Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Volumen I

Memoria – Esquemas – Presupuesto - Pliego de Condiciones - Anexos

TRABAJO DE FINAL DE GRADO

"DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO"

TFG presentado para optar al título de GRADO en
INGIENERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Por **Eugenio Molis Merino**

Barcelona, 5 de Mayo de 2015

Tutor proyecto: José Mª Huerta Sánchez
Departamento de ESAII
Universitat Politècnica de Catalunya (UPC)

INDICE MEMORIA

RESUMEN

En este proyecto, se ha realizado el diseño y el control de un sistema electromecánico inestable de dos grados de libertad. Aplicación al caso de un péndulo invertido, un sistema no lineal e inestable controlado mediante un microcontrolador para la etapa de control, un giroscopio para la lectura de datos para su posterior procesamiento y control y un driver para la etapa de potencia.

Para el diseño del controlador y la obtención de los diferentes parámetros de ajuste ha sido necesario el software Simulink de Mathworks con la herramienta PID Tuner, introduciendo esos parámetros posteriormente en el software de Arduino UNO en el cual se ha realizado el código fuente para la estabilización del sistema en la ejecución de dicho conjunto de instrucciones.

RESUM

En aquest projecte, s'ha realitzat el disseny i el control d'un sistema electromecànic inestable de dos graus de llibertat. Aplicació al cas d'un pèndol invertit, un sistema no lineal i inestable controlat mitjançant un microcontrolador per a l'etapa de control, un giroscopi per a la lectura de dades per al seu posterior processament i control i un driver per a l'etapa de potència.

Per al disseny del controlador i l'obtenció dels diferents paràmetres d'ajust ha estat necessari el software Simulink de Mathworks amb l'eina PID Tuner, introduint aquest paràmetres posteriorment en el software de Arduino UNO en el qual s'ha realitzat el codi font per a l'estabilització del sistema en l'execució d'aquest conjunt d'instruccions.

ABSTRACT

In this project, has made the design and control of an unstable electromechanical system with two degrees of freedom. Application to an inverted pendulum, a nonlinear and unstable system controlled by a microcontroller to control stage, a gyroscope for reading data for further processing and control and a driver for the power stage.

For the controller design and obtaining different tuning parameters has been necessary Mathworks Simulink software with PID Tuner tool, then deploying it to the Arduino UNO software which has made the source code for the stabilization of the system in the execution of the instruction set.

AGRADECIMIENTOS

En primer lugar querría agradecer a mi familia, especialmente a mi padre ya que no solo por este proyecto sino desde una visión más profunda ha sido una pieza importante en el camino que me ha llevado hasta aquí.

También querría hacer mención a mi tutor José María Huerta Sánchez, ya que para cualquier consulta durante todos estos meses, siempre ha tenido las puertas de su despacho abiertas para atenderme.

CAPÍTULO 1: INTRODUCCIÓN

El péndulo invertido es un problema muy común de resolver en la teoría de control. El péndulo se monta normalmente en un carro y se equilibra con el control del movimiento del carro.

La compañía Segway Inc, en diciembre de 2001 presentó su vehículo de transporte ligero eléctrico de dos ruedas, con autobalanceo controlado por ordenador, que está basado en la misma dinámica que el péndulo invertido. Dispone de dos ruedas con una plataforma entre ellas donde se sitúa el pasajero. El pasajero actúa como el péndulo al ser equilibrado, en este caso, el pasajero lo dirige al inclinarse hacia delante o hacia atrás, siendo el ordenador y los motores situados en la base los que mantienen la base del segway horizontal todo el tiempo.

Este proyecto apunta a construir un robot que consiga mantener el equilibrio de forma autónoma haciendo estable su sistema de control como hace un segway.

Fue diseñado para ser pequeño y poder ver su funcionalidad en cualquier espacio, incluso encima de una mesa, viendo todos sus componentes de los que está construido a simple vista en su estructura abierta.

1.1. Propósito y objetivos

El objetivo principal fue diseñar y construir un robot capaz de mantener el equilibrio en posición vertical sobre sus dos ruedas. El proceso de equilibrio se refiere típicamente como control de estabilidad. Las dos ruedas están situadas debajo de la base y permiten que el chasis del robot mantenga su posición vertical sobre ellas moviéndose en la dirección de inclinación, ya sea hacia

adelante o hacia atrás, en un intento de mantener el centro de masa por encima de los ejes de las ruedas.

