Universidad Tecnológica Nacional
Facultad Regional Bahía Blanca

Ingeniería en Energía Eléctrica

Trabajo Final de Grado

Robot vertical de dos ruedas para inspección de edificios

Autor

Santiago Ismael Salvatierra

Tutor

Dr. Ing. Andrés Gabriel García

Bahía Blanca | 18 de marzo de 2024

Resumen

Este proyecto final de grado de la carrera de Ingeniería en Energía Eléctrica se centra en el diseño y construcción de un robot vertical de dos ruedas. Se destacan las técnicas de control no lineal aplicadas para lograr la estabilización vertical del robot, un aspecto crítico para su funcionamiento autónomo. Además, se ha implementado un algoritmo de navegación que permite al robot desplazarse de manera eficiente en su entorno. Para complementar, se ha diseñado un sistema de carga de baterías eficiente y adaptado a las necesidades energéticas del robot, en dicho diseño se elaboró un control lineal (PID).

Palabras claves

Control no lineal, Control lineal, Robot, Monociclo, Cargador de baterías, PID, Raspberry Pi Pico W, MPU6050, Lyapunov.

Tabla de contenido

Introducción

La Ingeniería eléctrica abarca muchos rubros como pueden ser la generación, transmisión y la distribución de energía eléctrica, la optimización de recursos energéticos, los sistemas de control, entre otros.

Este proyecto aborda un problema de sistemas de control complejo que a la vez sirve de punto de referencia[1] para estudios de algoritmos futuros: un robot vertical de dos ruedas.

En este proyecto se diseña y construye un robot de dos ruedas con control vertical y procesamiento en Python, a la vez que se estudian algoritmos de control no-lineal aplicados, tanto para asegurar su estabilidad como para habilitar su navegación en el interior de un recinto. Las técnicas de control no lineal son un tema avanzado que va más allá del alcance de una carrera de grado, pero motivan al futuro profesional a investigar y adentrarse en una futura especialización en el campo del control. Esta familiarización con modelos teóricos conlleva varios desafíos, entre los cuales el autor debe enfrentar el más significativo: llevar el modelo teórico extraído de artículos de investigación a la práctica.[1]

En síntesis, este proyecto se enfoca en el desarrollo e implementación de un sistema de control para un robot vertical de dos ruedas con el propósito de explorar y aplicar los principios fundamentales de control lineal y/o no lineal para lograr un desempeño óptimo en términos de estabilidad y navegación.

Estado del Arte

Varias universidades de todo el mundo han realizado un proyecto similar abordándolo a partir de la idea de que el sistema mecánico es un péndulo invertido, pero utilizando distintas técnicas de control como puede ser LQR (Linear Quadratic Regulation), Pole-placement controller o PID controller [19]. Este tipo de robot tiene un gran potencial para resolver problemas globales del sector del transporte, entre otros posibles usos. Además, tiene una gran similitud con el famoso vehículo de transporte “Segway” [18].

Aquí se expondrán algunos de los artículos científicos vistos:

- Jayakody, D.P.V.J. & Sucharitharathna, K.P.G.C.. (2019). Control Unit for a Two-Wheel Self-Balancing Robot. Global Journal of Researches in Engineering. 7-12. 10.34257/GJREJVOL19IS1PG7.
- Nawawi, S.W., Ahmad, M.N., & Osman, J.H. (2008). Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot. International Journal of Electrical and Computer Engineering, 2, 406-412.
- Rich Chi Ooi (2003). “Balancing a Two-Wheeled Autonomous Robot”, Final Year Thesis 2003 de la Universidad de Western Australia School of Mechanical Engineering.

[1] En la jerga conocido como brenchmark.

Motivaciones

Las motivaciones de realizar un proyecto de este estilo son las siguientes:

- Este proyecto brinda la oportunidad de adentrarse en el mundo de la movilidad eléctrica. Representa una valiosa oportunidad para involucrarse en una primera experiencia significativa en proyectos de esta envergadura.
- Es una oportunidad de enfrentar desafíos técnicos y consolidar un conjunto de habilidades fundamentales en el campo de la ingeniería. El manejo y manipulación de motores eléctricos y sensores será una parte integral de este proyecto.
- Permite adquirir conocimientos en diversas áreas, incluyendo ingeniería mecánica, electrónica, control y programación.

Objetivos

El objetivo general de este proyecto es diseñar, construir y controlar un robot de dos ruedas con sensores y algoritmos de navegación. Su implementación permitirá:

- Trabajar en el diseño de prototipos en Fusion360.
- Estudiar técnicas de control no-lineal, las cuales serán aplicadas para el control del robot.
- Modelar un sistema mecánico utilizando Matlab.
- Proyectar y determinar los motores del robot.
- Presupuestar materiales para la construcción del sistema.
- Programar en Micro-Python y C++ los microcontroladores utilizados en el robot.
- Diseñar control de carga de Baterías
- Estudiar algoritmos de navegación.
- Trabajar en conjunto con estudiante del extranjero[2].

Objetivo secundario

Como objetivo secundario, una vez terminado el proyecto, el robot quedará a disposición para su uso de alumnos y docentes de la carrera de Ingeniería en Energía Eléctrica de la Facultad.

Alcances

Los alcances establecidos para alcanzar los objetivos planteados anteriormente son:

- Modelo cinemático/dinámico de robot de dos ruedas.
- Diseño de prototipo de robot.
- Diseño de cargadores de baterías de 30 V.
- Modelado final en Simulink Matlab.

[2] Estudiante de ingeniería de ciencia de la computación de la universidad de Hochschule für Technik und Wirtschaft, HTW en Berlín.

- Cálculo de especificaciones mínimas requeridas para la elección de los motores.
- Implementación del sistema de control no-lineal.
- Programa con el algoritmo de navegación interior.
- Construcción del robot.

1 Capítulo 1: Algoritmo de control vertical del robot.

En este capítulo se desarrollan los modelos necesarios en variables de estado de las diferentes técnicas de control y algoritmos, tanto sea el modelo del sistema electromecánico como las técnicas de control implementadas para el control de estabilidad como el control de navegación.

Debido a las complejidades y problemas, se desarrollarán distintos sistemas con su planteamiento desde el punto de vista teórico sobre el control.

1.1 Modelado no lineal del sistema mecánico.

El modelo matemático utilizado se basa en el trabajo denominado “Non-Lyapunov Control of a Balancing” (escrito por García *et al*., 2019) en el cual se presenta la implementación del control basado en una técnica que utiliza funciones de Lyapunov de un sistema de estabilización de un monociclo [11].

El modelo simplificado del robot se representa como un monociclo, que es una masa la cual está vinculada a la rueda mediante una barra de largo “*l*”, ver Figura 1.1.

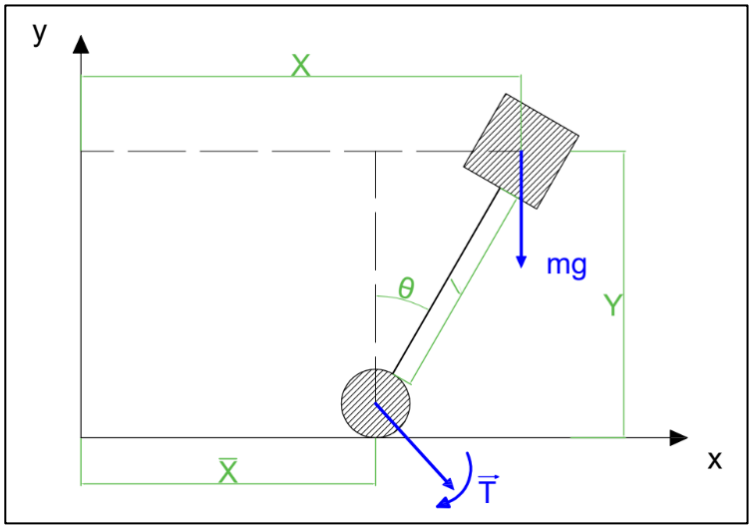


Figura 1.1.- Esquema del monociclo

El modelo matemático parte del planteo de la mecánica Lagrangiana de mi sistema.

$$L = \frac{1}{2} \cdot M \cdot \left(\dot{X}(t)^2 + \dot{Y}(t)^2\right) - M \cdot g \cdot (l - Y) \quad (1.1)$$

donde g es la gravedad, M la masa de todo el robot, l es la longitud da la barra del robot y $\{X, Y\}$ la posición del centro de masa del robot en coordenadas cartesianas.

A partir de la ecuación (1.1) se puede obtener el modelo en variables de estado (ver Sección 1 de [1]). La ecuación (1.2) es el resultado final del desarrollo antes mencionado, de allí se parte el planteo del control.

$$\begin{bmatrix} \ddot{\bar{X}}(t) \\ \ddot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \frac{l\cdot\dot{\theta}^2 - g\cdot\cos(\theta)}{\sin(\theta)} \\ \frac{-l\cdot\cos(\theta)\cdot\dot{\theta}^2 + g}{l\cdot\sin(\theta)} \end{bmatrix} + \begin{bmatrix} -\frac{l\cdot\cos(\theta)}{M\cdot l^2\cdot\sin^2(\theta)} \\ \frac{1}{M\cdot l^2\cdot\sin^2(\theta)} \end{bmatrix} \cdot u(t) \quad (1.2)$$

donde $u(t) = \tau(t)$ es el torque aplicado por el motor en la rueda.

Como conclusión de este apartado, se puede advertir en (1.2), que el sistema es **no lineal** ya que como primera observación $\cos(\theta)$ multiplica $\dot{\theta}$. Es bien sabido que la dinámica del péndulo es un problema no lineal clásico con sus consecuentes complejidades de control.

1.2 Control de estabilidad vertical.

La conclusión de la sección anterior trae consigo un análisis más complejo. Para garantizar la estabilidad del sistema, monociclo, se debe asegurar que el mismo sea asintóticamente estable, siendo la misma una condición necesaria y suficiente para garantizar la convergencia hacia la estabilidad. Dicha condición se desarrolla en el trabajo [2].

Retomando y analizando lo visto en [1]. Definiendo una nueva matriz a partir de la ecuación (1.2).

$$\begin{bmatrix} \dot{Z}_1(t) \\ \dot{Z}_2(t) \\ \dot{Z}_3(t) \\ \dot{Z}_4(t) \end{bmatrix} = \begin{bmatrix} Z_2 \\ \frac{l\cdot {Z_4}^2 - g\cdot cos(Z_3)}{sin(Z_3)} \\ Z_4 \\ \frac{-l\cdot cos(Z_3)\cdot {Z_4}^2 + g}{l\cdot sin(Z_3)} \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{l\cdot cos(Z_3)}{M\cdot l^2\cdot sin^2(Z_3)} \\ 0 \\ \frac{1}{M\cdot l^2\cdot sin^2(Z_3)} \end{bmatrix} \cdot u(t) \quad (1.3)$$

donde define la matriz $Z(t) = \left[\bar{X}, \dot{\bar{X}}, \theta, \dot{\theta}\right]'$ con ´ transpuesta. Entonces la siguiente condición debe satisfacerse para garantizar un control con estabilidad asintóticamente estable (ver desarrollo [2]):

$$f(V_1) = \alpha \cdot f(V_2), \alpha \neq 0, k\,\{V_1, V_2\} \in \delta_0$$

donde δ_0 es entorno al origen. El desarrollo tanto de la condición necesaria y suficiente y del diseño de la estabilidad asintótica se puede ver en [2] y [1] respectivamente.

La ley de control es

$$u(t) = M \cdot l^2 \cdot sin^2(Z_3) \cdot \left(-\frac{-l\cdot\cos(Z_3)\cdot {Z_4}^2 + g}{l\cdot\sin(Z_3)} + v\right), \ \{a_1, a_2\} < 0 \quad (1.4)$$

, donde $v = (a_1 \cdot Z_3 + a_2 \cdot Z_4)$ (1.5)

1.2.1 Simulación del control no lineal del monociclo.

Sobre la ley de control (1.4), se simuló en Matlab considerando M = 15 kg, l = 0.5 m y constante a_1 y a_2 son -500 y -105 respectivamente. Estos fueron los resultados (Ver Figura 1.2):

Considerando que la posición inicial es $\begin{bmatrix} x_0 \\ \dot{x}_0 \\ \theta_0 \\ \dot{\theta}_0 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0.05 \\ 0.1 \\ 0.05 \end{bmatrix}$

En la Figura 1.2, se puede observar la respuesta del sistema con respecto a θ (posición angular del monociclo). Esta muestra la convergencia del sistema al ángulo 0, objetivo del sistema de control. Lo mismo para la Figura 1.3, donde ahora la variable que se observa es la velocidad angular y la misma converge al 0.

Como era de esperar, el sistema simulado en Matlab es que es asintóticamente estable. De todos modos, no puede garantizarse que el modelo sea perfecto y que los mismos excelentes resultados de estabilidad se obtengan con el hardware elegido, como se analizará en las secciones subsiguientes.

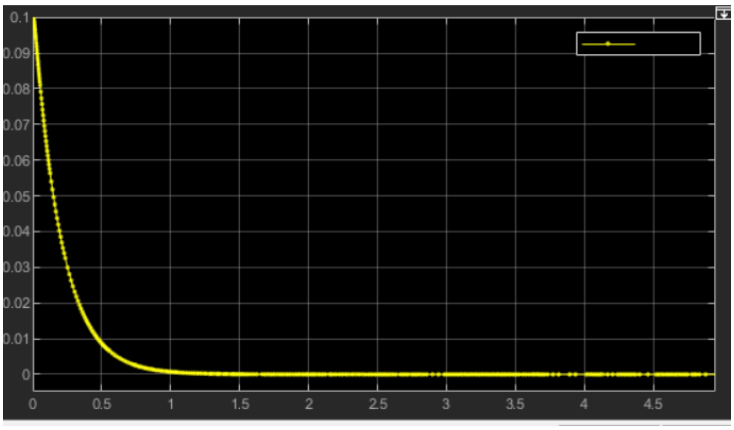


Figura 1.2.- Simulación θ(t)

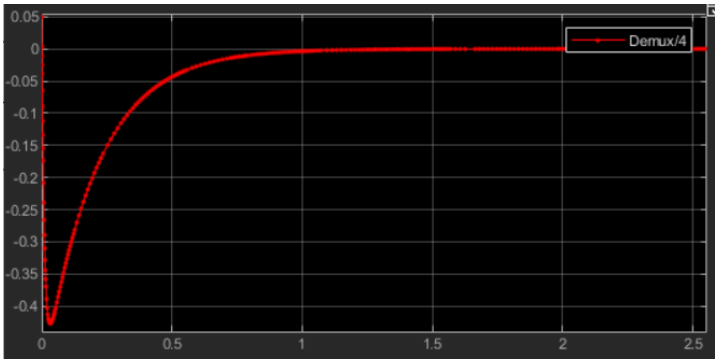


Figura 1.3.- Simulación dθ/dt(t)

1.2.2 Representación de Diagrama en bloque

El diagrama de bloque del sistema planteado en “Control de estabilidad vertical” es el visto en la Figura 1.4 en donde el modelo matemático es lo expresado en la ecuación (1.3) y el control es la ecuación (1.4), por tanto, se entiende que $u(t)$ es el torque y es la entrada de control de la ecuación (1.3) (modelo matemático) mientras que la salida de este es la matriz Z(t), en nuestro caso hacemos uso de θ(t) y dθ(t)/dt.

