UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CONTROL DE UN PÉNDULO INVERTIDO SOBRE DOS RUEDAS DE TRES GRADOS DE LIBERTAD

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO MECATRÓNICO

P R E S E N T A:

HUERTA GIL RUBEN DANIEL

DIRECTOR DE TESIS:

M.I. Serafín Castañeda Cedeño

MÉXICO, D.F. 2014

Agradecimientos

A mi madre

Porque desde mis primeros pasos en la escuela siempre estuviste a mi lado apoyándome cuando más lo necesitaba, recuerdo en la primaria cuando por las tardes me ayudabas a estudiar y hasta en la universidad siempre querías ir a conseguirme libros y material que facilitaran mis trabajos. Junto con mi papa son los pilares que sostienen y representan mi formación como persona íntegra y profesional.

A mi padre

Porque tuvo que realizar el sacrificio más grande, para que yo pudiera tener las comodidades y una buena educación, todos los días te extraño y ojalá hubieras estado en esta etapa de mi vida. Por siempre te recordaré y amaré.

A mi hermana

Siempre puedo contar contigo cuando tengo problemas y me ayudas a encontrarle solución, recuerda que siempre puedes contar conmigo.

A mis profesores

Gracias a todos los profesores que me dieron clase, pues gracias a los conocimientos que me aportaron podré ser el ingeniero que siempre quise ser. Gracias a mis sinodales por aceptar evaluarme y muchas gracias a mi asesor de tesis por guiarme y auxiliarme en los problemas con los que me topaba.

A mis amigos

Sin su apoyo durante este trayecto de mi vida, hubiera sido mucho más complicado pasar todas las materias de la carrera esperando volverlos a ver en un futuro y deseándoles lo mejor.

A mi Facultad y Universidad

Por permitirme ser parte de esta maravillosa institución de gran prestigio y de las muchas oportunidades que me brindó durante esta etapa de mi vida, siempre orgulloso de ser universitario.

Índice General

Índice de Figuras

1 INTRODUCCIÓN

En el presente trabajo se informa sobre el desarrollo del diseño físico de un Péndulo Invertido Sobre Dos Ruedas conocido como TWIP de sus siglas en inglés "*Two-Wheeled Inverted Pendulum"*, y pretende ser un trabajo de consulta que permita a estudiantes de ingeniería mecatrónica a conocer el modelo de un péndulo invertido sobre dos ruedas, el proceso básico para el diseño de un prototipo funcional así como del proceso a seguir para el diseño de un controlador.
Éste primer capítulo tiene como objetivo dar a conocer al lector el estado del arte de los péndulos invertidos sobre dos ruedas así como de algunas aplicaciones que se le han dado a lo largo de los años.

En el segundo capítulo encontraremos las ecuaciones que describen el comportamiento del péndulo invertido sobre dos ruedas y cómo obtener el modelo lineal en espacio de estados basándonos en esas ecuaciones, para después diseñar el controlador. En este capítulo también encontraremos el proceso de diseño que se siguió, pasando por el diseño conceptual y el diseño de configuración para poder definir el prototipo final del péndulo, y así poder realizar la identificación paramétrica que requiere el modelo matemático.

El tercer capítulo trata sobre el diseño del controlador, dónde se definen los objetivos de control, la instrumentación, la selección del controlador LQR de sus siglas en inglés "Linear-Quadratic Regulator" de acuerdo a los objetivos de control y características físicas para estabilizar el péndulo y que sea capaz de moverse en sus ejes, el análisis de estabilidad y controlabilidad del sistema así como de las simulaciones realizadas a los modelos no lineal y lineal en lazo abierto y con control.

En el cuarto capítulo se presentan los resultados de las pruebas realizadas a los controladores.

Finalmente en el quinto capítulo se presentan las conclusiones de este trabajo.

1.1 OBJETIVOS Y ALCANCES DEL PROYECTO

- Diseñar y modelar un péndulo invertido sobre dos ruedas.
- Identificar los parámetros del sistema.
- Diseñar el controlador que le permita al péndulo moverse en el plano mientras mantenga su posición vertical.

1.2 ESTADO DEL ARTE

Dada su dinámica inestable, los péndulos invertidos rara vez derivan en productos útiles, sin embargo su dinámica y control han sido bien estudiadas por los ingenieros.

En 1908, A. Stephenson examinó el péndulo invertido y demostró que podía ser estabilizado al aplicar oscilaciones armónicas, rápidas y verticales a su base [8].

En 1909 desarrolló condiciones de estabilidad para péndulos invertidos dobles y triples [8].

En 1932, E. R. Lowenstern desarrolló ecuaciones de movimiento general para los péndulos invertidos, las cuales se relacionaban con el trabajo de Stephenson [8].

Para los años 60´s la dinámica del péndulo invertido ya era bien conocida y varios artículos de esta época trataban sobre soluciones analíticas y aproximadas sobre como respondería ante distintas entradas. Estas entradas eran por lo general señales sinusoidales, aleatorias o impulsos [8].

En los años 80´s los artículos presentaban controladores que podían estabilizar el péndulo invertido para una mayor variedad de entradas como señales parabólicas o de diente de sierra [8].

M. Sahba utilizó un algoritmo de optimización para diseñar un servocontrolador en lugar de utilizar método usual de diseño de controladores por linealización del sistema. Después T. Yamakawa utilizó un controlador por lógica difusa con un procesador de gran poder con ese mismo propósito [8].

Hasta nuestros días el péndulo invertido en un carro representa una plataforma de pruebas para validar nuevas teorías de control.

1.2.1 Péndulo Invertido Sobre Dos Ruedas - *TWIP (Two-Wheeled Inverted Pendulum)*

La investigación de los robots que funcionan bajo el principio del péndulo invertido sobre dos ruedas ha ganado gran importancia a lo largo de la última década entre investigadores y entusiastas de la robótica, esto gracias a su dinámica inherentemente inestable y esto se demuestra debido a la gran cantidad de artículos publicados referentes a este tipo de sistemas [1-22].

Estos robots se caracterizan por su habilidad para mantener el equilibrio en tan solo dos ruedas y por poder girar en torno a su propio centro de masa, permitiéndole obtener grandes características de maniobrabilidad como moverse fácilmente por varios tipos de terreno, dar vueltas muy cerradas, sortear pequeños escalones y bordes, ocupando menor espacio.

Estas capacidades tienen el potencial de resolver gran número de retos de la industria y de la sociedad, por ejemplo la implementación de este sistema en sillas de ruedas robóticas daría al usuario mejor manejo y por tanto podría acceder a lugares que antes no podía, robots humanoides que caminan, sistemas de transporte personal e incluso en la industria militar para el control de misiles o cohetes.

En 1988 se dio uno de los primeros reportes sobre la implementación de un péndulo invertido en dos ruedas, por T. Kanamura.

Para 1996, Ha, Y. S. y Yuta desarrollaron un robot autónomo basado en un péndulo invertido de dos ruedas llamado “Yamabico Kurara”. Ese mismo año N. Shiroma diseñó robots similares que se coordinaban uno con el otro para poder cargar pesos cooperativamente.

En el año 2000, F. Ding diseño una plataforma basada en un TWIP que funcionara como un medio de transporte personal con asiento ajustable que mantuviera al usuario siempre en una postura vertical.

En 2001 la aplicación comercial más conocida del péndulo invertido sobre dos ruedas es revelada, el "SEGWAY", inventado por Dean Kamen; el SEGWAY es capaz de mantener el equilibrio de una persona que se encuentra parada sobre su plataforma, esta es una versión más complicada de un péndulo invertido ya que contiene una dinámica de tiempo variable incierta. Este dispositivo innovador cuenta con cinco giroscopios y toda una colección de inclinómetros para mantener al SEGWAY en su posición vertical, aunque solo son necesarios tres giroscopios para que el sistema funcione, los demás sensores son meramente de precaución.

Del dispositivo anterior se han derivado varios robots móviles de dos ruedas capaces de mantener su equilibrio, desarrollados por entusiastas de la robótica alrededor de todo el mundo.

En el 2002, en el Laboratorio de Electrónica Industrial del Instituto Federal Suizo de Tecnología, Felix Grasser construyó a "JOE" un TWIP que tiene pesas para simular a un operador, contiene un controlador por espacio de estados lineal el cual se auxilia de un giroscopio y encoders en los motores para estabilizar el sistema.

Figura 1.1: JOE **[12]**

En 2003, dos robots similares a JOE fueron construidos, el nBot, construido por David P. Anderson, el cual utiliza sensores inerciales comerciales y un sistema de posicionamiento por medio de los encoders de los motores y el LegWay por Steve Hassenplug quien utilizó el kit de robótica LEGO Mindstorms.

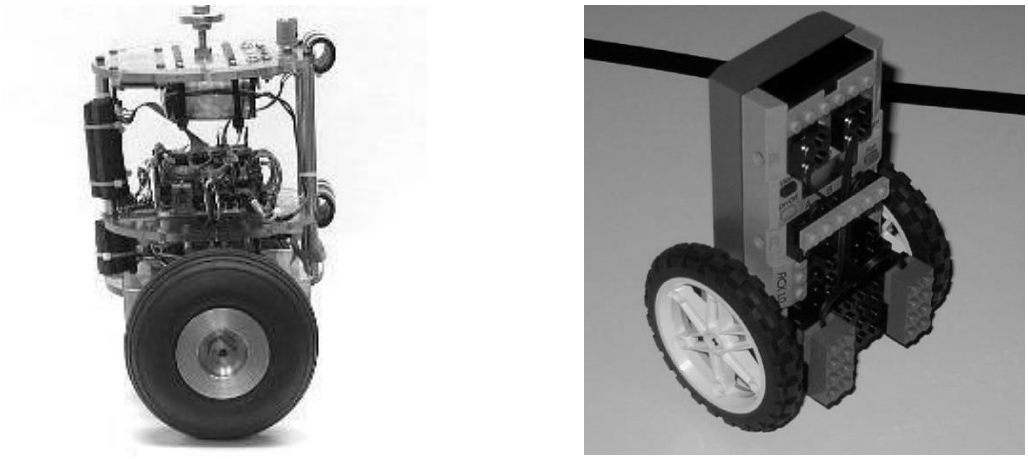


Figura 1.2: nBot (izquierda) - LegWay (derecha) [23,24]

También en el 2003, Bill Sherman construyó el Balibot, Ooi fue presentado como proyecto final de carrera en la Escuela de Ingeniería Mecánica de la Universidad del Oeste de Australia y T. H. Bui desarrolló un robot soldador móvil donde la punta soldadora está montada sobre un TWIP que puede seguir trayectorias específicas a soldar.

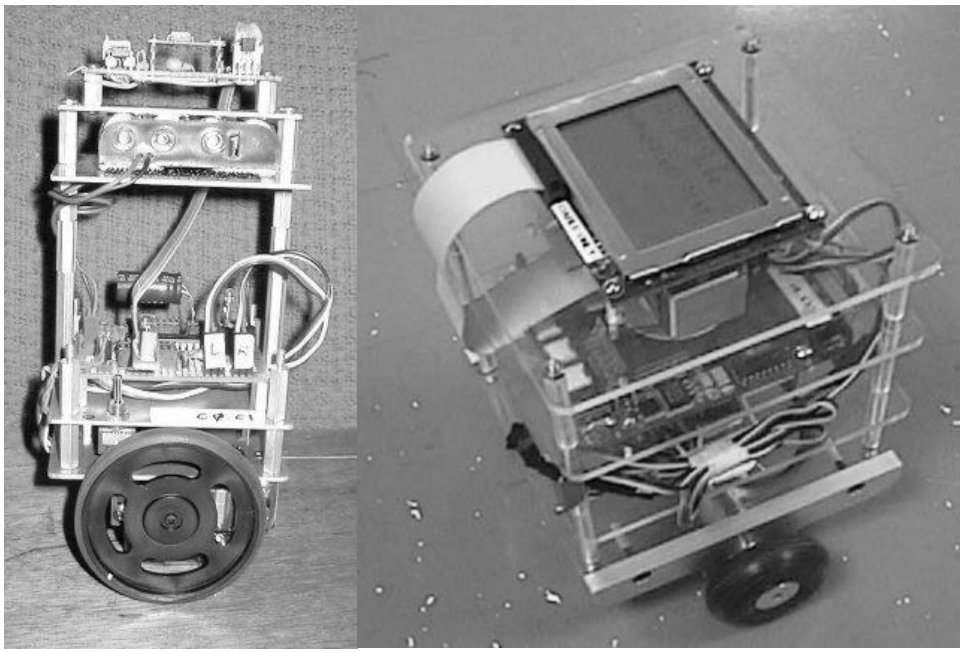


Figura 1.3: Balibot (izquierda) - Ooi (derecha) [25, 3]

En 2004, Sasaki y T. Matsumoto diseñaron un robot parecido al JOE pero sin un stick de dirección, la dirección era lograda por medio de inclinarse hacia algún lado.

En 2007, J. Li propuso un vehículo que pudiera funcionar tanto para llevar a una persona como llevar objetos en él.

También surgieron algunas versiones del LegWay con la nueva plataforma de robótica de LEGO, el NXT Mindstorms. Hurbain construyo el NXTway, y Ryo Watanabe creó una versión mejorada, el NXTway-G.

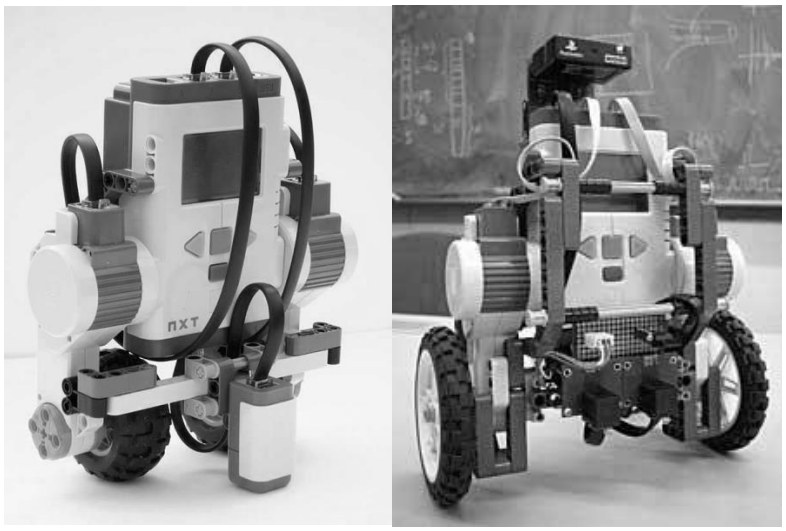


Figura 1.4: NXTway (izquierda) - NXTway-G (derecha) [26]

En 2008, surge GBOT de Googol Technology (HK) Ltd., un péndulo invertido comercial que permite hacer experimentos integrando un espacio de trabajo en tiempo real con el software MATLAB®.

En este mismo año Yorihisa Yamamoto desarrolla el NXTway-GS, el cual incluye el modelado por medio del método de Lagrange además de poderse maniobrar por medio de control remoto.

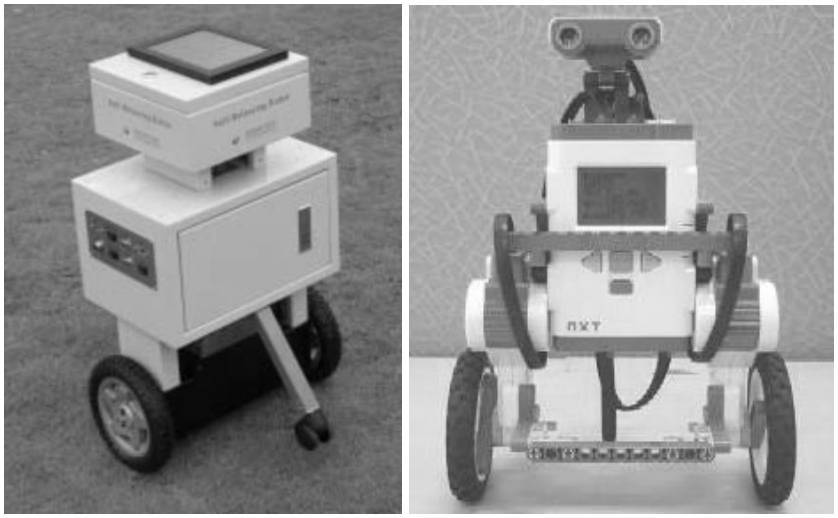


Figura 1.5: GBOT (izquierda) - NXTway-GS (derecha) [2, 1]

En 2009, se propuso un robot similar que funcionara como transportador de equipaje que pudiera seguir trayectorias predefinidas por T. Takei, R. lmamura y S. Yuta y un prototipo HTV (Human Transportation Vehicle) por S.-C. Lin y C.-C. Tsai [8].

En años recientes las plataformas móviles TWIP han sido propuestas como alternativas para los robots humanoides bípedos debido a su mejor movilidad y una dinámica mucho más sencilla.

Aunque existe gran número de trabajos sobre investigación en este campo tan solo unos cuantos han sido implementados como vehículos de transporte humano o como vehículos de carga.

Nos basaremos en las configuraciones de los péndulos sobre dos ruedas que se observan en las imágenes anteriores para el modelo matemático.

2 MODELADO DEL PÉNDULO Y PARAMETRIZACIÓN

2.1 MODELO MATEMÁTICO

No se realizó el modelo desde cero, ya que en la literatura existen varios modelos que resuelven el problema del péndulo invertido sobre dos ruedas. Se revisaron varios de estos modelos en los artículos investigados, seleccionando el modelo realizado por Yorihisa Yamamoto [1], el cual usa el método de Lagrange para encontrar las ecuaciones que modelan el sistema.

Partiendo de un modelo básico para el robot:

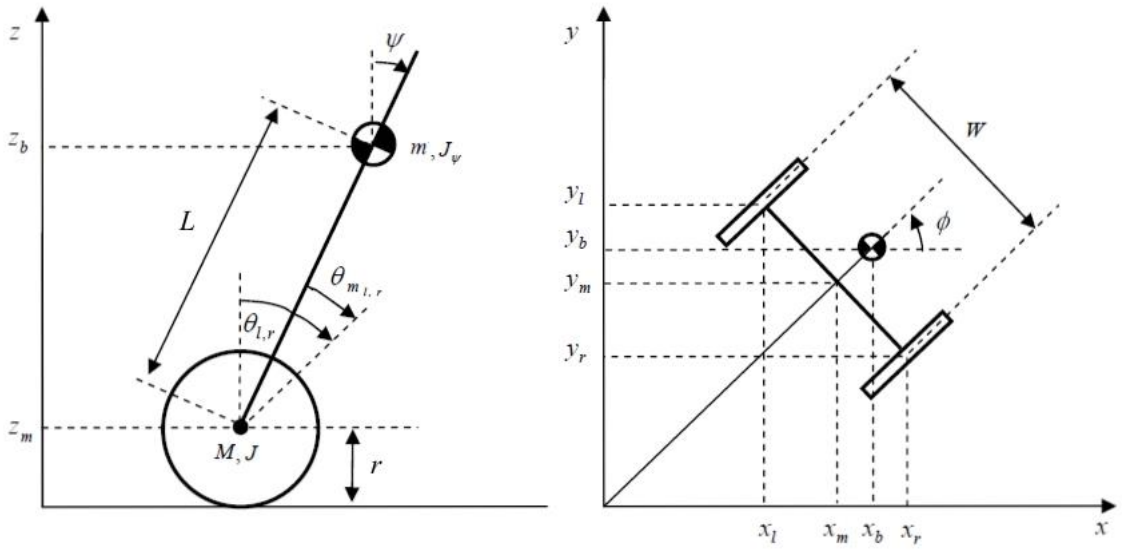


Figura 2.1: Modelo TWIP

Donde:

ψ: ángulo Pitch.
$\theta_{l,r}$: ángulo de la llanta (L y R representan izquierda y derecha respectivamente).
$\theta_{m\,l,r}$: ángulo del motor de CD.
$\emptyset$: ángulo Yaw.

2.1.1 Ecuaciones De Movimiento

Basándonos en el sistema coordenado de la Figura 2.1 se pueden deducir las siguientes ecuaciones:

$$\theta = \frac{1}{2}(\theta_l + \theta_r) \quad (1)$$

$$\dot{\theta} = \frac{1}{2}(\dot{\theta}_l + \dot{\theta}_r) \quad (1.1)$$

$$\dot{\theta}^2 = \frac{1}{4}\left(\dot{\theta}_l{}^2 + 2\dot{\theta}_l\dot{\theta}_r + \dot{\theta}_r{}^2\right) \quad (1.2)$$

$$\emptyset = \frac{r}{W}(\theta_r - \theta_l) \quad (2)$$

$$\dot{\emptyset} = \frac{r}{W}\left(\dot{\theta}_r - \dot{\theta}_l\right) \tag{2.1}$$

$$\dot{\emptyset}^2 = \frac{r^2}{W^2}\left(\dot{\theta}_r{}^2 - 2\dot{\theta}_r\dot{\theta}_l + \dot{\theta}_l{}^2\right) \tag{2.2}$$

$$z_m = r \tag{3}$$

$$\dot{x}_m = r\dot{\theta}\cos\emptyset \tag{4}$$

$$\dot{y}_m = r\dot{\theta}\sin\emptyset \tag{5}$$

$$\dot{z}_m = 0 \tag{6}$$

$$x_l = x_m - \frac{W}{2}\sin\emptyset \tag{7}$$

$$\dot{x}_l = \dot{x}_m - \frac{W}{2}\dot{\emptyset}\cos\emptyset \tag{8}$$

$$y_l = y_m + \frac{W}{2}\cos\emptyset \tag{9}$$

$$\dot{y}_l = \dot{y}_m - \frac{W}{2}\dot{\emptyset}\sin\emptyset \tag{10}$$

$$z_l = z_m = r \tag{11}$$

$$\dot{z}_l = \dot{z}_m = 0 \tag{12}$$

$$x_r = x_m + \frac{W}{2}\sin\emptyset \tag{13}$$

$$\dot{x}_r = \dot{x}_m + \frac{W}{2}\dot{\emptyset}\cos\emptyset \tag{14}$$

$$y_r = y_m - \frac{W}{2}\cos\emptyset \tag{15}$$

$$\dot{y}_r = \dot{y}_m + \frac{W}{2}\dot{\emptyset}\sin\emptyset \tag{16}$$

$$z_r = z_m = r \tag{17}$$

$$\dot{z}_r = \dot{z}_m = 0 \tag{18}$$

$$x_b = x_m + L\sin\psi\cos\emptyset \tag{19}$$

$$\dot{x}_b = \dot{x}_m + L\dot{\psi}\cos\psi\cos\emptyset - L\dot{\emptyset}\sin\psi\sin\emptyset \tag{20}$$

$$y_b = y_m + L\sin\psi\sin\emptyset \tag{21}$$

$$\dot{y}_b = \dot{y}_m + L\dot{\emptyset}\sin\psi\cos\emptyset + L\dot{\psi}\cos\psi\sin\emptyset \tag{22}$$

$$z_b = z_m + L\cos\psi \tag{23}$$

$$\dot{z}_b = \dot{z}_m - L\dot{\psi}\sin\psi \tag{24}$$

2.1.2 Modelo No Lineal

La energía cinética traslacional es:

$$EC_1 = \frac{1}{2}M\left(\dot{x_l}^2 + \dot{y_l}^2 + \dot{z_l}^2\right) + \frac{1}{2}M\left(\dot{x_r}^2 + \dot{y_r}^2 + \dot{z_r}^2\right) + \frac{1}{2}m(\dot{x_b}^2 + \dot{y_b}^2 + \dot{z_b}^2) \tag{25}$$

Sustituyendo las ecuaciones 8, 10, 12, 14, 16, 18, 20, 22, 24 y 6 en la ecuación 25:

$$\begin{aligned} EC_1 = {} & M\dot{x_m}^2 + M\dot{y_m}^2 + \frac{1}{4}MW^2\dot{\phi}^2\cos^2\phi + \frac{1}{4}MW^2\dot{\phi}^2\sin^2\phi + \frac{1}{2}m\dot{x_m}^2 \\ & + \frac{1}{2}m\dot{y_m}^2 + m\dot{x}_m L\dot{\psi}\cos\psi\cos\phi - m\dot{x}_m L\dot{\phi}\sin\psi\sin\phi \\ & + \frac{1}{2}mL^2\dot{\psi}^2\cos^2\psi\cos^2\phi + \frac{1}{2}mL^2\dot{\psi}^2\cos^2\psi\sin^2\phi \\ & + \frac{1}{2}mL^2\dot{\phi}^2\sin^2\psi\sin^2\phi + \frac{1}{2}mL^2\dot{\phi}^2\sin^2\psi\cos^2\phi \\ & + m\dot{y}_m L\dot{\phi}\sin\psi\cos\phi - m\dot{y}_m L\dot{\psi}\cos\psi\sin\phi + \frac{1}{2}mL^2\dot{\psi}^2\sin^2\psi \end{aligned} \tag{26}$$

Simplificando y agrupando obtenemos:

$$\begin{aligned} EC_1 = {} & M\left(\dot{x_m}^2 + \dot{y_m}^2 + \frac{1}{4}W^2\dot{\phi}^2\right) \\ & + \frac{1}{2}m\left(\dot{x_m}^2 + \dot{y_m}^2 + L^2\dot{\psi}^2 + L^2\dot{\phi}^2\sin^2\psi\right. \\ & + 2\dot{x}_m L\left(\dot{\psi}\cos\psi\cos\phi - \dot{\phi}\sin\psi\sin\phi\right) \\ & \left. + 2\dot{y}_m L\left(\dot{\phi}\sin\psi\cos\phi - \dot{\psi}\cos\psi\sin\phi\right)\right) \end{aligned} \tag{27}$$

Sustituyendo las ecuaciones 4 y 5 en la ecuación 27:

$$\begin{aligned} EC_1 = {} & M\left(r^2\dot{\theta}^2\cos^2\phi + r^2\dot{\theta}^2\sin^2\phi + \frac{1}{4}W^2\dot{\phi}^2\right) \\ & + \frac{1}{2}m\left(r^2\dot{\theta}^2\cos^2\phi + r^2\dot{\theta}^2\sin^2\phi + L^2\dot{\psi}^2 + L^2\dot{\phi}^2\sin^2\psi\right. \\ & + 2r\dot{\theta}\cos\phi\, L\left(\dot{\psi}\cos\psi\cos\phi - \dot{\phi}\sin\psi\sin\phi\right) \\ & \left. + 2r\dot{\theta}\sin\phi\, L\left(\dot{\phi}\sin\psi\cos\phi - \dot{\psi}\cos\psi\sin\phi\right)\right) \end{aligned} \tag{28}$$

Simplificando:

$$EC_1 = M\left(r^2\dot{\theta}^2 + \frac{1}{4}W^2\dot{\phi}^2\right) + \frac{1}{2}m\left(r^2\dot{\theta}^2 + L^2\dot{\psi}^2 + L^2\dot{\phi}^2\sin^2\psi + 2r\dot{\theta}L\dot{\psi}\cos\psi\right) \tag{29}$$

Expandiendo la ecuación 29:

$$\begin{aligned} EC_1 = {} & Mr^2\dot{\theta}^2 + \frac{1}{4}MW^2\dot{\phi}^2 + \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}mL^2\dot{\psi}^2 + \frac{1}{2}mL^2\dot{\phi}^2\sin^2\psi \\ & + mrL\dot{\theta}\dot{\psi}\cos\psi \end{aligned} \tag{30}$$

La energía cinética rotacional es:

$$EC_2 = \frac{1}{2}J\dot{\theta_l}^2 + \frac{1}{2}J\dot{\theta_r}^2 + \frac{1}{2}I_\psi\dot{\psi}^2 + \frac{1}{2}I_\phi\dot{\phi}^2 \tag{31}$$

Agrupando la ecuación 31:

$$EC_2 = \frac{1}{2}J\left(\dot{\theta_l}^2 + \dot{\theta_r}^2\right) + \frac{1}{2}I_\psi\dot{\psi}^2 + \frac{1}{2}I_\emptyset\dot{\emptyset}^2 \tag{32}$$

Utilizando las ecuaciones 1.2 y 2.2:

$$4\dot{\theta}^2 = \left(\dot{\theta_l}^2 + 2\dot{\theta}_l\dot{\theta}_r + \dot{\theta_r}^2\right)$$

$$\frac{W^2}{r^2}\dot{\emptyset}^2 = \left(\dot{\theta_r}^2 - 2\dot{\theta}_r\dot{\theta}_l + \dot{\theta_l}^2\right)$$

$$4\dot{\theta}^2 + \frac{W^2}{r^2}\dot{\emptyset}^2 = 2(\dot{\theta_l}^2 + \dot{\theta_l}^2)$$

$$2\dot{\theta}^2 + \frac{W^2}{2r^2}\dot{\emptyset}^2 = (\dot{\theta_l}^2 + \dot{\theta_l}^2)$$

Sustituyendo la ecuación anterior en la ecuación 32:

$$EC_2 = \frac{1}{2}J\left(2\dot{\theta}^2 + \frac{W^2}{2r^2}\dot{\emptyset}^2\right) + \frac{1}{2}I_\psi\dot{\psi}^2 + \frac{1}{2}I_\emptyset\dot{\emptyset}^2 \tag{33}$$

Expandiendo la ecuación 33:

$$EC_2 = J\dot{\theta}^2 + \frac{W^2J}{4r^2}\dot{\emptyset}^2 + \frac{1}{2}I_\psi\dot{\psi}^2 + \frac{1}{2}I_\emptyset\dot{\emptyset}^2 \tag{34}$$

La energía cinética total que expresada como $EC_1 + EC_2$, sumando las ecuaciones 30 y 34:

$$\begin{aligned} EC_T = {} & Mr^2\dot{\theta}^2 + \frac{1}{4}MW^2\dot{\emptyset}^2 + \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}mL^2\dot{\psi}^2 + \frac{1}{2}mL^2\dot{\emptyset}^2\sin^2\psi \\ & + mrL\dot{\theta}\dot{\psi}\cos\psi + J\dot{\theta}^2 + \frac{W^2J}{4r^2}\dot{\emptyset}^2 + \frac{1}{2}I_\psi\dot{\psi}^2 + \frac{1}{2}I_\emptyset\dot{\emptyset}^2 \end{aligned} \tag{35}$$

Agrupando términos:

$$\begin{aligned} EC_T = {} & \frac{1}{2}(2Mr^2 + mr^2 + 2J)\dot{\theta}^2 + \frac{1}{2}\left(mL^2 + I_\psi\right)\dot{\psi}^2 + \frac{1}{2}\left(\frac{1}{2}MW^2 + \frac{1}{2}\frac{W^2}{r^2}J + I_\emptyset\right)\dot{\emptyset}^2 \\ & + mrL\dot{\theta}\dot{\psi}\cos\psi + \frac{1}{2}mL^2\dot{\emptyset}^2\sin^2\psi \end{aligned} \tag{36}$$

La energía potencial es:

$$EP = Mgz_l + Mgz_r + mg(z_b - (r + L)) \tag{37}$$

Sustituyendo las ecuaciones 11, 17, 23 y 3 en la ecuación 35:

$$EP = 2Mgr - mgL + mgL\cos\psi \tag{38}$$

Siguiendo la ecuación del Lagrangiano:

$$L = EC_T - EP \tag{39}$$

$$L = \frac{1}{2}(2Mr^2 + mr^2 + 2J)\dot{\theta}^2 + \frac{1}{2}\left(mL^2 + I_\psi\right)\dot{\psi}^2 + \frac{1}{2}\left(\frac{1}{2}MW^2 + \frac{1}{2}\frac{W^2}{r^2}J + I_\emptyset\right)\dot{\emptyset}^2 + mrL\dot{\theta}\dot{\psi}\cos\psi + \frac{1}{2}mL^2\dot{\emptyset}^2\sin^2\psi - 2Mgr + mgL - mgL\cos\psi \quad (40)$$

Determinando las coordenadas generalizadas:

θ: $Promedio\ del\ ángulo\ de\ la\ llanta\ izquierda\ y\ derecha$

ψ: $Ángulo\ respecto\ al\ eje\ Y\ (ángulo\ pitch)$

$\emptyset$: $Ángulo\ respecto\ al\ eje\ Z\ (ángulo\ yaw)$

Las cuales fueron seleccionadas debido a que son la representación de menor orden de las variables que se encuentran en las ecuaciones de energía.

Siguiendo las ecuaciones de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = \mathcal{T}_\theta - 2b_w\dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = \mathcal{T}_\psi$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\emptyset}}\right) - \frac{\partial L}{\partial \emptyset} = \mathcal{T}_\emptyset - \frac{W^2}{2r^2}b_w\dot{\emptyset}$$

Donde las fuerzas generalizadas representadas por la variable $\mathcal{T}$ son los pares totales que sufre el sistema en cada una de sus coordenadas generalizadas y la energía disipada está determinada por las componentes de la fricción entre la rueda y el piso.

Obtenemos las ecuaciones no lineales que representan al sistema [1]:

$$\boldsymbol{\mathcal{T}_\theta = \left((2M + m)r^2 + 2J\right)\ddot{\theta} + 2b_w\dot{\theta} + mrL\cos\psi\,\ddot{\psi} - mrL\dot{\psi}^2\sin\psi} \quad (\mathbf{41})$$

$$\boldsymbol{\mathcal{T}_\psi = mrL\cos\psi\,\ddot{\theta} + \left(mL^2 + I_\psi\right)\ddot{\psi} - mgL\sin\psi - mL^2\dot{\emptyset}^2\sin\psi\cos\psi} \quad (\mathbf{42})$$

$$\boldsymbol{\mathcal{T}_\emptyset = \left(\frac{1}{2}MW^2 + \frac{1}{2}\frac{W^2}{r^2}J + I_\emptyset + mL^2\sin^2\psi\right)\ddot{\emptyset} + 2mL^2\dot{\emptyset}\dot{\psi}\sin\psi\cos\psi + \frac{W^2}{2r^2}b_w\dot{\emptyset}} \quad (\mathbf{43})$$

De la ecuación $(\mathbf{42})$ se puede apreciar que si no existe movimiento en las ruedas $\ddot{\theta} = \dot{\emptyset} = 0$, entonces obtenemos la ecuación correspondiente al modelo del péndulo simple.

2.1.3 Modelo Lineal

Considerando que se va a operar el sistema cerca de su punto de equilibrio ($\psi \to 0$), entonces se linealiza el modelo por medio del método de Serie de Taylor truncada:

$\psi \to 0$
$\sin\psi \to \psi$
$\cos\psi \to 1$
$\dot{\psi}^2 \to 0$
$\dot{\emptyset}^2 \to 0$
$\dot{\emptyset}\dot{\psi} \to 0$

Aplicando las relaciones anteriores en las ecuaciones no lineales del péndulo invertido sobre dos ruedas obtenemos el sistema linealizado:

$$\mathcal{T}_{\theta} = \left((2M+m)r^2 + 2J\right)\ddot{\theta} + 2b_w\dot{\theta} + mrL\ddot{\psi} \tag{44}$$

$$\mathcal{T}_{\psi} = mrL\ddot{\theta} + \left(mL^2 + I_{\psi}\right)\ddot{\psi} - mgL\psi \tag{45}$$

$$\mathcal{T}_{\emptyset} = \left(\frac{1}{2}MW^2 + \frac{1}{2}\frac{W^2}{r^2}J + I_{\emptyset}\right)\ddot{\emptyset} + \frac{W^2}{2r^2}b_w\dot{\emptyset} \tag{46}$$

2.1.4 Modelo De Los Motores De CD

Lo que resta es agregar el modelo de la dinámica de los actuadores que harán funcionar al sistema, que en este caso son dos motores de corriente directa, los cuales se supondrán iguales para facilitar los cálculos.