El robot debe ser capaz de trabajar por su cuenta sin ningún supervisor. Después de accionar el interruptor situado en la fuente de tensión, el robot debe mantener su posición vertical de forma autónoma.

También debe ser pequeño, así que no requiera mucho espacio, siendo fácil de transportar y exponer en cualquier espacio.

Los distintos objetivos han sido los siguientes:

- Diseño y posterior montaje mecánico de la estructura del robot.
- Diseño eléctrico del sistema.
- Obtener el modelo matemático.
- Estudio de la estabilidad y controlabilidad del sistema.
- Simular y validar el modelo matemático obtenido.
- Programación del algoritmo de control.

La teoría utilizada para mantener la estabilidad del robot, se basa en la teoría del péndulo invertido.

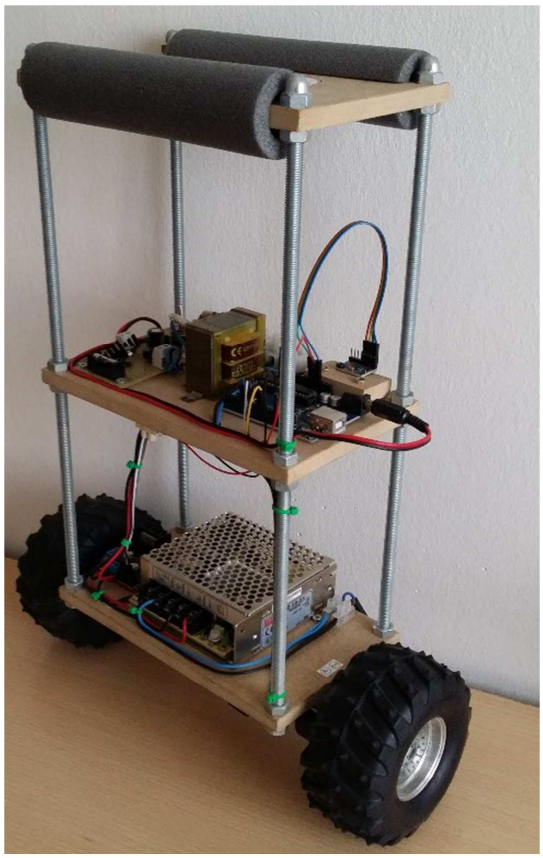


Figura 1. *Robot equilibrista*

CAPÍTULO 2: DESCRIPCIÓN DEL SISTEMA

En este apartado se hace una descripción de todos los componentes tanto mecánicos como electrónicos que se han utilizado para la realización de este robot.

2.1. Sistema mecánico

El diseño general del robot es un cuerpo rectangular sobre dos ruedas paralelas entre sí. El cuerpo rectangular se compone por tres bases de madera prensada separadas unos 18 cm entre ellas tipo estantería, unidas mediante cuatro varillas roscadas situadas en las cuatro esquinas de cada una de las bases de madera.

En la base de madera inferior, que se encuentra encima de ambos motores y en medio de las dos ruedas se ha situado la etapa de potencia, formado por el driver l298 y una fuente de alimentación de 24 V.

En la base de madera de en medio, 18 cm más arriba que esta última se sitúa la etapa de control. En ella se encuentra una fuente de tensión de 12 V continua, un transformador de 12 V, la placa Arduino UNO y el giroscopio/acelerómetro MPU6050.

Finalmente en la base de madera superior no se ha instalado nada, puesto que hace de techo del robot.

El diseño mecánico no se ha realizado de forma arbitraria, ya que se ha situado el componente de mayor peso en la base de en medio, lo más lejano posible del eje de giro, colocando el centro de gravedad lo más alto posible.

En la siguiente figura se puede ver la estructura del robot, descrita anteriormente.