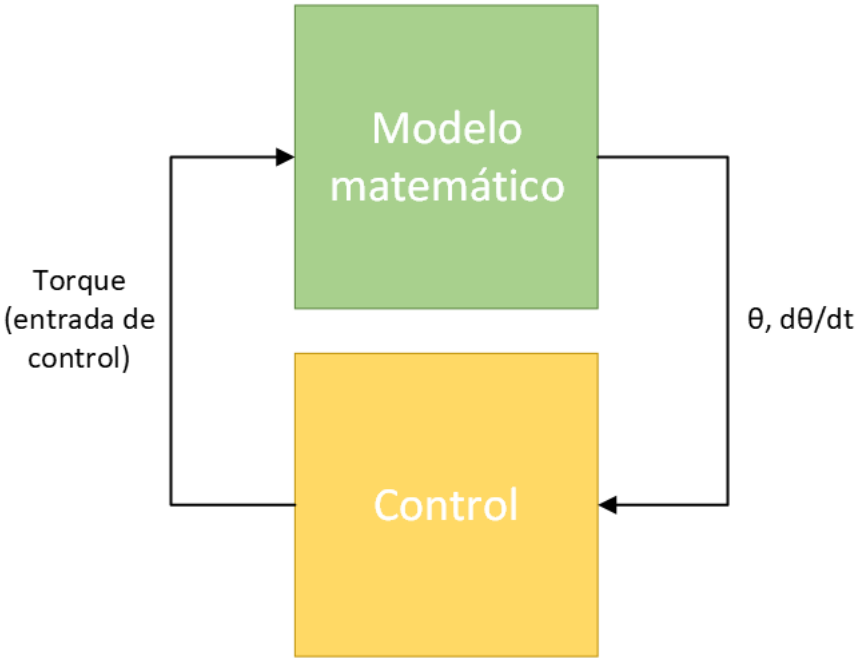


Figura 1.4.- Diagrama en Bloque

Una vez implementada esta estrategia de control en hardware usando el microcontrolador Raspberry Pi Pico W con motores BLDC y drivers comerciales, se observó que dichos drivers no poseen un control transparente de torque, con agregados de rampas de aceleración/desaceleración que modifican la dinámica y estabilidad.

En la sección siguiente se analiza el impacto sobre el diagrama de bloques presentado y sus consecuencias sobre la estabilidad del sistema.

1.3 Control con motores NEMA

Como se mostró en la sección previa, el uso de motores BLDC es muy adecuado cuando se poseen drivers transparentes que no posean algoritmos de lazo cerrado interno que lentifiquen el sistema de control. Por este motivo, como segunda opción se decide utilizar motores PAP de alto torque con drivers DM542, ver sección sistema de motores donde se detalla el cambio. Esta modificación altera el diagrama en bloque ya que el driver del motor paso a paso

se lo puede ver como una caja negra o gris[3], cuya entrada es velocidad angular y no con Torque como lo era con los motores Brushless. Por ende, el diagrama en bloque nuevo es el mostrado en la Figura 1.5. Como se puede ver en la figura, hay un inconveniente presentado en el punto rojo del diagrama, se trata de que la salida del control es torque y al driver de PAP se ingresa con velocidad.

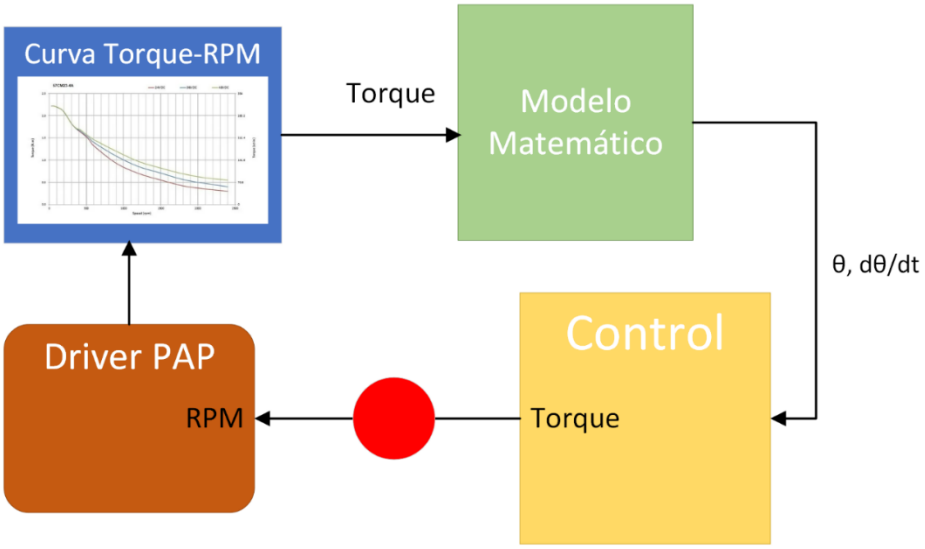


Figura 1.5.- Diagrama de bloque con el Driver

Claramente, el dinamismo que aporta el driver es un factor crucial en el análisis de sistemas de control. La inclusión de un nuevo bloque que contemple la inversa de esta dinámica, como se ilustra en la Figura 1.6, es esencial para una comprensión completa del comportamiento del sistema. Es importante destacar que la interacción de drivers que resulta en una dinámica discreta o no lineal en los sistemas de control es un área que ha recibido poca atención en la investigación actual. Por esta razón, se está preparando el artículo científico [3].

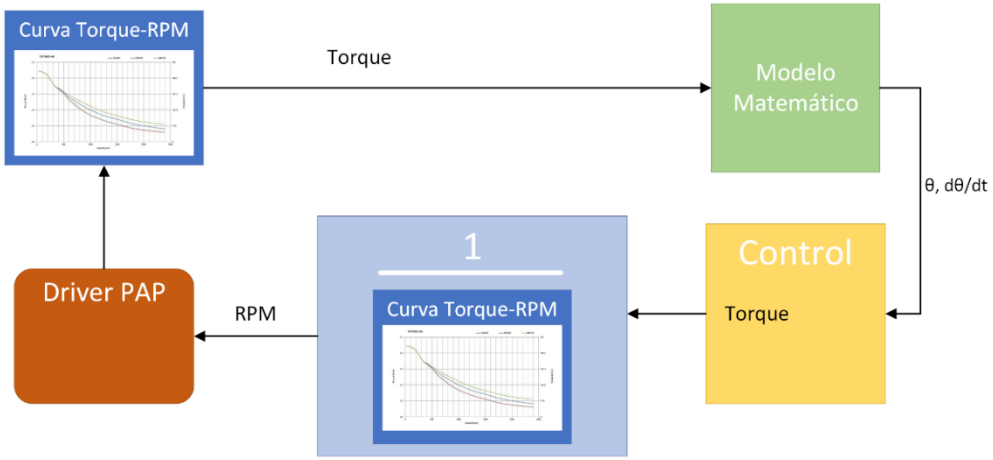


Figura 1.6.- Diagrama de bloque con el Driver

[3] Este término se utiliza en la jerga cuando un sistema se lo identifica mediante datos de simulación y no en base a modelos matemáticos.

1.3.1 Simulación del sistema con motores paso a paso.

Se integró el nuevo modelo al entorno de Matlab, utilizando la herramienta Simulink. Dentro de Matlab, se empleó SimMechanic para modelar físicamente el sistema, como se muestra en la Figura 1.7. Las condiciones de simulación se mantuvieron iguales a las descritas en la sección 1.2.1, con la adición de los bloques correspondientes al sistema del motor paso a paso ilustrados en la Figura 1.6. El diagrama en bloque de Simulink se presenta en la Figura 1.8, donde el bloque verde representa la aproximación de la curva del driver de velocidad angular – torque. Los bloques anteriores están asociados al control propuesto en [3].

Los resultados de la simulación, mostrados en la Figura 1.9, revelan una ligera oscilación alrededor del valor 0 con un desfase de -1.94°. Dicha oscilación es consistente con el comportamiento observado en el modelo real, como se evidencia en el video B [21].

En conclusión, el sistema diseñado es estable pero no alcanza una estabilidad asintótica completa. Esto se debe a que la estimación de la curva del driver no es perfecta, lo que impide garantizar su estabilidad asintótica.

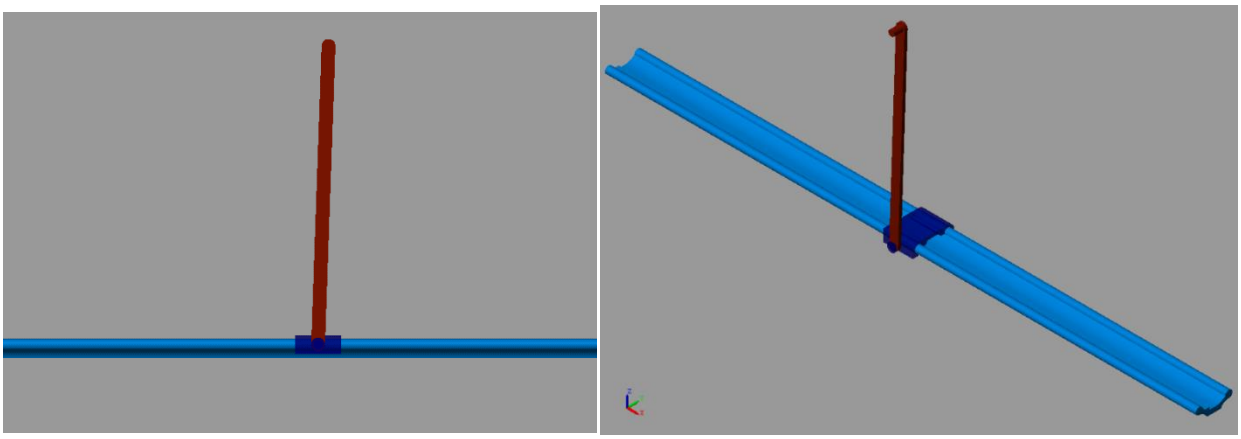


Figura 1.7.- SimMechanics

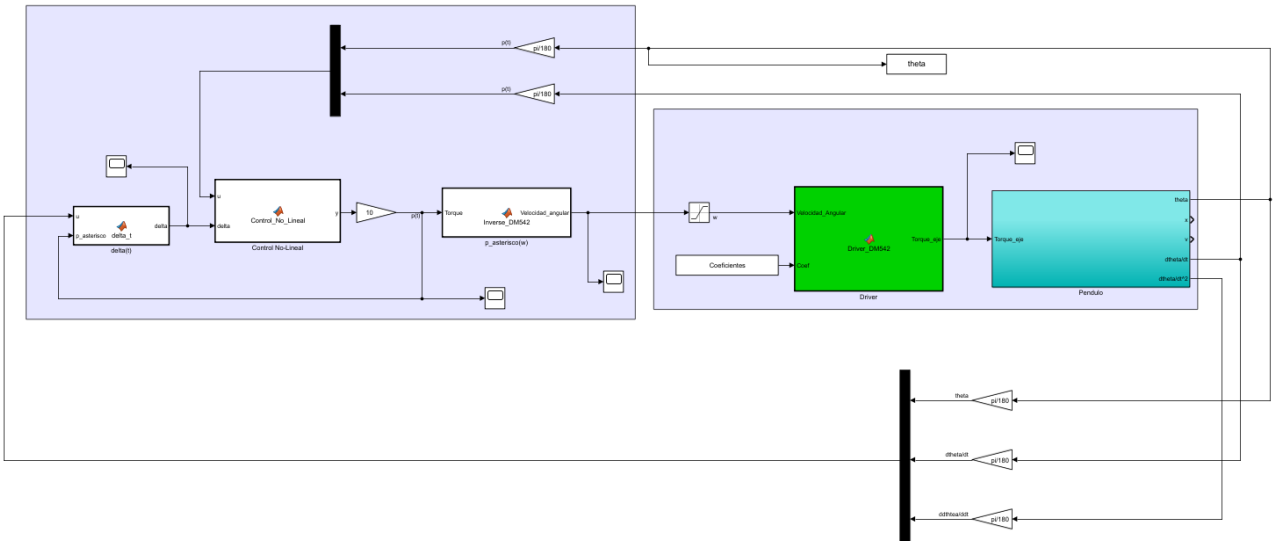


Figura 1.8.- Diagrama en Bloque en Simulink

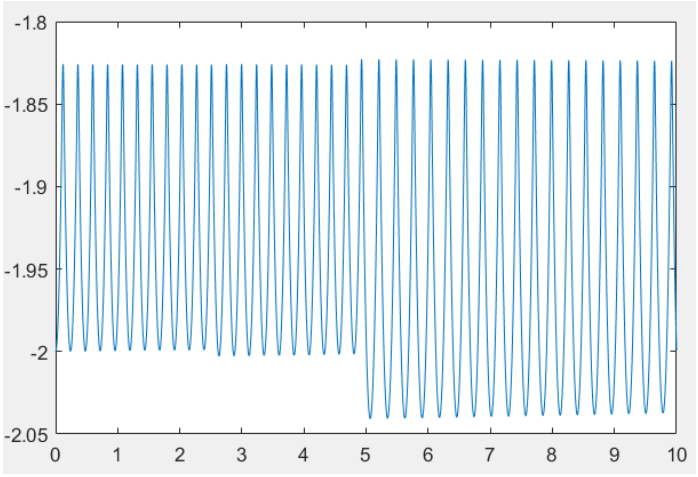


Figura 1.9.- Respuesta θ(t)

2 Capítulo 2: Algoritmo de control de navegación del robot.

2.1 Introducción

La navegación autónoma es el proceso de hacer que un vehículo se mueva de un punto a otro sin la ayuda de un conductor humano. Para lograr esto, el vehículo debe ser capaz de percibir su entorno, estimar su posición y orientación, y planificar una ruta óptima hacia su destino.