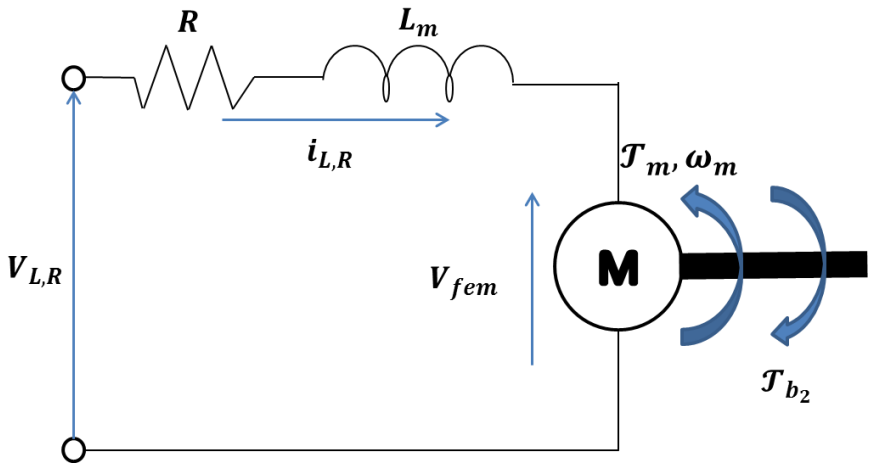


Figura 2.2: Diagrama eléctrico de un motor de CD

$$V_{L,R} - L_m\frac{d}{dt}i_{L,R} - Ri_{L,R} - V_{fem} = 0 \tag{47}$$

$$V_{fem} = K_b\dot{\theta}_m \tag{48}$$

$$\mathcal{T}_m = K_a\, i_{L,R} \tag{49}$$

$$\dot{\theta}_m = \dot{\theta}_{L,R} - \dot{\psi} \tag{50}$$

Si se considera la inductancia del motor despreciable debido a que $L_m \ll R$ y sustituyendo las ecuaciones 48, 49 y 50 en la ecuación 47 obtenemos:

$$i_{L,R} = \frac{V_{L,R} - K_b\left(\dot{\theta}_{L,R} - \dot{\psi}\right)}{R} \tag{51}$$

$$i_L + i_R = \frac{(V_L + V_R) - 2K_b\left(\dot{\theta} - \dot{\psi}\right)}{R} \tag{52}$$

$$i_R - i_L = \frac{(V_R - V_L) - \frac{W}{r}K_b\dot{\phi}}{R} \tag{53}$$

Realizando la suma de pares de acuerdo con la referencia [1]:

$$\mathcal{T}_{L,R} = \mathcal{T}_m - J_m\ddot{\theta}_m - b_2\dot{\theta}_m \tag{54}$$

$$\mathcal{T}_{L,R} = K_a\, i_{L,R} - J_m\ddot{\theta}_m - b_2\left(\dot{\theta}_{L,R} - \dot{\psi}\right) \tag{55}$$

$$\mathcal{T}_\theta = \mathcal{T}_L + \mathcal{T}_R \tag{56}$$

$$\mathcal{T}_\theta = K_a(\, i_L + \, i_R) - 2J_m\ddot{\theta} + 2J_m\ddot{\psi} - 2b_2\dot{\theta} + 2b_2\dot{\psi} \tag{57}$$

$$\mathcal{T}_\psi = -\,K_a i_{L,R} - J_m\ddot{\theta}_m + b_2\dot{\theta}_m \tag{58}$$

$$\mathcal{T}_\psi = -\,K_a(\, i_L + \, i_R) + 2J_m\ddot{\theta} - 2J_m\ddot{\psi} + 2b_2\dot{\theta} - 2b_2\dot{\psi} \tag{59}$$

$$\mathcal{T}_\phi = \frac{W}{2r}(\mathcal{T}_R - \mathcal{T}_L) \tag{60}$$

$$\mathcal{T}_\phi = \frac{W}{2r}\left(K_a\,(i_R - i_L) - J_m\left(\left(\ddot{\theta}_R - \ddot{\theta}_L\right) - \ddot{\psi} + \ddot{\psi}\right) - b_2\left(\left(\dot{\theta}_R - \dot{\theta}_L\right) - \dot{\psi} + \dot{\psi}\right)\right) \tag{61}$$

$$\mathcal{T}_\phi = \frac{W}{2r}\left(K_a\,(i_R - i_L) - J_m\left(\ddot{\theta}_R - \ddot{\theta}_L\right) - b_2\left(\dot{\theta}_R - \dot{\theta}_L\right)\right) \tag{62}$$

$$\mathcal{T}_\phi = \frac{W}{2r}\left(K_a\,(i_R - i_L) - \frac{W}{r}J_m\ddot{\phi} - \frac{W}{r}b_2\dot{\phi}\right) \tag{63}$$

Sustituyendo la ecuación 52 y 53 en las ecuaciones 57, 59 y 63:

$$\mathcal{T}_\theta = \frac{K_a}{R}(V_L + V_R) - 2J_m\ddot{\theta} - 2\left(\frac{K_aK_b}{R} + b_2\right)\dot{\theta} + 2J_m\ddot{\psi} + 2\left(\frac{K_aK_b}{R} + b_2\right)\dot{\psi} \tag{64}$$

$$\mathcal{T}_\psi = -\frac{K_a}{R}(V_L + V_R) + 2J_m\ddot{\theta} + 2\left(\frac{K_aK_b}{R} + b_2\right)\dot{\theta} - 2J_m\ddot{\psi} - 2\left(\frac{K_aK_b}{R} + b_2\right)\dot{\psi} \tag{65}$$

$$\mathcal{T}_\phi = \frac{W}{2\,r}\frac{K_a}{R}(V_R - V_L) - \frac{W^2}{2r^2}J_m\ddot{\phi} - \frac{W^2}{2r^2}\left(\frac{K_aK_b}{R} + b_2\right)\dot{\phi} \tag{66}$$

Para facilitar el manejo de las ecuaciones:

$$\alpha = \frac{K_a}{R} \tag{67}$$

$$\beta = \frac{K_a K_b}{R} + b_2 \tag{68}$$

Volviendo a escribir las fuerzas generalizadas:

$$\boldsymbol{\mathcal{T}_\theta = \alpha(V_L + V_R) - 2J_m\ddot{\theta} - 2\beta\dot{\theta} + 2J_m\ddot{\psi} + 2\beta\dot{\psi}} \tag{69}$$

$$\boldsymbol{\mathcal{T}_\psi = -\alpha(V_L + V_R) + 2J_m\ddot{\theta} + 2\beta\dot{\theta} - 2J_m\ddot{\psi} - 2\beta\dot{\psi}} \tag{70}$$

$$\boldsymbol{\mathcal{T}_\emptyset = \frac{W}{2r}\alpha(V_R - V_L) - \frac{W^2}{2r^2}J_m\ddot{\emptyset} - \frac{W^2}{2r^2}\beta\dot{\emptyset}} \tag{71}$$

2.1.5 Sistema Lineal En Representación De Espacio De Estados

Igualando las ecuaciones 44, 45 y 46 con las ecuaciones 69, 70 y 71 respectivamente y despejando, obtenemos las ecuaciones del sistema completo linealizado [1]:

$$\boldsymbol{\alpha(V_L + V_R) = \left((2M + m)r^2 + 2J + 2J_m\right)\ddot{\theta} + (mrL - 2J_m)\ddot{\psi} + 2(\beta + b_w)\dot{\theta} - 2\beta\dot{\psi}} \tag{72}$$

$$\boldsymbol{-\alpha(V_L + V_R) = (mrL - 2J_m)\ddot{\theta} + \left(mL^2 + I_\psi + 2J_m\right)\ddot{\psi} - 2\beta\dot{\theta} + 2\beta\dot{\psi} - mgL\psi} \tag{73}$$

$$\boldsymbol{\frac{W}{2r}\alpha(V_R - V_L) = \left(\frac{1}{2}MW^2 + \frac{W^2}{2r^2}(J + J_m) + I_\emptyset\right)\ddot{\emptyset} + \frac{W^2}{2r^2}(\beta + b_w)\dot{\emptyset}} \tag{74}$$

Ahora las ecuaciones pueden ser expresadas de la forma:

$$\boldsymbol{E}\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + \boldsymbol{F}\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + \boldsymbol{G}\begin{bmatrix}\theta\\ \psi\end{bmatrix} = \boldsymbol{H}\begin{bmatrix}V_L\\ V_R\end{bmatrix}$$

$$\boldsymbol{I\ddot{\emptyset} + J\dot{\emptyset} = K(V_R - V_L)}$$

Donde:

$$\boldsymbol{E} = \begin{bmatrix}(2M + m)r^2 + 2J + 2J_m & mrL - 2J_m\\ mrL - 2J_m & mL^2 + I_\psi + 2J_m\end{bmatrix} \tag{75}$$

$$\boldsymbol{F} = \begin{bmatrix}2(\beta + b_w) & -2\beta\\ -2\beta & 2\,\beta\end{bmatrix}0 \tag{76}$$

$$\boldsymbol{G} = \begin{bmatrix}0 & 0\\ 0 & -mgL\end{bmatrix} \tag{77}$$

$$\boldsymbol{H} = \begin{bmatrix}\alpha & \alpha\\ -\alpha & -\alpha\end{bmatrix} \tag{78}$$

$$\boldsymbol{I} = \frac{1}{2}MW^2 + \frac{W^2}{2r^2}(J + J_m) + I_\emptyset \tag{79}$$

$$\boldsymbol{J} = \frac{W^2}{2r^2}(\beta + b_w) \tag{80}$$

$$\boldsymbol{K} = \frac{W}{2r}\alpha \tag{81}$$

Ahora para pasar a forma en Espacio de Estados, consideremos X como los estados, u como la entrada y X^T representa la transpuesta de X:

$$\boldsymbol{X} = \begin{bmatrix} \theta & \dot{\theta} & \psi & \dot{\psi} & \emptyset & \dot{\emptyset} \end{bmatrix}^T \tag{82}$$

$$\boldsymbol{u} = \begin{bmatrix} V_L & V_R \end{bmatrix} \tag{83}$$

Obteniendo así el modelo del péndulo invertido sobre dos ruedas en dos espacios de estados:

$$\dot{\boldsymbol{X}} = A\boldsymbol{X} + B\boldsymbol{u} \tag{84}$$

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\theta} & \ddot{\theta} & \dot{\psi} & \ddot{\psi} & \emptyset & \dot{\emptyset} \end{bmatrix}^T \tag{85}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & A(2,2) & A(2,3) & A(2,4) & A(2,5) & A(2,6) \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & A(4,2) & A(4,3) & A(4,4) & A(4,5) & A(4,6) \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & A(6,2) & A(6,3) & A(6,4) & A(6,5) & A(6,6) \end{bmatrix} \tag{86}$$

$$B = \begin{bmatrix} 0 & 0 \\ B(2,1) & B(2,2) \\ 0 & 0 \\ B(4,1) & B(4,2) \\ 0 & 0 \\ B(6,1) & B(6,2) \end{bmatrix} \tag{87}$$

Para obtener los términos de las matrices A y B se realiza lo siguiente:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \boldsymbol{E}^{-1}\left(-\boldsymbol{F}\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \end{bmatrix} - \boldsymbol{G}\begin{bmatrix} \theta \\ \psi \end{bmatrix} + \boldsymbol{H}\begin{bmatrix} V_L \\ V_R \end{bmatrix}\right)$$

$$\ddot{\emptyset} = -\frac{\boldsymbol{J}}{\boldsymbol{I}}\dot{\emptyset} + \frac{\boldsymbol{K}}{\boldsymbol{I}}(V_R - V_L)$$

Donde:

$$E^{-1} = \frac{adj(E)}{\det(E)} = \begin{bmatrix} \dfrac{E(2,2)}{\det(E)} & -\dfrac{E(1,2)}{\det(E)} \\ -\dfrac{E(1,2)}{\det(E)} & \dfrac{E(1,1)}{\det(E)} \end{bmatrix} \tag{88}$$

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dfrac{E(2,2)}{\det(E)} & -\dfrac{E(1,2)}{\det(E)} \\ -\dfrac{E(1,2)}{\det(E)} & \dfrac{E(1,1)}{\det(E)} \end{bmatrix}\left(\begin{bmatrix} -2(\beta + f_w) & 2\beta \\ 2\beta & -2\,\beta \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & mgL \end{bmatrix}\begin{bmatrix} \theta \\ \psi \end{bmatrix} + \begin{bmatrix} \alpha & \alpha \\ -\alpha & -\alpha \end{bmatrix}\begin{bmatrix} V_L \\ V_R \end{bmatrix}\right) \tag{89}$$

$$\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix}=\begin{bmatrix}-2\dfrac{(\beta+b_w)E(2,2)+\beta E(1,2)}{\det(E)} & 2\beta\dfrac{E(2,2)+E(1,2)}{\det(E)}\\ 2\dfrac{(\beta+b_w)E(1,2)+\beta E(1,1)}{\det(E)} & -2\,\beta\dfrac{E(1,2)+E(1,1)}{\det(E)}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix}$$
$$+\begin{bmatrix}0 & -mgL\dfrac{E(1,2)}{\det(E)}\\ 0 & +mgL\dfrac{E(1,1)}{\det(E)}\end{bmatrix}\begin{bmatrix}\theta\\ \psi\end{bmatrix} \quad (90)$$
$$+\begin{bmatrix}\alpha\dfrac{E(2,2)+E(1,2)}{\det(E)} & \alpha\dfrac{E(2,2)+E(1,2)}{\det(E)}\\ -\alpha\dfrac{E(1,2)+E(1,1)}{\det(E)} & -\alpha\dfrac{E(1,2)+E(1,1)}{\det(E)}\end{bmatrix}\begin{bmatrix}V_L\\ V_R\end{bmatrix}$$

Las ecuaciones que quedan son:

$$\ddot{\theta}=-2\frac{(\beta+f_w)E(2,2)+\beta E(1,2)}{\det(E)}\dot{\theta}-MgL\frac{E(1,2)}{\det(E)}\,\psi+2\beta\frac{E(2,2)+E(1,2)}{\det(E)}\dot{\psi}$$
$$+\alpha\frac{E(2,2)+E(1,2)}{\det(E)}(V_L+V_R) \quad (91)$$

$$\ddot{\psi}=2\frac{(\beta+f_w)E(1,2)+\beta E(1,1)}{\det(E)}\dot{\theta}+MgL\frac{E(1,1)}{\det(E)}\psi-2\,\beta\frac{E(1,2)+E(1,1)}{\det(E)}\dot{\psi}$$
$$-\alpha\frac{E(1,1)+E(1,2)}{\det(E)}(V_L+V_R) \quad (92)$$

$$\ddot{\phi}=-\frac{\boldsymbol{J}}{\boldsymbol{I}}\dot{\phi}+\frac{\boldsymbol{K}}{\boldsymbol{I}}(V_R-V_L) \quad (93)$$

Por lo tanto:

$$A(2,2)=-2\frac{(\beta+b_w)E(2,2)+\beta E(1,2)}{\det(E)} \quad (94)$$

$$A(2,3)=-mgL\frac{E(1,2)}{\det(E)} \quad (95)$$

$$A(2,4)=2\beta\frac{E(2,2)+E(1,2)}{\det(E)} \quad (96)$$

$$A(4,2)=2\frac{(\beta+b_w)E(1,2)+\beta E(1,1)}{\det(E)} \quad (97)$$

$$A(4,3)=mgL\frac{E(1,1)}{\det(E)} \quad (98)$$

$$A(4,4)=-2\,\beta\frac{E(1,1)+E(1,2)}{\det(E)} \quad (99)$$

$$A(6,6)=-\frac{\boldsymbol{J}}{\boldsymbol{I}} \quad (100)$$

$$A(2,5)=A(2,6)=A(4,5)=A(4,6)=A(6,2)=A(6,3)=A(6,4)=A(6,5)=0 \quad (101)$$

$$B(2,1)=B(2,2)=\alpha\frac{E(2,2)+E(1,2)}{\det(E)} \quad (102)$$

$$B(4,1)=B(4,2)=-\alpha\frac{E(1,1)+E(1,2)}{\det(E)} \quad (103)$$

$$B(6,1) = -\frac{\boldsymbol{K}}{\boldsymbol{I}} \tag{104}$$

$$B(6,2) = \frac{\boldsymbol{K}}{\boldsymbol{I}} \tag{105}$$

$$\det(E) = E(1,1)E(2,2) - E(1,2)^2 \tag{106}$$

Obteniendo finalmente un sistema de sexto orden con dos entradas:

$$\begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \ddot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{\psi}} \\ \ddot{\boldsymbol{\psi}} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & A(2,2) & A(2,3) & A(2,4) & A(2,5) & A(2,6) \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & A(4,2) & A(4,3) & A(4,4) & A(4,5) & A(4,6) \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & A(6,2) & A(6,3) & A(6,4) & A(6,5) & A(6,6) \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \\ \boldsymbol{\psi} \\ \dot{\boldsymbol{\psi}} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ B(2,1) & B(2,2) \\ 0 & 0 \\ B(4,1) & B(4,2) \\ 0 & 0 \\ B(6,1) & B(6,2) \end{bmatrix} \begin{bmatrix} \boldsymbol{V_L} \\ \boldsymbol{V_R} \end{bmatrix} \tag{107}$$

Suponiendo que se puedan medir los tres ángulos y velocidades principales del sistema del péndulo invertido sobre dos ruedas y además no dependen de las entradas, entonces la ecuación de salidas que representada como:

$$\boldsymbol{Y} = C\boldsymbol{X} \tag{108}$$

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \dot{\boldsymbol{\theta}} \\ \boldsymbol{\psi} \\ \dot{\boldsymbol{\psi}} \\ \emptyset \\ \dot{\emptyset} \end{bmatrix} \tag{109}$$

Este sistema permitirá controlar las posiciones y velocidades angulares del péndulo, si es que resultan ser controlables; sin embargo podemos reducir el orden del sistema a cuarto grado, donde se podrá manipular únicamente las velocidades angulares, ya no las posiciones angulares del péndulo y todavía obedeciendo el objetivo del trabajo, controlar el TWIP en un plano XY mientras mantiene su posición vertical.

Para hacer esto simplemente podemos remover del sistema las ecuaciones referentes a las posiciones angulares quedando el siguiente sistema:

$$\begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{\psi}} \\ \ddot{\boldsymbol{\psi}} \\ \ddot{\emptyset} \end{bmatrix} = \begin{bmatrix} A(2,2) & A(2,3) & A(2,4) & 0 \\ 0 & 0 & 1 & 0 \\ A(4,2) & A(4,3) & A(4,4) & 0 \\ 0 & 0 & 0 & A(6,6) \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \boldsymbol{\psi} \\ \dot{\boldsymbol{\psi}} \\ \dot{\emptyset} \end{bmatrix} + \begin{bmatrix} B(2,1) & B(2,2) \\ 0 & 0 \\ B(4,1) & B(4,2) \\ B(6,1) & B(6,2) \end{bmatrix} \begin{bmatrix} \boldsymbol{V_L} \\ \boldsymbol{V_R} \end{bmatrix}$$

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \boldsymbol{\psi} \\ \dot{\boldsymbol{\psi}} \\ \dot{\emptyset} \end{bmatrix} \tag{110}$$

Esto debido a que al remover las variables θ y$\emptyset$, se están eliminando las columnas y renglones nulas de la matriz A y realmente no se está afectando el modelo del sistema debido a que estas variables no se encuentran en el modelo del sistema, se encuentran definidas como las derivadas de las variables que sí se encuentran en el modelo.

Demostrando más adelante que el sistema de sexto orden no es una realización mínima de la función de transferencia asociada al sistema y el sistema de cuarto orden es una realización mínima de la misma función de transferencia para el control de velocidades.

2.2 IDENTIFICACIÓN PARAMÉTRICA DEL SISTEMA

El sistema anterior cuenta con propiedades físicas constantes, es necesario conocer estos valores para poder facilitar la lectura de las ecuaciones y matrices del sistema y poder proceder con el diseño del controlador. Entre estas propiedades se pueden encontrar masas y distancias, las cuales son fácilmente medibles, otras no son tan fácilmente de medir como las inercias, por ello se hará uso de un software de diseño que apoye y verifique el cálculo de los parámetros.

A continuación se listan los parámetros físicos que interactúan en las ecuaciones que representan al sistema:

g	[m/s^2]	:	Aceleración producida por la gravedad
m	[kg]	:	Masa del péndulo
M	[kg]	:	Masa de la llanta
r	[m]	:	Radio de la llanta
$J = Mr^2/2$	[kgm^2]	:	Momento de inercia de la llanta
W	[m]	:	Distancia entre llantas
L	[m]	:	Distancia del eje de las llantas al centro de masa
I_ψ	[kgm^2]	:	Momento de inercia (Pitch) en el centro de masa
I_Φ	[kgm^2]	:	Momento de inercia (Yaw) en el centro de masa
R	[Ω]	:	Resistencia del motor de CD
K_a	[Nm/A]	:	Constante de par del motor de CD
K_b	[V s/rad]	:	Constante EMF del motor de CD
J_m	[kgm^2]	:	Momento de inercia del motor de CD
b_W	[kgm^2/s rad]	:	Coeficiente de fricción entre las llantas y el suelo
b_2	[kgm^2/s rad]	:	Coeficiente de fricción entre el péndulo y el motor de CD

Para poder asignar valores, tenemos que tener un modelo físico del péndulo, sin él las posibilidades de combinaciones de valores que se pueden aplicar a los parámetros del sistema son prácticamente ilimitadas y esto evitaría el diseño adecuado del controlador.

2.2.1 Diseño Conceptual Del TWIP

Para un péndulo invertido sobre dos ruedas (TWIP) podemos recurrir a distintos tipos de configuraciones dependiendo de qué características se modifiquen y estas se ven directamente reflejadas en los parámetros físicos que lo definen:

ALTURA DEL PÉNDULO:

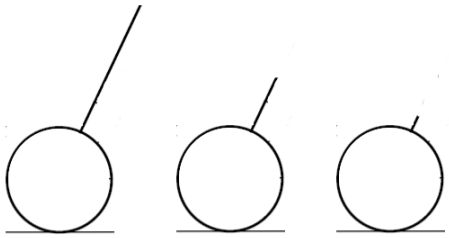


Figura 2.3: Configuración - Altura Del Péndulo

La altura del péndulo es una característica importante, en principio entre más alto sea, más fácil de controlar será, ya que un péndulo más alto tendrá un centro de masa más alto respecto al eje de las ruedas y también puede ser visto desde el punto de vista del brazo de palanca. También esperamos que sea una estructura rígida, de un material que no sufra flexiones debido a perturbaciones

DISTANCIA ENTRE RUEDAS:

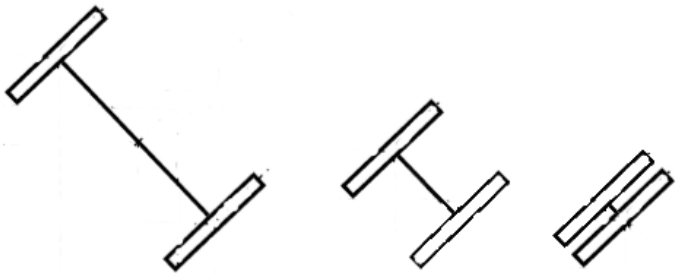


Figura 2.4: Configuración - Distancias Entre Ruedas

La distancia entre ruedas simplemente cambia la relación de giro del TWIP, pero entre más pequeña sea la distancia menos espacio de chasis se tendrá para sostener componentes que permitan su operación.

CENTRO DE MASA:

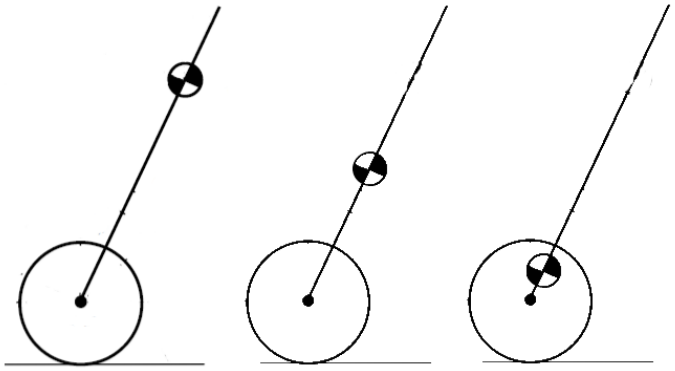


Figura 2.5: Configuración - Centro De Masa

El centro de masa está íntimamente ligado al modelo matemático por medio del parámetro "*L*" el cual representa la distancia del eje de las ruedas al centro de masa. Se recomienda que el centro de masa esté lo más alto posible pues así será más fácil de controlar el sistema, ya que los motores de las ruedas tendrán que aplicar menores pares para compensar el movimiento del péndulo. Aunque tampoco puede ser infinitamente alto pues estaríamos limitados por una parte por la capacidad de carga de los motores y por otro lado de las propiedades mecánicas del material del péndulo.

FORMA DEL CUERPO:

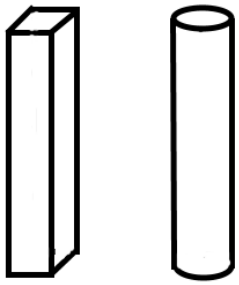


Figura 2.6: Configuración – Forma Del Cuerpo

La forma del cuerpo, chasis y/o el péndulo pueden ser prismáticas o cilíndricas para facilitar el cálculo de inercias, pero pueden ser prácticamente cualquier forma o combinación de formas y en cualquier tipo de configuración preferentemente obteniendo un diseño simétrico el cual también facilita el cálculo de las inercias.

RUEDAS:

Figura 2.7: Configuración: Tipo De Ruedas

Las ruedas pueden ser grandes o pequeñas en cuanto al radio, pueden ser anchas o delgadas aumentando o disminuyendo la superficie de contacto con el piso pero se deben poder ajustar perfectamente a los motores ya sea directamente o por medio de coples y adaptadores para evitar derrapes que afecten el funcionamiento del sistema.

Las ruedas son los elementos del TWIP que lo convierten en un sistema "*No Holonomónico"*, esto quiere decir que a pesar de que el sistema cuenta con tres grados de libertad, tiene restricciones físicas que le impiden moverse en uno de los ejes.

Si las ruedas se encuentran paralelas al eje “X”, no podrá moverse en sentido del eje “Y”, para poder llegar a una posición en el eje “Y”, el sistema forzosamente tendrá que realizar al menos una rotación para alcanzar la posición deseada a menos de que se utilicen las llamadas “Llantas Mecanum” que debido a su configuración le permiten al robot que las posee poderse mover no solo hacia adelante o atrás sino también moverse hacia los lados, pero las características de esta peculiares llantas deberán incorporarse al modelo si es que se quiere utilizarlas.

Este tipo de llantas volverían al TWIP en un sistema “*Holonomónico*”, facilitando su maniobrabilidad pero dificultaría el modelado general del sistema.

ESTRUCTURA DE FUNCIÓN PARA EL TWIP

El péndulo invertido sobre dos ruedas debe realizar una serie de operaciones para poder cumplir las funciones que se le han asignado, en el siguiente diagrama se pueden observar las funciones básicas que debe seguir para su correcto funcionamiento:

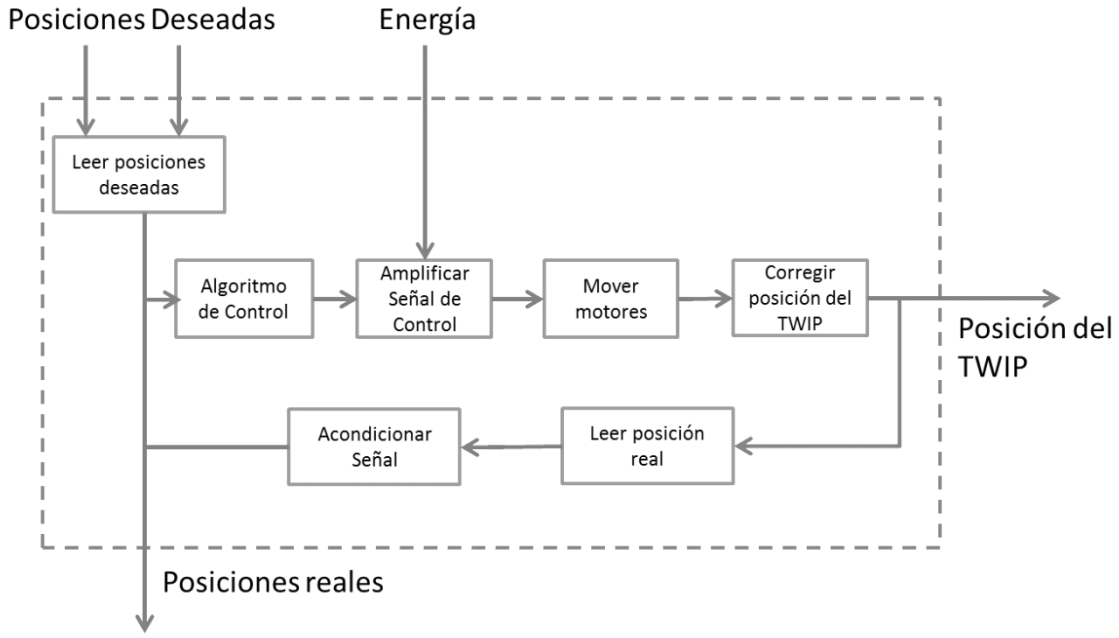


Figura 2.8: Estructura De Función Para El TWIP

LEER POSICIONES DESEADAS:

Para resolver esta función dependemos de la interfaz que se vaya a utilizar, se puede enviar directamente el valor deseado al algoritmo de control por medio de un protocolo de comunicación o si se utiliza un componente eléctrico como un potenciómetro, se requiere del uso de un convertidor analógico digital.

ALGORITMO DE CONTROL:

El algoritmo de control puede ser realizado en un computadora por medio de algún software matemático para enviar los cálculos al microcontrolador del TWIP, o puede ser programado directamente en el microcontrolador TWIP.

AMPLIFICAR SEÑAL DE CONTROL (ETAPA DE POTENCIA):

Para esta función se debe conocer el tipo de señal de control, si es una señal analógica, es necesario el uso de amplificadores; si es una señal digital, se requiere del uso de transistores.

Figura 2.9: Tipos De Amplificadores De Señal

MOTORES:

Existen motores de Corriente Directa, Corriente Alterna, servomotores y motores a pasos:

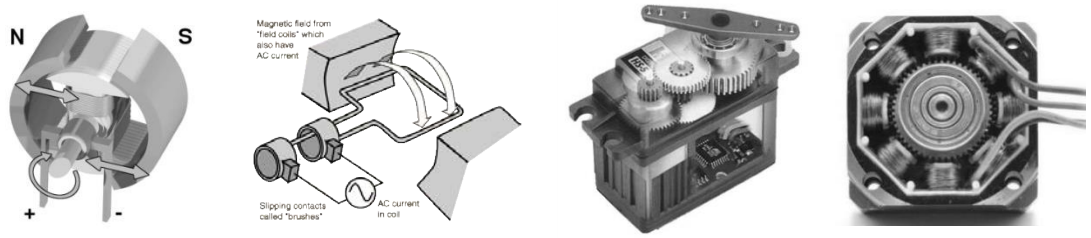


Figura 2.10: Tipos de Motores

Cada uno de ellos necesita un sistema de control diferente.

LEER POSICIONES REALES:

Para leer las posiciones reales del péndulo se requieren sensores, los sensores pueden ser de distintas clases y dependen también de qué variable se quiere leer. Para la posición de las ruedas se pueden usar potenciómetros, encoders ópticos, encoders incrementales o sensores de efecto hall. Para sensar los ángulos de orientación del péndulo se requieren de IMUs, las cuales son un conjunto de acelerómetros, giroscopios y magnetómetros en algunos casos.

ACONDICIONAMIENTO DE SEÑALES:

Depende del tipo de señal, algunas señales requieren del uso de algoritmos para convertir los datos leídos en algún otro tipo de dato como en las IMUs.

Si las señales son ruidosas, estas tienen que ser filtradas, los filtros pueden ser implementados con circuitos eléctricos o por medio de programación usando algoritmos matemáticos.

2.2.2 Diseño De Configuración Del TWIP

Existen diferentes sistemas comerciales de construcción que han sido utilizados para construir sistemas TWIP, como el LEGO® o Mecano® entre otros no tan conocidos.

En esta ocasión se optó por utilizar el sistema Actobotics™, el cual es un sistema de construcción de precisión basado en rodamientos de la compañía ServoCity™. Los patrones de agujeros

superpuestos únicos permiten posibilidades de montaje prácticamente ilimitadas, y los componentes de precisión ofrecen estructuras con tolerancias estrechas y de baja fricción, utilizando aluminio de alta calidad y componentes de acero inoxidable proporcionan una base rígida y robusta. El tamaño estandarizado permite el uso de componentes que no pertenecen a la línea Actobotics™ incluyendo servos, motores, actuadores, y componentes de vehículos de radio control. El diseño intuitivo hace que la construcción sea fácil, tanto para el ingeniero experimentado como para el aficionado novato [29].

Se eligieron todos los componentes mecánicos del TWIP de esta línea de componentes Actobotics™ para facilitar la construcción y evitar el diseño de componentes principales y así poder concentrar todos los esfuerzos en el principal objetivo de este trabajo, el diseño del controlador.

Se eligió como estructura principal del TWIP una forma de "T" invertida, compuesta de perfiles rectangulares de 12", el cual minimiza la cantidad de material y le da altura al péndulo sin un aumento importante de peso, ruedas de 4.9" conectadas a los motores por medio adaptadores y coples, los motores están sujetos a la parte baja de la estructura por medio de unas abrazaderas de aluminio especialmente diseñadas para los motores elegidos.

También se agregó un adaptador en el extremo del péndulo compuesto de un perfil rectangular de 1.5" y un tubo de aluminio de 1" de diámetro por 4" de longitud unidos por medio de una abrazadera, este adaptador permitirá colocar pesas en la parte superior del péndulo y de esta manera elevar la posición del centro de masa y realizar diferentes pruebas en caso de ser necesarias.

La empresa ServoCity™ proporciona los archivos "STEP" de la mayoría de sus componentes de la línea Actobotics™ permitiéndonos realizar cualquier diseño en un software CAD.

Se procedió a realizar el diseño del TWIP en el software SOLIDWORKS®, y además de obtener el modelo en 3D, se pueden agregar los materiales de cada uno de los componentes y así calcular cualquier parámetro físico de nuestro modelo real, incluyendo inercias.

En la siguiente figura se muestra el TWIP diseñado, para mayor información y planos de ensamble, favor de revisar los anexos a este trabajo.

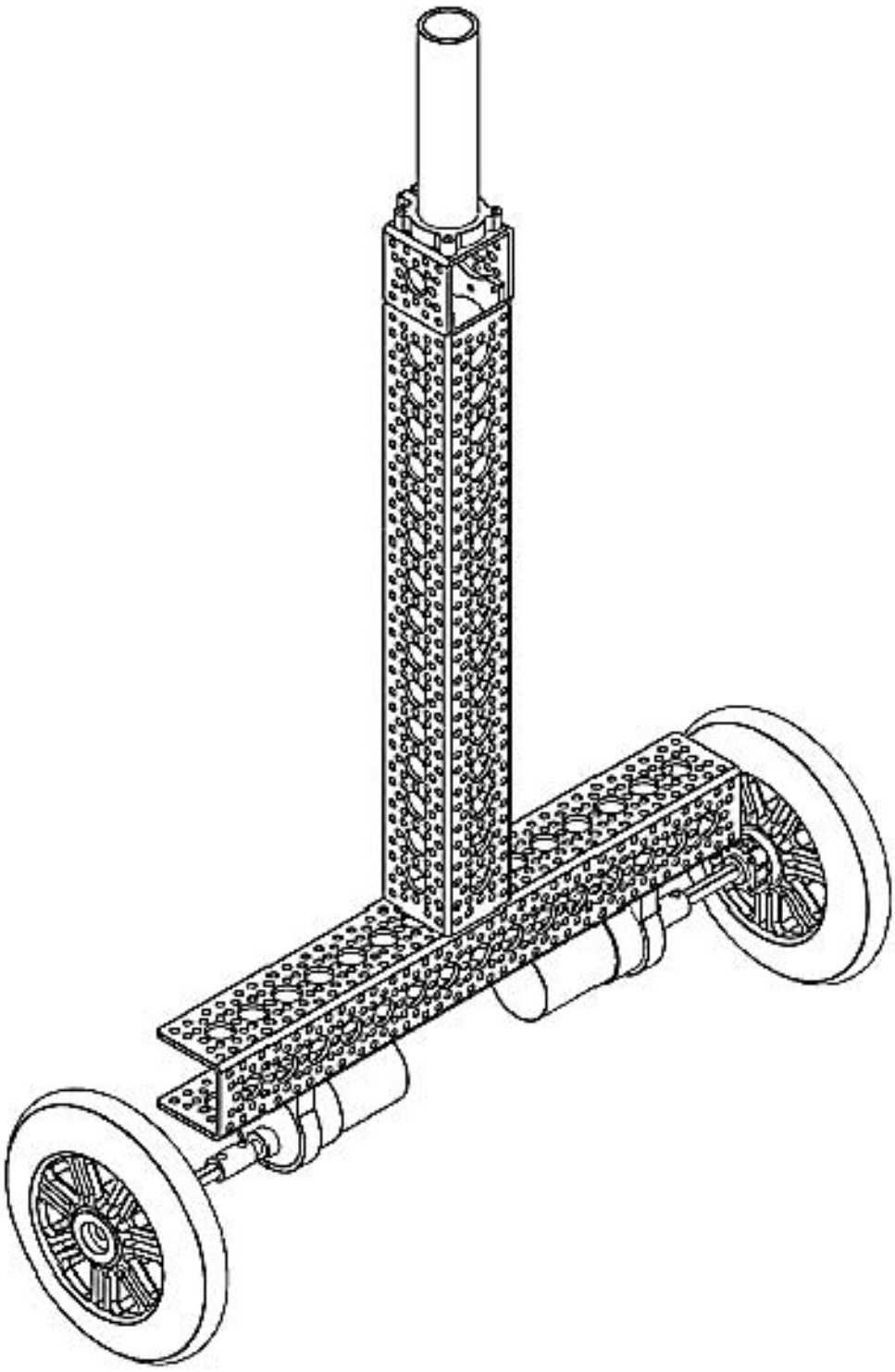


Figura 2.11: Diseño De Configuración Final

Para enviar la posición deseada al controlador se seleccionó el protocolo de comunicación Serial a través de la UART *(Unidad Asíncrona de Recepción y Transmisión)*, el cual es fácil de configurar en cualquier microcontrolador y permite el uso de gran variedad de tecnologías.

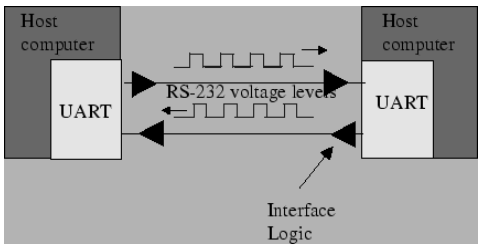


Figura 2.12: Comunicación Serial UART

Para la selección del microcontrolador se tienen que tomar en cuenta varios factores, como velocidad del reloj de operación, cantidad de puertos digitales, capacidad de interrupciones, dimensiones, voltaje de operación, voltaje lógico, capacidad de generar señales PWM, comunicación, etc.

A continuación se presenta una tabla comparativa de los microcontroladores comerciales más utilizados:

ESPECIFICACIONES TÉCNICAS					
CARACTERÍSTICA	**Arduino Uno**	**Arduino Mega**	**Raspberry Pi B**	**Teensy 3.1**	**Unidades**
PRECIO	25	54	35	20	USD
PROCESADOR	ATMega 328	ATMega 2560	BCM2835	MK20DX256VLH7	-
BITS	8	8	32	32	bits
NÚCLEO	AVR	AVR	ARM	CORTEX-M4	-
VELOCIDAD	16	16	700	72	MHz
VEL MAX	24	24	1000	96	MHZ
MEMORIA FLASH	32	256	SD socket	256	kbytes
RAM	2k	8k	512M	64k	bytes
EEPROM	1	4	32	2	kbytes
ACCESO DIRECTO A MEMORIA	N/A	N/A	N/A	16	Canales
E/S DIGITALES	14	54	26	14	
VOLTAJE SALIDA	5	5	5	3.3	Volts
VOLTAJE ENTRADA	5	5	5	5V TOLERANTE	Volts
INTERRUPCIONES	2	6	N/A*	14	pines
ENTRADA ANALÓGICA	6	16	0	21	pines
CONVERTIDORES	1	1	-	2	-
RESOLUCIÓN	10	10	-	16	bits
USABLE	10	10	-	13	bits
AMP DE GAN PROG	0	0	-	2	-
COMPARADORES	1	1	-	3	-
SALIDA ANALÓGICA	0	0	0	1	pines
RESOLUCIÓN	-	-	-	12	bits
TEMPORIZADORES	3	6	N/A*	12	-
SALIDAS PWM	6	12	software	12	pines
RTR	0	1	0	1	-
COMUNICACIÓN					
USB	1	1	2	1	-
SERIAL	1	3	1	3	-
SPI	1	1	1	1	-
I2C	1	1	1	2	-
CAN BUS	0	0	0	1	-
I2S AUDIO	0	0	0	1	-
VIDEO HDMI	0	0	1	0	-
AUDIO Jack	0	0	1	0	-
ETHERNET	0	0	1	0	-
DIMENSIONES	69 x 54 x 16	102 x 54 x16	86 x 56 x 21	35 x 18 x 5	mm

Figura 2.13: Comparación De Microcontroladores

Las características más importantes que se requieren para el funcionamiento TWIP se encuentran resaltadas en la tabla anterior.

En cuanto velocidad del procesador, entre más rápido sea, los cálculos serán más rápidos y el controlador responderá mejor, por lo que la Raspberry Pi sería la opción. En cuanto a entradas y salidas digitales, es necesario tener disponible al menos ocho E/S disponibles, cuatro entradas para los canales de los encoders y cuatro salidas para definir las direcciones de giro de los motores, así que cualquiera de los cuatro microcontroladores puede ser seleccionado. En cuanto a voltaje de entrada, los encoder generan un tren de pulsos de 5V, así que cualquier microcontrolador antes mencionado puede ser seleccionado.

Sobre interrupciones, son necesarios al menos cuatro entradas digitales con capacidad de interrupción debido a que los encoders generarán pulsos muy rápidos, hasta 540 pulsos por segundo, en sus cuatro canales de señal y sería inútil utilizar cambios de estado en los pines, esto generaría que el programa pierda pulsos y por tanto no se conocería la posición del encoder/llanta afectando el algoritmo de control. En la *Raspberry es muy difícil habilitar las interrupciones del kernel y el Arduino uno no tiene suficientes interrupciones, así que los elegidos serían el Mega y el Teensy 3.1. Hablando sobre señales PWM, de menos requerimos dos canales, uno para cada "Enable" del puente H que habilite o deshabilite el motor; cualquier microcontrolador puede ser elegido, pero la Raspberry tiene la desventaja de que sus señales son generadas por software, lo que las vuelve más lentas en comparación a las señales PWM por hardware. En cuanto a comunicación todos pueden conectarse y ser programados por medio de una computadora a través del puerto USB, también se requiere de comunicación I2C para comunicarse con la IMU y de menos un puerto Serial UART que permita en caso de ser necesario enviar y recibir datos del TWIP a la computadora y viceversa. Cualquiera de los microcontroladores cuenta con los protocolos de comunicación necesarios. Finalmente hablando de dimensiones, entre más pequeño sea menor será el efecto que tenga en el modelo matemático del TWIP y ocupará menos espacio al ser montado en el chasis, así que el elegido sería el Teensy 3.1. Evaluando lo anterior, el Teensy 3.1 es la opción a elegir ya que cuenta con todos los periféricos e interfaces necesarias para programar el controlador.

En cuanto a motores, se eligieron motores de corriente directa, pues son los motores que se encuentran modelados en el sistema, los cuales serán caracterizados en la siguiente sección.

La etapa de potencia es el elemento que permitirá convertir la señal de control, que generalmente tiene niveles lógicos de 5V o de 3V3 dependiendo del tipo de microcontrolador que se utilice. Será necesario controlar los motores en ambos sentidos, es por eso que se requiere de un arreglo de transistores en configuración de puente H, el cual convertirá la señal lógica de hasta 5V en señales de potencia. Aunque los motores pueden funcionar a 5V, es siempre necesario que la fuente que alimenta el circuito lógico y la etapa de potencia sean independientes, pues los motores pueden requerir de altos niveles de voltaje y corriente que podrían causar bajas de voltaje en el circuito lógico impidiendo su correcto funcionamiento, los motores también son capaces de funcionar como generadores eléctricos y cuando esto sucede la corriente puede ingresar al circuito lógico en sentido contrario causando daños irreparables.

Los sensores para medir la posición de la ruedas fueron encoders incrementales, y para medir los ángulos de orientación una IMU compuesta de un acelerómetro, un giroscopio y un magnetómetro.

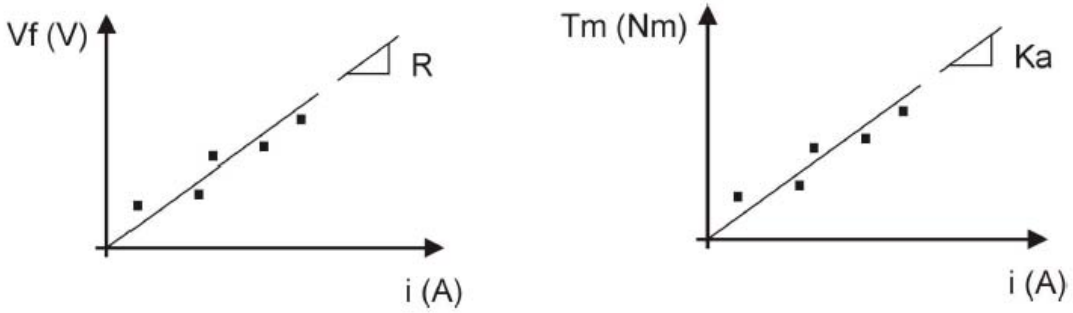
2.2.3 Selección y Caracterización De Los Motores

Los motores son los actuadores del sistema y fueron seleccionados de la línea Actobotics®, los cuales son compatibles con los adaptadores y abrazaderas de la misma línea. Para obtener los parámetros de los motores debemos caracterizarlos y así obtener la función de transferencia [27].

$$T(s) = \frac{\omega(s)}{Vf(s)} = \frac{\frac{Ka}{RJ}}{s + \frac{Ka\,Kb}{RJ}}$$

De acuerdo con las especificaciones del fabricante, los motores contienen una reducción de precisión dando 90 RPM @ 12 VCD sin carga y un par de 138.8 [oz-in] @ 12 VCD a rotor bloqueado.