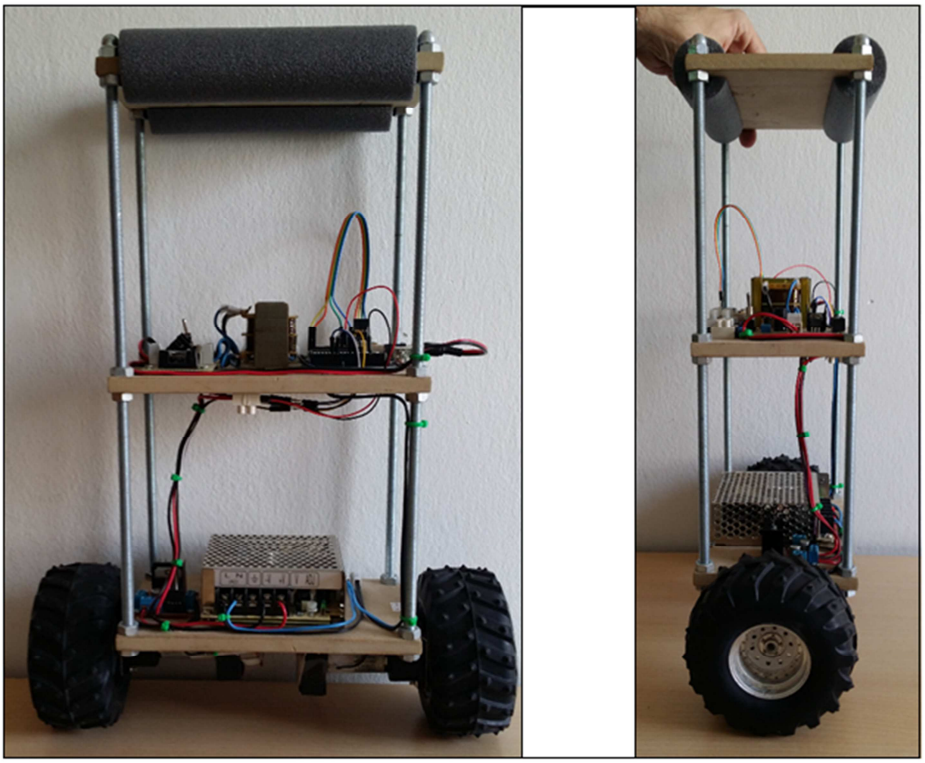


Figura 2. *Robot, vista frontal y de perfil.*

La distancia entre las ruedas es de 22 cm, ya que la anchura de los tablones de madera es de 20 cm de ancho por 12 cm de profundidad, una medida de superficie ideal para instalar los distintos elementos físicos como su cableado.

Respecto a la altura del robot, éste presenta unos 40 cm desde el eje de giro rotacional hasta el techo. Se podría haber construido con menor altura, pero se ha decidido una altura generosa teniendo en cuenta el momento de inercia del cuerpo, ya que interesa una inercia elevada para que presente mayor resistencia al movimiento rotacional respecto a su eje de giro, es decir interesa que caiga más despacio para tener mayor tiempo de reacción.

La distribución de masa, también ha sido un aspecto a tener en cuenta. El momento de inercia (I) se designa como la masa de un cuerpo por el cuadrado de la distancia en este caso al eje de giro, es decir que la inercia de un cuerpo cuando gira, no solo depende de su masa, sino que también depende de cómo esté distribuida esa masa respecto al eje de giro. Cuanto más alejada esté la masa del eje de giro, mayor inercia presentará y más costará su movimiento de forma circular. Por esta razón, el transformador de 12 V al ser el elemento más pesado de todos, se ha situado en la base de madera más lejana al eje de giro situado en las ruedas del robot.

Una vez explicado el montaje mecánico con una descripción de la estructura del robot, sus medidas y su distribución principal, se explicará de forma detallada todos los componentes mecánicos que la forman.

2.1.1. Tablero de madera prensada

Utilizada madera prensada de 1 cm de grosor, para los tres niveles distintos en la estructura del robot. Con una anchura (de rueda a rueda) de 20 cm y una profundidad de 12 cm.