La navegación autónoma es un campo amplio y dinámico que presenta constantes avances en las diferentes técnicas que se desarrollan día a día [12]. Este tema ofrece muchas posibilidades para un estudio más profundo, e incluso una tesis doctoral. Por lo que se limitará en exponer algunas ideas y en mostrar el algoritmo elegido.

Existen diferentes niveles de autonomía, desde los sistemas telecomandados que requieren un operador remoto, hasta los sistemas completamente autónomos que pueden tomar decisiones por sí mismos.

Dentro de los completamente autónomos, hay dos enfoques, ver tabla 1.0.

Tabla 1.0: Enfoques de la N.A.

Enfoque Heurístico	Enfoque Optimo
Comportamiento práctico	Requiere información de todo el medio ambiente
No hay garantías de ser optimo	Se busca soluciones optimas de trayectoria
No necesita información de todo el medio ambiente	

Este proyecto se focalizará en un intermedio entre sistema telecomandados y sistema autónomo.

2.1.1 Propuesta.

La navegación autónoma es un proceso complejo que requiere de varios componentes que trabajan de forma coordinada para lograr el desplazamiento de un vehículo sin intervención humana [13]. Estos componentes se pueden agrupar en subsistemas que cumplen una función específica dentro del sistema general. Algunos de estos subsistemas son: el subsistema de medición, que realiza la toma de datos; el subsistema de percepción, que se encarga de interpretar la información del entorno mediante sensores; el subsistema de planificación, que genera una ruta óptima para alcanzar el destino deseado; y el subsistema de control, que ejecuta las acciones necesarias para seguir la ruta establecida (Ver Figura 1.11).

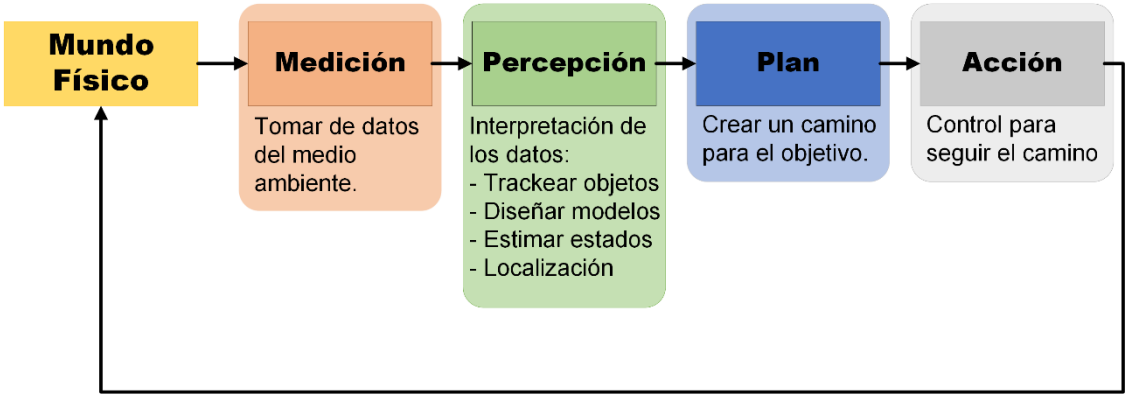


Figura 1.11.- Capacidades de los sistemas autónomos.

2.1.2 Dificultades de la navegación autónoma

La principal dificultad radica en que el entorno en el cual se va a navegar no se lo conoce perfectamente, el mismo es cambiante por ende es difícil de conocer. Cuanta más incertidumbre tenga sobre el medio ambiente, más errores en el modelo de navegación se tendrá. Cabe aclarar que existen entornos estructurados donde se conoce el entorno, pero estos son casos excepcionales.

2.1.3 Solución

La solución propuesta para este trabajo es poner en práctica un algoritmo de *"path planning"*, es decir, planificación de ruta. La idea es que se pueda introducir unas coordenadas y el robot busque una trayectoria que une dos puntos dados en un tiempo mínimo. Para ellos se tuvieron en cuenta el algoritmo planteado en [5].

2.1.4 Algoritmo de "path planning".

En referencia a lo demostrado por Murray y Shankar (1993), todo sistema no holonómico[4] se puede escribir de una manera universal. En particular, las ecuaciones de cinemáticas que describen el movimiento de un robot uniciclo se presentan en (1.5):

$$\begin{aligned} \dot{x}(t) &= \cos(\theta(t)) \cdot u_1(t) \\ \dot{y}(t) &= \sin(\theta(t)) \cdot u_1(t) \\ \dot{\theta}(t) &= u_2(t) \end{aligned} \tag{1.5}$$

Donde $u_1(t)$ y $u_2(t)$ son las velocidades lineales y rotacionales respectivamente, y variables de control.

Para poder programar dicha dinámica, debemos discretizarla:

$$\begin{aligned} x(k+1) &= x(k) + (\sin(\theta(k) + u \cdot \Delta_t) - \sin(\theta(k))) \cdot u \\ y(k+1) &= y(k) + (-\cos(\theta(k) + u \cdot \Delta_t) + \cos(\theta(k))) \cdot u \\ \theta(k+1) &= \theta(k) + u(k) \cdot \Delta_t \end{aligned} \tag{1.6}$$

[4] Sistemas no holonómico significa que el número de grados de libertad controlables es menor que el número de grados de libertad que experimenta el sistema.

Donde [$x(k+1),\ y(k+1), \theta(k+1)$] indican los valores de las variables de estado en los instantes $t + \Delta_t$. Además, se ha considerado el caso Dubins: Sólo trayectorias hacia delante de máxima velocidad lineal y por ello: $u(t) = [1, u(k)]', u(k) = \pm 1$ [6].

El algoritmo implementado asegura que el robot une los puntos iniciales X(0), donde se encuentra el robot, y el origen, el punto donde se dirige, en el menor tiempo posible y para ello se realiza una toma de decisiones en tiempo real, en el cual se selecciona entre: seguir derecho, doblar a la izquierda o derecha simplemente analizando la distancia desde el punto actual del robot hasta el origen. El código del algoritmo se encuentra en el Anexo IV.

3 Capítulo 3: Diseño e implementación del robot monociclo.

En este capítulo se muestra el hardware usado, la evolución del concepto-prototipo y se explica la estructura del sistema.

3.1 Estructura del robot

El robot presenta la siguiente estructura, con sus subsistemas (Ver Figura 2.0):

- Input: corresponde a la entrada de información del ambiente que le permite al robot navegar hacia un destino establecido. Esta información proviene de un acelerómetro y a eso se le adiciona una cámara.
- Sistema de Control: la estabilidad y la navegación, desarrollado en los capítulos, “Algoritmo de control de navegación” y “Algoritmo de control vertical del robot”.
- Motorización.
- Sistema de alimentación o baterías. Función principal carga de baterías.

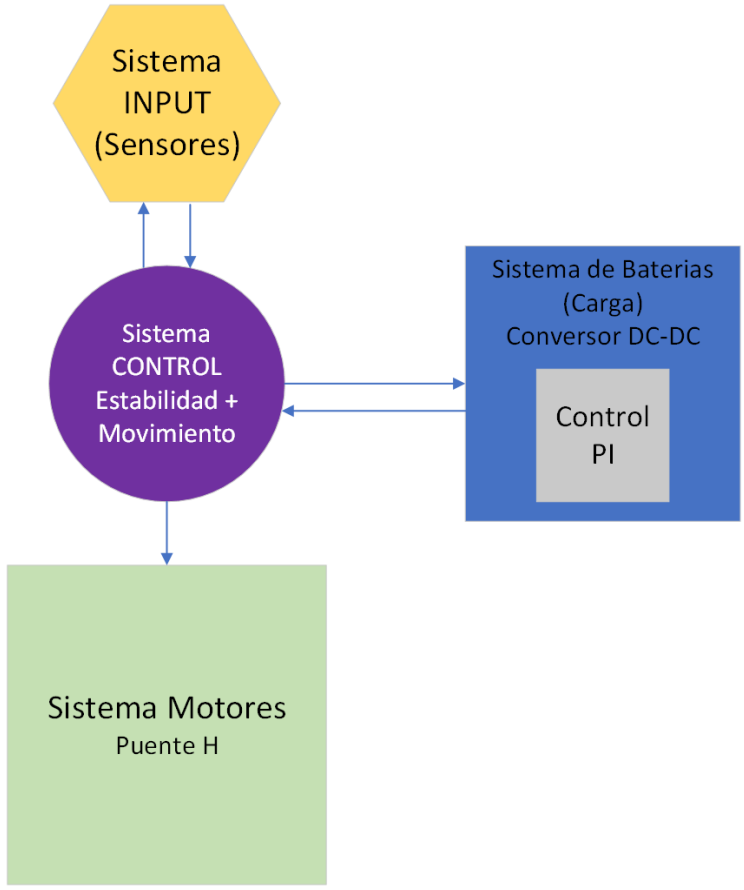


Figura 2.0.- Diagrama de la estructura.

3.2 Input

Las entradas de este sistema son el acelerómetro MPU6050 y la cámara RP Noir. El MPU está conectado a la Raspberry Pi Pico W 2022 vía comunicación I2C. En cambio, la cámara está conectada a otra placa con mayor capacidad computacional, que de hecho conforma una verdadera mini-computadora. Ver la siguiente tabla 2.1 resumen.

Tabla 2.1: Las entradas del sistema.

Imagen	Nombre	Descripción
	MPU6050	Acelerómetro-Giróscopo, mide velocidades y aceleraciones angulares en los tres ejes. Está conectado a las entradas de la Raspberry Pi Pico W.
	RaspberryPiNoir_v2	Cámara utilizada para procesamiento de imagen. Está conectada a la Raspberry Pi 4.

3.2.1 MPU6050

Es una unidad de medición inercial (IMU) que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es ampliamente usado en aplicaciones de navegación, goniometría, estabilización y más [14]. La comunicación con el módulo se realiza a través de I2C (Inter-Integrated Circuit).

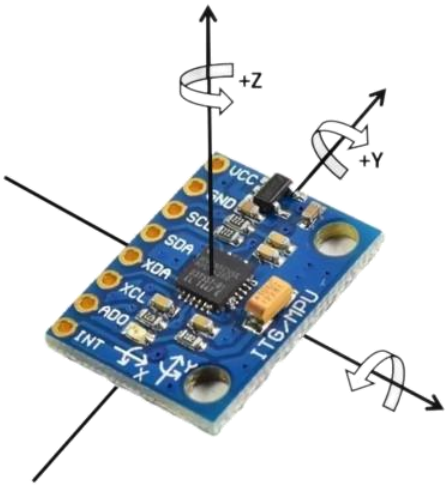


Figura 2.1.- MPU6050 y sus ejes.

En la Figura 2.1 se observan los ejes y sus orientaciones siendo este detalle muy importante a la hora de utilizar las mediciones junto al modelo matemático de control. El giro angular respecto al eje “x”, se lo denomina Roll; el giro angular del eje “y” se lo llama Pitch; y, por último, el giro angular del eje “z” se lo llama Yaw. En la Figura 2.2 se deja en claro lo mencionado anteriormente.

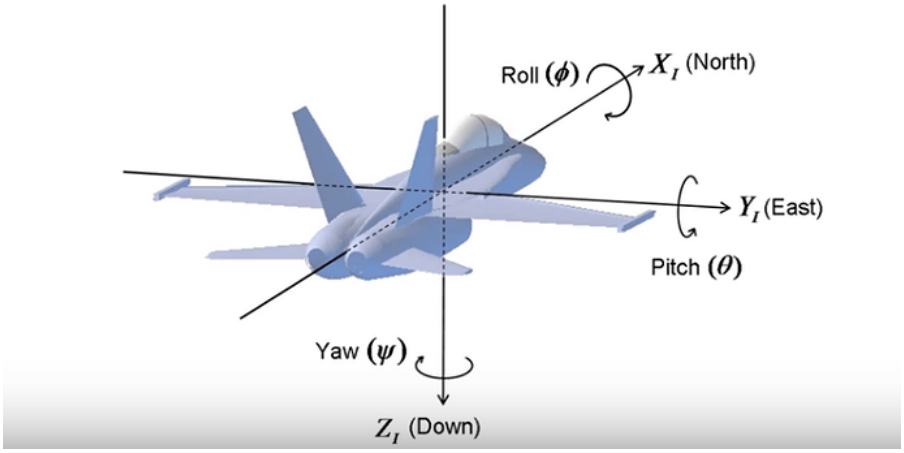


Figura 2.2.- Ejes.

3.2.1.1 Configuración

Luego de varias simulaciones y experiencias sobre este módulo, se concluyó que es importante la configuración para minimizar lo máximo posible los errores en el sensado.

Sobre su ubicación dentro del robot, es importante que el mismo esté aislado de posibles inducciones de drivers de motores, motores, etc., ya que el mismo sensor es muy sensible a ruidos electromagnéticos y esto complica el correcto funcionamiento del sistema. Desde la experiencia que dejan las distintas simulaciones realizadas, este punto es de vital importancia. En la Figura 2.3 se puede observar una de las ubicaciones donde se estudió el sensor.

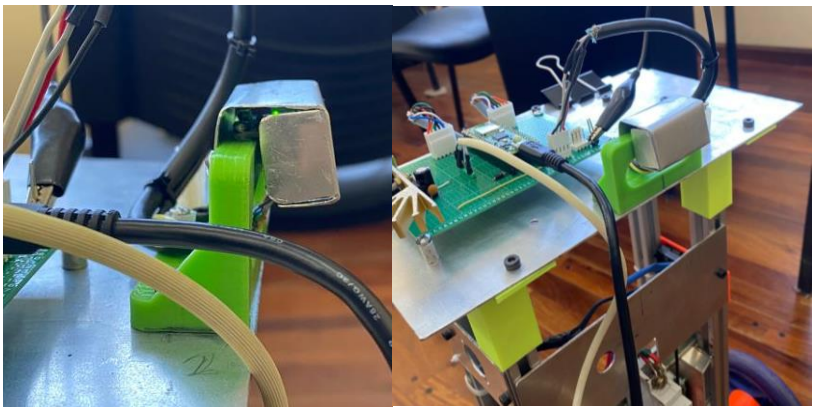


Figura 2.3.- MPU6050.

Sobre su conexión a la placa Raspberry Pi Pico W[5], la misma está conectada en los siguientes pines de la placa (Tabla 2.2):

Tabla 2.2: Entradas del sensor a las RPI pico W.

