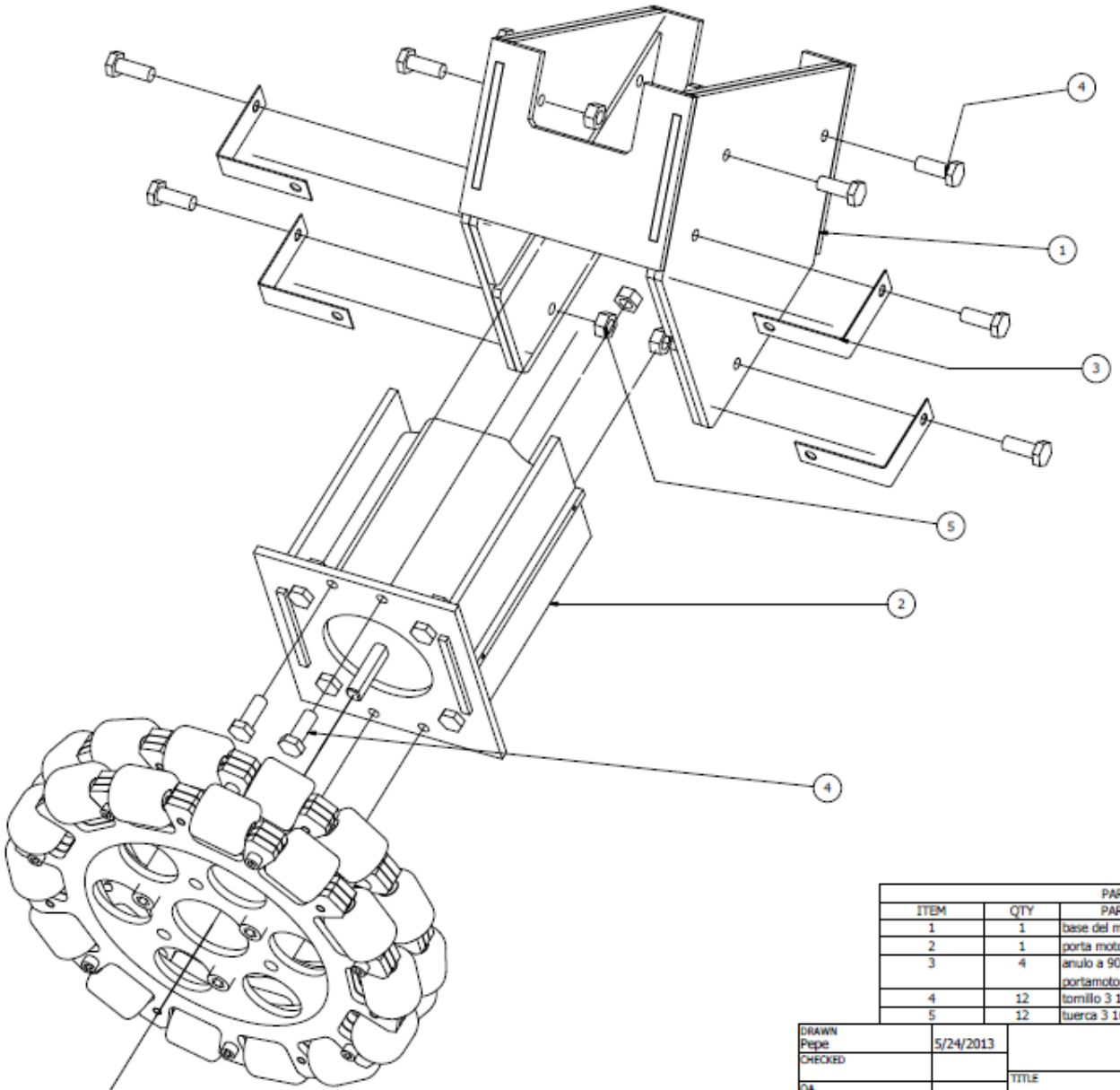


PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	base del robot	
2	3	ansamble porta motor y base	
3	12	anulo a 90 acople base con portamotor	
4	12	tornillo 3 16 1 pulgada	
5	12	tuerca 3 16	
6	6	separador	
7	1	segundo piso	
8	1	esfera basquetbol	
9	1	tercer piso	



