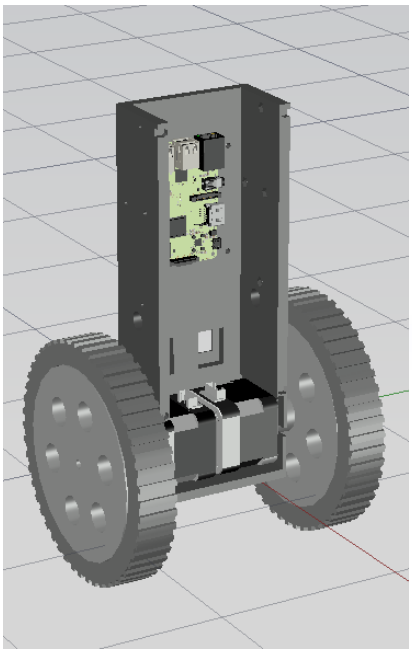
Máster en Ingeniería de Sistemas y Control

Construcción y control de robot de auto balanceo

Ricardo Camacho Calvo

Directores

Luis de la Torre Cubillo y Dictino Chaos García



Curso 2021-2022

Septiembre 2022

Máster en Ingeniería de Sistemas y Control

Construcción y control de robot de auto balanceo

Alumno: Ricardo Camacho Calvo

Directores: Luis de la Torre Cubillo y Dictino Chaos García

Dpto. de Informática y Automática UNED

Autorización

Autorizo a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Ricardo Camacho Calvo

Resumen

En este proyecto se ha diseñado, construido y controlado un robot de auto balanceo de dos ruedas. El robot utiliza una placa Raspberry Pi y motores paso a paso.

Se ha estudiado la representación del robot en el espacio de estados y el diseño de un controlador PID. Se ha diseñado un filtro de Kalman para la integración de las medidas del sensor giroscópico y acelerómetro.

Durante las pruebas del robot este se ha tenido que reemplazar por un robot Tumblller de la marca Elegoo. Se han realizado las modificaciones al código y controladores pertinentes.

Finalmente, se han estudiado las posibilidades de mejora del robot diseñado y se han discutido los pasos a seguir para la finalización del robot.

Palabras clave

Robótica

Control

Ingeniería de sistemas

Diseño de sistemas

Espacio de estados

PID

Filtro de Kalman

Integración de sensores

Raspberry Pi

Arduino

Índice

Lista de figuras

1. Introducción

Los robots de auto balanceo son una plataforma para el desarrollo y análisis de algoritmos de control debido a la dinámica inestable que presentan por naturaleza. Esta configuración presenta una dinámica análoga a la de un sistema de péndulo invertido.

Estos robots tienen la característica de poder girar sobre su propio eje, sin necesidad de desplazarse, lo cual les proporciona una ventaja de maniobrabilidad, pudiendo realizar giros pronunciados y navegar por superficies estrechas. Las ventajas de estos robots tienen el potencial de solventar problemas comunes en la industria y la sociedad. Como ejemplo, esta configuración se utiliza en los sistemas de transporte de tipo 'segway'. El estudio del control de estos sistemas forma la base para controlar sistemas más complejos como robots humanoides y robots bípedos con ruedas.

El objetivo principal de este trabajo es la construcción de un robot con dos ruedas paralelas capaz de mantener el equilibrio, así como el estudio de las estrategias de control necesarias para su correcto funcionamiento y el diseño e implementación de un controlador que permita al robot mantenerse erguido.

Los objetivos específicos del proyecto son los siguientes:

- Diseño del circuito del robot, incluyendo la elección de los componentes necesarios para la implementación del controlador, la actuación de los motores seleccionados y la medición de los parámetros deseados.
- Diseño de la estructura del robot que aloje los componentes y permita una fácil manipulación y reemplazo de los componentes en caso de ser necesario.
- Montaje del robot.
- Representación del robot en el espacio de estados y caracterización del sistema.
- Diseño e implementación de un controlador que permita que el robot se mantenga erguido.
- Diseño de un paquete de software que sirva como plataforma de integración de las funcionalidades del robot.

Para cumplir los objetivos del trabajo, se realizará un estudio de la literatura disponible relacionada con el control de sistemas de péndulo invertido.

Se analizarán las diferentes opciones que existen para la selección del controlador y los sensores adecuados para el proyecto. Se seleccionarán y dimensionarán los motores necesarios para el movimiento del robot. También se dimensionará una fuente de alimentación acorde con los requisitos de potencia del circuito.

Se diseñará una estructura que permita alojar todos los componentes de forma ordenada, permitiendo acceder a ellos en caso de ser necesario y manteniéndose estos estables y fijos durante la ejecución de las pruebas.

Se diseñará un paquete de software que permita un fácil control de las velocidades de los motores y una lectura de los sensores simple mediante el uso de las librerías

necesarias y el desarrollo de funciones que sirvan como plataforma para una fácil implementación del controlador.

Se diseñará un filtro de Kalman para la integración de dos sensores que permita una mejor medición del ángulo del cuerpo del robot.

Se analizarán los diferentes controladores entre los que se elegirá un controlador que cumpla los requisitos del sistema para una correcta estabilización del robot. Se representará el sistema en el espacio de estados y se obtendrá la función de transferencia del sistema para la simulación del sistema. En base a esta simulación del sistema se diseñará un controlador Proporcional-Integral-Derivativo (PID) que cumpla los requisitos.

Finalmente, se implementará el controlador en el software previamente diseñado y se realizarán las pruebas necesarias para comprobar el correcto funcionamiento de este.

Para ello, en este proyecto se analizarán las diferentes alternativas de diseño para la locomoción y medición de inclinación del robot, así como las alternativas de hardware de control disponibles.

2. Estado del arte y revisión literaria

En este apartado se presenta el estado del arte relacionado con el diseño de un robot de auto balanceo analizado como un sistema de péndulo invertido.

2.1. Péndulo invertido

El problema de control que supone un sistema de péndulo invertido consiste en el equilibrio de un centro de masas que se sitúa por encima de un punto de pivotaje (ver ilustración 1). Mientras que el sistema de péndulo clásico tiende a la estabilidad cuando este se encuentra en posición vertical, el péndulo invertido es un sistema inherentemente inestable y que debe ser controlado activamente [1].

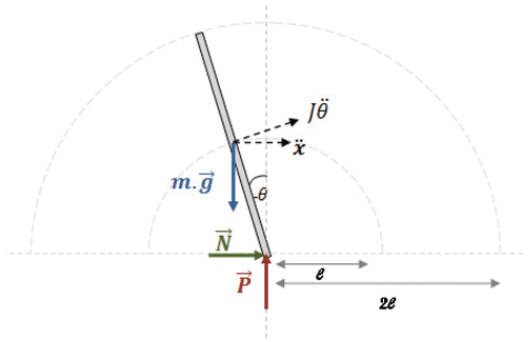


Ilustración 1: Sistema de péndulo invertido [1]

El problema clásico consiste en anclar una vara a un eje en un carro, lo cual restringe el movimiento de la vara a un grado de libertad (ver ilustración 2).

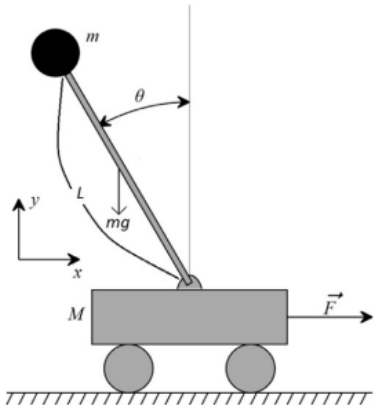


Ilustración 2: Sistema de péndulo invertido sobre carro [19]

Existen variaciones del problema que se extienden a dos grados de libertad, tradicionalmente mediante una bola situada debajo del sistema y controlada en todas direcciones.

El primer ejemplo desarrollado de este sistema es el robot Ballbot, construido en 2005 en la Universidad Carnegie Mellon [2]. Sin embargo, el ejemplo más famoso de este tipo de robot es el robot BB-8 de las películas de '*Star Wars'* [3].

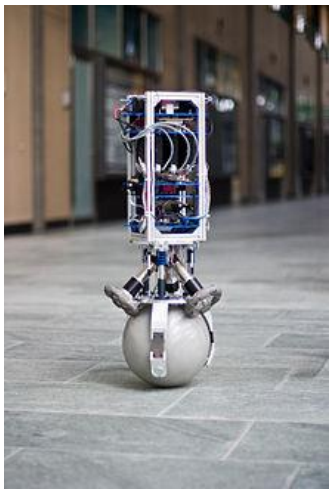


Ilustración 3: Rezero. desarrollado por la universidad ETH de Zurich [4]

Adicionalmente, el problema de control del péndulo invertido ha servido como base para el desarrollo del control de sistemas más complejos. En la conferencia internacional de robots inteligentes y sistemas de 2010 se presentó un control de estabilización de postura de un robot bípedo basado en un modelo de péndulo invertido lineal [20]. En este robot HRP-4C, desarrollado en el Instituto Nacional de Ciencias Industriales Avanzadas y Tecnología de Japón, el cuerpo se modela como un péndulo invertido. De esta manera, se controla la orientación del cuerpo de la misma forma que se controla el ángulo del péndulo invertido mediante un controlador de espacio de estados basado en asignación de polos.

Las estrategias de control del problema de péndulo invertido clásico se basan en aplicar un torque en el punto de pivotaje, en mover el punto de pivotaje de manera horizontal y perpendicular al eje, generando así un torque en el péndulo o en oscilar el punto de pivotaje verticalmente, también generando un torque en el punto de pivotaje.

Un ejemplo clásico de péndulo invertido controlado al mover el punto de pivotaje es el de equilibrar una escoba o martillo en la palma de la mano.

2.2. Robots de auto balanceo

Un robot de auto balanceo es un robot de dos ruedas paralelas que se mantiene erguido y estable. Estos robots son análogos al modelo de péndulo invertido sobre un carro, en el que se mueve el punto de pivotaje en el eje horizontal y perpendicular al eje de pivotaje.

La diferencia entre el modelo de péndulo invertido sobre carro y el robot de auto balanceo es que el centro de masas del robot completo corresponde a la masa al final del péndulo y el eje de las ruedas corresponde a su vez al punto de pivotaje del péndulo.

Debido a la inestabilidad y no linealidad que presentan estos robots, se han usado como plataforma de desarrollo de sistemas de control.

En el Industrial Electronics Laboratory del instituto de tecnología suizo se diseñó en 2002 un prototipo de robot de auto balanceo con dos ruedas estabilizado por un controlador de espacio de estados linear [5].

La principal limitación de este robot es que el controlador es fijo y está calculado para las características del sistema, no pudiéndose adaptar a variaciones de este como introducción de peso adicional. Esta limitación puede ser sobrepasada con controladores adaptativos como controladores basados en lógica difusa.

Ilustración 4: Robot JOE desarrollado en la EPFL [5]

En el entorno industrial y comercial, versiones de estos sistemas están disponibles para su uso en forma de aparatos de movilidad. Como ejemplo, la empresa 'SEGWAY INC.' desarrolla desde 2001 el sistema de transporte Segway [6].

Este robot presenta una mejora sustancial en el control de sistemas de péndulo invertido ya que es capaz de adaptarse a modificaciones del sistema en la forma de pilotos de diferentes pesos, siendo el sistema capaz de calibrar su controlador interno automáticamente para mantenerse funcional.

Ilustración 5: Sistema de transporte Segway auto balanceante [6]

En el mundo de la robótica educativa y robótica como hobby se han desarrollado muchas versiones de estos sistemas a bajo coste como el kit de robótica educativa 'Tumblller' desarrollado por la empresa ELEGOO en 2020 [7]. El fin de este robot es proporcionar una plataforma de aprendizaje para robótica a varios niveles, contando con un paquete de código precargado en el controlador Arduino en el que se basa. Este robot está controlado por un controlador PID y obtiene mediciones del ángulo de su cuerpo mediante una integración de sensor giroscópico y acelerómetro.

La principal limitación del robot reside en el controlador utilizado, que tiene una capacidad de memoria y procesamiento limitadas, lo cual imposibilita el uso de controladores con mayores requisitos de procesamiento.

De nuevo, como el robot JOE, este robot no es capaz de adaptarse a modificaciones de peso que se puedan introducir.

Ilustración 6: Robot educativo Tumblller [7]

2.3. Filtro de Kalman

En 1960 se publicó el estudio 'A New Approach to Linear Filtering and Prediction Problems', escrito por R.E. Kalman [8]. En este estudio se propuso un proceso que pretendía mejorar los sistemas de filtrado de señales de la época. Este filtro es conocido como el filtro de Kalman o estimación cuadrática lineal (LQE).

El proceso propuesto es un algoritmo que utiliza una serie de medidas en el tiempo para producir una estimación de variables desconocidas en momentos posteriores. En cada paso de la ejecución se actualiza la predicción del modelo con los datos reales obtenidos del sistema.

Entre las aplicaciones de la estimación cuadrática lineal se encuentran sistemas de guía, navegación y control de vehículos. El filtro es especialmente útil en vehículos aéreos y marítimos, donde es usado para facilitar la posición dinámica.

El algoritmo consiste en dos fases. La primera fase es la predicción de las variables de estado del sistema con sus respectivas incertidumbres. En la siguiente fase, tras recibir las medidas reales de las variables de estado predichas, se actualizan las predicciones mediante un sistema de media ponderada entre la predicción y el valor real. De esta manera, el algoritmo recorre estas dos fases de forma recursiva pudiendo ser utilizado en tiempo real, ya que solo necesita el valor actual y previo de las variables de estado del sistema.

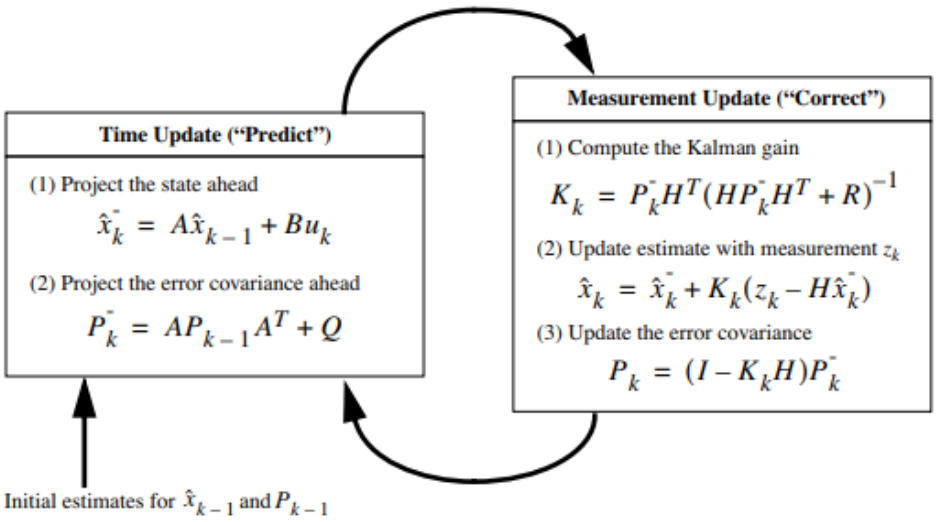


Ilustración 7: Algoritmo del filtro de Kalman [9]

Para el correcto funcionamiento del filtro de Kalman se realizan dos suposiciones. En primer lugar, se supone que el sistema dinámico es linear. En segundo lugar, se presupone que los errores tienen forma gaussiana con media cero.

