PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Roberto Felipe Chacón Espinosa

Modelado, Simulación y Control de Robots Esféricos

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 4 de Enero de 2019

Modelado, Simulación y Control de Robots Esféricos

Roberto Felipe Chacón Espinosa

Informe Final para optar al título de Ingeniero Civil Electrónico,
aprobada por la comisión de la
Escuela de Ingeniería Eléctrica de la
Facultad de Ingeniería de la
Pontificia Universidad Católica de Valparaíso
conformada por

Sr. Gonzalo Alberto Farías Castro
Profesor Guía

Sr. Héctor Vargas Oyarzún
Segundo Revisor

Sr. Sebastián Fingerhuth Massmann
Secretario Académico

Valparaíso, 4 de Enero de 2019

A mi madre que de no ser por ella, ya me hubiera rendido hace mucho tiempo.

Agradecimientos

Quiero agradecer a mi profesor Guía y segundo revisor por el soporte que me dieron al realizar el proyecto de tesis, a mi madre que me apoyo en los momentos difíciles y su amor incondicional de una madre.

También a mis compañeros y amigos que se tomaron el tiempo en explicarme partes de la materia de algún ramo que tenía problema, la paciencia y las ganas de ayudarme que me demostraron fue mucho mayor de lo que yo esperaba.

Valparaíso, 4 de Enero de 2019

Roberto Felipe Chacón Espinosa

Resumen

Este proyecto tiene como objetivo crear un robot esférico móvil parecido al robot de StarWars bb-8, para esto se tuvo que utilizar diferentes herramientas, como los diferentes módulos, placas de programación, componentes electrónicos y lenguajes de programación que se fueron adquiriendo en el transcurso de la carrera.

El investigar los diferentes tipos de mecanismos de movimientos que se necesitaron para que el robot pudiera desplazarse de un lugar a otro fue esencial, el resultado de esto fue la elección de un mecanismo que no fuese difícil de elaborar y tomando también la disponibilidad de los materiales para la fabricación.

Se optó por utilizar el software V-Rep que permite la simulación de un robot. Como resultado de las simulaciones realizadas se logró que el robot se desplazara sin problemas e incluso en pasar por encima de objetos. El diseño del robot se realizó para probar si el mecanismo elegido fue el correcto en términos de factibilidad de funcionamiento, se simulo en el software V-Rep el cual es un software de simulación de robot. El resultado de la simulación fue que el robot pudo desplazarse sin problemas por lo que de esta manera se reafirmó la elección del mecanismo de movimiento. El diseño del robot se realizó en el software Autodesk Fusion 360, el cual es un software bastante usado para modelamiento de figura en 3D.

La construcción de la estructura del robot se realizó con una impresora 3D y con plástico PLA usando el diseño realizado en el software Autodesk Fusion 360, los componentes se obtuvieron sin ningún problema por lo que se armó el armo el robot según como estaba diseñado. Se realizaron pruebas de movimiento con el objetivo de probar su funcionalidad con respecto a lo que se había simulado.

En el control del robot en un principio se utilizó un celular con una aplicación IOT, pero luego se requirió que el robot fuese autónomo en su movimiento, es decir, dado un punto de coordenadas objetivo el robot debía llegar a este punto de coordenadas sin ayuda. Se realizaron pruebas de control del robot mediante una plataforma de pruebas de robot, esta plataforma se componía de un área de prueba 2x1 metros aproximadamente, un computador para enviar los datos relevantes al robot y una cámara ubicada a lo alto encima del área de prueba y mirando hacia abajo.

Palabras claves: Robot esférico móvil, plataforma de prueba, Ejs, Swistrack, simulaciones.

Abstract

This project aims to create a mobile spherical robot similar to the StarWars bb-8 robot, for this it had to use different tools, such as the different modules, programming boards, electronic components and programming languages that were acquired in the course of the career.

Investigating the different types of movement mechanisms that were necessary for the robot to move from one place to another was essential, the result of this was the choice of a mechanism that was not difficult to elaborate and also taking into account the availability of the materials for manufacturing.

It was decided to use the V-Rep software that allows the simulation of a robot. As a result of the simulations carried out, the robot was able to move smoothly and even pass over objects. The design of the robot was made to test if the chosen mechanism was the correct one in terms of feasibility of operation, it was simulated in the software V-Rep which is a robot simulation software. The result of the simulation was that the robot was able to move without problems so that in this way the choice of movement mechanism was reaffirmed. The design of the robot was made in the software Autodesk Fusion 360, which is a software quite used for 3D figure modeling.

The construction of the robot structure was carried out with a 3D printer and with plastic PLA using the design made in Autodesk Fusion 360 software, the components were obtained without any problem so the robot was assembled according to how it was designed. Motion tests were performed in order to test its functionality with respect to what had been simulated.

In the control of the robot at first a cell with an IOT application was used, but then the robot was required to be autonomous in its movement, that is, given a point of target coordinates the robot had to reach this point of coordinates without help. Robot control tests were carried out using a robot test platform, this platform consisted of a test area approximately 2x1 meters, a computer to send the relevant data to the robot and a camera located high above the test area and looking down.

Key words: Mobile spherical robot, test platform, Ejs, Swistrack, Simulations.

Índice general

Introducción 1
Objetivo general 3
Objetivos específicos 3

1 Estado del arte 4
1.1 Antecedentes generales y propuesta 4
1.1.1 Descripción del problema 4
1.1.2 Descripción de la propuesta 4
1.2 Robot esférico móvil 4
1.2.1 Objetivo del robot esférico móvil 5
1.2.2 Usos 5
1.3 Tipos de mecanismos de movimiento del robot esférico 5
1.3.1 Esfera de hámster 6
1.3.2 Unidad de accionamiento interno (IDU) 6
1.3.3 Péndulo conducido (pendulum driven) 7
1.3.4 Mejoras notables (notable enhancements) 8
1.3.5 Cambio de masa múltiple (Multiple-Mass-Shifting) 9
1.3.6 Cuerpo deformable 10
1.3.7 Single ball (Una sola bola) 11

2 Diseño y simulaciones del robot esférico propuesto 12
2.1 Decisión de Diseño 12
2.2 Diseño propuesto y ecuaciones de su funcionamiento 14
2.3 Ecuaciones de movimiento 15
2.3.1 Ecuaciones de movimiento horizontal 15
2.3.2 Ecuaciones de movimiento de subida y bajada 17
2.3.3 Ecuaciones de movimiento al pasar por obstáculos 18
2.3.4 Ecuaciones de movimiento de direccionamiento del robot esférico 19
2.4 Simulaciones de robot esférico 20
2.4.1 Trayecto recto 21
2.4.2 Trayecto en zigzag 22
2.4.3 Trayecto subiendo planos inclinados 22

2.4.4 Trayecto pasando por encima de objetos ... 23

3 Construcción del robot esférico ... 25

3.1 Diseño de robot esférico ... 25

3.2 Construcción del robot esférico ... 29

3.2.1 Construcción de la carcasa esférica ... 29

3.2.2 Construcción de la base interior ... 30

3.2.3 Construcción del péndulo ... 31

3.3 Programación del robot esférico ... 31

3.4 Conexión del robot con el computador ... 32

3.4.1 Resolución de problema de comunicación de aplicación Blynk ... 33

3.5 Configuración de procedimiento de imagen en el software Swistrack ... 34

3.6 Configuracion de codigo de Ejs ... 35

3.7 Explicación del funcionamiento del sistema ... 35

3.8 Comportamiento del robot ... 38

3.8.1 Fórmulas para orientación con respecto al robot ... 38

4 Pruebas del robot esférico ... 40

4.1 Pruebas de trayectorias ... 40

4.1.1 Prueba 1: Trayectoria recta ... 40

4.1.2 Prueba 2: Maniobrabilidad ... 41

4.1.3 Prueba 3: Trayectoria pasando encima de objetos ... 41

4.2 Prueba 4: Reconocimiento de robot por software Swistrack ... 42

4.3 Prueba 5: Reconocimiento de posición mediante software Swistrack ... 43

4.4 Prueba 6: Giro con objetivo de 90 grados ... 43

4.4.1 Prueba de giro con objetivo de 180 grados ... 44

4.5 Propuesta de control del robot esférico ... 45

4.5.1 Propuesta de control de posicionamiento ... 45

Discusión y conclusiones ... 47

Bibliografía ... 50

A Apéndice ... 53

A.1 Pasos a seguir para la creación de un servidor local de Blynk en raspberry pi ... 56

A.2 Código de robot ... 59

A.3 Código de programa Ejs ... 62

Introducción

Hoy en día el campo de la robótica es muy extenso en lo que se refiere al diseño de un robot, se tiene que investigar y analizar los diferentes tipos de mecanismos que un robot pudiera llegar a integrar, para robot de 4 o 3 soportes este no es problema, pero para robot que tienen 2 o menos puntos de soportes este puede ser un gran desafío ya que no solo se debe investigar una forma de mantener al robot en un estado de equilibrio constante sino que también se debe integrar este equilibrio para cuando el robot se traslade de un lugar a otro o haga algún movimiento que cambie su centro de masa.

Un tipo de robot que requiere de un mecanismo activo de equilibrio es el de un robot esférico móvil el cual está compuesto de una cubierta esférica que protege la circuitería y que también usa para su desplazamiento. Por otro lado si bien los robot con 3 o más puntos de soportes tienen mejor estabilidad en lo que se refiere al camino que utiliza para desplazarse, estos también tienen el problema de que si se dan vuelta o alguna rueda o soporte se daña, estos no pueden seguir moviéndose de un lugar a otro, esto no pasa, en general, con el robot esférico ya que al ser la cubierto esférica es imposible que exista el problema de volcamiento y además al tener una cubierta esférica esto significaría tener una tolerancia a presiones exteriores mayores a las ruedas que otros robot utilizan.

Los robot esféricos se pueden utilizar en aplicaciones de rescate, reconocimiento, enseñanza o simplemente como un juguete, los tipos de mecanismos para lograr un equilibrio interno y una translación son variados, cada método tiene sus problemas y posibles soluciones.

La explicación y el entendimiento de cómo funcionan los mecanismos de un robot es de gran importancia para poder recrearlo, y si viene el caso, modificarlo para que cumpla ciertos trabajos. Con el conocimiento de cómo funciona un robot también se puede escoger de entre varios diseños de un mismo robot, para esto se puede hacer una tabla de ventajas y desventajas de cada diseño y escoger el cual cumple con los requisitos que un tiene en mente.

Una vez que se establece el diseño de cualquier robot se debe encontrar las ecuaciones físicas que gobiernan al diseño, para conocer las limitaciones de este y también ayudara en saber que componente usar y además saber cuáles podrían ser las especificaciones del robot esférico.

Una parte integral de cualquier proyecto es realizar simulaciones sobre el proyecto que se realizara, esto es posible gracias a programas de simulación que pueden dar datos útiles para el desarrollo del proyecto. En este caso el programa de simulación que se uso es el llamado V-Rep, el cual es un programa de simulación de robots. El programa V-Rep si bien es un buen programa en lo que respecta a entorno de simulación, este no tiene una gran variedad de robots con los que uno pueda simular por lo que si uno quiere realizar simulaciones con la estructura del robot de una forma detallada se tiene que recurrir a otro tipo de software, no así, si sólo se quiere simular un robot con estructura minimalista ya que el software V-Rep puede crear formas como cubos, cilindros, esferas, etc. Y darles propiedades que permitan ser utilizadas como partes de un robot. En el caso de que si se necesita diseñar una estructura detallada del robot, una buena alternativa es el Autodesk Fusion 360 el cual tiene una interfaz bastante sencilla además de tutoriales en internet.

Las fases de creación de un robot comienzan con el diseño del robot, los mecanismos que utilizara, la viabilidad de este, el presupuesto del proyecto, la construcción de este y luego está la etapa de puesta en marcha que consiste en pruebas y ajustes necesarios para su funcionamiento.

En variados papers, el método de construcción se basa en los siguientes pasos, considerando ya el paso de investigación y diseño del robot, se parte por calcular los parámetros que afectan al robot, ya sea en el movimiento que este hace o alguna funcionalidad que pueda tener, luego de esto se busca los componentes apropiados considerando los parámetros calculados, acto seguido se construye el cuerpo del robot, se arma y se coloca la circuitería que permite el funcionamiento de este, se programa el robot y se hacen pruebas para probar el funcionamiento de este, si el funcionamiento de este no llega a ser como se esperaba, sea por el diseño de este o por algún error de programación, se vuelve a diseñar el robot o se arregla el error de programación según sea el caso, el proceso de crear un robot es uno iterativo.

Los avances en la creación de robot no son pocos en la actualidad esto también es cierto para los diferentes tipos de mecanismos que hacen mover a estos, así también como las formas de control de posición y movimiento, de este último podemos hablar de distintas formas en que el operador comande al robot a efectuar diferentes movimientos y trayectos. Los diferentes tipos de comunicación utilizan los medios de red inalámbrica, radio señal, li-fi, ondas infrarrojas, etc. Las señales de la red inalámbrica en general no son aptas para el uso a grandes distancias y además estas tienden a rebotar en superficies, es por este motivo que estas se utilizan en espacios cerrados en donde la señal de una red inalámbrica (WiFi) es más fuerte y confiable, es decir, que si uno quiere controlar un robot en un espacio cerrado este tipo de señal es la indicada para tener un buen manejo de cualquier robot que utilice las redes inalámbricas como medio de comunicación.

Una forma en que un operador se comunique con un robot utilizando la red inalámbrica es mediante la aplicación "Blynk". Esta aplicación puede usarse desde un celular o mediante comandos api a través de una dirección URL, esto último ayuda mucho en el momento de usar un computador para programar un patrón de trayecto para cualquier tipo de robot

Una parte integral de cualquier robot es el sistema de control que este puede tener, este control puede ser para controlar procesos internos del robot o también puede ser para accionar mecanismos físicos del robot. El Swistrack y el Ejs pueden formar parte de tal sistema ya que el primero puede otorgar los valores de posición y orientación del robot y puede enviar tales datos al segundo (Ejs), el cual a su vez se puede generar un código para que se pueda comunicar con cualquier robot sea directamente o indirectamente, es decir, mediante servidores, y así el robot pueda responder con respecto a los datos que se le mandan.

Para que un sistema de control pueda cumplir con su objetivo es necesario que este contenga un feedback que permita corregir algún error del proceso que se está ejecutando, este feedback puede ser una cámara mirando desde una posición elevada, esto es con motivo de mantener una buena observación de lo que se quiere rastrear, en este caso un robot.

También una parte importante para el posicionamiento del robot es el rastreo ininterrumpido y en lo posible exacto de este. Una incorrecta información de datos influirá en la estabilidad del funcionamiento del sistema.

Objetivo general

- Diseñar e implementar un sistema de control de un robot esférico.

Objetivos específicos

  - Estudiar los modelos de robots esféricos que existen actualmente.
  - Simular el diseño de un sistema de control de un robot esférico.
  - Implementar el sistema de control de un robot esférico.

1 Estado del arte

1.1 Antecedentes generales y propuesta

1.1.1 Descripción del problema

El principal problema que tienen los robot esféricos es la estabilidad del robot en general, es por esto que muchas veces se ven robot de 3 o más puntos de suportes, ya estos tienen mayor estabilidad y permiten enfocar la atención a otros problemas de diseño que estos robots podrían tener. El robot esférico al tener solo un punto de soporte hace que éste no pueda mantenerse en un punto sin oscilar, diferentes tipos de mecanismo se idearon para la contrarrestar esta oscilación pero estos también requieren una calibración para controlar el movimiento.

1.1.2 Descripción de la propuesta

La solución que se propone es estudiar y analizar los diferentes mecanismos de equilibrio y movilidad para un robot esférico y encontrar cual es el mejor en lo que respecta a movilidad de tal forma de tratar de imitar los movimientos del robot de Star Wars BB-8. Además de buscar soluciones a la construcción del robot esférico como de control de este, ya sea a través de un control manual o por coordenadas en plataforma y de fácil uso.

1.2 Robot esférico móvil

Este tipo de robot se empezó a popularizarse hace poco tiempo. El robot esférico consta de una carcasa esférica y que en su interior están todos los componentes electrónicos que permiten su funcionamiento. Las características de este robot esférico móvil son las siguientes:

- Estos tipos de robots (Robots esféricos móviles) tienen, en general, la ventaja de no verse afectado por el volcamiento de este mismo por la razón de que su interior se adapta a la orientación en que su exterior se coloca. [1]
- Se usa para reconocimiento de lugares en donde el terreno no es accesible sea por algún químico en el aire que pueda afectar a los componentes electrónicos o por el terreno del lugar. [2]
- Se usa también como herramienta de aprendizaje interactivo en varios ámbitos de la electrónica. [3]
- Se usa también en el campo del entretenimiento, en el de vigilancia y de exploración. [4] [5]
- En general tiene problemas en subir a terrenos elevados. [2]
- Estos tipos de robots tienen una gran variedad de mecanismos que se pueden usar para su movilización. [6]

1.2.1 Objetivo del robot esférico móvil

El objetivo del desarrollo de este tipo de robot es el de crear un robot que pueda maniobrar en cualquier tipo de ambientes en que otros tipos de robot no pueden, ya sea por razones medio-ambientales o por razones de terreno.

1.2.2 Usos

Los usos que se le puede dar a este tipo de robot son variados y se pueden clasificar en los siguientes:

- Reconocimiento de un área en expediciones militares, haciendo más seguro el trabajo de los soldados que participan en estas expediciones.

- Exploración extra planetaria ya que este robot presenta menos probabilidades de atascarse en el terreno.

- Explorar lugares en los cuales el medio-ambiente es toxico para humanos y robots convencionales.

- Se pueden utilizar en el campo del aprendizaje ya que a estos robots se les aplica aprendizaje relacionado con el control automático y la programación

En esencia este tipo de robot solo sirve como un robot observador o de investigación ya que por su diseño, es muy difícil agregar algún actuador que manipule el medio-ambiente.

1.3 Tipos de mecanismos de movimiento del robot esférico

Los Robots esféricos tienen diferentes mecanismos que se pueden usar para su movimiento, en general hay 7 tipos de mecanismos [6] [7], los demás son una combinación de estos 7 tipos. La mayoría de los mecanismos utilizan el llamado desfase de baricentro el cual es un término que se utiliza para mecanismos que varían el centro de masa para obtener un resultado en particular, son muy pocos los mecanismos que no utilizan este fenómeno para el movimiento. Los tipos de mecanismos que utilizan los robots esféricos son los siguientes.

1.3.1 Esfera de hámster

Este diseño fue uno de los primero en aparecer, es llamado así porque el mecanismo imita a una esfera con un hámster dentro, como puede verse en la Figura 1-1 el mecanismo consiste en un robot con ruedas dentro de una esfera. Cuando el robot con rueda se mueve, este provoca que la esfera se mueva así con él, este tipo de movimiento es no holonómico [8], es decir, para cambiar de dirección el mecanismo interno de robot tiene que reubicarse.

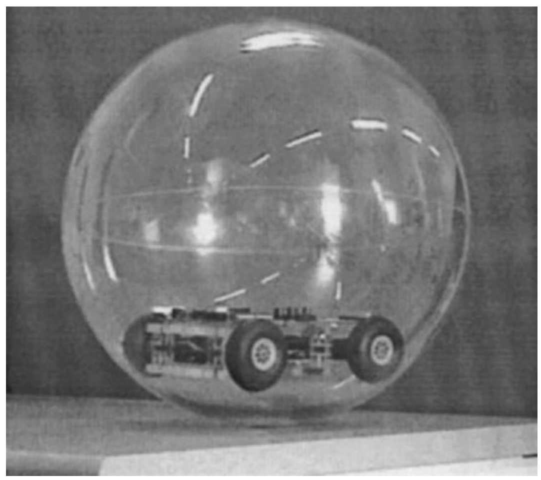


Figura 1-1: Robot esférico con mecanismo de Esfera de Hámster.

Además se debe decir que este mecanismo tiene diversos problemas, uno de estos es debido a que como el robot dentro de la esfera no está sujeto a la esfera este puede variar su trayectoria fácilmente, otro problema se debe a que las ruedas del robot pueden resbalar y como consecuencia no producir movimiento.

Actualmente no hay robots esféricos comercialmente disponibles con este tipo de mecanismo ya que, como se señaló anteriormente, este mecanismo presenta variados problemas en su diseño como para ser exitosa su venta.

1.3.2 Unidad de accionamiento interno (IDU)

Este tipo de mecanismo obliga a los componentes internos encargados del movimiento a siempre permanecer en contacto con la esfera, utiliza para esto, rodamiento de bolas, resortes, un eje, ruedas y caja de control como es indicado en la figura 1-2. Si bien este es uno de los mecanismos que más se usan y que además es de bajo costo, a velocidades altas es difícil el control de dirección

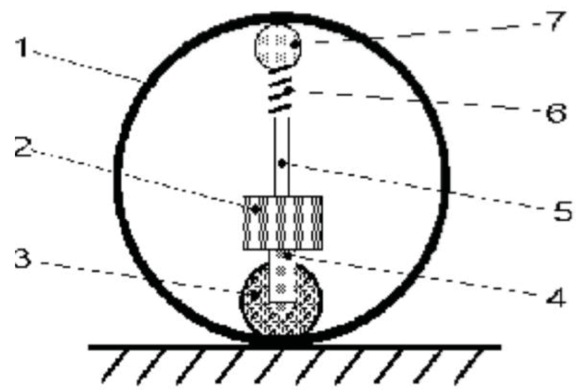


Figura 1-2: Estructura interna de robot con mecanismo IDU 1. Esfera que encierra a los componentes 2. Caja de control 3. Ruedas 4. Eje de dirección 5. Eje de soporte 6. Resorte 7. Rodamiento de bola.