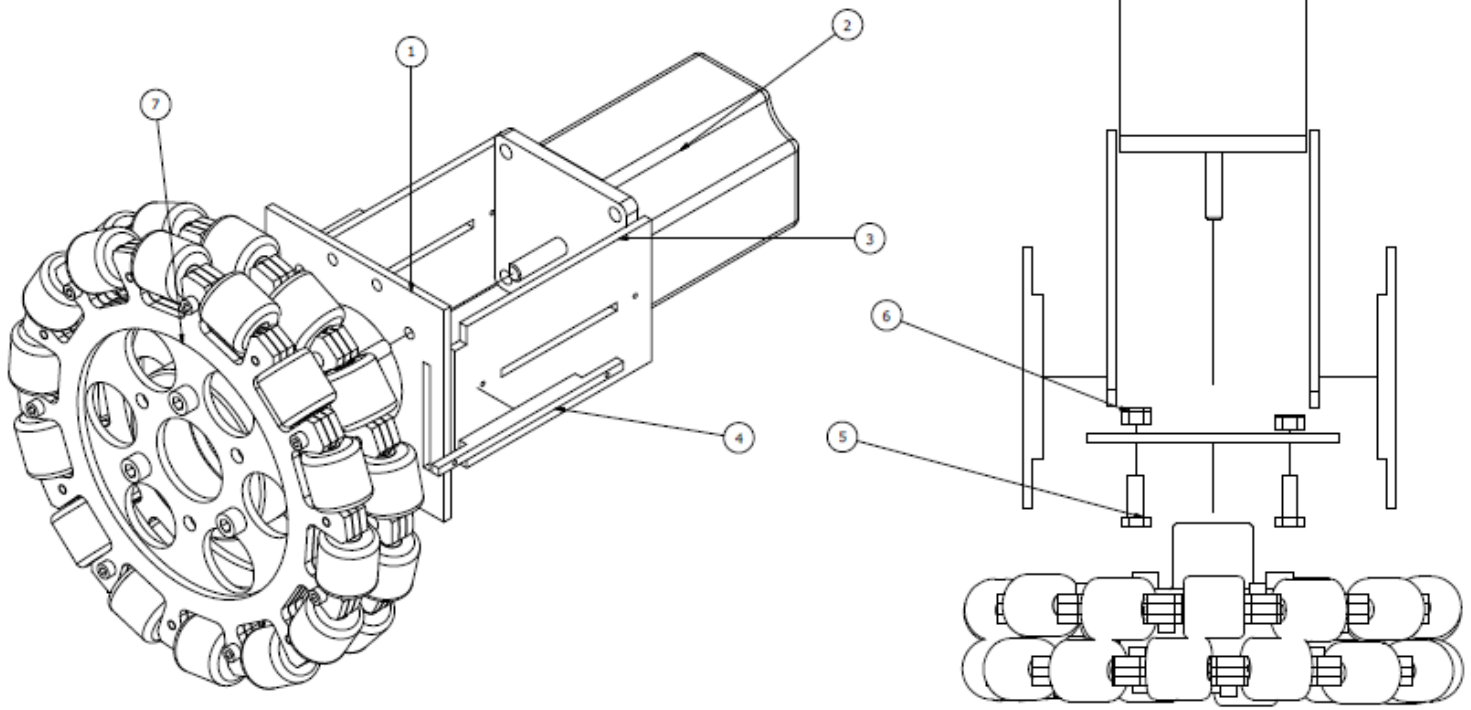


DRAWN Pepe	5/24/2013		
CHECKED		TITLE	
QA			
MFG			
APPROVED			
Tolerancia 0.1mm		SIZE C	DWG NO Planos
		SCALE	SHEET 2 OF 27



PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	base del motor	
2	1	porta motor	
3	4	anulo a 90 acople portamotor y base	
4	12	tornillo 3 16 1 pulgada	
5	12	tuerca 3 16	

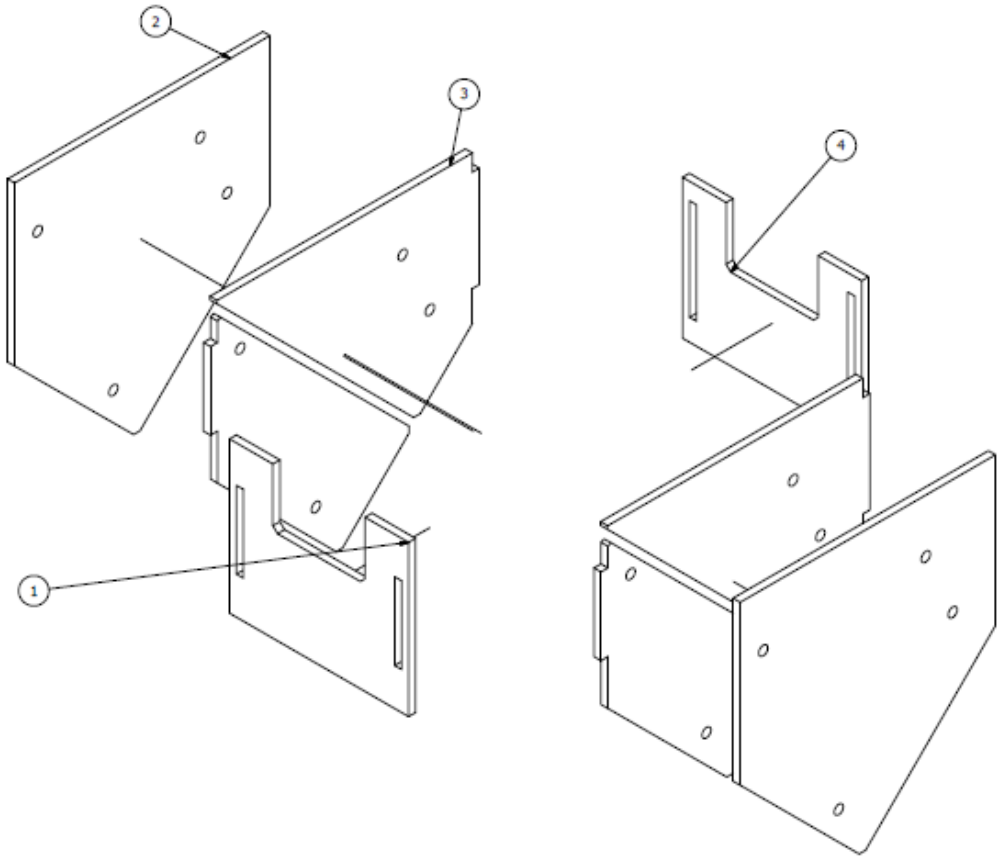
DRAWN Pepe	5/24/2013			
CHECKED		TITLE		
QA				
MFG				
APPROVED				
Tolerancia 0.1mm		SIZE C	DWG NO Planos	REV
		SCALE	SHEET 3 OF 27	



PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	caratula motor	
2	1	Motor	
3	2	placa lateral del motor	
4	2	costilla lateral de la placa	
5	4	tornillo 3 16 1 pulgada	
6	4	tuerca 3 16	
7	1	ensamble llanta doble	

DRAWN Pepe	5/24/2013		
CHECKED		TITLE	
QA			
MFG			
APPROVED		SIZE C	DWG NO Planos REV
Tolerancia 0.1mm		SCALE	SHEET 7 OF 27

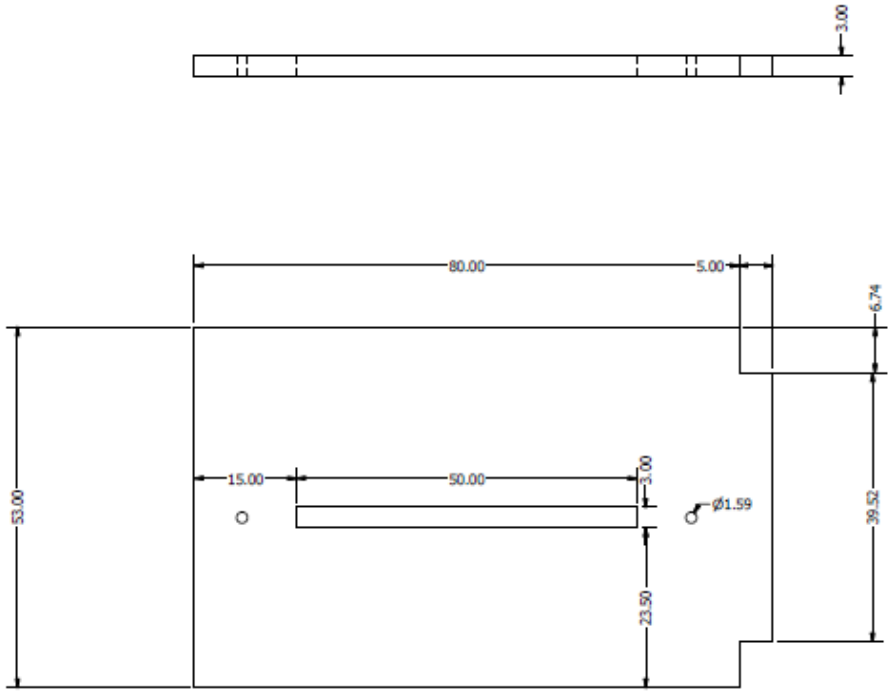
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT



PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	pared 1 placa lateral del montaje del motor	
2	2	placa lateral de el montaje del motor	
3	2	placa lateral de el montaje del motor con riel	
4	1	pared 2 placa lateral del montaje del motor	