Existen variaciones del filtro de Kalman que se pueden aplicar a sistemas físicos no lineales. Sin embargo, se puede aplicar el filtro original en sistemas no lineales previamente linealizados.

2.3.1 Aplicación del filtro de Kalman en integración de sensores

Ciertas aplicaciones en el ámbito del control requieren de mediciones extremadamente precisas para un correcto funcionamiento del sistema. Para ello se pueden utilizar sensores con una alta sensibilidad, precisión y resistencia a perturbaciones.

Sin embargo, una forma de obtener mediciones precisas mediante sensores que no presentan características físicas tan fiables es la integración de sensores. Mediante la integración de diferentes sensores que miden el mismo fenómeno de maneras distintas se puede obtener una medición más precisa sin necesidad de equipamiento costoso.

Para integrar sistemas multisensor se puede utilizar el filtro de Kalman. Este filtro se puede utilizar para combinar datos redundantes provenientes de diferentes sensores que miden la misma magnitud.

En 1996 Borenstein y Feng publicaron el estudio “Medición y corrección de errores sistemáticos de odometría en robots móviles” en el que se propone la integración de datos obtenidos por sonar y datos obtenidos por odometría para la navegación de robots móviles [10].

Este estudio se centró en robots de dos ruedas con locomoción diferencial y los errores en mediciones de odometría derivados de inexactitudes en el sistema real como desalineaciones en los ejes y de perturbaciones en la ejecución esperada del robot como suelos resbaladizos.

Una de las soluciones propuestas en este estudio fue la integración de medición por sonar en las mediciones odométricas de la posición del robot para corregir los errores en la medición de la posición. Esta integración se realiza mediante un filtro de Kalman.

De forma similar, se pueden integrar las mediciones de un giróscopo y un acelerómetro para obtener medidas más precisas en la navegación de robots de dos ruedas [11].

2.4. Sistemas de control

Para un correcto control de un sistema es necesario estudiar las alternativas de controlador que pueden ser utilizadas para su fin.

En la extensa literatura existente respecto al control de sistemas de péndulo invertido predominan tres estrategias de control: el controlador por asignación de polos, el controlador regulador lineal cuadrático y el controlador proporcional-integral-derivativo (PID).

2.4.1.Controlador por asignación de polos

El método de asignación de polos o control por realimentación de estados consiste en una estrategia de control por realimentación en la que se colocan los polos de un sistema en lugares predefinidos del plano S.

El objetivo del controlador es colocar los polos del sistema en bucle cerrado en lugares deseados para obtener las características de respuesta del sistema requeridas.

Partiendo de un sistema representado en el espacio de estados:

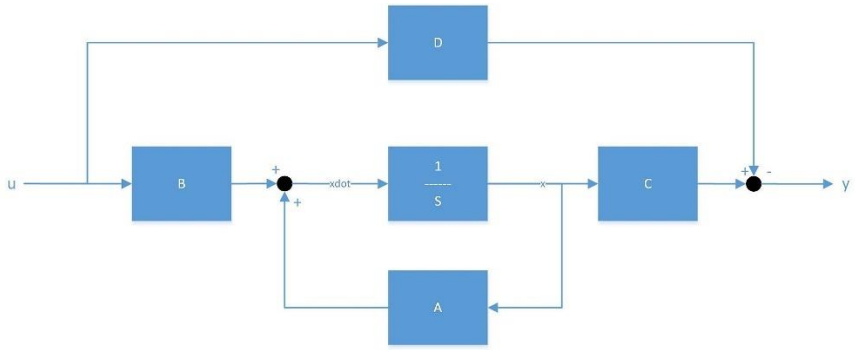


Ilustración 8: Sistema en bucle abierto [12]

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

$$y = Cx + Du$$

Se pueden obtener los polos del sistema como:

$$|sI - A| = 0$$

Si se introduce un controlador en la realimentación:

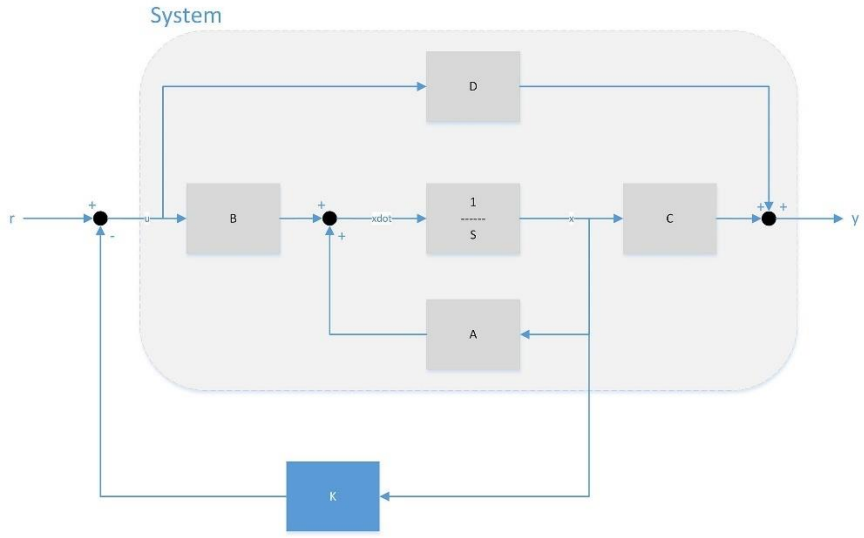


Ilustración 9: Sistema en bucle cerrado con controlador [12]

Se pueden calcular los polos del sistema en bucle cerrado como:

$$|sI - (A - BK)| = 0$$

De esta manera, se puede diseñar los valores de la matriz K para que el sistema presente los polos deseados en bucle cerrado, cancelando así los polos del sistema original y obteniendo la respuesta temporal del sistema que se quiera.

Un ejemplo de un robot de auto balanceo controlado con esta estrategia es el robot JOE explicado en anteriores apartados [5].

2.4.2.Controlador regulador lineal cuadrático

El control basado en el regulador lineal cuadrático parte de una descripción del sistema en forma de ecuaciones diferenciales lineares, a las cuales se les asigna un coste mediante la función cuadrática.

Teniendo un sistema con la forma:

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

Se define la función de coste como:

$$J = \int_0^\infty \left(x_{Ref} - x(t)\right)^T Q \left(x_{Ref} - x(t)\right) + u(t)^T R u(t) dt$$

Donde Q y R son las matrices encargadas de eliminar el error entre la salida deseada y el controlador.

Se puede obtener el controlador K mediante la función:

$$-Kx = -R^{-1}B^T P x$$

Donde P es la solución de la ecuación:

$$PA + A^T P + Q - PBR^{-1}B^T P = 0$$

El regulador lineal cuadrático se ha utilizado para controlar robots de auto balanceo de tipo ‘Segway’ [28].

2.4.3.Controlador proporcional-integral-derivativo

El controlador proporcional-integral-derivativo es un controlador de tres componentes que aplica una corrección al valor de salida en base a unas ganancias proporcional, integral y derivativa para acercar la salida a un valor deseado en bucle cerrado (ver ilustración 9).

La forma matemática del controlador es la siguiente:

$$u(t) = K_P e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

Donde e(t) corresponde al error de la salida del sistema con respecto a la salida deseada, es decir:

$$e(t) = r(t) - y(t)$$

El diseño de un controlador PID se basa en la obtención de los parámetros Kp, Ki y Kd que hagan que la respuesta temporal del sistema deseada se satisfaga.

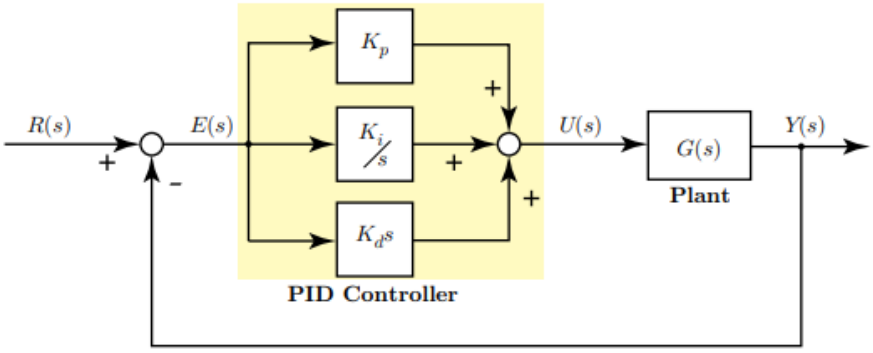


Ilustración 10: Controlador PID [13]

La ventaja del uso de controladores PID frente a los mostrados en los apartados anteriores es que la forma del controlador PID hace que los efectos físicos que producen los parámetros a definir (Kp,Ki,Kd) tengan un sentido físico más intuitivo que los controladores anteriores.

La ganancia proporcional (Kp) corresponde a la magnitud de respuesta con respecto a la magnitud del error. La ganancia integral (Ki) corresponde a la magnitud de respuesta a la acumulación del error residual en momentos pasados. La ganancia derivativa (Kd) corresponde a la magnitud de respuesta a la estimación del error futuro en base a la variación del error.

Los controladores PID son ampliamente utilizados en el control de robots de auto balanceo, especialmente en proyectos de robótica educativa [7].

En este proyecto se utilizará un controlador PID debido a su simpleza y la amplia documentación existente respecto a este tipo de controlador, lo cual facilita su implementación y modificación.

3. Diseño del robot de auto balanceo

El diseño del robot de auto balanceo se ha basado en la elección de los componentes disponibles y la integración de estos con el objetivo de crear una plataforma de prueba estable y fiable que permitiera diseñar un controlador adecuado para el sistema.

Los componentes mínimos necesarios para la construcción de un robot de auto balanceo son actuadores, sensores, controlador y fuente de alimentación.

Los actuadores deben ser dos motores que puedan ser controlados en velocidad.

Los sensores deben permitir el control de los motores y la medición del ángulo de inclinación del robot.

El controlador, debe tener la capacidad de realizar los cálculos necesarios para controlar los motores en base a las mediciones recibidas por los sensores ,

La fuente de alimentación debe tener la potencia suficiente para alimentar todos los componentes, debe tener la capacidad suficiente para permitir la realización de pruebas durante un periodo prolongado de tiempo y debe ser independiente para permitir el libre movimiento del robot.

3.1. Selección de los componentes

En este apartado se presentan las alternativas que se han tenido en cuenta para elegir los diferentes componentes que se han utilizado para el desarrollo del robot.

3.1.1 Selección de los actuadores

Dado que el diseño del robot se realiza desde cero, la selección de los actuadores es completamente libre y se ha basado en mantener un equilibrio entre tamaño y potencia con el objetivo de que tengan la fuerza necesaria para mover el robot completo sin requerimientos elevados de potencia, lo que sería perjudicial al aumentar el tamaño de la fuente de alimentación y del robot en general.

En la elección de los actuadores hay dos tipos de motores que se han tenido en cuenta, motores paso a paso y motores de corriente continua.

La ventaja principal de los motores paso a paso es que no requieren de sensores adicionales para su control, ya que su control en velocidad se puede realizar mediante el control de los pasos de entrada. Otra ventaja destacable es la precisión en el movimiento que se puede obtener con motores paso a paso frente a motores de corriente continua.

Ilustración 11. Motor paso a paso [21]

La ventaja principal de los motores de corriente continua frente a motores paso a paso es la alta velocidad que ofrecen y el hecho de que el torque que proporcionan es mayor a altas velocidades. La desventaja de la necesidad de encoder para el control de estos motores en velocidad puede ser evitada con un sistema de motor DC con encoder integrado.

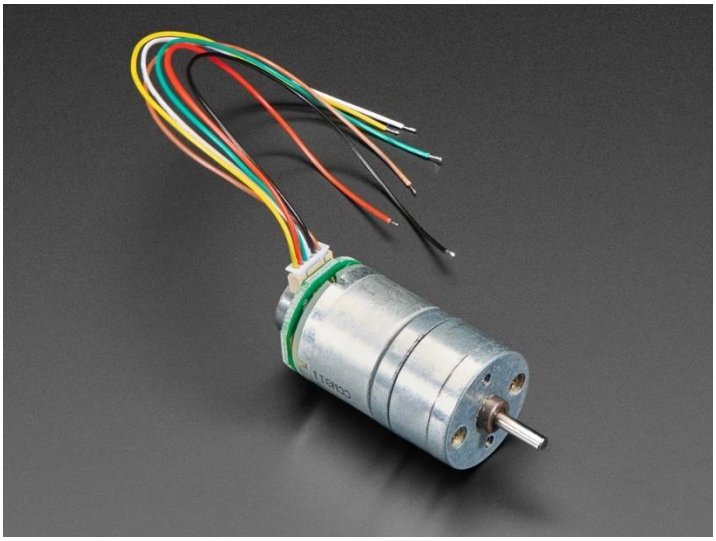


Ilustración 12: Motor de corriente continua con encoder [22]

Para este proyecto se han seleccionado motores paso a paso de los que se disponía previamente.

Los motores seleccionados finalmente han sido motores paso a paso Nema 17, mostrado en la ilustración 11, provenientes de una impresora 3D. Estos motores tienen un par de 0.22Nm y una precisión de 1.8 grados por paso con una necesidad de 1.5A de corriente y 12-24V, lo cual será importante en el diseño de la fuente de alimentación.

3.1.2. Selección del sensor

Dado que el actuador seleccionado no necesita de encoder para medir la velocidad o la posición, el único sensor necesario es el que nos proporcione la información de inclinación del cuerpo del robot. Las alternativas para medir la inclinación del robot son varias.

La primera opción son los inclinómetros o sensores de inclinación. Estos sensores se basan en la variación de la fuerza gravitacional para medir la inclinación. La desventaja de estos sensores es que son poco fiables cuando existe mucho movimiento que puede afectar en la medición de la aceleración de la gravedad.

Ilustración 13: Sensores de inclinación [26]

La segunda opción para valorar son los sensores giroscópicos. Estos sensores pueden medir la velocidad angular. La desventaja de estos sensores es que necesitan una referencia inicial ya que solo miden la variación del ángulo y no tienen forma de medir la inclinación absoluta del robot.