Este tipo de robot no puede hacer uso del momento de inercia guardado, es decir, si la rueda del robot se detiene este se comportará de forma errática. Cuando un sistema con este mecanismo baja por una colina este debe mantener las ruedas en movimiento, de lo contrario, empezará a realizar movimientos impredecibles. Para ello, el tipo de diseño debe estar bien analizado y estudiado con la finalidad de llegar a un buen funcionamiento. Su sistema de control es pasivo, es decir, que no requiere un circuito que se encargue de contrarrestar oscilaciones internas que puedan haber cuando este se mueva. Un ejemplo de este tipo de mecanismo que esta comercialmente disponible es el llamado Sphero (ver Figura 1-3) el cual tiene un precio de $130 USD que equivale a $77.000 CLP.

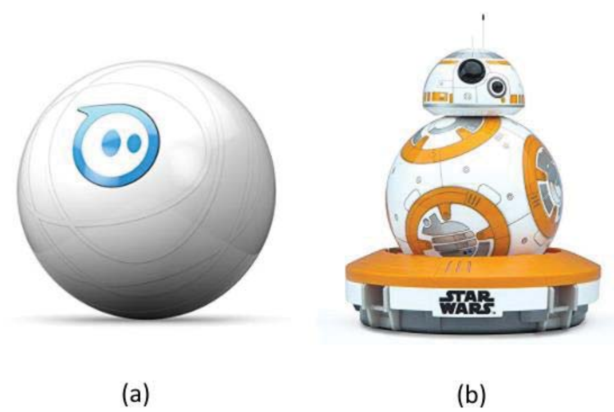


Figura 1-3: (a) Robot esférico Sphero 2.0 (b) Robot BB-8 de Star Wars

El juguete que se puede encontrar del robot BB-8 de Star Wars (Ver Figura 1-3 (b)) fue desarrollado por la misma compañía que creo el robot Sphero y este tiene precio de $120.000 CLP.

1.3.3 Péndulo conducido (pendulum driven)

Este diseño de mecanismo es popular en el ámbito de la industria y en lo académico, este mecanismo consiste en un eje fijo que pasa por el centro de la carcasa del robot, este eje tiene un péndulo que puede rotar alrededor del eje como lo indica la Figura 1-4, cuando el péndulo se mueve este cambia el centro de masa del robot, lo cual con esto provoca movimiento en el robot. Si el péndulo se balancea hacia la izquierda y por debajo del ecuador del robot, este producirá que el robot se desplace hacia la izquierda, en caso contrario, el péndulo se balancea hacia la derecha. La velocidad que puede generar dependerá de cuan pesado sea el péndulo.

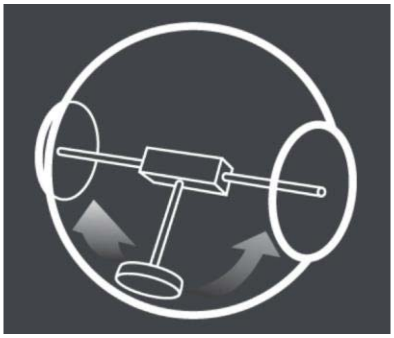


Figura 1-4: Esquema de un robot con mecanismo Péndulo Conducido (Pendulum Driven).

El más notable problema que tiene este mecanismo es que no puede viajar por pendientes pronunciadas, en la práctica un robot bien diseñado puede viajar por pendientes de hasta 30 grados. Para que un robot pueda movilizarse por pendientes mayores a 30 grados este tendría que ser diseñado con técnicas que no son comercial o económicamente convenientes. Este tipo de mecanismo no es holonómico. Los robots con este tipo de mecanismo en general son diseñados para tener mínimo costo y poca complejidad.

Un robot esférico comercialmente disponible de este mecanismo es el llamado Queball (Ver Figura 1-5) el cual está pensado para terapia de personas autistas. Este robot tiene un valor de $2995 euros, el cual en pesos chilenos el valor asciende a $2.204.320 CLP.

Figura 1-5: Robot esférico Queball.

1.3.4 Mejoras notables (notable enhancements)

Muchas veces el diseñador de este tipo de robots ha optado por variar no solo el mecanismo interno sino que también cambia como la carcasa esférica interactúa con el medio ambiente. Estos cambios son diversos, como por ejemplo, hacer la carcasa esférica transparente para poder instalar una cámara dentro del robot, instalar un mecanismo en la carcasa para que el robot pueda saltar o también instalar un mecanismo que haga que el robot tenga una segunda forma de desplazarse, como por ejemplo, tener piernas que al abrirse el robot este pueda caminar con ellas. En la Figura 1-6 se muestran algunos ejemplos de esta clasificación de mecanismos de movimiento de este robot.

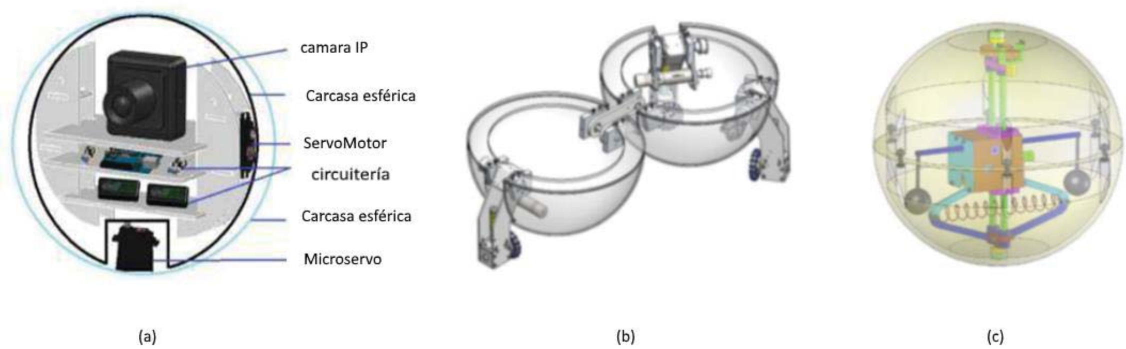


Figura 1-6: (a) Robot esférico con cámara (b) Robot con Piernas ocultas dentro del robot (c) Robot Esférico con mecanismo que posibilita saltar.

Un ejemplo de este tipo de mecanismo es el llamado Rotundus (ver Figura 1-7) el cual es un robot pensado en la vigilancia y que además tiene un exterior hecho de policarbonato con un revestimiento de alta fricción, dos cámaras a cada lado de su cuerpo y control mediante wifi. Actualmente este robot no está a la venta para uso doméstico.

Figura 1-7: Robot de Vigilancia Rotundus.

1.3.5 Cambio de masa múltiple (Multiple-Mass-Shifting)

Como se muestra en la Figura 1-8 este tipo de mecanismo utiliza 3 o 4 masas que se mueven por guías lineales para cambiar el centro de masa y en consecuencia provocar movimiento

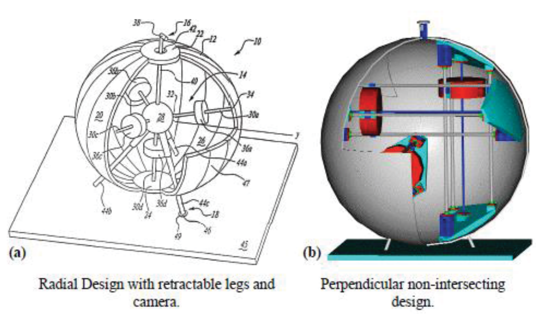


Figura 1-8: Esquema de 2 tipos de Multiple-Mass-Shifting. (a) Diseño radial (b) diseño perpendicular.

En el diseño de la Figura 1-8(a), hay 4 masas que están conectadas y que de cierta manera forman un tetraedro regular por medio de diferentes radios que se conectan en el centro de la esfera. En la Figura 1-8(b) se muestra un diseño más optimizado en donde hay 3 masas posicionadas de forma que cada masa se mueva por los 3 ejes de coordenada, es decir, X, Y y Z.

La característica más significativa de este tipo de mecanismo es que es completamente holonómico, es decir, no tiene que reposicionarse para cambiar de dirección. El movimiento de este tipo de mecanismo puede llegar a ser muy preciso, pero el control de movimiento es complicado y el diseño mecánico suele ser complicado. Para que el robot pueda moverse en una línea recta rápidamente los actuadores tienen que ser de alto poder, lo cual haría el diseño ineficiente.

Por razones de construcción, diseño y control, no es sabio, para las empresas, invertir en este mecanismo, por lo que este mecanismo solo se ve en el ámbito académico.

1.3.6 Cuerpo deformable

Este tipo de mecanismo cambia la forma de la carcasa del robot de tal forma que produzca movimiento en la dirección deseada, esto lo hace gracias a que cuando se deforma la carcasa del robot se puede controlar el centro de masa del robot y con esto se puede producir movimiento en la dirección deseada. Este tipo de mecanismo es muy reciente y no se encuentran muchos prototipos que contengan este tipo de mecanismo. En la figura 1-9 se puede observar un robot con este tipo de diseño, el robot fue hecho en la universidad de Ritsumeikan en Japón, este fue llamado Koharo y funciona usando cables y una estructura flexible para la carcasa, al contraer y expandir los cables en coordinación es posible causar que la estructura se deforme y con esto que se produzca movimiento.

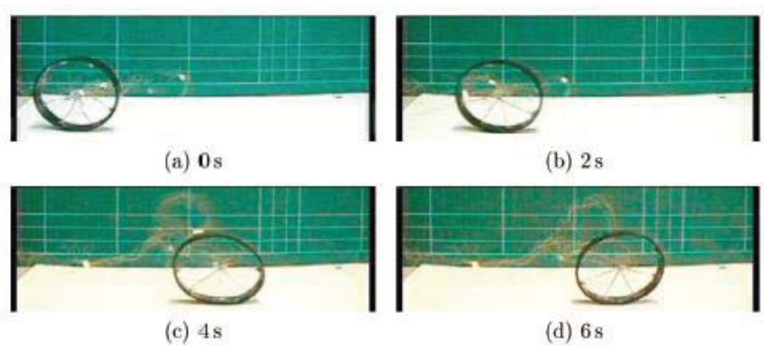


Figura 1-9: Robot Koharo.

Los cables que se usan para este robot son especiales ya que al aplicar corriente, estos cables se calientan y retraen. Una vez enfriados los cables se expanden a su longitud original. Desafortunadamente este proceso es lento por lo que no sirve para operaciones rápidas.

1.3.7 Single ball (Una sola bola)

Este mecanismo sigue el mismo principio que los mouse tradicionales con bola, excepto que en este caso el trabajo está invertido, es decir, la bola ejerce movimiento a la carcasa esférica del robot y en consecuencia hace que el robot se traslade de un lugar a otro, un esquema de un robot esférico llamado Omniqiu que utiliza el mismo mecanismo se muestra en la figura 1-10.

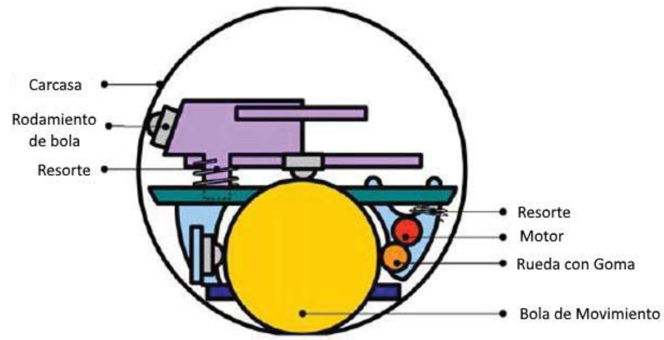


Figura 1-10: Robot esférico que utiliza el mecanismo de una bola.

Para el movimiento este mecanismo usa rodillos conectados a motores para ejercer movimiento a la bola. Actualmente no se pudo encontrar un robot comercialmente disponible que contenga este mecanismo.

2 Diseño y simulaciones del robot esférico propuesto

2.1 Decisión de Diseño

El mecanismo propuesto se eligió en base a 5 factores los cuales son: bajo costo, realizable en un corto plazo, facilidad de obtención de materiales y poca complejidad de diseño y buen control de movimiento. Al considerar estos factores y volviendo a la tabla de ventajas y desventajas que se mostró en el informe anterior (Ver Tabla 2-1) se analizaron las alternativas que se presentaron y se dio una mejor explicación del porque se eligió el mecanismo de péndulo conducido.

Tabla 2-1: Tabla de ventajas y desventajas de los diferentes diseños

Mecanismos de movimiento	Ventajas	Desventajas
Esfera de Hámster	• Es de fácil construcción • No se requiere invertir mucho dinero en el robot	• No tiene buen control de dirección • Si el robot esférico se cae, es probable que el robot del interior se de vuelta y no pueda seguir moviéndose • Es muy probable que el robot interior no tenga buena tracción lo que llevaría a un pésimo desempeño en el movimiento. • No es un mecanismo holonómico.
Unidad de accionamiento interno (IDU)	• Es de bajo costo. • Es uno de los que más se usan en la industria. • El mecanismo interno no presentaría ningún problema de ser separado de	• No puede hacer uso de su momento. • No puede alcanzar altas velocidades ya que se comporta de forma errática. • Cuando el robot baja una colina no puede desactivar los motores dc ya

	la carcasa por algún golpe o movimiento fuerte.	que esto provocaría movimientos erráticos. • Su sistema de control es pasivo. • Tiene que estar bien diseñado para que funcione bien. • No es un mecanismo holonómico
Péndulo conducido (Pendulum Driven)	• Es uno de los más usados en la industria y en proyectos académicos. • Es de fácil construcción. • Generalmente de bajo costo.	• El movimiento está ligado al peso del péndulo del robot. • No es un mecanismo holonómico.
Mejoras Notables (Notable Enhancements)	• Muchos de estos mecanismos son innovadores. • Muchos arreglan las fallas que otros mecanismos tienen, como por ejemplo, subir escaleras o saltar obstáculos, o agregar funcionalidades que los diseños convencionales no tienen.	• Algunos pueden llegar a ser complejos de construir. • No todos pueden ser accesibles de construir. • Algunos tienen tantos cambios que al final dejan de ser un robot esférico.
Multiple-Mass-Shifting	• Son mecanismos holonómicos. • El movimiento de este tipo de mecanismo puede ser muy preciso.	• El control de movimiento es complicado. • El diseño mecánico suele ser complicado • Requiere actuadores de alto poder para respuestas rápidas.
Cuerpo deformable	• Es una forma innovadora de producir movimiento.	• Es un área nueva de este tipo de mecanismo por lo que puede que no haya mucha información al respecto. • Los materiales para su construcción son especiales y en consecuencia son difíciles de conseguir.
Single ball (Una Sola Bola)	• Es un mecanismo Holonomico.	• Es mecanismo que requiere precisión en lo que respecta a diseño. • Presenta dificultad en el control ya que no solo se tiene que manejar el control de movimiento del robot

		sino que también internamente tiene que haber un control de equilibrio interno para que el robot no se tilde hacia un lado.

Como se puede ver en la Tabla 2-1 los mecanismos de Mejoras Notables (Notable Enhancements), Multiple-Mass-Shifting, Cuerpo deformable, Una Sola Bola (Single ball), no cumplen con uno o de dos de los factores que se mencionó anteriormente, por lo cual se tuvo que enfocar de entre los tres primero en la tabla, los mecanismos de Esfera de Hámster, Unidad de Accionamiento Interno (IDU) y el Péndulo Conducido, de entre estos tres, dos tienen problema de control sea todo el tiempo o en ciertas condiciones como por ejemplo en el mecanismo de esfera de hámster tiene un direccionamiento impredecible y el mecanismo de unidad de accionamiento interno tiene problemas de control al bajar por superficies inclinadas, es decir, se comporta de forma errática, esto deja al mecanismo de Péndulo Conducido, ya que este no tiene problema de control, los materiales para su construcción están disponibles comercialmente hablando, el precio de los materiales no cuestan mucho, su diseño no es complicado, por lo tanto se puede realizar en poco tiempo y además varias universidades han construido un robot esférico utilizando este tipo de mecanismo por lo que hay más información en forma de papers a los que uno pudiera acudir.

2.2 Diseño propuesto y ecuaciones de su funcionamiento

Una vez elegido el diseño del robot esférico, se presentó el siguiente bosquejo para el robot esférico. Este diseño no es muy elaborado y, por lo tanto, su fabricación es sencilla. El bosquejo del diseño del robot se puede apreciar en la Figura 2-1.

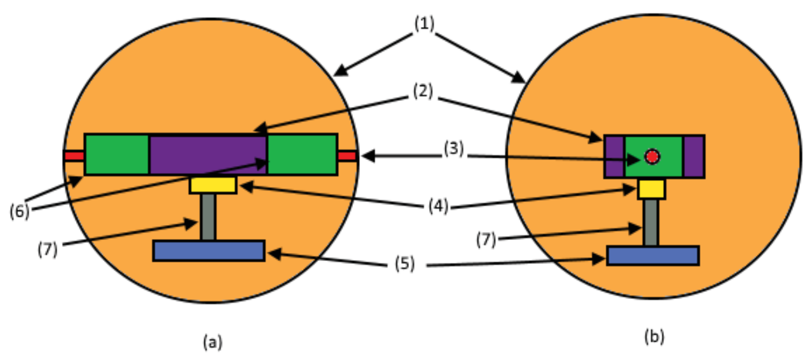


Figura 2-1: Diseño propuesto para el robot esférico, (a) Vista frontal, (b) Vista Lateral, (1) esfera (carcasa), (2) Caja de Circuitería, (3) Eje paralelo al suelo, (4) Servomotor, (5) Masa, (6) Motores DC, (7) Eje perpendicular al suelo

Este diseño se encontró en un video en el link [9], en donde se realizó un test de cómo se mueve el robot de este diseño. Una explicación de este mecanismo es que, los motores DC se utilizan para hacer avanzar y retroceder al robot, estos motores son mantenidos relativamente fijos por la masa que se encuentra descrita en la figura 2-1(5), esta masa, además de mantener relativamente fijo el interior del robot, también sirve para cambiar la dirección del robot esférico, esto lo hace haciendo un movimiento de rotación sobre si misma, produciendo momento de inercia en el robot y con esto es posible el cambio de la dirección del robot. El robot también tiene un sensor giroscopio el cual permite saber cuántos grados a girado el robot esférico.

2.3 Ecuaciones de movimiento

Para poder hablar de las ecuaciones físicas por la que está regido el robot esférico que se quiere construir primero tenemos que definir las variables y constantes que entran en juego. En la figura 2-2 se muestra la correspondencia de las variables y constantes de la parte del mecanismo que se encarga del movimiento de adelanto y retroceso.

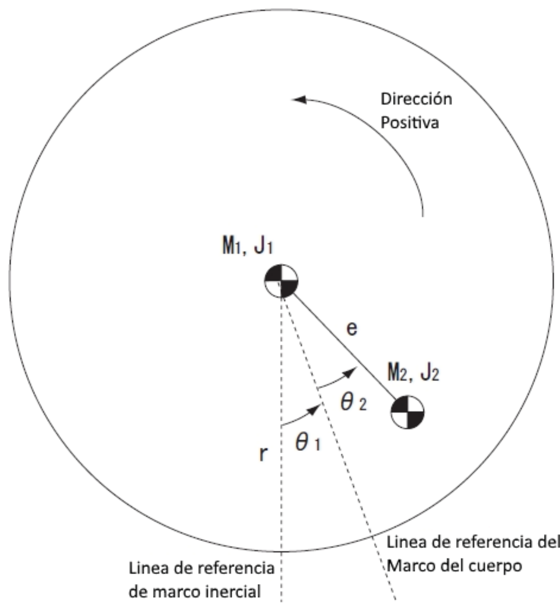


Figura 2-2: esquema del robot simplificado con las correspondientes variables.

Las ecuaciones que se muestran para el movimiento horizontal, movimiento de subida y bajada y movimiento al pasar por obstáculos y, además, de las figura 2-2, la figura 2-3 y la figura 2-4, se sacaron del paper que se encuentra en [10]. Se decidió utilizar estas, porque el diseño del robot que se muestra en ese paper tiene una similitud con estos mecanismos de movimientos, con el que se pretende hacer.