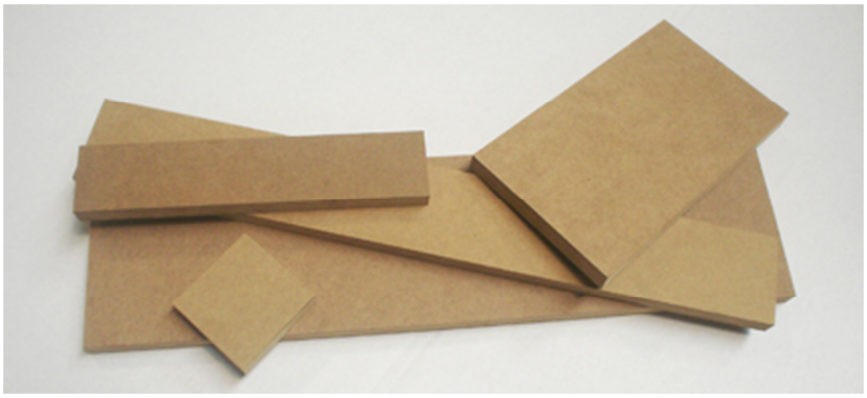


Figura 3. *Madera prensada.*

2.1.2. Varilla roscada

La varilla roscada de zinc de 8 mm de diámetro es la encargada de unir las tres bases de madera prensada, formando la estructura del robot en forma rectangular. Cada una de las cuatro tiene una longitud de 38 cm, situándose en las respectivas esquinas de cada superficie rectangular de madera. Cada base de madera se ajusta con ocho tuercas, tanto en su parte superior como inferior a la propia varilla roscada. La base de madera que hace de techo del robot, en su parte superior se ha puesto cuatro tuercas ciegas, por motivo de estética.

Figura 4. *Varilla roscada de zinc, tuerca normal y tuerca ciega.*

2.1.3. Pieza de hierro en forma de L

Para unir el motor a la base de madera inferior, se ha fabricado una pieza de hierro en forma de L para cada motor. De esta manera mediante cuatro tornillos se ajusta un lado de la pieza de hierro a la parte inferior del tablón de madera, mientras que por la otra cara de la pieza de hierro mediante otros cuatro tornillos se ajusta el motor a ésta, dejando un orificio en su parte central para que pueda entrar el eje del motor que irá introducido a la rueda correspondiente. De este modo tenemos unidos la estructura del robot con los motores y los ejes de éstos a cada una de las ruedas.

En la siguiente imagen se puede observar el montaje mecánico realizado.

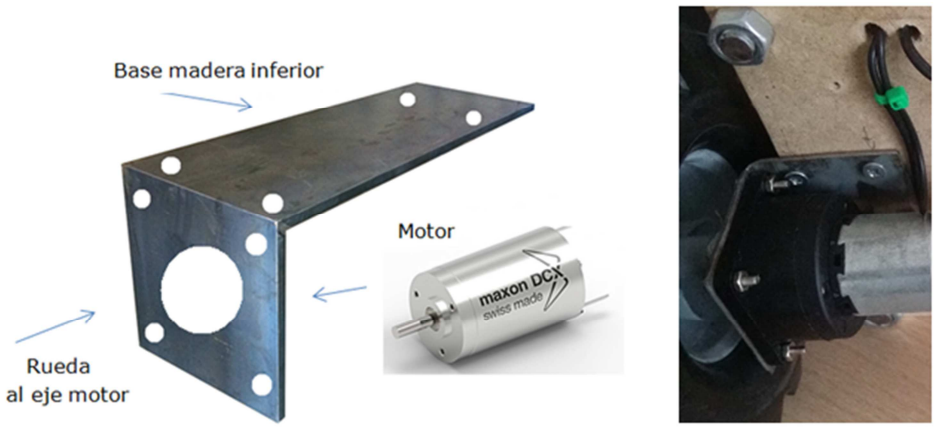


Figura 5. *Mecanización Rueda-Motor-Estructura.*

2.1.4. Ruedas

Las ruedas han sido escogidas con un diámetro de 12 cm y 5 cm de ancho para tener una mejor estabilidad teniendo en cuenta las dimensiones del cuerpo del robot y la velocidad angular proporcionada por los motores.

Las ruedas están hechas de plástico ligero con un neumático de caucho y espuma en su interior proporcionando un mayor agarre y estabilidad. La rueda va unida al eje del motor por su parte interior central.

Figura 6. *Ruedas del robot.*

2.2. Sistema eléctrico

En este apartado se realizará una descripción de todos los componentes eléctricos que hay instalados en el robot, tanto los que forman la parte de control, el sensor de medida del ángulo como la etapa de potencia que alimenta los motores que mueven las ruedas.