Pin sensor	Pin placa RPI Pico W
Vcc (Alimentación)	Pin 36 (3V3 (Out))
GND	Pin 3 (GND)
SCL	Pin 5 (I2C1 SCL)
SDA	Pin 4 (I2C1 SDA)

La lógica de su programación para obtener los ángulos Roll, Pitch y Yaw se basa en utilizar las mediciones del Acelerómetro y Giroscopio. La misma no basta ya que aparecen distintos errores, como lo son el ruido, entre otros [7]. Para ello se diseña un filtro complementario, donde unimos lo mejor de los dos apartados y eliminamos el ruido[6] y el drifting[7] [15].

Estos errores que mitiga el filtro complementario se tratan en el trabajo de investigación [7]. Donde se concluye que los errores acumulativos, aun usando estos filtros, son difíciles de mitigar y que en sistemas donde existe vibraciones, la performance de este tipo de sensores no es eficiente y en sus mediciones se inducen diferentes errores.

Este punto es de gran valor contando los aportes del proyecto, ya que provee, además, un estudio profundo dejando como problema abierto, la necesidad de un algoritmo de estimación eficiente de ángulos para navegación con sensores MPU 6050.

El filtro complementario es una técnica que combina las mediciones de acelerómetro y giróscopo para mejorar la precisión de los datos sensados. Por ejemplo, en el análisis del Pitch (sección 3.2.1.2), se utiliza el acelerómetro para determinar este ángulo y se complementa con la información proporcionada por el giróscopo, corrigiendo así cualquier error potencial.

3.2.1.2 Filtro complementario de Pitch

$$Pitch_{comp} = Pitch_{acel} \cdot 0.05 \ + \ 0.995 \cdot (Pitch_{comp} + X_{Gyro} \cdot t) + Error_{Pitch} \cdot 0.002$$

, donde

$Pitch_{acel} = \tan^{-1}(\frac{y_{acel}}{z_{acel}})$ [rad]

$Error_{Pitch} = \ Error_{Pitch} + (Pitch_{acel} \ - \ Pitch_{comp}) \ \cdot \ t$

[5] En Anexo III, se puede observar la Raspberry con sus pines.
[6] Se refiere a las fluctuaciones no deseadas o aleatorias que afectan las señales o los datos obtenidos.
[7] Se refiere al cambio gradual y continuo en el valor de una medición a lo largo del tiempo, incluso cuando las condiciones aparentemente no han cambiado.

3.2.1.3 Filtro complementario de Roll

$$Roll_{comp} = Roll_{acel} \cdot 0.05 \ + \ 0.995 \cdot (Roll_{comp} \ + \ Y_{Gyro} \cdot t) + \ Error_{Roll} \cdot 0.002$$

, donde

$$Roll_{acel} = \tan^{-1}(\frac{x_{acel}}{z_{acel}}) \text{ [rad]}$$

$$Error_{Roll} = Error_{Roll} + (Roll_{acel} \ - \ Roll_{comp}) \ \cdot \ t$$

3.2.1.4 Filtro complementario de Yaw

Sobre este ángulo es imposible realizar un filtro complementario ya que el acelerómetro no puede brindar dicho ángulo [7]. Se optó a utilizar la medición del giroscopio y asumir el error de drifting. Esto es una limitación del hardware porque no se puede obtener "yaw" con el acelerómetro y no se puede mitigar el drifting mediante el filtro complementario. Como se mencionará, con este proyecto, también es posible concluir que se posee allí una ventana de estudio para el desarrollo de algoritmos es estimación de sensores comerciales usados para navegación de robots.

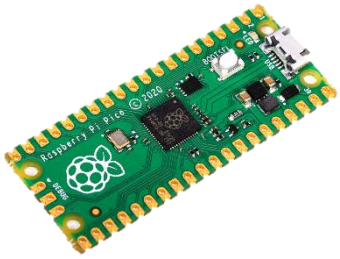
Este problema con dicho MPU6050 es algo que aún hoy en día no se encuentran artículos o trabajos donde lo resuelvan efectivamente.

3.3 Sistema de control

En este apartado solo se trata de mostrar los dos "cerebros" del robot. En la siguiente tabla resumen se detalla su función dentro del sistema.

Tabla 2.3: Los dos "cerebros".

Imagen	Nombre	Descripción
	Raspberry Pi 4 8gb RAM	Ordenador el cual su objetivo principal es procesar las imágenes que le ingresen de la cámara. El procesamiento de imágenes no está dentro de los alcances de este proyecto.

	Raspberry Pi Pico W 2022	Microcontrolador el cual llevará consigo sistema de control de estabilidad y navegación. El sensor antes mencionado se conectará a esta placa.

3.4 Sistema de Motores

En este capítulo se establecerán las especificaciones mínimas requeridas para los motores, lo que permitirá su selección adecuada. Se utilizará el contenido estudiado en la asignatura de **Accionamientos y Controles eléctricos**[8] como base para este fin.

Se emplea una herramienta en forma de hoja de cálculo proporcionada por la cátedra mencionada para determinar las especificaciones mínimas necesarias. Esta hace uso de la Ecuación 2.1, que está fundamentada en la Ley de Newton. Esta ecuación se utiliza para analizar al robot como objeto de estudio en cuanto a las fuerzas que actúan sobre él, tal como se muestra en la Figura 2.4.

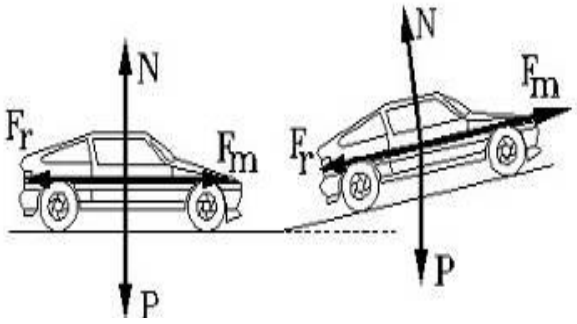


Figura 2.4.- Diagrama de cuerpo libre.

Tabla 2.4.- Parámetros y Constantes

Parámetro	Descripción	Valor	Unidad
θ	Ángulo de pendiente	5	°
P	Peso del robot	147	Newton
N	Normal	146,44	Newton
F_r	Fuerza de Rozamiento	73,22	Newton
F_m	Fuerza del motor		Newton
F_v	Fuerza del viento	0,00047	Newton
c_d	Coeficiente de drag[9]	1,05	Adimensional
$\mu_{estático}$	Coeficiente de roza. estático[10]	0,5	Adimensional
A_f[11]	Sección efectiva al viento del robot	0,00945	m^2
ρ	Densidad volumétrica del aire	1,225	kg/m^3
v_v	Velocidad del viento	0,277	m/s
v_r	Velocidad del Robot	0,55	m/s
d_w	Diámetro de las ruedas	0,16	m

Se utilizarán los parámetros, variables y constantes enumerados en la Tabla 2-4 para el desarrollo matemático. Las ecuaciones 2.2 y 2.3 describen las

[8] Cátedra dada en 5to año de la carrera Ingeniería Eléctrica.
[9] Dicho coeficiente extraído por la tabla del Anexo I.
[10] Coeficiente del rozamiento estático para suelo de baldosa lisa.
[11] Considerando que el área de impacto del robot de manera frontal es muy bajo porque la superficie es abierta, tomaremos solo el aporte del área de las baterías.

fuerzas que inciden en el Diagrama de Cuerpo Libre (DCL) del robot, presentando las componentes referentes a los ejes x e y, respectivamente.

$$\sum F_q = m.\ddot{x}_q \text{ (2.1)}$$

$$F_x\colon\ F_{motor} - (F_r + F_v) - P.sen(\theta) = m.\ddot{x} \text{ (2.2)}$$

$$F_y\colon\ N - P.\cos(\theta) = 0 \text{ (2.3)}$$

, donde

$$P = m.g \text{ (2.4)}$$

$$F_{rozamiento} = \mu_d.N \text{ (2.5)}$$

$$F_{viento} = \frac{1}{2}.\rho.A_f.c_d.v_{viento}^2 \text{ (2.6)}$$

A través de la mencionada hoja de cálculo en Excel, hemos obtenido los valores fundamentales que resultan críticos para la determinación de los motores necesarios. (Tabla 2.5)

Tabla 2.5.- Especificaciones mínimas del motor

Especificación	Valor	Unidad
Potencia mínima del motor	47,8	W
Velocidad angular de la rueda	66,31	RPM
Torque en el eje	70,18	Kgf*cm

Es crucial reconocer la necesidad de ajustar los valores de Potencia y Torque indicados en la Tabla 2.5. Esta modificación se debe realizar dividiendo dichos valores por el número de motores ($n_{motores}$) del robot, ya que la contribución total de potencia y torque depende directamente de esta cantidad. Para este proyecto en particular, donde $n_{motores}$ es igual a 2, implica que hay un motor por cada rueda del robot. Además, es esencial consultar la Tabla 2.6 para conocer los valores mínimos requeridos para las especificaciones de un motor.

Tabla 2.6.- Especificaciones mínimas del motor unitario

Especificación	Valor	Unidad
Potencia mínima del motor	23,9	W
Velocidad angular de la rueda	66,31	RPM
Torque en el eje	35,09	Kgf*cm

En un primer momento los dos motores elegidos son dos motores de dos taladros atornillador Brushless SBD201S2K Stanley (Figura 2.5). Dicho modelo

presenta una prestación en torque 560,9 kgf*cm y en velocidad sin carga 0-1500 rpm. Tensión nominal 18V.

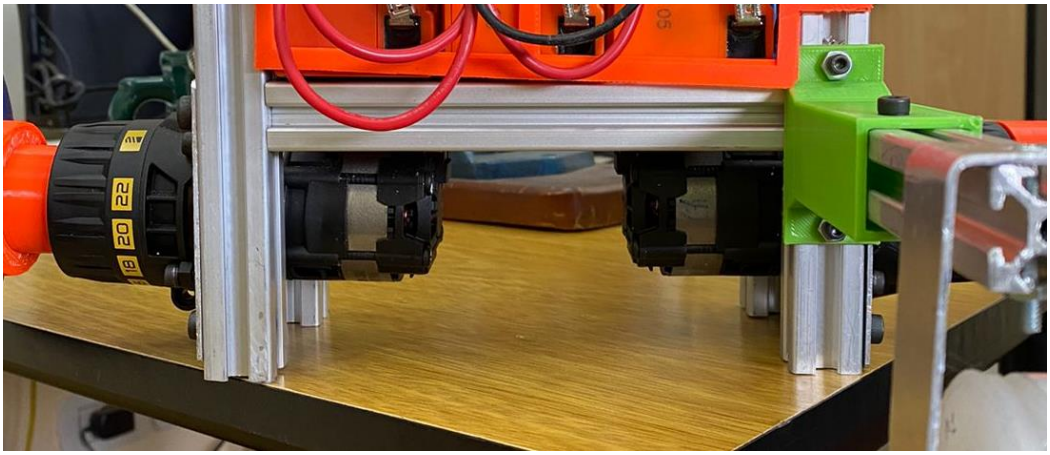


Figura 2-5.- Motores Brushless SBD201S2K Stanley

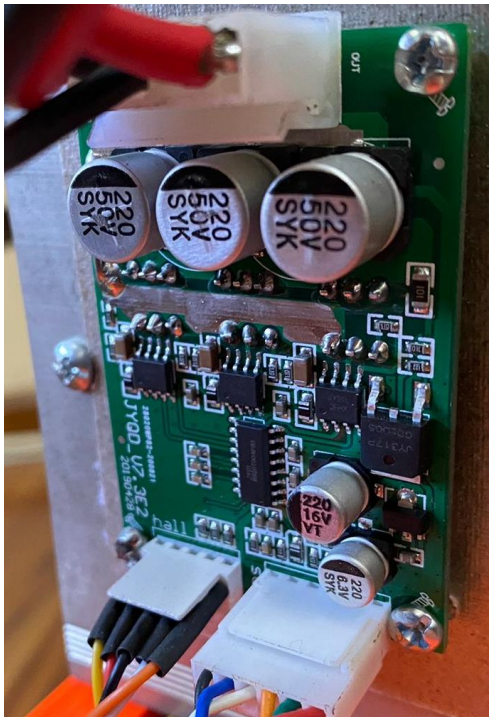


Figura 2-6.- Driver Motor Brushless DC

En [21] se puede ver el video A en donde se observa el modelo real de este sistema. Se puede advertir la lenta reacción del sistema.

Luego, debido a problemas con el hardware en las experiencias realizadas, se optó por cambiar los motores de Brushless DC a motores paso a paso, pero cabe aclarar que se tuvo en cuenta los cálculos realizados en esta sección. Este cambio, si bien no es óptimo, permite utilizar una tecnología industrial (motor paso a paso Nema 23) que en la práctica no se lo utiliza para estas aplicaciones, y que permitió el estudio de aplicaciones en robótica.

¿Cuál fue la razón detrás de este cambio? La limitada disponibilidad y acceso reducido a controladores de motores Brushless DC en Argentina influyó en la decisión de optar por controladores diseñados para monopatines (ver Figura 2-6). Estos controladores presentan una restricción particular: una rampa de aceleración. Durante las experiencias realizadas con el conjunto motor + controladores proporcionados se concluyó que esta limitación imposibilitaba el adecuado control de estabilidad del robot.

Como antes se aclaró, los nuevos motores son motores paso a paso NEMA 23 (Figura 2.7) con su correspondiente driver o controlador DM542T (Figura 2.8). Estos motores son alimentados con 24 V, por ende, se cambió las baterías de un banco de 18V a un banco mayor de 24V@7Ah. Si tenemos en cuenta que, mientras que el banco de 18V constaba de 3 baterías de 6 V en serie (18 V), luego pasaron a ser 2 baterías de 12 V en serie (24 V).

Figura 2.7.- NEMA 23

Figura 2.8.- DM542T

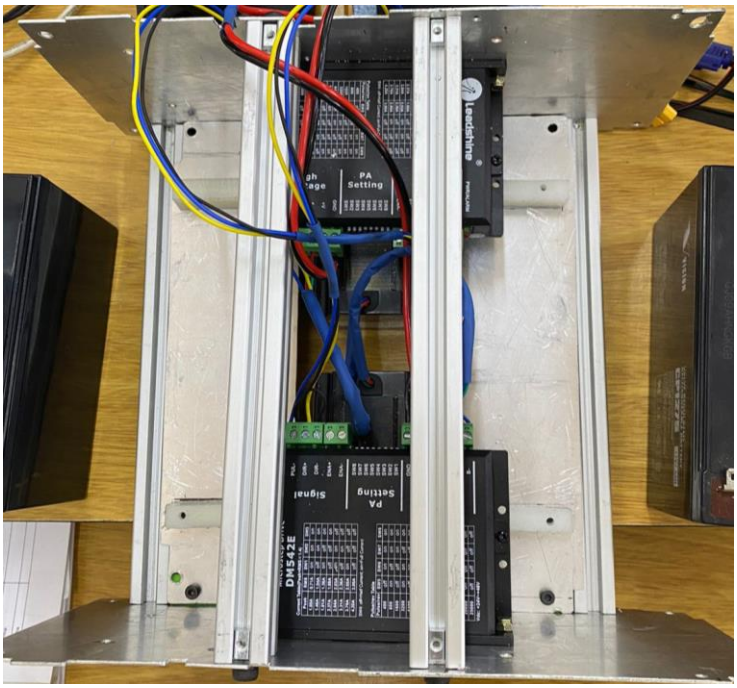


Figura 2.9.- NEMA 23 + DM542T

En la Figura 2.9, se puede ver la nueva base de robot con su nuevo banco de batería de 24V@7Ah. Un punto positivo sobre este cambio es que con esta nueva base se logró bajar el centro de masa del robot lo que permite facilitar el control de este. En la Figura 2.10 se puede ver como quedaría el robot en términos de estructura con la nueva base, motores + controlador, baterías y ruedas.