2.3.1 Ecuaciones de movimiento horizontal

Se considera que la carcasa en la figura 2-2 siempre está tocando el suelo, no hay resbalamiento o deslizamiento entre el piso y la carcasa. La ecuación de LaGrange se utilizó para derivar las ecuaciones de movimiento. El movimiento horizontal contiene las siguientes ecuaciones de energía potencial, energía cinética y energía rotacional, respectivamente.

$$\begin{cases} U_1 = 0 \\ U_2 = -M_2 ge\cos(\theta_1 + \theta_2) \end{cases} \tag{2-1}$$

$$\begin{cases} K_1 = \dfrac{1}{2} M_1 (r\omega_1)^2 \\ K_2 = \dfrac{1}{2} M_2 \left\{ \left(r\omega_1 - e\cos(\theta_1 + \theta_2)\,(\omega_1 + \omega_2) \right)^2 + \left(e\sin(\theta_1 + \theta_2)\,(\omega_1 + \omega_2) \right)^2 \right\} \end{cases} \tag{2-2}$$

$$\begin{cases} T_1 = \dfrac{1}{2} J_1 \omega_1^2 \\ T_2 = \dfrac{1}{2} J_2 (\omega_1 + \omega_2)^2 \end{cases} \tag{2-3}$$

Las letras de las formulas tienen el siguiente significado:

U_1: Energía Potencial de la carcasa esférica con respecto a la altura de su centroide

U_2: Energía Potencial del péndulo con respecto a la altura del centroide de la carcasa esférica

K_1: Energía cinética de la carcasa esférica

K_2: Energía cinética del péndulo

T_1: Energía rotacional de la carcasa esférica

T_2: Energía rotacional del péndulo

r: Radio de la carcasa esférica

e: Distancia entre el centroide de la carcasa esférica y el péndulo

θ_1: Angulo de rotación de la carcasa esférica

θ_2: Angulo de rotación del péndulo con respecto a la carcasa esférica

ω_1: Velocidad angular de la carcasa esférica

ω_2: Velocidad angular del péndulo con respecto de la carcasa esférica

J_1: Momento de inercia de la carcasa esférica con respecto a su centroide

J_2: Momento de inercia del péndulo con respecto a su centroide

M_1: Masa de la carcasa esférica

M_2: Masa del péndulo

g: Aceleración de Gravedad

Luego se obtiene la ecuación de Lagrange puede ser calculada de la manera siguiente.

$$L = K_1 + K_2 + T_1 + T_2 - U_1 - U_2 \tag{2-4}$$

Las ecuaciones de movimiento de Lagrange son las siguientes

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \omega_1}\right) - \frac{\partial L}{\partial \theta_1} = -T + T_f \tag{2-5}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \omega_2}\right) - \frac{\partial L}{\partial \theta_2} = T \tag{2-6}$$

En donde t es tiempo, T es el torque aplicado entre la carcasa y la caja de circuitería y T_f es el torque que aparece con la fuerza de roce que se produce entre la carcasa y el suelo. Se resuelven las ecuaciones anteriores (2-5 y 2-6) y se obtiene.

$$\begin{aligned} -T + T_f = {} & a_1(J_1 + J_2 + M_1 r^2 + M_2 r^2 + M_2 e^2 - 2M_2 re\cos(\theta_1 + \theta_1)) \\ & + a_2(J_2 - M_2 re\cos(\theta_1 + \theta_2) + M_2 e^2) \\ & + M_2 re\sin(\theta_1 + \theta_2)\,(\omega_1 + \omega_2)^2 + M_2 ge\sin(\theta_1 + \theta_2) \end{aligned} \tag{2-7}$$

$$T = a_1(J_2 - M_2 re\cos(\theta_1 + \theta_2) + M_2 e^2) + a_2(J_2 + M_2 e^2) + M_2 ge\sin(\theta_1 + \theta_2) \tag{2-8}$$

La fórmula calculada sirve para determinar que motor servirá para la construcción del robot esférico y también para poder tomar consideraciones en lo que respecta al diseño, peso y dimensiones, del robot.

2.3.2 Ecuaciones de movimiento de subida y bajada

Los movimientos de subida y bajada se puede calcular en base a las ecuaciones anteriores, esto se debe a que solo se tiene que tener en cuenta que la dirección de la aceleración de gravedad, dicho esto, se hicieron los cambios y ajustes de fórmulas para su funcionalidad.

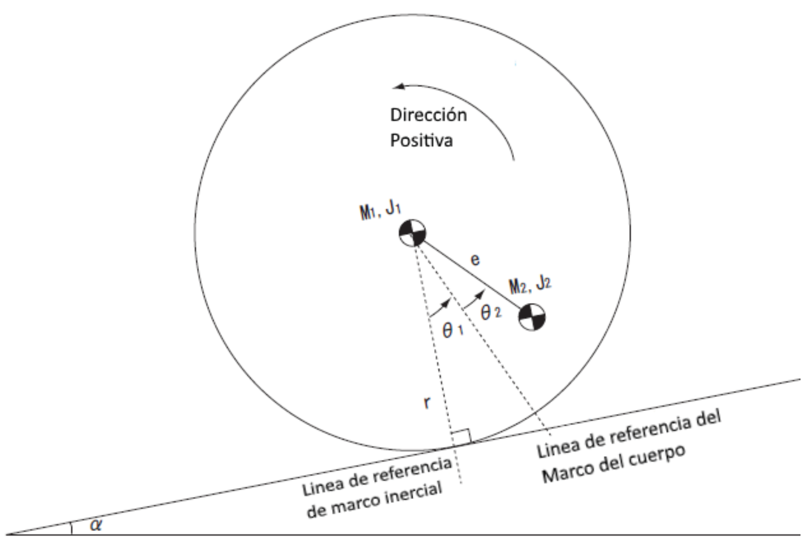


Figura 2-3: Robot esférico vista lateral subiendo por un camino empinado.

$$U_1 = -M_1 g r \theta_1 \sin\alpha \tag{2-9}$$

$$U_2 = -M_2 g r \theta_1 \sin\alpha - M_2 g e \cos(\theta_1 + \theta_2 + \alpha) \tag{2-10}$$

En donde α es el ángulo de inclinación del camino que el robot esférico está siguiendo

$$\begin{aligned} -T + T_f = {} & a_1(J_1 + J_2 + M_1 r^2 + M_2 r^2 + M_2 e^2 - 2M_2 re\cos(\theta_1 + \theta_2)) \\ & + a_2(J_2 - M_2 re\cos(\theta_1 + \theta_2) + M_2 e^2) \\ & + M_2 re\sin(\theta_1 + \theta_2)\,(\omega_1 + \omega_2)^2 + M_2 ge\sin(\theta_1 + \theta_2 + \alpha) \\ & - (M_1 + M_2) gr\sin\alpha \end{aligned} \tag{2-11}$$

$$T = a_1(J_2 - M_2 re\cos(\theta_1 + \theta_2) + M_2 e^2) + a_2(J_2 + M_2 e^2) + M_2 ge\sin(\theta_1 + \theta_2 + \alpha) \tag{2-12}$$

Al igual que las formulas del movimiento horizontal esta fórmula servirá para determinar el tipo de motor que se utilizará para el movimiento hacia adelante y hacia atrás, también se debe considerar el motor para el caso en que se requiera subir un plano inclinado con un ángulo en particular.

2.3.3 Ecuaciones de movimiento al pasar por obstáculos

Un buen diseño del robot puede llegar a pasar por obstáculos u otros relieves abruptos que el terreno puede tener, eso sí, esto tiene una limitante, la cual es la altura del obstáculo o relieve que el robot tiene que cruzar. Las ecuaciones que se encontraron, también se obtuvieron de [10].

Para que el robot pueda pasar por encima de obstáculos el torque producidos por los motores del robot debe ser mayor que el contra-torque producido por la fuerza de gravedad.

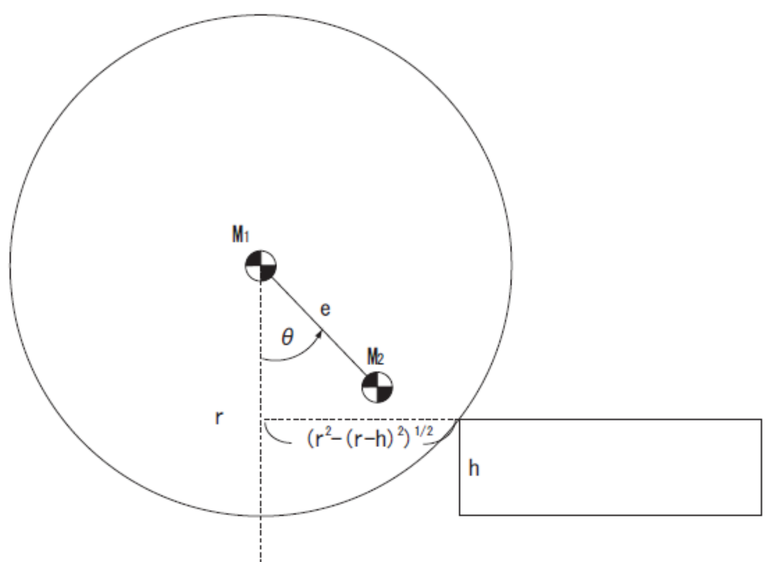


Figura 2-4: Esquema del robot esférico pasando por un obstáculo.

En donde θ es el ángulo rotatorio del péndulo, h es la altura del objeto u obstáculo.

(Torque de motores) > (Contra torque)

$$M_2 ge \sin(\theta) > (M_1+M_2) g (r^2-(r-h)^2)^{1/2}$$

$$\therefore (M_2)^2 e^2 \sin^2(\theta) > (M_1 + M_2)^2 \{r^2 - (r - h)^2\}$$

$$\therefore (r - h)^2 > r^2 - (e^2 \sin(\theta)(M_2)^2) \; / \; (M_1 + M_2)^2$$

De aquí se despeja h y queda de la siguiente forma:

$$h < r - \sqrt{r^2 - \left(\frac{e \sin\theta \, M_2}{M_1 + M_2}\right)^2} \tag{2-13}$$

Considerando el caso extremo, es decir, cuando sin (θ) es igual a 1, quedaría de la siguiente manera.

$$h_{Max} = r - \sqrt{r^2 - \left(\frac{eM_2}{M_1 + M_2}\right)^2} \tag{2-14}$$

En donde, como se puede ver en la figura 2-4, h es la altura del obstáculo. Esta fórmula servirá para más adelante para averiguar los límites que los motores deben tener para poder pasar por encima de obstáculo u objetos.

2.3.4 Ecuaciones de movimiento de direccionamiento del robot esférico

Para el movimiento que permite cambiar de dirección del robot se pensó en usar un CMG (control moment gyroscope) como masa del péndulo. El cálculo de este movimiento se obtuvo del link [11] ya que no se encontró un diseño idéntico o parecido al que se usó en la propuesta por lo que se tuvo que desarrollar las ecuaciones de movimiento que se pretenden analizar.

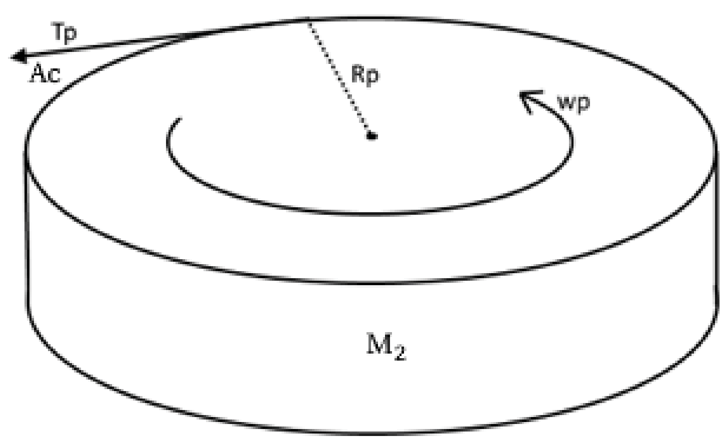


Figura 2-5: Esquema del péndulo (CMG) (T_p: Torque del péndulo, R_p: Radio del péndulo, w_p: Velocidad angular del péndulo, Ac: Aceleración de giro del péndulo, M_2: Masa del péndulo).

Cuando el robot esférico esta estacionado la ecuación que gobierna el movimiento del péndulo (Figura 2-5) para rotar es la siguiente.

$$T_p = 4 * R_p * M_2 * Ac \tag{2-15}$$

Esta fórmula solo toma en cuenta las 4 masas que irían en el péndulo ya que el peso de la estructura del péndulo en los extremos del radio en donde irían las masas es despreciable en comparación a las masas. Los valores R_p , M_2 y Ac son respectivamente el radio del péndulo, el valor de una masa del péndulo y la aceleración que se pretende que el péndulo se esté moviendo.

2.4 Simulaciones de robot esférico

Antes de empezar a usar el software V-Rep primero se debió crear un modelo del robot y sus distintas partes que se pretenden simular. Para esto, se usó el programa Autodesk Fusion 360 [12] el cual es una herramienta muy útil y sencilla para el modelado en 3D. La figura 2-6 muestra el modelo con sus diferentes componentes para la simulación en V-Rep.

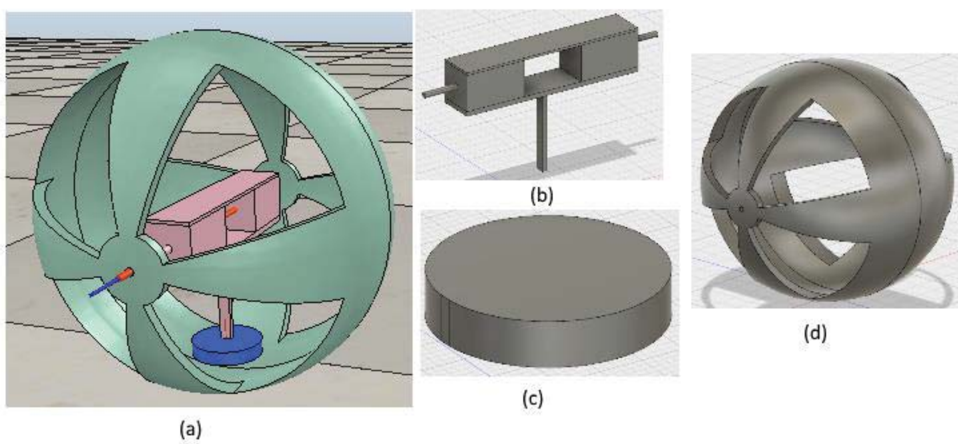


Figura 2-6: Modelo de robot y sus respectivas partes. (a) Modelo armado del robot Esférico en la simulación de V-Rep, (b) Caja de circuitería, (c) Péndulo, (d) carcasa esférica del robot esférico.

Estos se guardaron en formato .STL para ser exportados al programa V-Rep. En este se importaron los archivos y se prepararon siguiendo la guía en [13]. Luego se escribió el código para el robot para ser controlado en la simulación, los cuales se pueden encontrar en el apéndice según se diera el caso.

Se realizan simulaciones de un robot con diversos objetivos, uno de estos es el de comprobar el diseño, es decir, la factibilidad de este. También se realizaron simulaciones para saber cómo el robot se comportara en diferentes situaciones, como por ejemplo, una trayectoria en zigzagueo o una trayectoria pasando por encima de objetos de diferentes alturas, el éxito del robot en tales situaciones dependerá también del porque y para que se construye tal robot. Las simulaciones que se hicieron son las siguientes:

- Trayecto recto
- Trayecto en zigzag
- Trayecto subiendo planos inclinados
- Trayecto pasando por encima de objetos

2.4.1 Trayecto recto

En esta trayectoria se pretende que el trayecto sea lo más recto posible. Como se puede ver en la figura 2-7, se puede observar que el modelo en la simulación presenta una trayectoria un tanto curva.

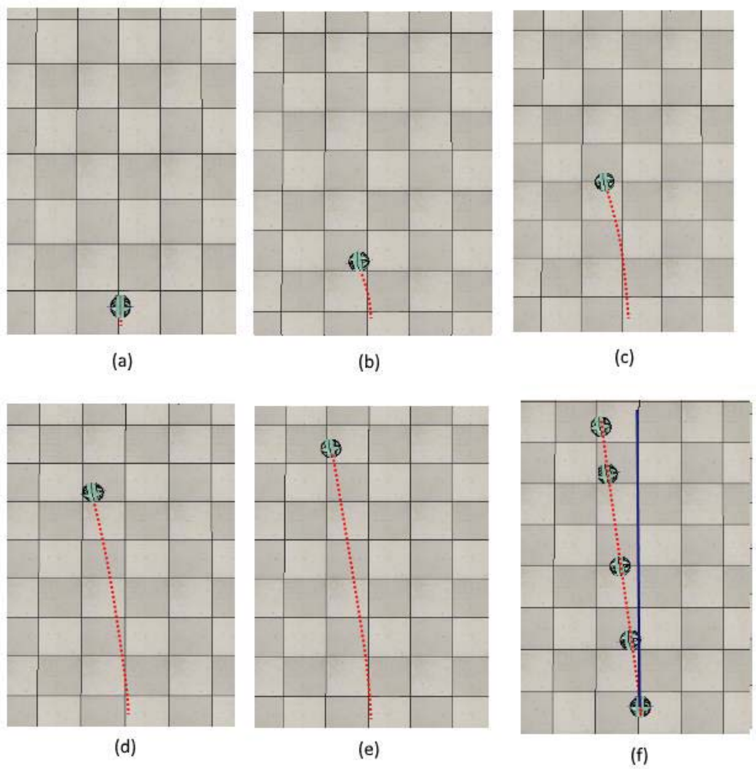


Figura 2-7: imágenes de simulación del robot esférico con supuesta trayectoria recta (a) a los 0 [seg], (b) a los 2 [seg], (c) a los 3 [seg], (d) a los 4 [seg], (e) a los 5 [seg], (f) línea roja punteada trayecto del robot, línea azul trayecto esperado antes de la simulación

Como se puede ver en la figura 2-7 el robot al principio no sigue una trayectoria recta, se buscó cual podría ser el problema y se llegó a 3 posibles causas la primera puede ser que el diseño del robot este mal balanceado lo cual, al principio, el robot al no tener tanto momento de inercia en su movimiento hace que este sea más susceptible al desbalanceo del diseño del robot, la segunda opción puede ser que alguna variable este afectando a la simulación y la última posible causa es que puede ser que el robot actúe así en la realidad, lo cual significaría hacer cambios mínimos en el diseño para asegurar en la estabilidad del trayecto.

En esta simulación no se utilizó ningún código, solo se activaron los motores del eje paralelo al suelo para hacer que se moviera hacia adelante.

2.4.2 Trayecto en zigzag

El objetivo de este trayecto es simular un movimiento en Zigzagueo para probar la movilidad que este robot pueda tener. Para esta simulación se utilizó un código para realizar tal movimiento. Se trató que el robot diera un giro cercano a los 45 grados o 50 grados. En la Figura 2-8 se puede apreciar el trayecto que toma el robot en la simulación.

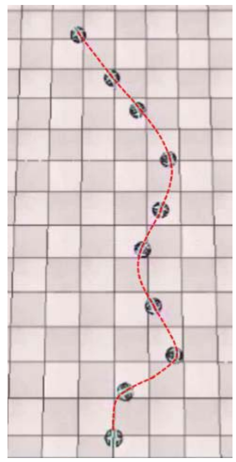


Figura 2-8: simulación del robot esférico evitando objetos

El código que se utilizó en esta simulación se puede observar en el apéndice.

2.4.3 Trayecto subiendo planos inclinados

Los planos inclinados siempre han sido un desafío para este tipo de robots, es por esto que se realizó una simulación del robot subiendo por una superficie inclinada (escalera) con 25, 30 y 35 grados de inclinación aproximadamente. Imágenes de la simulación se pueden observar en la figura 2-9.

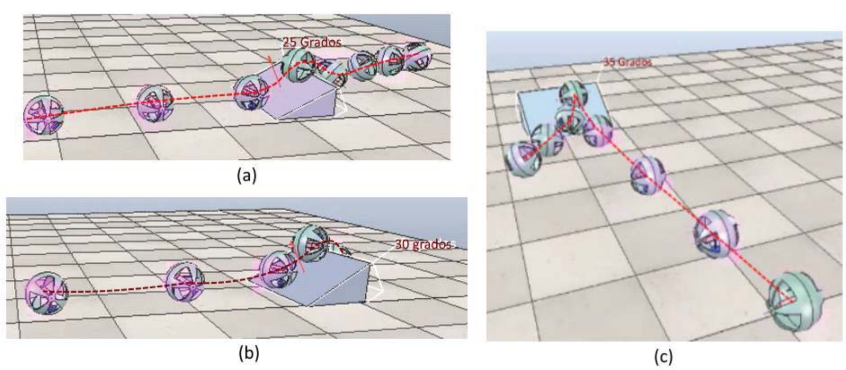


Figura 2-9: Simulación del robot esférico subiendo planos inclinados en diferentes momentos del trayecto, (a) trayecto en una subida con inclinación de 25 grados con respecto al suelo, (b) Trayecto en una subida con inclinación de 30 grados con respecto al suelo, (c)Trayecto en una con inclinación de 35 grados con respecto al suelo.

Como se puede ver en la figura 2-9(a) en donde la inclinación es de 25 grados de inclinación con respecto al suelo, el robot esférico pudo pasar sin mayor problema por el obstáculo, lo cual está entre lo esperado ya que para este diseño el límite de grados de inclinación que puede cruzar este tipo de robot es de aproximadamente de 30 grados con respecto al suelo. En la figura 2-9(b) se puede ver que, si bien el robot pudo pasar por el obstáculo de 30 grados de inclinación, se pudo notar que este se estaba casi deteniendo al llegar a la cima del obstáculo. En la figura 2-9(c) se pudo observar que el robot esférico no pudo sobrepasar el obstáculo de 35 grados de inclinación con respecto al suelo. Estos resultados están en lo que uno podría considerar como esperados considerando el mecanismo que usa este tipo de robot.

2.4.4 Trayecto pasando por encima de objetos

Se simuló también la opción de que el robot esférico pase por encima de objetos. En este caso, solo se hizo que pasara por cajas de distintas alturas para saber si este podría pasar por encima de estas y en consecuencia saber cuál es la altura máxima que el robot puede hacerse cruzar.