Para la prueba de rotor bloqueado se realizó una regresión lineal debido a que el fabricante aconseja severamente no realizar esta prueba pues daños en el tren de engranes pueden ocurrir.



$$R = \frac{Vf}{i} \qquad Ka = \frac{Tm}{i}$$

Figura 2.14: Gráficas Para Caracterización 1 [27]

Para la prueba de rotor libre se acoplaron las flechas de los dos motores para que uno de ellos funcionara como generador eléctrico y se usó como sensor de velocidad angular un tacómetro láser.

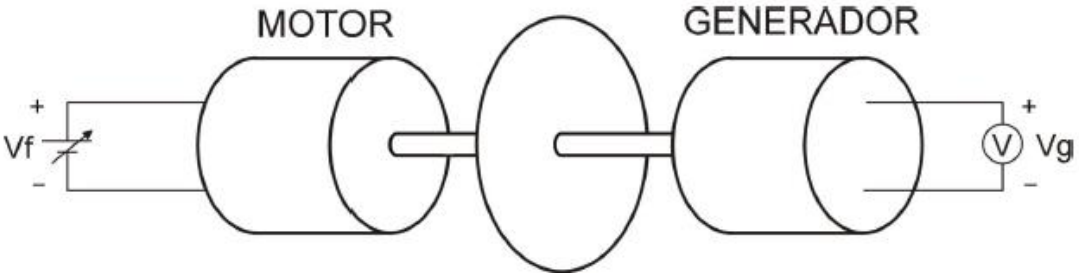
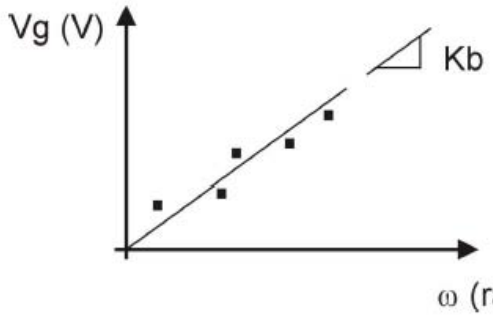


Figura 2.15: Motor – Generador (Caracterización) [27]



$$Kb = \frac{Vg}{\omega}$$

Figura 2.16: Gráficas Para Caracterización 2 [27]

Los resultados de ambas pruebas fueron los siguientes:

PRUEBA	ROTOR BLOQUEADO			ROTOR LIBRE		
Dato	Vf [V]	I [A]	Tm [Nm]	ω [rpm]	ω [rad/s]	Vg [V]
0	0.00	0.00	0.0000	0.0	0.0000	0.00
1	0.60	0.05	0.0491	0.0	0.0000	0.00
2	1.20	0.10	0.0981	0.0	0.0000	0.00
3	1.80	0.15	0.1472	0.0	0.0000	0.00
4	2.40	0.20	0.1963	8.2	0.8587	1.14
5	3.00	0.25	0.2454	13.0	1.3614	1.78
6	3.60	0.30	0.2944	18.4	1.9268	2.34
7	4.20	0.35	0.3435	22.8	2.3876	3.04
8	4.80	0.40	0.3926	26.8	2.8065	3.48
9	5.40	0.45	0.4417	31.6	3.3091	4.16
10	6.00	0.50	0.4907	36.8	3.8537	4.84
11	6.60	0.55	0.5398	40.6	4.2516	5.36
12	7.20	0.60	0.5889	46.0	4.8171	6
13	7.80	0.65	0.6380	50.0	5.2360	6.56
14	8.40	0.70	0.6870	54.8	5.7386	7.04
15	9.00	0.75	0.7361	59.2	6.1994	7.6
16	9.60	0.80	0.7852	63.8	6.6811	8.32
17	10.20	0.85	0.8343	68.4	7.1628	8.88
18	10.80	0.90	0.8833	73.4	7.6864	9.36
19	11.40	0.95	0.9324	79.0	8.2729	9.92
20	12.00	1.00	0.9815	86.0	9.0059	10.5

Figura 2.17: Resultados Caracterización 1

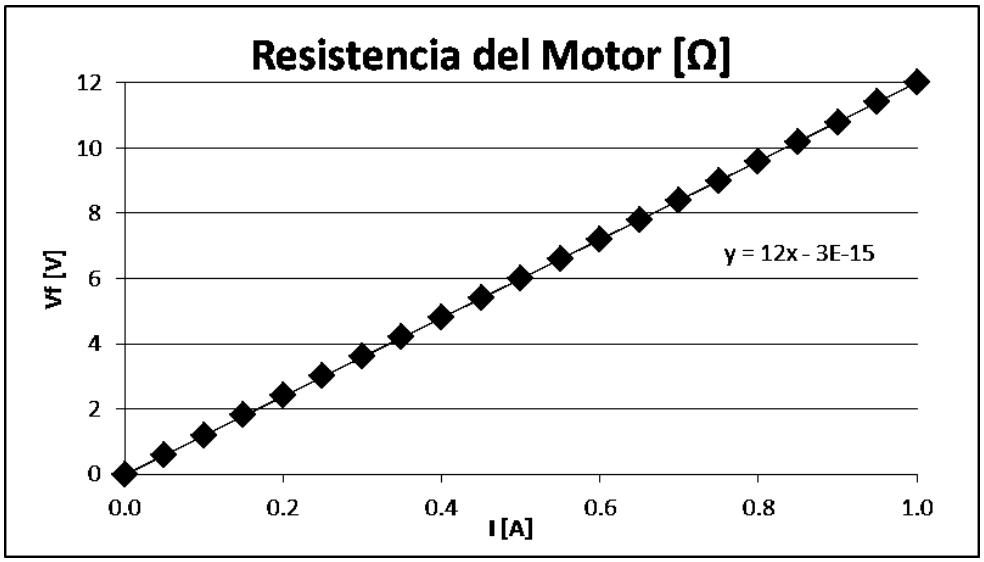


Figura 2.18: Gráfica - Resistencia Del Motor

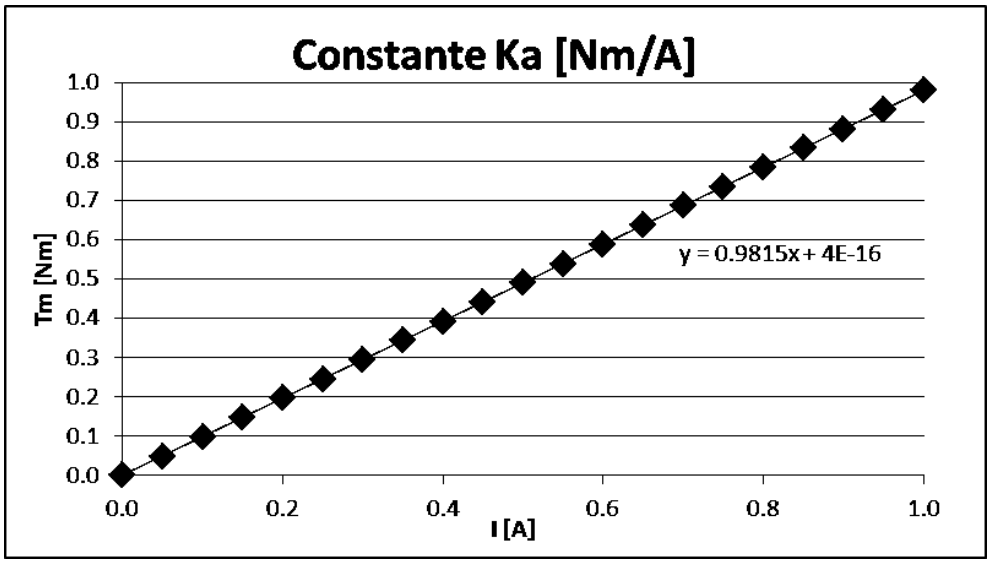


Figura 2.19: Gráfica – Constante "Ka" del Motor

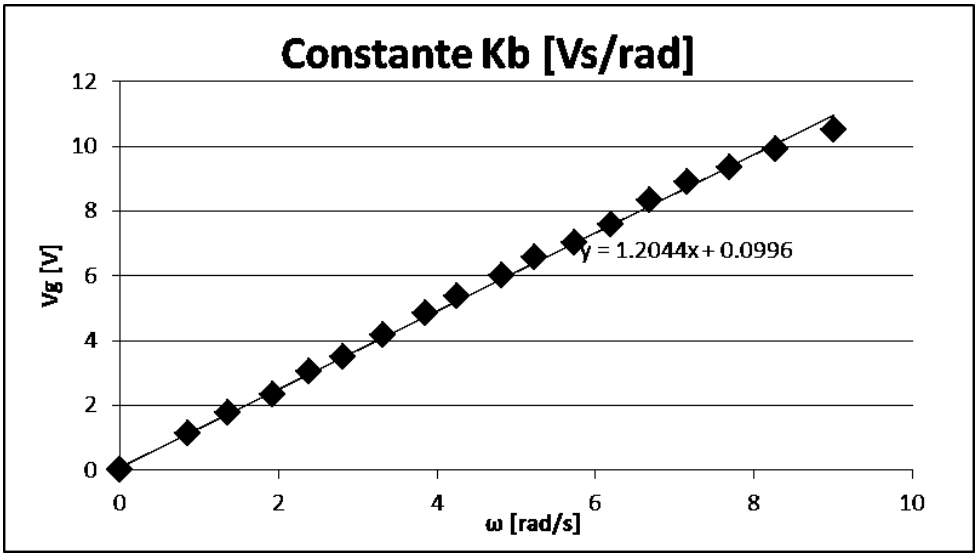


Figura 2.20: Gráfica – Constante "Kb" Del Motor

Utilizando la misma configuración de las flechas de los motores acopladas, se prosigue a realizar la prueba de Respuesta ante una entrada Escalón, conectando el motor que funciona como generador eléctrico al osciloscopio para observar la gráfica, obteniendo los siguientes resultados:

Prueba Escalón	Valor
Vf_{inf} [V]	12
Vg_{inf} [V]	10.4
Vg@63.2% [V]	6.5728
τ [s]	0.15
ω_{inf} [rad/s]	8.6347
b	6.6667
a	4.7970
J1 [Nm/A]	0.0171
J2 [Nm/A]	0.0148
J [Nm/A]	0.0159

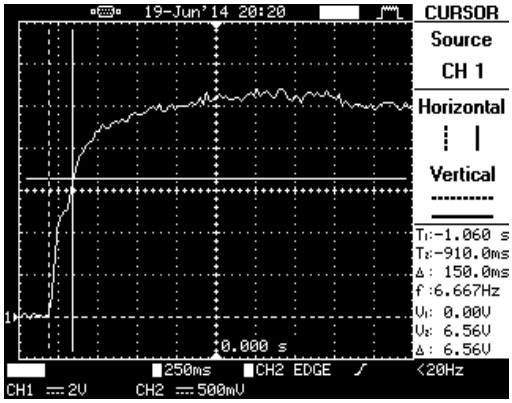


Figura 2.21: Resultados Caracterización 2

Dónde:

$$a = \frac{Ka}{R\,J} \qquad b = \frac{Ka\ Kb}{R\,J} \qquad \omega_{\infty} = \frac{Vg_{\infty}}{Kb} \qquad T(0) = \frac{\omega_{\infty}}{Vf_{\infty}} = \frac{a}{b} \qquad b = \frac{1}{\tau}$$

Obteniendo los resultados de la caracterización de los motores:

RESULTADOS	
Parámetro	Valor
R [Ω]	12.0000
Ka [Nm/A]	0.9815
Kb [Vs/rad]	1.2044
J [Kgm^2]	0.0159

Figura 2.22: Resultados Caracterización 3

Para comprobar los resultados de la parametrización se simuló la función de transferencia del motor en Matlab®.

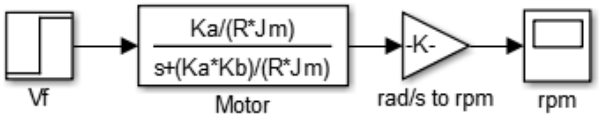


Figura 2.23: Función De Transferencia Del Motor

El modelo está cerca de las 90 RPM Nominales cuando se le aplica una entrada escalón de 12 VCD.

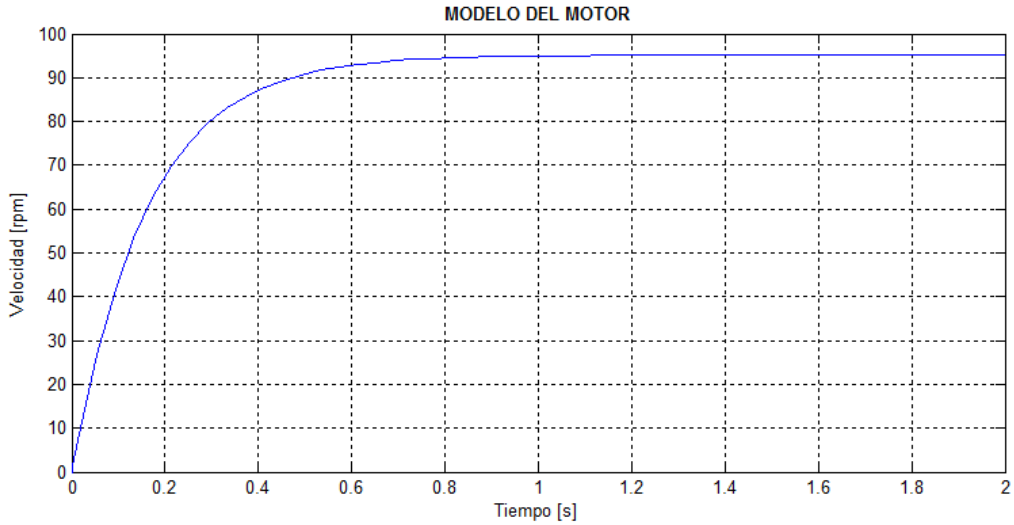


Figura 2.24: Modelo Del Motor 1

Cuando ha pasado una constante de tiempo τ=0.15 [s], se obtiene el 63.2% de 90 [RPM], que equivalen a 56.88 [RPM], lo que se puede apreciar fácilmente en la gráfica anterior.

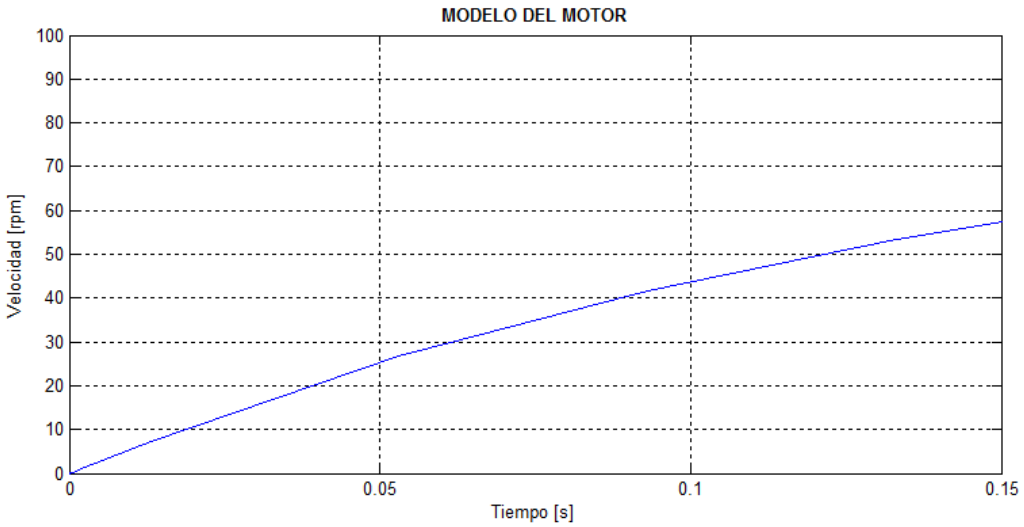


Figura 2.25: Modelo Del Motor 2

Verificando así que los valores obtenidos de la caracterización son aceptables para usarse en el modelo del TWIP.

2.2.4 Tabla De Parámetros Físicos Del TWIP

Finalmente se presenta la tabla con todos los parámetros del TWIP, las distancias fueron medidas con Vernier o con regla, las masas fueron medidas con báscula, las inercias fueron obtenidas del modelo CAD hecho en SOLIDWORKS® y los parámetros de los motores fueron obtenidos de la caracterización.

g = 9.78	[m/s^2]	:	Aceleración producida por la gravedad
m = 1.8	[kg]	:	Masa del péndulo
M = 17.8e-2	[kg]	:	Masa de la llanta
r = 62.23e-3	[m]	:	Radio de la llanta
$J = Mr^2/2$	[kgm^2]	:	Momento de inercia de la llanta
W = 388.8456e-3	[m]	:	Distancia entre llantas
L = 0.1075	[m]	:	Distancia del eje de las llantas al centro de masa
I_ψ = 0.0196	[kgm^2]	:	Momento de inercia (Pitch) en el centro de masa
I_Φ = 0.0179	[kgm^2]	:	Momento de inercia (Yaw) en el centro de masa
R = 12	[Ω]	:	Resistencia del motor de CD
K_a = 0.9815	[Nm/A]	:	Constante de par del motor de CD
K_b = 1.2044	[V s/rad]	:	Constante EMF del motor de CD
J_m = 0.0159	[kgm^2]	:	Momento de inercia del motor de CD
b_W = 0.00395	[kgm^2/s rad]	:	Coeficiente de fricción entre las llantas y el suelo
b_2 = 0	[kgm^2/s rad]	:	Coeficiente de fricción entre el robot y el motor de CD

Las fricciones fueron ajustadas durante las simulaciones para que el modelo y la planta real se parecieran lo más posible.

3 DISEÑO DEL CONTROLADOR

Una vez obtenidos los parámetros reales del TWIP físico se procederá al diseño del controlador, el cual se encargará de mantener al TWIP en su posición vertical, así como permitirle moverse libremente en un plano "XY".

Se compararán dos controladores, uno para el modelo de sexto orden y otro para el modelo de cuarto orden, presentados al final del capítulo 2.1.

3.1 OBJETIVOS DE CONTROL

Se busca mantener el péndulo invertido en su posición vertical, mientras que se pueda desplazar a lo largo del eje perpendicular al eje de las ruedas además de poder girar respecto a su centro de masa.

Por lo tanto se busca que el controlador tenga al menos estabilización (llevar la variable ψ a cero) y regulación (llevar las variables θ y φ a un valor deseado constante).

3.2 ANÁLISIS DEL SISTEMA

3.2.1 Instrumentación

El TWIP construido consta de un chasis de aluminio al cual se montan los motores (actuadores), baterías, una IMU (sensor), dos encoders de cuadratura por los cuales pasa la flecha de cada motor, el microprocesador y demás componentes que permitirán el completo funcionamiento del mismo.

Actuadores

El actuador es elemento de un sistema mecatrónico capaz de transformar energía con la finalidad de generar un efecto sobre el sistema y éste recibe la orden de un controlador, la señal de control generalmente tiene que pasar por una etapa de potencia.

En este caso los actuadores del TWIP son los motores de las ruedas, los cuales transforman la energía eléctrica en movimiento rotacional que le permitirán al TWIP compensar el movimiento del péndulo, desplazarse en línea recta y rotar respecto a su centro.

Los motores seleccionados son dos motorreductores de la compañía ServoCity® los cuales operan a 12 VCD:

Figura 3.1: Motor De CD (Físico)

Baterías

Las baterías son los elementos que proporcionan la energía eléctrica a todos los elementos del sistema para que puedan funcionar, existen diferentes tipos de baterías pero debido a su gran capacidad de carga, tiempo de descarga, tamaño compacto y el hecho de que son recargables, se seleccionaron las baterías de Polímero Litio-Ion.

La etapa de potencia funciona con dos baterías de 7.4 [VCD] @ 2200 [mAh] conectadas en serie, dando un total de 14.8 [VCD] @ 2200 [mAh].

Figura 3.2: Batería De Potencia (Física)

Para la alimentación lógica se utilizó una batería de 7.4 [VCD] @ 1000 [mAh] conectada a una fuente que regula el voltaje a 5 [VCD] que alimenta todos los componentes lógicos como el microcontrolador y sensores.

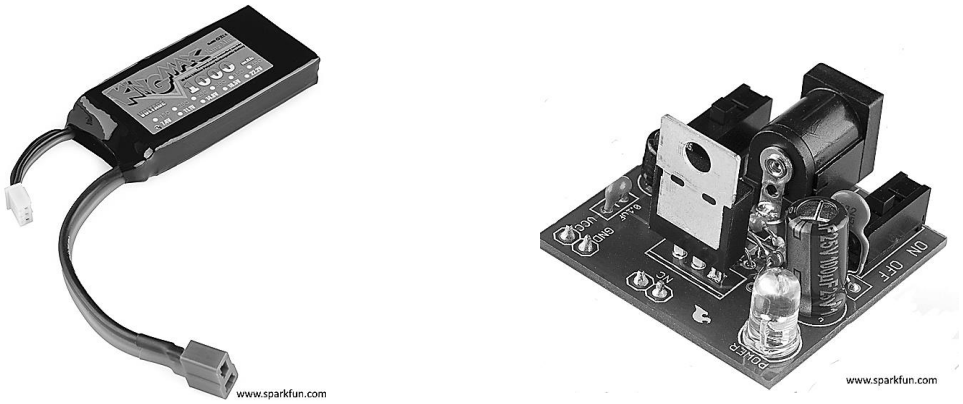


Figura 3.3: Alimentación Lógica (Física)

Etapa de Potencia

Se seleccionó un driver de motores de CD basado en el circuito integrado L298N, capaz de operar dos motores independientemente por medio de señales PWM, con capacidad de 3[A] a 36[VCD].

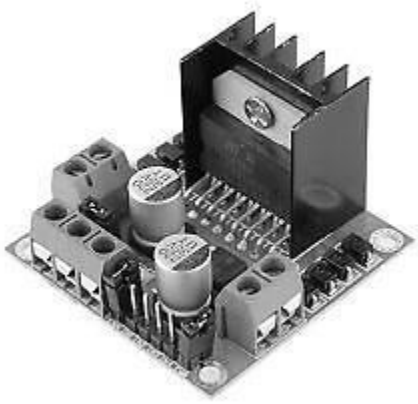


Figura 3.4: Puente H (Físico)

Microcontrolador

Teensy 3.1, el cual ofrece una buena velocidad de cálculo, y la cantidad necesaria de periféricos necesarios para realizar la interconexión de todos los elementos del TWIP.

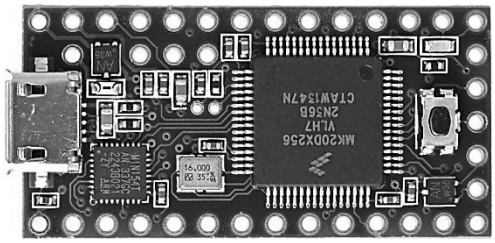


Figura 3.5: Microcontrolador Teensy 3.1 (Físico)

Sensores

1. **IMU (*Inertial Measurement Unit*) De 9 Grados De Libertad**

   Es una placa muy pequeña que integra tres sensores independientes:

   - **Acelerómetro ADXL345 de 3 grados de libertad**
   - **Magnetómetro HMC5883L de 3 grados de libertad**
   - **Giroscopio ITG-3200 de tecnología MEMS de 3 grados de libertad**

   De los datasheets de cada sensor se obtienen las frecuencias máximas de operación, las cuales son de 400Hz, 75Hz y 50Hz respectivamente.

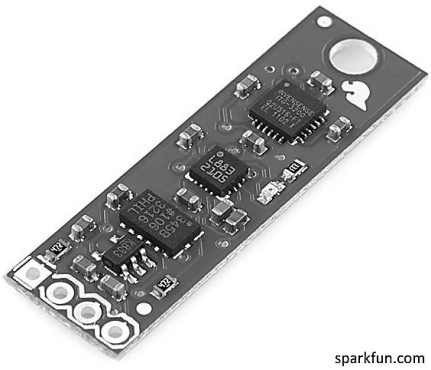


Figura 3.6: Sensor IMU 9DOF (Físico)

La IMU nos permitirá conoce el ángulo de inclinación (Pitch - ψ) del péndulo respecto a la vertical ya sea en grados o radianes, así como velocidades angulares, opera con una alimentación de 3.3 [VCD].

2. Encoders De Cuadratura

Un encoder óptico consiste en un disco rotatorio, una fuente de luz, un fotoreceptor y circuitos eléctricos. Un problema con este tipo de encoders es que la dirección de giro no puede ser determinada con un solo fotoreceptor, la salida del fotoreceptor será igual si el disco gira en sentido horario como en sentido antihorario, no existe indicador alguno hacia qué sentido se encuentra girando.

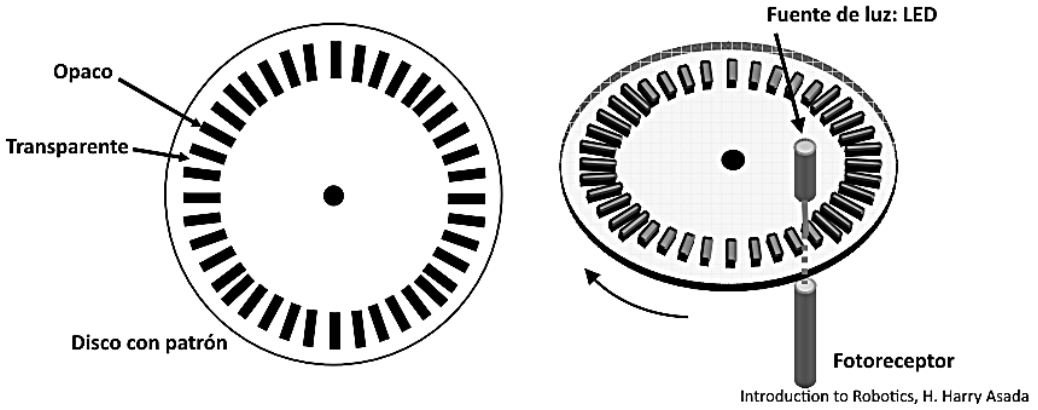


Figura 3.7: Diagrama - Encoder Óptico

Para medir posición angular, la dirección de giro debe ser determinada, una manera de obtener esta información es agregando otra fuente de luz y otro fotoreceptor y una segunda red de patrones con una diferencia de 90 grados respecto a la primera red. La siguiente figura muestra la doble red de patrones y el tren de pulsos resultante para cada sentido de giro. Se puede notar que el canal A adelanta el canal B por 90 grados cuando gira en sentido horario y el canal B adelanta 90 grados al canal A cuando gira en sentido antihorario. Simplemente hay que alimentar las señales de los canales A y B a un contador UP/DOWN.

Este tipo de encoders requieren de una inicialización de la cuenta antes de empezar a medir la posición, estos encoders son conocidos como encoders de cuadratura o encoders incrementales.

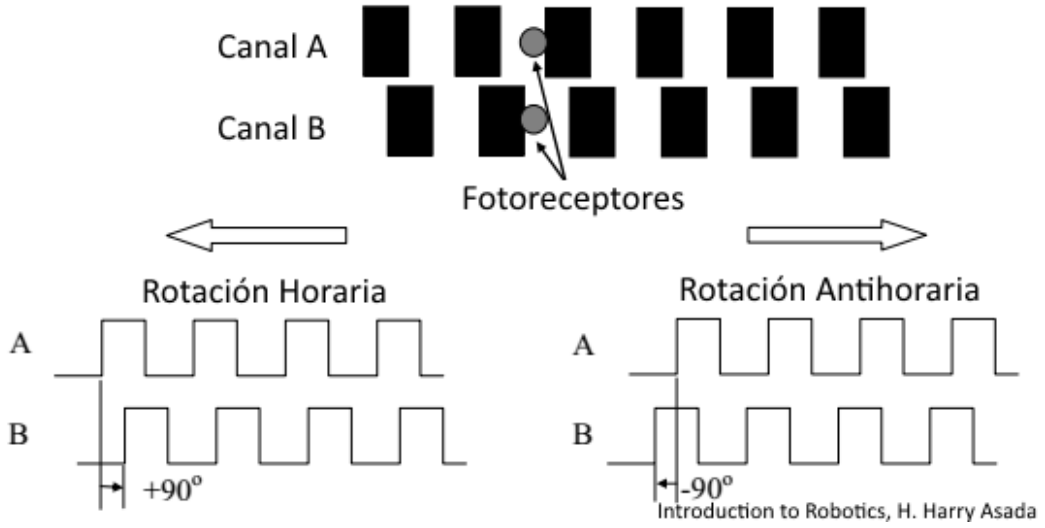


Figura 3.8: Diagrama – Encoder Óptico 2

El siguiente concepto que es necesario tener en consideración es la forma en que los trenes de pulsos de cada canal se convierten en una posición. El proceso por el cual los flancos que se han contado se convierten en posición depende del tipo de codificación utilizado.
Hay tres tipos básicos de codificación:

- ***Codificación X1***
  La siguiente figura muestra un ciclo de cuadratura y los incrementos y decrementos resultantes para una codificación X1. Cuando el pulso en el canal A ocurre antes que en el B, el incremento se produce por el flanco de subida del canal A. Cuando el pulso en el canal B ocurre antes que el canal A, el decremento se produce por el flanco de bajada del canal A.

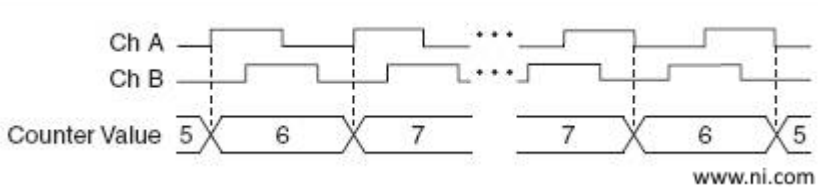


Figura 3.9: Codificación X1 (Encoders)

- ***Codificación X2***
  El mismo comportamiento tiene lugar con la codificación X2, excepto que los incrementos o decrementos del contador ocurren con cada flanco del canal A, según el pulso tenga lugar en un canal antes que en el otro. Cada ciclo se traduce en dos incrementos o decrementos, tal y como se muestra en la siguiente figura.

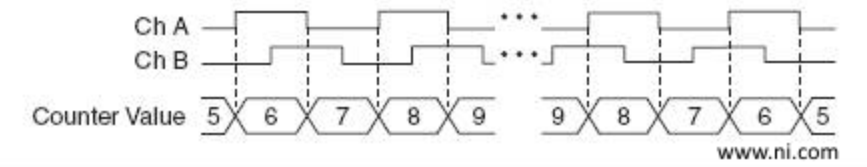


Figura 3.10: Codificación X2 (Encoders)

- ***Codificación X4***
  El contador se incrementa o decrementa con cada flanco de los canales A y B para la codificación X4. El contador se incrementa o decrementa dependiendo de si los pulsos ocurren en un canal antes que en el otro. Cada ciclo se traduce en cuatro incrementos o decrementos, tal y como se muestra en la siguiente figura.

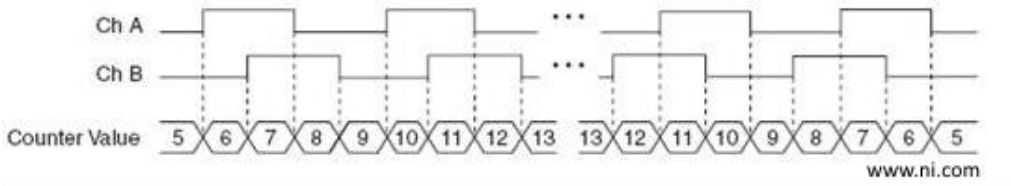


Figura 3.11: Codificación X4 (Encoders)

Una vez establecido el tipo de codificación y se han contado los impulsos, la conversión a una posición es una cuestión de utilizar la siguiente fórmula:

$$[°] = \frac{flancos}{x\ N} 360$$

Donde N = número de pulsos generados en cada giro del eje y x = tipo de codificación.

El tipo de encoder elegido para el TWIP fue el E4P-360-250-N-S-H-D-B de US Digital®, el cual es un encoder de cuadratura miniatura de 360 cuentas por revolución para ser montado en flechas de ¼" y funcionan con alimentación de 5 VCD y generan señales lógicas de 5 VCD.

Figura 3.12: Encoder De Cuadratura (Físico)

Si se utiliza la codificación X4, entonces el encoder tendrá una resolución de 0.25° por la fórmula antes mencionada:

$$[°] = \frac{1}{(4)(360)} 360 = 0.25°$$

Para detectar los cambios del tren de pulsos de cada canal de ambos encoders es necesario que el procesador cuente con capacidad de interrupción en al menos 4 entradas digitales.

3.3 SELECCIÓN DEL CONTROLADOR A DISEÑAR

Considerando el modelo en espacio de estados, sensores y plataforma de desarrollo, se pretenderá utilizar el regulador cuadrático lineal (*LQR*) como base para el control debido a sus siguientes características:

- Funciona mejor cuando se tiene acceso a todos los estados.
- Podemos darle prioridad a cada una de las variables de control.
- Puede minimizar la energía requerida para estabilizar el péndulo.
- Busca optimizar las señales de control pero podemos maximizar o minimizar la acción de control sobre alguna de las variables de manera intuitiva, lo que facilita la sintonización.
- Fácil de programar una vez conocidas las matrices de ganancias.

Debido a que algunos estados están siendo obtenidos por medio de derivación numérica y además algunas señales serán filtradas para eliminar ruidos que pueden afectar la respuesta de control, no se podrá decir que el control aplicado sea un control óptimo, sin embargo su implementación será más sencilla en un microcontrolador.

3.4 ANÁLISIS DE ESTABILIDAD Y CONTROLABILIDAD

En este apartado se analizará la estabilidad y controlabilidad tanto del sistema de sexto orden como para el sistema de cuarto orden.

ESTABILIDAD

A simple vista se puede deducir que el sistema TWIP, o cualquier péndulo invertido son inestables, pero hay que demostrarlo de manera matemática, para un sistema en variables de estado, la estabilidad queda definida por los valores característicos de la matriz A.

Para que un sistema sea estable se requiere que todos los valores característicos tengan parte real negativa, si al menos uno tiene parte real cero se considera un sistema asintóticamente estable y es condición suficiente para que un sistema sea inestable que uno de los valores característicos tenga parte real positiva.

CONTROLABILIDAD

Se dice que un sistema es controlable si se puede llevar cualquier estado inicial a cualquier otro estado mediante un vector de control en un intervalo de tiempo finito.

La controlabilidad de un sistema en representación de variables de estado queda definida por su matriz de controlabilidad:

$$\mathcal{C} = [B \quad AB \quad \dots \quad A^{n-1}B]$$

Donde n representa el orden del sistema.

Para que el sistema sea controlable, el rango de la matriz de controlabilidad debe ser igual al orden del sistema, el rango de una matriz nos indica el número de renglones o columnas linealmente independientes que posee.

OBSERVABILIDAD

Es la capacidad de un sistema para calcular el vector de estados a partir del conocimiento del vector de salidas.

La observabilidad de un sistema en representación de variables de estado queda definida por su matriz de observabilidad:

$$\boldsymbol{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ C^{n-1}A \end{bmatrix}$$

Donde n representa el orden del sistema.
Para que el sistema sea observable, el rango de la matriz de observabilidad debe ser igual al orden del sistema.

REALIZACIÓN MÍNIMA

Una función de transferencia $G(s)$ se dice realizable si existe una representación de estados que tiene a $G(s)$ como su función de transferencia, si una función de transferencia es realizable, entonces existe un número infinito de realizaciones no necesariamente del mismo orden.

Una realización mínima es la representación de menor orden posible de la función de transferencia, para el sistema en espacio de estados (*A,B,C*), sí y solo sí (*A,B*) es controlable y *(A,C*) es observable se dice que es una realización mínima.

3.4.1 Sistema De Sexto Orden

Las matrices A, B y C obtenidas para el sistema TWIP de sexto orden son:

$$A = \begin{bmatrix} 0 & +1.00 & 0 & 0 & 0 & 0 \\ 0 & -4.26 & +14.61 & +4.03 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & +0.00 \\ 0 & +1.56 & +30.21 & -1.62 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -5.73 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 1.67 & 1.67 \\ 0 & 0 \\ -0.67 & -0.67 \\ 0 & 0 \\ -0.73 & 0.73 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FUNCIONES DE TRANSFERENCIA

Por medio de Matlab® podemos obtener rápidamente las funciones de transferencia del espacio de estados por medio del comando “ss2tf(A,B,C,D,n)”:

$$De\ \boldsymbol{V_L}\text{ó }\boldsymbol{V_R}\ a\ \dot{\boldsymbol{\theta}}\text{:}\quad G(s)_{1,2} = \frac{1.677\,s^5 + 9.624\,s^4 - 60.51\,s^3 - 347.3\,s^2}{s^6 + 11.63\,s^5 + 4.178\,s^4 - 321.5\,s^3 - 870.1\,s^2}$$

$$De\ \boldsymbol{V_L}\text{ó }\boldsymbol{V_R}\ a\ \dot{\boldsymbol{\psi}}\text{:}\quad G(s)_{3,4} = \frac{-0.674\,s^5 - 4.121\,s^4 - 1.45\,s^3}{s^6 + 11.63\,s^5 + 4.178\,s^4 - 321.5\,s^3 - 870.1\,s^2}$$

$$De\ \boldsymbol{V_L}\ a\ \dot{\emptyset}\text{:}\quad G(s)_5 = \frac{-0.7333\,s^5 - 4.316\,s^4 + 21.71\,s^3 + 111.2\,s^2}{s^6 + 11.63\,s^5 + 4.178\,s^4 - 321.5\,s^3 - 870.1\,s^2}$$

$$De\ \boldsymbol{V_R}\ a\ \dot{\emptyset}\text{:}\quad G(s)_6 = \frac{0.7333\,s^5 + 4.316\,s^4 - 21.71\,s^3 - 111.2\,s^2}{s^6 + 11.63\,s^5 + 4.178\,s^4 - 321.5\,s^3 - 870.1\,s^2}$$

ESTABILIDAD

Por medio del software Matlab® se obtuvieron los valores característicos del sistema:

$$eig(A) = \begin{bmatrix} 0 \\ -7.02 \\ -4.10 \\ 5.25 \\ 0 \\ -5.73 \end{bmatrix}$$

Por lo que el sistema TWIP de sexto orden es un sistema inestable, debido a su cuarto valor característico.

CONTROLABILIDAD

En Matlab® la función "ctrb(A,B)" nos permite calcular rápidamente la matriz de controlabilidad del sistema, la cual es demasiado grande para calcular a mano, y la función "rank(M)" permite calcular el rango de una matriz:

$$rank\big(ctrb(A,B)\big) = 6$$

Por lo tanto el sistema TWIP de sexto orden es completamente controlable.

OBSERVABILIDAD

En Matlab® la función "obsv(A,C)" nos permite calcular rápidamente la matriz de controlabilidad del sistema, la cual es demasiado grande para calcular a mano, y la función "rank(M)" permite calcular el rango de una matriz:

$$rank\big(obsv(A,C)\big) = 4$$

Por lo tanto el sistema TWIP de sexto orden no es completamente observable.

En resumen el sistema TWIP de sexto orden no es estable, es controlable y no es completamente observable y por lo tanto no es una realización mínima.