2.2.1. Placa Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. Pertenecen a lo que se conoce como código abierto.

Existen multitud de tipos de placas Arduino dependiendo de los requerimientos, con más entradas/salidas, con wifi, para aplicaciones de miniaturización etc.

En este caso se ha utilizado el modelo de placa Arduino UNO, que es la placa básica, la más extendida y según sus características, ideal para iniciarse en este proyecto.

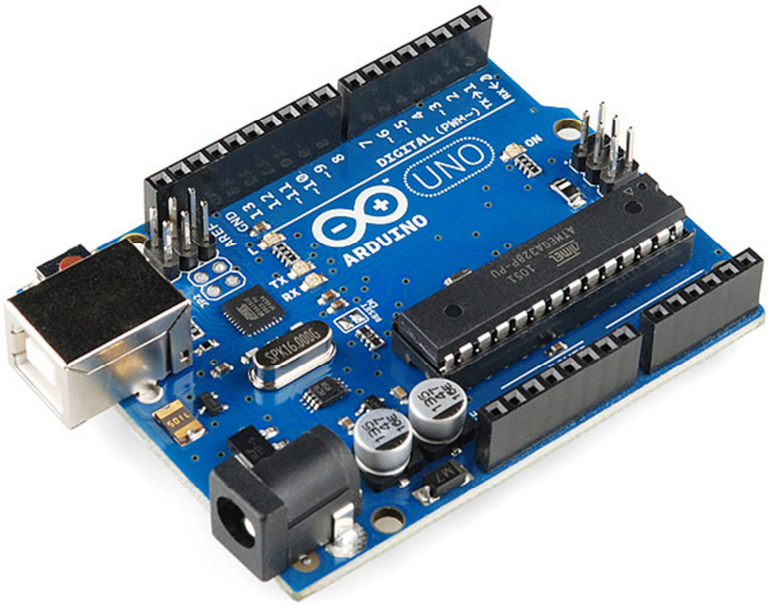


Figura 7. *Placa electrónica Arduino UNO.*

Esta placa Arduino Uno está basada en el procesador ATmega 328. Cuenta con 14 entradas/salidas digitales (de las cuales 6 proporcionan salida PWM), 6 pines de entradas analógicas, un reloj de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reset. Contiene todo lo necesario para dar apoyo al microcontrolador. Puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa a través del conector Jack o a través del Pin denominado Vin. El rango recomendado de tensión de alimentación DC es de 7 a 12 voltios (normalmente 9V).

Arduino Uno posee también un número de instalaciones para la comunicación con un ordenador, otros Arduino, u otros microcontroladores. El ATmega328 ofrece UART TTL (5V) de comunicación serial, que está disponible en los pines digitales 0 (RX) y 1 (TX). Un ATmega16U2 en la placa usa esta comunicación serie a través de USB y aparece como un puerto COM virtual en el software en el ordenador. El software de Arduino a través de comunicación serie permite que simples datos de texto se envíen desde y hacia la placa Arduino. El RX y TX LED en el tablero parpadea cuando los datos se transmiten a través del chip USB a serie y la conexión USB al ordenador. El ATmega328 también es compatible con I2C comunicación (librería Wire) y SPI (mediante la biblioteca SPI). La placa tiene un multifusible reseteable que protege los puertos USB de tu ordenador de cortocircuitos y sobrecorrientes y las dimensiones de ella son de 6,9 y 5,3 cm.

2.2.2. IMU

La Unidad de Medición Inercial o IMU es un dispositivo electrónico que mide y registra información obtenida acerca de la velocidad, la orientación y los efectos de las fuerzas gravitatorias.

En este caso se ha trabajado con la IMU MPU-6050, encargada de conseguir unas lecturas precisas sobre el ángulo de rotación de nuestro robot, e ideal para trabajar con la placa Arduino.