En la Figura 2-10 se puede ver la trayectoria del robot al pasar por encima de objeto de 1 [cm]. Como se puede ver en esta Figura el robot no tiene problema al pasar por esta altura, se espera que en las pruebas del robot ya construido este comportamiento sea el mismo.

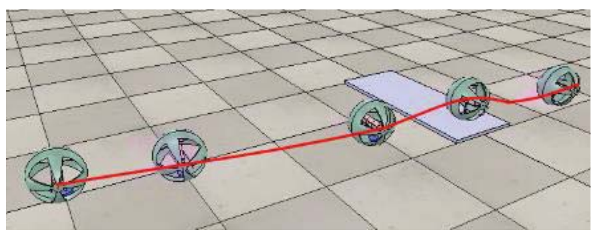


Figura 2-10: Trayectoria del robot al pasar por un objeto de 1 [cm].

Como se puede ver en esta Figura el robot no tiene problema al pasar por esta altura, se espera que en las pruebas del robot ya construido este comportamiento sea el mismo.

Luego se realizó una prueba (Figura 2-11) con un objeto de 1.7 [cm] de altura, en donde se puede ver que el robot tiene problemas para poder pasar por encima del objeto, esto está indicado por la cantidad de imágenes del robot mostradas en la figura 2-11, estas imágenes dan a conocer la cantidad de tiempo que el robot paso en un lugar, mientras más se ve el robot en un lugar, mayor es la cantidad de tiempo que el robot se quedó en un lugar en específico.

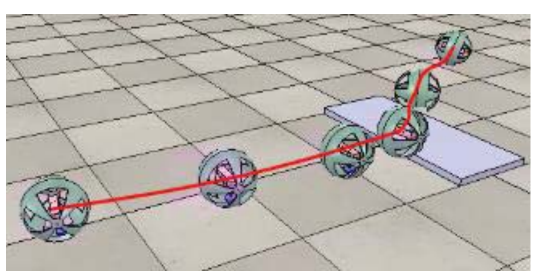


Figura 2-11: Trayectoria del robot al pasar por un objeto de 1.7 [cm].

En la siguiente simulación (Figura 2-12) se varió la altura del objeto a 1.8 [cm], lo que resulto que el robot en la simulación no pudiera pasar por encima del robot.

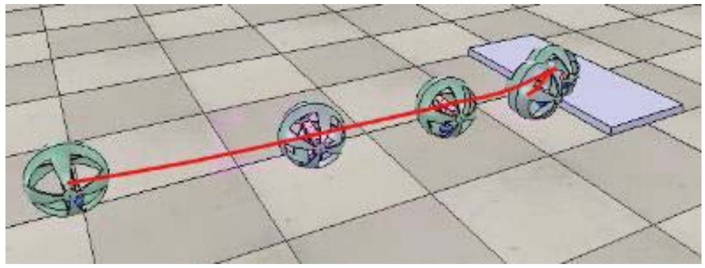


Figura 2-12: Trayectoria del robot al pasar por un objeto de 1.8 [cm].

Esto demuestra la altura máxima que el robot esférico en la simulación puede llegar a subir sin que su trayectoria se vea afectada en gran medida.

3 Construcción del robot esférico

3.1 Diseño de robot esférico

El diseño del robot esférico se obtuvo al tratar de imitar el diseño que se puede encontrar en la referencia [9], la razón de esto es que el que se mostró en tal enlace es uno simplista y fácil de recrear. Luego se llegó al diseño de la Figura 3-1.

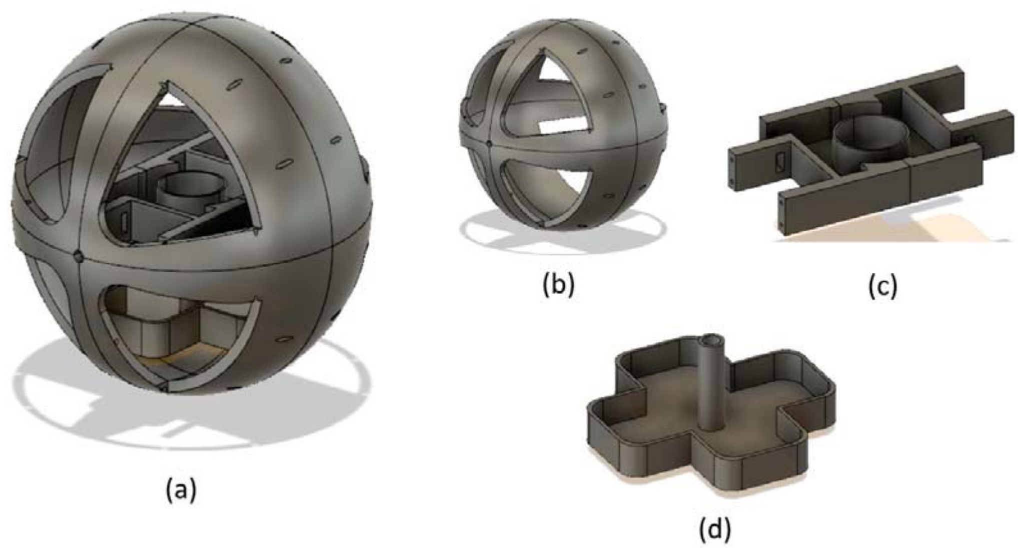


Figura 3-1: Diseño final del robot esférico, (a) robot armado, (b) Carcasa del robot esférico, (c) Base interior, (d) Péndulo.

El robot esférico que se optó por construir tiene un diámetro de 18 [cm] de exterior y 17 [cm] interior, desde un comienzo se pensó en usar PLA y una impresora 3D para crear la carcasa esférica y las partes estructurales del robot, esto se hizo por razones de que si se usaba otro material el trabajo sería más difícil y demoroso. Los componentes electrónicos que se estarían utilizando son los siguientes:

- Una tarjeta de programación con wifi NodeMCU V2 amica
- 3 x servomotores con giro continuo
- 1 x sensor giroscopio y acelerómetro (MPU6050)
- Un módulo driver para control de dirección del motor DC
- Un conversor DC-DC step-up XL600e
- Baterías 18560 4.2 [V].
- Cables

La configuración del circuito que ira dentro del robot se muestra en la figura 3-2

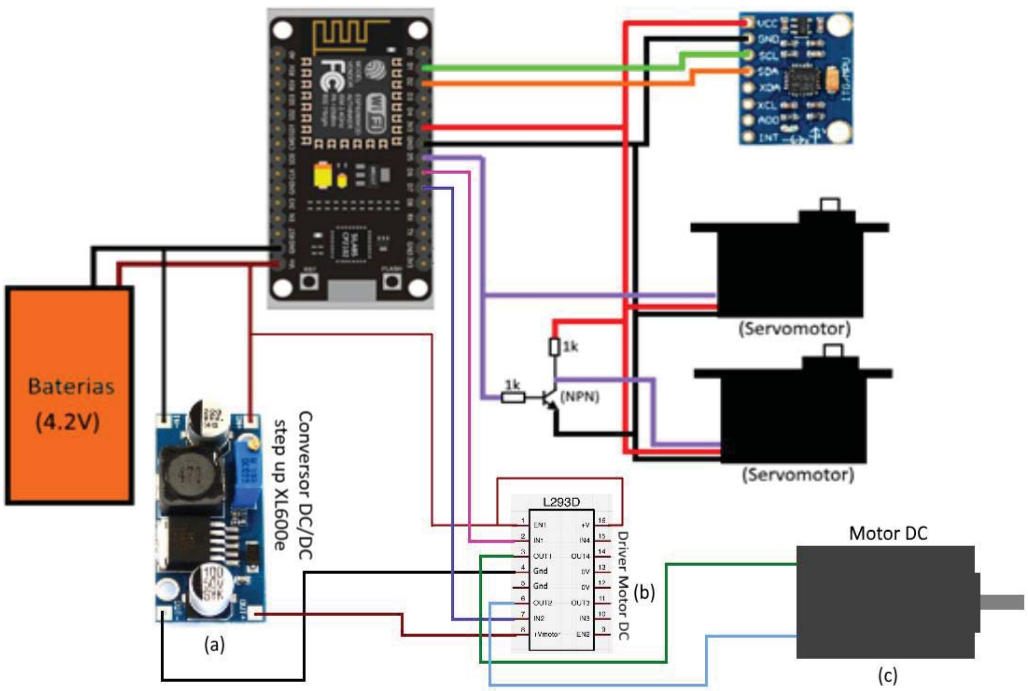


Figura 3-2: Circuito del robot.

El NodeMCU es una tarjeta de programación que tiene integrada un módulo ESP8266 que permite la comunicación por Wifi con otros dispositivos de forma inalámbrica, se utilizó esta tarjeta junto a un computador para controlar el robot. Adicionalmente se puede controlar el robot mediante una aplicación para el celular.

Se pretende colocar un sensor giroscopio y acelerómetro (Figura 3-3) de tal forma de que sea posible saber en qué dirección el robot se esté dirigiendo.

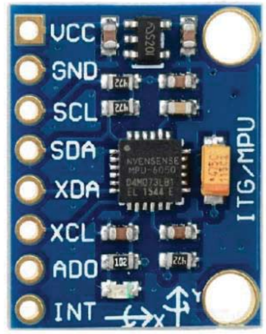


Figura 3-3: Sensor giroscopio/acelerómetro (MPU6050)

Los servomotores se modificaron para que tengan giro continuo, siguiendo la guía que se puede encontrar en [14]. Cabe señalar que la velocidad y dirección de giro de estos servos se controlan con una señal PWM. Para el movimiento de avance y retroceso se escogió los servos en base a la ecuación 2-12 que se mencionaron anteriormente.

$$T = a_1(J_2 - M_2re\cos(\theta_1 + \theta_2) + M_2e^2) + a_2(J_2 + M_2e^2) + M_2ge\sin(\theta_1 + \theta_2 + \alpha) \quad (2\text{-}12)$$

Los valores para calcular con la fórmula 2-19 se pueden encontrar en la tabla 2-2

Tabla 3-1: tabla de valores de peso del robot esférico

Parte del Robot	**Valores**
Péndulo (M_2)	0.292 [kg]
Carcasa esférico (M_1)	0.246 [kg]
Radio de la carcasa (r)	9 [cm]
Distancia entre el péndulo y el centroide de la carcasa esférico (e)	7 [cm]

El peso de los servos y la Bateria no se consideraron ya que estos están muy cerca del centroide de la carcasa y no harían un gran cambio en los valores buscados. Al considerar α con un valor de 30 grados, y además $a_1 + a_2 << 1$ según [10], se reemplazan estos valores en la formula lo cual nos lleva a la ecuación 3-2.

$$T = a_1\left(0.292[kg] * 9.8\left[\frac{m}{s^2}\right] * 0.07[m] * \sin 0 - 0.292[kg] * 0.09[m] * 0.07[m]\cos(0) + 0.292[kg](0.07^2)\right) + a_2\left(0.292[kg] * 9.8\left[\frac{m}{s^2}\right] * 0.07[m] * \sin 0 + 0.292[kg](0.07^2(m^2))\right) + 0.292[kg] * 9.8\left[\frac{m}{s^2}\right] * 0.07[m] * \sin(30) = a_1\left(-0.0004088\left[\frac{kg*m^2}{s^2}\right]\right) + a_2\left(0.0014308\left[kg * \frac{m^2}{s^2}\right]\right) + 0.100156[kg * \frac{m^2}{s^2}] \quad (3\text{-}2)$$

Como los valores de a_1 y a_2 son muy pequeños solo se consideran el valor $0.100156[kg * \frac{m^2}{s^2}]$, el cual convertido en [kg-cm] tiene un valor de 1,0213[kg-cm]. Este valor es el mínimo que se puede ocupar para los servomotores de avance y retroceso. Considerando que es posible que en proyectos futuros se agreguen dispositivos adicionales al robot se decidió utilizar los servomotores FUTABA S3003 (una de pequeña descripción de este se puede encontrar en la sección de apuntes) los cuales dan un torque de 3.17 [Kg-cm] cuando son alimentados con 4.2[V] y 4.1[Kg-cm] cuando son alimentados con 6[V].

Para elegir el motor que permitiría mover el péndulo rotatorio se utilizó la ecuación 2-15, esta ecuación entregara el valor mínimo que el motor DC debe cumplir. Al reemplazar los valores de los pesos en la formula se puede obtener el valor aludido. En donde R_p, en este caso es 5.9[cm], una masa M_2 del péndulo tiene aproximadamente 73 [grs] o 0.073 [Kg] y la aceleración Ac se pretende que sea 5.5 [m/s^2].

$$T_p = 4 * 0.059[cm] * 0.073[Kg] * 1.5\left[\frac{m}{s^2}\right] = 0.0947[Nm] = 0,9662[Kg * cm] \tag{3-3}$$

Este valor al compararlo con el valor de un motor DC de 12[V] cuyo torque es de 600[kg*cm] uno puede observar que el motor está sobredimensionado, lo que puede dar espacio a futuras modificaciones si bien se ve el caso.

El péndulo se diseñó con respecto a la formula anterior es decir se tomó en consideración las variables de la fórmula 2-18, es decir, la distancia entre los pesos y el centro de la plataforma del péndulo en donde irán los pesos del péndulo, los pesos y la longitud máxima del péndulo, se usó la distancia máxima posible entre el centroide de la carcasa esférica y la masa del péndulo por razones de poseer el mayor contra-torque en el movimiento de avanzado y retroceso. El diseño de este péndulo se puede observar en la figura 3-4.

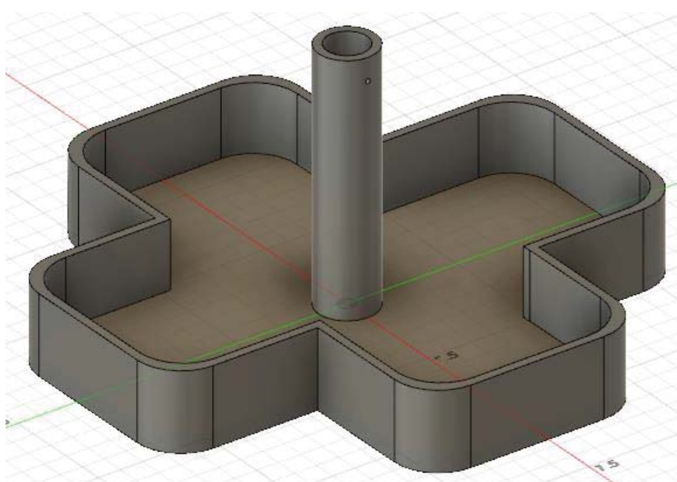


Figura 3-4: Diseño del péndulo de robot esférico.

El diseño del péndulo consiste en una plataforma en donde se acomodara 4 pesos de 73 [gramos] aproximadamente y una columna de 4.7 [cm] aproximadamente que se conectara al motor DC del péndulo, permitiendo que este mueva el péndulo de forma rotacional con respecto al eje ubicado en la columna del péndulo.

La base interior se diseñó de manera que pueda sostener el peso del péndulo y además mantener los servomotores, el motor DC y la circuitería fijos. Para llegar al diseño mostrado en la figura 3-5 se tuvo que considerar el peso posible del péndulo y planear con respecto a la posibilidad de que se podría aumentar el peso del péndulo, también se tuvo en consideración la simetría del diseño ya que si el diseño del interior no es simétrico el movimiento del robot podría verse afectado.

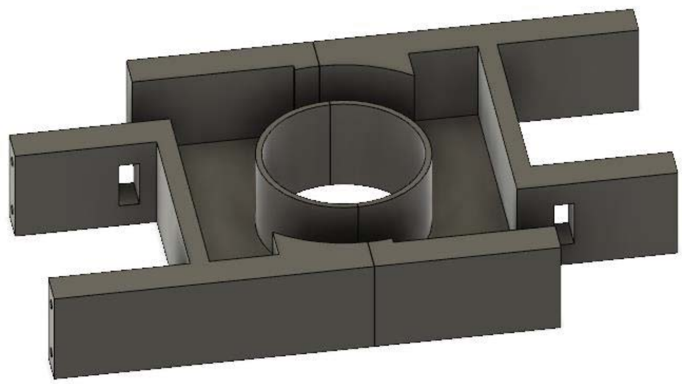


Figura 3-5: Base interior del robot esférico.

A los lados de esta base se colocaran los servomotores de avance y retroceso que permiten dicho movimiento, en la parte del medio se colocara el micro servomotor, el cual también estará mirando hacia abajo y sosteniendo al péndulo.

3.2 Construcción del robot esférico

En esta sección se empezó por la construcción de la carcasa esférica ya que esta se demoraba más en comparación a las otras partes del robot, luego se siguió por la base interior y se finalizó por el péndulo.

3.2.1 Construcción de la carcasa esférica

Como la dimensión de la impresora 3D disponible no podía imprimir ni una mitad de la carcasa esférica se optó por dividir la carcasa en 8 partes iguales (Figura 3-6(a)) de tal forma que fuese posible armar una esfera lo más simétrica posible.

La forma en que se mantienen juntas las partes y de manera precisa es a través de pequeños pedazos de alambre grueso que se colocan en orificios que se diseñaron con esta finalidad. La carcasa armada entera se puede ver en la figura 3-6(b).

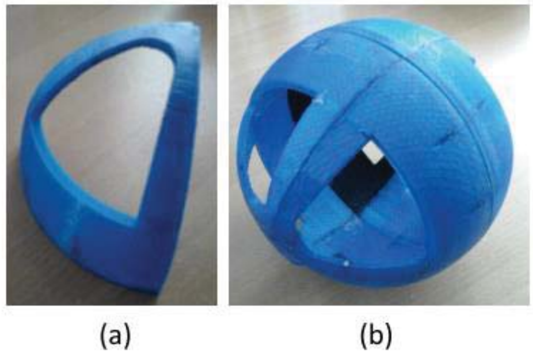


Figura 3-6: (a) 1/8 de la carcasa esférica, (b) Carcasa Armada completa.

Las partes se unieron inicialmente con alambres gruesos y silicona caliente para después ser unidas derritiendo las junturas, de tal modo de que se aumentara la rigidez de la carcasa.

3.2.2 Construcción de la base interior

La base interior del robot se diseñó de modo que se pueda colocar los servomotores de avance y retroceso, el motor DC del péndulo y la circuitería de manera fija y que además solo el eje de los servomotores de avance y retroceso estén de manera horizontales y que se conecten en la carcasa esférica. La base interior fue diseñada robusta con el objetivo de soportar el peso del péndulo y mantener en posición el motor DC y los servomotores que utiliza el robot. En la Figura 3-7 se muestra el interior de la base con la circuitería ya armada y además están mostrados los diferentes componentes del robot.

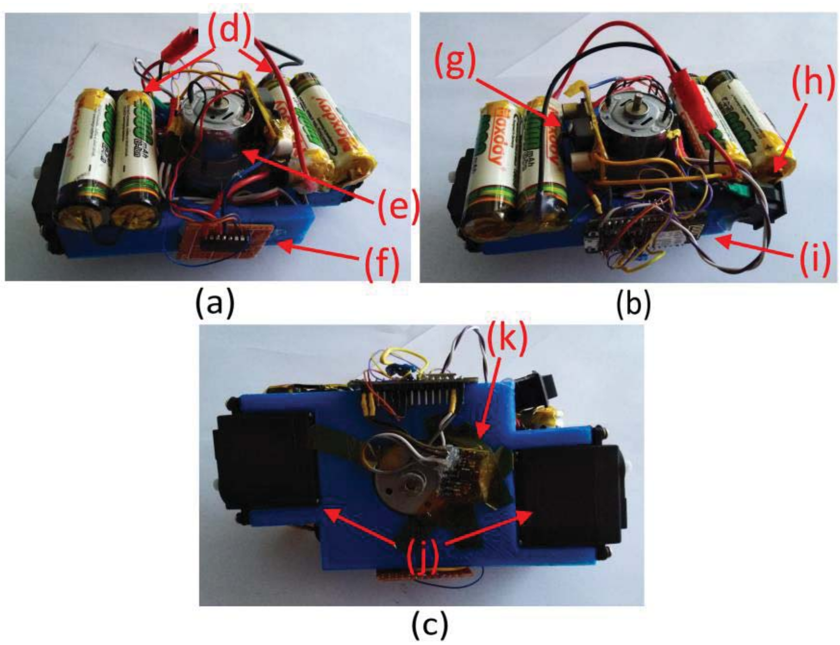


Figura 3-7: Base interna del robot esférico, (a) Vista frontal, (c) Vista trasera, (c) Vista inferior, (d) Baterías 18560, (e) Motor DC, (f) Driver motor DC, (g) Conversor step-up, (h) interruptor, (i) Tarjeta NodeMCU, (j) Servomotores de avance y retroceso, (k) Sensor giroscopio/acelerómetro MPU6050.

Cabe señalar que el sensor giroscopio se encuentra ubicado en la parte inferior de la base interna, como se muestra en la Figura 3-7(c)(k).

3.2.3 Construcción del péndulo

Siguiendo el diseño del péndulo mencionado anteriormente, este fue construido usando una impresora 3D con filamento PLA. La Figura 3-8 muestra el péndulo construido y con los 4 pesos de ~73 [grs] cada uno, incorporados en este. Para asegurar el péndulo al motor se colocaron pequeños tornillos que al aprovecharse de la irregularidad del engranaje del motor DC pudieron mantener el péndulo a sujetado al motor DC.