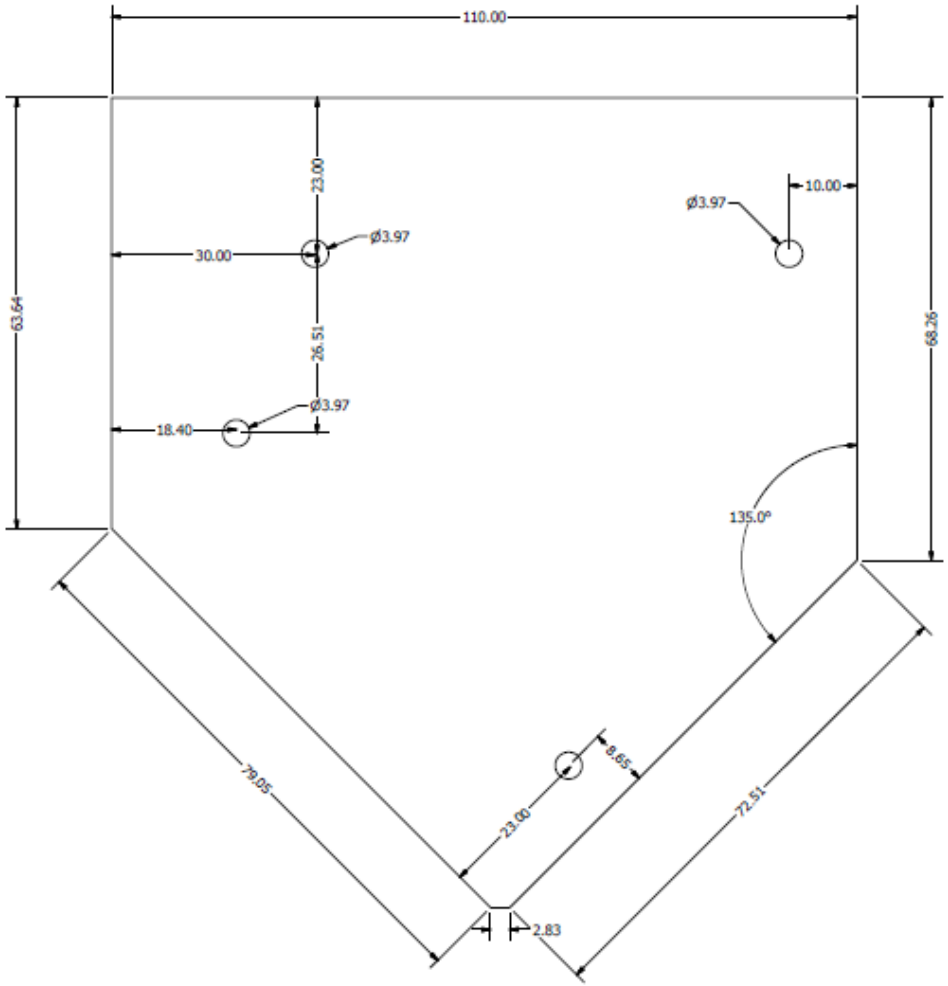
DRAWN Pepe	5/24/2013	
CHECKED		TITLE
QA		
MFG		
APPROVED		
Tolerancia 0.1mm		SIZE C / DWG NO Planos / REV
		SCALE / SHEET 8 OF 27

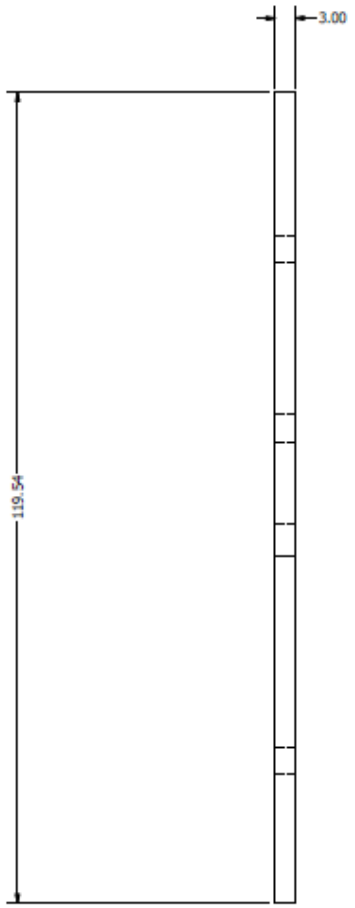
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT