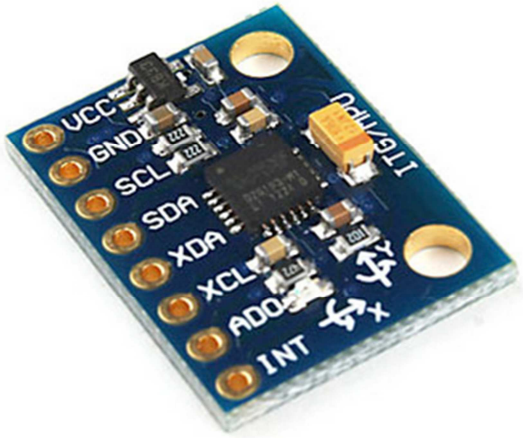


Figura 8. *IMU MPU-6050.*

Utiliza una combinación de acelerómetro y giróscopos para obtener sistemas de 6 grados de libertad, que pueden ser ampliados hasta los 9 incluyendo magnetómetros.

Para los requerimientos de nuestro robot será suficiente con emplear el sistema de 6 grados de libertad que incluye 3 acelerómetros dispuestos de forma ortogonal y 3 giróscopos dispuestos también de forma ortogonal.

La implementación de los sensores en este formato IMU se trata a continuación.

- Acelerómetro

Los acelerómetros MEMS ("Microelectromechanical Systems") son de tamaño reducido y pueden estar incluidos en las IMU anteriormente descritas. La principal ventaja de estos dispositivos MEMS es que pueden ser creados mediante técnicas de fabricación microelectrónica en un chip de silicio, a la vez que toda la electrónica necesaria para el sistema de acondicionamiento, adquisición y comunicación a un precio y un tamaño muy reducidos.

La estructura de estos dispositivos puede apreciarse en la imagen, contienen por lo general placas capacitivas internas, algunas fijas y otras móviles. Las fuerzas de aceleración que actúan sobre el sensor variarán la disposición de unas con respecto a las otras modificando la capacitancia existente entre ambas. Estos cambios serán traducidos posteriormente en señales que podremos utilizar.

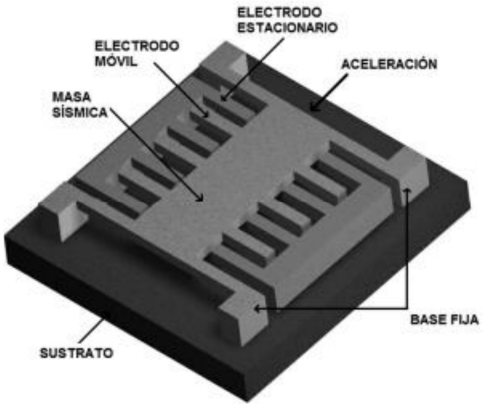


Figura 9. *Acelerómetro MEMS.*

- Giroscopio

Los sensores giroscópicos incluidos en el IMU son también MEMS, su funcionamiento está basado en una pequeña masa que varía su posición al variar la velocidad angular, el dispositivo convierte estos cambios en una señal medible.

Gracias a este diseño, ha sido posible reducir de forma notable el tamaño de estos dispositivos y poder así incluirlos en un circuito integrado.

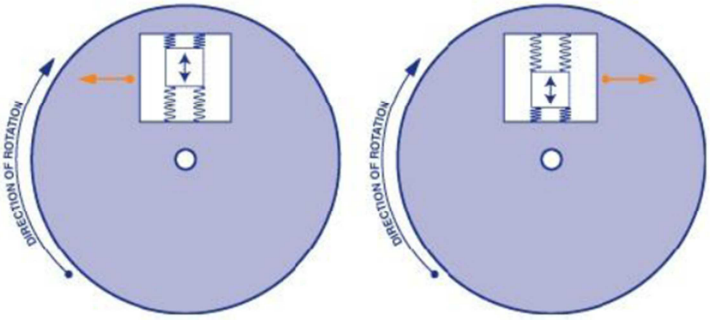


Figura 10. *Giróscopo MEMS.*

El MPU-6050 opera con 3.3 voltios, y se alimenta directamente desde la placa Arduino, ya que éste dispone de una fuente de alimentación de ese voltaje en uno de sus pines. El MPU-6050 utiliza el protocolo de comunicación I²C, descrito con más detalle más adelante.

2.2.3. Driver

Para el control y alimentación de los dos motores, se ha usado el driver o controlador L298N. Este módulo doble puente H basado en el chip L298N te permite controlar dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios.