3.4.2 Sistema De Cuarto Orden

Las matrices A, B y C obtenidas para el sistema TWIP de cuarto orden son:

$$A = \begin{bmatrix} -4.26 & +14.61 & +4.03 & 0 \\ 0 & 0 & 1 & 0 \\ +1.56 & +30.21 & -1.62 & 0 \\ 0 & 0 & 0 & -5.73 \end{bmatrix} \quad B = \begin{bmatrix} 1.67 & 1.67 \\ 0 & 0 \\ -0.67 & -0.67 \\ -0.73 & 0.73 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FUNCIONES DE TRANSFERENCIA

Por medio de Matlab®:

$$De\ \boldsymbol{V_L} \text{ó } \boldsymbol{V_R}\ a\ \dot{\boldsymbol{\theta}}: \quad G(s)_{1,2} = \frac{1.677\,s^3 + 9.624\,s^2 - 60.51\,s - 347.3}{s^4 + 11.63\,s^3 + 4.178\,s^2 - 321.5\,s - 870.1}$$

$$De\ \boldsymbol{V_L} \text{ó } \boldsymbol{V_R}\ a\ \dot{\boldsymbol{\psi}}: \quad G(s)_{3,4} = \frac{-0.674\,s^3 - 4.121\,s^2 - 1.45\,s}{s^4 + 11.63\,s^3 + 4.178\,s^2 - 321.5\,s - 870.1}$$

$$De\ \boldsymbol{V_L}\ a\ \dot{\emptyset}: \quad G(s)_5 = \frac{-0.7333\,s^3 - 4.316\,s^2 + 21.71\,s + 111.2}{s^4 + 11.63\,s^3 + 4.178\,s^2 - 321.5\,s - 870.1}$$

$$De\ \boldsymbol{V_R}\ a\ \dot{\emptyset}: \quad G(s)_6 = \frac{0.7333\,s^3 + 4.316\,s^2 - 21.71\,s - 111.2}{s^4 + 11.63\,s^3 + 4.178\,s^2 - 321.5\,s - 870.1}$$

ESTABILIDAD

Por medio del software Matlab® se obtuvieron los valores característicos del sistema:

$$eig(A) = \begin{bmatrix} -7.02 \\ -4.10 \\ 5.25 \\ -5.73 \end{bmatrix}$$

Por lo que el sistema TWIP de cuarto orden es un sistema inestable, debido a su tercer valor característico.

CONTROLABILIDAD

En Matlab®:

$$rank\big(ctrb(A,B)\big) = 4$$

Por lo tanto el sistema TWIP de cuarto orden es completamente controlable.

OBSERVABILIDAD

En Matlab®:

$$rank\big(obsv(A,C)\big) = 4$$

Por lo tanto el sistema TWIP de sexto orden es completamente observable.

Por lo tanto el sistema TWIP de cuarto orden no es estable pero es controlable y observable, y como las funciones de transferencia son iguales a las del sistema de sexto se puede demostrar que es una realización mínima.

3.5 CONTROL LQR

La arquitectura básica de un controlador LQR es la siguiente:

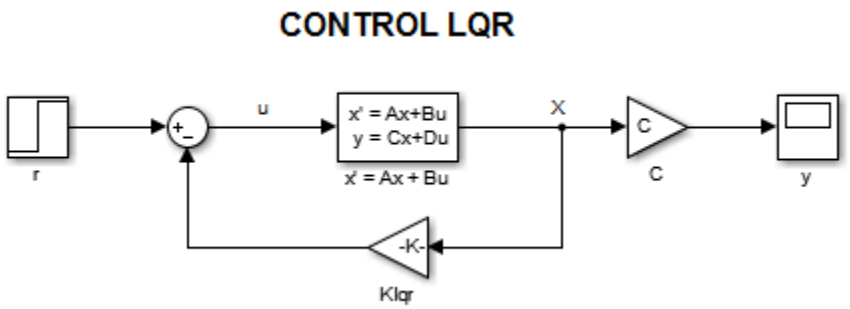


Figura 3.13: Diagrama LQR Básico

En esencia es un controlador por realimentación de estados, esto se puede apreciar al comparar las leyes de control de cada uno de los controladores:

CONTROL RE
$u = r - Kx$

CONTROL LQR
$u = r - K_{lqr}x$

La diferencia radicaen que en el control RE el cálculo de la matriz de ganancia "K" se realiza por medio de la ubicación de polos y en el control LQR la matriz de ganancia se calcula por medio de una función de costo:

$$J = \int_0^\infty (x^T Q\, x + u^T R\, u)dt$$

Donde Q es la matriz de pesos de las variables de estado y R es la matriz de pesos para la señal de control.

Si se considera que r = 0 y que el término $(x^T Q\, x\,)$ se pueda expresar como:

$$x^T Q\, x = -\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,)$$

$$J = \int_0^\infty -\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,)dt$$

Siendo P, una matriz simétrica y positiva definida:

$$P = P^T$$

$$P > 0$$

Entonces:

$$\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,) = x^T P\, \dot{x} + \dot{x}^T\, Px$$

$$\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,) = x^T P\, (Ax) + (Ax)^T\, Px$$

$$\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,) = x^T P\, A\, x + x^T A^T\, Px$$

$$\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,) = x^T (P\, A + A^T\, P)x$$

$$Q = -(P\, A + A^T\, P)$$

Sustituyendo:

$$J = \int_0^\infty -\frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x\,)dt = -x^T P\, x|_0^\infty = \, -x(\infty)^T P\, x(\infty) + x(0)^T P\, x(0)$$

Como el sistema es siempre estable: $x(\infty) \to 0$

El índice de desempeño:

$$J = x(0)^T P\, x(0)$$

El sistema en lazo cerrado:

$$\dot{x} = \left(A - BK_{lqr}\right)x + Br$$

$$Br \to 0$$

$$\overline{A} = A - BK_{lqr}$$

$$\dot{x} = \overline{A}x$$

K_{lqr} es tal que el sistema en lazo cerrado es estable.

Sustituyendo la ley de control en J:

$$J = \int_0^\infty \left(x^T Q\, x + \left(-K_{lqr}x\right)^T R\, \left(-K_{lqr}x\right)\right)dt$$

$$J = \int_0^\infty \left(x^T Q\, x + {K_{lqr}}^T x^T R\, K_{lqr}\, x\right) dt$$

$$J = \int_0^\infty \left(x^T \left(Q + {K_{lqr}}^T R\, K_{lqr}\right) x\right) dt$$

$$\overline{Q} = Q + {K_{lqr}}^T R\, K_{lqr}$$

Para el sistema en lazo cerrado:

$$-x^T \overline{Q} x = \frac{\mathrm{d}}{\mathrm{dt}}(x^T P\, x) = x^T \left(P\, \overline{A} + \overline{A}^T\, P\right) x$$

$$P\, \overline{A} + \overline{A}^T\, P = -\left(Q + {K_{lqr}}^T R\, K_{lqr}\right)$$

$$J = \int_0^\infty -\frac{\mathrm{d}}{\mathrm{dt}}\left(x^T \overline{Q}\, x\right) dt = -x^T P\, x|_0^\infty = -x(\infty)^T P\, x(\infty) + x(0)^T P\, x(0)$$

$$J = x(0)^T P\, x(0)$$

Desarrollando $P\, \overline{A} + \overline{A}^T\, P = -\overline{Q}$

$$P\left(A - BK_{lqr}\right) + \left(A - BK_{lqr}\right)^T P = -\left(Q + {K_{lqr}}^T R\, K_{lqr}\right)$$

Se asume que $R = T^T T$

$$PA - PBK_{lqr} + A^T P - {K_{lqr}}^T B^T\, P = -Q - {K_{lqr}}^T T^T T\, K_{lqr}$$

$$PA + A^T P - PBK_{lqr} - {K_{lqr}}^T B^T\, P + {K_{lqr}}^T T^T T\, K_{lqr} = -Q$$

Utilizando la relación:

$$x^T x$$

$$[S\, I - M]^T [S\, I - M]$$

Entonces:

$$PA + A^T P + \left[TK_{lqr} - (T^T)^{-1} B^T P\right]^T \left[TK_{lqr} - (T^T)^{-1} B^T P\right] = -Q$$

$$PA + A^T P - PBR^{-1} B^T P = -Q$$

Si la expresión dentro del corchete vale cero:

$$TK_{lqr} - (T^T)^{-1} B^T P = 0$$

$$K_{lqr} = T^{-1} (T^T)^{-1} B^T P$$

$$K_{lqr} = (T^T T)^{-1} B^T P$$

Finalmente:

$$K_{lqr} = R^{-1} B^T P$$

$$PA + A^T P - PBR^{-1} B^T P = -Q$$

Afortunadamente Matlab® ofrece un comando que calcula los parámetros del control LQR de un sistema en representación de espacio de estados y proporcionado las matrices Q y R:

$$lqr(A, B, Q, R)$$

Recordando que:

$$Q = Q^T > 0$$

$$R = R^T > 0$$

También se aplicará una matriz de ganancia a las señales de referencia, quedando la arquitectura del control de la siguiente forma:

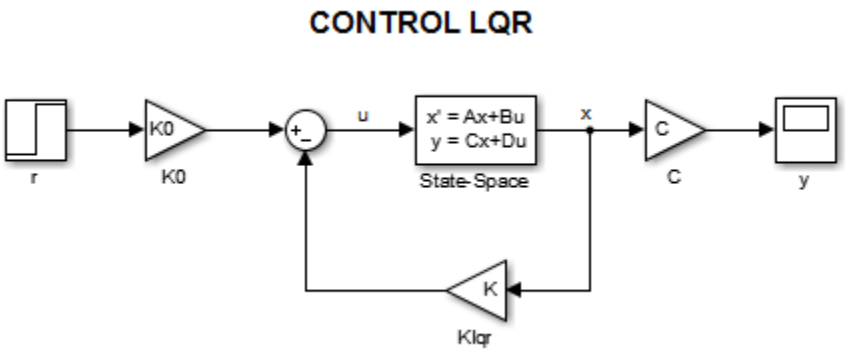


Figura 3.14: Control LQR Con Ganancia De Entradas

Donde:

$$K0 = \left[C(-A + BK_{lqr})^{-1} B\right]^{-1}$$

3.6 SIMULACIÓN DEL CONTROLADOR

Para la representación y simulación de los modelos del TWIP, así como el diseño y simulación del controlador se utilizó el software de Simulink® de Matlab®.

3.6.1 Comparación Del Modelo No Lineal Con El Modelo Lineal

Representando las ecuaciones no lineales del sistema en un diagrama de bloques en Simulink®:

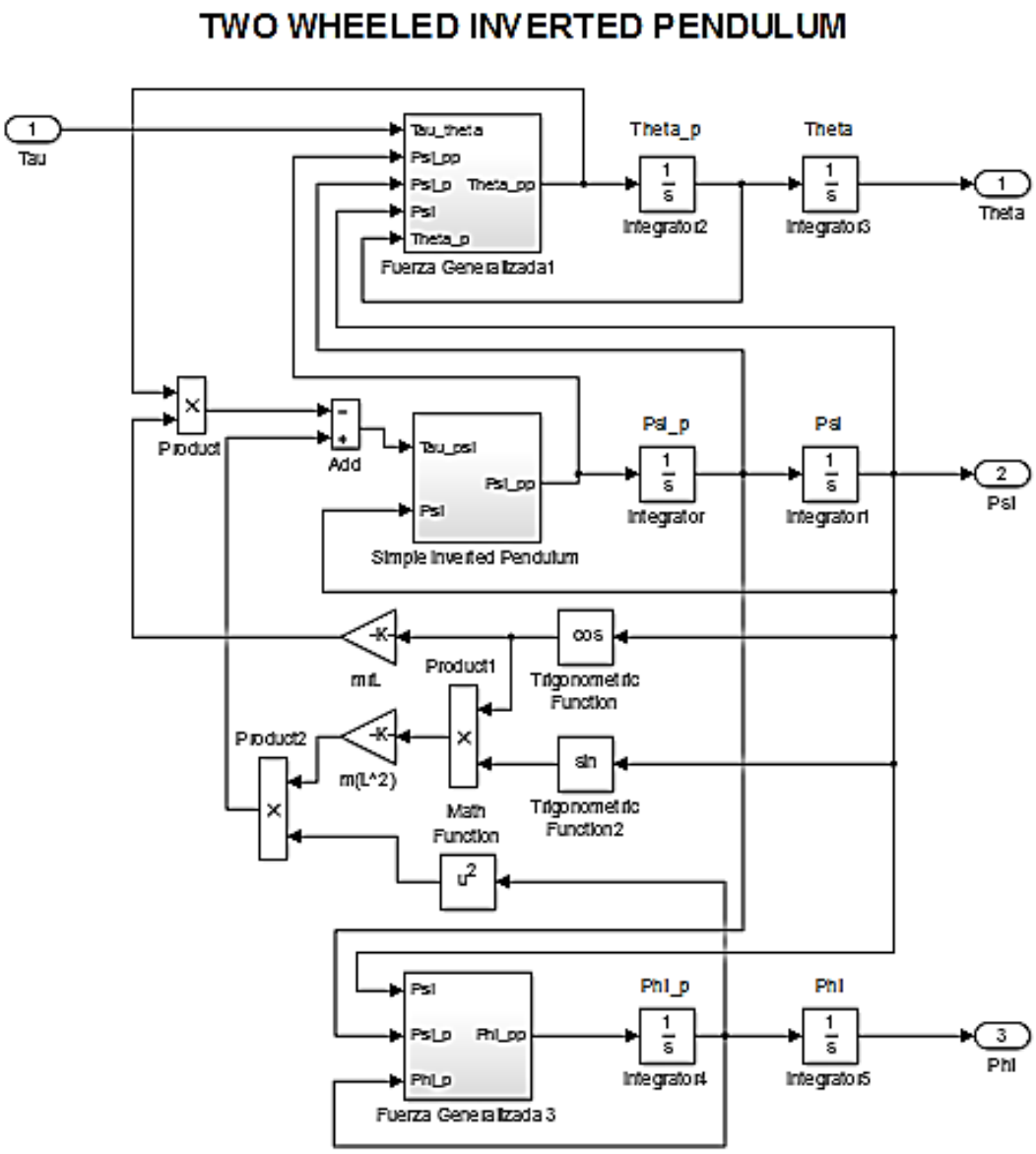


Figura 3.15: Diagrama De Bloques 1 (Modelo No Lineal)

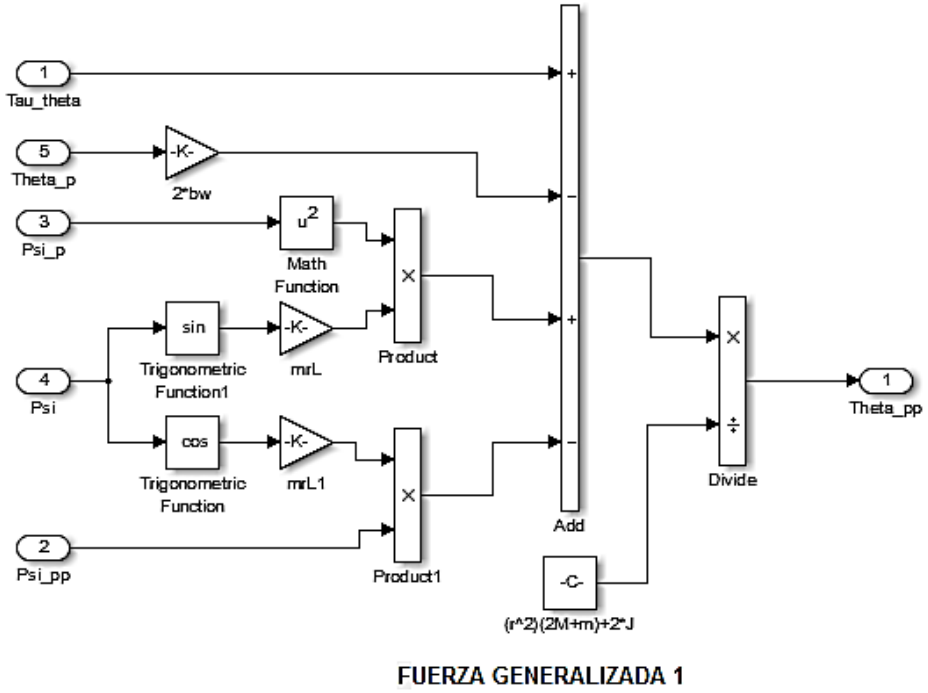


Figura 3.16: Diagrama De Bloques 2 (Modelo No Lineal)

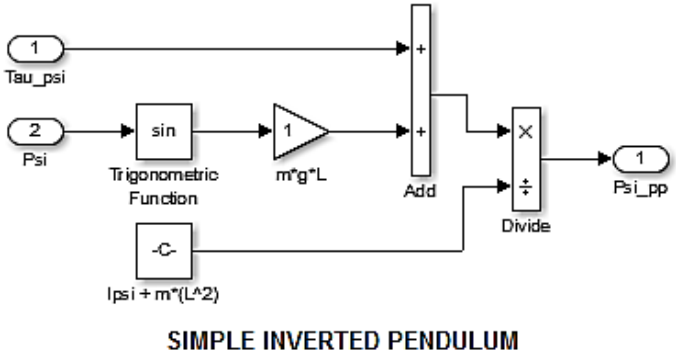


Figura 3.17: Diagrama De Bloques 3 (Modelo No Lineal)

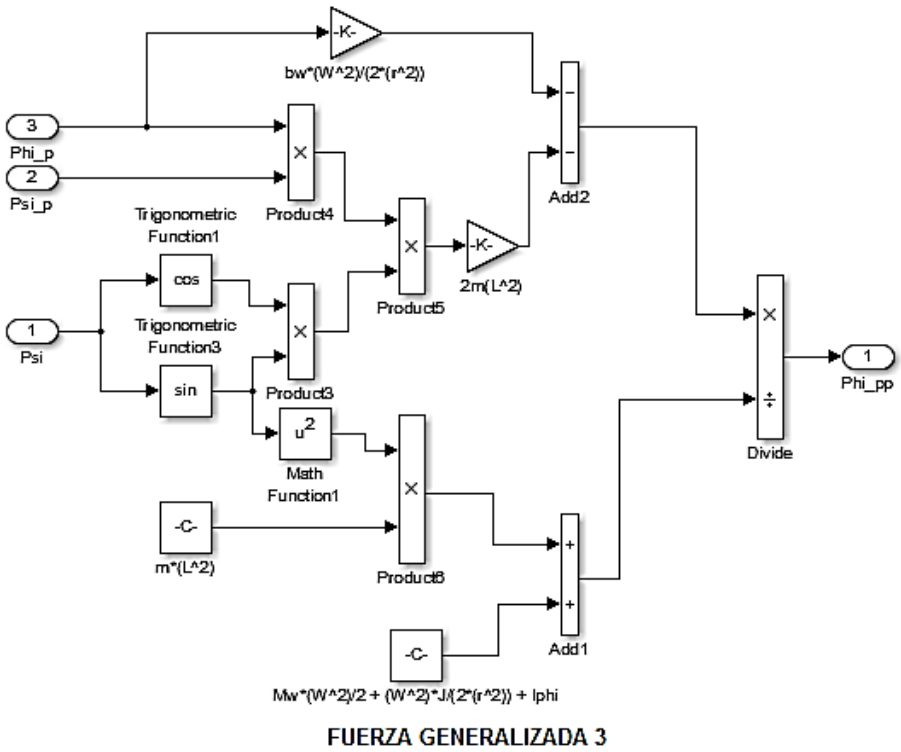


Figura 3.18: Diagrama De Bloques 4 (Modelo No Lineal)

La representación es espacio de estados del sistema lineal queda:

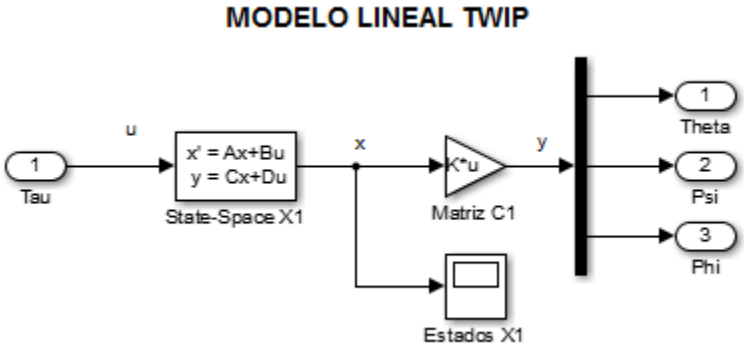


Figura 3.19: Diagrama De Bloques (Modelo Lineal)

3.6.2 Simulación En Lazo Abierto

Se simularon tanto el modelo no lineal, como el lineal, graficando la posición del ángulo ψ (psi), que como se mencionó con anterioridad representa el ángulo respecto de la vertical del péndulo invertido.

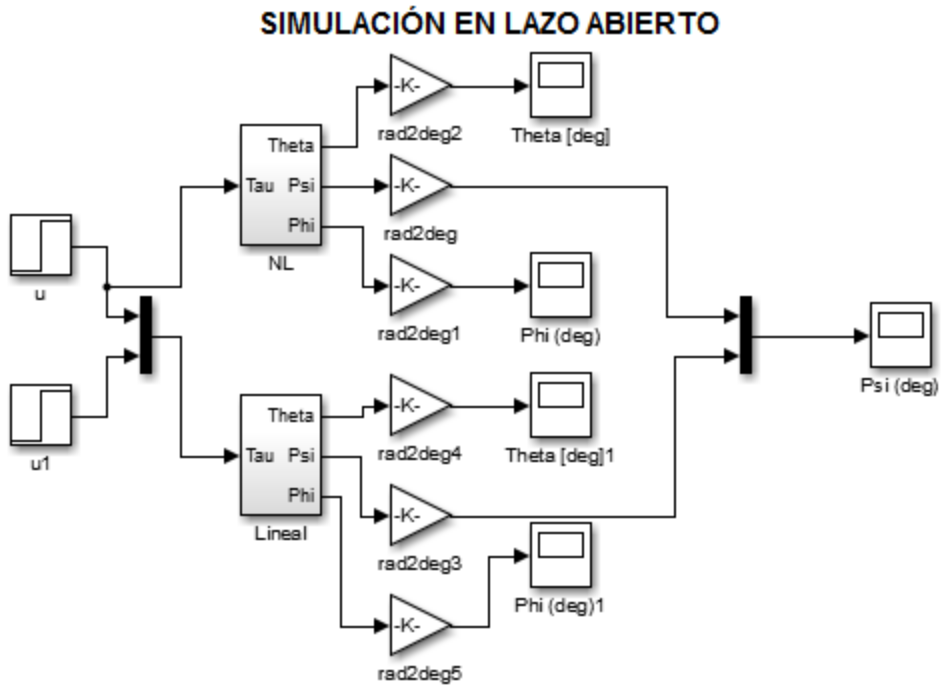


Figura 3.20: Diagrama De Bloques (Simulación En Lazo Abierto)

3.6.3 Comparación Del Modelo No Lineal Con La Planta Real

Para poder continuar con el proceso de diseño del controlador, debemos asegurarnos de que el modelo no lineal obtenido se asemeje al comportamiento real de nuestro TWIP, una vez instrumentado el TWIP, se mandaron en tiempo real las señales de los sensores al Simulink® para poder comparar el comportamiento con el modelo no lineal.

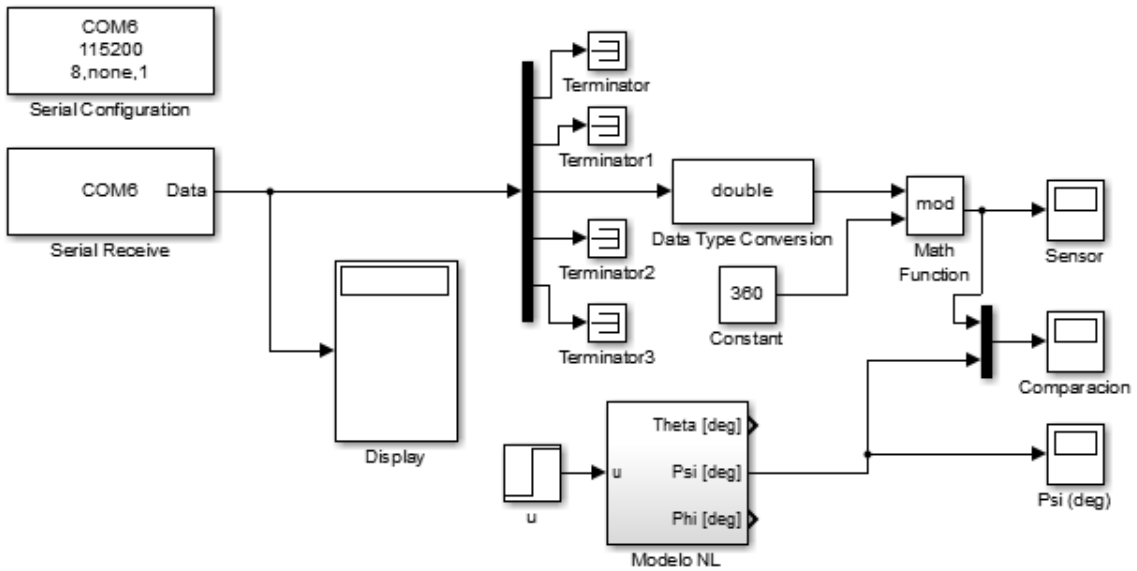


Figura 3.21: Diagrama De Bloques (Planta Real)

Donde se puede apreciar el bloque de inicialización de comunicación serial, el bloque de recepción de datos

3.7 SIMULACIÓN DEL CONTROLADOR LQR

La arquitectura final del control LQR para el sistema de sexto orden se muestra en la siguiente figura:

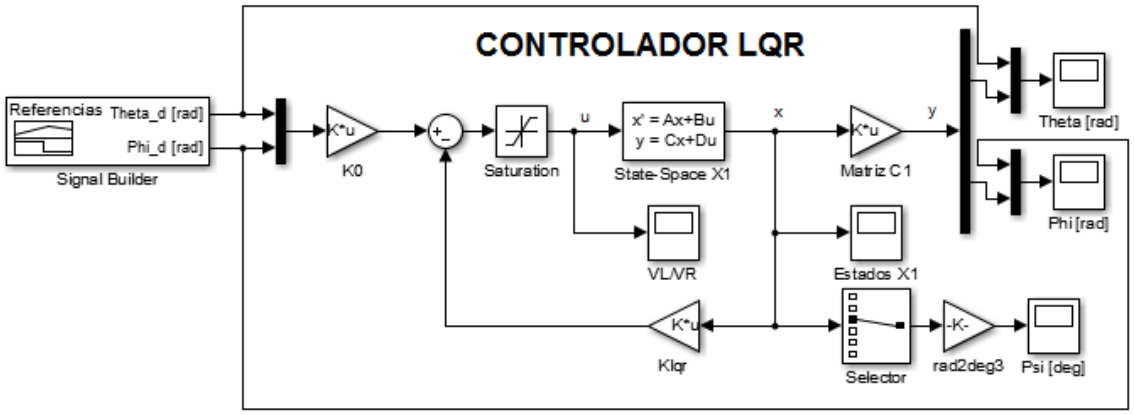


Figura 3.22: Diagrama De Bloques (Simulación Control LQR Sistema 6to Orden)

Rápidamente podemos encontrar todos los elementos necesarios para llevar a cabo la simulación del controlador LQR:

- Tenemos las señales de referencia para las posiciones angulares deseadas.
- La matriz de ganancia K0,
- Una saturación para la señal de control, la cual está definida de -15 a +15, debido a que las baterías del TWIP no pueden dar más que estos voltajes.
- Gráficas de las señales de control (voltajes para el motor izquierdo y para el derecho).
- Espacio de estados que representa el modelo lineal del TWIP.
- La matriz de ganancia K_{lqr}
- Gráfica de los estados.
- Gráfica del ángulo ψ (Psi) para verificar que el control mantendrá en todo momento al péndulo en su posición vertical.
- Comparación de las posiciones angulares de referencia con las posiciones del controlador.

El código que habilita la simulación se encuentra en el apéndice anexo a este trabajo.

La arquitectura final del control LQR para el sistema de cuarto orden se muestra en la siguiente figura:

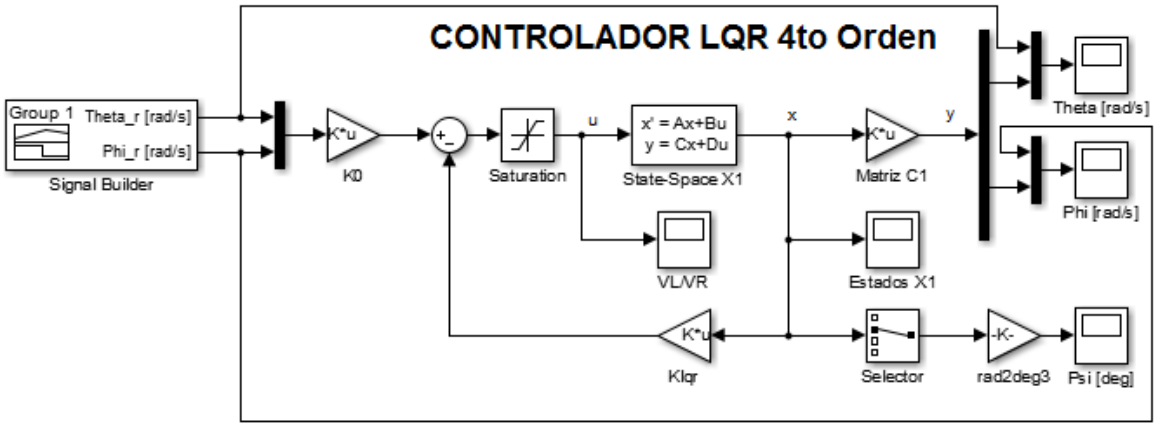


Figura 3.23: Diagrama De Bloques (Simulación Control LQR Sistema 4to Orden)

Tiene los mismos componentes que el diagrama de bloques anterior, las diferencias son las siguientes:

- El generador de señales de referencias son ahora velocidades angulares.
- El selector de señales tiene cuatro opciones y no seis, de la cual el ángulo Psi se obtiene seleccionando el segundo dato y no el tercero, que es el caso del sistema de sexto orden.
- Las salidas del sistema son las velocidades angulares, no las posiciones angulares.

El código que habilita la simulación se encuentra en el apéndice anexo a este trabajo.

4 PRUEBAS Y RESULTADOS

4.1 PRUEBAS

En este apartado se presentan todas las pruebas necesarias para lograr el control del TWIP.

4.1.1 Simulación En Lazo Abierto

Con esta prueba se busca verificar que el comportamiento del modelo no lineal coincida a simple vista con el comportamiento del TWIP, y conocer la correspondencia que existe entre el modelo no lineal y el modelo lineal.

Al comparar ambos modelos en una prueba de lazo abierto, es decir, sin acción de control, se obtuvo la siguiente gráfica:

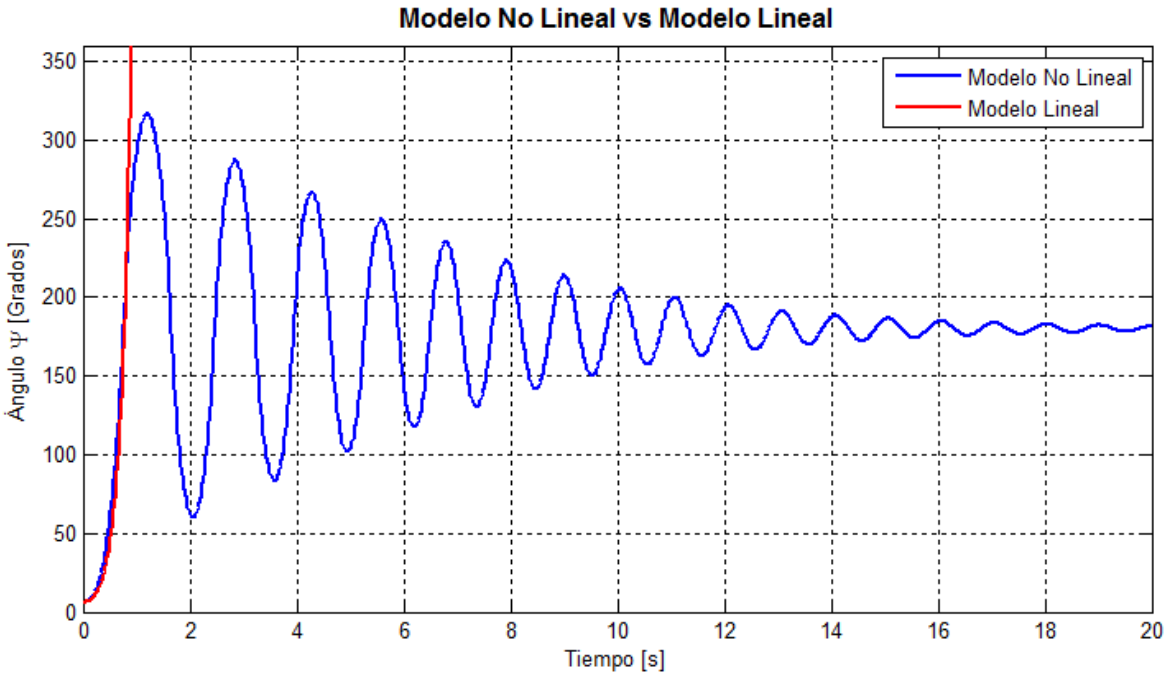


Figura 4.1: Gráfica - Modelo No Lineal vs Modelo Lineal

Los parámetros para simulaciones incluyeron un ángulo inicial ψ = 6° y entradas igual a cero, con un tiempo de simulación de 20 segundos.

Donde la gráfica azul representa el comportamiento de ψ en el modelo no lineal y la gráfica roja representa el comportamiento de ψ en el modelo lineal.

El modelo no lineal muestra como el péndulo invertido cae del punto inicio (6°) hasta su punto de equilibrio inferior (180°), oscilando cada vez menos hasta que eventualmente permanecerá en equilibrio.

Rápidamente se puede apreciar que el modelo lineal sigue de manera casi perfecta al modelo no lineal hasta cierto punto donde ya no puede seguir representando la dinámica del sistema y entonces se pierde.

Se puede concluir que este modelo lineal será útil para la implementación del TWIP.

4.1.2 Modelo No Lineal VS Planta Real

Esta prueba tiene como finalidad asegurar que el modelo no lineal planteado en el capítulo 2 de este trabajo funcionará como base para el diseño del controlador.

Como condición inicial para el modelo no lineal se propuso 170° y leyendo el sensor se colocó el TWIP en posición de 170° y se arrancó la simulación, soltando simultáneamente el TWIP para la comparación generando la siguiente gráfica:

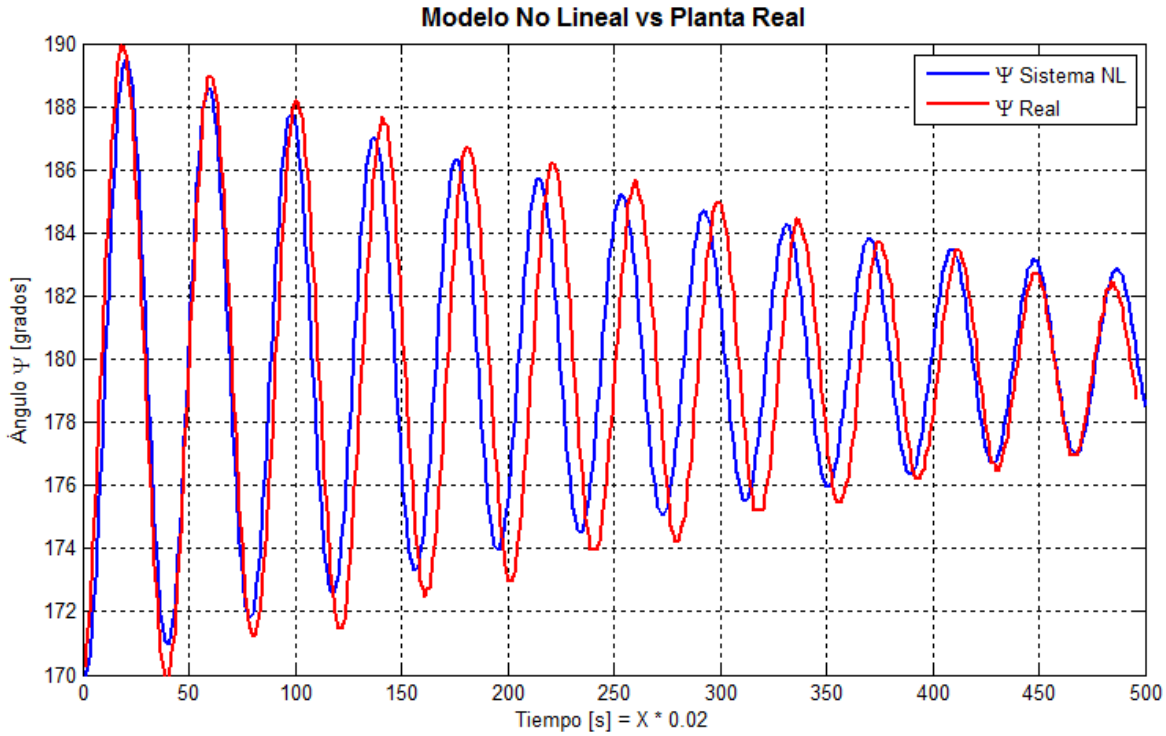


Figura 4.2: Gráfica - Modelo No Lineal vs Planta Real

La gráfica azul representa el comportamiento del TWIP real, a través de la lectura de la IMU, mientras que la gráfica roja representa el modelo no lineal simulado.

Debido a que no se logró soltar el TWIP en el momento exacto en el que arranca la simulación, por medio de Matlab® se ajustó la gráfica que representa el sistema real para que cuando pasara por 170° quedara sincronizada con la gráfica de la simulación del sistema no lineal y poder compararlas con facilidad, aunque no es lo indicado pues no se pueden asegurar las mismas condiciones iniciales en la planta real que en el modelo no lineal, creo que es una buena aproximación con la instrumentación que se tiene.

Se aprecia que empieza a haber un desfase entre las señales conforme avanza el tiempo, esto debido a las incertidumbres paramétricas que todavía existen en el modelo, como las fricciones, las cuales fueron ajustadas y no calculadas.

A continuación se presenta el error entre ambas señales:

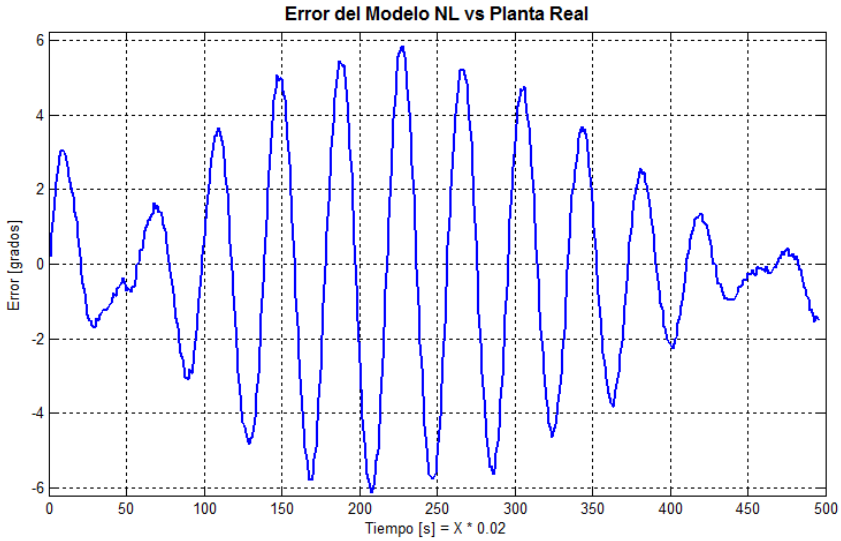


Figura 4.3: Gráfica – Error Del Modelo No Lineal vs Sistema Real

Se puede notar un comportamiento muy peculiar en el error, debido al desfase que empieza a existir entre las señales, el error empieza a aumentar y luego a disminuir pero jamás rebasa los 6° de error, lo que beneficiará al controlador en gran medida.

Cabe hacer notar que para el cálculo de esta diferencia, el número de muestras y tiempo de muestreo fue el mismo para ambas señales.

4.1.3 Control LQR Para El Sistema De Sexto Orden

Como primera aproximación del control LQR, se propuso un peso para las variables de la siguiente forma:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6e3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Enfocando toda la atención del controlador a mantener la posición vertical del péndulo, mientras que R por simplicidad se eligió una matriz identidad.

Dando como ganancias resultantes:

$$K_{lqr} = \begin{bmatrix} -0.7071 & -2.6921 & -114.8326 & -16.4585 & -0.7071 & -0.1828 \\ -0.7071 & -2.6921 & -114.8326 & -16.4585 & +0.7171 & +0.1828 \end{bmatrix}$$

$$K0 = \begin{bmatrix} -0.7071 & -0.7071 \\ -0.7071 & +0.7071 \end{bmatrix}$$

Rápidamente se puede apreciar que la igualdad de términos es congruente, puesto que los dos motores deben recibir las mismas señales, excepto por los últimos términos, los cuales tienen signo opuesto, debido a que si se requiere de rotación en el TWIP, los motores deberán moverse opuestamente.