Ilustración 14: Sensor giroscópico [24]

Por último, se puede medir la inclinación de un robot con respecto al suelo con sensores infrarrojos. Una dupla de sensores infrarrojos apuntando al suelo pueden medir la orientación de la base del robot con respecto a este. La gran desventaja de este sistema es que solo es válido para suelos completamente lisos y que un bache u objeto extraño pueden interferir gravemente en la medición.

Ilustración 15: Sensor de distancia infrarrojo [23]

Como alternativa a estos ejemplos, existen sensores que integran giróscopo y acelerómetro para obtener una medición de la inclinación absoluta del robot de mayor precisión.

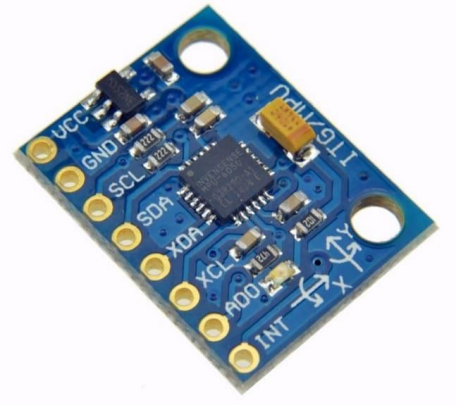


Ilustración 16: Sensor giroscópico y acelerómetro [25]

Finalmente, se ha seleccionado esta última categoría de sensores por su utilidad para la aplicación de un robot móvil y por su bajo coste. El sensor seleccionado es un módulo MPU6050 que integra sensor giroscópico y acelerómetro para los tres ejes X,Y,Z.

El sensor trabaja con comunicación serial I2C.

El giróscopo tiene una resolución de 0.61º/s, una sensibilidad de ±250º/s y un error de ±3%. El acelerómetro tiene una resolución de 0.0005g, una sensibilidad de ±2g y un error de ±50mg.

3.1.3. Selección del controlador

Para la aplicación del robot de auto balanceo no es necesario un controlador especialmente potente ya que la computación de un controlador PID no es exigente. Sin embargo, la velocidad de respuesta del controlador y la facilidad de integración de los actuadores y sensores a este son un factor limitante en las opciones a elegir.

Las dos principales alternativas de controlador para este proyecto son diferentes tipos de placas microcontroladoras o placas microprocesadoras. Las ventajas de utilizar una placa basada en microprocesador frente a una placa basada en microcontrolador es que las primeras tienen más potencia de procesamiento ya que son, en esencia, pequeños ordenadores. La ventaja principal de usar placas microcontroladoras es que estas necesitan de menor potencia y son generalmente más baratas.

Las principales gamas de placas microcontroladoras se basan en el hardware libre Arduino (ver ilustración 17). Estas placas tienen una frecuencia de 16MHz, dependiendo del modelo y se basan en una arquitectura de 8 bits [14]. Una gran ventaja de utilizar estas placas es que existe un gran repositorio de código en el que basarse a la hora de diseñar sobre ellas. La programación de estas placas se hace mediante programa C/C++.

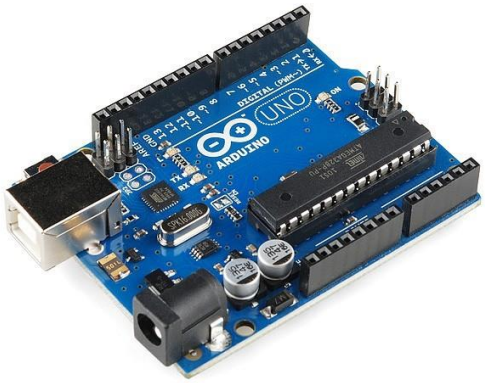


Ilustración 17: Placa microcontroladora Arduino Uno [14]

La principal marca de placas basadas en microprocesadores del mercado es Raspberry (ver ilustración 18). Las placas de Raspberry tienen una frecuencia de 1.5GHz y están basadas en una arquitectura de 64 bits [15]. Una gran ventaja de estas placas frente Arduino es la capacidad de procesamiento que tienen para aplicaciones más complejas ya que se basan en sistemas Linux y pueden correr programas más complejos.

Ilustración 18: Ordenador de una sola placa Raspberry Pi 4 [15]

Para este proyecto se ha utilizado una placa Raspberry Pi 3B+ con el fin de que el desarrollo de este proyecto sirva de base para un proyecto más ambicioso, que requiera más capacidad de computación e integración con otras tecnologías.

Durante el desarrollo del proyecto también se ha utilizado una placa Arduino Uno para probar código abierto disponible en internet sobre el hardware del robot.

3.1.4. Selección de la fuente de alimentación

Como se ha indicado en anteriores apartados, la fuente de alimentación debe cumplir ciertos requisitos que permitan el correcto funcionamiento del robot.

En primer lugar, la fuente de alimentación ha de proporcionar suficiente voltaje para todos los componentes. En este caso, los motores paso a paso requieren como mínimo 12 voltios para su correcto funcionamiento.

En segundo lugar, la fuente debe proporcionar suficiente corriente para alimentar todos los componentes a la vez. En este caso, los motores necesitan 1.5A como corriente de pico y la Raspberry Pi necesita 0.4A en estado de reposo y puede llegar a necesitar 2A dependiendo de los núcleos que se utilicen. En total, será necesario que la fuente de alimentación pueda proporcionar una corriente estable de alrededor de 4A.

En tercer lugar, la fuente debe tener capacidad suficiente para alimentar el robot durante un periodo de tiempo razonable para realizar las pruebas. Para este proyecto se ha diseñado la fuente para que el tiempo de trabajo del robot sea de una hora.

Finalmente, la fuente de alimentación debe ser independiente y debe ser recargable para facilitar el uso del robot durante las pruebas.

Para este proyecto se ha optado por utilizar una fuente basada en cuatro pilas Li-Po de 3.7V y 3.200mAH en serie (ver ilustración 19). La ventaja del uso de estas pilas frente a una batería es que tienen un tamaño estándar que permiten ser intercambiadas por pilas normales y que encajan en un soporte de pilas AA estándar.

Ilustración 19: Pila de Ion Litio utilizada para el proyecto

La fuente de alimentación resultante es una fuente de alrededor de 16.1V (las pilas de litio tienen un voltaje de entre 3.0V y 4.2V) y una capacidad de 3.200mAH con una corriente de descarga máxima de 5A. Estas características cumplen los requerimientos definidos al principio de este apartado.

3.1.5. Selección de componentes electrónicos adicionales

Además de los componentes mencionados en apartados anteriores, ciertos componentes adicionales son necesarios para el correcto funcionamiento del sistema.

En primer lugar, el voltaje de salida de la fuente de alimentación es de 16V, lo cual es ideal para los motores, pero demasiado para la placa Raspberry. Para solventar el problema se han utilizado dos módulos LM2596 de reducción de voltaje (ver ilustración 20). Estos módulos tienen como entrada entre 7V y 28V y como salida 5V con un máximo de 3A de salida. Por esta razón se ha utilizado un módulo para alimentar la placa Raspberry y otro módulo para alimentar los componentes periféricos como el sensor y los controladores de los motores paso a paso.

Ilustración 20: Módulo convertidor reductor LM2596 [27]

En segundo lugar, la placa Raspberry no puede controlar los motores paso a paso directamente, ya que no puede interrumpir directamente un voltaje tan alto. Para controlar estos motores se han seleccionado controladores A4988 para motores paso a paso (ver ilustración 21). Adicionalmente, se ha incluido un condensador de desacoplamiento de 330µF y 25V en la entrada de estos controladores para evitar interferencias entre los motores y el resto del circuito. La tabla de pines del controlador se puede ver en la ilustración 22.

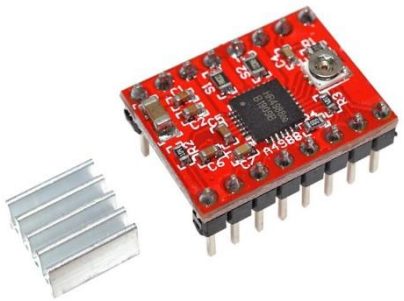


Ilustración 21: Controlador motor paso a paso A4988 [16]

Pin	Descripción
Enable	Deshabilitar el controlador
MS1, MS2, MS3	Pines de configuración de micropasos
Reset	Reiniciar el controlador
Sleep	Mantener el controlador en espera
Step	Pasos del motor
Dir	Dirección del motor
Vmot	Entrada del voltaje del motor
GND	Tierra del motor
1A, 1B, 2A, 2B	Pines de salida al motor
VDD	Entrada de alimentación
GND	Tierra

Ilustración 22: Tabla de pines del controlador A4988

3.2. Diseño del circuito

Tras la elección de los componentes descrita en el apartado anterior, el circuito final del sistema es el siguiente:

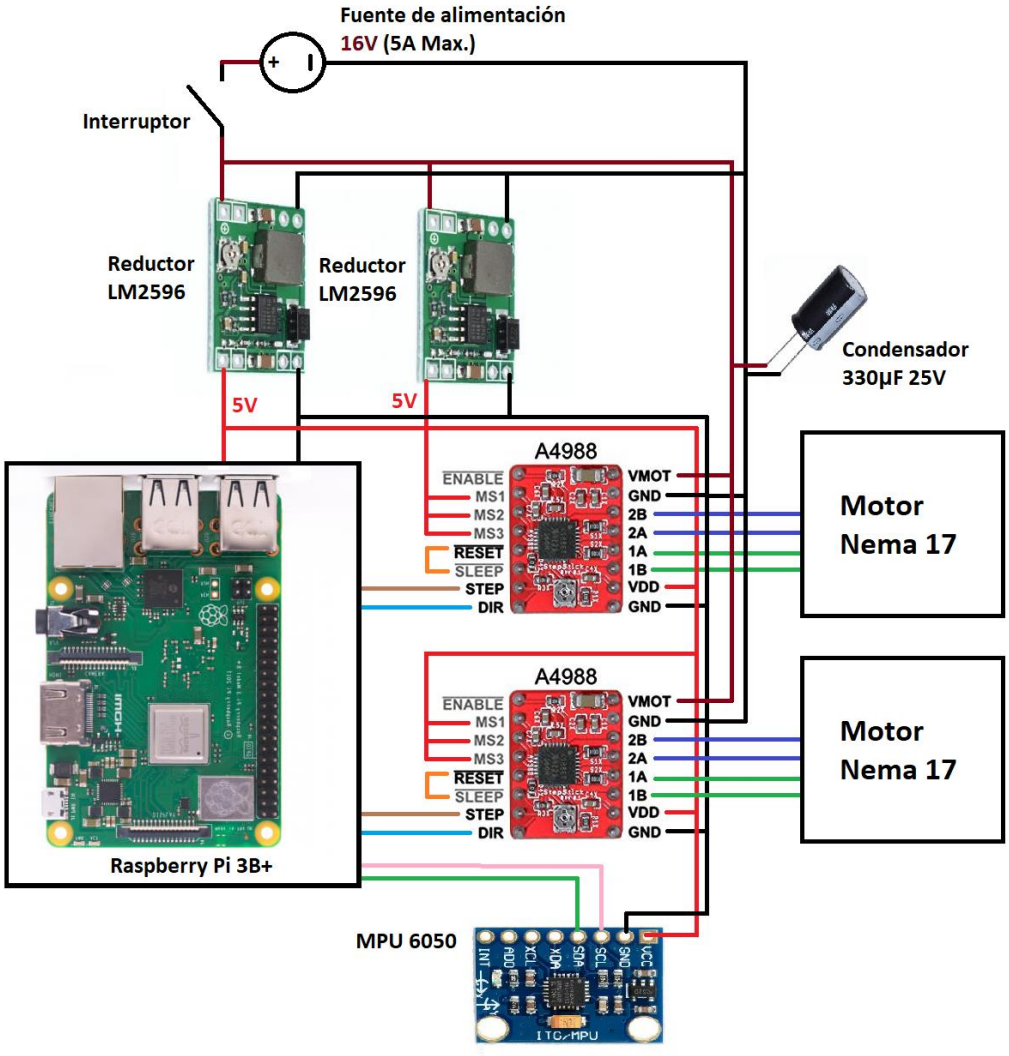


Ilustración 23: Diseño final del circuito

Las conexiones al módulo MPU 6050 se realizan a pines dedicados a conexiones I2C en la placa Raspberry Pi. El resto de las conexiones a la placa Raspberry son a pines dedicados para alimentación y salida/entrada de 3.3V [15].

Es importante destacar que los controladores A4988 deben ser calibrados previamente para suministrar la corriente adecuada a los motores [16]. Los pines MS1, MS2 y MS3 están conectados a 5V para fijar la configuración de los controladores a micro pasos de 1/16 de la distancia original de los pasos del motor. Esto permite un control más preciso de los motores y añade resolución al control del robot.

3.3. Diseño de la estructura del robot

Una vez diseñado el circuito y seleccionados los componentes necesarios, se ha procedido a diseñar una estructura para alojar dichos componentes. Se ha utilizado el programa AutoCAD para realizar el diseño y se han importado versiones en 3D de algunos de los componentes más importantes del robot obtenidos en la página TinkerCAD [17].

El diseño del robot se ha basado en minimizar el ancho del robot y maximizar el alto. De esta manera los componentes están organizados de forma vertical hasta la placa Raspberry Pi.

Adicionalmente, se han colocado los enganches de las pilas del robot en los laterales y fuera de la caja principal para tener un acceso óptimo durante la ejecución de las pruebas, ya que es necesario reemplazar las pilas frecuentemente. También se ha orientado la placa Raspberry con los puertos USB y Ethernet apuntando a una apertura en la parte superior del robot para facilitar la conexión con el robot durante su ejecución. El diseño original de la estructura del robot es el siguiente:

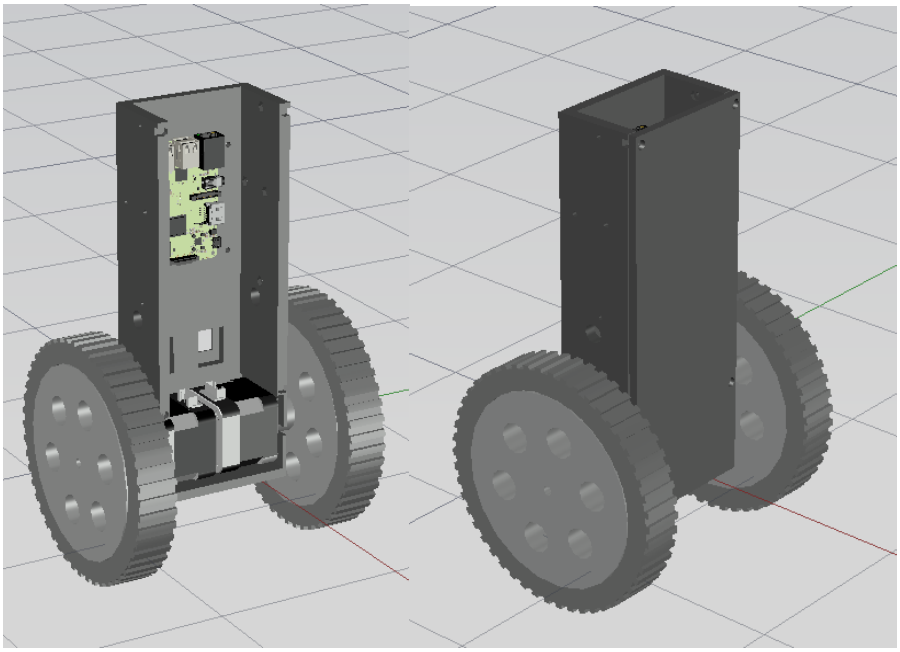


Ilustración 24: Diseño de la estructura del robot

Finalmente, todos los componentes están anclados a la estructura del robot con tornillos M2.5 y M3. La estructura del robot y la parte central de las ruedas se ha impreso en ácido poliláctico (PLA). La parte exterior de las ruedas se ha impreso en PLA flexible para facilitar la adhesión del robot con el suelo y evitar que el robot resbale.