En [21] se puede ver el video B en donde se observa el modelo real de este sistema. Se puede advertir las altas vibraciones que alteran al sensor MPU6050 como se menciona en las conclusiones de [7]

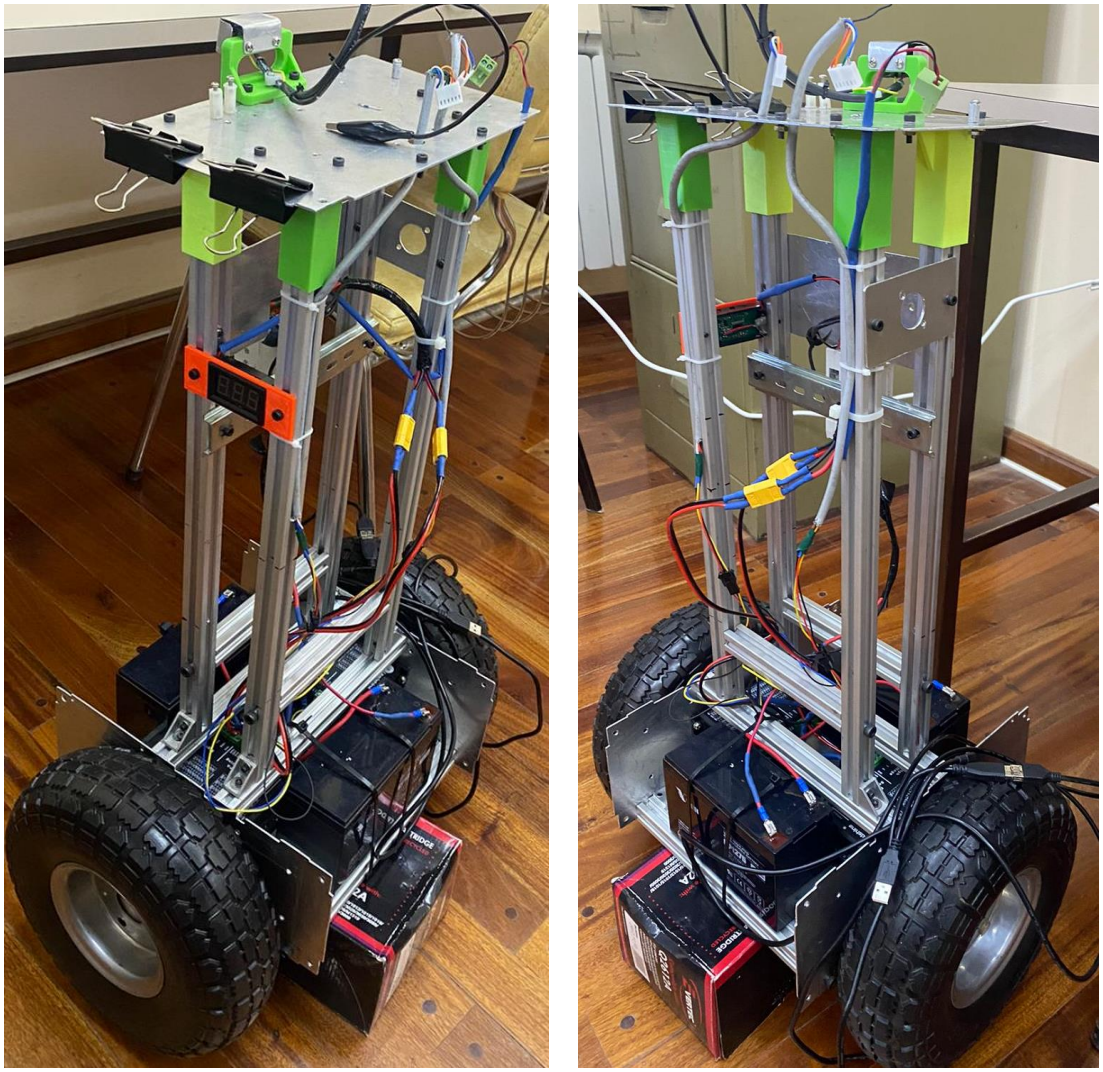


Figura 2.10.- Robot

4 Cargador

En este capítulo se llevará acabo el diseño del cargador de las baterías. Primero se explicará qué tipo de conversor se optó, su funcionamiento; las simulaciones del circuito diseñado y los softwares involucrados.

Uno de los softwares profesionales que permitieron el diseño inicial, además de Matlab/Simulink es LTspice; y por último el diseño llevado al software KiCAD donde se podrá ver la placa PCB del cargador. Se recurrirá a lo visto en la materia Electrónica de Potencia o llamada en el plan **Electrónica 2**[12] y materia electiva **Electrónica Aplicada**.

Primero se optó por un conversor DC-DC reductor.

4.1 Convertidores

La Figura 3-1(a)[13] muestra en diagrama del circuito un convertidor de CC-CC. La entrada a estos convertidores consiste normalmente en una tensión continua no controlada procedente de un rectificador de diodos. La carga es una batería, el filtro capacitivo se encuentra después del convertidor CC-CC. El convertidor es **reductor**.

El transistor o MOSFET (Q1) funciona como interruptor, cuando la base se alimenta por el circuito de control, la misma funciona como llave cerrada y abre el circuito cuando el control lo demande. La Figura 3-1(b y c) [14] muestra los dos ciclos de funcionamiento en base a si Q1 abre o cierra el circuito, los dos ciclos o modos de trabajo.

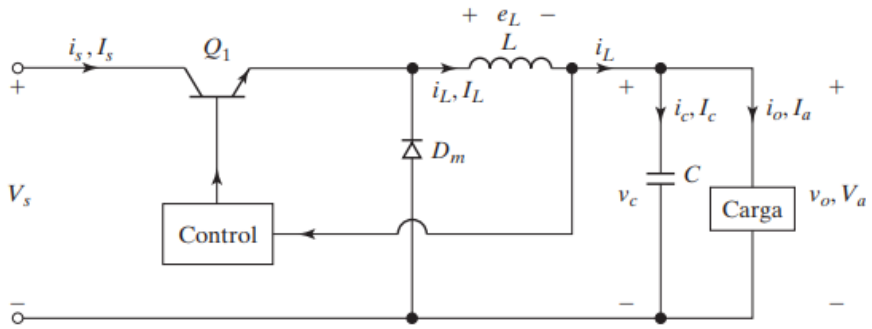


Figura 3-1.- Convertidor reductor.

[12] Cátedra dada en 5to año de Ingeniería Eléctrica en la FRBB UTN.
[13,14] Imágenes tomadas de “Electrónica de potencia”, M. H. Rashid, cuarta edición

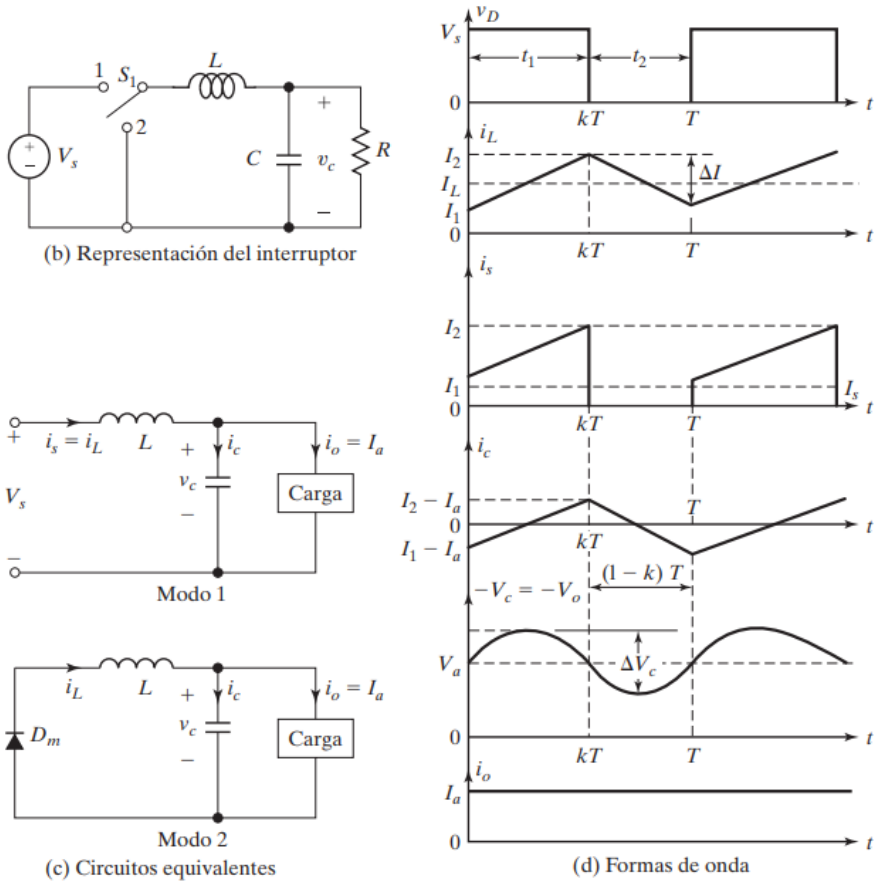


Figura 3-1.- Convertidor reductor.

El diagrama de control que muestra los elementos del regulador en modo de conmutación se muestra en la Figura 3-2[15].

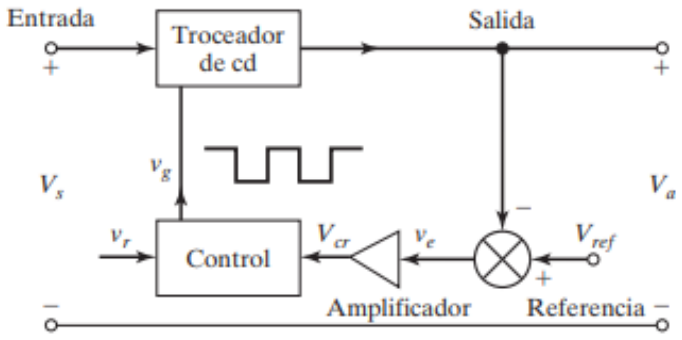


Figura 3-2.- Parte control.

[15] Imágenes tomadas de “Electrónica de potencia”, M. H. Rashid, cuarta edición

4.2 Diseño de la placa

Lo antes mencionado es meramente conceptual, para el diseño dicho cargador de baterías se utilizó conceptualmente el conversor CC-CC reductor (Buck) pero con una configuración sincrónica, lo que lleva a incorporar un MOSFET más y el control es vía modulación por ancho de pulsos (PWM) generados por un AT-MEGA 328P[16]. Para el excitador de las compuertas MOS se utiliza un Driver llamado IR2104[17].

En el Anexo VI, se puede ver el esquemático de la PCB. El transformador y el puente de diodos se encuentran fuera de la placa. El "J3 Input" presenta dos entradas alimentadas y una a GND. El pin 3 se vincula a la alimentación de toda la placa a un potencial de 5 V, el cual vincula al ATMEGA 328P, y LCD-016N002L (display). Del pin 2 la alimentación bifurca en una de 30 V al conversor, por ende, baterías y por otro lado mediante una técnica de regulación de tensión vía Diodo Zener (Regulación paralela) alimentar con 15 V el controlador IR2104 (sistema de control del conversor).

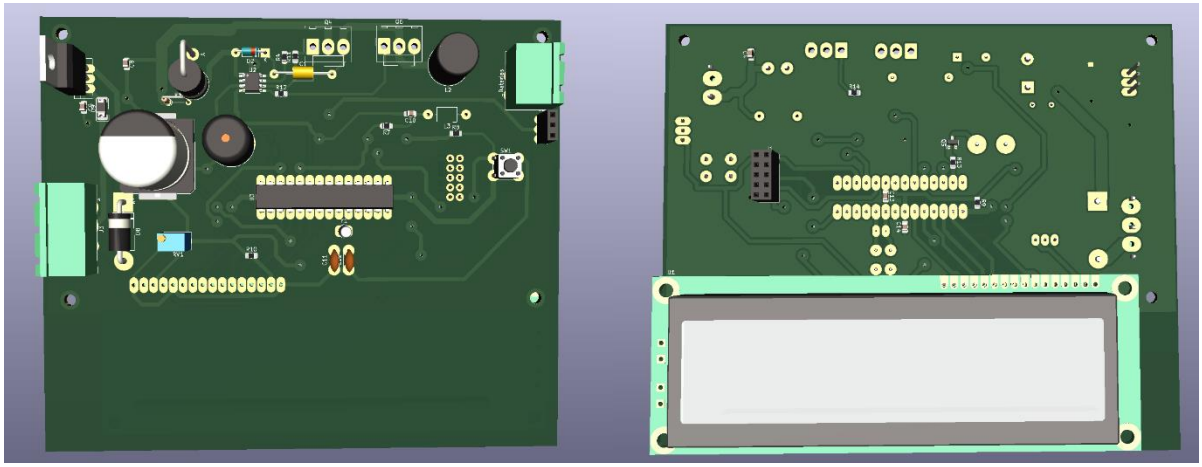


Figura 3-3.- PCB vista 3D.

En la Figura 3-3 se observa la placa en vista 3D creada por el software KiCad 7.0.

[16] Es un chip microcontrolador.

[17] Comercialmente encontrado como SOIC 8 IR2104S. Ver Anexo V donde se puede ver el datasheet del mismo.

4.3 Simulación del cargador via LTSpice

El circuito en LTSpice del cargador es el mostrado en la Figura 3-4.