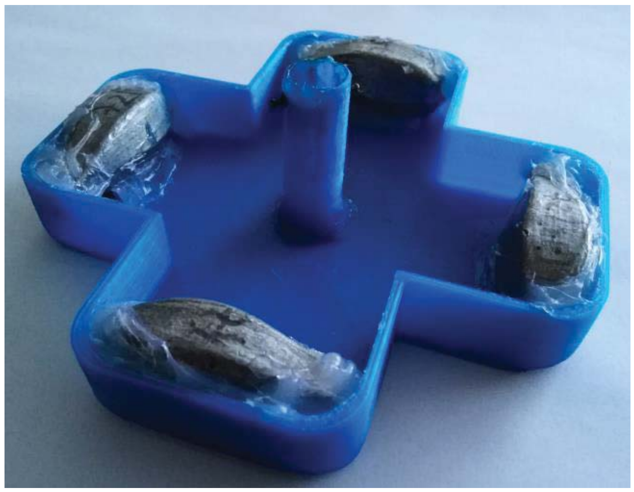


Figura 3-8: Péndulo construido con los pesos incorporados.

Cabe mencionar que se tuvo que calentar el engranaje del motor y presionarlo a la abertura en donde se a sujetaría el péndulo, esto se hizo con el propósito de derretir un poco el plástico de tal forma de dar a la abertura un mejor ajuste para conectar el péndulo a el motor DC.

3.3 Programación del robot esférico

Para controlar el robot por la red inalámbrica desde el celular, se utilizó una aplicación IOT llamada Blynk [15] (Figura 3-9) la cual tiene una interfaz de aplicación para celular muy intuitiva y fácil de usar, además, se utilizó en conjunto una tarjeta de programación llamada NodeMCU [16], esta tiene integrada un módulo wifi ESP8266.

Blynk

Figura 3-9: Logo de la aplicación Blynk

Para programar la tarjeta NodeMCU se utilizó Arduino IDE [17] ya que este tiene compatibilidad con este tipo de tarjeta y la aplicación Blynk también trabaja con este software, en la página de Blynk de documentos [18] se puede encontrar las librerías que se deben agregar en el Arduino IDE, además de la documentación y ejemplos que ayudan en la programación de este tipo de tarjetas y aplicación.

El método en que se controló en principio los servomotores fue programando desde el celular una señal PWM en los pines en los que están conectados los servos, para uno de los dos servos de avance y retroceso se tuvo que colocar una lógica negativa en el cable de señal de uno de ellos ya que con la posición en que este se encontraba el servo en el diseño de la base interior este giraría de forma opuesta si se enviaba la misma señal que el otro servo de avance y retroceso.

La lectura del sensor Giroscopio/acelerómetro (MPU6050) [19](Figura 3-3), se realizó con ayuda del video en [20] y el link que contenía el video en su comentario, se utilizó el código que el video y el link proporcionaba y se intentó modificarlo para programarlo en la tarjeta NodeMCU junto con el código de programación Blynk, desgraciadamente no se pudo lograr obtener una lectura coherente, por lo que se desechó por el momento utilizar el sensor y se decidió utilizar la cámara de la plataforma para obtener la orientación del robot.

3.4 Conexión del robot con el computador

Esta parte del proyecto requirió hacer que un computador pueda controlar el robot esférico, de tal modo que el robot pueda ser probado desde el computador en una plataforma de prueba (Figura 3-10).

Figura 3-10: Plataforma de prueba con el computador con cámara y computador para controlar robots.

La comunicación entre el robot y el computador de la plataforma funciona de la siguiente manera, antes que nada el robot esférico tiene que estar conectado a el servidor local de Blynk, después el computador de la plataforma informa al robot a través del servidor local la posición de robot, la dirección en que este está apuntando y la posición que se quiere llegar (posición objetivo), luego el robot toma esos datos, calcula el ángulo al cual debe apuntar y acto seguido intenta girar en dirección de la posición objetivo y avanza hacia esta.

3.4.1 Resolución de problema de comunicación de aplicación Blynk

La aplicación que se utilizó en el robot fue la misma que se utilizó en la construcción del robot, la aplicación Blynk [18] [15] tiene comandos api (application programming interface) [21] que permiten enviar datos mediante una dirección URL. Tales series de comandos fueron llamados por el desarrollador como "Blynk HTTP/S RESTful API" [22]. Para una exitosa comunicación entre el robot y el computador se creó un servidor local. En la página Web de la aplicación Blynk se encontró diferentes formas de crear un servidor local utilizando diferentes dispositivos y sistemas operativos, entre estos estaban Windows, Ubuntu y Raspberry pi. Se utilizó una Raspberry pi [23] para crear le servidor.

Figura 3-11: Tarjeta Raspberry pi 2 [24].

Se decidió crear un servidor local en una Raspberry pi 2 (Figura 3-11) del laboratorio, utilizando la Raspberry pi 2 se pudo crear el servidor sin mayor problemas. Los pasos a seguir para establecer el servidor local de Blynk se encuentran en apéndices. Ingeniería civil electrónica

La plataforma en la que se probó el robot esférico está compuesta de una superficie lisa cubierta por un tela negra, una cámara que toma imágenes/videos desde una posición elevada a tal superficie y un computador, para poder controlar al robot se usó el software Ejs (Easy java simulations) el cual puede comunicarse con el robot mediante direcciones URL que "Blynk

HTTP/S RESTful API" proporciona para mandar valores que harán que el robot se mueva hacia una posición específica, luego para conocer la posición del robot se utilizó la cámara ya antes mencionada junto a un software de captura de movimiento llamado Swistrack [25].

3.5 Configuración de procedimiento de imagen en el software Swistrack

Antes que nada se configuro el software Swistrack para que pudiera reconocer el robot esférico para esto se requirió un método de determinar la posición y dirección del robot, como el robot es esférico y su carcasa siempre está en movimiento cuando se traslada de un lugar a otro, se decidió procesar la señal de video de la cámara de la plataforma en color y así colocar una cinta de un color en un lado del robot y en el otro lado de este otra cinta de diferente color (ver Figura 3-12), esto se hiso con motivo de que cuando el computador reconozca los colores, no solo la ubicación del robot ininterrumpidamente sino que también la dirección que este está tomando, cabe también mencionar que se pintó el robot negro para que el color de su cuerpo no cause alteraciones de las mediciones.

Figura 3-12: Robot esférico negro con cinta de color rojo y verde para su reconocimiento en software Swistrack.

Los diferentes procesamientos que se utilizó en la imagen de la cámara de la plataforma en el software Swistrack fueron los siguientes en forma cronológica (ver Tabla 3-2): primero se partió por la imagen obtenida de la cámara, a esta imagen se aplicó un filtro de conversión de color, luego a la imagen resultante se aplicó un filtro llamado "Adaptive Background Subtraction" el cual elimina el fondo de una imagen y lo reemplaza con un negro, este proceso solo realiza una sola vez cuando se presiona el botón de "run" del programa es por esto que es importante de que los objetos que se requieran reconocer no estén en la plataforma en un principio, para que los procesos no este corriendo todo el tiempo se agregó una función llamada "Timer Trigger" el cual controla el tiempo en que se obtienen imágenes de la cámara para ser procesadas, el proceso que viene después es el llamado "Red-Green Marker Detection" [26] el cual los colores rojo y verde de los marcadores del robot para detecta la posición y la orientación del robot en la plataforma, el proceso siguiente es el denominado "Calibration with a Linear Metod" que hace coincidir las imágenes conseguidas por la cámara con las coordenadas reales de la plataforma, el siguiente proceso se llama "Dynamic Nearest Neighbor Tracking" que rastrea y marca por poco tiempo el trayecto que el robot realiza. Para finalizar se mandó los valores obtenidos después del penúltimo proceso a clientes TCP mediante el proceso llamado "Output Particules".

Tabla 3-2: procesos utilizados en el software Swistrack

Input from USB Camera
Conversion to Color
Adaptative Backgroung Subtraction Color
Timer Trigger
Red-Green Marker Detection
Calibration with a Linear Metod
Dynamic Nearest Neighbor Tracking
Output Particules

3.6 Configuracion de codigo de Ejs

Se analizó la configuración que se tenía para los robots khepera que se encontraba en el software Ejs y se modificó para enviar datos al robot para así controlar hacia donde se dirigiría. Al estudiar "Blynk HTTP/S RESTful API" se aprendió que se pueden utilizar direcciones URL para enviar y recibir valores de los diferentes pines de la tarjeta NodeMCU, con esto es posible mandar las coordenadas del robot, la orientación de este y las coordenadas objetivas a las que se quiere mandar al robot, se enviaron estos datos porque se planea que el robot esférico realice las operaciones necesarias para llegar a las coordenadas objetivos.

3.7 Explicación del funcionamiento del sistema

El sistema del robot para que este pueda funcionar adecuadamente está dividido en las siguientes partes, el robot, el servidor local Blynk y la plataforma que contiene además una cámara y un computador.

Para que el sistema sea más entendible, en la Figura 3-13 se muestra un diagrama que muestra las conexiones entre los diferentes partes. La razón que se utilizó un servidor local fue simplemente por razones de la red inalámbrica de la universidad no permite que el robot se conecte al servidor externo de la aplicación Blynk [18], afortunadamente tal aplicación tiene instructivos de como montar un servidor local haciendo más fácil el proceso de implementar el control del robot.

Originalmente la aplicación con que se controla el robot solo estaba pensada para smartphones por decisión de los creadores de tal aplicación, sin embargo tales desarrolladores habían creado una forma de controlar variables en el servidor mediante direcciones URL [22] con lo cual se puede usar para controlar al robot desde un programa de computador en códigos java, el programa que se uso es el llamado "Ejs" (Easy java simulations) [27].

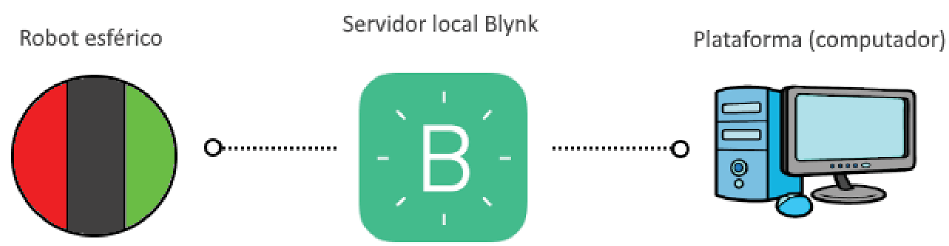


Figura 3-13: Diagrama del sistema de robot esférico.

Primero para explicar el diagrama de forma general, el robot debe estar conectado al servidor local Blynk, este servidor sirve para almacenar las variables que el robot lee y si se da la oportunidad el robot también puede cambiar tales variables. El computador, mediante la cámara de la plataforma y un programa de rastreo de robot (Swistrack) [25], puede saber la posición del robot en la plataforma y entregar estos valores a otro programa para enviar los datos al servidor, tal programa es el Ejs (Easy java simulations), el servidor almacena estos valores que recibió del computador y los guarda para que el robot los lea y realice las acciones que está programado a hacer, como rotar hacia la dirección en donde queda el objetivo y después de esto avanzar hacia el objetivo.

Para dar una explicación más específica de los diferentes procesos del sistema se requerirá que uno se enfoque en la Figura 3-14, en esta figura se pueden ver los procesos del sistema en más detalle, lo cual se espera que con esto se deje más claro el funcionamiento de cada parte del sistema. El proceso del sistema comienza en la plataforma, más específicamente en el computador, se inicia los programas necesarios con los códigos y configuraciones anteriormente hechos y luego se revisa si el la condición de conectado, si esta es "False", se envía una señal de detenido al robot y se vuelve a esperar la conexión, si la condición es "True", se obtienen imágenes de la cámara de la plataforma, se procesan las imágenes y se obtienen los datos de posición y orientación del robot además de la posición a la que se quiere que el robot se mueva, luego se los datos son enviados al servidor local Blynk en donde se almacenan y esperan ser llamados por el robot, al pedir y recibir los datos del servidor el robot los procesa, es decir, calcula la orientación a la que el robot tiene que llegar para que el robot este mirando a las coordenadas objetivos, una vez que hace esto compara el ángulo calculado con el ángulo entregado por la plataforma, si este no es el mismo el robot accionara el péndulo rotatorio e intentara corregir el ángulo rotando a la derecha o a la izquierda, si el ángulo no necesita corrección el robot avanzara en la dirección de las coordenadas objetivos.

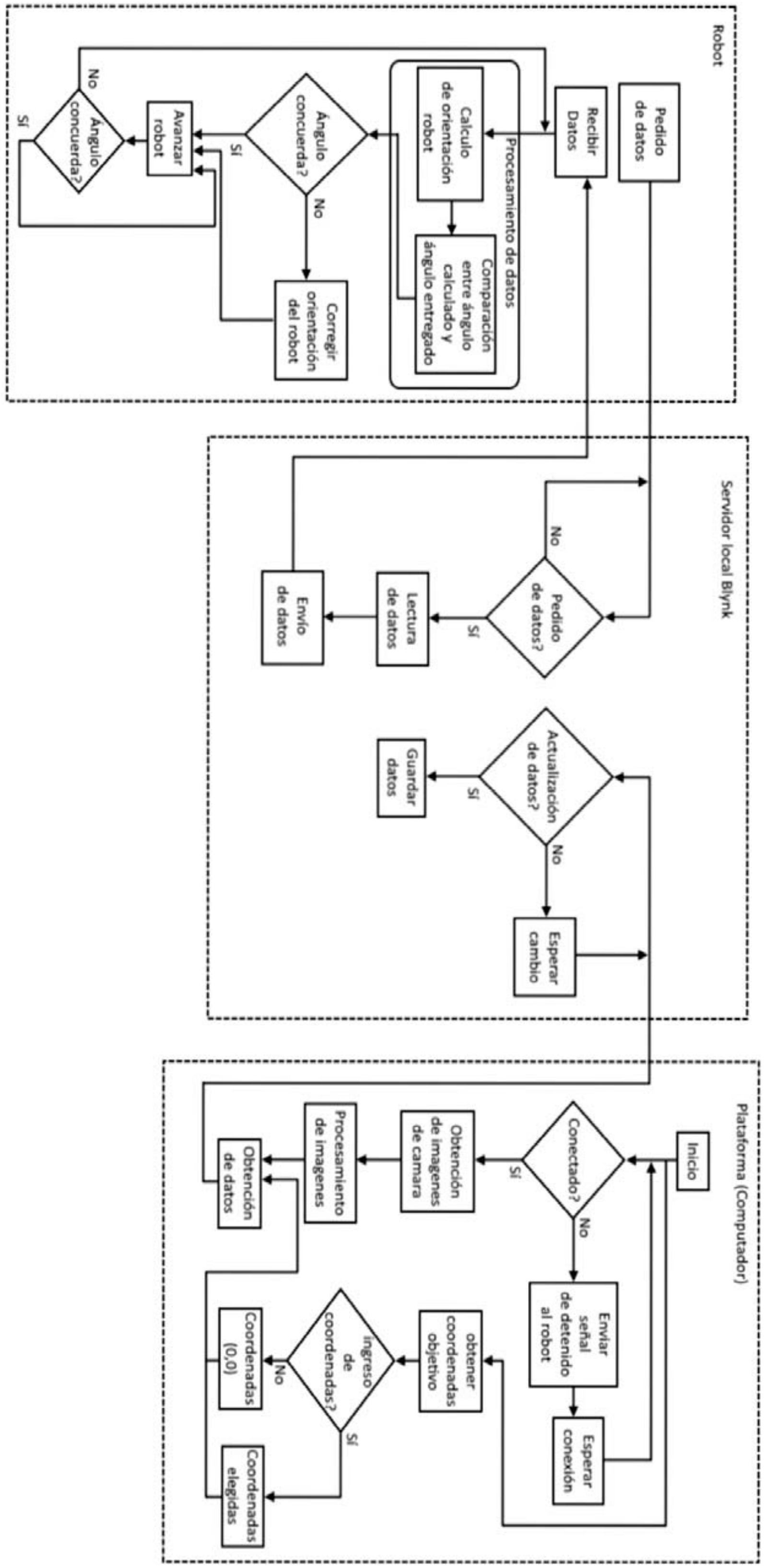


Figura 3-14: Diagrama con más detalle del sistema.

3.8 Comportamiento del robot

En este proyecto, el robot puede seguir dos tipos de caminos en lo que respecta al cálculo de valores y tomas de decisiones, en estos procesos está involucrado, generalmente, el emisor de una señal que impulsa al robot a moverse y el robot. El primer tipo de camino que un proyecto puede seguir es que el mismo emisor de la señal calcule los valores de orientación con los valores de posición que entrega el software Swistrack y que envié las direcciones a las que el robot tenga que moverse, el segundo tipo es que el emisor de la señal solo entregue información de posición del punto objetivo y la posición del robot esférico y el robot al recibir esta información calcule la orientación a la cual se tiene que mover. En este caso se utilizó el último tipo de comportamiento ya que a uno le es más fácil programar con arduino que con java.

3.8.1 Fórmulas para orientación con respecto al robot

El robot está programado para leer valores de las variables en el servidor local, estas variables son: coordenadas del robot en la plataforma, orientación del robot y coordenadas del punto objetivo, es decir, coordenadas del punto a donde se quiere que el robot se dirija, una vez que el robot ha leído los valores este puede calcular la dirección en donde está el punto de objetivo (figura 3-15) usando trigonometría, tales formulas se puede ver desde la ecuación 3-4 hasta la ecuación 3-7.

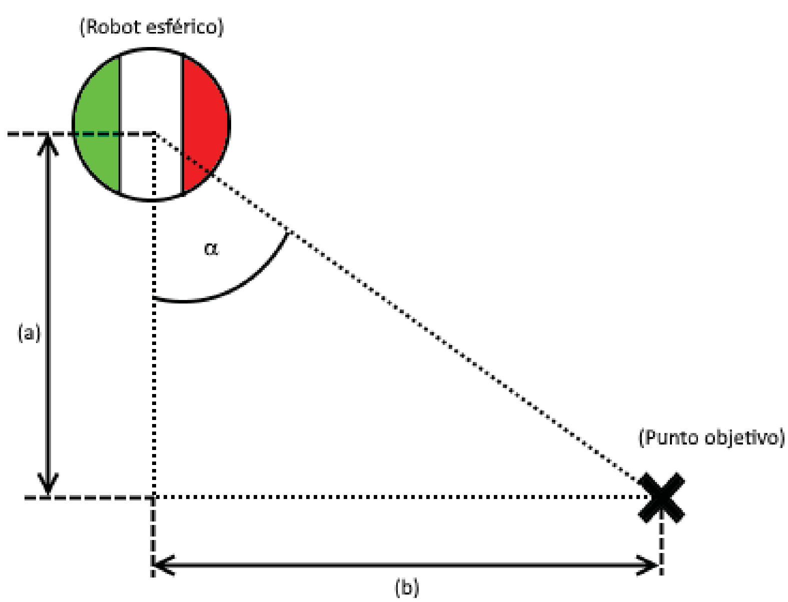


Figura 3-15: Ilustración de robot esférico y punto objetivo con variables asociados para formulas (a) coordenada X entre el robot y el punto objetivo, (b) coordenada Y entre el robot y el punto objetivo, (α).

Para mejor ilustración de los cálculos que se tienen que realizar en las diferentes regiones alrededor del robot esférico, esto esta se puede observar en la Figura 3-16.

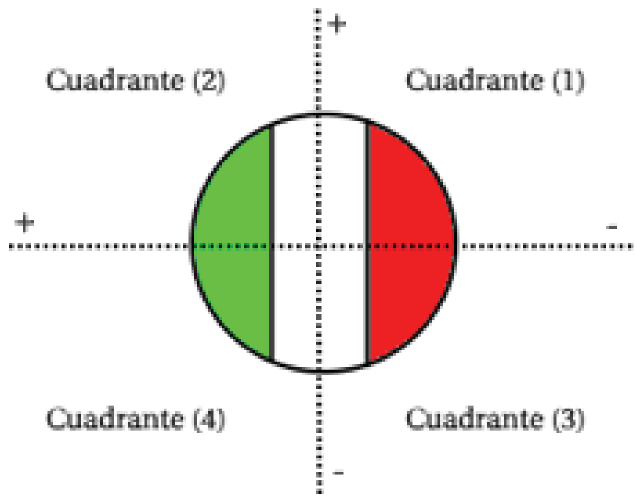


Figura 3-16: Enumeración de las diferentes regiones alrededor del robot esférico.

$$\alpha = 180 + \tan^{-1}\left(\frac{b}{a}\right) \qquad para\ cuadrante\ 1 \tag{3-4}$$

$$\alpha = \tan^{-1}\left(\frac{b}{a}\right) \qquad para\ cuadrante\ 2 \tag{3-5}$$

$$\alpha = 180 + \tan^{-1}\left(\frac{b}{a}\right) \qquad para\ cuadrante\ 3 \tag{3-6}$$

$$\alpha = 360 + \tan^{-1}\left(\frac{b}{a}\right) \qquad para\ cuadrante\ 4 \tag{3-7}$$

Luego de calcular la dirección que el robot debe tomar para llegar al punto objetivo, el robot, antes que nada, este de rotar en una dirección hasta alcanzar la orientación calculada y cuando llegue a tal orientación este se moverá hacia adelante hasta llegar a la coordenada objetivo. El código que se utilizó para el robot se puede encontrar en el apéndice de este informe, así también para el código que se ocupó en el computador de la plataforma.