4. Montaje y puesta a punto del robot

Para comprobar que el circuito diseñado funciona correctamente, se ha montado en placas de prototipado fuera de la estructura del robot en primer lugar. Cabe destacar que el condensador de desacoplamiento ha de estar lo más cerca posible de los controladores de motores. También se incluyen los disipadores de calor en los controladores.

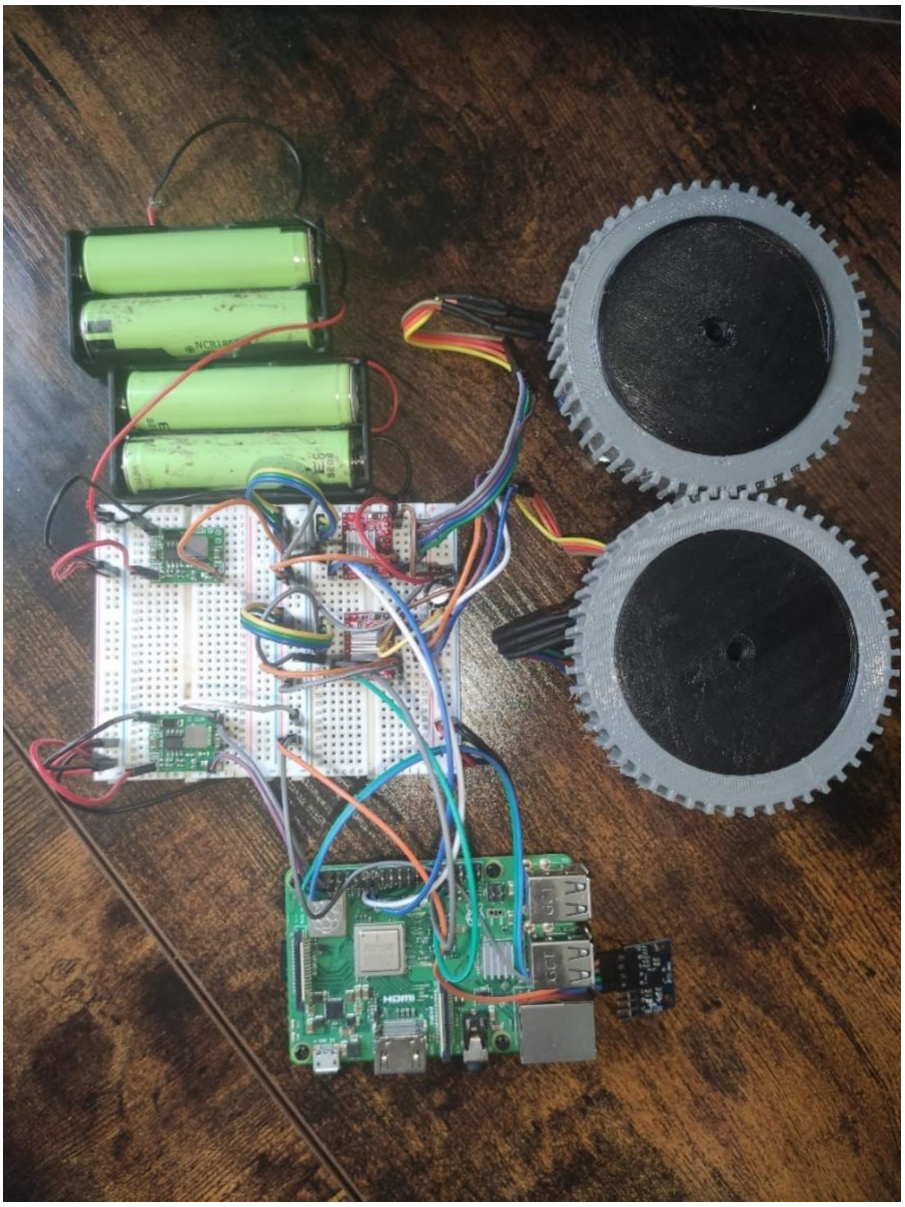


Ilustración 25: Prototipo del circuito completo montado

Una vez montado el circuito correctamente, se procede a calibrar los controladores de motores A4988. Para realizar esto se mide el voltaje entre el tornillo de calibración y el pin de tierra.

El voltaje de referencia a fijar se rige por la ecuación [16]:

$$V_{Ref} = I_{mot} * 8 * R_{Sen}$$

Donde Imot es la corriente del motor, en este caso 1.5A y Rsen es el valor de las resistencias shunt del motor. Este valor es 0.1Ω según se ve en las marcas de las resistencias. Por lo tanto, el valor del voltaje de referencia es 1.2V. Se fija este voltaje en los dos controladores de motores con el tornillo integrado.

En este momento, se ha procedido a comprobar la ratio de giro de los motores paso a paso. Para ello, se ha hecho una marca en las ruedas y se han movido los motores 10º (ver ilustración 26). La media de pasos necesarios para mover los motores han sido 286 y 289 pasos para los motores izquierdo y derecho respectivamente, realizada en 10 medidas.

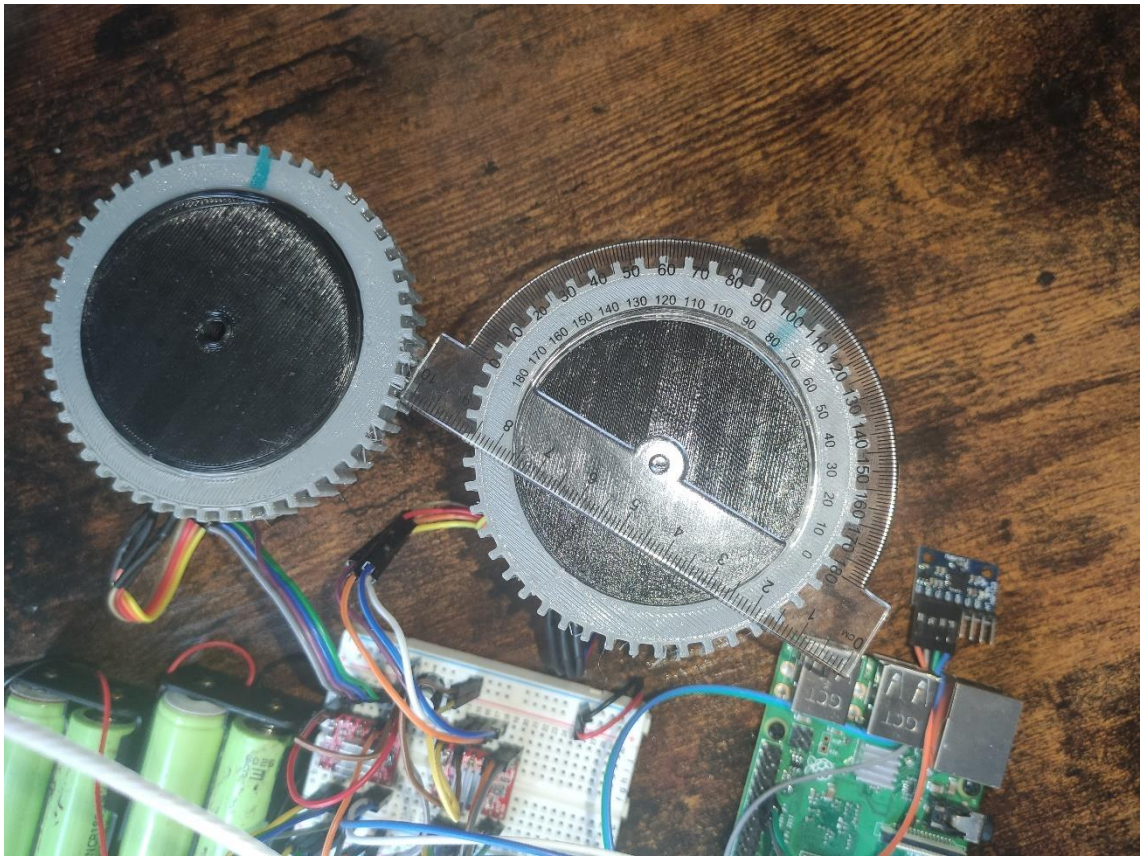


Ilustración 26: Medición de ratio grado/paso de los motores

Este valor se incluye en el código para rectificar la diferencia de pasos entre los motores y mantener la velocidad constante entre ellos.

Finalmente, tras comprobar y calibrar el circuito, se ha procedido a volver a montarlo en la estructura del robot. El diseño original de la estructura no incluye un hueco para el interruptor, por lo que este ha tenido que ser pegado en un lateral del robot.

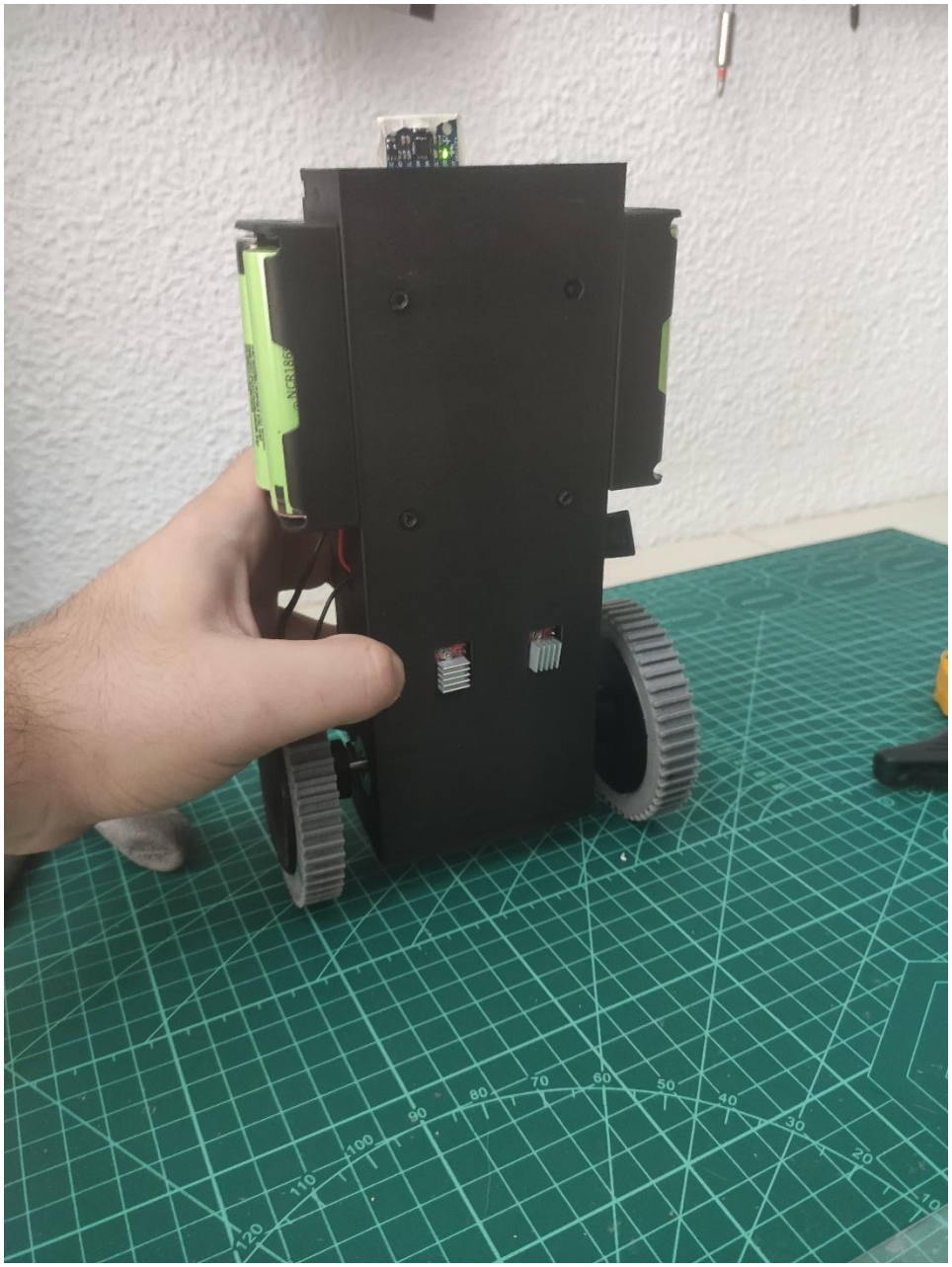


Ilustración 27: Robot montado

5. Diseño del controlador

En este apartado se obtienen las ecuaciones de movimiento del sistema para representar el robot en el espacio de estados con el fin de diseñar un controlador que pueda mantenerlo erguido en ejecución. También se analiza el diseño de un filtro de Kalman para la fusión de las medidas del sensor giroscópico y el acelerómetro.

5.1. Ecuaciones de movimiento del sistema

Como se ha explicado en los primeros capítulos, el controlador a utilizar es un controlador proporcional-integral-derivativo. Para diseñar el controlador se procede a calcular las ecuaciones de movimiento del sistema.

En primer lugar, se definen las coordenadas del robot.

z	Posición del robot en el eje Z.
z_c	Posición del centro de masas en el eje Z.
x_c	Posición del centro de masas en el eje X.
θ	Ángulo del robot con respecto al eje X.
φ	Ángulo de la rueda con respecto al momento inicial.
L	Distancia entre el eje de las ruedas y el centro de masas.
m	Masa del robot.
m_r	Masa de la rueda.
R	Radio de la rueda.
τ_0	Par aplicado.