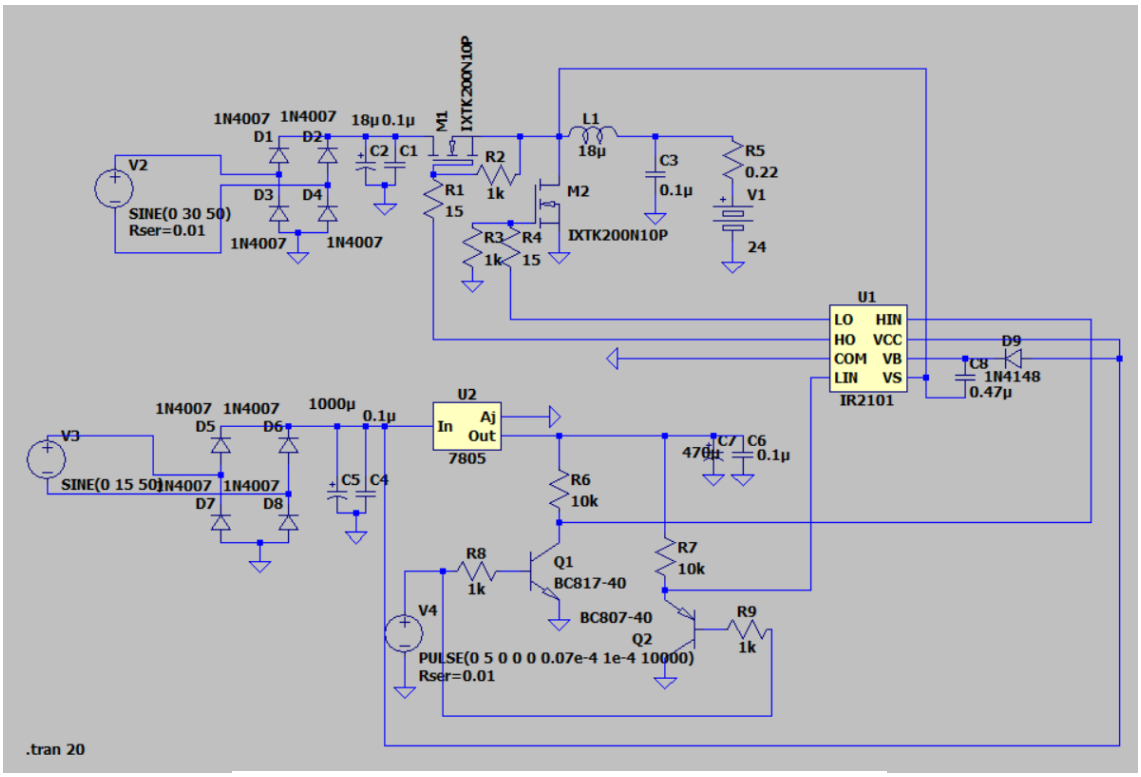


Figura 3-4.- Circuito para simular.

La simulación del circuito es el mostrado en la Figura 3-5, la tensión en la bateria es constante. La tensión 24 V es el valor nominal pero el mismo se puede regular a cualquiera de los valores de voltajes: bulk, float, etc. Es decir, el circuito mostrado en Figura 3-4 vale para todas las regiones de carga e inclusive para la región de corriente constante, donde varía la tensión. Estas regiones se determinan en la lógica impuesta en la programación del ATmega (Anexo IV).

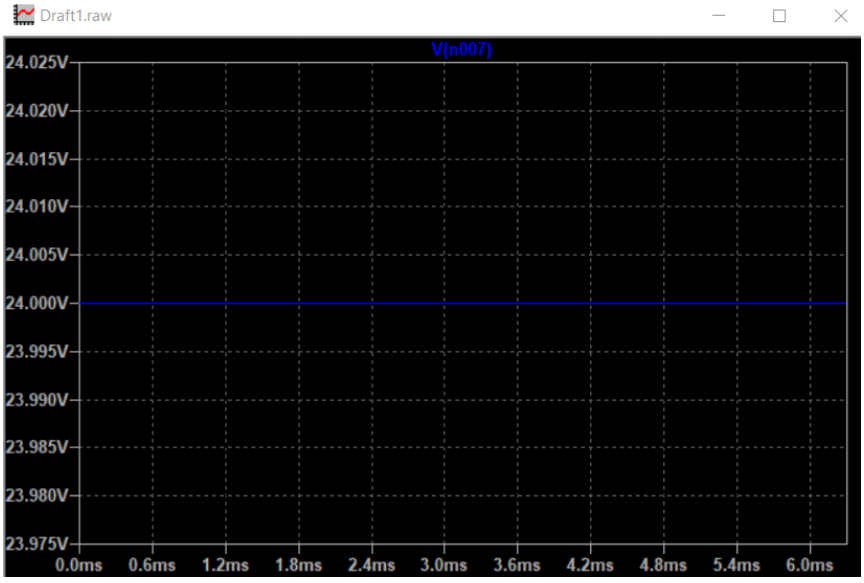


Figura 3-5.- Gráfica de la simulación de la tensión en la batería.

4.4 Control de cargador de baterías.

Para el control de baterías, se diseñó un PID como estrategia. Utilizando la herramienta de Simulink Matlab y su función de PID tuning, se determinaron los valores óptimos del PID para asegurar una respuesta estable del cargador. Estos valores se ajustaron basándose en la respuesta observada en la Figura 3-6. Los parámetros definidos fueron un setpoint de tensión de 18 V y un setpoint de corriente máxima de 1.5 A.

Incorporando un PI en el lazo de corriente y otro en el lazo de tensión, se logró la respuesta ilustrada en la Figura 3-6. El sobre pico registrado fue del 8 %.

Los valores del PI de corriente son:

- Proporcional = 0.16
- Integrativo = 0.18

Los valores del PI de tensión son:

- Proporcional = 7.3
- Integrativo = 3.2

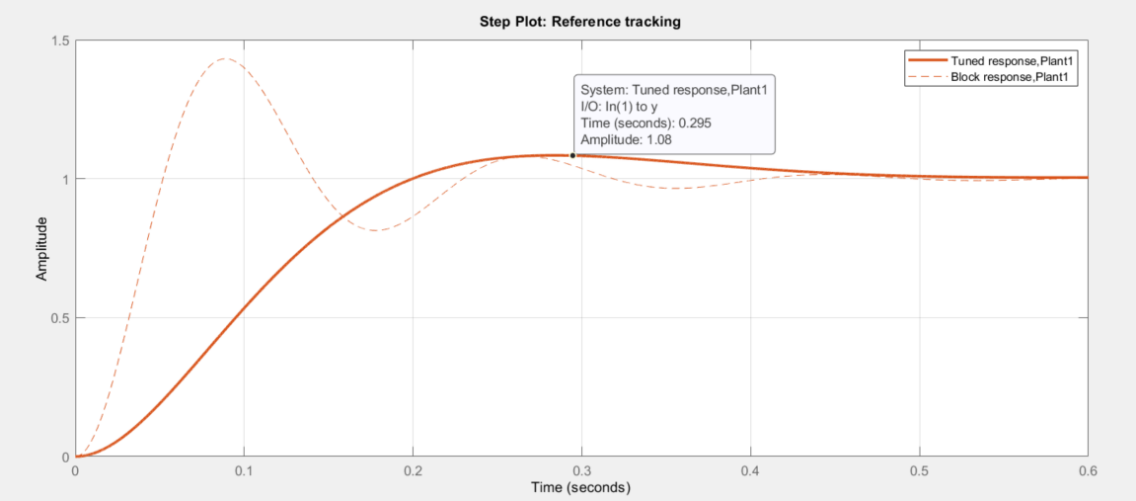


Figura 3-6.- Respuesta al escalón del PI

4.5 Programación del AT-Mega.

La programación del AT-Mega se realizó en el mismo entorno de programación que Arduino. Este código está diseñado para cargar una batería siguiendo un perfil CCCV (Corriente Constante Voltaje Constante) y mostrar los valores de tensión y corriente en una pantalla LCD. Utilizando un PI (Proporcional e Integrativo) para controlar el lazo cerrado de la carga de baterías. El mismo se puede observar en el Anexo IV.

5 Conclusiones

Para concluir, el hardware impone restricciones significativas y decisivas para alcanzar o no metas específicas. Aunque en este proyecto no se consiguió una convergencia clara a la estabilidad con las herramientas empleadas, esto se debió a las perturbaciones y limitaciones que una labor teórica no considera.

Se destacaría principalmente que las restricciones de acceder a drivers y actuadores especializados para este campo conlleva el uso de hardware limitado que necesita de una mayor matemática para superar los problemas que surgen.

Las incertidumbres del MPU6050 y la falta de técnicas que las eliminen son clave a la hora de cerrar el lazo de control, ya que esto es otro problema que impide estabilizar el sistema.

Por último, en la Figura 4-0 se muestra como fue evolucionando el trabajo final. Las experiencias reales de cada sistema se pueden ver en [21]. En el Anexo V se encuentra el algoritmo completo.

Figura 4-0.- Evolución

Dado que la literatura actual no presenta soluciones satisfactorias para el uso del MPU6050 como sensor, como parte de las líneas futuras de trabajo, se pretende implementar un método basado en free energy [20] para estimar el ángulo Yaw a partir de los datos del sensor antes mencionado. Además, se estudiará la posibilidad de diseñar una ley de control multivariable que controle el péndulo de estabilización vertical utilizando los motores de tracción(robot) para cumplir con ambos objetivos, navegación y estabilidad.

6 Agradecimientos

Este proyecto no hubiera sido posible sin el apoyo y la colaboración del departamento de ingeniería eléctrica, de la facultad y del profesor tutor. Quiero expresar mi más sincero agradecimiento a todos ellos.

7 Bibliografía y/o referencias

[1] García, A. G. & Arnaude, E. (2019). Non-Lyapunov control of a balancing robot. EN: Matemática Aplicada, Computacional e Industrial, 7, 553-556.

[2] Andrés García. Necessary and sufficient condition for asymptotic stability of nonlinear ODE's, Nonlinear Science Letters A, 9-2 (2018), pp.501-503.

[3] The impact of commercial drivers' nonlinearities on the upright control of balancing robots. Andrés García and Santiago Salvatierra. En preparación para enviar a una revista. 2024

[4] Jadlovska, Slavka & Jadlovska, Anna. (2010). Inverted Pendula Simulation and Modeling - a Generalized Approach.

[5] Luciano Pons, Ignacio Moyano & García A. G. (2012). Robot móvil en tiempo mínimo: Desafío de la NASA.

[6] Dubins, L.E. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. American Journal of Mathematics 79 (3) (July 1957), 497–516.

[7] Al-baghdadi, Ahmed & Ali, Abduladhem. (2019). An Optimized Complementary Filter For An Inertial Measurement Unit Contain MPU6050 Sensor. Iraqi Journal for Electrical and Electronic Engineering. 15. 71-77. 10.37917/ijeee.15.2.8.

[8] David Morin. "Introduction to Classical Mechanics with Problems and Solutions". Cambridge.

[9] Duvan Felipe Rodriguez Millan. (2019). "Modelado y simulación de una estación de carga de baterías para portabilidad energética en bicicletas eléctricas". Universidad autónoma de Bicaramanga.

[10] Isaac Gandarilla, Victor Santibáñez y Jesús Alberto Sandoval Galarza. (2018). "Stabilization of a self-balancing robot by energy shaping". Universidad Autónoma de Sinaloa y Asociación Mexicana de Robótica e Industria.

[11] A. BACCIOTTI AND L. ROSIER, Liapunov Functions and Stability in Control Theory, Springer Verlag, 2005.

[12] Miranda, V.R.F., Rezende, A.M.C., Rocha, T.L. et al. Autonomous Navigation System for a Delivery Drone. J Control Autom Electr Syst 33, 141–155 (2022). https://doi.org/10.1007/s40313-021-00828-4

[13] R. O. de Santana, L. A. Mozelli and A. A. Neto, "Vision-based Autonomous Landing for Micro Aerial Vehicles on Targets Moving in 3D Space," 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2019, pp. 541-546, doi: 10.1109/ICAR46387.2019.8981643.

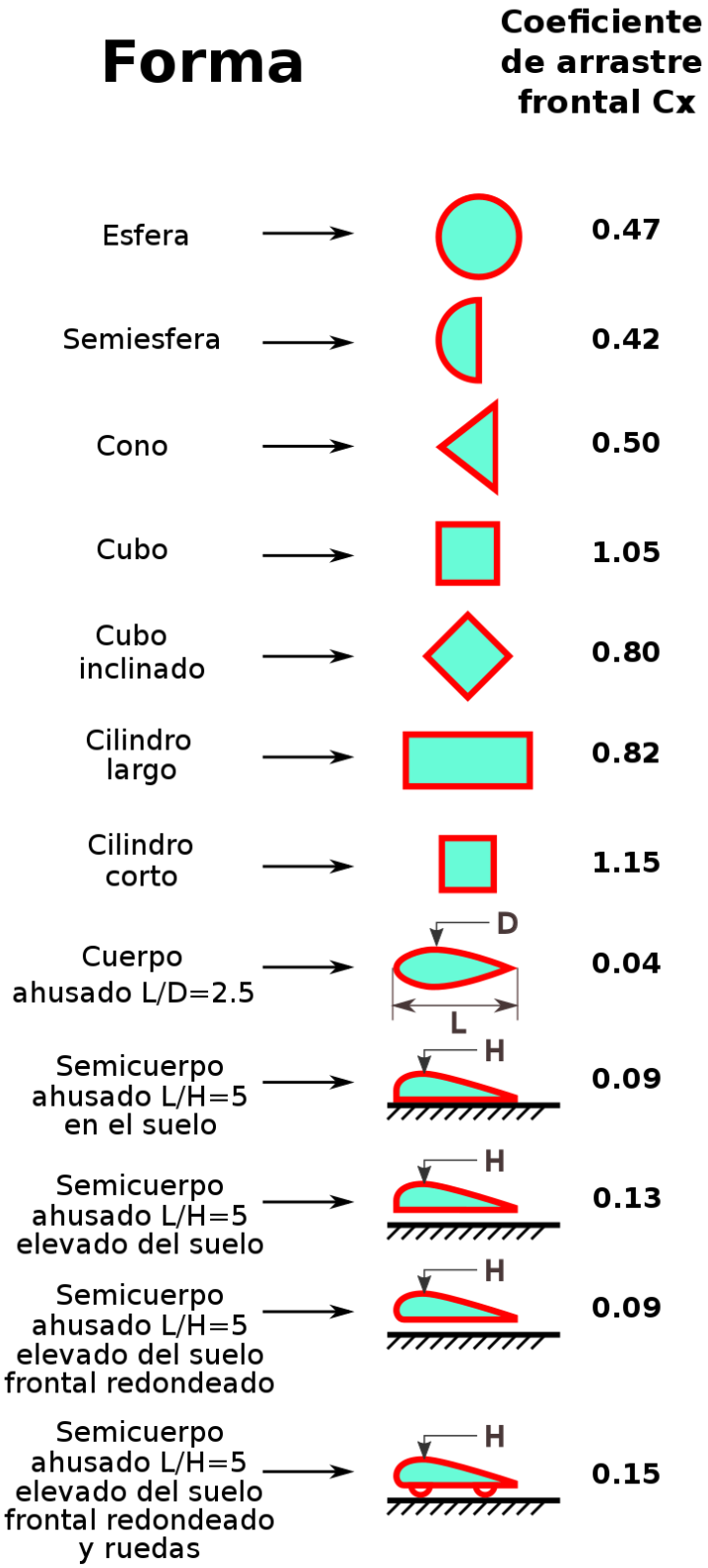
[14] Ooi, K R, Rosli, M A A, Latiff, A R A, Othman, W A F W, Alhady, S S N, Wahab, A A A. "Design of hoeckens linkage based walking robot with MPU6050 IMU as navigation sensor", 2021. https://dx.doi.org/10.1088/1742-6596/1969/1/012004

[15] https://toptechboy.com/ (Último acceso 7/3/2024)

[16] Jayakody, D.P.V.J. & Sucharitharathna, K.P.G.C.. (2019). Control Unit for a Two-Wheel Self-Balancing Robot. Global Journal of Researches in Engineering. 7-12. 10.34257/GJREJVOL19IS1PG7.