4 Pruebas del robot esférico

4.1 Pruebas de trayectorias

Las pruebas que se realizaron al robot fueron con motivo de probar cuales son las limitantes del robot, gracias a estas más adelante es posible que se realicen controles en su movimiento o también para considerar algunos aspectos por si se hace otra versión del robot. No se realizó pruebas con respecto a si el robot podía subir por objetos inclinados ya que no se consideró que fuese necesario porque sólo se quiere probar el control que se tiene en el robot en una superficie plana y sin grados de elevaciones de este.

4.1.1 Prueba 1: Trayectoria recta

La prueba se llevó a cabo colocando el robot en una superficie relativamente pareja, en esta se puede ver (Figura 4-1) que el trayecto que el robot recorre tienen poca variación, parecido a lo que en la simulación se observó.

Figura 4-1: Trayecto recto del robot esférico

Cabe también mencionar que, si bien el robot recorrió una trayectoria semirrecta este también tuvo mucha oscilación en movimiento de avance y también en el movimiento de la carcasa, lo cual no se puede mostrar en una fotografía ya que su movimiento no es tan evidente en una fotografía, este movimiento podría deberse a que la carcasa del robot no este pareja en las uniones de sus partes como uno esperaba.

4.1.2 Prueba 2: Maniobrabilidad

En esta prueba se probó la capacidad de realizar un trayecto en zigzag, lo cual requiere que el robot realice giros y que también avance. En la figura 4-2 se puede ver la trayectoria realizada en la prueba de maniobrabilidad.

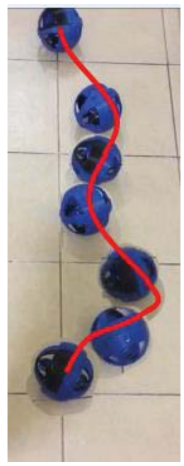


Figura 4-2: trayecto zigzag del robot esférico.

El trayecto fue realizado por control manual, es decir, no se programó al robot para que siguiera una trayectoria predeterminada, si no que se dio instrucciones al robot en qué dirección virar y en qué tiempo debería hacerlo. Como se puede ver en la figura 4-2, la trayectoria no fue un zigzag perfecto, esto se puede deber a que las señales se dieron de forma manualmente por lo que esto introduce error a la trayectoria esperada, aunque vale también mencionar que el robot realizo una trayectoria zigzag.

4.1.3 Prueba 3: Trayectoria pasando encima de objetos

Para poder realizar esta prueba se debe primero saber cuál es la altura teórica máxima que el robot puede pasar, es por esto que se utilizó una ecuación que en el informe anterior se mostró y que sirve para determinar esta altura. Tal altura se puede observar en la ecuación 4-1.

$$h_{Max} = r - \sqrt{r^2 - \left(\frac{eM_2}{M_1 + M_2}\right)^2} \tag{4-1}$$

Se reemplazan los valores de la ecuación 4-1 con los valores de la tabla 3-1 para obtener la altura máxima que el robot puede subir, el resultado se encuentra en la ecuación 4-2.

$$h_{Max} = r - \sqrt{r^2 - \left(\frac{eM_2}{M_1 + M_2}\right)^2} = 9 - \sqrt{9^2 - \left(\frac{7 * 0.292}{0.246 + 0.292}\right)^2} = 0.81[cm] \quad (4\text{-}2)$$

Este valor ahora se comparara con las pruebas que se realizaron al robot. En la figura 4-3(a) se puede observar que el robot no puede subir por el objeto de 0.83 [cm], lo que está de acuerdo con el resultado anterior.

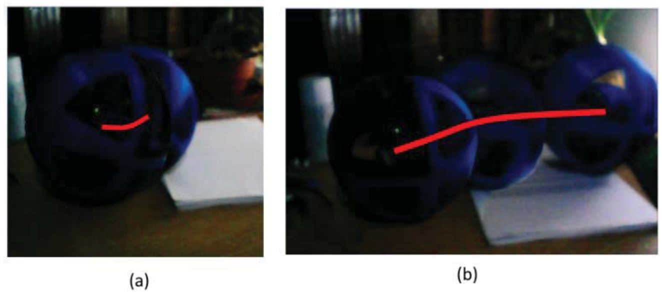


Figura 4-3: Pruebas de trayecto del robot subiendo por objetos, (a) robot subiendo por objeto de 0.83 [cm], (b) robot subiendo por objeto de 0.71 [cm].

En la figura 4-3(b) se puede ver que el robot pudo subir por el objeto de 0.71 [cm] lo cual está de acuerdo al valor que se obtuvo de la ecuación 4-2, esto demuestra que esta fórmula es confiable con respecto a este tipo de cálculo.

4.2 Prueba 4: Reconocimiento de robot por software Swistrack

En esta prueba se utilizó el software Swistrack para reconocer la posición y dirección del robot en la plataforma. Ya habiendo hablado de los procesos para reconocer al robot en el capítulo anterior, se hizo una prueba en donde se implementó todos los procesos de los que se habló anteriormente y se mostró la imagen del robot resultante en la figura 4-4.

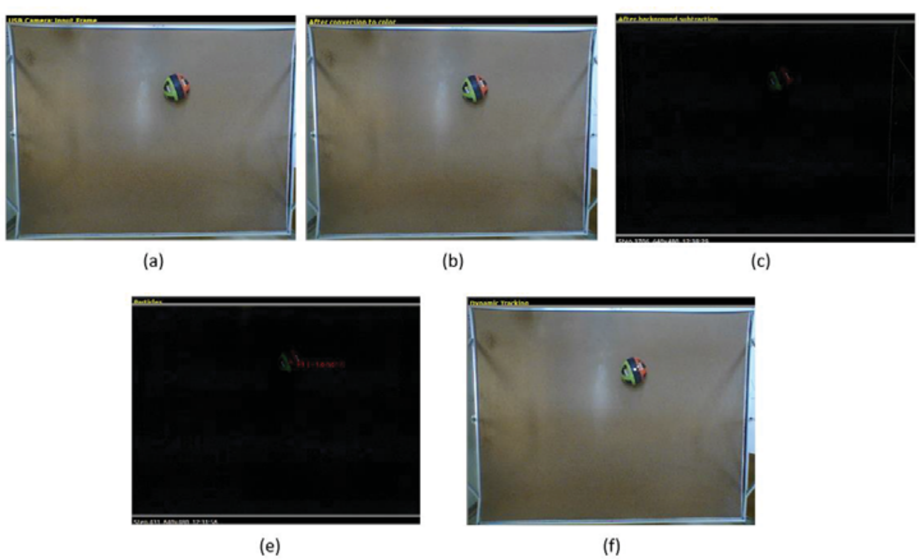


Figura 4-4: Salida de los procesos utilizados, (a) Input from USB camera, (b) Conversion to color, (c) Adaptative Backgroung Subtraction Color, (d)Red-Green Marker Detection, (e)Dynamic Nearest Neighbor Tracking.

Como podemos ver en la figura 4-4 esta configuración de procesos hacen posible el reconocimiento del robot. Cabe decir que cuando el robot se mueve de una posición a otra el programa en ocasiones pierde la posición de este y también no capta bien la orientación del robot.

4.3 Prueba 5: Reconocimiento de posición mediante software Swistrack

El reconocimiento del robot mediante el software Swistrack es una parte importante del proyecto, esto es porque sin este, el programa Ejs no podría dar los datos necesarios al robot para realizar el trayecto hacia las coordenadas deseadas. Para las pruebas realizadas se utilizaron posiciones conocidas de tal forma que al colocar al robot en estas posiciones se pudo comparar los valores obtenidos con los que deberían haber sido. En las figura 4-5 se puede observar las pruebas realizadas.

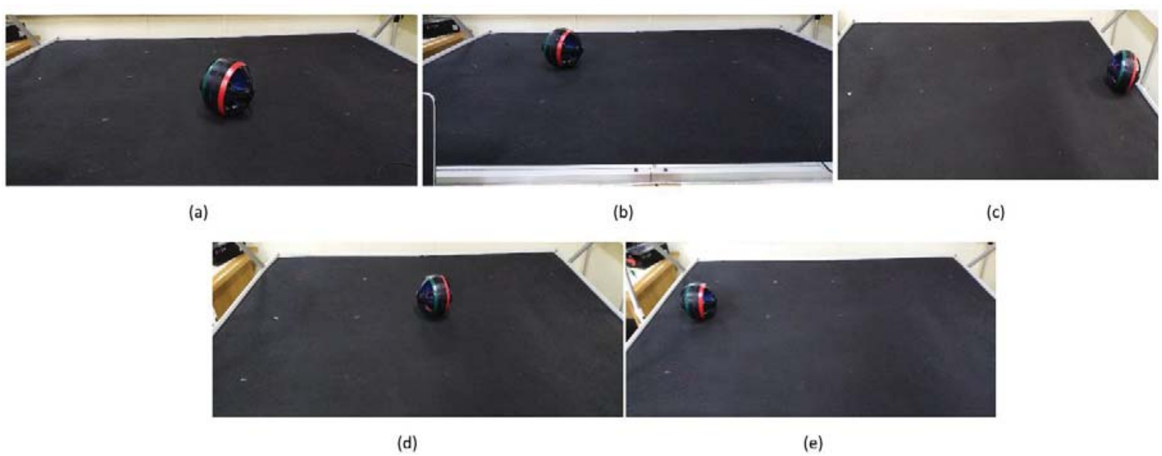


Figura 4-5 : Pruebas de reconocimiento de posición del robot esférico, (a) posición marca (0,-40), posición robot (-1,-40), (b) posición marca (-40,0), posición robot (-42,-3), (c) posición marca (-80,0), posición robot (-86,-5), (d) posición marca (0,0), posición robot (5,1), (b) posición marca (80,0), posición robot (80,1).

Si bien las coordenadas del robot se acercan bastante a las de las posiciones de las marcas, el sistema tiene problemas cuando el robot se mueve de una posición a otra, el problema aludido es cuando el robot se mueve hay interrupciones del reconocimiento del robot, es decir, la marca del robot desaparece y reaparece en periodos de tiempos pequeños lo que introduce errores en los cálculos, además cabe mencionar que el reconocimiento de orientación es demasiado inestable por lo que se necesita un mejoramiento.

4.4 Prueba 6: Giro con objetivo de 90 grados

El objetivo de esta prueba es saber si al perturbar la orientación del robot en 90 grados, el robot puede volver a la misma orientación sin ayuda. Para esta prueba se posiciono el robot apuntando inicialmente hacia los 90 grados (Figura 4-6) según el punto de vista desde la cámara de la plataforma, cabe decir que el robot siempre estará tratando de apuntar a los 90 grados, luego se giró aproximadamente -90 grados el robot para ver como este respondía al cambio. Los resultados de esta prueba se pueden ver la figura 4-7.

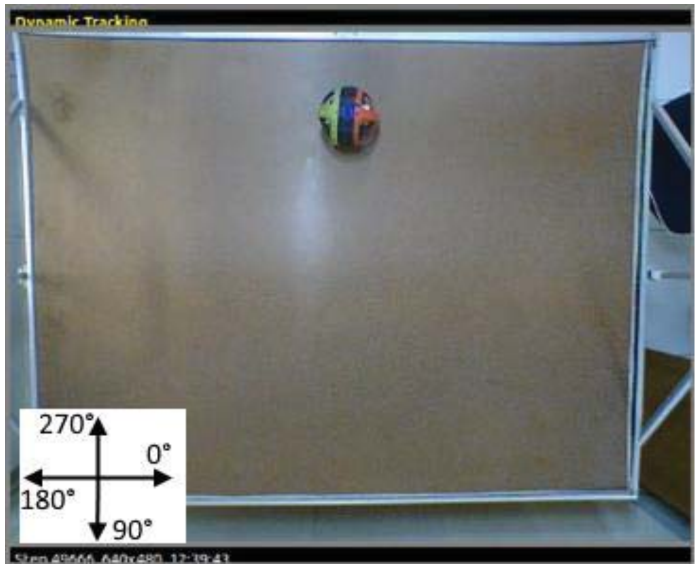


Figura 4-6: Posicionamiento del robot en la plataforma.

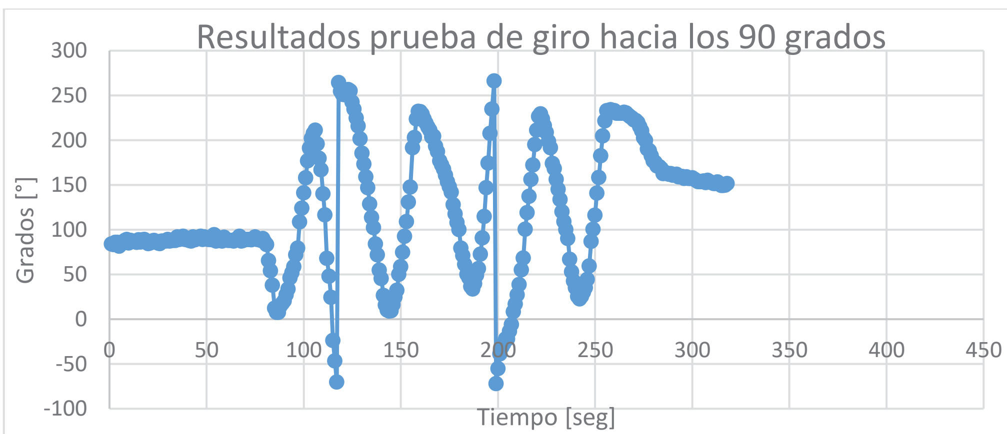


Figura 4-7: Grafico de resultados de prueba.

En la Figura 4-7 podemos ver que el resultado del giro, si bien se observa que el robot puede girar hasta más allá de los 90 grados con poco problema, este tiene problemas en la estabilidad por lo que se necesitaría un control para poder manejar el giro del robot apropiadamente.

4.4.1 Prueba de giro con objetivo de 180 grados

El objetivo de esta prueba es observar que pasa con el robot cuando tiene que girar 180 aproximadamente. Los resultados de esta prueba se pueden ver en la figura 4-8.

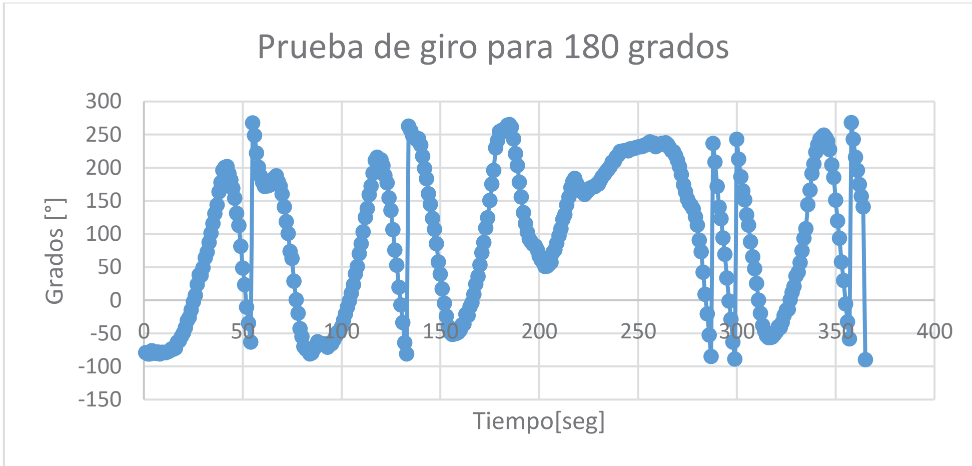


Figura 4-8: resultados de prueba de giro para 180 grados

Como se puede ver el robot realiza un movimiento giratorio un tanto sinusoidal, para poder arreglar este tipo de comportamiento se reitera que es necesario un tipo de sistema de control.

4.5 Propuesta de control del robot esférico

Dada la inestabilidad del robot al moverse se proponen un control de posición y un control de orientación. El primero para mejorar el posicionamiento y evitar la oscilación del robot al moverse y el segundo para que el robot pueda orientarse mejor ya que su comportamiento al tratar de orientarse a una dirección en específico este al girar a un lado se pasa de tal orientación y acto seguido gira a la dirección opuesta para tratar de llegar a la orientación objetivo provocándose lo mismo que se produjo en el primer giro, esto llega a crear una oscilación que no termina nunca en la mayoría de los casos.

4.5.1 Propuesta de control de posicionamiento

Una de las razones de la oscilación del movimiento es la falta de un sistema de control que se haga cargo del movimiento es por esto que se propone un sistema de control que está representado por el diagrama de bloques de la figura 4-9.

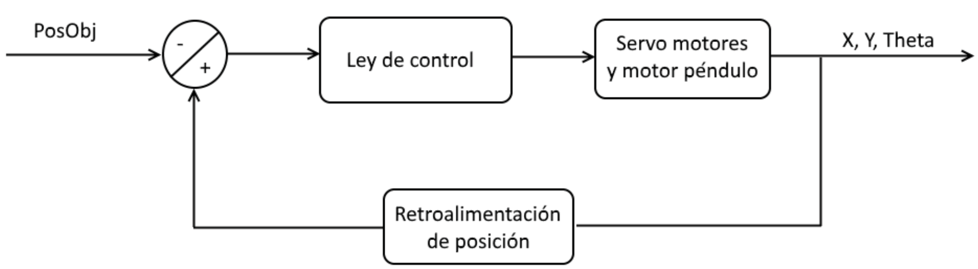


Figura 4-9: Diagrama de bloques de sistema de control para posición del robot.

Donde PosObj representa la posición objetivo a que se quiere llegar y X, Y, Theta representan la posición y la orientación de robot.

En la figura 4-10 se muestran las variables involucradas en las ecuaciones propuestas para el sistema de control.

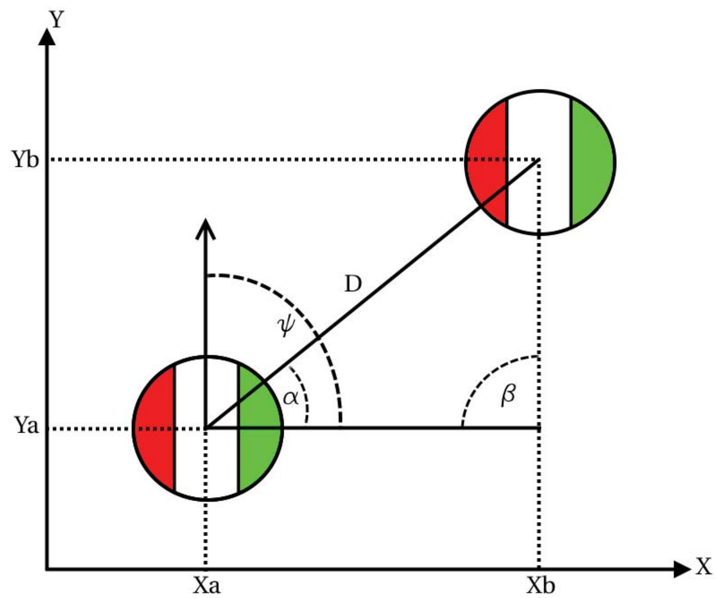


Figura 4-10: Variable de control de posición.

Con las variables de la figura 4-10 y también la ayuda del paper [28] se puede obtener las siguientes ecuaciones. Lo que se intenta hacer es disminuir el error de orientación es decir la diferencia de ψ y α sean cero o cercano a ese valor.

$$D = \sqrt{((Yb - Ya)^2 + (Xb - Xa)^2)} \tag{4-3}$$

$$\alpha = \tan^{-1}\left(\frac{Yb - Ya}{Xb - Xa}\right) \tag{4-4}$$

Estas ecuaciones serán utilizadas en el bloque punto suma del diagrama de bloques de la figura 4-9 al usar como referencia el punto PosObj y la posición actual del robot (Xa,Ya). La salida del bloque punto suma se procesara en el bloque ley de control para que se pueda tomar una decisión y se pueda también calcular la velocidad lineal (v) del robot al llegar al punto objetivo y la velocidad angular (w) del movimiento de giro al accionar el péndulo rotatorio como se muestran en las ecuaciones 4-5 y 4-6.

$$v = \begin{cases} v_{\max} & si\ |D|>K \\ D\left(\frac{v_{max}}{k}\right) & si\ |D| < k \end{cases} \tag{4-5}$$

$$w = w_{max}\sin(\alpha - \psi) \tag{4-6}$$

En donde $v_{\max}$ es la velocidad lineal máxima, k es el radio de un área de acoplamiento que está en la cerca del punto objetivo y w_{max} es la velocidad angular máxima del robot.

Discusión y conclusiones

El robot que se construyó en este proyecto difiere de la norma en esta universidad por dos razones, la primera trata que es un tipo de robot nuevo en la universidad, ya que este tipo de robot nunca se había construido o trabajado en la escuela de ingeniería, y la otra razón es que generalmente se opta por diseños tradicionales como un robot de 4 ruedas.

Si bien los mecanismos de los robots esféricos móviles son variados, muy pocos de estos son económicamente viables, ya que muchos requieren de un diseño preciso y materiales costosos para su buen funcionamiento y control. Es por esto que las empresas mayormente se enfocan en uno o dos tipos de mecanismos y los modifican ligeramente para sus fines.