Como señales de referencia se propusieron las siguientes:

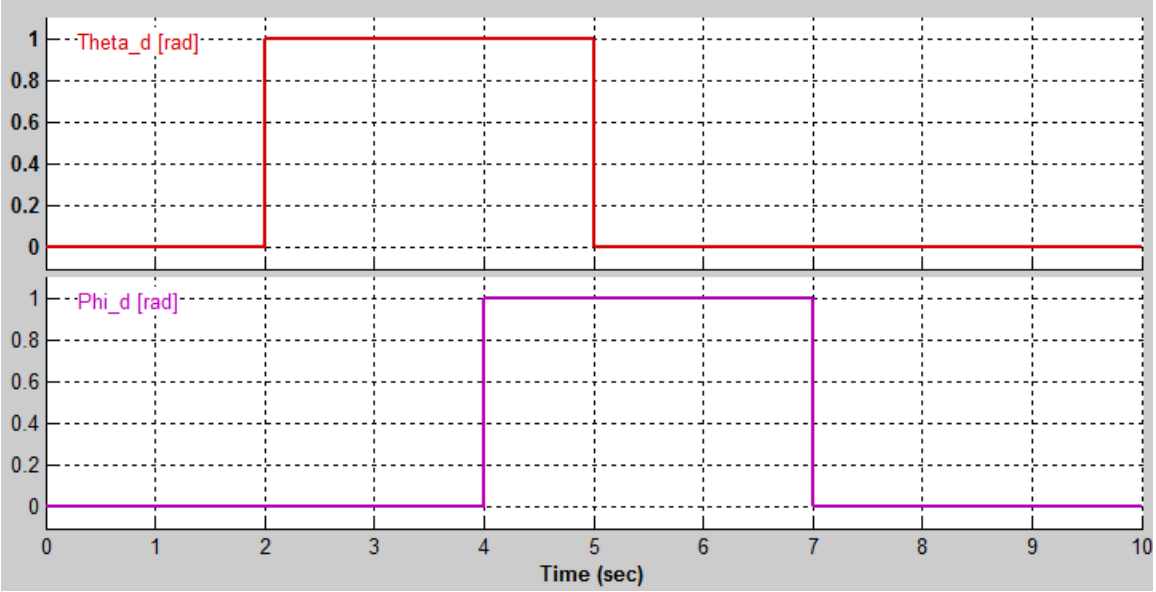


Figura 4.4: Gráfica – Señales De Referencia 1

Donde la gráfica roja representa la posición deseada de las ruedas, y la gráfica morada representa el ángulo de rotación deseada para el TWIP; se puede apreciar que las gráficas coinciden en el lapso de 4 a 5 segundos, esto con la finalidad de conocer cómo responderá el controlador ante tal tipo de situación.

Las condiciones iniciales son nulas con excepción del ángulo de inclinación, la cual fue de $\psi = 6°$. Al correr la simulación se obtienen los siguientes resultados:

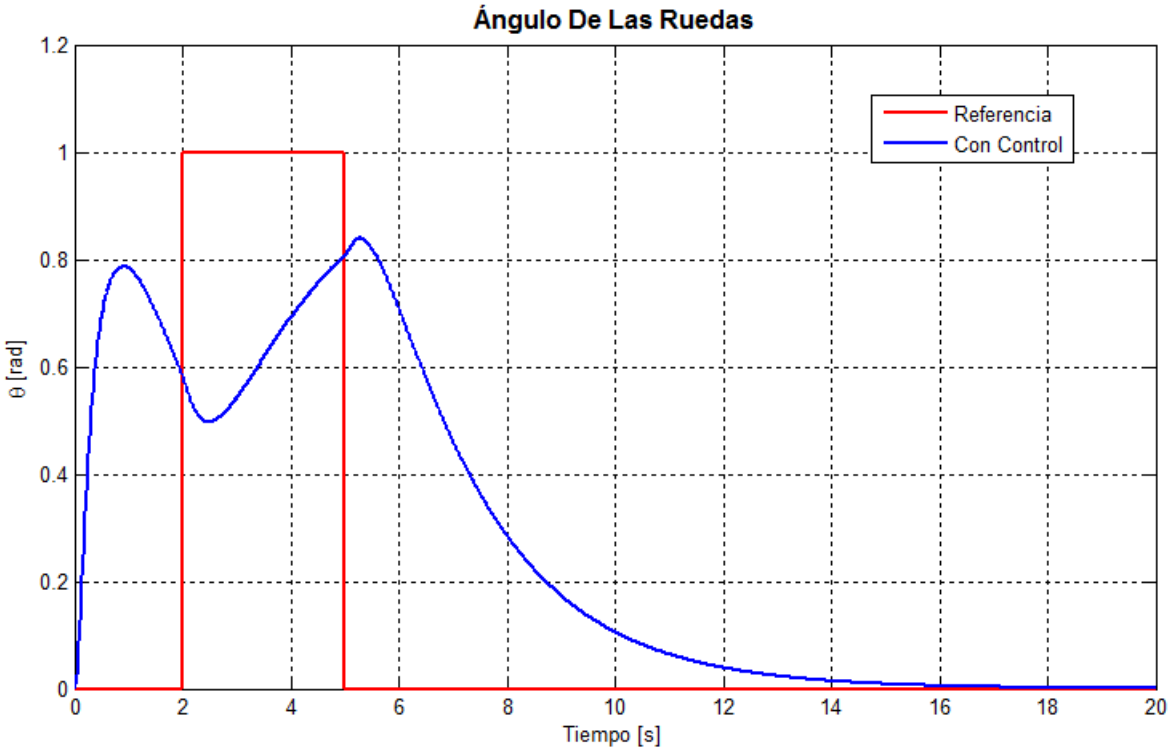


Figura 4.5: Gráfica – LQR 6to Orden (Theta) P1

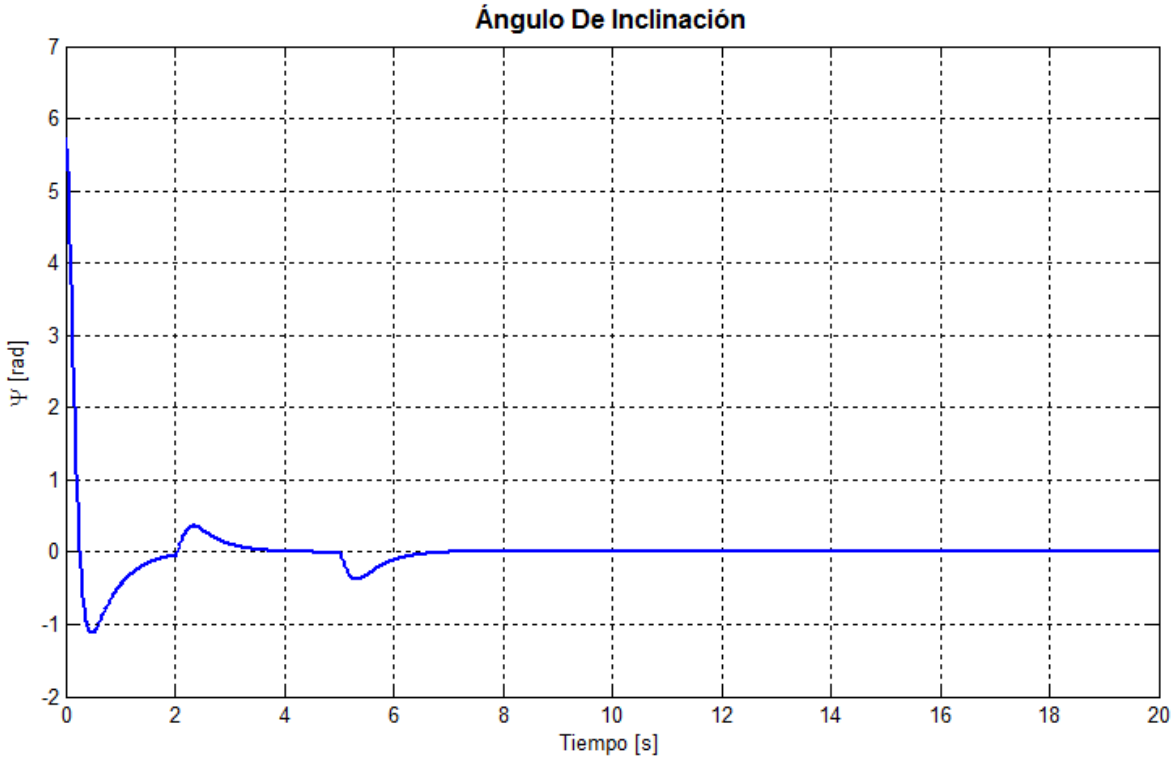


Figura 4.6: Gráfica – LQR 6to Orden (Psi) P1

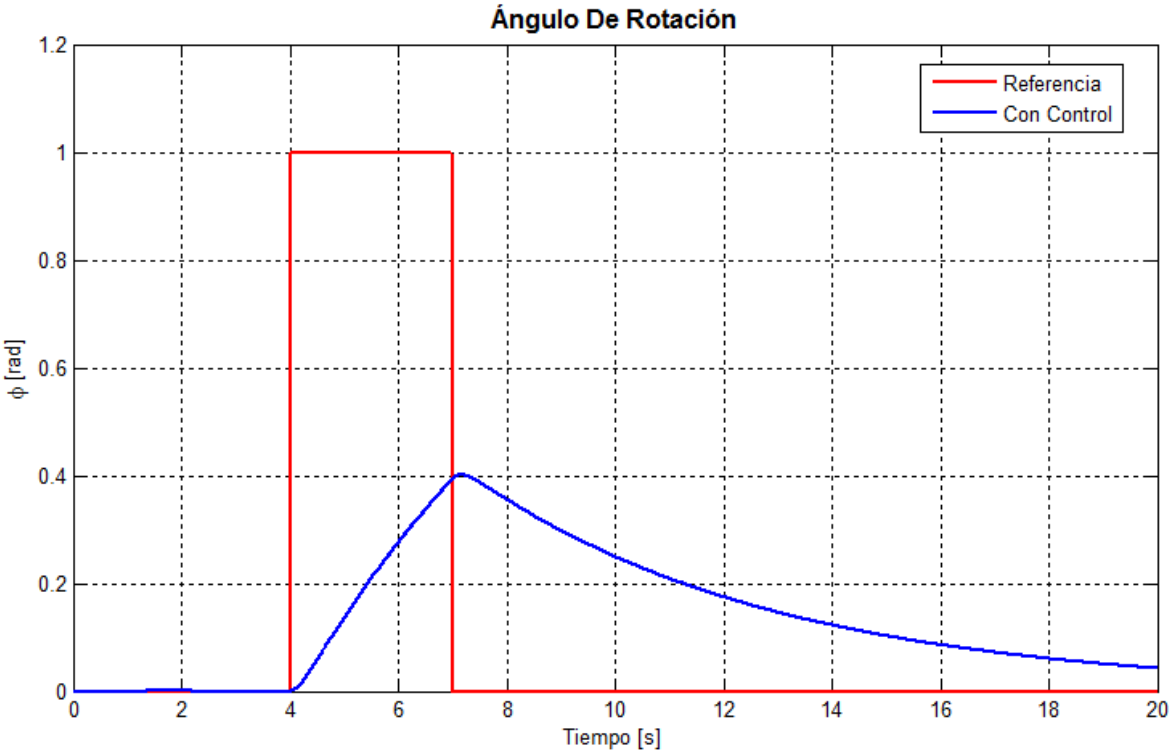


Figura 4.7: Gráfica – LQR 6to Orden (Phi) P1

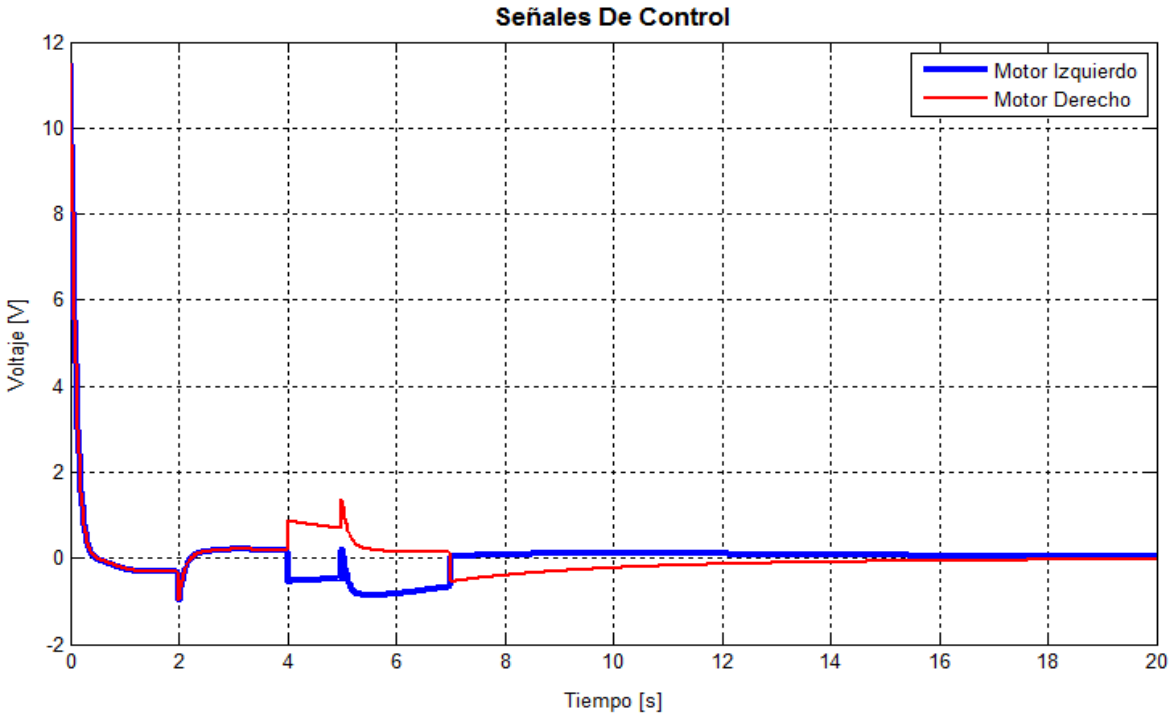


Figura 4.8: Gráfica – LQR 6to Orden (Voltajes) P1

Comparándolas todas juntas:

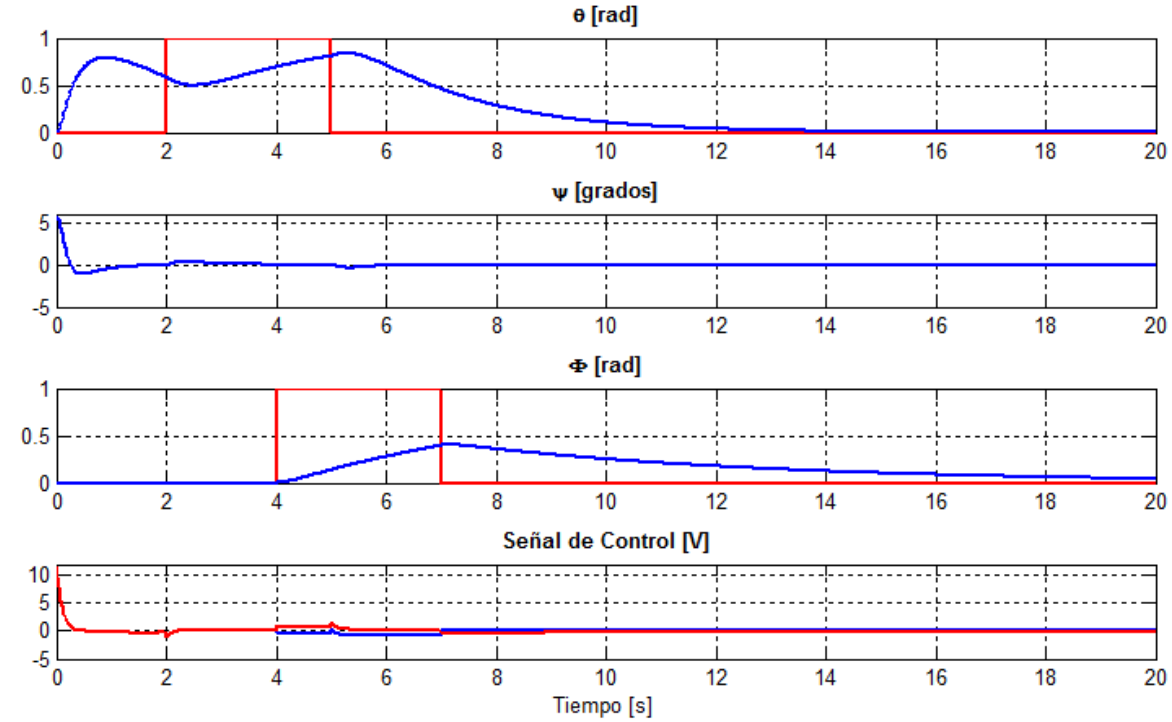


Figura 4.9: Gráficas – LQR 6to Orden (Todas) P1

Se puede apreciar que no puede alcanzar las referencias deseadas, pero que el ángulo de inclinación converge rápidamente a cero desde el inicio, las señales de entrada afectan mínimamente la posición vertical, esto debido a que como se mencionó con anterioridad, el peso

principal es para el ángulo de inclinación del TWIP, por lo que el controlador enfoca todos sus esfuerzos en esta variable sin importarle las demás.

Después se simuló la siguiente configuración de parámetros del controlador:

$$Q = \begin{bmatrix} 1e3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6e3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Se aumentaron los pesos para las variables θ y $\emptyset$, esto para asegurar que las referencias sean alcanzadas.

Obteniendo las siguientes ganancias:

$$K_{lqr} = \begin{bmatrix} -22.3607 & -11.7535 & -273.2140 & -44.1100 & -22.3607 & -2.8915 \\ -22.3607 & -11.7535 & -273.2140 & -44.1100 & +22.3607 & +2.8915 \end{bmatrix}$$

$$K0 = \begin{bmatrix} -22.3607 & -22.3607 \\ -22.3607 & +22.3607 \end{bmatrix}$$

Con las mismas condiciones iniciales y señales de referencia del caso anterior, se obtuvieron los siguientes resultados:

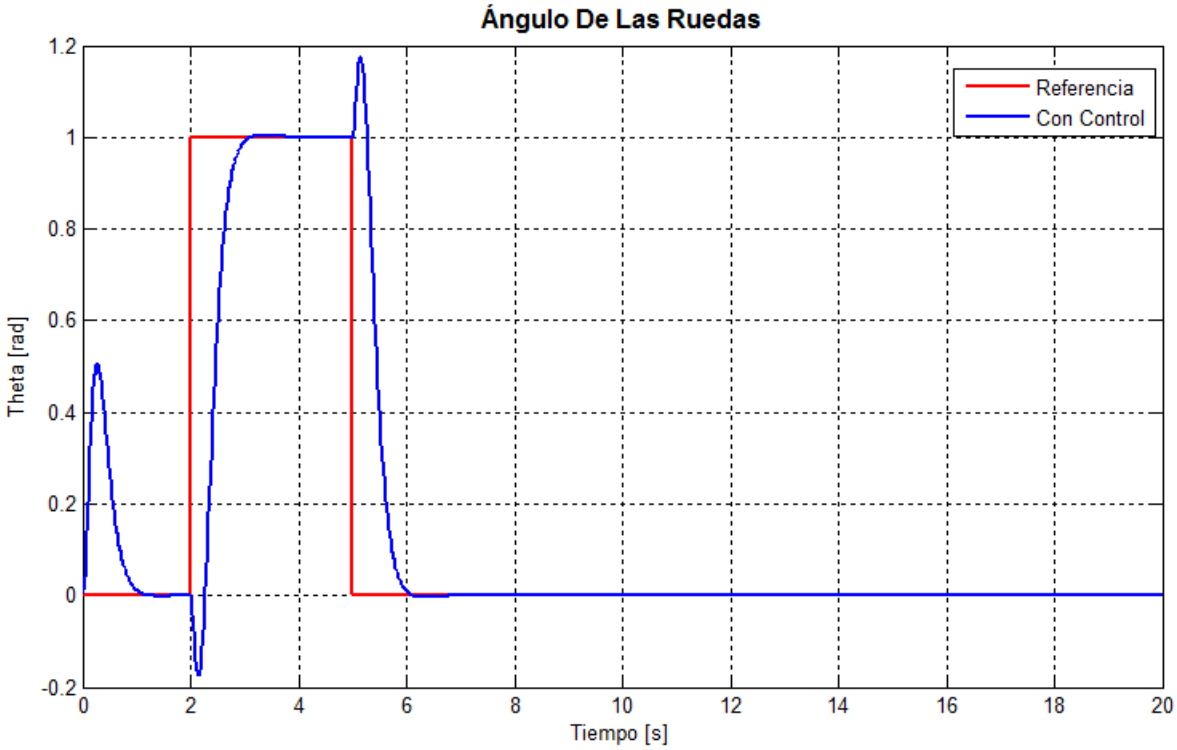


Figura 4.10: Gráfica – LQR 6to Orden (Theta) P2

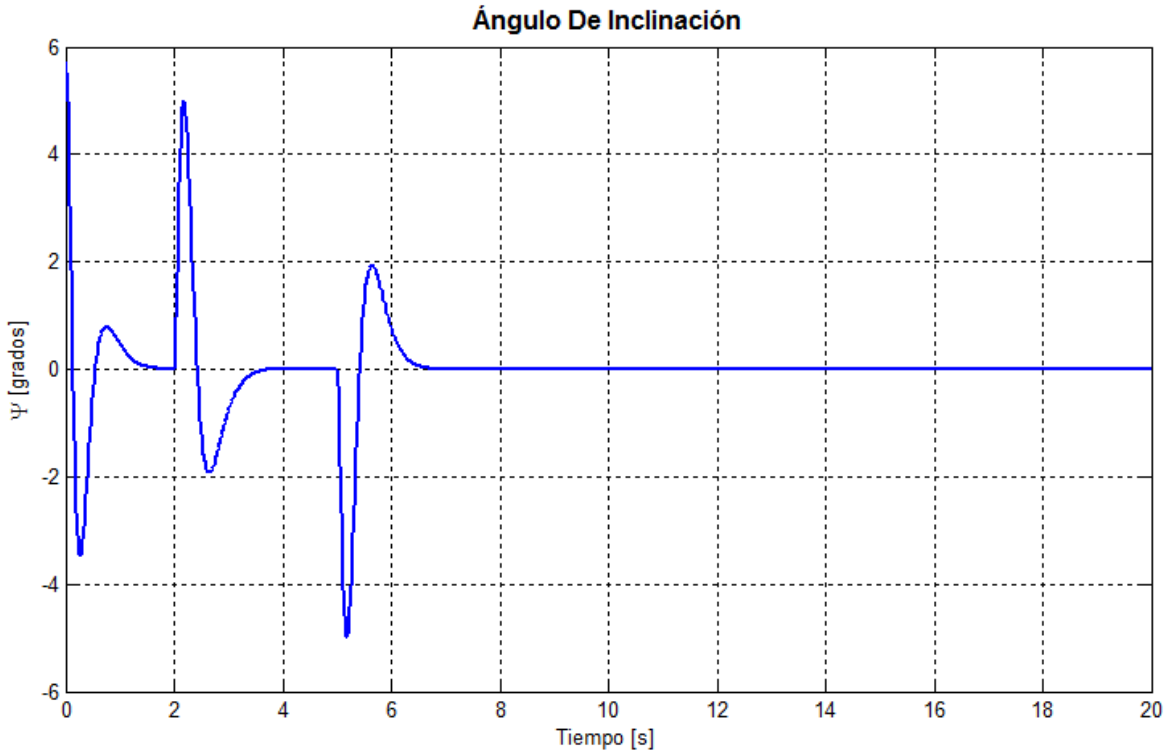


Figura 4.11: Gráfica – LQR 6to Orden (Psi) P2

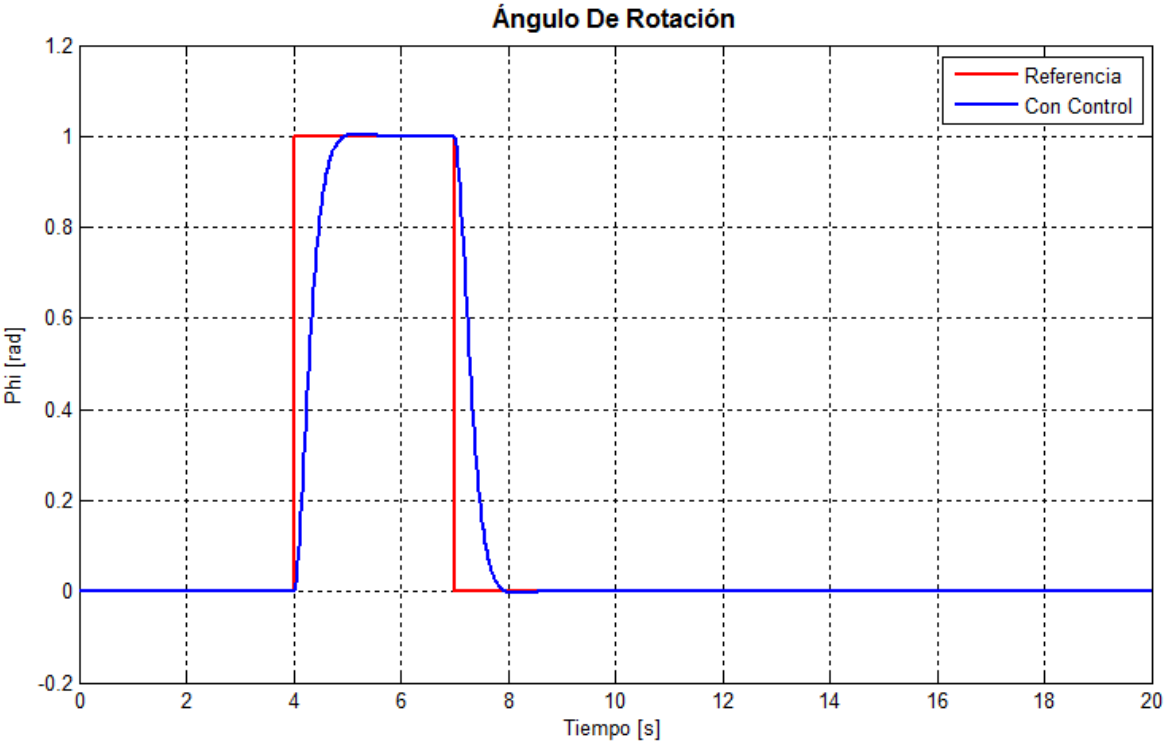


Figura 4.12: Gráfica – LQR 6to Orden (Phi) P2

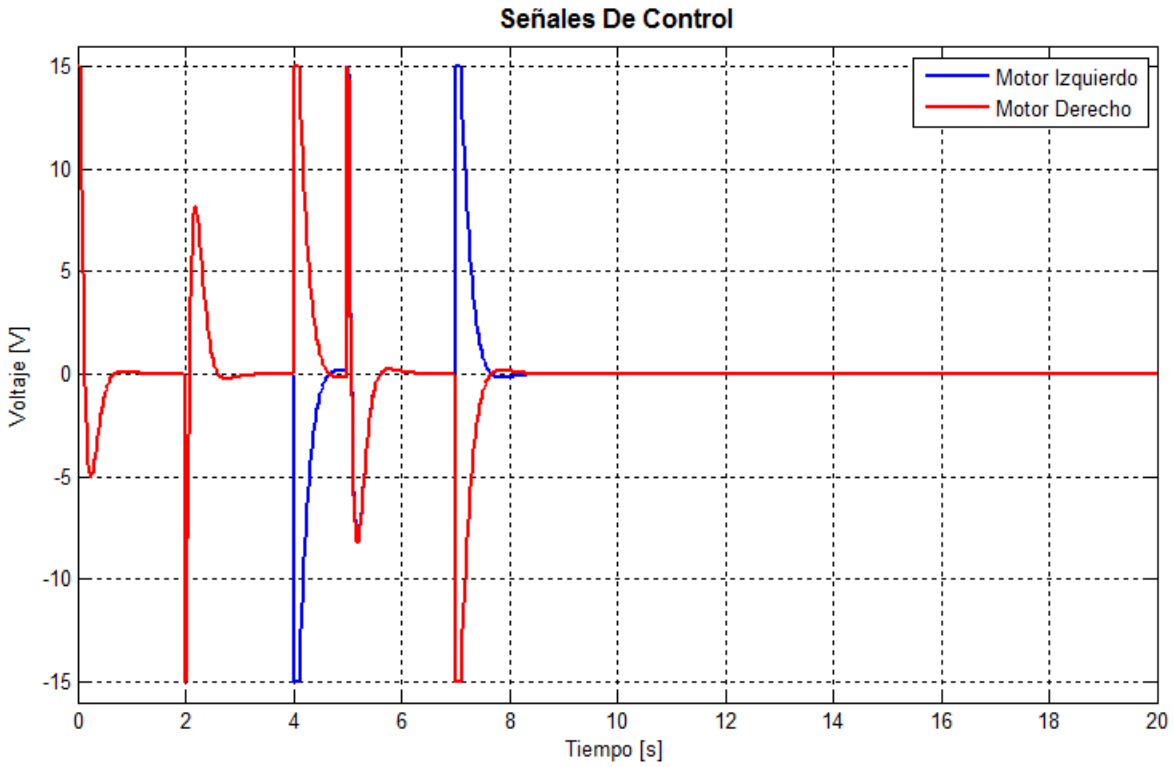


Figura 4.13: Gráfica – LQR 6to Orden (Voltajes) P2

Comparándolas todas juntas:

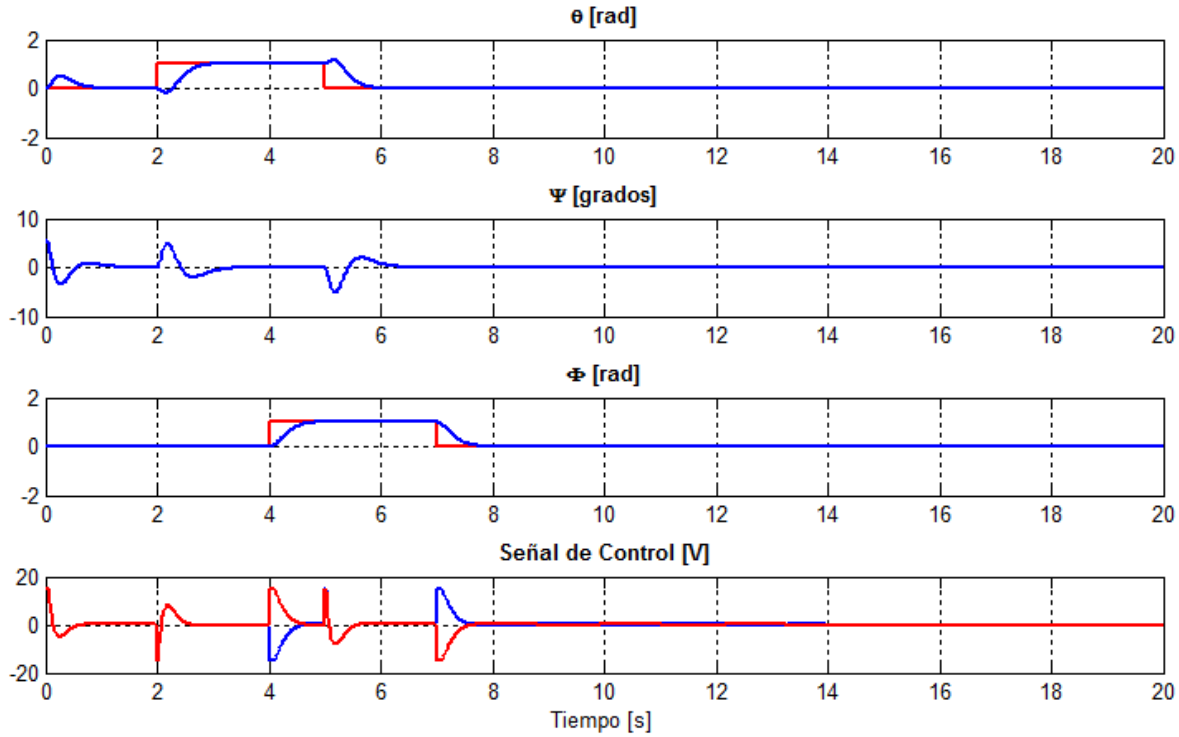


Figura 4.14: Gráficas – LQR 6to Orden (Todas) P2

Con esta configuración de parámetros, sí se logran alcanzar las referencias marcadas por las señales de entrada, pero se aprecian sobrepasos en las señales del θ y ψ, lo cual es lógico pues las ruedas tienen que compensar el movimiento del péndulo y además hacer rotar todo el TWIP cuando sea necesario, mientras que la inclinación del TWIP se ve afectada por el movimiento de las ruedas.

También se puede notar que se alcanzan las referencias relativamente rápido debido a que las señales de control se saturan rápidamente hasta los 15 [V], cabe hacer notar que la saturación de las señales puede causar efectos desfavorables en el sistema.

En el tiempo de 5 segundos cuando tan solamente existen 5° de inclinación se manda una señal de 15 [V] para compensar el error, esto no necesariamente es algo bueno, pues se pueden generar inestabilidades en el controlador como el fenómeno conocido como "Chattering".

4.1.4 Control LQR Para El Sistema De Cuarto Orden

Al igual que para el sistema de sexto orden las condiciones iniciales son nulas con excepción del ángulo de inclinación, la cual fue de $\psi = 6°$ y las señales de referencia en lugar de ser posiciones angulares, ahora son las velocidades angulares:

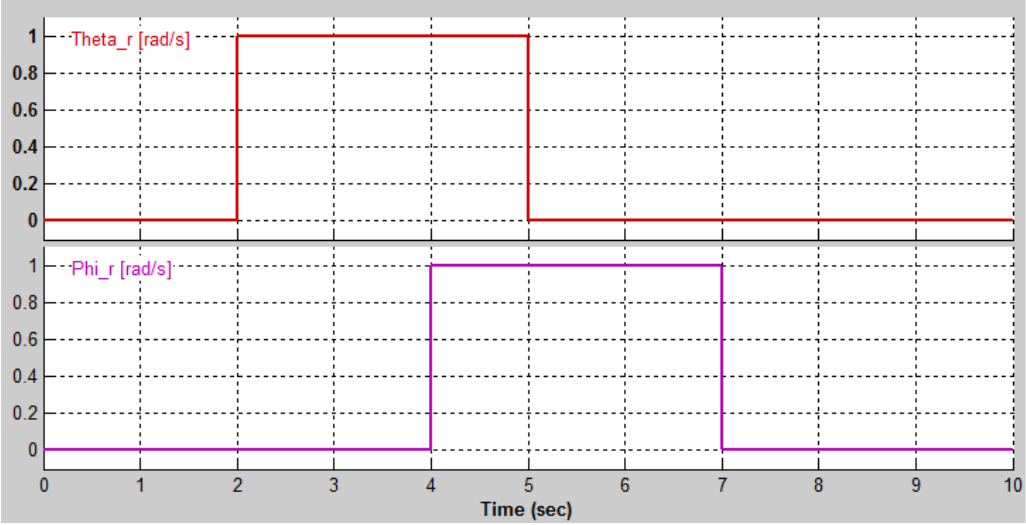


Figura 4.15: Señales De Referencia 2

En la primera aproximación del control LQR para el sistema de cuarto orden, se propuso un peso para las variables de la siguiente forma:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 6e3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Enfocando toda la atención del controlador a mantener la posición vertical del péndulo, mientras que R por simplicidad se eligió una matriz identidad.

Dando como ganancias resultantes:

$$K_{lqr} = \begin{bmatrix} -2.2608 & -107.4008 & -15.0564 & -0.0006 \\ -2.2608 & -107.4008 & -15.0564 & +0.0006 \end{bmatrix}$$

$$K0 = \begin{bmatrix} -1.4565 & -3.9771 \\ -1.4565 & +3.9771 \end{bmatrix}$$

Una vez más se puede apreciar que la igualdad de términos es congruente, puesto que los dos motores deben recibir las mismas señales, excepto por los últimos términos, los cuales tienen signo opuesto, debido a que si se requiere de rotación en el TWIP, los motores deberán moverse opuestamente.