[17] Nawawi, S.W., Ahmad, M.N., & Osman, J.H. (2008). Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot. International Journal of Electrical and Computer Engineering, 2, 406-412.
[18] M. U. Draz, M. S. Ali, M. Majeed, U. Ejaz and U. Izhar, "Segway electric vehicle," 2012 International Conference of Robotics and Artificial Intelligence, Rawalpindi, Pakistan, 2012, pp. 34-39, doi: 10.1109/ICRAI.2012.6413423.
[19] L. Sun and J. Gan, "Researching of Two-Wheeled Self-Balancing Robot Base on LQR Combined with PID," 2010 2nd International Workshop on Intelligent Systems and Applications, Wuhan, China, 2010, pp. 1-5, doi: 10.1109/IWISA.2010.5473610.
[20] Tesis doctoral “Planeamiento de trayectoria con robot móviles aplicados a la sociedad” de Andrés Roteta. Lugar de trabajo: GIMAP. Beca CIC.
[21] Ver videos de simulaciones. Link: https://drive.google.com/drive/folders/1Vxw9k8-csfo48kvqDdO3AL8RmO6vmb7S?usp=sharing

8 Anexo I: Coeficiente de Drag.



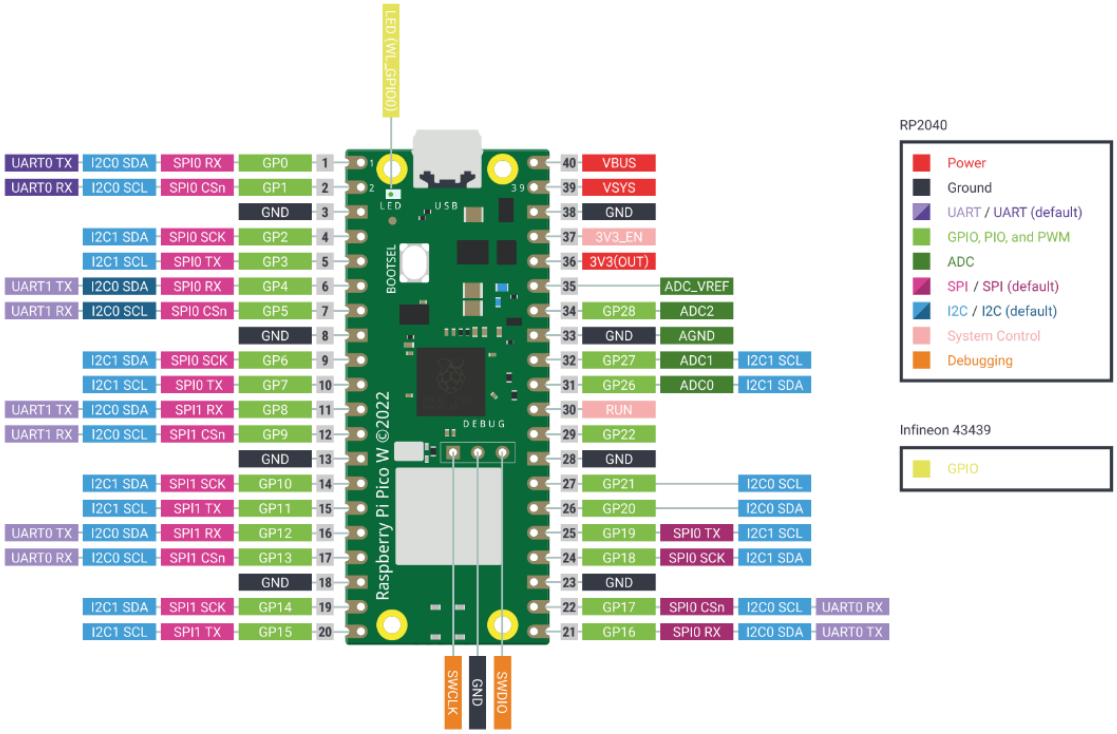
Valores del coeficiente de arrastre

9 Anexo II: Algoritmo de navegación.

```
from imu import MPU6050
from machine import I2C, Pin, PWM
import math
import time

#------------------MOTORS-------------------------
"-------Motores Stepper     Movilidad--------"

Motor_2 = PWM(Pin(13,Pin.OUT))    #Pin 17 del Micro
Motor_3 = PWM(Pin(16,Pin.OUT))    #Pin 21 del Micro

Motor_2.duty_u16(32768)  #Duty entre [0,65535] derecha
Motor_3.duty_u16(32768)  #Duty entre [0,65535] izquierda

#Defino cambio de giro
Cambio_giro_m2 = Pin(12,Pin.OUT) #Pin 16 del Micro
Cambio_giro_m3 = Pin(17,Pin.OUT) #Pin 22 del Micro

def Dubins(X_k,m_previous):

    global condic

    Delta_t = 0.001
    epsilon = 0.001

    u = 1 #Decisión de girar para algún lado
    Z = [X_k, u, Delta_t]
    [x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
    m1 = math.sqrt(x_k_1**2 + y_k_1**2)  # Euclidean norm of (x, y)

    u = -1  # Decisión de girar para otro lado
    Z = [X_k, u, Delta_t]
    [x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
    m2 = math.sqrt(x_k_1**2 + y_k_1**2)

    u = 0  # Decisión de ir en Linea Recta
    Z = [X_k, u, Delta_t]
    [x_k_1, y_k_1, tita_k_1]= Dinamica_Dubins(Z)
    m3 = math.sqrt(x_k_1**2 + y_k_1**2)


    #print(m1,m2,m3)

    #--------Searching for the minimum-----
    m = [m1, m2, m3]
    #print(m1,m2,m3)
    if m[0]<m[1]:
        Min=m[0]
        i=0
    else:
        Min=m[1]
        i=1

    if Min > m[2]:
        Min=m[2]
```

```
        i=2

    if i==0:
         Cambio_giro_m2.value(1) #giro
         Cambio_giro_m3.value(1) #a la derecha
         u=1
         print("giro der")

    if i==1:
         Cambio_giro_m2.value(0) #giro
         Cambio_giro_m3.value(0) #a la izq
         print("giro izq")
         u=-1

    if i==2:
         Cambio_giro_m2.value(0) #Recto
         Cambio_giro_m3.value(1) #Recto
         print("...")
         u=0


    if Min > m_previous or Min < epsilon:
        condic = -1
    else:
        condic=1


    #----Updating with the minimum prediction
    Z = [X_k, u, Delta_t]
    [x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
    #print(x_k_1, y_k_1, tita_k_1)

    global X
    X = [x_k_1, y_k_1, tita_k_1]

    global m_prev
    m_prev = Min

    #print(m,m_previous,condic,u)

    return X,m_prev,condic

Motor_2.freq(abs(150))
Motor_3.freq(abs(150))

condicion = 1

X_k = [10, 0, math.pi ] #10 son la marca
m_previous = 1e8
Dubins(X_k,m_previous)
n=0


while condicion > 0:
    Dubins(X,m_prev)
    condicion=condic
```

10 Anexo III: Pines de la Raspberry Pi Pico W



11 Anexo IV: Código de cargador de baterías.

```
// Llamo a las librerías
#include <PID_v1.h>  // Librería PID
#include <LiquidCrystal.h> // Libreria Display

// Declaro las variables, defino parámetros y pines
const int PWM = 5; // pin 5 esta la salida pwm
const int voltageSense = A0; // pin donde mido tensión
const int currentSense = A1; // pin de lectura de corriente
const int buzzerPin = A4; // pin buzzer
const double setpointVoltage = 18; // Tensión deseada para la bateria
const double kpVoltage = 7.3; // Ganancia proporcional del PID de
tensión
const double kiVoltage = 3.2; // Ganancia integral del PID de tensión
const double kdVoltage = 0; // Ganancia derivativa del PID de tensión

const double setpointCurrent_max = 1.5; // Corriente deseada para la
carga en amperios
const double setpointCurrent_min = 0.2;
const double kpCurrent = 0.16; // Ganancia proporcional del PID de
corriente
const double kiCurrent = 0.18; // Ganancia integral del PID de
corriente
const double kdCurrent = 0; // Ganancia derivativa del PID de
corriente

double voltage, current;
double outputVoltage, outputCurrent;

PID voltagePID(&voltage, &outputVoltage, &setpointVoltage, kpVoltage,
kiVoltage, kdVoltage, DIRECT);
PID currentPID(&current, &outputCurrent, &setpointCurrent_max,
kpCurrent, kiCurrent, kdCurrent, DIRECT);

//DISPLAY
LiquidCrystal lcd(8, A3, 0, 1, 2, 4); // Initialize the LCD with the
pins you've connected
void setup() {
              pinMode(PWM, OUTPUT);
              // Configura aquí las entradas analógicas o el hardware
necesario para la medición de tensión y corriente
              voltagePID.SetMode(AUTOMATIC);
              currentPID.SetMode(AUTOMATIC);

              pinMode(buzzerPin, OUTPUT);
              // DISPLAY.
              lcd.begin(16, 2); // Set the LCD to 16x2 characters
              lcd.setCursor(0, 0); // Set the cursor to the top-left
corner
              lcd.print("Voltage: ");
              lcd.setCursor(0, 1); // Set the cursor to the second
line
              lcd.print("Current: ");
              }

void loop() {
```

```
            // Lee la tensión de la batería y la corriente de carga
            voltage = (double)analogRead(voltageSense) * (5.0 /
1023.0); // Conversión de la lectura analógica a voltios
            current = (double)analogRead(currentSense) * (5.0 /
1023.0); // Conversión de la lectura analógica a amper
            voltage = voltage * 10.0;  // Divisor Resistivo
            current = current * 20.0; // Adapto en base a la medición
del ACS712

           //Programa a CCCV
            while(voltage < setpointVoltage) {
                                                currentPID.Compute();
                                                // la corriente tiene
que ser igual a 3 A constante
                                                int pwmValue =
constrain(outputCurrent, 0, 255);
                                                // Ajusta el PWM para
controlar la corriente de carga
                                                // Asegúrate de que el
valor esté en el rango correcto
                                                analogWrite(PWM,
pwmValue);
                                                // Lee la tensión de
la batería y la corriente de carga
                                                voltage =
(double)analogRead(voltageSense) * (5.0 / 1023.0); // Conversión de la
lectura analógica a voltios
                                                current =
(double)analogRead(currentSense) * 5.0 / 1023.0; // Conversión de la
lectura analógica a amper
                                                voltage = voltage *
10.0;  // Divisor Resistivo
                                                current = current *
20.0; // Adapto en base a la medición del ACS712

display(current,voltage);
                                              }
            while (current> setpointCurrent_min){

voltagePID.Compute();
                                                  int pwmValue =
constrain(outputVoltage, 0, 255);
                                                  // Ajusta el PWM
para controlar la corriente de carga
                                                  // Asegúrate de que
el valor esté en el rango correcto
                                                  analogWrite(PWM,
pwmValue);
                                                  // Lee la tensión de
la batería y la corriente de carga
                                                  voltage =
(double)analogRead(voltageSense) * (5.0 / 1023.0); // Conversión de la
lectura analógica a voltios
                                                  current =
(double)analogRead(currentSense) * 5.0 / 1023.0; // Conversión de la
lectura analógica a amper
```

```
                                                    voltage = voltage *
10.0;  // Divisor Resistivo
                                                    current = current *
20.0; // Adapto en base a la medición del ACS712

display(current,voltage);
                                                    }
            while (1){
                        digitalWrite(buzzerPin,HIGH);
                        delay(1000);
                        digitalWrite(buzzerPin,LOW);
                        delay(1000);
                        voltage = (double)analogRead(voltageSense) *
(5.0 / 1023.0); // Conversión de la lectura analógica a voltios
                        current = (double)analogRead(currentSense) * 5.0
/ 1023.0; // Conversión de la lectura analógica a amper
                        voltage = voltage * 10.0;  // Divisor Resistivo
                        current = current * 20.0; // Adapto en base a la
medición del ACS712
                        display(current,voltage);
                        }

             }
void display(double Corriente, double Tension) {
            // Display the values on the LCD
            lcd.setCursor(9, 0); // Position the cursor after
"Voltage: "
            lcd.print(Tension, 2); // Display voltage with 2 decimal
places

            lcd.setCursor(9, 1); // Position the cursor after
"Current: "
            lcd.print(Corriente, 2); // Display current with 2 decimal
places

            delay(1000); // Update the display every second
           }
```