Las razones de porque se eligió un mecanismo de movimiento por sobre los otros es porque: de los 7 mecanismos de movimientos del robot esférico, 4 son difíciles de construir, estos son, Single ball, cuerpo deforme, multiple-mass-shifting y mejora notable, con esto se deja como alternativas 3 mecanismos estos son, péndulo conducido, unidad de accionamiento interno y esfera de hámster, luego, de esos tres se puede descartar la esfera de hámster ya que por razones de que su diseño tiene graves fallas que afectan el direccionamiento y en consecuencia el trayecto que realizan, luego se tiene las dos opciones restante, se decidió elegir el mecanismo de péndulo conducido ya que este tiene un mejor mecanismo de direccionamiento, además de que en general es más estable en su trayecto y hay más información de este tipo ya que es uno de lo que más se han investigado en el ámbito académico.

En lo que respecta a las formulas obtenidas, a excepción de la fórmulas del péndulo, no se tuvo mucho problema obteniéndolas ya que estas fórmulas se pudieron encontrar en la mayoría de los papers que se investigaron, no así las formulas del péndulo ya que la configuración del péndulo en el diseño es muy distinto a lo visto en los papers, por lo que se tuvo que hacer un intento en obtener tal formula de manera deductiva.

En el ámbito de las simulaciones en general se realizaron sin problema, las simulaciones dieron entre lo que se esperaba, en la simulación de trayectoria recta se puede decir que a pesar de que al principio del trayecto el robot se desvió, después de esa desviación inicial el trayecto es recto, una posible causa de tal comportamiento puede ser como se pudo haber posicionado en la simulación el modelo. En la simulación de la trayectoria en Zigzag, si bien el trayecto descrito en la simulación es un Zigzagueo, se puede observar que el efecto del péndulo pierde un poco de eficacia en hacer los giros en el robot, pero eso podría deberse a algún parámetro en la simulación que podría no estar bien configurado. En el trayecto subiendo plano inclinados, las simulaciones dieron como se esperaban, es decir, según lo investigado en otros papers el máximo por el cual el robot podría desplazarse seria de 30 grados con respecto al suelo, lo cual en la simulación se puede observar que se cumple tal información.

En la parte de diseño del robot se tuvo que pensar en la simetría y en los torques que podrían afectar a la movilidad del robot. La base interna del robot se diseñó para que sea robusta con el propósito de soportar el peso del péndulo y mantener los elementos internos del robo de forma fija. Con respecto a los servomotores se eligieron estos por razones de que puede ser que más adelante se podría colocar algún aditamento que podría afectar los pesos del robot provocando que se requiera un torque mayor. El péndulo de robot se pensó tratando de imitar el robot que se observó en el [9], en este el péndulo tenia colocado pesos de pesca y se puede observar que no hay problemas con el direccionamiento de este robot, es por esto que se pensó en hacer lo mismo para el robot de este proyecto.

En el ámbito de programación del robot, este se simplifico bastante al usar Blynk ya que este tiene una aplicación para celular bastante intuitiva y además la documentación de esta aplicación se encuentra bastante organizada y con diferentes ejemplos entorno a arduino. Una dificultad que se puede mencionar es la de no poder obtener valores coherente con respecto al sensor MPU6050, esto se podría mejorar utilizando otra forma de obtener los valores buscados, pero por el momento se dejó esto de lado por razones que se utilizó la cámara de la plataforma para conocer la orientación del robot además de otros datos como la posición del robot y la posición del objetivo.

Con respecto a las pruebas realizadas, en la prueba de trayecto recto (prueba 1) se puede observar que hay una desigualdad en la superficie de la carcasa del robot lo que lleva a que este tenga una oscilación en su movimiento, se detectó que al momento de detenerse el robot presento oscilaciones. Esto se podría corregir con una nueva carcasa que no tenga imperfecciones e implementar un control de movimiento para evitar estas oscilaciones.

En la prueba con respecto a la maniobrabilidad del robot (prueba 2) se puede observar que el robot cumplió con la trayectoria de zigzag, pero su movimiento en esta trayectoria tuvo gran cantidad de oscilación porque el control que se espera implementar baje estas oscilaciones o las elimine, cabe mencionar que si se acciona el péndulo por mucho tiempo este deja de ejercer la fuerza que permite el cambio de dirección del robot y además, si se deja de accionar el péndulo cuando alcanza este comportamiento, el robot girara en la dirección opuesta al movimiento que se pretendía que girase.

En la prueba 3 en donde el robot pasa por encima de un objeto, se puede observar que la formula implementada entrega datos exactos y confiables. Según la formula al disminuir el peso del robot, este puede subir mayores alturas si se disminuye el peso de este, dado esto se podría disminuir el peso de la carcasa para que el robot pueda subir por objetos más altos.

Con respecto a la conexión del robot con la plataforma. En lo que más se tuvo problema, además de la edición del código para controlar al robot, fue en crear un servidor local para el robot esférico que no estuviera conectado fuera de la red de la universidad de forma que esta permita el control de robot. Luego de que se creó el servidor fue fácil crear una cuenta en la aplicación Blynk y utilizar esta como si estuviera conectada al servidor externo de Blynk.

El cambio de código en la plataforma se realizó gracias a referencias en internet y otras fuentes. Un aspecto del que se debe notar es que los comandos para controlar el robot se vienen dados en direcciones URL lo cual facilita en gran medida para la codificación futura del software “Ejs” de tal forma que este pueda controlar el robot.

Una observación valida fue con respecto al rastreo del robot mediante la cámara de la plataforma en este rastreo hubo ocasiones en que la diferencias de iluminación en las diferentes zonas de la plataforma causaban errores de direccionamiento y posición del robot, las causas pueden deberse a que el proceso “Adaptative Backgroung Subtraction Color” y el proceso “Red-Green Marker Detection” no toman en cuenta el cambio de iluminación del ambiente que ocurre de forma natural en lugares en que la luz del exterior puede entrar.

También se pudo explicar cómo funciona y como está compuesto el sistema que se compone del robot esférico, el servidor local Blynk y la plataforma para la prueba de robots, tal explicación sirve para poder entender mejor como funciona todo el sistema, para así sacar como poder modificar algún elemento del sistema en caso de tener otra idea de cómo controlar el robot esférico.

Además se da una pequeña explicación de las modificaciones que se hicieron al programa Swistrack y al robot para poder rastrear mejor a este último, tales modificaciones fueron, cambios en los procesos de software y cambio de marcador rojo y verde por una tonalidad más clara, todo esto llevo a que el rastreo del robot fuese menos interrumpido y por ende más fluido su recorrido.

Con respecto al código del programación de robot no está de más decir el código puede ser mejorado o modificado con respecto a lo que uno quiere, en este caso solo se usó para que siguiera unas coordenadas objetivo pero también es posible modificar la carcasa esférica para colocar una cámara y obtener una visión del trayecto que recorre, o también tenga capacidad de reconocimiento de objetos.

Bibliografía

[1] «A Spherical Robot for Planetary Surface Exploration,» Canadian Space Agency, Quebec, Canada, 2001.

[2] G. C. Schroll, «DYNAMIC MODEL OF A SPHERICAL ROBOT FROM FIRST PRINCIPLES,» Colorado State University, Fort Collins, Colorado, 2010.

[3] J.-F. Laplante, P. Masson y F. Michaud, «Analytical Longitudinal and Lateral Models of a Spherical Rolling Robot».

[4] P. Piejko, «New Atlas,» 24 octubre 2011. [En línea]. Available: https://newatlas.com/rotundus-groundbot/20259/. [Último acceso: 6 abril 2018].

[5] S. T. Ylikorpi y J. , «Ball-shaped Robots,» Itech Education and Publishing, Vienna, Austria, 2007.

[6] P. R. Chase y A. , «A Review of Active Mechanical Driving Principles of Spherical Robots,» Wayne State University, Detroit, USA, 2012.

[7] A. R. Justus, «Degree of Achievability of Omnidirectional Motion in Various Mobile Robot Designs: A Review,» International Journal of Robotics and Automation (IJRA), 2016.

[8] admin, «Sistemas holonómicos,» 16 julio 2010. [En línea]. Available: http://blog.electricbricks.com/2010/07/sistemas-holonomicos/.

[9] RCnerd, «Youtube-DIY BB8 droid final test before paint,» 15 junio 2015. [En línea]. Available: https://www.youtube.com/watch?v=iQ_PKRjQpDY&t=1s. [Último acceso: 19 junio 2018].

[10] M. Nagai, «Control System for a Spherical Robot,» Lulea University of Technology, Kiruna, 2008.

[11] «Torque,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/P%C3%A9ndulo_de_torsi%C3%B3n.

[12] Autodesk, «Autodesk Fusion 360,» [En línea]. Available: https://www.autodesk.com/products/fusion-360/overview. [Último acceso: 27 mayo 2018].

[13] c. robotics, «V-Rep tutorial - Building a clean model tutorial,» coppelia robotics, [En línea]. Available: http://www.coppeliarobotics.com/helpFiles/en/buildingAModelTutorial.htm. [Último acceso: 14 mayo 2018].

[14] C. 3D, «Youtube - How to fix servo 180º to 360º - FULL ROTATION - Futaba s3003,» 8 noviembre 2014. [En línea]. Available: https://www.youtube.com/watch?v=sBtxttzMcoo. [Último acceso: 25 julio 2018].

[15] «Blynk,» mayo 2015. [En línea]. Available: https://www.blynk.cc/. [Último acceso: 29 junio 2018].

[16] N. amica, «NodeMCU wikipedia,» 30 Diciembre 2013. [En línea]. Available: https://en.wikipedia.org/wiki/NodeMCU. [Último acceso: 29 junio 2018].

[17] Arduino, «Arduino IDE,» [En línea]. Available: https://www.arduino.cc/en/Main/OldSoftwareReleases.

[18] Blynk, «Blynk documentos,» mayo 2015. [En línea]. Available: http://docs.blynk.cc/#intro. [Último acceso: 29 junio 2018].

[19] «Sensor acelerometro/giroscopio MPU6050,» [En línea]. Available: https://playground.arduino.cc/Main/MPU-6050. [Último acceso: 29 junio 2018].

[20] SinapTec, «ESP8266 – Obtener Inclinación con MPU6050 (GY-521),» 15 octubre 2017. [En línea]. Available: https://www.youtube.com/watch?v=uN8SYfGwYVw. [Último acceso: 29 junio 2018].

[21] Wikipedia, «Application programming interface,» 28 Septiembre 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Application_programming_interface. [Último acceso: 30 Septiembre 2018].

[22] Blynk, «Blynk HTTP RESTful API,» [En línea]. Available: https://blynkapi.docs.apiary.io/#reference/0/get-pin-value. [Último acceso: 10 octubre 2018].

[23] Raspberry, «Raspberry pi,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 30 Septiembre 2018].

[24] R. pi, «Raspberry pi 2,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Último acceso: 30 Septiembre 2018].

[25] swistrack, «swistrack,» [En línea]. Available: https://en.wikipedia.org/wiki/SwisTrack. [Último acceso: 30 Septiembre 2018].

[26] Swistrack, «Swistrack Blob Detection Red and Green,» [En línea]. Available: https://en.wikibooks.org/wiki/SwisTrack/Components/BlobDetectionRedGreen. [Último acceso: 20 Noviembre 2018].

[27] F. Esquembre, «Easy Java Simulations,» [En línea]. Available: http://fem.um.es/Ejs/. [Último acceso: 22 Noviembre 2018].

[28] «A Khepera IV library for robotic control,» IFAC PapersOnLine, Valparaiso, 2017.

[29] C. Robotics, «Coppelia Robotics,» Coppelia Robotics, [En línea]. Available: http://www.coppeliarobotics.com/. [Último acceso: 26 mayo 2018].

[30] «Wikipedia - Control moment gyroscope (CMG),» [En línea]. Available: https://en.wikipedia.org/wiki/Control_moment_gyroscope. [Último acceso: 26 mayo 2018].

A Apéndice

Robot Esférico Móvil:

Es un tipo de robot que está cubierto por una carcasa esférica, en general, que a su vez también usa para desplazarse, esta carcasa protege a los circuitos del robot del medio ambiente. La forma en que se mueve de un lugar a otro, en general, es variando su centro de masa de tal manera que se pueda impulsar hacia una dirección deseada.

Holonómico:

Es un término que se usa con respecto a la movilidad de un robot. Cuando un robot es holonómico este no tiene que reposicionarse para cambiar de dirección. Un ejemplo de esto son los drones, estos pueden realizar acrobacias en el aire sin reposicionarse hacia la dirección de su movimiento.

Módulo Conversor Wifi Serial TTL ESP8266

El Módulo WiFi Serial ESP8266 ofrece una solución completa y muy económica para conexión de sistemas electrónicos a redes inalámbricas, permitiendo al diseñador delegar todas las funciones relacionadas con WiFi y TCP/IP del procesador que ejecuta la aplicación principal. Este módulo es compatible con la tarjeta de programación Arduino.

Módulo L298N Puente H Controlador de Motores

Este módulo basado en el circuito integrado L298N permite controlar hasta dos motores de corriente continua DC o un motor paso a paso bipolar de hasta 2 Amperes. Este módulo es compatible con tarjeta de programación Arduino.

Arduino Nano

Tarjeta de programación open-source, es una tarjeta pequeña, completa y compatible con Protoboard, tiene las mismas funcionalidades de la tarjeta Arduino Uno.

Software De Modelación V-Rep [29]

Es uno de los software más populares en lo que se refiere a la simulación de robots. El simulador de robots Vrep, con un entorno de desarrollo integrado, está basado en una arquitectura de control distribuido: cada objeto/modelo puede ser individualmente controlado a través de un script incrustado, un complemento, un nodo ROS o BlueZero, cliente API remoto o una solución personalizada. Todo esto hace a V-Rep una aplicación muy versátil para diferentes tipos de robots. El programa trabaja con lenguajes de programaciones como: C/C++, Python, Java, Lua, Matlab o Octave.

V-Rep es usado para el desarrollo algorítmico rápido, simulación de automatización de fábricas, rápido prototipado y verificación, aprendizaje relacionado con la robótica, monitoreo remoto, etc.

CMG (control moment gyroscope) [30]

Es un mecanismo que consiste en rotores giratorios y uno o más cardanes motorizados, este permite controlar el momento de inercia usando los cardanes para girar los rotores y así inclinar el momento angular de los rotores. Cuando el rotor se inclina el momento angular causa que el torque del giroscopio cambie y a su vez rote el sistema al cual está instalado este mecanismo.

Encoder

Es un dispositivo que permite saber la posición, dirección y velocidad de algún eje giratorio. Los encoders convierten el movimiento en una señal eléctrica que puede ser leída por un algún tipo de dispositivo de control en sistema de control de movimiento, como por ejemplo un contador o PLC. Los encoders usan diferentes tipos de tecnologías para crear la señal, como por ejemplo, mecánica, magnética, resistiva, y óptica, en donde la óptica es la más usada, en esta el encoder crea un feedback para basado en la interrupción de luz para crea la señal.

Autodesk Fusion 360 [12]

Es uno de los software más populares en lo que respecta a la creación de modelos en 3D, es ampliamente utilizado para crear piezas y modelos para posteriormente ser imprimidas en un impresora 3D, su interfaz es fácil de aprender y además hay varios tutoriales en internet que permiten resolver problemas de diseño para una cantidad diversas de piezas.

Código de trayecto en Zigzag

Tabla apéndice A-1: Código que se usó en el robot esférico.

```
#define BLYNK_PRINT Serial
//teleco----5i5telecoa2017
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SimpleTimer.h>
#include <Wire.h>

// You should get Auth Token in the Blynk App.
```

```
// Go to the Project Settings (nut icon).
char auth[] = "b74c85827186462ca835ad5e61919a74";
BlynkTimer timer;
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "GChacon";//GChacon
char pass[] = "do051609";//do051609
//---------------------------------------------------------------
#include <Wire.h>

//Direccion I2C de la IMU
#define MPU 0x68

//Ratios de conversion
#define A_R 16384.0 // 32768/2
#define G_R 131.0 // 32768/250

//Conversion de radianes a grados 180/PI
#define RAD_A_DEG = 57.295779

//MPU-6050 da los valores en enteros de 16 bits
//Valores RAW
int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;

//Angulos
float Acc[2];
float Gy[3];
float Angle[3];
String valores;
String valores1;
String valores2;
String valores3;
long tiempo_prev;
float dt;

//------------------------------------------------------------------
//SimpleTimer timer;

void setup()
{
Wire.begin(4,5); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
Wire.beginTransmission(MPU);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
  // Debug console
  Serial.begin(9600);
timer.setInterval(500L, sendUptime);
  Blynk.begin(auth, ssid, pass);
}
void sendUptime()                               //Function Central Out
{
   Wire.beginTransmission(MPU);
   Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true);   //A partir del 0x3B, se piden 6 registros
   AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   AcY=Wire.read()<<8|Wire.read();
   AcZ=Wire.read()<<8|Wire.read();

   //A partir de los valores del acelerometro, se calculan los angulos Y, X
   //respectivamente, con la formula de la tangente.
   Acc[1]        =        atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2)        +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
   Acc[0]         =         atan((AcY/A_R)/sqrt(pow((AcX/A_R),2)         +
pow((AcZ/A_R),2)))*RAD_TO_DEG;

   //Leer los valores del Giroscopio
   Wire.beginTransmission(MPU);
   Wire.write(0x43);
   Wire.endTransmission(false);
```

```
   Wire.requestFrom(MPU,6,true);   //A partir del 0x43, se piden 6 registros
   GyX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   GyY=Wire.read()<<8|Wire.read();
   GyZ=Wire.read()<<8|Wire.read();

   //Calculo del angulo del Giroscopio
   Gy[0] = GyX/G_R;
   Gy[1] = GyY/G_R;
   Gy[2] = GyZ/G_R;

   dt = (millis() - tiempo_prev) / 1000.0;
   tiempo_prev = millis();

   //Aplicar el Filtro Complementario
   Angle[0] = 0.98 *(Angle[0]+Gy[0]*dt) + 0.02*Acc[0];
   Angle[1] = 0.98 *(Angle[1]+Gy[1]*dt) + 0.02*Acc[1];

   //Integración respecto del tiempo paras calcular el YAW
   Angle[2] = Angle[2]+Gy[2]*dt;

   //Mostrar los valores por consola
   valores = "90, " +String(Angle[0]) + "," + String(Angle[1]) + "," +
String(Angle[2]) + ", -90";
   //valores1 = String(Angle[0]);
   //valores2 = String(Angle[1]);
   //valores3 = +String(Angle[2]);
   Serial.println(valores);
   Blynk.virtualWrite(V2, valores);
}
void loop()
{
  Blynk.run();
  timer.run();
}
```

A.1 Pasos a seguir para la creación de un servidor local de Blynk en raspberry pi

Para poder crear un servidor local de Blynk se requirió seguir la guía que se muestra en los documentos de la aplicación Blynk [18] y además ayuda externa en forma de video ya que algunos pasos de la instalación no se explican bien en los documentos de Blynk. Los pasos a seguir para la instalar el servidor local Blynk son los siguientes:

1. Conectar los periféricos (mouse, teclado, memoria SD y pantalla) a la tarjeta Raspberry pi 2 junto también a una conexión Ethernet
2. Instalar sistema operativo Raspbian (Figura_apéndice 1) en la tarjeta Raspberry pi 2

Figura_apéndice 1: Sistema operativo Raspbian.

3. Instalar java 8 jdk utilizando el siguiente comando en una ventana terminal

```
sudo apt-get install oracle-java8-jdk
```

4. Asegúrese que se instaló correctamente java 8 usando el siguiendo el comando en el terminal

```
java -version
```

❖ Si la instalación fue exitosa el terminal devolverá como respuesta lo siguiente

```
Output: java version "1.8"
```

5. luego se descarga el archivo .jar para la instalación del servidor local utilizando el siguiente comando.

```
wget "https://github.com/blynkkk/blynk-
server/releases/download/v0.39.10/server-0.39.10-java8.jar"
```

6. En la carpeta home cree una carpeta llamada Blynk y coloque el archivo descargado en esta carpeta.
7. En el terminal coloque el comando “cd Blynk/” sin las comillas para poder interactuar con archivos en esa carpeta.
8. Luego coloque el siguiente comando en el terminal para poder instalar el servidor local Blynk en la raspberry pi.

```
java -jar server-0.39.10-java8.jar -dataFolder /home/pi/Blynk
```

9. Después de esto en el terminal debería aparecer lo siguiente

```
Blynk Server successfully started.
All server output is stored in current folder in 'logs/blynk.log' file.
```

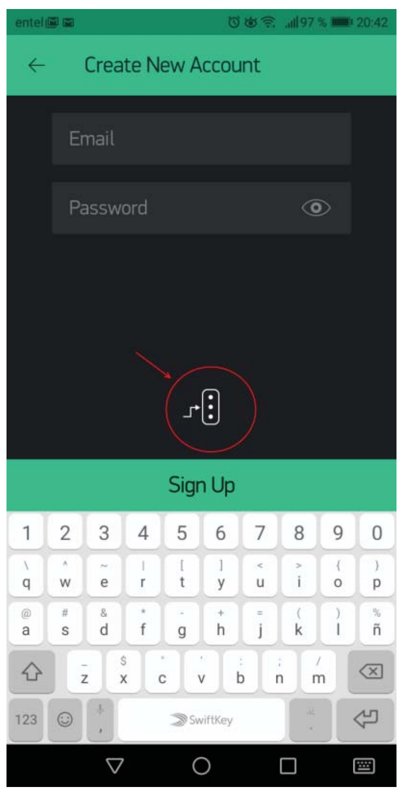
10. En el teclado presione Ctrl + c para detener el servidor y en la carpeta Blynk que anteriormente creo y creo un documento de texto y cambie el nombre del documento a “mail.properties” al hacer esto el archivo cambiara de formato a .properties.
11. Abra el archivo de texto “mail.properties” y escriba lo siguiente

```
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.host=smtp.gmail.com
mail.smtp.port=587
mail.smtp.username=YOUR_EMAIL_HERE
mail.smtp.password=YOUR_EMAIL_PASS_HERE
```

12. Cambie los campos en donde diga YOUR_EMAIL_HERE y YOUR_EMAIL_PASS_HERE por el email que quiera con la contraseña correspondiente.
13. Después guarde los cambios en el documento y en el terminal coloque lo siguiente para saber la dirección IP que usa el servidor local.