Al correr la simulación se obtienen los siguientes resultados:

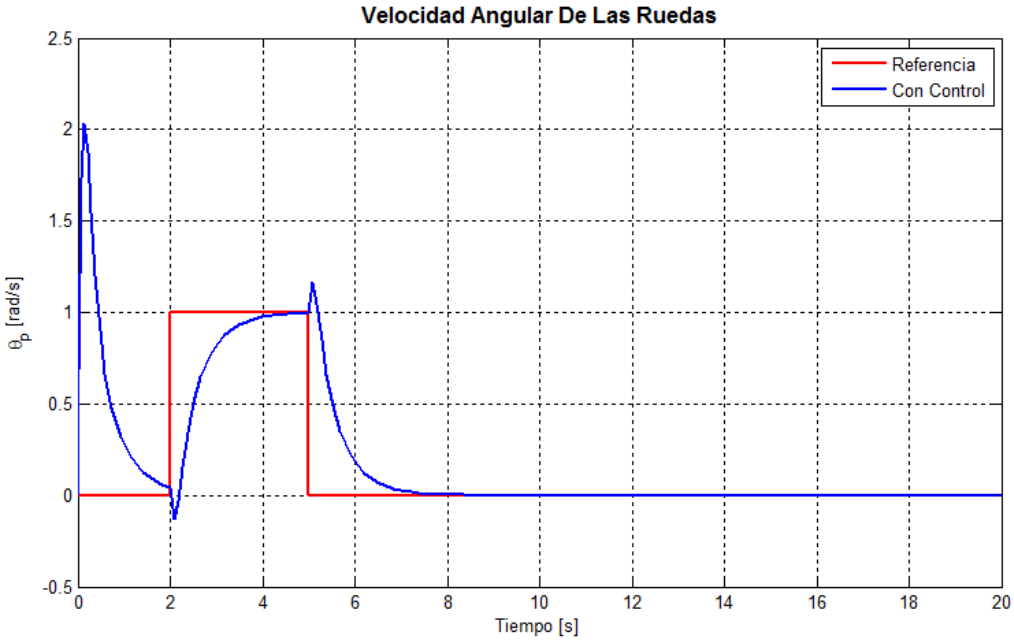


Figura 4.16: Gráfica – LQR 4to Orden (Theta_p) P1

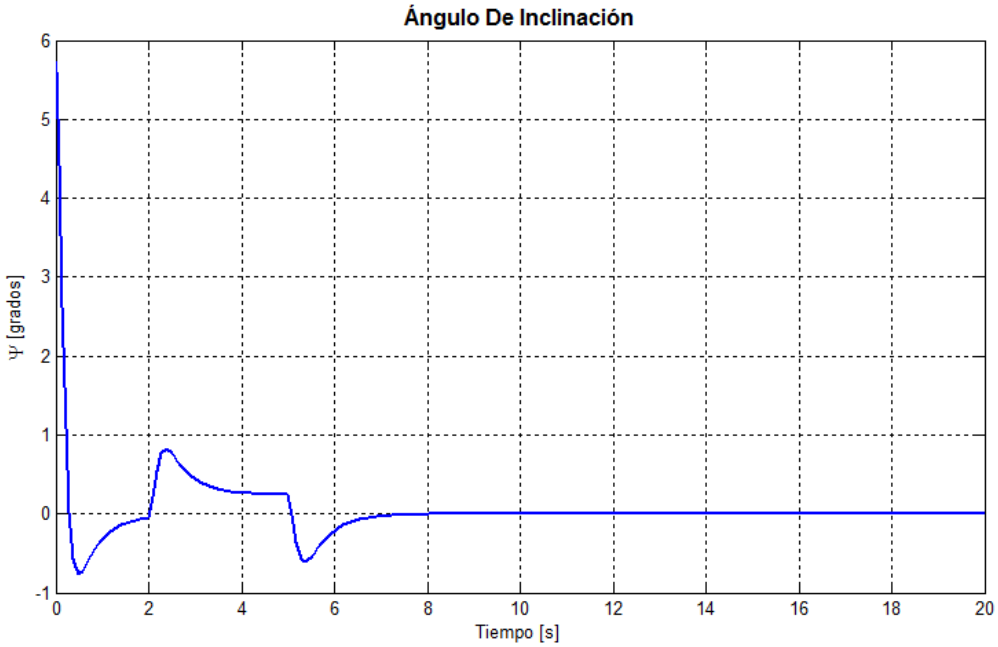


Figura 4.17: Gráfica – LQR 4to Orden (Psi) P1

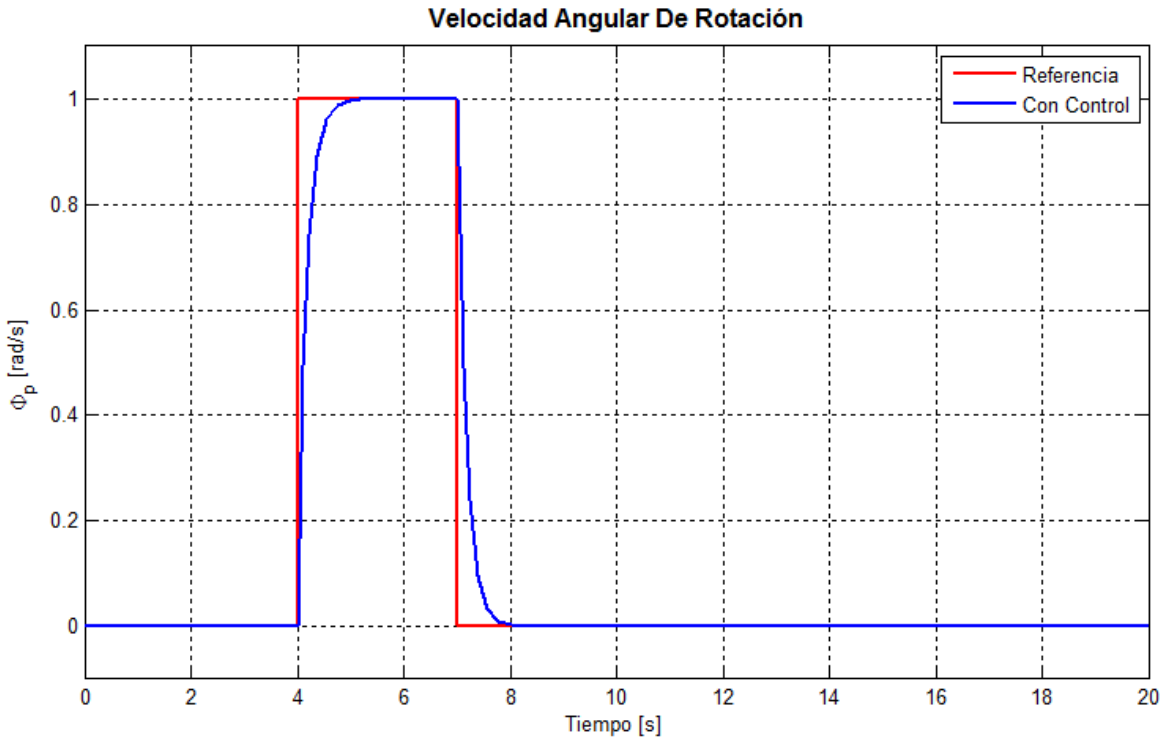


Figura 4.18: Gráfica – LQR 4to Orden (Phi_p) P1

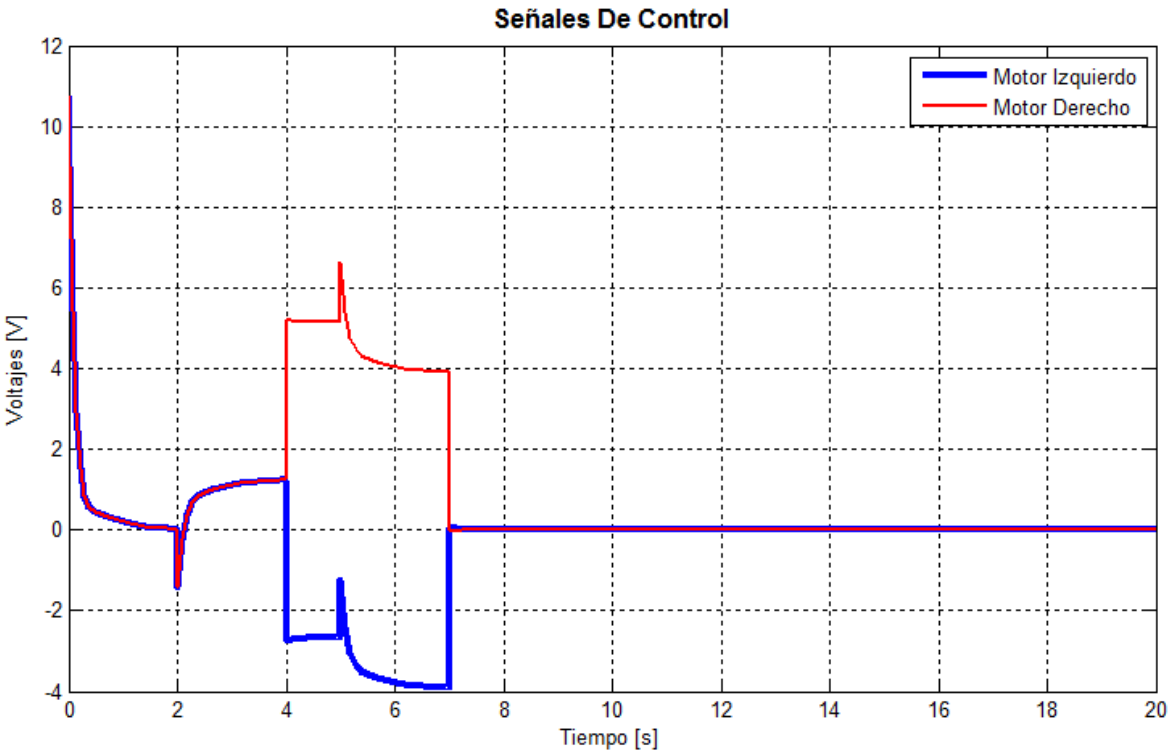


Figura 4.19: Gráfica – LQR 4to Orden (Voltajes) P1

Comparándolas todas juntas:

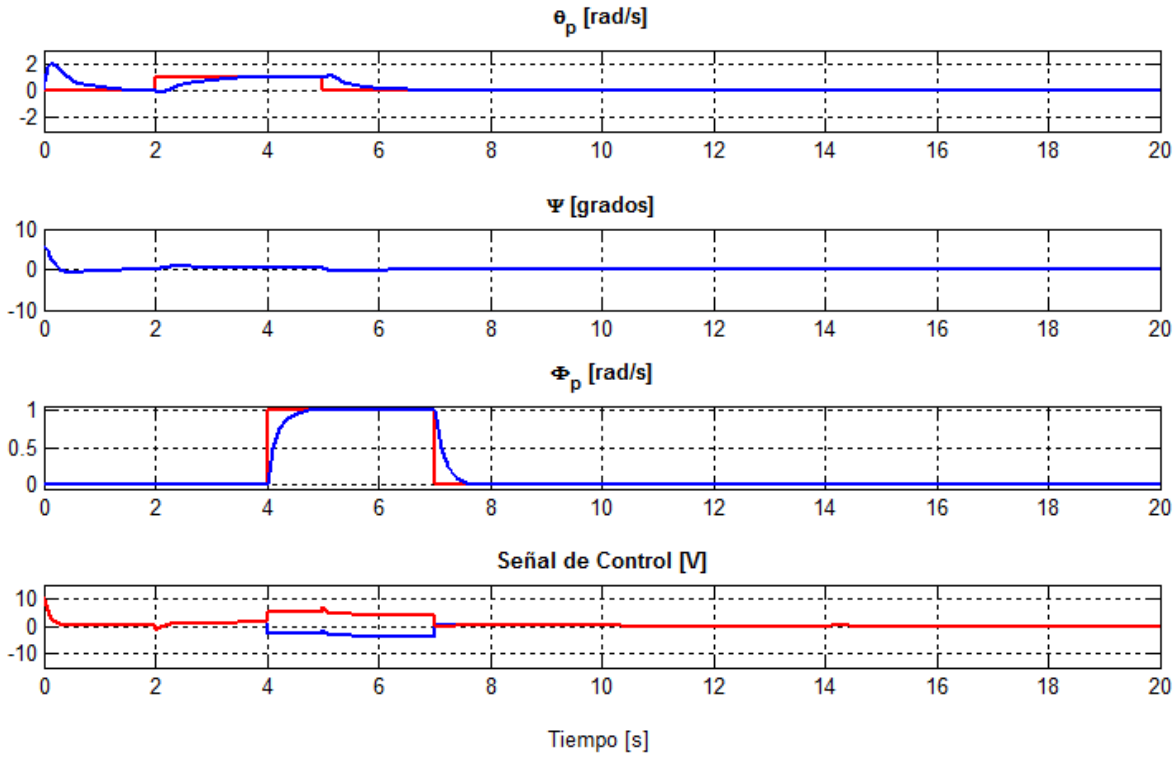


Figura 4.20: Gráficas – LQR 4to Orden (Todas) P1

Se puede apreciar que aun dando la mayor prioridad al ángulo de inclinación, las referencias son alcanzadas sin problema, y la señal de control tiene un buen comportamiento.

En la siguiente prueba, se intenta mejorar el uso de energía mientras también se logre llegar a las señales de referencia, esto ajustando la matriz R para conseguir un control menos drástico:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 6e3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1e4 & 0 \\ 0 & 1e4 \end{bmatrix}$$

Dando como ganancias resultantes:

$$K_{lqr} = \begin{bmatrix} -2.1638 & -81.8227 & -13.1739 & -0.000006 \\ -2.1638 & -81.8227 & -13.1739 & +0.000006 \end{bmatrix}$$

$$K0 = \begin{bmatrix} -1.2527 & -3.9137 \\ -1.2527 & +3.9137 \end{bmatrix}$$

Siendo los resultados de la simulación:

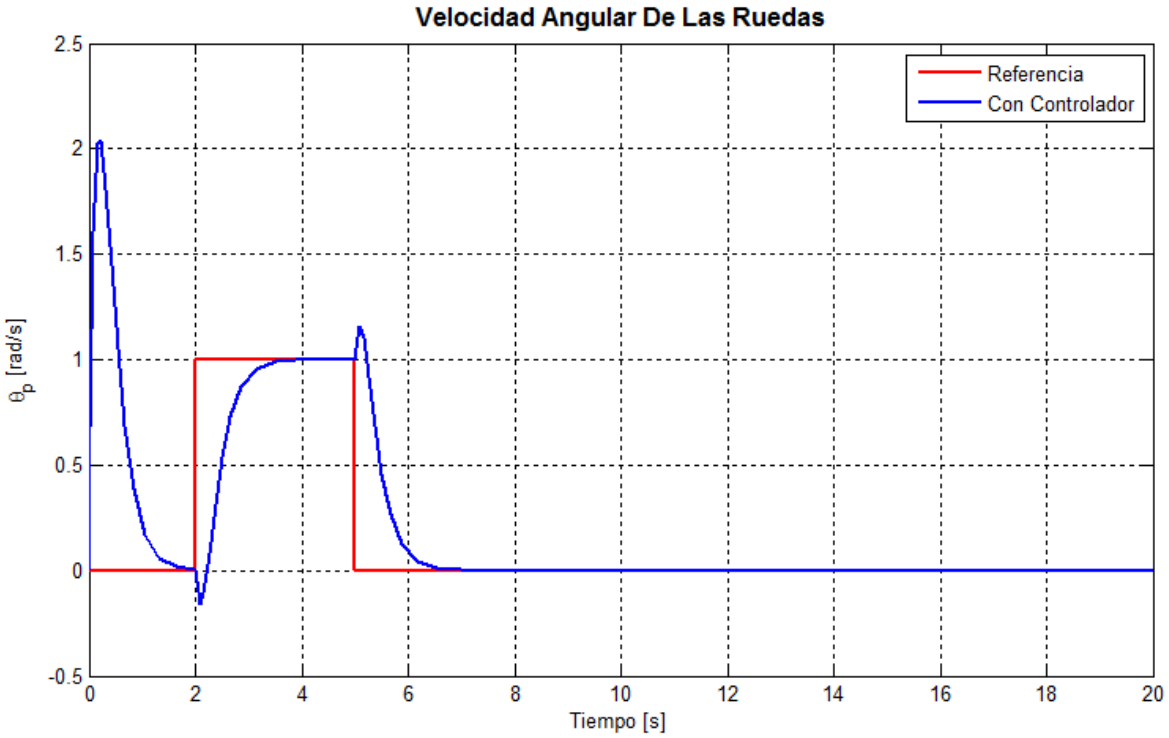


Figura 4.21: Gráfica – LQR 4to Orden (Theta_p) P2

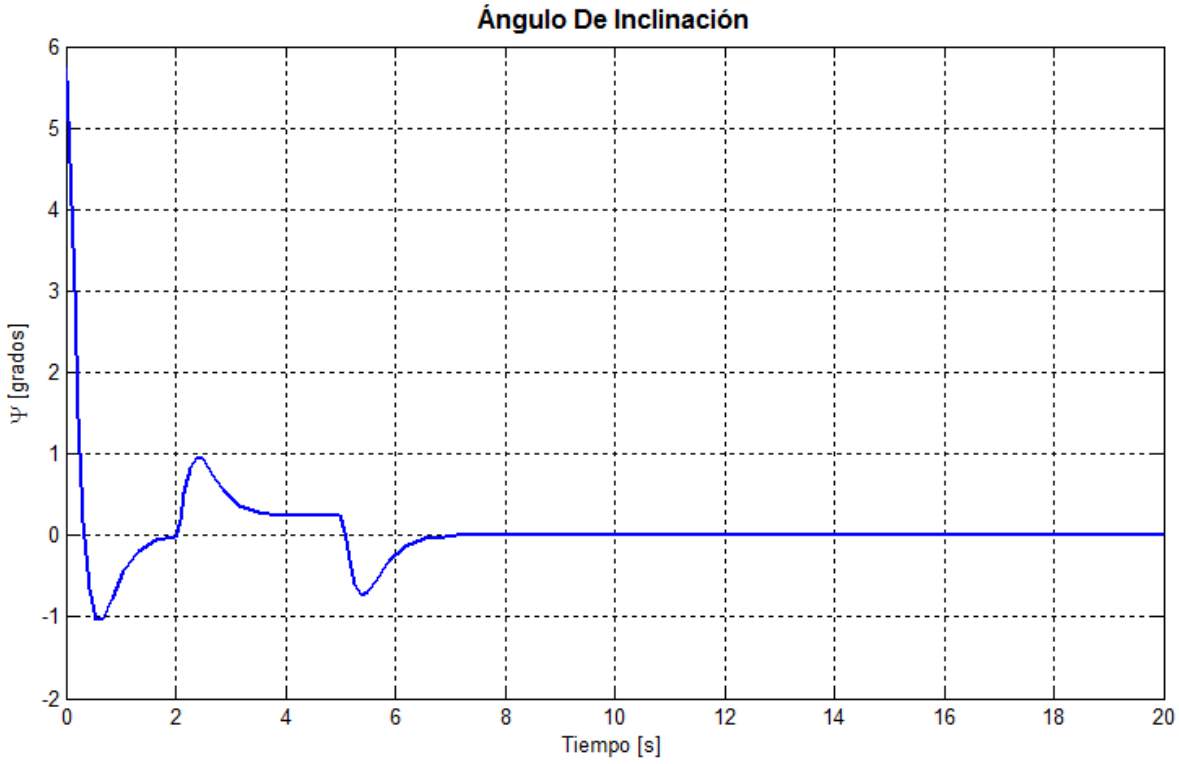


Figura 4.22: Gráfica – LQR 4to Orden (Psi) P2

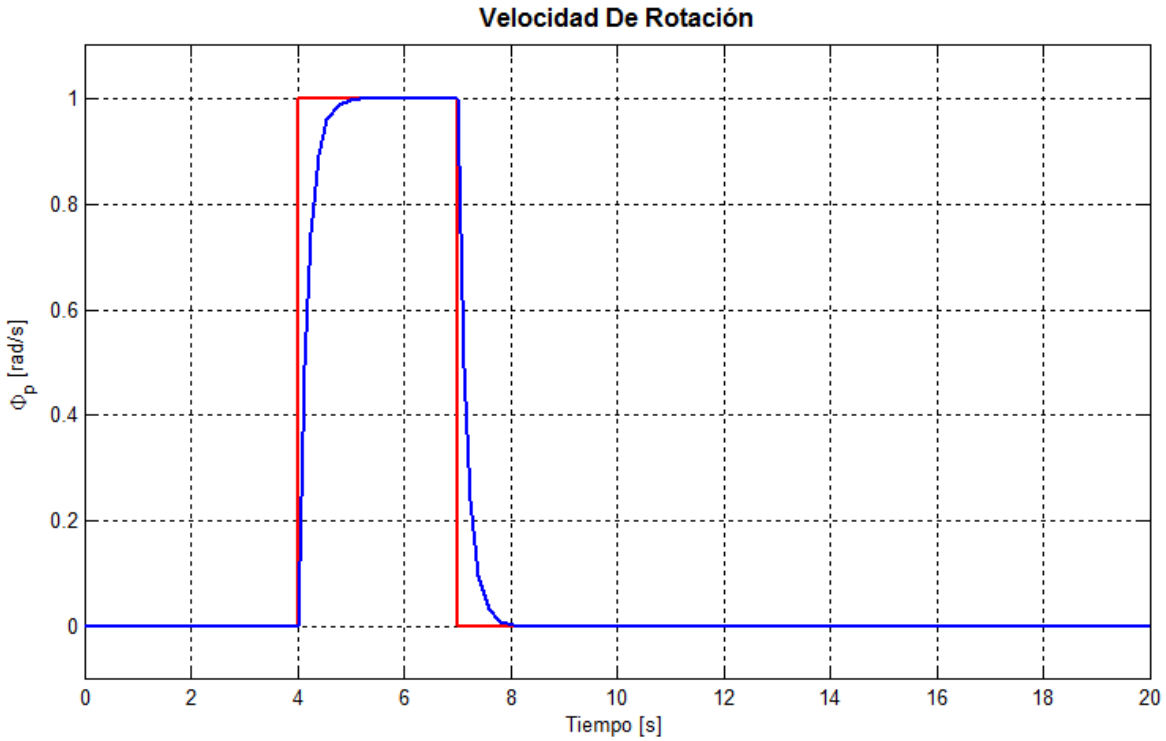


Figura 4.23: Gráfica – LQR 4to Orden (Phi_p) P2

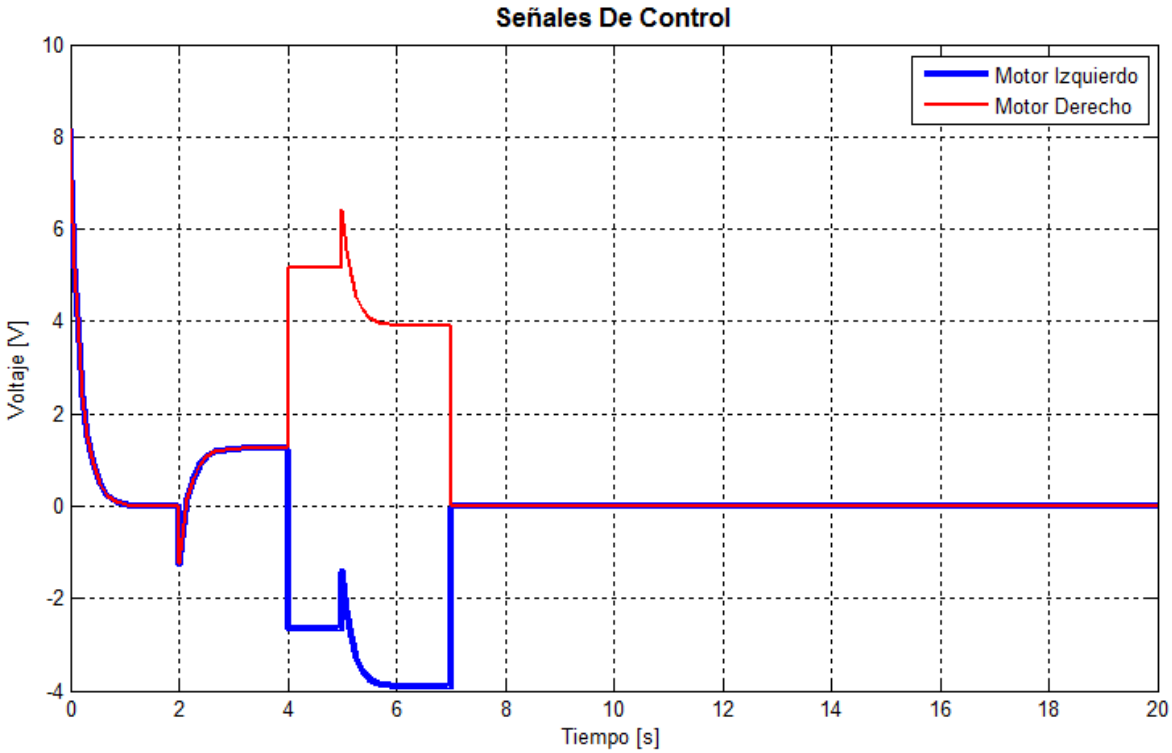


Figura 4.24: Gráfica – LQR 4to Orden (Voltajes) P2

Comparándolas todas:

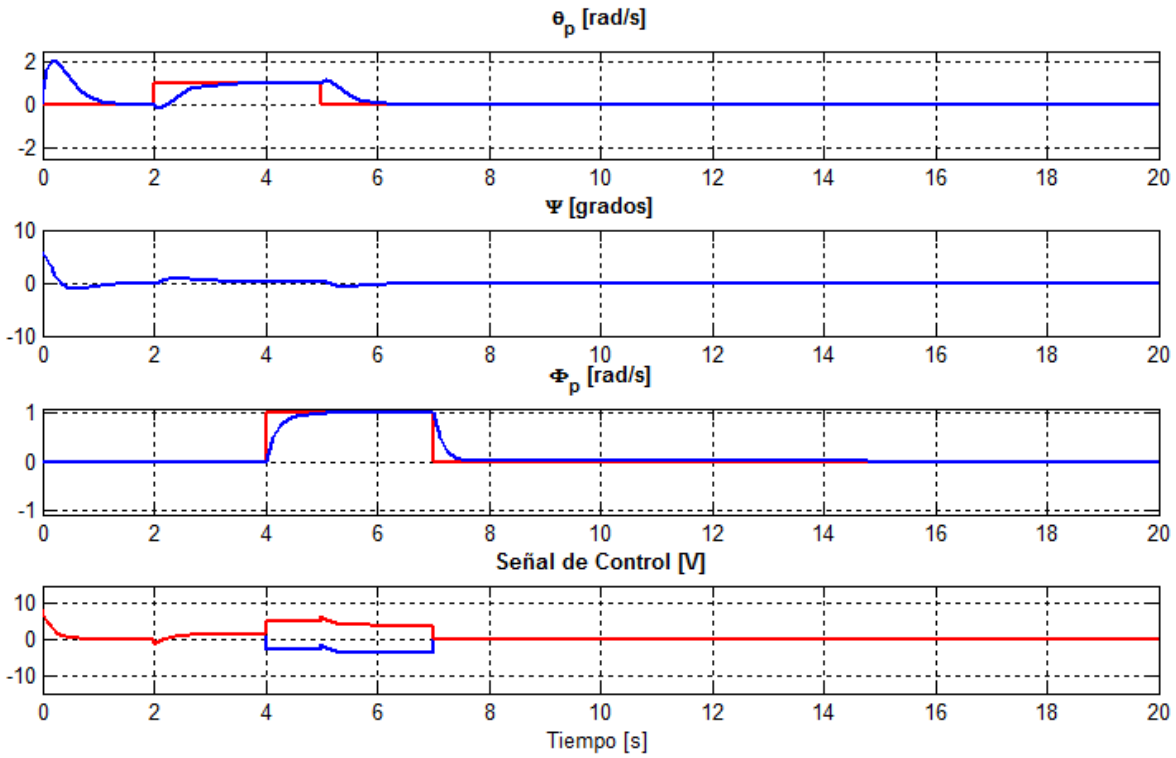


Figura 4.25: Gráficas – LQR 4to Orden (Todas) P2

A primera instancia parece que se han obtenido las mismas gráficas que la prueba anterior, para confirmarlo, se compararán a continuación:

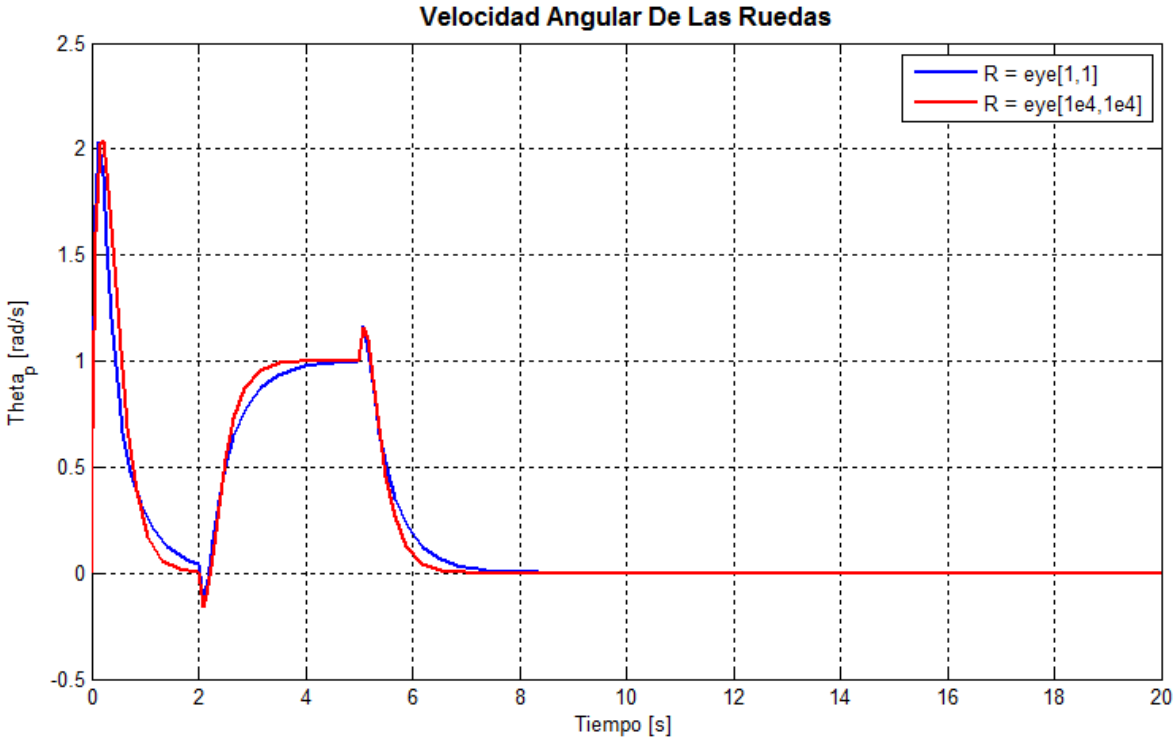


Figura 4.26: Comparación 4to Orden P1 vs P2 (Theta_p)

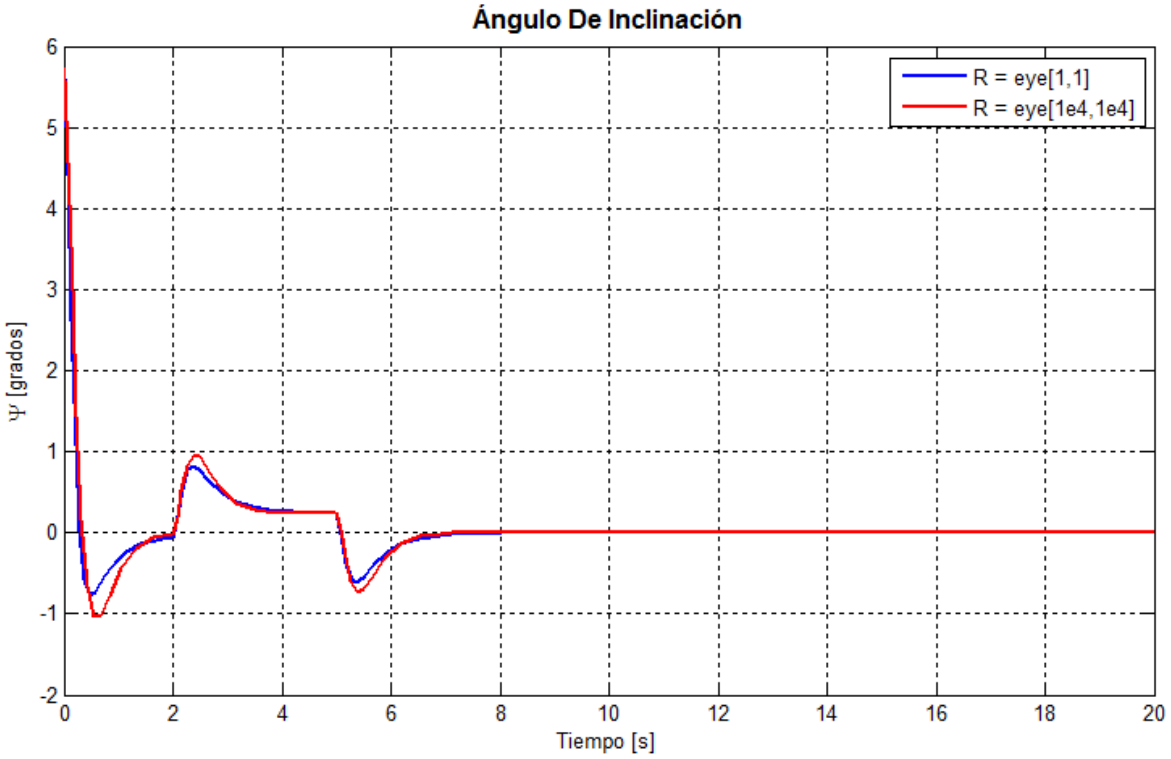


Figura 4.27: Comparación 4to Orden P1 vs P2 (Psi)

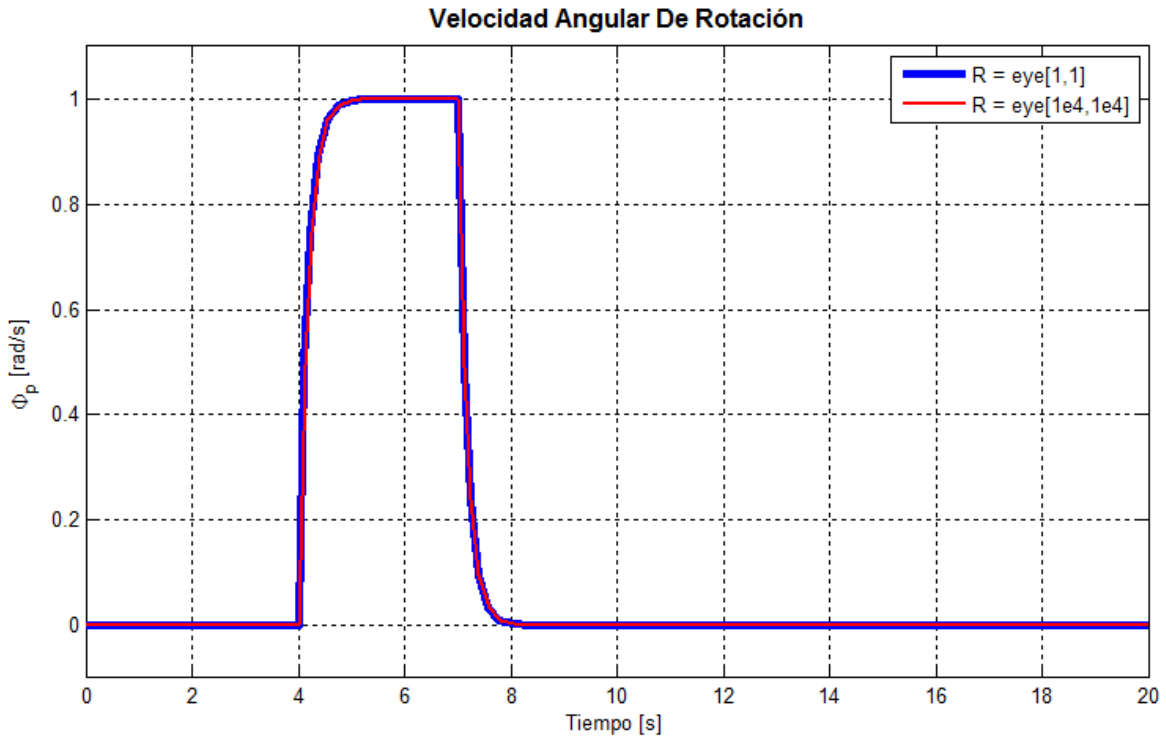


Figura 4.28: Comparación 4to Orden P1 vs P2 (Phi_p)

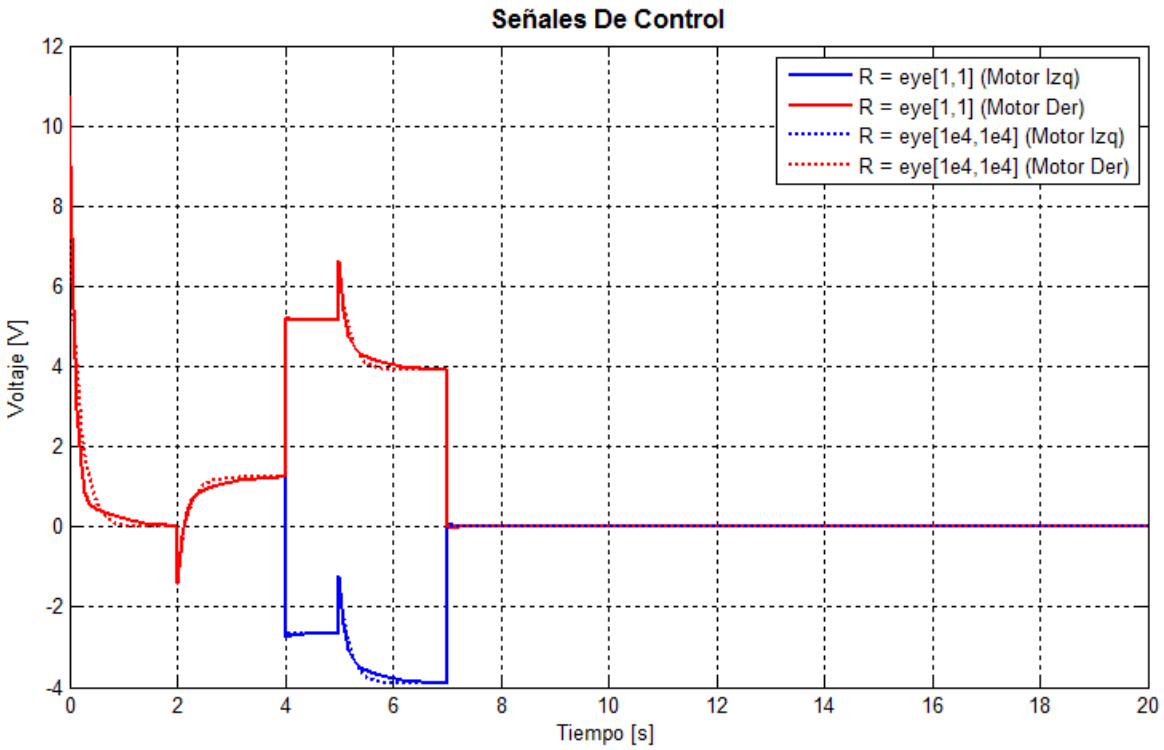


Figura 4.29: Comparación 4to Orden P1 vs P2 (Voltajes)

Las gráficas al ser comparadas se puede decir que son prácticamente iguales, esto indica que sin importar que tan agresivo se quiera el control para la inclinación del péndulo tenemos un comportamiento asintótico.

Al comparar las señales de control entre el sistema de sexto orden y la de cuarto orden podemos observar que debido a que la de sexto orden se satura, sería una mejor opción utilizar el control del sistema de cuarto orden a menos que se logre sintonizar de una manera que no se sature la señal, pues este comportamiento de las señales de control puede inducir efectos no deseados en el sistema al momento de implementar el control.

4.1.5 Control LQR (Respuesta Real)

Una vez implementado el controlador anterior, se realizó una prueba para verificar el comportamiento del controlador, el TWIP debe de ser conectado por medio de un cable USB para enviar los datos a Matlab® para ser utilizados en una gráfica.

Para la prueba se colocó el TWIP en su posición vertical y se encendió, una vez establecida la comunicación con Matlab® se envió la señal deseada de rotación logrando obtener la gráfica que representa la respuesta real del controlador para la velocidad de rotación ($\dot{\emptyset}$) del TWIP:

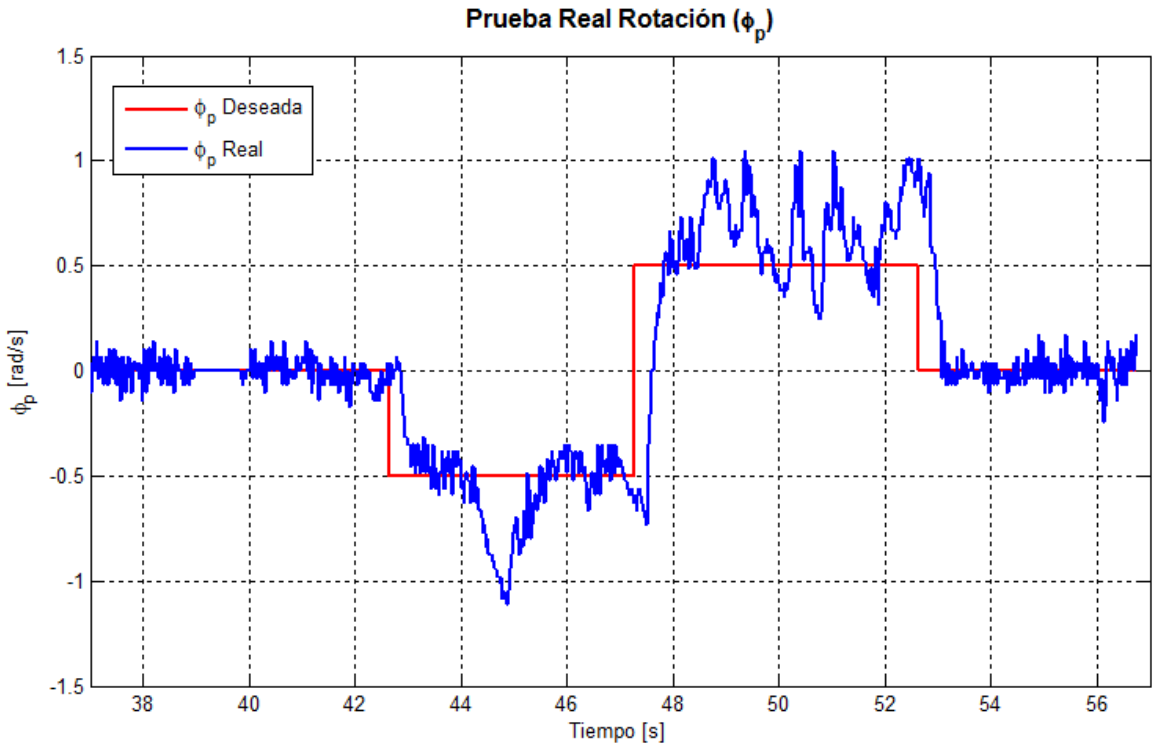


Figura 4.30: Respuesta Real Del Controlador (Rotación)

También se graficó la inclinación del péndulo para demostrar que no se cayó durante la rotación:

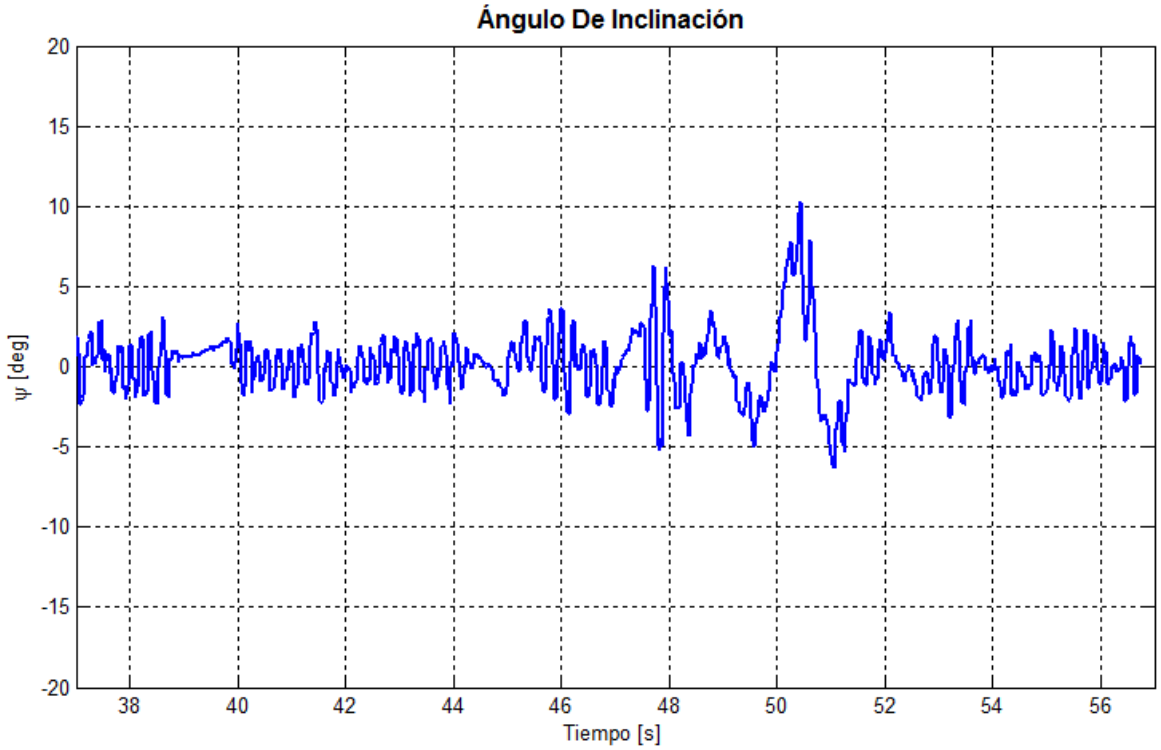


Figura 4.31: Respuesta Real Del Controlador (Inclinación)

Se puede apreciar fácilmente que a pesar de que no es una señal limpia, pues las ruedas están haciendo un doble trabajo (girar y mantener el equilibrio del TWIP), el controlador responde de manera correcta a las señales de referencia, con un desempeño muy cercano al de la simulación.