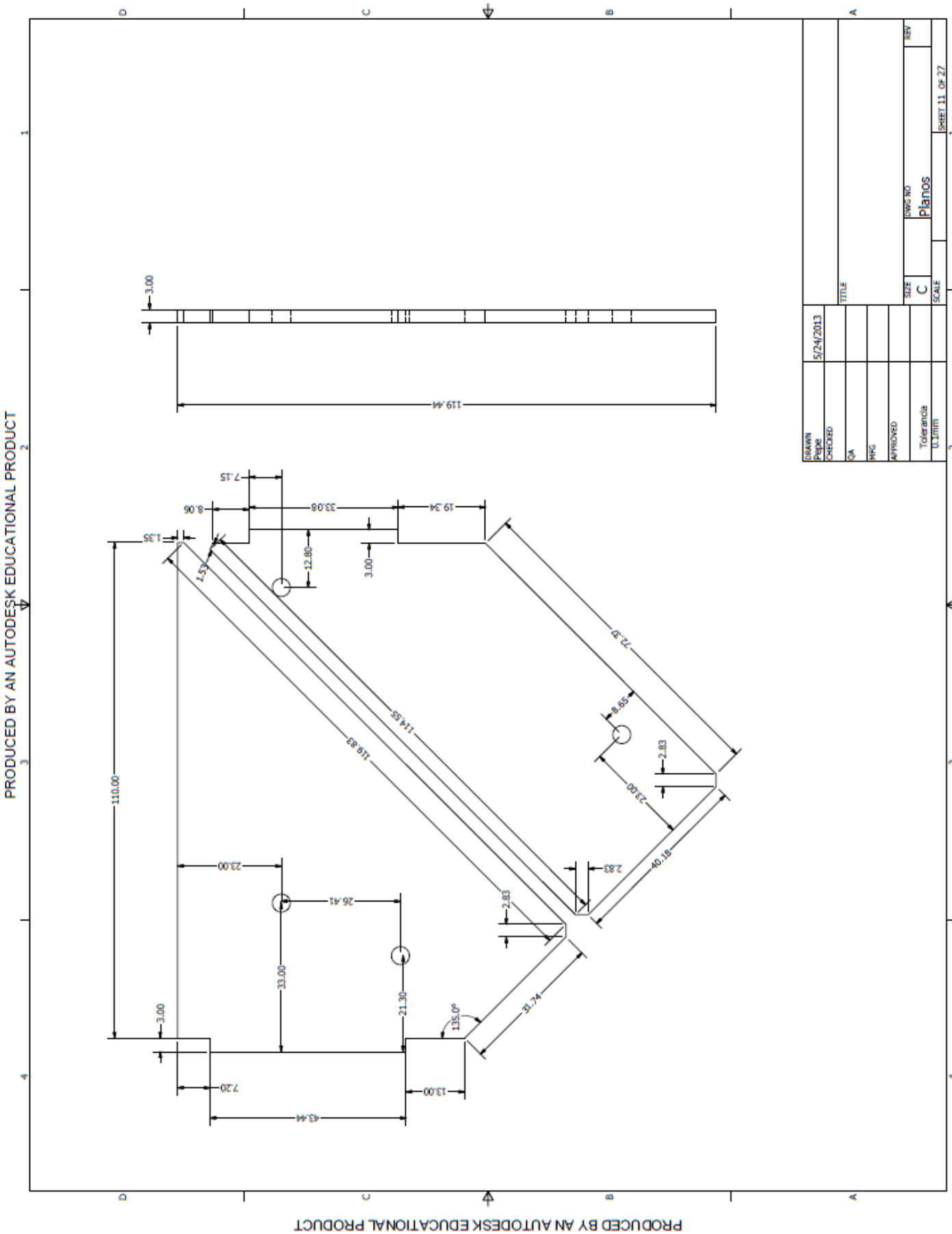
DRAWN Pepe	5/24/2013	
CHECKED		TITLE
QA		
MFG		
APPROVED		
Tolerancia 0.1mm		SIZE C / DWG NO Planos / REV / SCALE / SHEET 9 OF 27

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

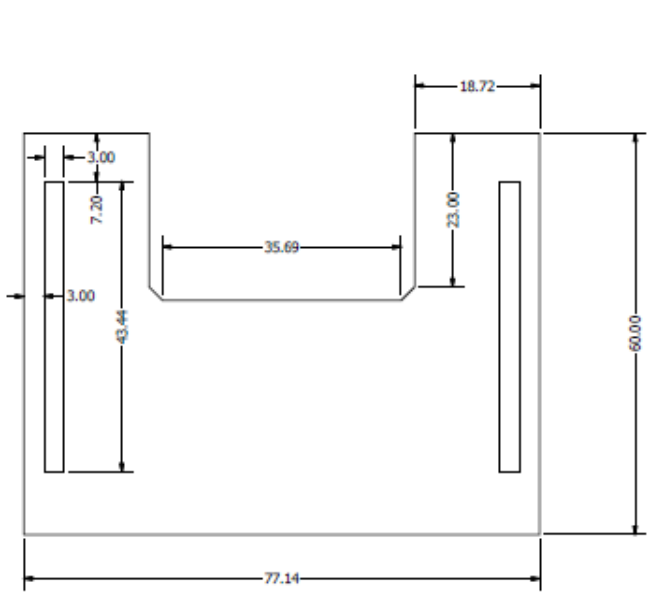




DRAWN Pepe	5/24/2013	
CHECKED		TITLE
QA		
MFG		
APPROVED		
Tolerancia 0.1mm		SIZE C / DWG NO Planos / REV / SCALE / SHEET 10 OF 27