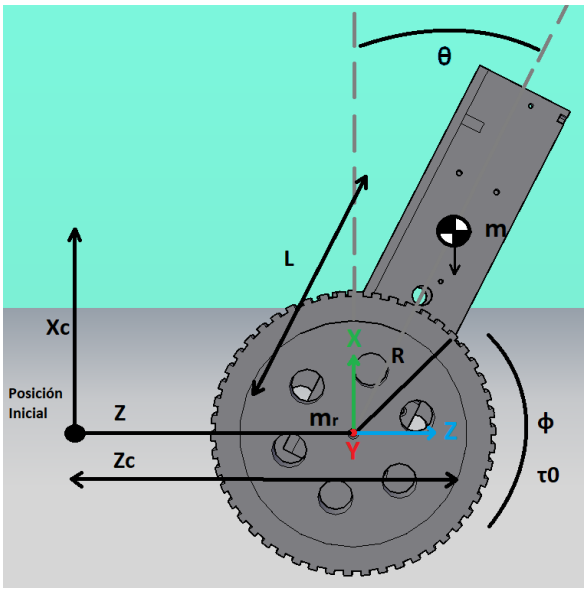


Ilustración 28: Sistema de coordenadas del robot

Se pueden derivar las ecuaciones de las variables del robot como:

$$z = R\varphi \quad (1) \qquad \dot{z} = R\dot{\varphi} + \dot{\theta}Lcos\theta \quad (4)$$

$$z_c = R\varphi + Lsin\theta \quad (2) \qquad \dot{z}_c = R\dot{\varphi} + \dot{\theta}Lcos\theta \quad (5)$$

$$x_c = R + Lcos\theta \quad (3) \qquad \dot{x}_c = -L\dot{\theta}sin\theta \quad (6)$$

Se pueden definir las ecuaciones de energía potencial y energía cinética del robot como:

$$E_p = mg(R + Lcos\theta) - mg(R + L) = mgLcos(\theta - 1) \quad (7)$$

$$E_k = \frac{1}{2}m_r\dot{z}^2 + \frac{1}{2}I_r\dot{\varphi}^2 + \frac{1}{2}m\dot{z_c}^2 + \frac{1}{2}m\dot{x_c}^2 + \frac{1}{2}I\dot{\theta}^2 \quad (8)$$

Donde I_r es la inercia de la rueda e I es la inercia del robot.

Reemplazando las ecuaciones anteriores (4,5,6) en la ecuación de energía cinética (8):

$$E_k = \frac{1}{2}m_r(R\dot{\varphi} + \dot{\theta}Lcos\theta)^2 + \frac{1}{2}I_r\dot{\varphi}^2 + \frac{1}{2}m(R\dot{\varphi} + \dot{\theta}Lcos\theta)^2 + \frac{1}{2}m(-L\dot{\theta}sin\theta)^2 + \frac{1}{2}I\dot{\theta}^2$$

$$E_k = \frac{1}{2}(I_r + m_rR^2 + mR^2)\dot{\varphi}^2 + mRL\dot{\varphi}\dot{\theta}cos\theta + \frac{1}{2}(I + mL^2)\,\dot{\theta}^2 \quad (9)$$

El lagrangiano de este sistema se puede definir como la resta de la energía cinética y la energía potencial.

$$\mathcal{L} = E_k - E_p$$

$$\mathcal{L} = \frac{1}{2}(I_r + m_rR^2 + mR^2)\dot{\varphi}^2 + mRL\dot{\varphi}\dot{\theta}cos\theta + \frac{1}{2}(I + mL^2)\dot{\theta}^2 - mgLcos(\theta - 1) \quad (10)$$

Se deriva el lagrangiano para obtener las fórmulas de Euler-Lagrange según las variables φ y θ.

Variable φ:

$$\frac{\partial\mathcal{L}}{\partial\dot{\varphi}} = (I_r + m_rR^2 + mR^2)\dot{\varphi} + mRL\dot{\theta}cos\theta \quad (11)$$

$$\frac{\partial\mathcal{L}}{\partial\varphi} = 0 \quad (12)$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\varphi}}\right) - \frac{\partial\mathcal{L}}{\partial\varphi} = (I_r + m_rR^2 + mR^2)\ddot{\varphi} + mRL\ddot{\theta}cos\theta - mRL\dot{\theta}^2sin\theta = T \quad (13)$$

Variable θ:

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}} = mRL\dot{\varphi}cos\theta + (I + mL^2)\dot{\theta} \quad (14)$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = mgLsin\theta - mRL\dot{\varphi}\dot{\theta}sin\theta \quad (15)$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = (I + mL^2)\ddot{\theta} + mRL\ddot{\varphi}cos\theta - mgLsin\theta = -T \quad (16)$$

La variable T es el par que ejercen los motores.

Finalmente, se linealizan las ecuaciones del sistema en torno al punto de equilibrio ($\varphi = 0, \dot{\varphi} = 0, \theta = 0, \dot{\theta} = 0, sin\theta = \theta, cos\theta = 1$):

$$(I_r + m_r R^2 + mR^2)\ddot{\varphi} + mRL\ddot{\theta} = T \quad (17)$$

$$(I + mL^2)\ddot{\theta} + mRL\ddot{\varphi} - mgL\theta = -T \quad (18)$$

De este sistema de ecuaciones se puede obtener una sola ecuación que relacione el ángulo de inclinación del robot con el par aplicado a las ruedas:

$$(I + mL^2)\ddot{\theta} + mRL\left(\frac{T - mRL\ddot{\theta}}{(I_r + m_r R^2 + mR^2)}\right) - mgL\theta = -T \quad (19)$$

5.2. Representación del sistema en el espacio de estados

Una vez se han obtenido las ecuaciones del sistema, se procede a representar el sistema en el espacio de estados para facilitar el análisis y control.

Para ello, se seleccionan las variables de estado:

$$\dot{q} = Aq + Bu$$

$$y = Cq + Du$$

$$q = \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \end{bmatrix}$$

Finalmente, el sistema es el siguiente:

$$\begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & a_{22} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} T \tag{20}$$

$$[\theta] = [0 \quad 1] \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} \tag{21}$$

Donde:

$$a_{22} = \frac{mgL}{I + mL^2 - \frac{mRL}{(I_r + m_r R^2 + mR^2)}} \tag{22}$$

$$b_1 = \frac{-(I_r + m_r R^2 + mR^2) - mRL}{(I + mL^2)(I_r + m_r R^2 + mR^2) - mRL} \tag{23}$$

A continuación, se sustituyen los parámetros del robot para poder estudiar la controlabilidad:

$$m = 0.836Kg$$

$$m_r = 0.060Kg$$

$$R = 0.04m$$

$$L = 0.05m$$

$$I_r = \frac{1}{2} m_r R^2 = 4.8 \cdot 10^{-5}\ Kgm^2$$

Para calcular la inercia del cuerpo del robot, se tiene en cuenta que la gran parte de la masa del robot se divide entre las pilas y los motores. Los motores se pueden identificar

como una masa rectangular m_m de lado d_m y longitud rotando sobre su eje y las pilas se pueden identificar como una masa rectangular m_p de lados d_a y d_b rotando a un eje a una distancia l_p.

Por lo tanto, el momento de inercia total del robot viene dado por la fórmula:

$$I = \frac{1}{3}\frac{m_p}{d_a + d_b}{l_p}^3 + \frac{1}{12}m_m{d_m}^2 = \frac{1}{3}\cdot\frac{0.364}{0.07 + 0.03}0.13^3 + \frac{1}{12}0.444 \cdot 0.02^2 = 0.0027Kgm^2$$

Reemplazando estos valores en la representación del espacio de estados, obtenemos las siguientes matrices:

$$\begin{bmatrix}\ddot{\theta}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & -0.36\\ 1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ \theta\end{bmatrix} + \begin{bmatrix}1.89\\ 0\end{bmatrix}T \quad (24)$$

$$[\theta] = [0 \quad 1]\begin{bmatrix}\dot{\theta}\\ \theta\end{bmatrix} \quad (25)$$

A continuación, se estudia la controlabilidad de este sistema:

$$C = [B \quad AB]$$

$$C = \begin{bmatrix}1.89 & 0\\ 0 & 1.89\end{bmatrix}$$

$$Rango(C) = 2$$

Dado que el rango de la matriz de controlabilidad es igual a n, se considera que la salida del sistema θ es completamente controlable en base a la entrada del sistema T.

Finalmente, se estudia la observabilidad del sistema.

$$O = \begin{bmatrix}C\\ CA\end{bmatrix}$$

$$C = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$$Rango(O) = 2$$

Dado que el rango de la matriz de observabilidad es igual que n, se considera que el estado interno del sistema es completamente observable en base a la salida de este.

5.3. Diseño del controlador proporcional-integral-derivativo

Para diseñar el controlador PID, se transforma el sistema representado en el espacio de estados (24,25) en su correspondiente función de transferencia:

$$Tf = \frac{1.89}{s^2 + 0.36}$$

El sistema consta de dos polos en el eje real, lo que quiere decir que el sistema es marginalmente estable y tenderá a oscilar:

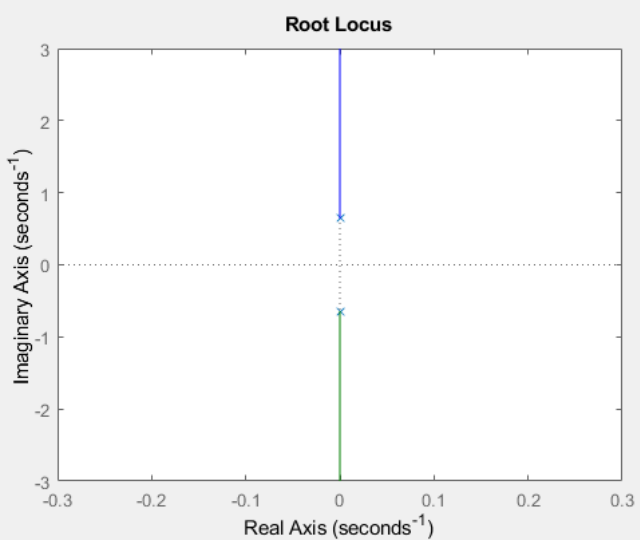


Ilustración 29: Lugar de raíces del sistema

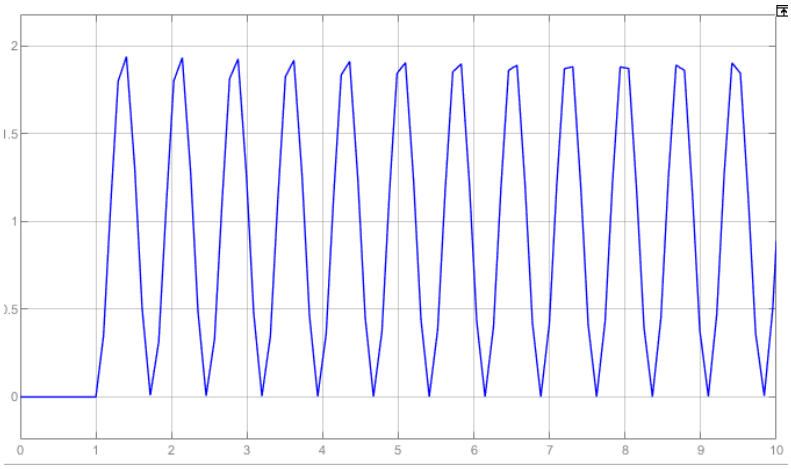


Ilustración 30: Respuesta a escalón unitario del sistema en bucle cerrado

En base a este lugar de raíces, podemos analizar cómo, añadiendo un componente proporcional al sistema este puede llegar a un estado estable.

Añadiendo un componente derivativo al sistema obtenemos un controlador PD. El cero que añade el componente derivativo crea un lugar de raíces en el plano negativo del eje real, lo que corresponde a un sistema estable.

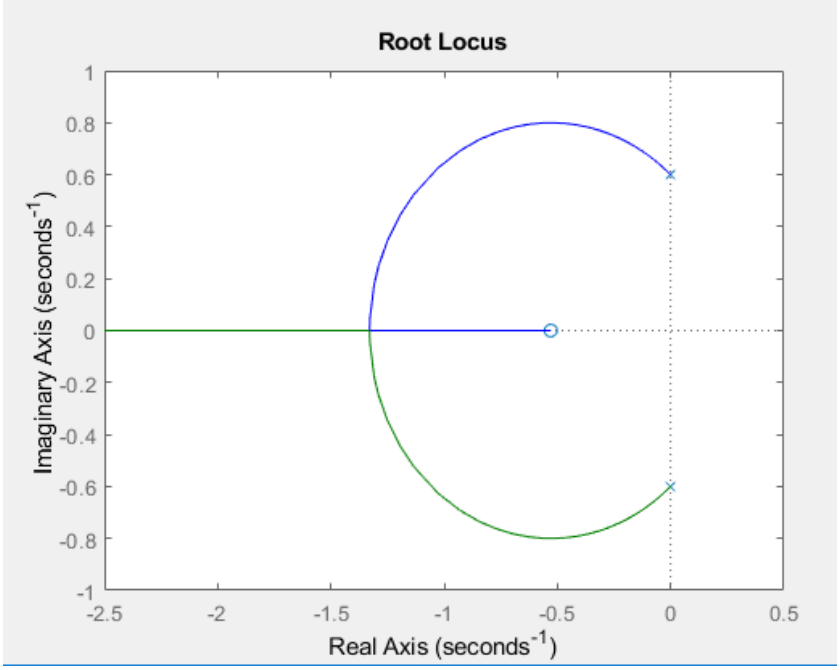


Ilustración 31: Lugar de raíces del sistema con controlador PD

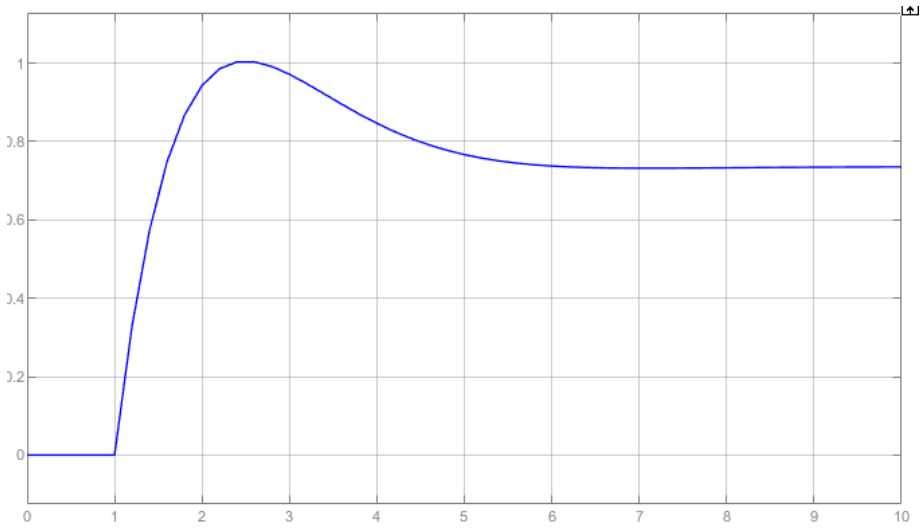


Ilustración 32: Respuesta a escalón unitario del sistema con controlador PD sin calibrar

Aunque este controlador pueda ser suficiente para controlar el sistema, se añade un componente integral para asegurar una mejor calibración de la respuesta del sistema.