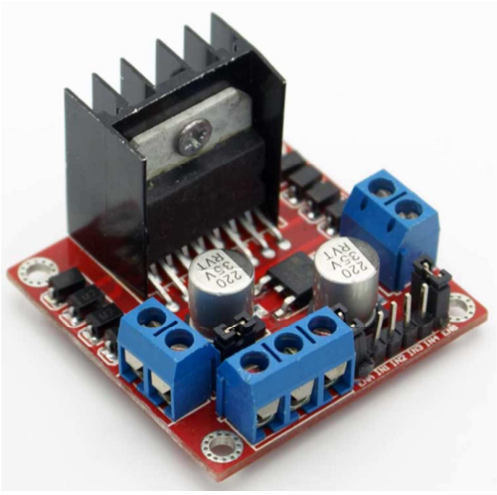


Figura 11. *Driver L298N.*

El controlador permite controlar el sentido de giro y velocidad mediante señales TTL que podrás sacar de tu microcontrolador favorito, en este caso el de Arduino UNO. El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales, entre ellos diodos de protección y un regulador LM7805 que suministra 5V a la parte lógica del integrado L298N. Cuenta con jumpers de selección para habilitar cada una de las salidas del módulo (A y B) para cada uno de los dos motores. La salida A está conformada por OUT1 y OUT2 y la salida B por OUT3 y OUT4. Los pines de habilitación son ENA y ENB respectivamente.

Este módulo se puede alimentar de 2 maneras gracias al regulador integrado LM7805.

Cuando el jumper de selección de 5V se encuentra activo, el módulo permite una alimentación de entre 6V a 12V DC. Como el regulador se encuentra activo, el pin marcado como +5V tendrá un voltaje de 5V DC. Este voltaje se puede usar para alimentar la parte de control del módulo ya sea un microcontrolador o un Arduino, siempre y cuando que el consumo no sea mayor a 500 mA.

Cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC. Como el regulador no está funcionando, tendremos que conectar el pin de +5V a una tensión de 5V para

alimentar la parte lógica del L298N. Usualmente esta tensión es la misma de la parte de control, ya sea un microcontrolador o Arduino.

En nuestro caso se ha utilizado la segunda opción, dejando el jumper de selección de 5 V inactivo, por lo tanto el driver se alimenta a 24 V DC mediante una fuente de alimentación externa, y entonces la parte lógica del L298N se alimenta mediante una fuente de alimentación que posee el propio Arduino de +5 V DC, de esta manera el driver tiene más potencia para transmitir a las salidas que están conectados los motores.

2.2.4. Actuadores

Para corregir la posición del robot y la desviación del ángulo, se necesitará ejercer un par adecuado en el punto de balanceo, esto se logrará utilizando motores.

Se han utilizado dos motores de corriente continua, uno para el giro de cada rueda. Son motores de la casa maxon motor, de 6 W de potencia, con una tensión nominal de 48 V y un Par nominal de 13.7 mNm. Han resultado motores ideales para este proyecto, con una potencia más que suficiente para la funcionalidad exigible.

Estos motores son controlados mediante salidas digitales por ancho de pulso PWM del microcontrolador. El sentido de giro de estos motores es también controlable según la disposición de la alimentación, cambiando el sentido al invertir la polaridad de sus terminales como se aprecia en la imagen.

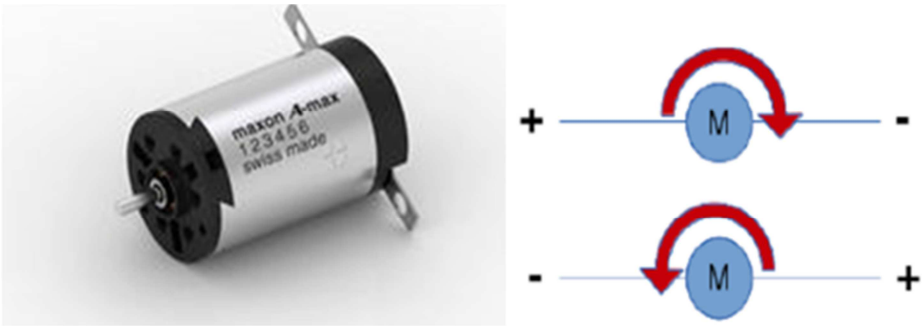


Figura 12. *Izda. Motor DC. Dcha. Relación sentido de giro/polaridad.*

Este será el tipo de actuador seleccionado ya que es el más sencillo y el que mejor se ajusta a las necesidades. De entre la variedad de motores existentes de este tipo, se escogió un motor con reductora como el de la imagen, lo cual incrementará el par obtenido en contra de la velocidad.