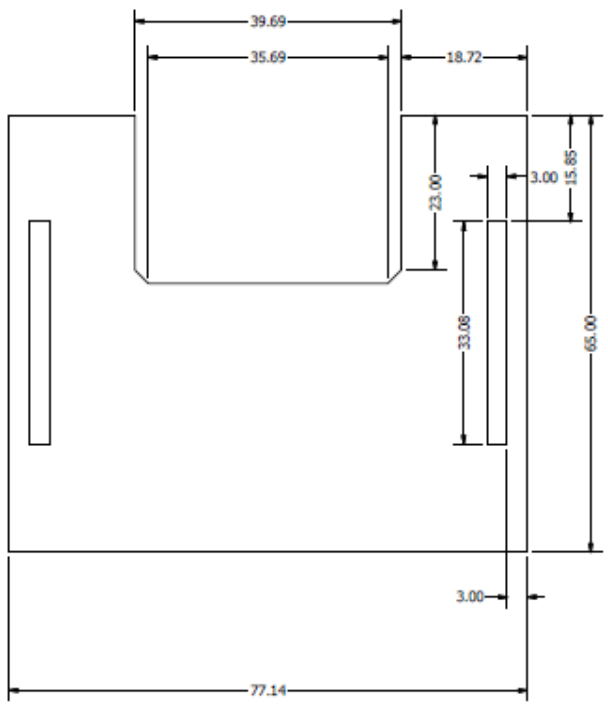


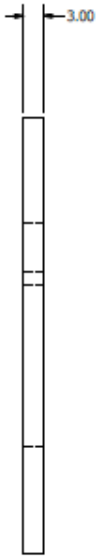
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT



DRAWN Pepe	5/24/2013			
CHECKED		TITLE		
QA				
MFG				
APPROVED				
Tolerancia 0.1mm		SIZE C	DWG NO Planos	REV
		SCALE	SHEET 12 OF 27	

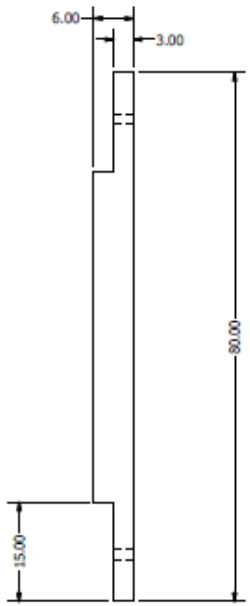
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

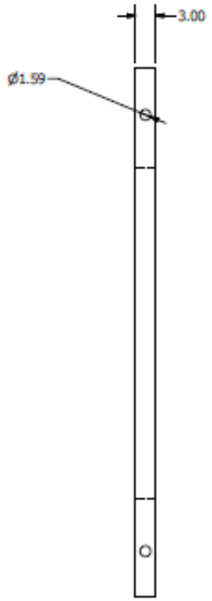




DRAWN Pepe	5/24/2013			
CHECKED		TITLE		
QA				
MFG				
APPROVED		SIZE C	DWG NO Planos	REV
Tolerancia 0.1mm		SCALE	SHEET 13 OF 27	

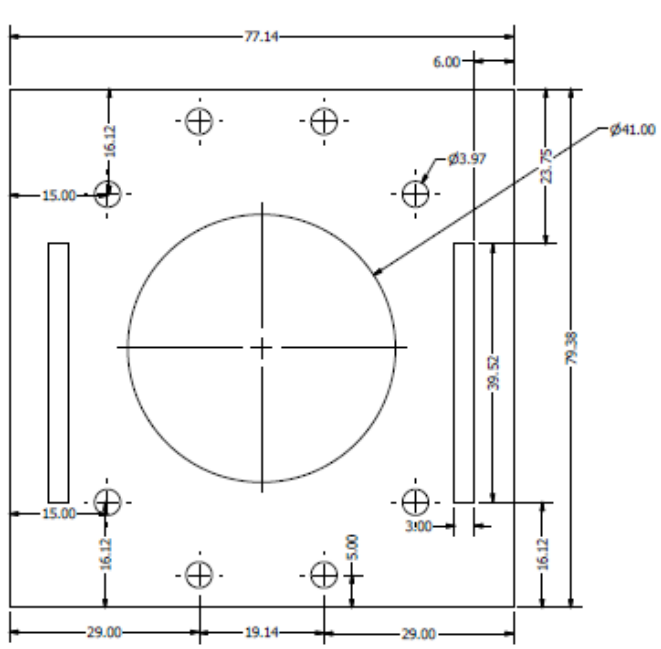
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