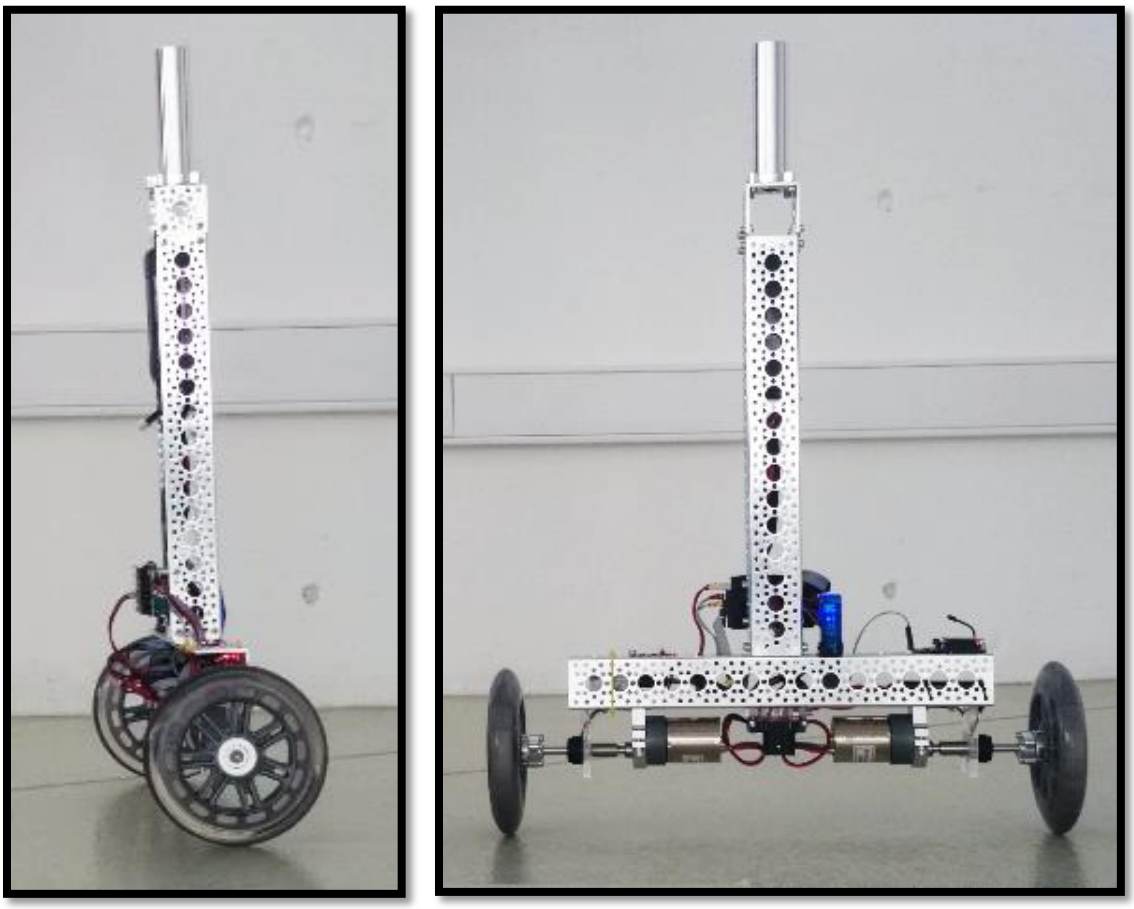


Figura 4.32: TWIP Real Durante Pruebas

5 CONCLUSIONES Y TRABAJO FUTURO

El péndulo invertido es uno de los sistemas más recurridos en clases de control, ya que pueden demostrar varios conceptos importantes sobre los controladores, sin embargo es uno de los sistemas más difíciles de controlar debido a su inherente inestabilidad.

En esta tesis se muestran las características principales para el diseño de un péndulo invertido sobre dos ruedas así como el procedimiento que se tiene que seguir para el diseño e implementación del controlador, desde el modelo matemático, identificación paramétrica, selección del controlador, simulaciones e implementación y se compararon los controladores para el sistema de 6to orden donde se pueden controlar posiciones y velocidades angulares y el sistema de 4to orden donde solamente se pueden controlar velocidades angulares.

Mientras se desarrolló el modelo matemático del péndulo invertido sobre dos ruedas se pudo notar que existen similitudes con los modelos de péndulos invertidos más simples, de hecho las ecuaciones que definen a un péndulo invertido simple se encuentra dentro del modelo del péndulo invertido sobre un carro, si se cambia el carro por una rueda, el modelo no cambia, pero cuando se modela en péndulo invertido sobre dos ruedas realmente tampoco cambia solamente estamos dividiendo el par de una rueda en dos, considerando que solamente se moverá en un eje, como en el caso del carro. Debido a que deseamos la capacidad de que el péndulo invertido se mueva en el plano "XY" entonces solo se agrega la ecuación que define la rotación de todo el sistema.

El uso de partes comerciales para la construcción del TWIP no es necesaria pero redujo en gran medida el tiempo en que se completó el trabajo, además de que se buscaba concentrar todos los esfuerzos en el diseño del controlador; se podrían haber diseñado y construido todas las piezas estructurales por medio de distintos métodos de manufactura.

Al realizar las pruebas principales al modelo lineal del TWIP se demostró que era un sistema inestable pero controlable, tanto para el modelo de 6to orden como para el modelo de 4to orden. La observabilidad sirvió para demostrar que el sistema de cuarto orden era la realización mínima del sistema para el control de las velocidades.

Debido a que se tiene acceso a todos los estados y las condiciones ambientales sobre las que trabajará el TWIP serán controladas, es decir no existirán vientos o irregularidades en el suelo que causen perturbaciones no consideradas en el modelo, se eligió el controlador LQR el cual desempeña mejor en esas condiciones.

Las pruebas y simulaciones demostraron que se tenía un buen modelo sobre el cual trabajar para diseñar el controlador, el controlador 4to orden se eligió principalmente porque requiere de menor carga computacional para el microcontrolador.

A pesar de tener todos los parámetros teóricos y un controlador que funciona perfectamente en simulaciones, al implementarlo y probarlo en la realidad, no necesariamente funcionará como se espera, siempre es necesario realizar ajustes y sintonizaciones de parámetros ya sea en el controlador o en el código del microcontrolador, un ejemplo de esto se presentó en la zona muerta que tienen los motores, en el caso de los motores que se utilizaron en el TWIP de este trabajo, empiezan a girar

hasta los ±3V, debido a la inercia y reducción del mismo, por lo que en el código del microcontrolador se mapeo la señal de control a PWM no desde de cero sino desde el valor correspondiente a un poco menos de 3V.

Físicamente el TWIP presentó un comportamiento adecuado, cumpliendo los objetivos principales moverse en el plano "XY" manteniendo la vertical, teniendo una tolerancia de hasta 15° donde el controlador todavía puede recuperar la vertical; la buena carga de las baterías influyen severamente en la acción del controlador.

Como trabajo futuro se pretende sustituir el controlador LQR por un controlador robusto o por un controlador no lineal para mejorar el desempeño de control e incluir en las simulación ruido y derivaciones numéricas y el filtro de señales, las cuales retrasan la señal y afectan el diseño del controlador, así como agregar sensores infrarrojos y ultrasónicos que le permitan calcular distancias y detectar obstáculos y así moverse libremente y de manera totalmente autónoma.

También se pretende reducir el tamaño de todo el TWIP, y realizar réplicas que se comuniquen entre ellas para integrar algoritmos de enjambre *(Swarm Robotics)*, para lograr movimientos sincronizados entre varios TWIP's.

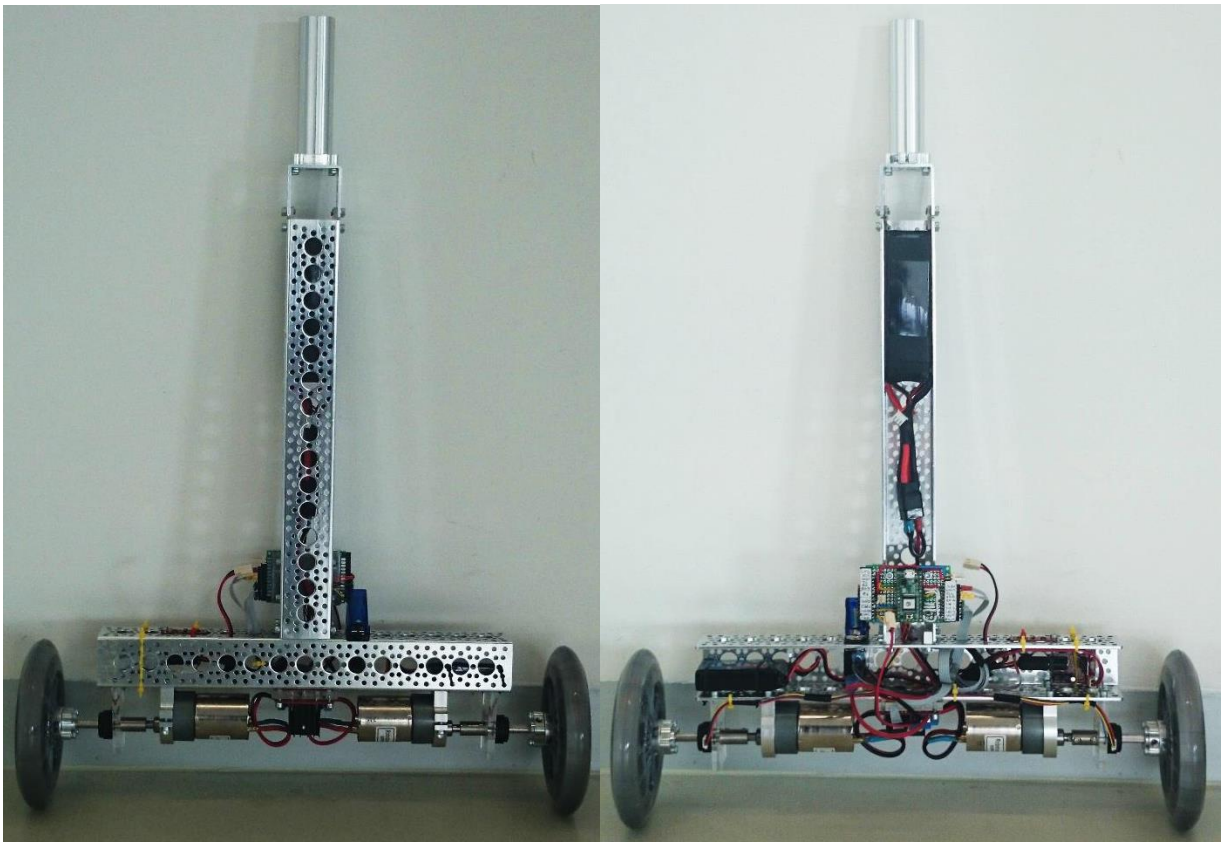


Figura 5.1: Prototipo Físico

APÉNDICE

A.1 PLANOS DE ENSAMBLE

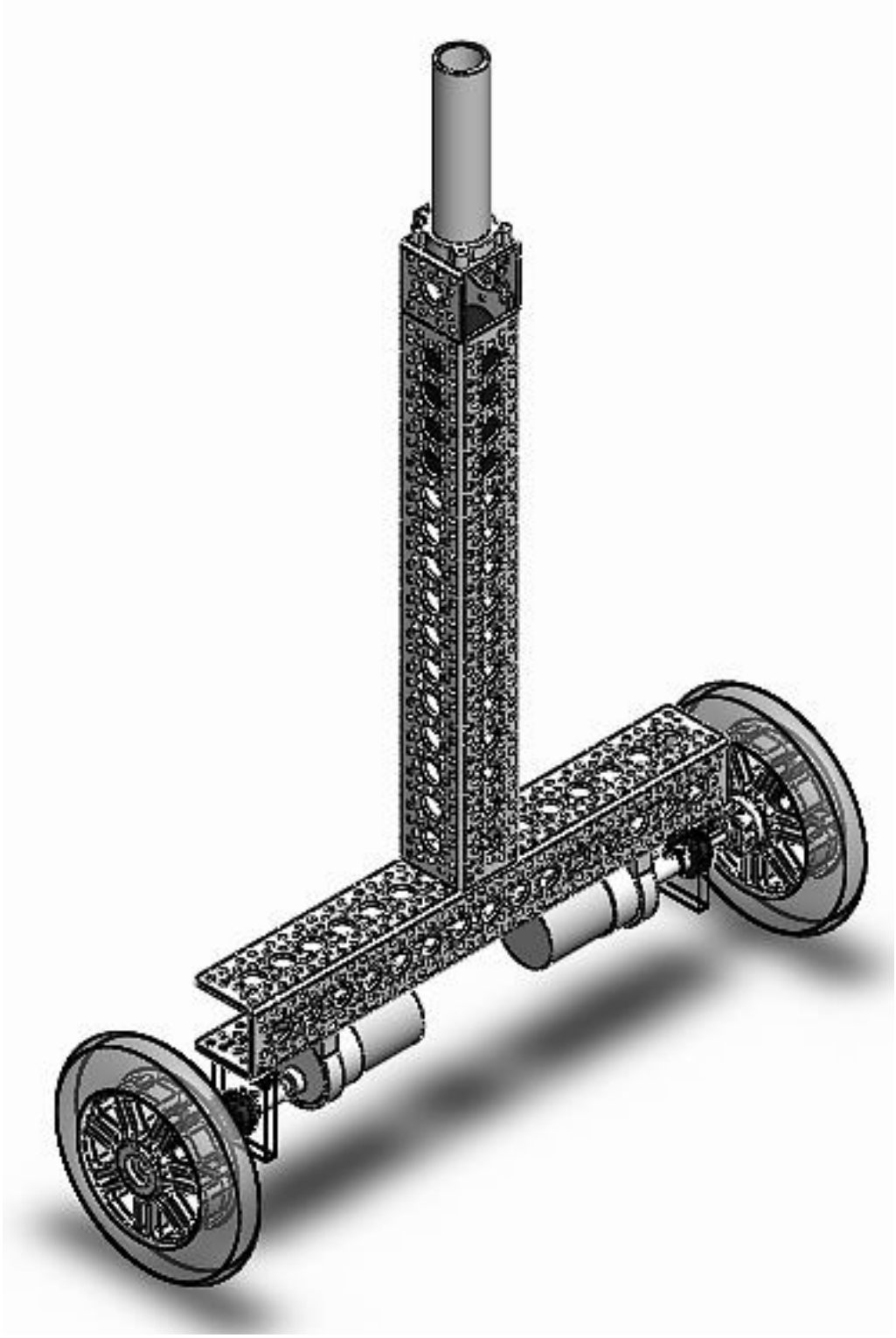


Figura A.0.1: TWIP (Modelo Final)

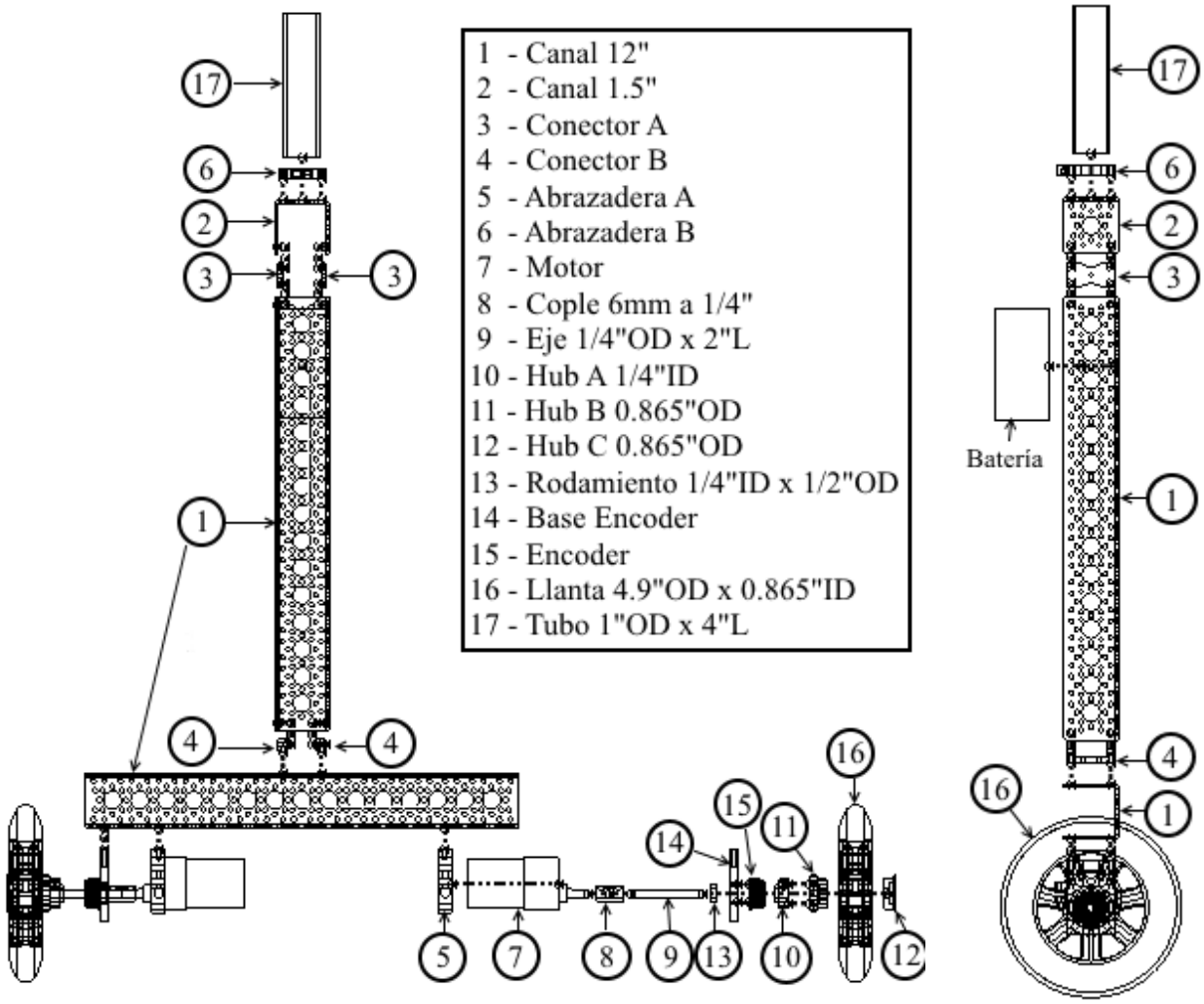


Figura A.0.2: Plano De Ensamble 1

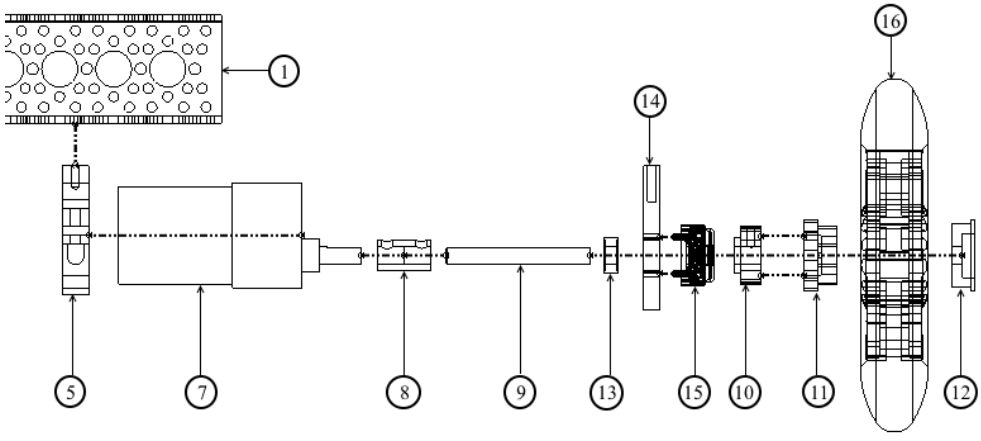


Figura A.0.3: Plano De Ensamble 2

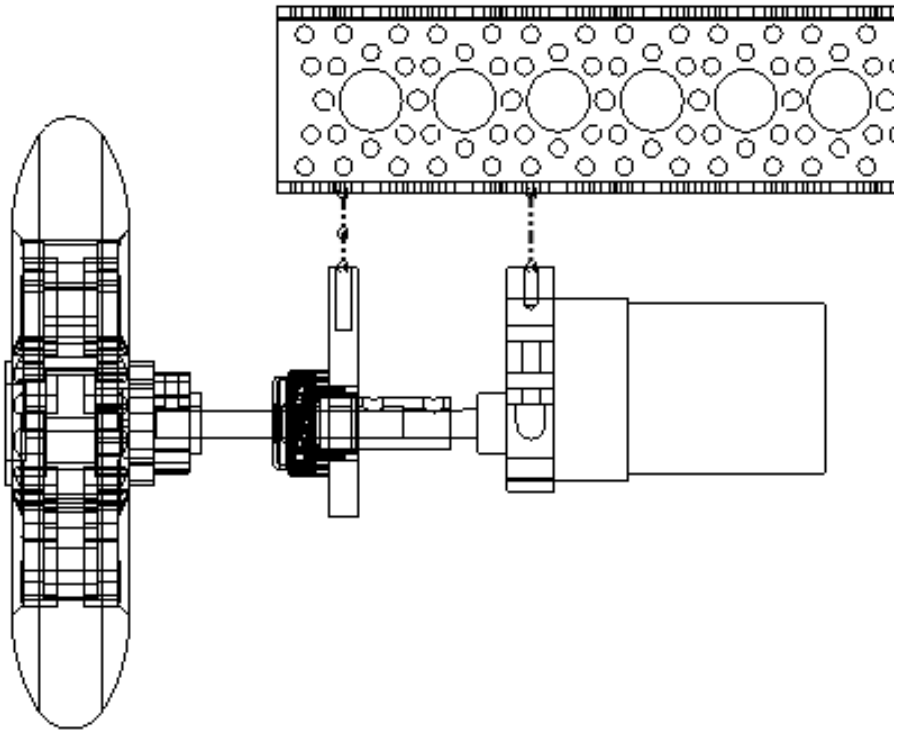


Figura A.0.4: Plano De Ensamble 3

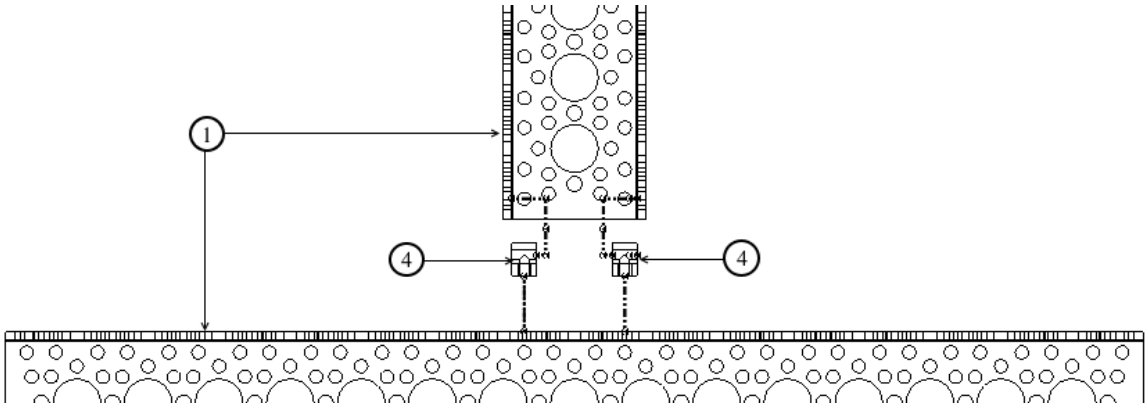


Figura A.0.5: Plano De Ensamble 4

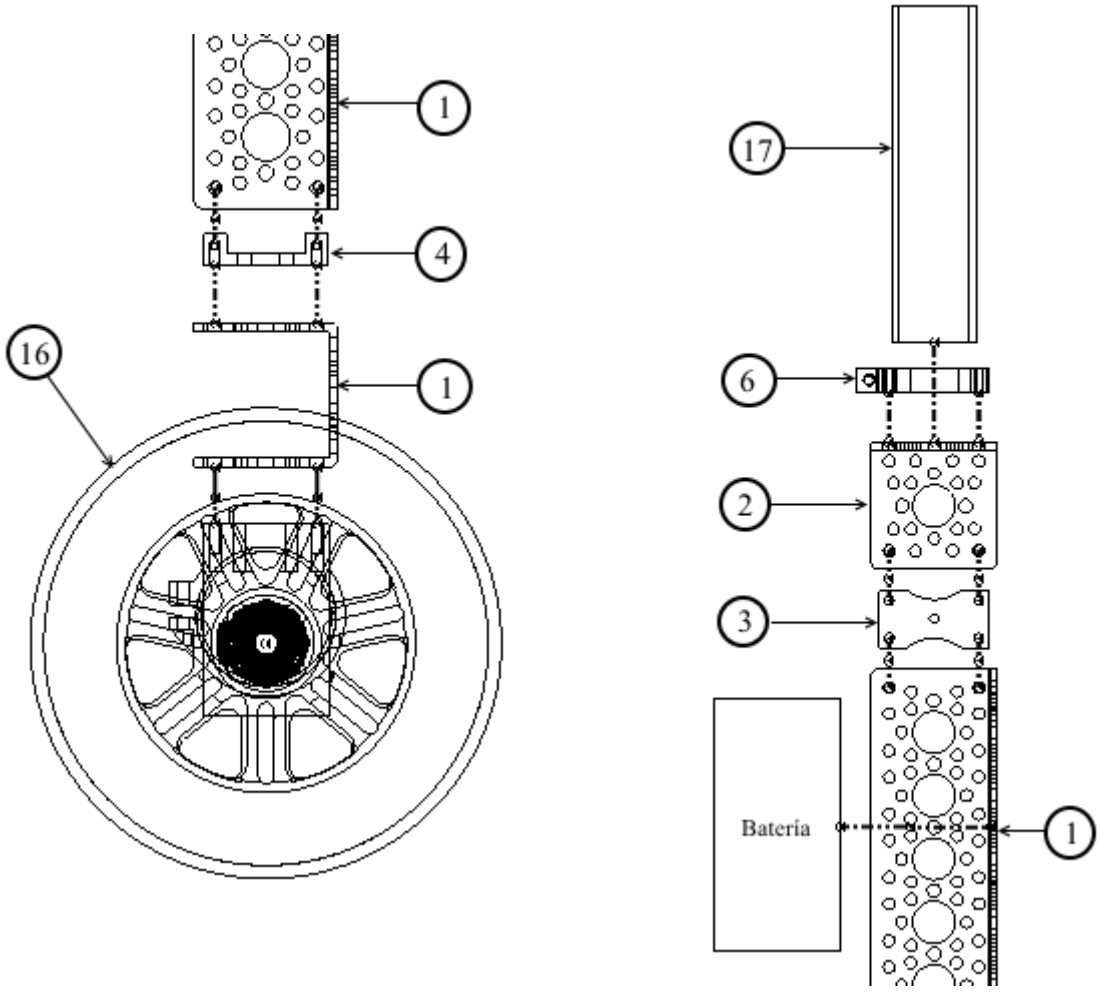


Figura A.0.6: Plano De Ensamble 5

A.2 PROGRAMAS MATLAB®

Los siguientes programas realizan los cálculos necesarios para las simulaciones:

Programa Para Sistema de Sexto Orden

```
%% %%%%%%%%%%%%%%%%%%%% TWO WHEELED INVERTED PENDULUM %%%%%%%%%%%%%%%%%%%% %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SEXTO ORDEN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc

%% %%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETROS FISICOS %%%%%%%%%%%%%%%%%%%%%%%%%% %%
g  = 9.78;            %[m/s^2]- Aceleración de la gravedad
m  = 1.8;             %[kg]   - Masa péndulo
M = 17.8000e-2;       %[kg]   - masa de las ruedas (c/u)
r = 62.2300e-3;       %[m]    - Radio de las ruedas
J = M*(r^2)/2;        %[kgm2] - Momento de inercia de las ruedas
W = 388.8456e-3;      %[m]    - Distancia entre ruedas
L = 0.1075;           %[m]    - distancia del eje de giro al centro de masa
Ipsi = 0.0196;        %[kgm2] - Momento de inercia (Centro de Masa)
Iphi = 0.0179;        %[kgm2] - Momento de inercia (YAW)
bw = 0.00395;         %[-]    - Coeficiente de fricción de las llantas con el
piso
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MOTORES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
R = 12;               %[Ohm]   - Resistencia del motor
ka = 0.9815;          %[Nm/A]  - Constante de par del motor 90RPM
kb = 1.2044;          %[Vs/rad]- Constante de FEM del motor 90RPM
Jm = 0.0159;          %[kgm^2] - Momento de inercia del rotor 90RPM
b2 = 0;               %[-]     - Coeficiente de fricción entre el cuerpo y el
motor
CI = 170*(pi/180);    %[rad]   - Condición Inicial para simulaciones
CI2 = 6*(pi/180);

alpha = ka/R;
beta = ka*kb/R + b2;
beta1 = beta + bw;

%% %%%%%%%%%%%%%%%%%%%%%%%%%% ESPACIO DE ESTADOS %%%%%%%%%%%%%%%%%%%%%%%%%% %%
E = [
     (2*M+m)*(r^2)+2*J+2*Jm m*r*L-2*Jm;
    m*r*L-2*Jm m*L^2+Ipsi+2*Jm];
I = (M*W^2)/2 + Iphi +(J+Jm)*(W^2)/(2*r^2);
J2 = (W^2)*beta1/(2*r^2);
K = W*alpha/(2*r);

A=[
    0 1 0 0 0 0;
    0 -2*(beta1*E(2,2)+beta*E(1,2))/det(E) -m*g*L*E(1,2)/det(E)
2*beta*(E(2,2)+E(1,2))/det(E) 0 0;
    0 0 0 1 0 0;
    0 2*(beta1*E(1,2)+beta*E(1,1))/det(E) m*g*L*E(1,1)/det(E) -
2*beta*(E(1,2)+E(1,1))/det(E) 0 0;
    0 0 0 0 0 1;
    0 0 0 0 0 -J2/I];
B=[
```

```
    0 0;
    alpha*(E(2,2)+E(1,2))/det(E) alpha*(E(2,2)+E(1,2))/det(E);
    0 0;
    -alpha*(E(1,1)+E(1,2))/det(E) -alpha*(E(1,1)+E(1,2))/det(E);
    0 0;
    -K/I K/I];
C=[0 1 0 0 0 0;
   0 0 0 1 0 0;
   0 0 0 0 0 1];
D=zeros(3,2);

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%Verificando Estabilidad
Stability = eig(A);              %INESTABLE!
EcCaracterisitica=poly(A);

%Verificando Controlabilidad
MC = ctrb(A,B);
rankMC = rank(MC);               %CONTROLABLE!

%Verificando Observabilidad
MO = obsv(A,C);
rankMO = rank(MO);               %NO OBSERVABLE!

%CONTROLADOR LQR
Q = [1e3 0 0 0 0 0;
     0 1 0 0 0 0;
     0 0 6e3 0 0 0;
     0 0 0 1 0 0;
     0 0 0 0 1e3 0;
     0 0 0 0 0 1];

R = [1 0; 0 1];

[K,P,E1] = lqr(A,B,Q,R);
K

C1=[1 0 0 0 0 0;  %Matriz C de las variables que
    0 0 0 0 1 0]; %se quieren regular

K0 = inv(C1*inv(-A+B*K)*B)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%FUNCIONES DE TRANSFERENCIA
[num1,den1]=ss2tf(A,B,C,D,1);
[num2,den2]=ss2tf(A,B,C,D,2);
H1 = tf(num1(1,:),den1,'inputn','VL or VR','outputn','Theta_p');
H2 = tf(num1(2,:),den1,'inputn','VL or VR','outputn','Psi_p');
H3 = tf(num1(3,:),den1,'inputn',{'VL'},'outputn',{'Phi_p'});
H4 = tf(num2(3,:),den2,'inputn',{'VR'},'outputn',{'Phi_p'});
```

Programa Para Sistema De Cuarto Orden

```
%% %%%%%%%%%%%%%%%%%%%% TWO WHEELED INVERTED PENDULUM %%%%%%%%%%%%%%%%%%%% %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CUARTO ORDEN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
%% %%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETROS FISICOS %%%%%%%%%%%%%%%%%%%%%%%%%% %%
g  = 9.78;           %[m/s^2]- Aceleración de la gravedad
m  = 1.8;            %[kg]   - Masa péndulo
M = 17.8000e-2;      %[kg]   - masa de las ruedas (c/u)
r = 62.2300e-3;      %[m]    - Radio de las ruedas
J = M*(r^2)/2;       %[kgm2] - Momento de inercia de las ruedas
W = 388.8456e-3;     %[m]    - Distancia entre ruedas
L = 0.1075;          %[m]    - distancia del eje de giro al centro de masa
Ipsi = 0.0196;       %[kgm2] - Momento de inercia (Centro de Masa)
Iphi = 0.0179;       %[kgm2] - Momento de inercia (YAW)
bw = 0.00395;        %[-]    - Coeficiente de fricción de las llantas con el
piso
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MOTORES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
R = 12;              %[Ohm]  - Resistencia del motor
ka = 0.9815;         %[Nm/A] - Constante de par del motor 90RPM
kb = 1.2044;         %[Vs/rad]- Constante de FEM del motor 90RPM
Jm = 0.0159;         %[kgm^2]- Momento de inercia del rotor 90RPM
b2 = 0;              %[-]    - Coeficiente de fricción entre el cuerpo y el
motor
CI = 170*(pi/180);   %[rad]  - Condición Inicial para simulaciones
CI2 = 6*(pi/180);
alpha = ka/R;
beta = ka*kb/R + b2;
beta1 = beta + bw;
%% %%%%%%%%%%%%%%%%%%%%%%%%%% ESPACIO DE ESTADOS %%%%%%%%%%%%%%%%%%%%%%%%%% %%
E = [
     (2*M+m)*(r^2)+2*J+2*Jm m*r*L-2*Jm;
      m*r*L-2*Jm m*L^2+Ipsi+2*Jm];
I = (M*W^2)/2 + Iphi +(J+Jm)*(W^2)/(2*r^2);
J2 = (W^2)*beta1/(2*r^2);
K = W*alpha/(2*r);

A=[
    -2*(beta1*E(2,2)+beta*E(1,2))/det(E) -m*g*L*E(1,2)/det(E)
2*beta*(E(2,2)+E(1,2))/det(E) 0;
    0 0 1 0;
    2*(beta1*E(1,2)+beta*E(1,1))/det(E) m*g*L*E(1,1)/det(E) -
2*beta*(E(1,2)+E(1,1))/det(E) 0;
    0 0 0 -J2/I];
B=[
    alpha*(E(2,2)+E(1,2))/det(E) alpha*(E(2,2)+E(1,2))/det(E);
    0 0;
    -alpha*(E(1,1)+E(1,2))/det(E) -alpha*(E(1,1)+E(1,2))/det(E);
    -K/I K/I];
C=[1 0 0 0;
   0 0 1 0;
   0 0 0 1];
D=zeros(3,2);
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%Verificando Estabilidad
```

```
Stability = eig(A);              %INESTABLE!
EcCaracterisitica=poly(A);
%Verificando Controlabilidad
MC = ctrb(A,B);
rankMC = rank(MC);               %CONTROLABLE!
%Verificando Observabilidad
MO = obsv(A,C);
rankMO = rank(MO);               %OBSERVABLE!
%CONTROLADOR LQR
Q = [1 0 0 0;
     0 6e3 0 0;
     0 0 1 0;
     0 0 0 1];
R = [1e4 0; 0 1e4];
[K,P,E1] = lqr(A,B,Q,R);
format long
K
C1=[1 0 0 0;  %Matriz C de las variables que
    0 0 0 1]; %se quieren regular

K0 = inv(C1*inv(-A+B*K)*B) %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
%FUNCIONES DE TRANSFERENCIA
[num1,den1]=ss2tf(A,B,C,D,1);
[num2,den2]=ss2tf(A,B,C,D,2);
H1 = tf(num1(1,:),den1,'inputn','VL or VR','outputn','Theta_p');
H2 = tf(num1(2,:),den1,'inputn','VL or VR','outputn','Psi_p');
H3 = tf(num1(3,:),den1,'inputn',{'VL'},'outputn',{'Phi_p'});
H4 = tf(num2(3,:),den2,'inputn',{'VR'},'outputn',{'Phi_p'});
```