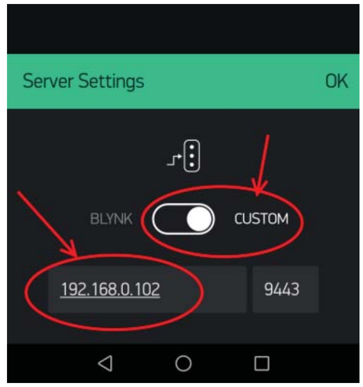
```
Hostname -I
```

14. Luego de esto le aparecerá la dirección IP que debe utilizar en la aplicación Blynk
15. En la aplicación Blynk cree una nueva cuenta pero cambiando a servidor local apretando el símbolo indicado en la figura_apéndice 2



Figura_apéndice 2: Imagen de la aplicación para crear una nueva cuenta

16. Cambie a la opción CUSTOM y coloque la IP obtenida anteriormente, colocando también el puerto 9443 (Figura_apéndice 3).



Figura_apéndice 3: Opciones para cambiar a servidor local y IP

17. Luego ingrese a la cuenta que acaba de crear cambiando a servidor local y ya puede usar la aplicación de forma normal.

A.2 Código de robot

Tabla apéndice A-2: Código que se usó en el robot esférico.

```
#define BLYNK_PRINT Serial
#include <math.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char          auth[]          =         "31de7b2c7ceb43fa98fb056c5d1b5071";
//31de7b2c7ceb43fa98fb056c5d1b5071

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "labr";//labr
char pass[] = "";    //
 float angulo_Rob;  // angulo direccion de robot
float angulo_objetivo; //angulo direccion del objetivo
float angulo_objetivo2;
float Xrob;  // posicion x del robot
float Yrob; //  posicion y del robot
float Xcomp =0; // posicion x objetivo
float Ycomp =0; // posicion y objetivo
int velocidad = 60; // velocidad del robot  60
int velocidad_Giro_izq = 1021;
int velocidad_Giro_der = 40;
int difAng=5;

BLYNK_WRITE(V1){
  angulo_Rob = param.asInt();
}
BLYNK_WRITE(V2){
  Xrob = param.asInt();
}
BLYNK_WRITE(V3){
  Yrob = param.asInt();
}
BLYNK_WRITE(V4){
  Xcomp = param.asInt();
}
```

```
BLYNK_WRITE(V5){
  Ycomp = param.asInt();
  movimiento();
}
/*BLYNK_READ(V6){
  //Ycomp = param.asInt();
  Blynk.virtualWrite(V6, String(angulo_objetivo2));
}*/
void setup()
{
  // Debug console
  Serial.begin(9600);

  //Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  Blynk.begin(auth, ssid, pass, IPAddress(192,168,0,102), 8080);

}

void loop()
{
  Blynk.run();

  }

void movimiento(){
  float x_diferencia = Xcomp - Xrob;
  float y_diferencia = Ycomp - Yrob;


  if (Xcomp > Xrob && Yrob > Ycomp){ //cuadrante 4
    angulo_objetivo =atan2(y_diferencia,x_diferencia)*180/3.141592;
    angulo_objetivo2 = angulo_objetivo+360;
    Serial.println(angulo_objetivo2);
    Serial.println("cuadrante 4");
     if(angulo_Rob > angulo_objetivo2+difAng ||angulo_Rob <
angulo_objetivo2-difAng){
            if((angulo_objetivo2-180) < angulo_Rob && (angulo_objetivo2-
difAng) > angulo_Rob){  //(1)
              analogWrite(D8,velocidad_Giro_der);
              analogWrite(D5,0);
            }else if((angulo_objetivo2-180) >= angulo_Rob ||
(angulo_objetivo2+difAng) < angulo_Rob){  //(3)  (2)
              analogWrite(D8,velocidad_Giro_izq);
              analogWrite(D5,0);
            }
     }else if (angulo_Rob < angulo_objetivo2+difAng && angulo_Rob >
angulo_objetivo2-difAng){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }

  if(Xcomp < Xrob && Yrob > Ycomp){ //cuadrante 3
    angulo_objetivo = atan2(y_diferencia,x_diferencia)*180/3.141592+360;
     angulo_objetivo2 = angulo_objetivo-180;
     Serial.println(angulo_objetivo);
     Serial.println("cuadrante 3");
         if(angulo_Rob > angulo_objetivo+difAng || angulo_Rob <
angulo_objetivo-difAng){
          if ((angulo_objetivo-difAng) > angulo_Rob && angulo_objetivo2 <
angulo_Rob){  //(1)
            analogWrite(D8,velocidad_Giro_der);
            analogWrite(D5,0);
          }else if(((angulo_objetivo2) >= angulo_Rob) ||
((angulo_objetivo+difAng) < angulo_Rob)) {  //(3)  (2)
            analogWrite(D8,velocidad_Giro_izq);
            analogWrite(D5,0);
          }
     }else if (angulo_Rob < (angulo_objetivo+difAng) && angulo_Rob >
(angulo_objetivo-difAng)){
```

```
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }

  if(Xcomp < Xrob && Yrob < Ycomp){ //cuadrante 1
    angulo_objetivo = atan2(y_diferencia,x_diferencia)*180/3.141592;
     angulo_objetivo2 = angulo_objetivo+180;
    Serial.println(angulo_objetivo2);
    Serial.println("cuadrante 1");
        if(angulo_Rob    >    angulo_objetivo+difAng    ||angulo_Rob    <
angulo_objetivo-difAng){
            if((angulo_objetivo+difAng) < angulo_Rob && angulo_objetivo2 >
angulo_Rob){  //(1)
              analogWrite(D8,velocidad_Giro_izq);
              analogWrite(D5,0);
            }else if((angulo_objetivo2) <= angulo_Rob || (angulo_objetivo-
difAng) > angulo_Rob){  //(2)  (3)
              analogWrite(D8,velocidad_Giro_der);
              analogWrite(D5,0);
            }
     }else  if  (angulo_Rob  <  (angulo_objetivo+difAng)  &&  angulo_Rob  >
(angulo_objetivo-difAng)){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }

  if(Xcomp > Xrob && Yrob < Ycomp){//cuadrante 2

    angulo_objetivo = atan2(y_diferencia,x_diferencia)*180/3.141592;
     angulo_objetivo2 = angulo_objetivo+180;


    Serial.println(angulo_objetivo2);
    Serial.println("cuadrante 2");
     if(angulo_Rob > angulo_objetivo+difAng ||angulo_Rob < angulo_objetivo-
difAng){
          if( (angulo_objetivo+difAng) < angulo_Rob && angulo_objetivo2 >
angulo_Rob){  //(1)
            analogWrite(D8,velocidad_Giro_izq);
             analogWrite(D5,0);
          }else  if((angulo_objetivo2)  <=  angulo_Rob  ||  (angulo_objetivo-
difAng) > angulo_Rob){//(2) (3)
            analogWrite(D8,velocidad_Giro_der);
             analogWrite(D5,0);
          }
     }else  if  (angulo_Rob  <  (angulo_objetivo+difAng)  &&  angulo_Rob  >
(angulo_objetivo-difAng)){
      analogWrite(D5,velocidad);
       analogWrite(D8,0);
     }
  }
//-----------------------------------------------------------------
  if (Xcomp == Xrob && Ycomp >= Yrob){  // cua 1, 2
    angulo_objetivo2 = 90;
    Serial.println(angulo_objetivo2);
     if(angulo_Rob != angulo_objetivo2){
          if((angulo_objetivo2-180)<angulo_Rob){
            analogWrite(D8,velocidad_Giro_izq);
            analogWrite(D5,0);
          }else  if((angulo_objetivo2<=angulo_Rob)  &&  (angulo_objetivo2-
180)>=angulo_Rob ){
            analogWrite(D8,velocidad_Giro_der);
            analogWrite(D5,0);
          }
     }else if (angulo_Rob == angulo_objetivo2){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }
```

```

  if(Xcomp == Xrob && Ycomp < Yrob){     // cua 3, 4
    angulo_objetivo2 = 270;
     Serial.println(angulo_objetivo2);
         if(angulo_Rob != angulo_objetivo2){
          if ((angulo_objetivo2+180) > angulo_Rob){
            analogWrite(D8,velocidad_Giro_der);
            analogWrite(D5,0);
          }else if((angulo_objetivo2+180) <= angulo_Rob && angulo_objetivo2
>= angulo_Rob){
            analogWrite(D8,velocidad_Giro_izq);
            analogWrite(D5,0);
          }
     }else if (angulo_Rob == angulo_objetivo2){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }
if (Xcomp >= Xrob && Ycomp == Yrob){  //cua 2, 3
    angulo_objetivo2 = 0;
    Serial.println(angulo_objetivo2);
     if(angulo_Rob != angulo_objetivo2){
          if((angulo_objetivo2-180)<angulo_Rob){
            analogWrite(D8,velocidad_Giro_izq);
            analogWrite(D5,0);
          }else  if((angulo_objetivo2<=angulo_Rob)  &&  (angulo_objetivo2-
180)>=angulo_Rob ){
            analogWrite(D8,velocidad_Giro_der);
            analogWrite(D5,0);
          }
     }else if (angulo_Rob == angulo_objetivo2){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }

  if(Xcomp < Xrob && Ycomp == Yrob){  // cua 3, 4
        angulo_objetivo2 = 180;
    Serial.println(angulo_objetivo2);
     if(angulo_Rob != angulo_objetivo2){
          if((angulo_objetivo2-180)<angulo_Rob){
            analogWrite(D8,velocidad_Giro_izq);
            analogWrite(D5,0);
          }else  if((angulo_objetivo2<=angulo_Rob)  &&  (angulo_objetivo2-
180)>=angulo_Rob ){
            analogWrite(D8,velocidad_Giro_der);
            analogWrite(D5,0);
          }
     }else if (angulo_Rob == angulo_objetivo2){
      analogWrite(D5,velocidad);
      analogWrite(D8,0);
     }
  }
  if(Xrob<(Xcomp+difObj)&&   Xrob>(Xcomp-difObj)&&   Yrob<(Ycomp+difObj)&&
Yrob>(Ycomp-difObj)){
      analogWrite(D8,0);
      analogWrite(D5,0);
 }
}

```

A.3 Código de programa Ejs

Tabla apéndice A-3: Código que se usó en el robot esférico.

```
// confirma que el robot este conectado al servidor
if(condicion_envio == 0){

                        try {
```

```
                              URL url = new
URL("http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/isHardwareConne
cted");
                          try {
                                URLConnection yc = url.openConnection();
                                BufferedReader in = new BufferedReader(new
InputStreamReader(url.openStream()));
                                String inputLine;
                                while ((inputLine = in.readLine()) != null)
                                System.out.println(inputLine);
                                in.close();
                                }catch (IOException e) {e.printStackTrace();}
                            }catch (MalformedURLException e1)
{e1.printStackTrace();}
                            condicion_envio=1;
                          }
///------------------------------------------------------------------------------
---------------
// se obtienen los datos relevantes del swistrack como las coordenadas del robot
y el objetivo y ademas la direccion del robot


_setDelay(80);

try {
     int recvSize;
     int totalRecvSize = 0;


   if(dale==1){


     int cant = cis.available();
     byte[] buffer = new byte[1024];
     recvSize = cis.read(buffer, totalRecvSize, 1024-totalRecvSize);
     //System.out.println("Paquete recibido: "+recvSize);

     String s1 = new String(buffer);
     //System.out.println(s1);

     String[] lineas = s1.split("\n");
     int size = lineas.length;

     for (int b = size-1; b >= 0; b--){
            if (lineas[b].startsWith("$FRAMENUMBER,")){
               beginFrame = b;
               numberOfRobots=0;
               break;
            }
      }


      for (int i = beginFrame+1; i<size-1; i++) {



              if(lineas[i].startsWith("$PARTICLE,")){
                    //System.out.println("PRE-$PARTICLE");
                    processSwiss(lineas[i]);
                    //System.out.println("POST-$PARTICLE");
              }

              if(lineas[i].startsWith("$STEP_STOP") && numberOfRobots!=0){

                  for(int c=1; c<=numberOfRobots; c++){
                      System.out.print("Robot: "+robotArray[c][0]);
                      System.out.print("  x= "+robotArray[c][1]);
                      System.out.print("  y= "+robotArray[c][2]);
                      //System.out.println("th= "+robotArray[c][3]);

                      guardarDatosPrev=1;
```

```
                      xTest=robotArray[c][1];
                      yTest=robotArray[c][2];
                      thiTest=robotArray[c][3];
                      thiTest=thiTest/100;
                      if(thiTest<0){
                        thiTest=2*3.141592654+thiTest;

                        }
                      angulo =thiTest*180/3.141592654-90;
                      //int angulo = thiTest;
                      if(angulo<0){
                        angulo = angulo+360;

                        }
                      System.out.println("  th= "+thiTest);
                      nRobot=robotArray[c][0];

                      time_end=System.currentTimeMillis();
                      recordingData(c,numberOfRobots,(time_end -
time_start),nRobot);


                      byte[] x_buff = intToByteArray(robotArray[c][1]);
                      byte[] y_buff = intToByteArray(robotArray[c][2]);
                      byte[] th_buff = intToByteArray(robotArray[c][3]);

                      byte[] xp = intToByteArray(robotArray[c][4]);
                      byte[] yp = intToByteArray(robotArray[c][5]);
                      byte[] errorCop=intToByteArray(0);
                      byte[] kCop=intToByteArray(0);

                      if (nRobot==1){
                        xp = intToByteArray(0); //byte[] xp =
intToByteArray(robotArray[c][4]);
                        yp = intToByteArray(0); //byte[] yp =
intToByteArray(robotArray[c][5])
                        pXR1=robotArray[c][1];
                        pYR1=robotArray[c][2];
                        tR1=robotArray[c][3];
                        //errorCop=intToByteArray(0);

errorCop=intToByteArray((((int)errorR2)+((int)errorR3)));
                        kCop=intToByteArray(10);
                      }
                      if (nRobot==2){
                        xpAux=-pXR1-
(int)(30*Math.cos(((float)tR1)/100+(3*3.14/4)));

ypAux=pYR1+(int)(30*Math.sin(((float)tR1)/100+(3*3.14/4)));
                        xp = intToByteArray(xpAux); //byte[] xp =
intToByteArray(robotArray[c][4]);
                        yp = intToByteArray(ypAux); //byte[] yp =
intToByteArray(robotArray[c][5])

errorR2=Math.sqrt(Math.pow(xpAux+robotArray[c][1],2)+Math.pow(ypAux-
robotArray[c][2],2));
                        //System.out.println("errorR2: "+errorR2);
                      }
                      if (nRobot==3){
                        xpAux=-pXR1-(int)(30*Math.cos(((float)tR1)/100-
(3*3.14/4)));
                        ypAux=pYR1+(int)(30*Math.sin(((float)tR1)/100-
(3*3.14/4)));
                        xp = intToByteArray(xpAux); //byte[] xp =
intToByteArray(robotArray[c][4]);
                        yp = intToByteArray(ypAux); //byte[] yp =
intToByteArray(robotArray[c][5])

errorR3=Math.sqrt(Math.pow(xpAux+robotArray[c][1],2)+Math.pow(ypAux-
robotArray[c][2],2));
                      }
```

```

                      //System.out.println("********************************");

                      byte[] message = new byte[28]; //20

                      for (int k=0; k<4; k++){
                        message[k]=x_buff[k];
                      }
                      for (int l=4; l<8; l++){
                        message[l]=y_buff[l-4];
                      }
                      for (int g=8; g<12; g++){
                        message[g]=th_buff[g-8];
                      }
                      for (int h=12; h<16; h++){
                        message[h]=xp[h-12];
                      }
                      for (int j=16; j<20; j++){
                        message[j]=yp[j-16];
                      }
                      for (int j=20; j<24; j++){
                        message[j]=errorCop[j-20];
                      }
                      for (int j=24; j<28; j++){
                        message[j]=kCop[j-24];
                      }

                     // cliente(robotArray[c][0]);
                      enviar(message,robotArray[c][0]);

                   }  // end for(int c=1; c<=numberOfRobots; c++){
               //System.out.println("Robots: "+numberOfRobots);

           }
        }
        }
           if (connected==false && guardarDatosPrev==1){

           grabarTexto();
           guardarDatosPrev=0;
           dale=0;

           }
     }
     catch(Exception e) {
        // System.out.print("No se pudo procesar el SwissTrack\n");
      }

 //-----------------------------------------------------------------------------
----------------------------------------------------------------------


 if(connected==true){

  try {
         String rOBY =Integer.toString(robotArray[1][2]);
         String rOBX =Integer.toString(robotArray[1][1]);
         System.out.println(robotArray[1][3]);
         String DR =Integer.toString(robotArray[1][3]);

      ///-------------------------------------------------------------------------
--------------------
      // se establecen las url sin modificacion

      String                                                                  Ry
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/V3?value="
+ rOBY;
      //System.out.println(Ry);
      String                                                                  Rx
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/V2?value="+
rOBX;
```

```
      String                                                          Oby
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/V5?value="+
ob_y;
      String                                                          Obx
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/V4?value="+
ob_x;
      String                                                         DirR
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/V1?value="+
angulo;
      //System.out.println(DirR);

      ///-----------------------------------------------------------------------
---------------------

      //-----------------------------------------------------------------------
---------------------
      // se establecen las direcciones url
        URL urlDirRob = new URL(DirR);// direccion que apunta el rob
        URL urlPosRoby = new URL(Ry);// coordenada y del robot
        URL urlPosRobx = new URL(Rx);// coordenada x del robot
        URL urlPosObjy = new URL(Oby);// coordenada y del objetivo
        URL urlPosObjx = new URL(Obx);// coordenada x del objetivo

      //-----------------------------------------------------------------------
---------------------

                          try {
                                URLConnection yc0 = urlDirRob.openConnection();
                                URLConnection yc1 = urlPosRoby.openConnection();
                                URLConnection yc2 = urlPosRobx.openConnection();
                                URLConnection yc3 = urlPosObjy.openConnection();
                                URLConnection yc4 = urlPosObjx.openConnection();


                                BufferedReader  inDir  =  new  BufferedReader(new
InputStreamReader(urlDirRob.openStream()));
                                String inputLineDir;

                                BufferedReader  inRobY  =  new  BufferedReader(new
InputStreamReader(urlPosRoby.openStream()));
                                String inputLineRY;

                                BufferedReader  inRobX  =  new  BufferedReader(new
InputStreamReader(urlPosRobx.openStream()));
                                String inputLineRX;

                                BufferedReader  inObjY  =  new  BufferedReader(new
InputStreamReader(urlPosObjy.openStream()));
                                String inputLineOy;

                                BufferedReader  inObjX  =  new  BufferedReader(new
InputStreamReader(urlPosObjx.openStream()));
                                String inputLineOx;





                                while ((inputLineDir = inDir.readLine()) != null
|| (inputLineRY = inRobY.readLine()) != null ||
                                (inputLineRX  =  inRobX.readLine())  !=  null  ||
(inputLineOy = inObjY.readLine()) != null ||
                                (inputLineOx = inObjX.readLine()) != null)
                                System.out.println(inputLineDir);
                                inDir.close();
                                inRobY.close();
                                inRobX.close();
                                inObjY.close();
                                inObjX.close();
                                }catch (IOException e) {e.printStackTrace();}

```

```
                             }catch          (MalformedURLException          e1)
{e1.printStackTrace();}


    }
  if(connected==false){
    try{
      String                                                       detenida_giro
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/D8?value=0"
;
      String                                                     detenida_avance
="http://192.168.0.102:8080/31de7b2c7ceb43fa98fb056c5d1b5071/update/D5?value=0"
;
      URL urlDetGiro = new URL(detenida_giro);// direccion que apunta el rob
      URL urlDetavan = new URL(detenida_avance);// coordenada y del robot

       try{
         URLConnection DetG = urlDetGiro.openConnection();
         URLConnection DetA = urlDetavan.openConnection();

         BufferedReader       inDetG       =       new       BufferedReader(new
InputStreamReader(urlDetGiro.openStream()));
         String inputLineDetG;

         BufferedReader       inDetA       =       new       BufferedReader(new
InputStreamReader(urlDetavan.openStream()));
         String inputLineDetA;

         while((inputLineDetG = inDetG.readLine()) !=null || (inputLineDetA =
inDetA.readLine()) != null)
         inDetG.close();
         inDetA.close();
         }catch (IOException w) {w.printStackTrace();}
         }catch (MalformedURLException w1) {w1.printStackTrace();}
 }
```