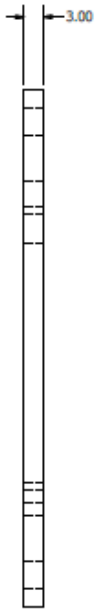




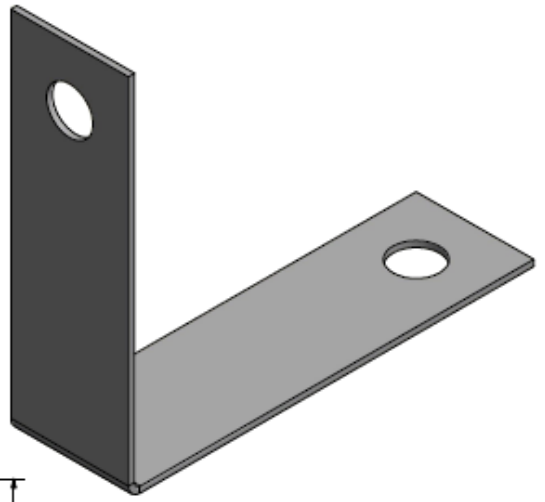
DRAWN Pepe	5/24/2013			
CHECKED		TITLE		
QA				
MFG				
APPROVED				
Tolerancia 0.1mm		SIZE C	DWG NO Planos	REV
		SCALE	SHEET 14 OF 27	

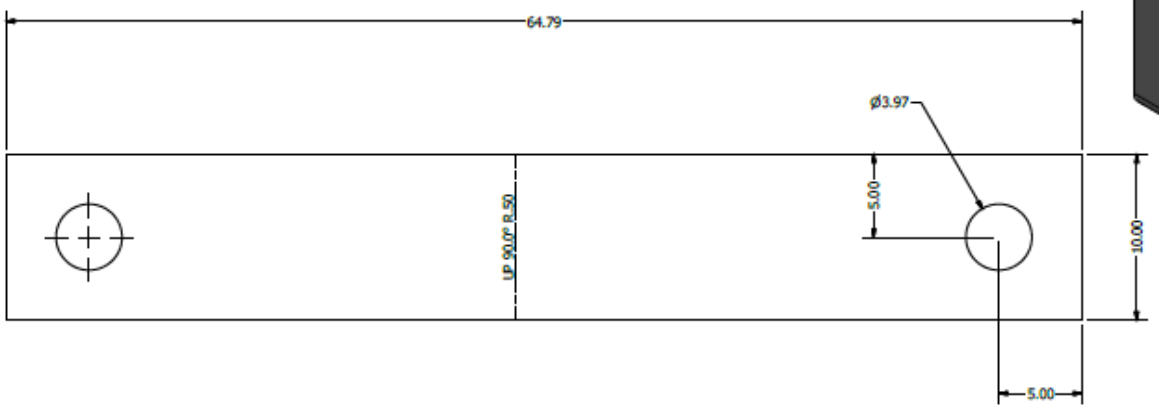
PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT





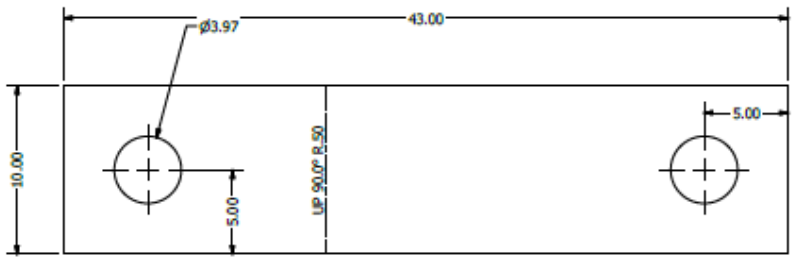
DRAWN Pepe	5/24/2013			
CHECKED		TITLE		
QA				
MFG				
APPROVED				
Tolerancia 0.1mm		SIZE C	DWG NO Planos	REV
		SCALE	SHEET 15 OF 27	

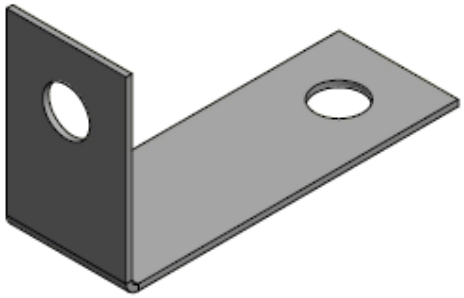


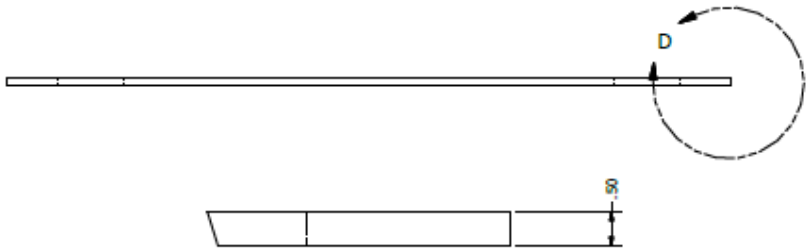


DETAIL C
SCALE 10 : 1

DRAWN Pepe	5/24/2013		
CHECKED		TITLE	
QA			
MFG			
APPROVED		SIZE C	DWG NO Planos REV
Tolerancia 0.1mm		SCALE	SHEET 16 OF 27