Programa Para Dibujar Gráficas De Las Señales De Los Diagramas De Bloques

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%% Graficar %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%
figure(1)
plot(theta_s1.time,theta_s1.signals.values(:,1),'r')
hold on
plot(theta_s1.time,theta_s1.signals.values(:,2))

figure(2)
plot(psi_s1.time,psi_s1.signals.values(:,1))
hold on

figure(3)
plot(phi_s1.time,phi_s1.signals.values(:,1),'r')
hold on
plot(phi_s1.time,phi_s1.signals.values(:,2))

figure(4)
plot(volt_s1.time,volt_s1.signals.values(:,1))
hold on
plot(volt_s1.time,volt_s1.signals.values(:,2),'r')
```

A.3 DIAGRAMA ELÉCTRICO

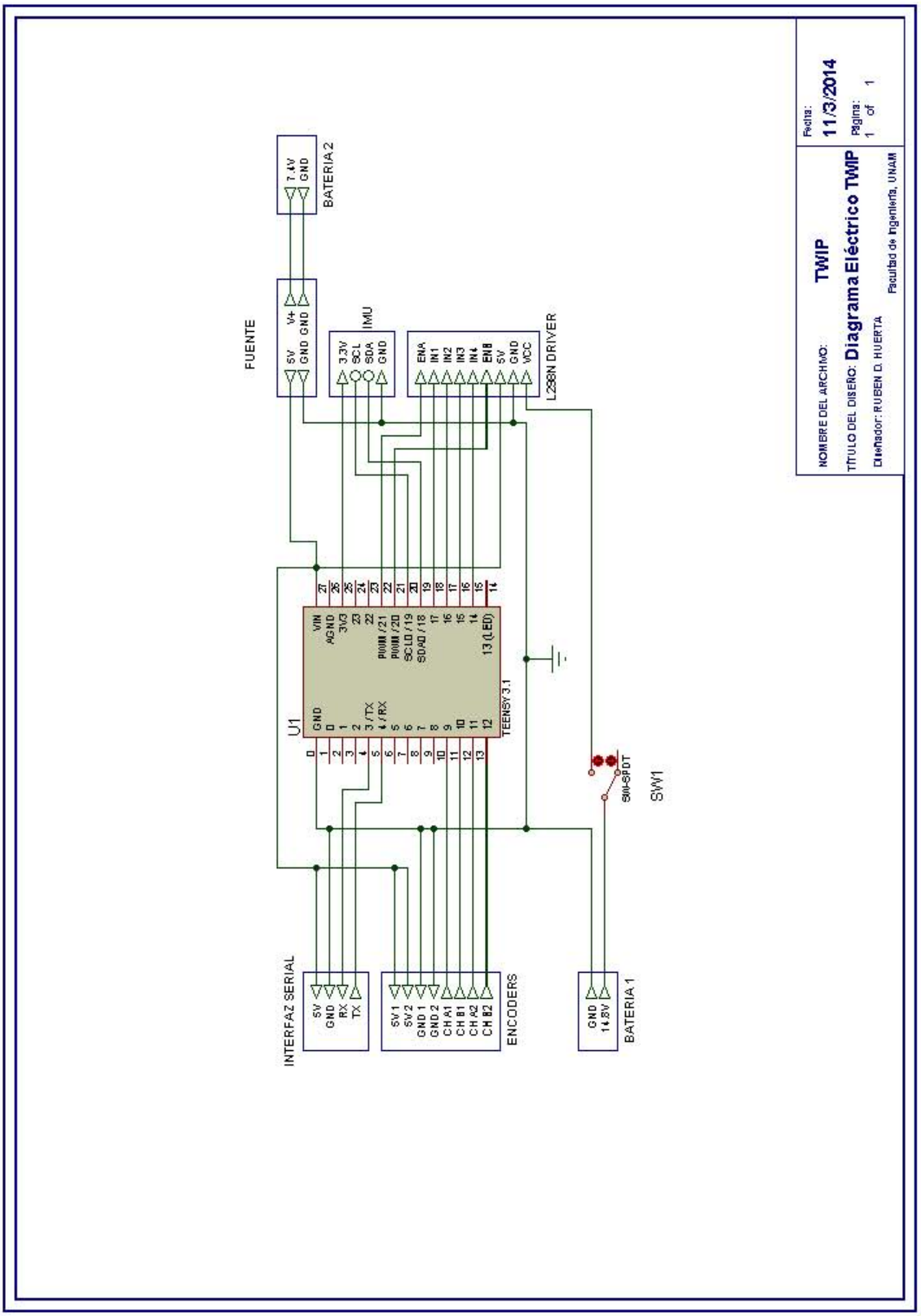


Figura A.0.7: Diagrama Eléctrico

A.4 PROGRAMAS DEL MICROCONTROLADOR (TEENSY 3.1)

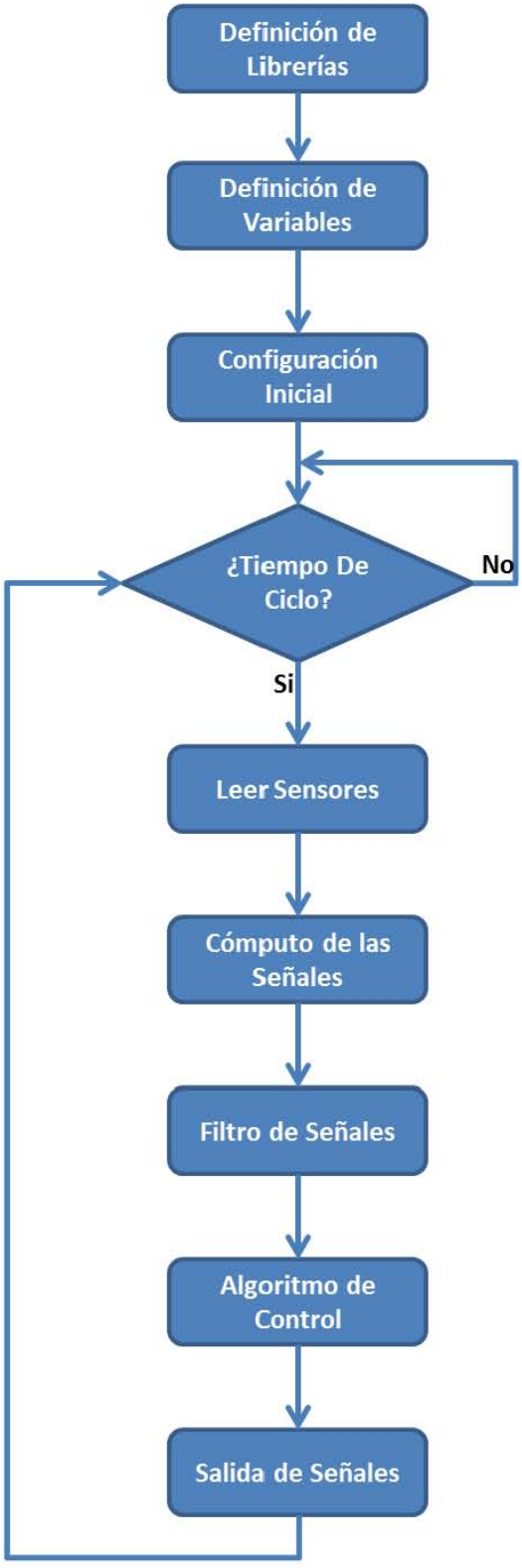


Figura A.0.8: Diagrama De Flujo General (Teensy 3.1)

Programa Principal

```
#include <Wire.h>
#include <Encoder.h>

// SENSOR CALIBRATION
/*****************************************************************/
// Accelerometer
// "accel x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX  Z_MIN/Z_MAX"
#define ACCEL_X_MIN ((float) -250)
#define ACCEL_X_MAX ((float) 250)
#define ACCEL_Y_MIN ((float) -250)
#define ACCEL_Y_MAX ((float) 250)
#define ACCEL_Z_MIN ((float) -250)
#define ACCEL_Z_MAX ((float) 250)

// Magnetometer (standard calibration mode)
// "magn x,y,z (min/max) = X_MIN/X_MAX  Y_MIN/Y_MAX  Z_MIN/Z_MAX"
#define MAGN_X_MIN ((float) -600)
#define MAGN_X_MAX ((float) 600)
#define MAGN_Y_MIN ((float) -600)
#define MAGN_Y_MAX ((float) 600)
#define MAGN_Z_MIN ((float) -600)
#define MAGN_Z_MAX ((float) 600)

// Gyroscope
// "gyro x,y,z (current/average) = .../OFFSET_X  .../OFFSET_Y  .../OFFSET_Z
#define GYRO_AVERAGE_OFFSET_X ((float) 0.0)
#define GYRO_AVERAGE_OFFSET_Y ((float) 0.0)
#define GYRO_AVERAGE_OFFSET_Z ((float) 0.0)

// Sensor calibration scale and offset values
#define ACCEL_X_OFFSET ((ACCEL_X_MIN + ACCEL_X_MAX) / 2.0f)
#define ACCEL_Y_OFFSET ((ACCEL_Y_MIN + ACCEL_Y_MAX) / 2.0f)
#define ACCEL_Z_OFFSET ((ACCEL_Z_MIN + ACCEL_Z_MAX) / 2.0f)
#define ACCEL_X_SCALE (GRAVITY / (ACCEL_X_MAX - ACCEL_X_OFFSET))
#define ACCEL_Y_SCALE (GRAVITY / (ACCEL_Y_MAX - ACCEL_Y_OFFSET))
#define ACCEL_Z_SCALE (GRAVITY / (ACCEL_Z_MAX - ACCEL_Z_OFFSET))

#define MAGN_X_OFFSET ((MAGN_X_MIN + MAGN_X_MAX) / 2.0f)
#define MAGN_Y_OFFSET ((MAGN_Y_MIN + MAGN_Y_MAX) / 2.0f)
#define MAGN_Z_OFFSET ((MAGN_Z_MIN + MAGN_Z_MAX) / 2.0f)
#define MAGN_X_SCALE (100.0f / (MAGN_X_MAX - MAGN_X_OFFSET))
#define MAGN_Y_SCALE (100.0f / (MAGN_Y_MAX - MAGN_Y_OFFSET))
```

```
#define MAGN_Z_SCALE (100.0f / (MAGN_Z_MAX - MAGN_Z_OFFSET))

// Gain for gyroscope (ITG-3200)
#define GYRO_GAIN 0.06957 // Same gain on all axes
#define GYRO_SCALED_RAD(x) (x * TO_RAD(GYRO_GAIN)) // Calculate the scaled gyro readings in radians per second

//Stuff
#define GRAVITY 256.0f // "1G reference" used for DCM filter and accelerometer calibration
#define TO_RAD(x) (x * 0.01745329252)  // *pi/180
#define TO_DEG(x) (x * 57.2957795131)  // *180/pi
#define RoundToQuarters(x) (round(x / 0.25) * 0.25) // Round to closest quarter

// DCM parameters
#define Kp_ROLLPITCH 0.02f
#define Ki_ROLLPITCH 0.00002f
#define Kp_YAW 1.2f
#define Ki_YAW 0.00002f

//Sensor Variables
float accel[3];
float accel_min[3];
float accel_max[3];
float magnetom[3];
float magnetom_min[3];
float magnetom_max[3];
float gyro[3];
float gyro_average[3];
int gyro_num_samples = 0;

// Euler Angles
float yaw;
float pitch;
float roll;
float roll_ant = 0.00;
float f_roll;
float d_roll;
float fd_roll;
float theta;
float theta_ant = 0.00;
float d_theta;
float phi;
float phi_ant = 0.00;
float d_phi;
```

```
float desfase = -2.85; //Desfase del Roll en grados

// DCM Variables
float MAG_Heading;
float Accel_Vector[3]= {0, 0, 0}; // Store the acceleration in a vector
float Gyro_Vector[3]= {0, 0, 0}; // Store the gyros turn rate in a vector
float Omega_Vector[3]= {0, 0, 0}; // Corrected Gyro_Vector data
float Omega_P[3]= {0, 0, 0}; // Omega Proportional correction
float Omega_I[3]= {0, 0, 0}; // Omega Integrator
float Omega[3]= {0, 0, 0};
float errorRollPitch[3] = {0, 0, 0};
float errorYaw[3] = {0, 0, 0};
float DCM_Matrix[3][3] = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
float Update_Matrix[3][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
float Temporary_Matrix[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};

// DCM timing in the main loop
unsigned long timestamp;
unsigned long timestamp_old;
float G_Dt; // Integration time for DCM algorithm

// For Encoders
Encoder EncL(9, 10);  // Left Encoder Pins for Channel A and Channel B
Encoder EncR(11,12);  // Right Encoder Pins for Channel A and Channel B
long oldL  = -999;    // Previous Tick count for Left Encoder
long oldR  = -999;    // Previous Tick count for Right Encoder
long newL, newR;      // Update Tick count for Encoders
float r = 0.06223;    // [m] - Wheels radio
float W = 0.388871;   // [m] - Distance between wheels

//For Control
// Q = eye(1,6e3,1,1); R = eye(1e4,1e4);
float u[2] = {0,0}; //Control Signal
float K0[2][2] = {{-1.2527, -3.9137},{-1.2527, 3.9137}}; //K0 gain
float R[2] = {0.0,0.0}; //Reference inputs
float Klqr[2][6] = {{-2.1638,-81.8227,-13.1739,-6.3877e-6},
                    {-2.1638,-81.8227,-13.1739, 6.3877e-6}};
                    //K gain from LQR control
float x[4]; //States
float kx[2]; // Klqr * x auxiliar variable

//For Motors
int ENA = 21;
int ENB = 20;
```

```
int IN1 = 17; //Forward
int IN2 = 16; //Reverse
int IN3 = 15; //Forward
int IN4 = 14; //Reverse
float PWM_L;
float PWM_R;

//For matlab
byte matlab_in[10]; //Save reference input from Matlab in an array

void setup()
{
  //digitalWrite(ENA,LOW);
  //digitalWrite(ENB,LOW);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  Serial.begin(115200);
  delay(50);
  Wire.begin();
  Accel_Init();
  Magn_Init();
  Gyro_Init();
  delay(20);
  reset_sensor_fusion();
}

void loop()
{
  // Time to read the sensors again?
  if((millis() - timestamp) >= 20) //milliseconds
  {
    timestamp_old = timestamp;
    timestamp = millis();
    if (timestamp > timestamp_old)
    {
      G_Dt = (float) (timestamp - timestamp_old) / 1000.0f; // Real time of loop run. We use this on the
DCM algorithm (gyro integration time)
    }
    else G_Dt = 0;
    read_sensors();                    // UPDATE SENSOR READINGS
```

```
   //printData();
   newL = EncL.read();                  // UPDATE LEFT ENCODER
   newR = EncR.read();                  // UPDATE RIGHT ENCODER
   theta = (newL + newR) / 8.0;         // GET Theta [degress]
   phi = (r * (newR - newL)) / (4.0 * W); // GET Phi [degrees]
   compensate_sensor_errors();          // APPLY SENSOR CALIBRATION
   // RUN DCM ALGORITHM //
   Compass_Heading();
   Matrix_update();
   Normalize();
   Drift_correction();
   Euler_angles();
   ///////////////////////
   d_roll = (roll - roll_ant) / G_Dt;    // CALCULATE TIME DERIVATIVE OF Psi
   //f_roll = low_filter2(roll);         // FILTER - ROLL
   roll_ant = roll;
   fd_roll = low_filter2(d_roll);       // FILTER - ROLL VELOCITY
   theta = TO_RAD(theta);               // CONVERT DEGREES TO RADIANS
   phi = TO_RAD(phi);                   // CONVERT DEGREES TO RADIANS
   d_theta = (theta - theta_ant) / G_Dt;  // CALCULATE TIME DERIVATIVE OF Theta
   d_phi = (phi - phi_ant) / G_Dt;      // CALCULATE TIME DERIVATIVE OF Phi
   theta_ant = theta;
   phi_ant = phi;
   input_matlab();                      // RECEIVE REFERENCE INPUTS FROM MATLAB
   control();                           // RUN CONTROL ALGORITHM
   if(u[0] >  15.00) u[0] = 15.00;
   if(u[0] < -15.00) u[0] = -15.00;
   if(u[1] >  15.00) u[1] = 15.00;
   if(u[1] < -15.00) u[1] = -15.00;
   PWM_motores(u[0],u[1]);              // WRITE CONTROL SIGNALS TO MOTORS BY PWM
   //output_angles();
   output_matlab();                     // SEND DATA TO MATLAB FOR DEBUG
  }
}

void read_sensors()
{
  Read_Gyro();   // Read gyroscope
  Read_Accel();  // Read accelerometer
  Read_Magn();   // Read magnetometer
}

void reset_sensor_fusion()
{
```

```
  float temp1[3];
  float temp2[3];
  float xAxis[] = {1.0f, 0.0f, 0.0f};
  read_sensors();
  timestamp = millis();
  // GET PITCH
  // Using y-z-plane-component/x-component of gravity vector
  pitch = -atan2(accel[0], sqrt(accel[1] * accel[1] + accel[2] * accel[2]));
  // GET ROLL
  // Compensate pitch of gravity vector
  Vector_Cross_Product(temp1, accel, xAxis);
  Vector_Cross_Product(temp2, xAxis, temp1);
  // Normally using x-z-plane-component/y-component of compensated gravity vector
  // roll = atan2(temp2[1], sqrt(temp2[0] * temp2[0] + temp2[2] * temp2[2]));
  // Since we compensated for pitch, x-z-plane-component equals z-component:
  roll = atan2(temp2[1], temp2[2]);
  // GET YAW
  Compass_Heading();
  yaw = MAG_Heading;
  // Init rotation matrix
  init_rotation_matrix(DCM_Matrix, yaw, pitch, roll);
}

void compensate_sensor_errors()
{
    // Compensate accelerometer error
    accel[0] = (accel[0] - ACCEL_X_OFFSET) * ACCEL_X_SCALE;
    accel[1] = (accel[1] - ACCEL_Y_OFFSET) * ACCEL_Y_SCALE;
    accel[2] = (accel[2] - ACCEL_Z_OFFSET) * ACCEL_Z_SCALE;
    // Compensate magnetometer error
    magnetom[0] = (magnetom[0] - MAGN_X_OFFSET) * MAGN_X_SCALE;
    magnetom[1] = (magnetom[1] - MAGN_Y_OFFSET) * MAGN_Y_SCALE;
    magnetom[2] = (magnetom[2] - MAGN_Z_OFFSET) * MAGN_Z_SCALE;
    // Compensate gyroscope error
    gyro[0] -= GYRO_AVERAGE_OFFSET_X;
    gyro[1] -= GYRO_AVERAGE_OFFSET_Y;
    gyro[2] -= GYRO_AVERAGE_OFFSET_Z;
}

void Compass_Heading()
{
  float mag_x;
  float mag_y;
  float cos_roll;
```

```
  float sin_roll;
  float cos_pitch;
  float sin_pitch;

  cos_roll = cos(roll);
  sin_roll = sin(roll);
  cos_pitch = cos(pitch);
  sin_pitch = sin(pitch);

  // Tilt compensated magnetic field X
  mag_x = magnetom[0] * cos_pitch + magnetom[1] * sin_roll * sin_pitch + magnetom[2] * cos_roll *
sin_pitch;
  // Tilt compensated magnetic field Y
  mag_y = magnetom[1] * cos_roll - magnetom[2] * sin_roll;
  // Magnetic Heading
  MAG_Heading = atan2(-mag_y, mag_x);
}
```

Control

```
////////////////////// DIAGRAMA DE BLOQUES /////////////////////////////
//          w              r           u                       x         y       //
//[step]----->[ K0 ]----->(+_)----->[ dx=Ax+Bu ]----->[ C ]-----> //
//                                ^                          |                 //
//                                 |                          |                 //
//                                 |------[ Klqr ]<-----------|                 //
////////////////////////////////////////////////////////////////////////////////////////

//Ley de Control -> u = K0 * r - Klqr * x
//Dimensiones
//u = [2x1]
//K0 = [2x2]
//r = [2x1]
//Klqr = [2x4]
//x = [4x1] = [d_theta, roll, fd_roll, d_phi]^T
//kx = Klqr*x
//kx[0] = klqr[0][0]*x[0] + klqr[0][1]*x[1] + klqr[0][2]*x[2] +
//       klqr[0][3]*x[3];
//kx[1] = klqr[1][0]*x[0] + klqr[1][1]*x[1] + klqr[1][2]*x[2] +
//       klqr[1][3]*x[3];
// u = K0 * r - kx
// u[0] = K0[0][0] * r[0] + K0[0][1] * r[1] - kx[0]
// u[1] = K0[1][0] * r[0] + K0[1][1] * r[1] - kx[1]
void control(void)
```

```
{
  x[0] = d_theta;
  x[1] = roll;
  x[2] = fd_roll;
  x[3] = d_phi;
  kx[0] = 0.0;
  kx[1] = 0.0;
  for(int i=0; i<2; i++)
  {
    for(int j=0; j<4; j++)
    {
      kx[i] += Klqr[i][j] * x[j];
    }
  }
  for (int i=0; i<2; i++)
  {
    u[i] = (K0[i][0] * R[0] + K0[i][1] * R[1] - kx[i])/1.5;
  }
}
```

Algoritmo DCM (Direction Cosine Matrix)

```
/* This file is part of the Razor AHRS Firmware */
// DCM algorithm - https://github.com/ptrbrtz/razor-9dof-ahrs
/**************************************************/
void Normalize(void)
{
  float error=0;
  float temporary[3][3];
  float renorm=0;
  error= -Vector_Dot_Product(&DCM_Matrix[0][0],&DCM_Matrix[1][0])*.5; //eq.19
  Vector_Scale(&temporary[0][0], &DCM_Matrix[1][0], error); //eq.19
  Vector_Scale(&temporary[1][0], &DCM_Matrix[0][0], error); //eq.19

  Vector_Add(&temporary[0][0], &temporary[0][0], &DCM_Matrix[0][0]);//eq.19
  Vector_Add(&temporary[1][0], &temporary[1][0], &DCM_Matrix[1][0]);//eq.19

  Vector_Cross_Product(&temporary[2][0],&temporary[0][0],&temporary[1][0]); // c= a x b //eq.20

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[0][0],&temporary[0][0])); //eq.21
  Vector_Scale(&DCM_Matrix[0][0], &temporary[0][0], renorm);

  renorm= .5 *(3 - Vector_Dot_Product(&temporary[1][0],&temporary[1][0])); //eq.21
  Vector_Scale(&DCM_Matrix[1][0], &temporary[1][0], renorm);
```

```
  renorm= .5 *(3 - Vector_Dot_Product(&temporary[2][0],&temporary[2][0])); //eq.21
  Vector_Scale(&DCM_Matrix[2][0], &temporary[2][0], renorm);
}
/**************************************************/
void Drift_correction(void)
{
  float mag_heading_x;
  float mag_heading_y;
  float errorCourse;
  //Compensation the Roll, Pitch and Yaw drift.
  static float Scaled_Omega_P[3];
  static float Scaled_Omega_I[3];
  float Accel_magnitude;
  float Accel_weight;

  //*****Roll and Pitch***************
  // Calculate the magnitude of the accelerometer vector
  Accel_magnitude = sqrt(Accel_Vector[0]*Accel_Vector[0] + Accel_Vector[1]*Accel_Vector[1] +
Accel_Vector[2]*Accel_Vector[2]);
  Accel_magnitude = Accel_magnitude / GRAVITY; // Scale to gravity.
  // Dynamic weighting of accelerometer info (reliability filter)
  // Weight for accelerometer info (<0.5G = 0.0, 1G = 1.0 , >1.5G = 0.0)
  Accel_weight = constrain(1 - 2*abs(1 - Accel_magnitude),0,1);  //

  Vector_Cross_Product(&errorRollPitch[0],&Accel_Vector[0],&DCM_Matrix[2][0]); //adjust the
ground of reference
  Vector_Scale(&Omega_P[0],&errorRollPitch[0],Kp_ROLLPITCH*Accel_weight);

  Vector_Scale(&Scaled_Omega_I[0],&errorRollPitch[0],Ki_ROLLPITCH*Accel_weight);
  Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);

  //*****YAW***************
  // We make the gyro YAW drift correction based on compass magnetic heading
  mag_heading_x = cos(MAG_Heading);
  mag_heading_y = sin(MAG_Heading);
  errorCourse=(DCM_Matrix[0][0]*mag_heading_y) - (DCM_Matrix[1][0]*mag_heading_x);
//Calculating YAW error
  Vector_Scale(errorYaw,&DCM_Matrix[2][0],errorCourse); //Applys the yaw correction to the XYZ
rotation of the aircraft, depeding the position.

  Vector_Scale(&Scaled_Omega_P[0],&errorYaw[0],Kp_YAW);//.01proportional of YAW.
  Vector_Add(Omega_P,Omega_P,Scaled_Omega_P);//Adding  Proportional.

  Vector_Scale(&Scaled_Omega_I[0],&errorYaw[0],Ki_YAW);//.00001Integrator
```

```
  Vector_Add(Omega_I,Omega_I,Scaled_Omega_I);//adding integrator to the Omega_I
}

void Matrix_update(void)
{
  Gyro_Vector[0]=GYRO_SCALED_RAD(gyro[0]); //gyro x roll
  Gyro_Vector[1]=GYRO_SCALED_RAD(gyro[1]); //gyro y pitch
  Gyro_Vector[2]=GYRO_SCALED_RAD(gyro[2]); //gyro z yaw

  Accel_Vector[0]=accel[0];
  Accel_Vector[1]=accel[1];
  Accel_Vector[2]=accel[2];

  Vector_Add(&Omega[0], &Gyro_Vector[0], &Omega_I[0]);  //adding proportional term
  Vector_Add(&Omega_Vector[0], &Omega[0], &Omega_P[0]); //adding Integrator term
  // Use drift correction
  Update_Matrix[0][0] = 0;
  Update_Matrix[0][1] = -G_Dt * Omega_Vector[2];//-z
  Update_Matrix[0][2] = G_Dt * Omega_Vector[1];//y
  Update_Matrix[1][0] = G_Dt * Omega_Vector[2];//z
  Update_Matrix[1][1] = 0;
  Update_Matrix[1][2] = -G_Dt * Omega_Vector[0];//-x
  Update_Matrix[2][0] = -G_Dt * Omega_Vector[1];//-y
  Update_Matrix[2][1]= G_Dt * Omega_Vector[0];//x
  Update_Matrix[2][2] = 0;

  Matrix_Multiply(DCM_Matrix,Update_Matrix,Temporary_Matrix); //a*b=c

  for(int x=0; x<3; x++) //Matrix Addition (update)
  {
    for(int y=0; y<3; y++)
    {
      DCM_Matrix[x][y]+=Temporary_Matrix[x][y];
    }
  }
}

void Euler_angles(void)
{
  pitch = -asin(DCM_Matrix[2][0]);
  roll = atan2(DCM_Matrix[2][1],DCM_Matrix[2][2]) + TO_RAD(desfase);
  yaw = atan2(DCM_Matrix[1][0],DCM_Matrix[0][0]);
}
```

Entrada De Datos Desde Matlab®

```
void input_matlab()
{
 union u_tag1{byte b1[4]; float fval1;} u1;
 union u_tag2{byte b2[4]; float fval2;} u2;

 if(Serial.available()>=10)
 {
  for(int i=0; i<=9; i++)
  {
   matlab_in[i] = Serial.read();
  }
  if(matlab_in[0] == 'Y')
  {
   if(matlab_in[9] == '\r')
   {
    for(int j=0; j<4; j++)
    {
     u1.b1[j] = matlab_in[j+1];
     u2.b2[j] = matlab_in[j+5];
    }
   }
  }
  R[0] = u1.fval1;
  R[1] = u2.fval2;
 }
 else
 {
  R[0] = 0;
  R[1] = 0;
 }
}
```

Algoritmos De Filtros

```
//Low pass butterworth filter order=1 alpha1=0.21221
float v[2];
float low_filter(float x)
{
 v[0] = v[1];
 v[1] = (0.4403591379666 * x) + (  0.1192817241 * v[0] );
 return (v[0] + v[1]);
}
//Low pass butterworth filter order=1 alpha1=0.05
float vv[2];
```

```
float low_filter2(float xx)
{
  vv[0] = vv[1];
  vv[1] = (0.1367287359973 * xx) + (0.7265425280 * vv[0] );
  return (vv[0] + vv[1]);
}

float y;
float low_filterx(float x)
{
  y += 0.05 * (x - y);
  return y;
}
```

Algoritmos Matemáticos

```
/* This file is part of the Razor AHRS Firmware */

// Computes the dot product of two vectors
float Vector_Dot_Product(const float v1[3], const float v2[3])
{
  float result = 0;

  for(int c = 0; c < 3; c++)
  {
    result += v1[c] * v2[c];
  }

  return result;
}

// Computes the cross product of two vectors
// out has to different from v1 and v2 (no in-place)!
void Vector_Cross_Product(float out[3], const float v1[3], const float v2[3])
{
  out[0] = (v1[1] * v2[2]) - (v1[2] * v2[1]);
  out[1] = (v1[2] * v2[0]) - (v1[0] * v2[2]);
  out[2] = (v1[0] * v2[1]) - (v1[1] * v2[0]);
}

// Multiply the vector by a scalar
void Vector_Scale(float out[3], const float v[3], float scale)
{
  for(int c = 0; c < 3; c++)
  {
```

```
    out[c] = v[c] * scale;
  }
}

// Adds two vectors
void Vector_Add(float out[3], const float v1[3], const float v2[3])
{
  for(int c = 0; c < 3; c++)
  {
    out[c] = v1[c] + v2[c];
  }
}

// Multiply two 3x3 matrices: out = a * b
// out has to different from a and b (no in-place)!
void Matrix_Multiply(const float a[3][3], const float b[3][3], float out[3][3])
{
  for(int x = 0; x < 3; x++)  // rows
  {
    for(int y = 0; y < 3; y++)  // columns
    {
      out[x][y] = a[x][0] * b[0][y] + a[x][1] * b[1][y] + a[x][2] * b[2][y];
    }
  }
}

// Multiply 3x3 matrix with vector: out = a * b
// out has to different from b (no in-place)!
void Matrix_Vector_Multiply(const float a[3][3], const float b[3], float out[3])
{
  for(int x = 0; x < 3; x++)
  {
    out[x] = a[x][0] * b[0] + a[x][1] * b[1] + a[x][2] * b[2];
  }
}

// Init rotation matrix using euler angles
void init_rotation_matrix(float m[3][3], float yaw, float pitch, float roll)
{
  float c1 = cos(roll);
  float s1 = sin(roll);
  float c2 = cos(pitch);
  float s2 = sin(pitch);
  float c3 = cos(yaw);
```

```
  float s3 = sin(yaw);

  // Euler angles, right-handed, intrinsic, XYZ convention
  // (which means: rotate around body axes Z, Y', X")
  m[0][0] = c2 * c3;
  m[0][1] = c3 * s1 * s2 - c1 * s3;
  m[0][2] = s1 * s3 + c1 * c3 * s2;

  m[1][0] = c2 * s3;
  m[1][1] = c1 * c3 + s1 * s2 * s3;
  m[1][2] = c1 * s2 * s3 - c3 * s1;

  m[2][0] = -s2;
  m[2][1] = c2 * s1;
  m[2][2] = c1 * c2;
}
```

Salida De Datos

```
void output_angles()
{
   Serial.print("#Estados=");
   Serial.print(roll); Serial.print(",");
   Serial.print(fd_roll); Serial.print(",");
   Serial.print(theta); Serial.print(",");
   Serial.print(d_theta); Serial.print(",");
   Serial.print(phi); Serial.print(",");
   Serial.print(d_phi); Serial.print(",");
   Serial.print(u[0]); Serial.print(",");
   Serial.print(u[1]); Serial.print(",");
   Serial.println(G_Dt);
}

void output_matlab()
{
  float estados[6];
  estados[0] = fd_roll;
  estados[1] = theta;
  estados[2] = d_theta;
  estados[3] = d_phi;
  estados[4] = u[0];
  estados[5] = u[1];
  Serial.write("Y");              //Header -> LF ('\n')
  Serial.write((byte*) estados, 24); //Data block
  Serial.write("\r");             //Terminator -> CR ('\r')
```

```
}

void printData()
{
  Serial.println("-----------------------------");
  Serial.print("#Acc(X,Y,Z)"); Serial.print('=');
  Serial.print(accel[0]); Serial.print(",");
  Serial.print(accel[1]); Serial.print(",");
  Serial.print(accel[2]); Serial.println();

  Serial.print("#Mag(X,Y,Z)"); Serial.print('=');
  Serial.print(magnetom[0]); Serial.print(",");
  Serial.print(magnetom[1]); Serial.print(",");
  Serial.print(magnetom[2]); Serial.println();

  Serial.print("#Gyro"); Serial.print('=');
  Serial.print(gyro[0]); Serial.print(",");
  Serial.print(gyro[1]); Serial.print(",");
  Serial.print(gyro[2]); Serial.println();
}
```

PWM

```
void PWM_motores(float VL, float VR)
{
  PWM_L = long_map(abs(VL),0,15,45,255);
  PWM_R = long_map(abs(VR),0,15,45,255);
  analogWrite(ENA,PWM_L);
  analogWrite(ENB,PWM_R);
  if(VL<0)  //Left Reverse
  {
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
  }
  else      //Left Fordward
  {
    digitalWrite(IN1,HIGH);
    digitalWrite(IN2,LOW);
  }
  if(VR<0)  //Right Reverse
  {
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,HIGH);
  }
  else      //Right Fordward
```

```
  {
   digitalWrite(IN3,HIGH);
   digitalWrite(IN4,LOW);
  }
}

long long_map(long x, long in_min, long in_max, long out_min, long out_max)
{
 return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Sensores

```
//////////////////////////SENSORES//////////////////////////

#define ACC_add 83
#define MAG_add 30
#define GYR_add 104

/********************************** ACCELEROMETER
***************************************/
void Accel_Init()
{
 Wire.beginTransmission(ACC_add);
 Wire.write((byte) 0x2D); //Power Register
 Wire.write((byte) 0x08); //Measurement Mode
 Wire.endTransmission();
 delay(5);
 Wire.beginTransmission(ACC_add);
 Wire.write((byte) 0x31);  // Data format register
 Wire.write((byte) 0x08);  // Set to full resolution
 Wire.endTransmission();
 delay(5);
 // Because our main loop runs at 50Hz we adjust the output data rate to 50Hz (25Hz bandwidth)
 Wire.beginTransmission(ACC_add);
 Wire.write((byte) 0x2C);  // Rate Register
 Wire.write((byte) 0x09);  // Set to 50Hz, normal operation
 Wire.endTransmission();
 delay(5);
}

void Read_Accel()
{
 int i = 0;
 byte buff[6];
```

```
  Wire.beginTransmission(ACC_add);
  Wire.write((byte) 0x32);  // Send address to read from
  Wire.endTransmission();
  Wire.beginTransmission(ACC_add);
  Wire.requestFrom(ACC_add, 6);  // Request 6 bytes
  while(Wire.available())  // ((Wire.available())&&(i<6))
  {
   buff[i] = Wire.read();  // Read one byte
   i++;
  }
  Wire.endTransmission();
  if (i == 6)  // All bytes received?
  {
   // No multiply by -1 for coordinate system transformation here, because of double negation:
   // We want the gravity vector, which is negated acceleration vector.
   accel[0] = (int16_t)((((int) buff[3]) << 8) | buff[2]);  // X axis (internal sensor y axis)
   accel[1] = (int16_t)((((int) buff[1]) << 8) | buff[0]);  // Y axis (internal sensor x axis)
   accel[2] = (int16_t)((((int) buff[5]) << 8) | buff[4]);  // Z axis (internal sensor z axis)
  }
  else
  {
   Serial.println("!ERR: reading accelerometer");
  }
}

/********************************** MAGNETOMETER
************************************/
void Magn_Init()
{
  Wire.beginTransmission(MAG_add);
  Wire.write((byte) 0x02);   // Access Mode Register
  Wire.write((byte) 0x00);   // Set continuous mode (default 10Hz)
  Wire.endTransmission();
  delay(5);
  Wire.beginTransmission(MAG_add);
  Wire.write((byte) 0x00);          // Access Configuration Register A
  Wire.write(0b00011000);    // Set 1MpO - 75Hz - Normal Measurement
  Wire.endTransmission();
  delay(5);
}

void Read_Magn()
{
  int i = 0;
```

```
 byte buff[6];

 Wire.beginTransmission(MAG_add);
 Wire.write(0x03);  // Send address to read from
 Wire.endTransmission();
 Wire.beginTransmission(MAG_add);
 Wire.requestFrom(MAG_add, 6);  // Request 6 bytes
 while(Wire.available())        // ((Wire.available())&&(i<6))
 {
  buff[i] = Wire.read();  // Read one byte
  i++;
 }
 Wire.endTransmission();
 if (i == 6)  // All bytes received?
 {
  // 9DOF Sensor Stick SEN-10724 using HMC5883L magnetometer
  // MSB byte first, then LSB; Y and Z reversed: X, Z, Y
  magnetom[0] = (int16_t)((((int) buff[0]) << 8) | buff[1]);         // X axis (internal sensor x axis)
  magnetom[1] = -1 * (int16_t)((((int) buff[4]) << 8) | buff[5]);  // Y axis (internal sensor -y axis)
  magnetom[2] = -1 * (int16_t)((((int) buff[2]) << 8) | buff[3]);  // Z axis (internal sensor -z axis)
 }
 else
 {
  Serial.println("!ERR: reading magnetometer");
 }
}

/********************************** GYROSCOPE
*************************************/
void Gyro_Init()
{
 // Power up reset defaults
 Wire.beginTransmission(GYR_add);
 Wire.write((byte) 0x3E);      //Access PWR_MGM Register
 Wire.write((byte) 0x80);      //Reset device, No low power, Normal, Normal, Normal, Internal Oscillator
 Wire.endTransmission();
 delay(5);
 // Select full-scale range of the gyro sensors
 // Set LP filter bandwidth to 42Hz
 Wire.beginTransmission(GYR_add);
 Wire.write((byte) 0x16);  //Access DLPF, Full Scale Register
 Wire.write((byte) 0x1B);  // FS_SEL = 3, DLPF_CFG = 3
 Wire.endTransmission();
 delay(5);
```

```
  // Set sample rato to 50Hz
  Wire.beginTransmission(GYR_add);
  Wire.write((byte) 0x15);  // Access Sample Rate Divider Register
  Wire.write((byte) 0x0A);  // SMPLRT_DIV = 10 (50Hz)
  Wire.endTransmission();
  delay(5);
  // Set clock to PLL with z gyro reference
  Wire.beginTransmission(GYR_add);
  Wire.write((byte) 0x3E);  //Access PWR_MGM Register
  Wire.write((byte) 0x00);  //Reset to Default setings
  Wire.endTransmission();
  delay(5);
}

// Reads x, y and z gyroscope registers
void Read_Gyro()
{
  int i = 0;
  byte buff[6];

  Wire.beginTransmission(GYR_add);
  Wire.write((byte) 0x1D);  // Sends address to read from GYRO_XOUT_H Register
  Wire.endTransmission();
  Wire.beginTransmission(GYR_add);
  Wire.requestFrom(GYR_add, 6);  // Request 6 bytes
  while(Wire.available())  // ((Wire.available())&&(i<6))
  {
    buff[i] = Wire.read();  // Read one byte
    i++;
  }
  Wire.endTransmission();
  if (i == 6)  // All bytes received?
  {
    gyro[0] = -1 * (int16_t)((((int) buff[2]) << 8) | buff[3]);    // X axis (internal sensor -y axis)
    gyro[1] = -1 * (int16_t)((((int) buff[0]) << 8) | buff[1]);    // Y axis (internal sensor -x axis)
    gyro[2] = -1 * (int16_t)((((int) buff[4]) << 8) | buff[5]);    // Z axis (internal sensor -z axis)
  }
  else
  {
    Serial.println("!ERR: reading gyroscope");
  }
}
```

BIBLIOGRAFÍA

[1] Y. Yamamoto, «NXTway-GS Model-Base Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT -,» 2009.

[2] «Self-balancing Robot,» Googol Technology (HK) Ltd., 2008.

[3] R. C. Ooi, «Balancing a Two-Wheeled Autonomous Robot,» 2003.

[4] J. Wu, W. Zhang y S. Wang, «A Two-Wheeled Self-Balancing Robot with the Fuzzy PD Control Method,» 2012.

[5] M. R. Bageant, «Balancing a Two-Wheeled Segway Robot,» Massachusetts Institute of Technology, Cambridge, 2011.

[6] S. Balasubramanian y M. N. Lathiff, «Self Balancing Robot,» The University of British Columbia, 2011.

[7] W. An y Y. Li, «Simulation and Control of a Two-wheeled Self-balancing Robot,» *International Conference on Robotics and Biomometics (ROBIO),* 2013.

[8] A. Castro, «Modeling And Dynamic Analysis Of A Two-Wheeled Inverted-Pendulum,» Georgia Institute of Technology, Georgia, 2012.

[9] M. Arvidsson y J. Karlsson, «Design, construction and verification of a self-balancing vehicle,» Chalmers University of Technology, Göteborg, Sweden, 2012.

[10] S. W. Nawawi, M. N. Ahmad y J. H. S. Osman, «Development of a Two-Wheeled Inverted Pendulum Mobile Robot,» The 8th Student Conference on Research and Development, Malaysia, 2007.

[11] Y. Kim, S. H. Kim y Y. K. Kwak, «Dynamic Analysis of a Nonholonomic Two-Wheeled Inverted Pendulum Robot,» *Journal of Intelligent and Robotic Systems,* pp. 25-46, 2005.

[12] F. Grasser, A. D'Arrigo, S. Colombi y A. Rufer, «JOE: A Mobile, Inverted Pendulum,» 2002.

[13] C. R. Magers y A. N. Gündes, «Low Order Decentralized Stabilixing Controller Design for a Mobile Inverted Pendulum Robot,» *2009 American Control Conference,* pp. 1-2, 2009.

[14] A. Hernández, M. Legaspi y J. Peláez, «Control Inteligente del Péndulo Invertido,» Universidad Complutense de Madrid, Madrid, 2012.

[15] T. Gutierrez y S. P., «Péndulo Invertido sobre dos ruedas,» Benemérita Universidad Autónoma de Puebla, Puebla, México.

[16] S. W. Nawawi, M. N. Ahmad y J. H. S. Osman, «Control of Two-wheels Inverted Pendulum Mobile Robot Using Full Order Sliding Mode Control,» de *Proceedings of International Conference on Man-Machine Systems*, Langkawi, Malaysia, 2006.

[17] J. S. Hu, J. J. Wang y G. C. Sun, «Self-balancing Control and Manipulation of a Glove Puppet Robot on a Two-wheel Mobile Platform,» *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2009.

[18] T. Tomasic, A. Demetlika y M. Crnekovic, «Self-Balancing Mobile Robot Tilter,» University of Zagreb, Zagreb, Croatia, 2012.

[19] B. Bonafilia, N. Gustafsson y S. Nilsson, «Self-balancing two-wheeled robot,» Chalmers University of Technology, Göteborg, Sweden, 2013.

[20] M. Triverio, «Progetto E Implementazione Del Sistema Di Controllo Per Un Pendolo Inverso,» Politecnico di Milano, Milano, 2008.

[21] K. Pathak, JaumeFranch y S. K. Agrawal, «Velocity and Position of a Wheeled Inverted Pendulum by Partial Feedback Linearization,» *IEEE TRANSACTIONS ON ROBOTICS,* vol. 21, nº 3, pp. 505-513, 2005.

[22] P. k. Tripathy, «Self-Balancing Bot Using Concept of Inverted Pendulum,» National Institute of Technology Rourkela, Rourkela, India, 2013.

[23] http://www.geology.smu.edu/~dpa-www/robo/nbot/

[24] http://www.teamhassenplug.org/robots/legway/

[25] http://www.arrickrobotics.com/robomenu/balibot.html

[26] http://www.philohome.com/nxtway/nxtway.htm

[27] Dr. E. G. Rocha Cózatl, «Caracterización de un motor de CD - Control Automático»

[28] www.sparkfun.com

[29] http://www.servocity.com/html/actoboticstm.html

[30] https://github.com/ptrbrtz/razor-9dof-ahrs/wiki/Tutorial

[31] https://github.com/ptrbrtz/razor-9dof-ahrs