Al añadir un componente integral al controlador, el lugar de raíces sigue estando siempre en el plano negativo del eje real:

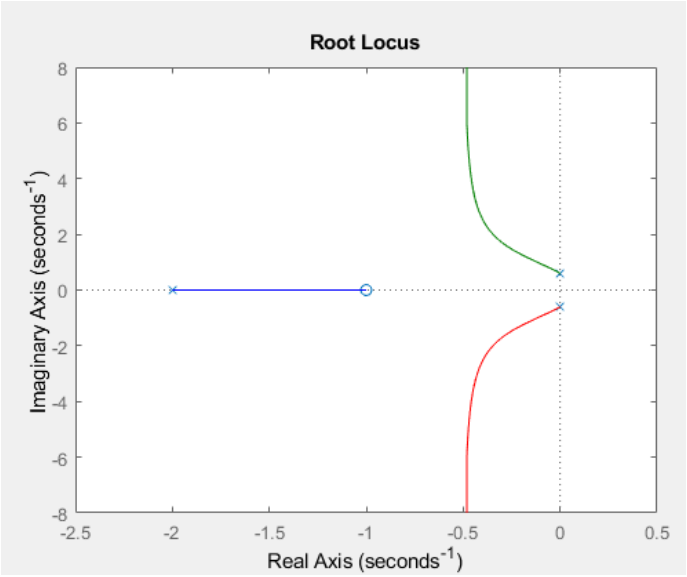


Ilustración 33: Lugar de raíces del sistema con controlador PID

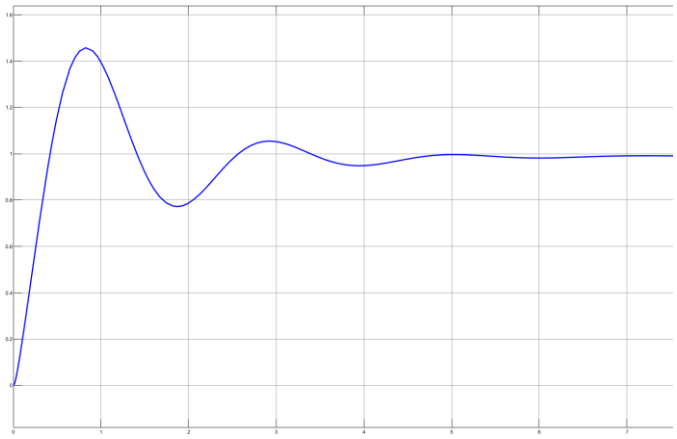


Ilustración 34: Respuesta a escalón unitario del sistema con controlador PID sin calibrar

En base a esta función de transferencia, se crea un bucle con controlador PID en Matlab para simular el sistema y su controlador:

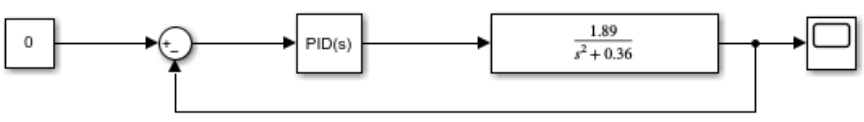


Ilustración 35: Sistema con controlador PID en Simulink

Al calibrar este controlador, se pueden obtener directamente los valores de las ganancias P, I y D. Se han calibrado los parámetros para aumentar el tiempo de respuesta, reducir oscilaciones y reducir el pico de la señal:

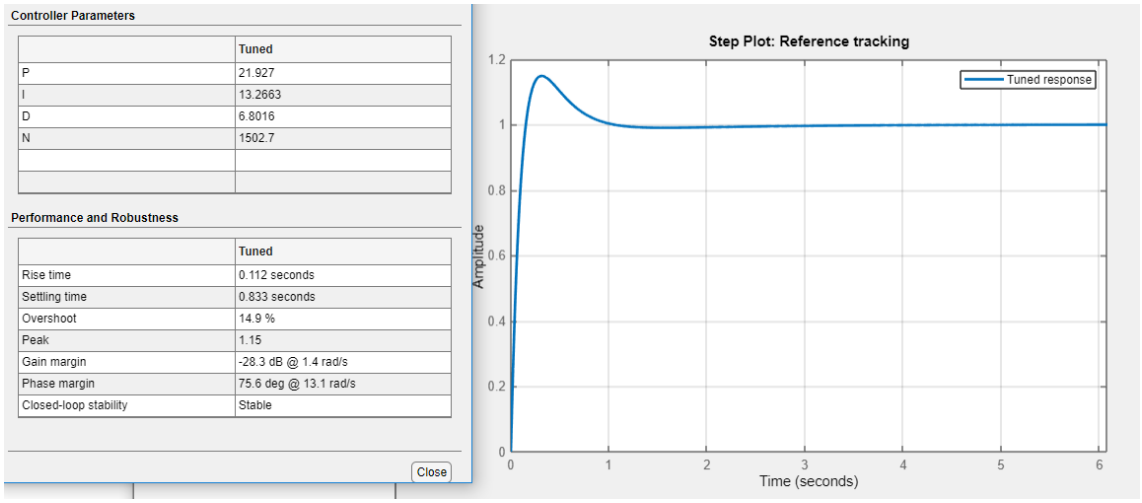


Ilustración 36: Simulación de la respuesta a escalón del sistema con PID calculado

Estos parámetros se han modificado más adelante durante las pruebas realizadas en el robot.

5.4. Filtro de Kalman

Existe un problema inherente al uso de sensores en los sistemas y este es el error que generan los sensores en el sistema. Como referencia, se ha medido el error existente al medir el ángulo de inclinación del robot en posición vertical solo utilizando la información del giróscopo:

Ilustración 37: Error del ángulo del giróscopo (radianes)

También se puede ver el error del acelerómetro:

Ilustración 38: Error del ángulo del acelerómetro (grados)

Se puede ver que hay dos fuentes principales de error en el sensor. La fuente principal es el error general que se puede caracterizar como ruido blanco.

La segunda fuente son los picos que se pueden ver en las dos gráficas de error. Estos picos corresponden a momentos en los que la placa pierde conexión con el sensor y durante este momento no existe lectura válida. Este segundo error se puede solventar ignorando las lecturas del sensor cuando este no tiene conexión.

Simulando este error en Matlab, se puede apreciar como el error puede crear un problema en el control del sistema:

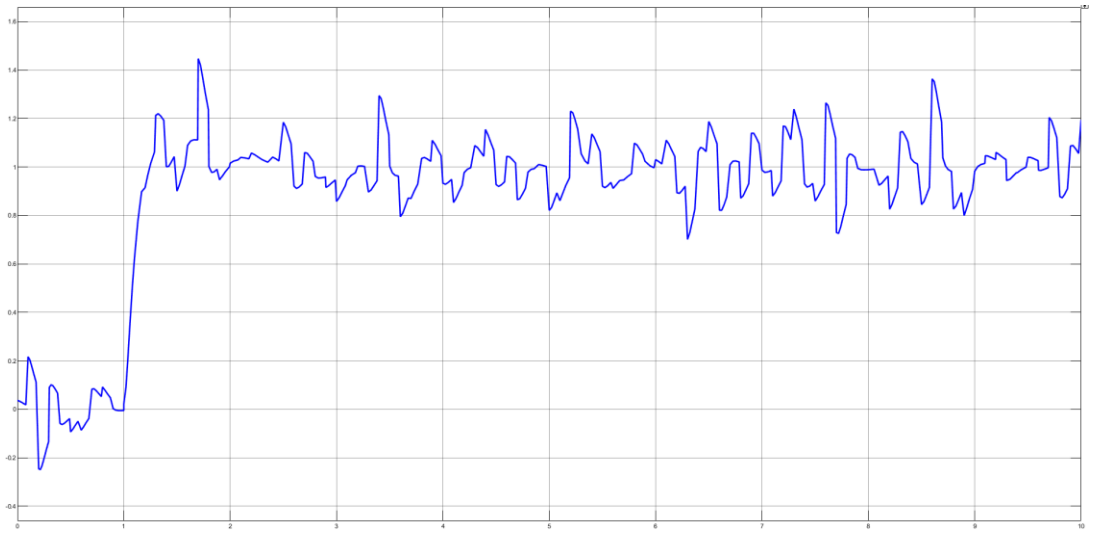


Ilustración 39: Respuesta a escalón del sistema con error gaussiano de potencia 0.001

Para solucionar este problema, se pretende utilizar un filtro de Kalman que rectifique el error e integre las dos medidas del sensor giroscópico y el acelerómetro.

Se ha de caracterizar el error de los sensores para poder calibrar correctamente el filtro de Kalman. Para esto se han quitado las medidas en las que el sensor ha perdido conexión y se han obtenido los histogramas de los errores de medida del acelerómetro y el giróscopo:

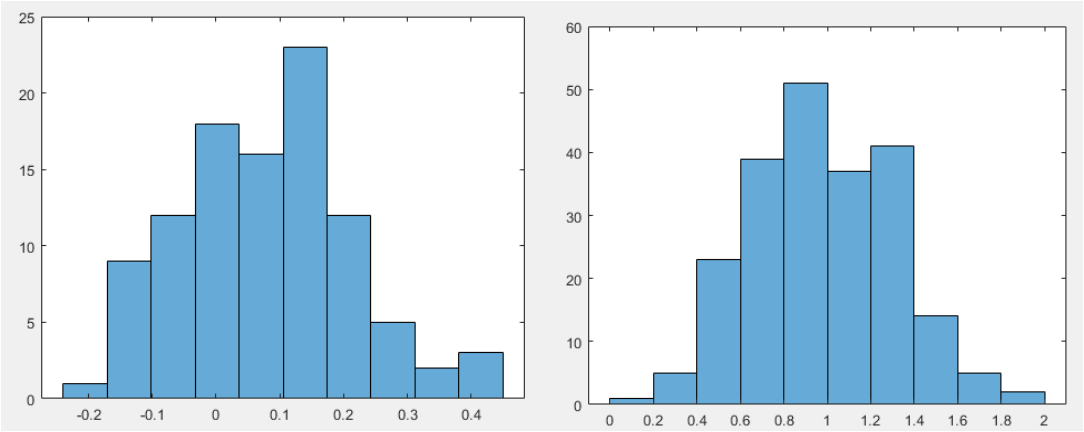


Ilustración 40: Histograma de errores de medida del sensor giroscópico (izquierda) y el acelerómetro (derecha) en radianes.

Se han ajustado los histogramas anteriores a una distribución gaussiana obteniendo los siguientes resultados:

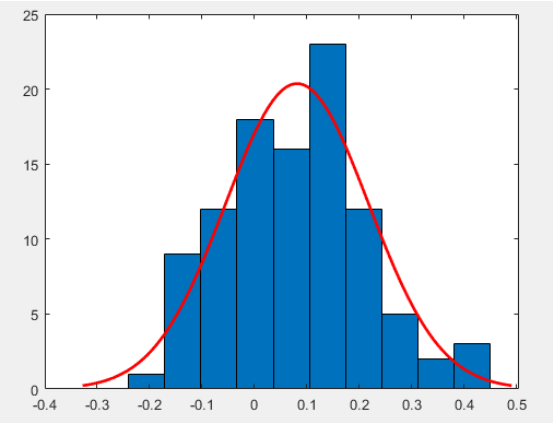


Ilustración 41: Histograma de errores de medida del giróscopo ajustado (radianes)

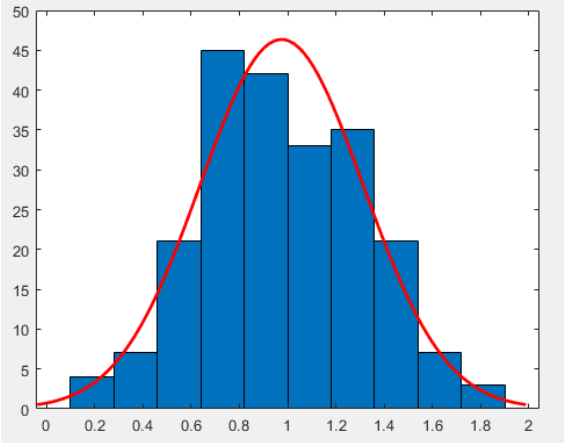


Ilustración 42: Histograma de errores de medida de acelerómetro ajustado (radianes)

Los valores del ajuste de error del giróscopo son:

$$\nu = 0.0816$$

$$\sigma^2 = 0.0186$$

Los valores del ajuste de error del acelerómetro son:

$$\nu = 0.9748$$

$$\sigma^2 = 0.1140$$

Se diseña un filtro de Kalman como el descrito en el apartado 2.4.1. en el que se estima el valor de las variables en el momento k como:

$$x_k = F_x x_{k-1} + B_k u_k + w_k$$

Donde w_k es el error del proceso:

$$w_k \sim N(0, Q)$$

Donde:

$$Q = \begin{bmatrix} Q_\alpha & 0 \\ 0 & Q_{bias} \end{bmatrix} = \begin{bmatrix} 0.114 & 0 \\ 0 & 0.0007 \end{bmatrix}$$

$$R = 0.0186$$

Aplicando este filtro, se puede apreciar como la medida integrada de los dos sensores mejora frente a las medidas de cada sensor o una fusión de sensores simple basada en la media:

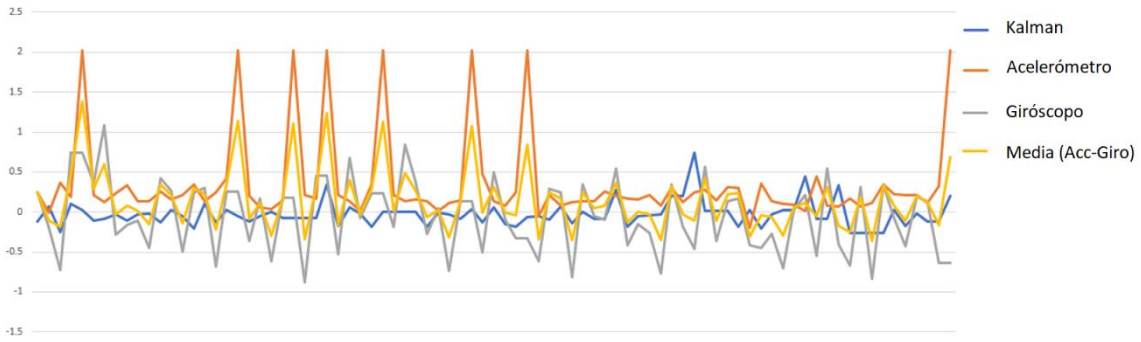


Ilustración 43: Medición de inclinación del robot con filtro de Kalman (radianes)

6. Diseño del software

Como se ha comentado al principio, la placa utilizada es una placa Raspberry Pi 3B+. El sistema operativo instalado en la placa es una distribución de Raspbian simple y limpia. Con el fin de facilitar el desarrollo del software del robot, se pretende utilizar librerías existentes.