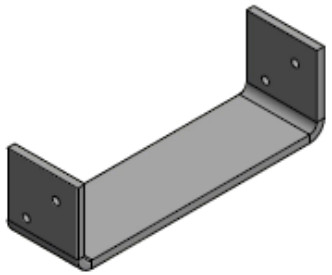


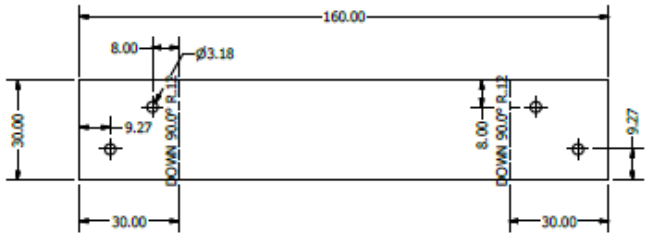


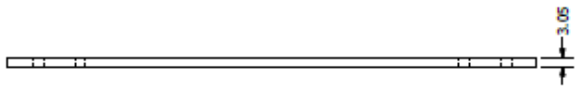
DETAIL D
SCALE 20 : 1

DRAWN Pepe	5/24/2013	
CHECKED		
QA		TITLE
MFG		
APPROVED		
Tolerancia 0.1mm		SIZE C / DWG NO Planos / REV / SCALE / SHEET 17 OF 27

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT







DRAWN Pepe	5/24/2013	
CHECKED		TITLE
QA		
MFG		
APPROVED		
Tolerancia 0.1mm		SIZE C / DWG NO Planos / REV / SCALE / SHEET 4 OF 27

REFERENCIAS

[1] B. García y Y. Minami, “Diseño, construcción y control de estabilidad de un robot que se balancea sobre una esfera”, Facultad de Ingeniería, UNAM, Tesis de licenciatura, 2012.

[2] T. B. Lawers, G. A. Kantor, y R. L. Hollis, “A dinamically stable single-wheeled mobile robot with inverse mouse-ball drive”, IEEE Int’l. Conf. on Robotics and Automation, Orlando, FL, Mayo 15-19, 2006.

[3] M. Kumagai y T. Ochiai, “Development of a Robot Balanced on a Ball - First Report, Implementation of the Robot and Basic Control”, Tohoku Gakuin University, 2009.

[4] J. Fong y S. Uppilletz, “Ballbot - Preliminary report”, University of Adelaide, 2009.

[5] P. Fankhauser y C. Gwerder, “Modelling and control of a Ballbot”, Swiss Federal Institute of Technology Zurich, Bachelor Tesis, 2010.

[6] P. T. Hernández, “Control de robots móviles: el caso del Ballbot”, Facultad de Ingeniería UNAM, Tesis de licenciatura, 2010.

[7] B. García y Y. Minami,”Aplicación de una unidad de medición inercial en tiempo real para el control de un ballbot”,SOMI XXVII Congreso de Instrumentación, Cualiacán, Sinaloa, Octubre, 2012.

[8] Sparkfun (2013, Junio 24). Degrees of Freedom – Sensor Stick [En línea].

Disponible en: https://www.sparkfun.com/products/10724

[9] AndyMark (2013, Junio 24). 6" Aluminum Dualie Omni Wheel (am-0432) [En línea]. Disponible en: http://www.andymark.com/product-p/am-0432.htm

[10] HobbyKing (2013, Junio 24). Turnigy 3000mAh 3S 30C Lipo Pack [En línea].

Disponible en: http://www.hobbyking.com/hobbyking/store/__9497__turnigy_3000mah_3s_30c_lipo_pack.html

[11] Sparkfun (2014, Febrero 14). Quadstepper Motor Driver Board [En línea].

Disponible en: https://www.sparkfun.com/products/10507

[12] Arduino (2013, Junio 24). Arduino Uno y Caracteristicas del Arduino Uno [En línea].

Disponible en: http://arduino.cc/en/Main/ArduinoBoardUno

[13] Arduino (2013, Junio 24). Arduino Due y Caracteristicas del Arduino Due [En línea].

Disponible en: http://arduino.cc/en/Main/ArduinoBoardDue

[14] Holonomic Motion Control (2013, Agosto 9). Movimiento Holonómico [En línea].

Disponible en: http://www.cs.cmu.edu/~pprk/physics.html

[15] Mike McCauley (2014, Febrero 14). AccelStepper [En línea].

Disponible en: http://www.airspayce.com/mikem/arduino/AccelStepper/

[16] Nick Gammon (2014, Febrero 14). I2C_Anything [En línea].

Disponible en: http://www.gammon.com.au/forum/?id=10896

[17] Negtronics (2014, Febrero 14). SimPlot[En línea].

Disponible en: http://www.negtronics.com/simplot