12 Anexo V: Algoritmo completo

```
from imu import MPU6050
from machine import I2C, Pin, PWM
import math
import time

#------------------MOTORS-------------------------
"-------Motores Stepper     Movilidad--------"

Motor_2 = PWM(Pin(13,Pin.OUT))    #Pin 17 del Micro
Motor_3 = PWM(Pin(16,Pin.OUT))    #Pin 21 del Micro

Motor_2.duty_u16(32768)  #Duty entre [0,65535] derecha
Motor_3.duty_u16(32768)  #Duty entre [0,65535] izquierda

#Defino cambio de giro
Cambio_giro_m2 = Pin(12,Pin.OUT) #Pin 16 del Micro
Cambio_giro_m3 = Pin(17,Pin.OUT) #Pin 22 del Micro

" -------Motor DC     Estabilidad--------"
Cambio_giro_m11 = Pin(4,Pin.OUT)
Cambio_giro_m12 = Pin(5,Pin.OUT)
Motor_1 = PWM(Pin(9,Pin.OUT))
Motor_1.duty_u16(8000)
Motor_1.freq(300)


i2c=I2C(1, scl=Pin(3), sda=Pin(2), freq=400000)
mpu = MPU6050(i2c)

rollG=0
pitchG=0

rollComp=0
pitchComp=0

yaw=0
tLoop=0
cnt=0

errorP=0
errorR=0
rangpos= 4
rangneg=0.1
k=10000

def Dubins(X_k,m_previous):

    global condic

    Delta_t = 0.001
    epsilon = 0.001

    u = 1 #Decisión de girar para algún lado
    Z = [X_k, u, Delta_t]
    [x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
```

```
m1 = math.sqrt(x_k_1**2 + y_k_1**2)  # Euclidean norm of (x, y)

u = -1  # Decisión de girar para otro lado
Z = [X_k, u, Delta_t]
[x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
m2 = math.sqrt(x_k_1**2 + y_k_1**2)

u = 0  # Decisión de ir en Linea Recta
Z = [X_k, u, Delta_t]
[x_k_1, y_k_1, tita_k_1]= Dinamica_Dubins(Z)
m3 = math.sqrt(x_k_1**2 + y_k_1**2)


#print(m1,m2,m3)

#--------Searching for the minimum-----
m = [m1, m2, m3]
#print(m1,m2,m3)
if m[0]<m[1]:
    Min=m[0]
    i=0
else:
    Min=m[1]
    i=1

if Min > m[2]:
    Min=m[2]
    i=2

if i==0:
     Cambio_giro_m2.value(1) #giro
     Cambio_giro_m3.value(1) #a la derecha
     u=1
     print("giro der")

if i==1:
     Cambio_giro_m2.value(0) #giro
     Cambio_giro_m3.value(0) #a la izq
     print("giro izq")
     u=-1

if i==2:
     Cambio_giro_m2.value(0) #Recto
     Cambio_giro_m3.value(1) #Recto
     print("...")
     u=0


if Min > m_previous or Min < epsilon:
    condic = -1
else:
    condic=1


#----Updating with the minimum prediction
Z = [X_k, u, Delta_t]
[x_k_1, y_k_1, tita_k_1] = Dinamica_Dubins(Z)
```

```
    #print(x_k_1, y_k_1, tita_k_1)

    global X
    X = [x_k_1, y_k_1, tita_k_1]

    global m_prev
    m_prev = Min

    #print(m,m_previous,condic,u)

    return X,m_prev,condic

Motor_2.freq(abs(150))
Motor_3.freq(abs(150))

condicion = 1

X_k = [10, 0, math.pi ] #10 son la marca
m_previous = 1e8
Dubins(X_k,m_previous)
n=0

while True:
    # Estabilidad
    tStart=time.ticks_ms()

    "Empieza la medición con el MPU "
    xGyro=mpu.gyro.x
    yGyro=-mpu.gyro.y
    zGyro=mpu.gyro.z

    xAccel=mpu.accel.x
    yAccel=mpu.accel.y
    zAccel=mpu.accel.z

    rollA=(math.atan(xAccel/zAccel)/2/math.pi*360)+7
    rollComp= rollA*.005 + .995*(rollComp+yGyro*tLoop)+errorR*.002
    errorR= errorR+(rollA-rollComp)*tLoop

    cnt=cnt+1

    if cnt==10:
        cnt=0

    " Condición para mover pendulo de un lado a otro y mantener el
equilibrio"

    if rollComp>rangpos:
        Cambio_giro_m11.value(0) #
        Cambio_giro_m12.value(0)
    elif rangpos >rollComp >0:
        Cambio_giro_m11.value(0) # Cabecea para delante
        Cambio_giro_m12.value(1) #
        #print("Adelante")
    elif rollComp<-rangneg:
        Cambio_giro_m11.value(1) # Cabecea hacia atras
        Cambio_giro_m12.value(0) #
```

```
    #print("Atras")
else:
    Cambio_giro_m11.value(0) #
    Cambio_giro_m12.value(0)
    #print("Equil")

u = k * rollComp
Motor_1.duty_u16(abs(int(u)))

# Movilidad
tStop = time.ticks_ms()
tLoop = (tStop-tStart)*0.001
#print(n)
n=n+1

if condicion > 0 and n >5:
    Dubins(X,m_prev)
    condicion = condic
    n=0
```

13 Anexo VI: Información adicional al capítulo Electrónica.

13.1 IR2104

Información adicional sobre el datasheet de dicho driver utilizado, para más info.

International
IOR Rectifier

Data Sheet No. PD60046-S

IR2104(S)&(PbF)

HALF-BRIDGE DRIVER

Features

- Floating channel designed for bootstrap operation
  Fully operational to +600V
  Tolerant to negative transient voltage
  dV/dt immune
- Gate drive supply range from 10 to 20V
- Undervoltage lockout
- 3.3V, 5V and 15V input logic compatible
- Cross-conduction prevention logic
- Internally set deadtime
- High side output in phase with input
- Shut down input turns off both channels
- Matched propagation delay for both channels
- Also available LEAD-FREE

Product Summary

V_{OFFSET}	600V max.
I_O+/-	130 mA / 270 mA
V_{OUT}	10 - 20V
$t_{on/off}$ (typ.)	680 & 150 ns
Deadtime (typ.)	520 ns

Packages

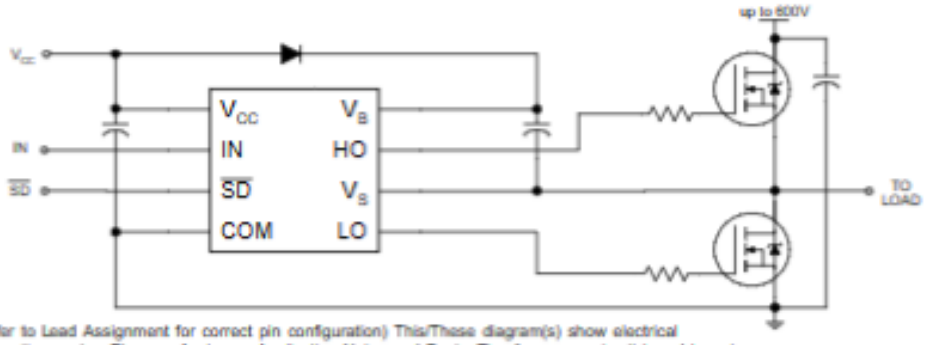
8 Lead SOIC
IR2104S

8 Lead PDIP
IR2104

Description

The IR2104(S) are high voltage, high speed power MOSFET and IGBT drivers with dependent high and low side referenced output channels. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. The logic input is compatible with standard CMOS or LSTTL output, down to 3.3V logic. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates from 10 to 600 volts.

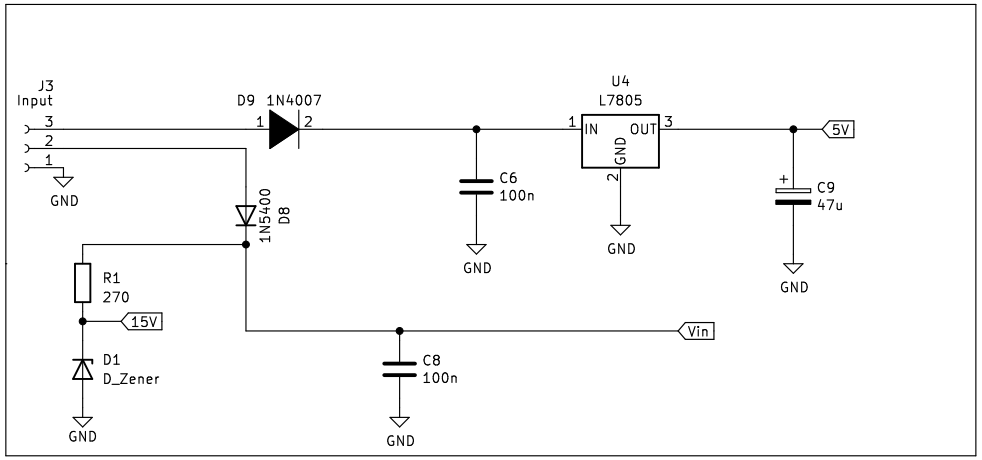
Typical Connection

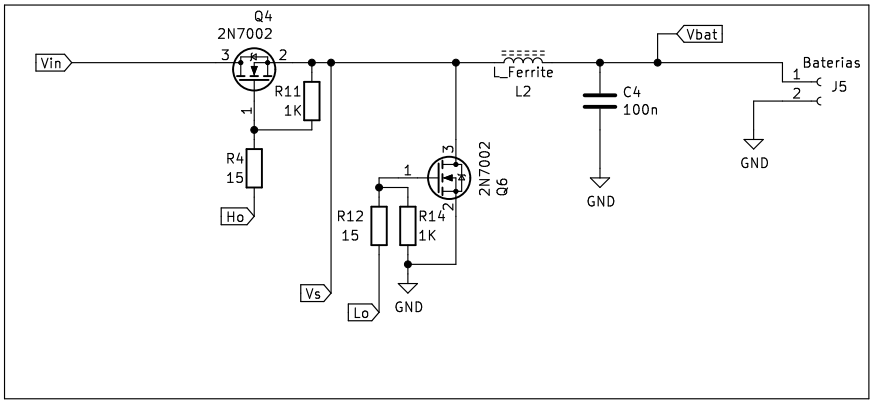


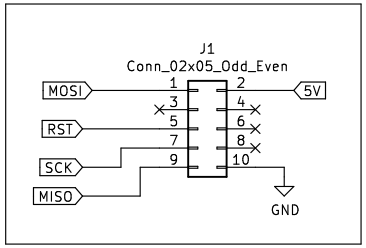
(Refer to Lead Assignment for correct pin configuration) This/These diagram(s) show electrical connections only. Please refer to our Application Notes and DesignTips for proper circuit board layout.

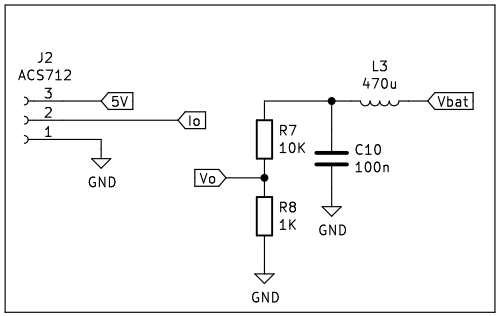
13.2 Esquemático de PCB en KiCad

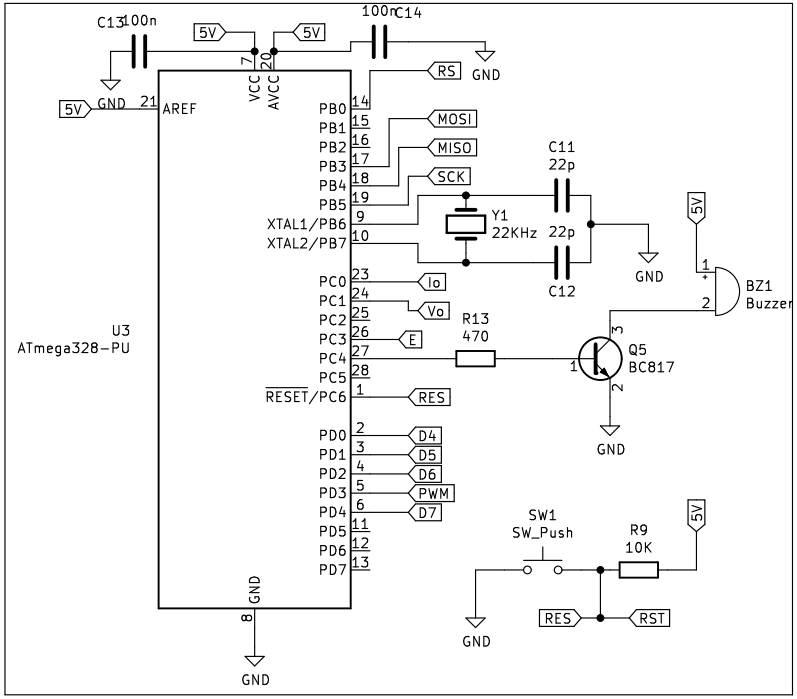
Se adjunta ambas representaciones del diseño de la placa del cargador de baterías.

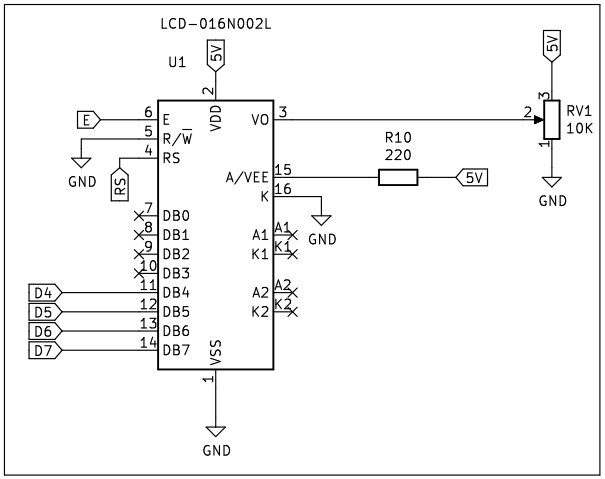


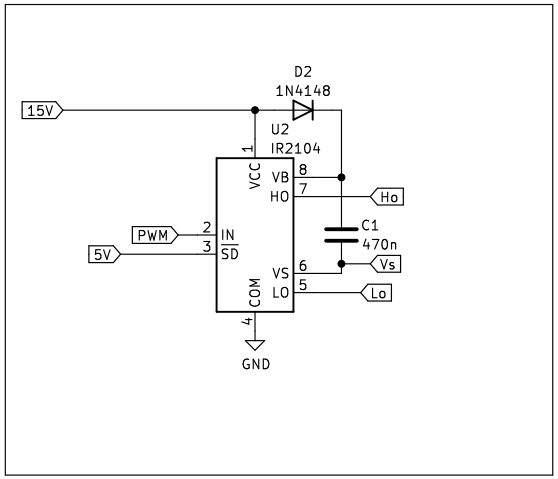












Sheet: /
File: CargadorModificado.kicad_sch

Title: CargadorBaterias

Size: A4	Date:	Rev:
KiCad E.D.A. kicad 7.0.2		Id: 1/1