El código se ha escrito en Python y se han utilizado las librerías “I2C”, “GPIO”, “Multiprocessing” y “time”.

El código tiene dos hilos de ejecución en paralelo y el flujo del software es el siguiente:

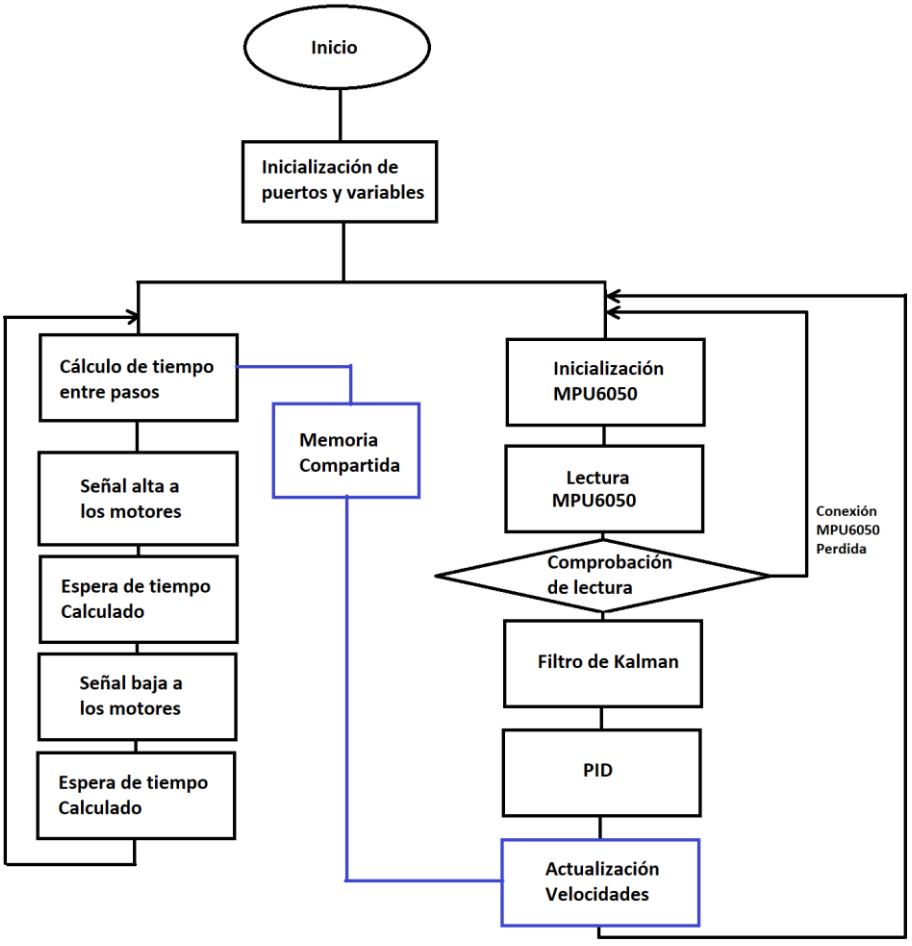


Ilustración 44: Flujo del programa del robot

Los dos hilos tienen dos variables de memoria compartida correspondiente a la velocidad de los motores.

Los valores del filtro PID serán recalibrados durante las pruebas.

7. Implementación y pruebas

Tras el montaje y programación del robot se han realizado varias pruebas. En primer lugar, se ejecutó el código con los valores del controlador PID calculados:

Kp = 22

Kd = 6.8

Ki = 13

El resultado de esta prueba fue que el robot intentaba mantenerse erguido, sin embargo, las oscilaciones producían que el robot terminase cayendo:

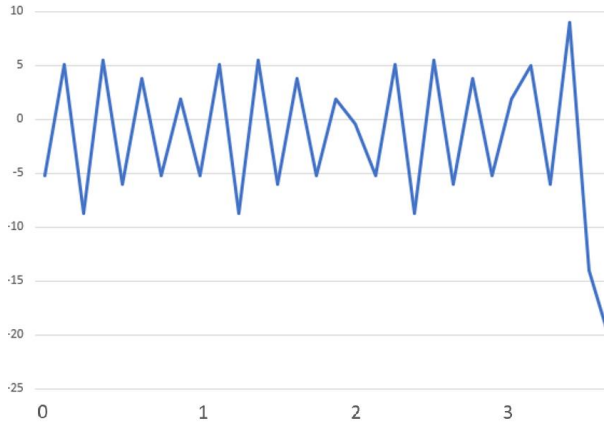


Ilustración 45: Ángulo del robot con controlador PID calculado (22, 6.8, 13) en grados

Para solventar el problema se modificaron los parámetros del controlador PID para intentar obtener unos valores que permitiesen que el robot no se cayera.

El método de utilizado para modificar los parámetros fue reducir los tres parámetros para obtener una respuesta del controlador más lenta, que produjera menos oscilaciones, al no sobrepasar tanto el valor de referencia. Los parámetros obtenidos fueron 10.2, 2.3 y 5 para los componentes proporcional, integral y derivativo respectivamente. Tras varias modificaciones se consiguió que el robot se mantuviese prácticamente estático durante un breve periodo de tiempo previo a caerse:

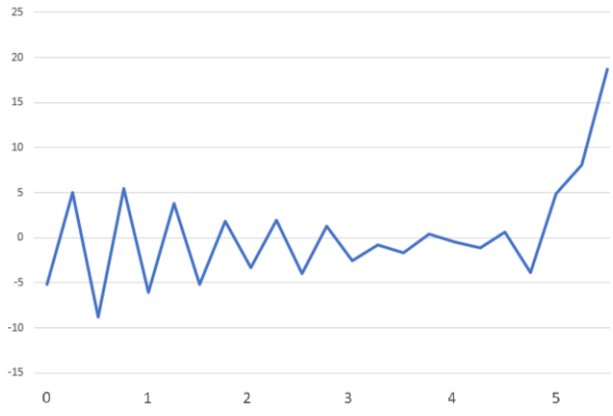


Ilustración 46: Ángulo PID (10.2, 2.3, 5) en grados

Tras varias pruebas se ha concluido que el código controlador de los motores generaba un problema durante la ejecución. Al recibir velocidades pequeñas de la salida del controlador, los tiempos de espera entre pasos aumentaban. Esto produce que durante tiempos demasiado prolongados la velocidad de las ruedas no se modificara, impidiendo el correcto control del robot.

Se modificó el código para realizar interrupciones en el tiempo de espera en caso de ser necesario. El tiempo máximo del bucle se fijó en 0.001s. Si el tiempo entre pasos calculado es mayor que el tiempo de espera máximo, no se realizarán pasos durante los bucles que sean necesarios hasta llegar al tiempo de espera calculado. Si durante este tiempo de espera se calcula un tiempo entre pasos menor, este se ejecutará interrumpiendo el tiempo de espera anterior.

Adicionalmente, se sincronizaron los pasos del motor para que correspondiesen a múltiplos de un periodo fijo, lo que evita problemas de vibración en los motores al resonar a ciertas frecuencias.

Tras estas modificaciones se procedió a volver a calibrar el robot. Sin embargo, durante una de las pruebas de calibración, la estructura impresa del robot se partió, quedando el robot inutilizado al no poder sujetar los motores:

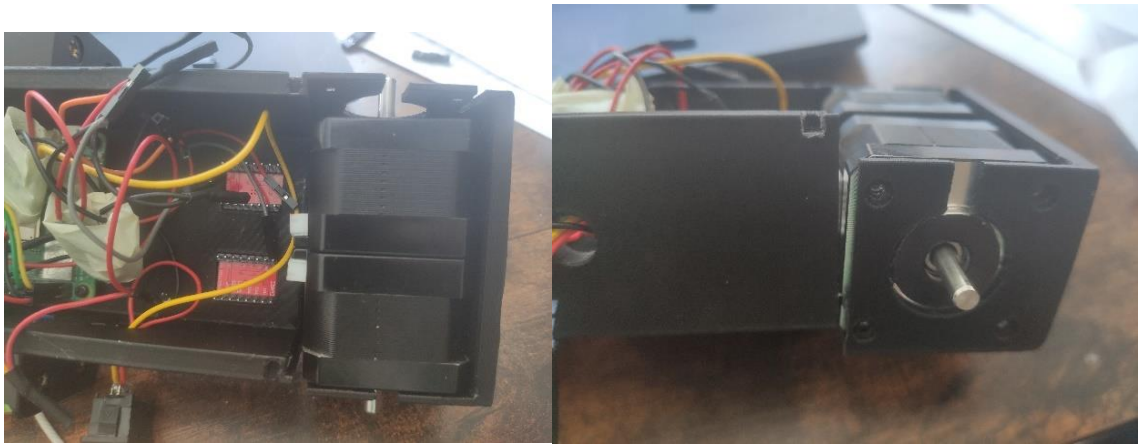


Ilustración 47: Rotura de la estructura del robot

Para evitar roturas parecidas en el futuro y mejorar la fijación de los motores y otros componentes a la estructura, se han realizado cambios a esta, creando una caja más ajustada para los motores y dejando más holgura en los huecos para las tuercas de los tornillos.

Por motivos de tiempo y problemas con la impresora 3D utilizada hasta el momento, se ha decidido continuar el desarrollo del trabajo con un robot comprado con características similares.

El robot elegido es el robot Tumblller de Elegoo [7]. Este robot viene en un kit fácil de montar con una PCB con los conectores para los componentes integrados, lo que evita falsas conexiones y desconexiones de componentes durante las pruebas, problema que había ocurrido con el anterior robot.

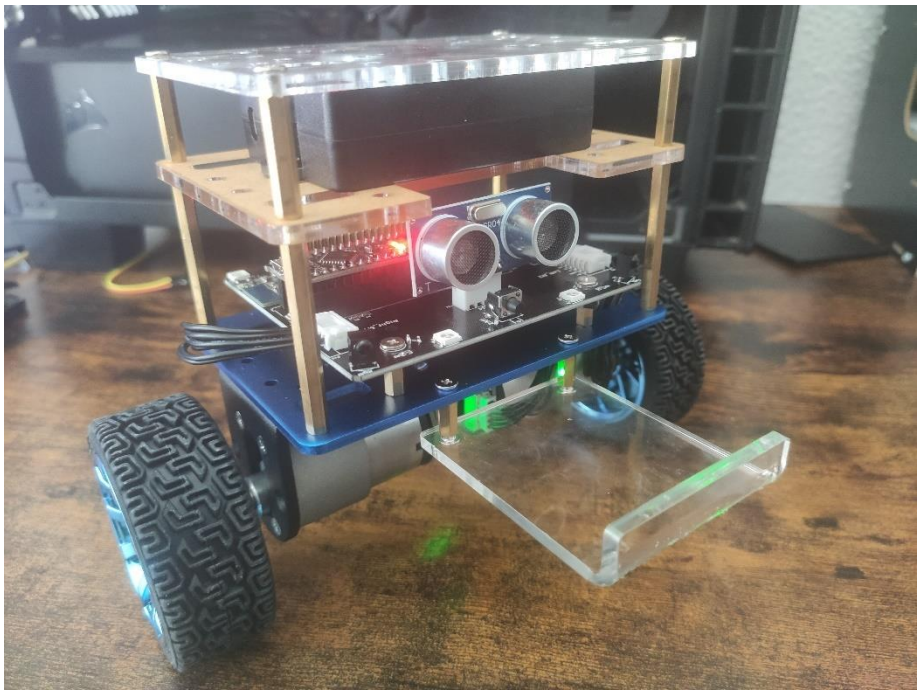


Ilustración 48: Robot Tumblller utilizado para continuar el proyecto

Este robot se basa en una placa Arduino Uno y utiliza motores DC con encoders integrados. Como sensor, el robot utiliza un sensor MPU6050 como se ha usado hasta ahora. El robot también utiliza un controlador de motores DC TB6612FNG, mediante el cual se pueden controlar los motores con señales PWM.

Durante el montaje se ha utilizado el sensor MPU6050 del robot anterior, ya que la caracterización del error del sensor y su rectificación con el filtro de Kalman han sido satisfactorias.

El código original del robot Tumblller contiene las librerías necesarias para utilizar el sensor MPU6050. Se ha trasladado el código Python para el filtro de Kalman desarrollado anteriormente a código C++. Se ha modificado el código para funcionar en un solo hilo de ejecución.

Los sensores adicionales del robot no se han utilizado durante la ejecución.

7.1. Representación del robot Tumblller en el espacio de estados

Partiendo de las fórmulas (20, 21, 22, 23) de la representación del espacio de estados del robot anterior, se reemplazan los valores de los parámetros del robot a los del robot Tumblller:

$$m = 0.737Kg$$

$$m_r = 0.102Kg$$

$$R = 0.035m$$

$$L = 0.04m$$

$$I_r = \frac{1}{2} m_r R^2 = 6.2 \cdot 10^{-5}\, Kgm^2$$

Para calcular la inercia del cuerpo del robot Tumblller se realizan las mismas asunciones que en el robot anterior.

Los motores se pueden identificar como una masa rectangular m_m de lado d_m y longitud rotando sobre su eje y las pilas se pueden identificar como una masa rectangular m_p de lados d_a y d_b rotando a un eje a una distancia l_p. Por lo tanto, el momento de inercia total del robot viene dado por la fórmula:0,000834

$$I = \frac{1}{3}\frac{m_p}{d_a + d_b} {l_p}^3 + \frac{1}{12} m_m {d_m}^2 = \frac{1}{3} \cdot \frac{0.401}{0.075 + 0.02} 0.084^3 + \frac{1}{12} 0.294 \cdot 0.015^2 = 0.00084 Kgm^2$$

Reemplazando estos valores en las ecuaciones (22) y (23) se obtiene el sistema representado en el espacio de estados:

$$\begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & -0.30 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} + \begin{bmatrix} 2.05 \\ 0 \end{bmatrix} T$$

$$[\theta] = [0 \quad 1] \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix}$$

7.2. Diseño de controlador PID para robot Tumbller

Se obtiene la función de transferencia en base a la representación en el espacio de estados del robot (24, 25):

$$Tf = \frac{2.05}{s^2 + 0.3}$$

Se rediseña el controlador PID en Matlab para este nuevo sistema:

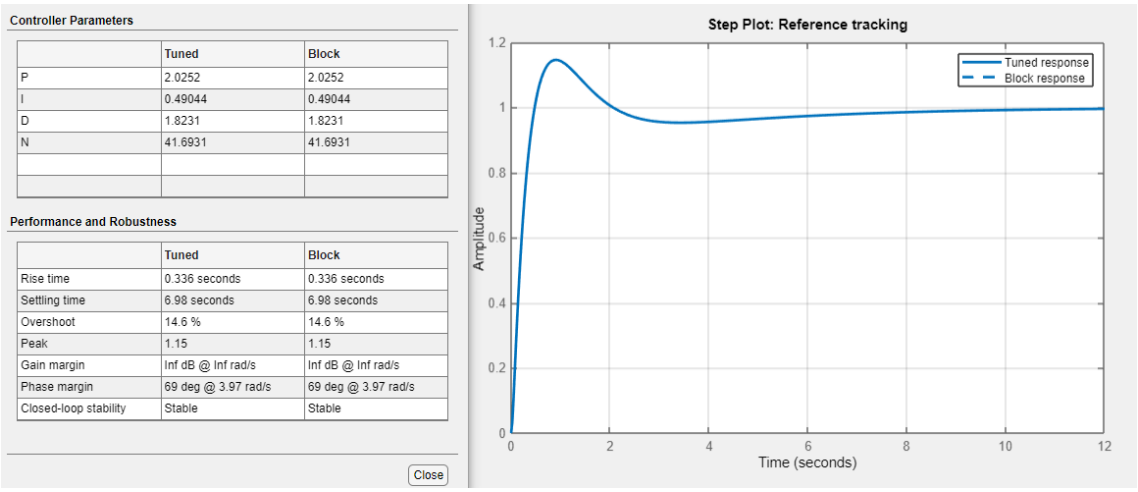


Ilustración 49: PID para el robot Tumbller

Los valores del controlador utilizados son los siguientes:

Kp = 2

Kd = 1.8

Ki = 0.5

Con estos valores del controlador, se ejecuta el código modificado en el robot Tumbller:

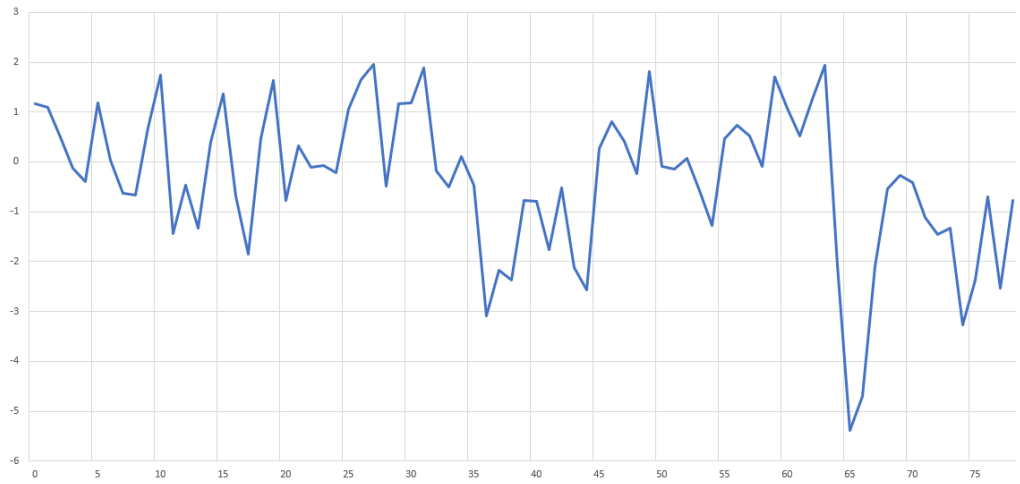


Ilustración 50: Ángulo de inclinación del robot (grados)

Tras varias modificaciones de los parámetros para corregir las oscilaciones, se redujo el parámetro Ki a 0.2, obteniendo el siguiente resultado:

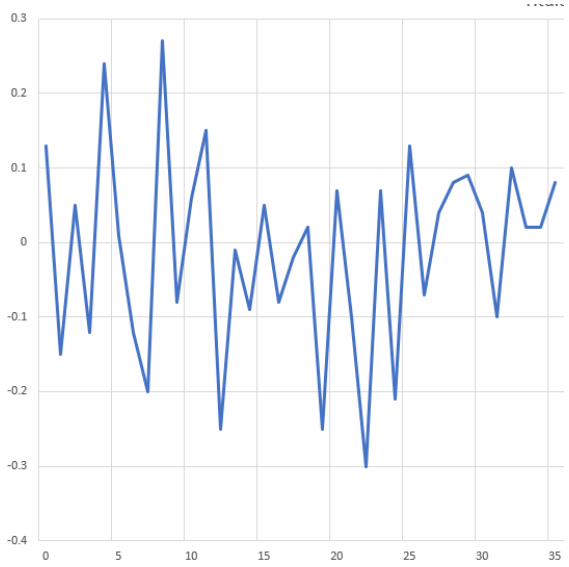


Ilustración 51: Ángulo de inclinación del robot con PID (25, 1.1, 0.2) en grados

Con estos valores del controlador, el robot es capaz de recuperarse tras una perturbación:

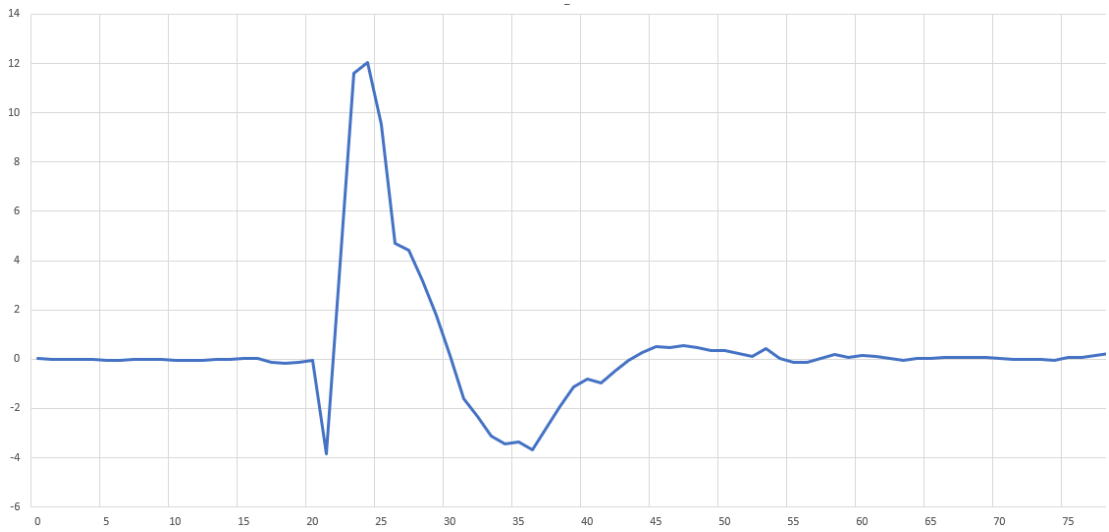


Ilustración 52: Recuperación del robot tras una perturbación (grados)

8. Conclusiones y trabajo futuro

En conclusión, el método de caracterización del sistema y representación de este en el espacio de estados para diseñar un controlador ha sido útil para el diseño del controlador PID. Sin embargo, en los dos robots se han tenido que realizar modificaciones a los controladores calculados para obtener el punto de trabajo idóneo. Se considera que la razón de la necesidad de estas modificaciones está en la falta de caracterización de las ecuaciones entre el par y la velocidad de los motores.

Aunque un controlador PD es suficiente para controlar el sistema, el controlador PID aporta rechazo a perturbaciones. El sistema con el controlador diseñado es capaz de recuperarse de empujes. Cabe destacar que el controlador solamente controla el ángulo del robot y no el ángulo de las ruedas o la posición de este, por lo cual la posición del robot varía sin ser controlada mientras el robot se estabiliza.

El filtro de Kalman ha proporcionado una medida del ángulo del robot suficientemente precisa y estable para el control del robot. Comparando las mediciones del filtro de Kalman frente a mediciones directas del giróscopo y el acelerómetro, el filtro de Kalman es un método más preciso y robusto. Las mediciones del ángulo con el filtro de Kalman y el ajuste a pérdidas de conexión con el sensor proporcionan una medición fiable y estable, que resulta finalmente en un movimiento fluido del robot.

8.1. Trabajo futuro

Es necesario volver a construir el robot diseñado con las modificaciones oportunas para continuar las pruebas y poder comparar las ventajas de la configuración con motores paso a paso frente a motores de corriente continua.

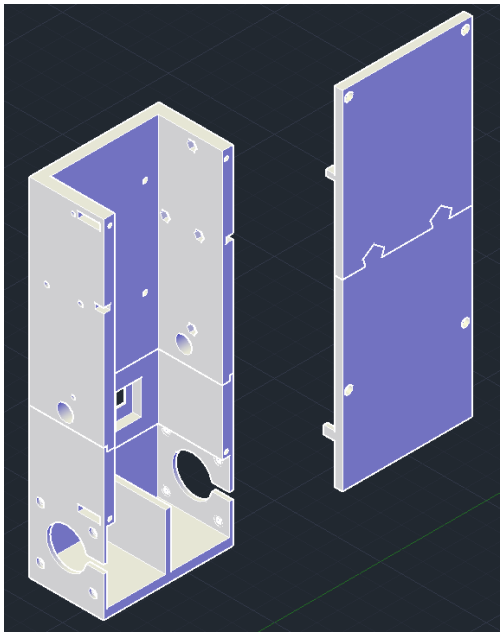


Ilustración 53: Segunda versión de la estructura del robot

Sería necesario caracterizar la ecuación entre la velocidad y el par de los motores, teniendo en consideración las pérdidas de los motores, para realizar un diseño de controlador más preciso.

Se podría modificar el controlador y la representación del sistema en el espacio de estados para introducir el control de la posición del robot y así poder introducir movimiento al robot.

Finalmente, sería interesante comparar el controlador diseñado con otros tipos de controladores como el controlador lineal cuadrático en ambos sistemas.

Referencias y bibliografía

[1] Boubaker, O. (2013). The Inverted Pendulum Benchmark in Nonlinear Control Theory: A Survey. International Journal of Advanced Robotic Systems.

https://doi.org/10.5772/55058.

[2] Ballbot. (2013). MSL.

http://www.msl.ri.cmu.edu/projects/ballbot/.

[3] Murphy, R. R. (2018). Astromech robots in Star Wars. Science Robotics, 3(15).

[4] Waibel, M. (2010, 8 mayo). Project Rezero: Ball Balancing Robot With Style. IEEE Spectrum. https://spectrum.ieee.org/project-rezero-ball-balancing-elegance.

[5] Grasser, F., D'Arrigo, A., Colombi, S., & Rufer, A.C. (2002). JOE: a mobile, inverted pendulum. IEEE Trans. Ind. Electron., 49, 107-114. https://www.semanticscholar.org/paper/JOE%3A-a-mobile%2C-inverted-pendulum-Grasser-D%27Arrigo/91c69e169d94d8a02a240359b039e6f5f7059186.

[6] Segway. (2020). Segway.

https://es-es.segway.com/.

[7] Elegoo Tumbller. (2022). Elegoo.

https://www.elegoo.com/products/elegoo-tumbller-self-balancing-robot-car.

[8] Kálmán, R.E. (1960). A new approach to linear filtering and prediction problems. transaction of the asme. Journal of basic.

https://www.semanticscholar.org/paper/A-new-approach-to-linear-filtering-and-prediction-K%C3%A1lm%C3%A1n/255a77422b1da74da05d1714b7875356187385bd

[9] Bishop, G., & Welch, G. (2001). An introduction to the kalman filter, 24.

[10] Borenstein, J., & Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots. IEEE Trans. Robotics Autom., 12, 869-880.

https://www.semanticscholar.org/paper/Measurement-and-correction-of-systematic-odometry-Borenstein-Feng/a184c71f70ba7a4aaedec1583c6861c15aaedfec

[11] Scherr, C., Lommatsch, G., Hooson, S., & Kaiser, T. (2010). Underwater Navigation System. Kalman Filter. https://ece.montana.edu/seniordesign/archive/SP14/UnderwaterNavigation/kalman_filter.html.

[12] Full State Feedback. (2022). Wikipedia. https://en.wikipedia.org/wiki/Full_state_feedback#:~:text=Full%20state%20feedback%20(FSF)%2C,locations%20in%20the%20s%2Dplane.

[13] Paz, R. A. (2001). The design of the PID controller. Klipsch school of Electrical and Computer engineering, 8, 1-23.

[14] Arduino Uno. (2022). Arduino.

https://store.arduino.cc/products/arduino-uno-rev3.

[15] Raspberry Pi. (2022). Raspberry

https://www.raspberrypi.org/.

[16] A4988 Driver. (2022). Polou.

https://www.pololu.com/product/1182.

[17] Tinkercad. (2022). Tinkercad.

https://www.tinkercad.com/.

[18] Willner, D. & Chang, Chaw-Bing & Dunn, Keh-Ping. (1977). Kalman filter algorithms for a multi-sensor system. 570 - 574. https://www.researchgate.net/publication/224680746_Kalman_filter_algorithms_for_a_multi-sensor_system.

[19] Roose, A. I., Yahya, S., & Al-Rizzo, H. (2017). Fuzzy-logic control of an inverted pendulum on a cart. Computers & Electrical Engineering, 61, 31–47.

[20] Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., & Yokoi, K. (2010). Biped walking stabilization based on linear inverted pendulum tracking. IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings. 4489 - 4496.

[21] https://components101.com/motors/nema17-stepper-motor.

[22] https://www.adafruit.com/product/4416.

[23] https://www.sensorae.com/inicio/1827-sensor-de-infrarrojos-detector-obstaculo-ir-infrarojos-infrared-obstacle.html.

[24] https://cursos.mcielectronics.cl/2019/06/18/giroscopio/.

[25] https://descubrearduino.com/mpu6050/.

[26] https://www.te.com/usa-en/products/sensors/position-sensors/tilt-sensors-inclinometers.html.

[27] https://www.arduiner.com/wp-content/uploads/2016/09/12131-MP1584EN-3A-DC-to-DCStep-down-Voltaggio-Regolatore-Buck-Convertitore-Modulo.jpg.

[28] Çakan, Abdullah & Botsali, Fatih. (2016). Modeling and LQR Control of a Wheeled Self-Balancing Robot.

https://www.researchgate.net/publication/311962773_Modeling_and_LQR_Control_of_a_Wheeled_Self-Balancing_Robot