La reducción absoluta es de 18 unidades, equivalente a una reducción 18:1.

Figura 13. *Reductor engranaje recto GS 38A.*

2.2.5. Transmisión de potencia

La limitación que tienen los microcontroladores a la hora de entregar corriente, necesaria para alimentar los motores, hace necesaria la inclusión de un dispositivo que sea capaz de suministrarla. Por esta razón se ha utilizado el driver L298N descrito anteriormente, circuitos integrados encargados de suministrar la información del microcontrolador con la potencia adecuada.

Anteriormente ya se ha citado que se trata de un driver doble puente en H.

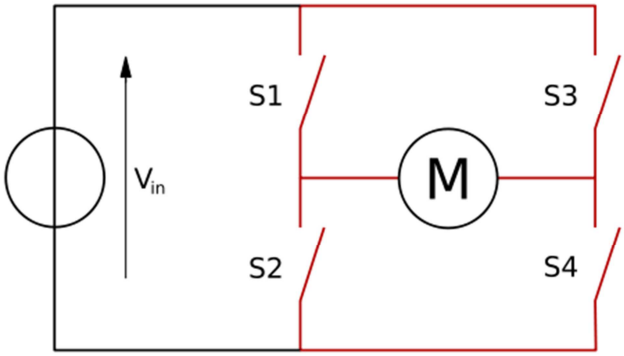


Figura 14. *Estructura de un puente H (marcado en rojo).*

Este tipo de integrados, los puentes en H, son circuitos permiten el cambio del sentido del giro del motor de una forma sencilla, además, son totalmente compatibles con el control de velocidad por PWM. Conseguiríamos así la posibilidad de modificar sentido y velocidad a conveniencia.

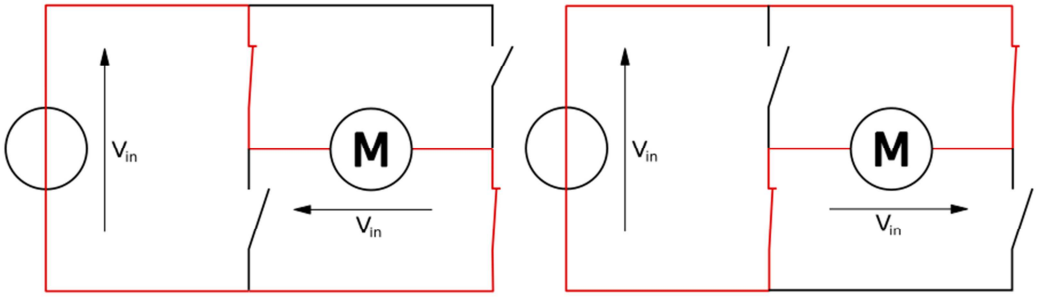


Figura 15. *Los dos estados básicos del circuito (sentido de giro 1 y 2).*

2.2.6. Modulación por ancho de pulsos (PWM)

Arduino UNO dispone de salidas digitales, que pueden trabajar como salidas de modulación por ancho de pulso (PWM), una técnica para obtener resultados análogos con medios digitales.

Figura 16. *Salidas digitales PWM en Arduino UNO.*

A menudo se necesita algo más que una señal de 0 o 1 en nuestros proyectos, para variar la velocidad de giro de un motor, para variar la intensidad con la que luce un diodo, para transmitir los grados de giro de un servo, etc.

Para todo esto, y mucho más, nos servirá el PWM, que emula una señal analógica a partir de una señal digital.

Lo que hace este tipo de señal es emitir, en lugar de una señal continua en nuestra salida, el control digital se utiliza para crear una onda cuadrada, una señal cambia entre encendido y apagado emitiendo una serie de pulsos que podremos variar su duración (ancho de pulso) con una frecuencia constante de aproximadamente 490Hz, de manera que la tensión promedio resultante, es directamente proporcional a la duración de éstos dentro del rango de nuestro periodo, es decir, cuanto más anchos sean esos pulsos de +5v de amplitud, mayor será la tensión promedio de nuestra salida, y cuanto más estrechos sean, menor será dicha tensión:

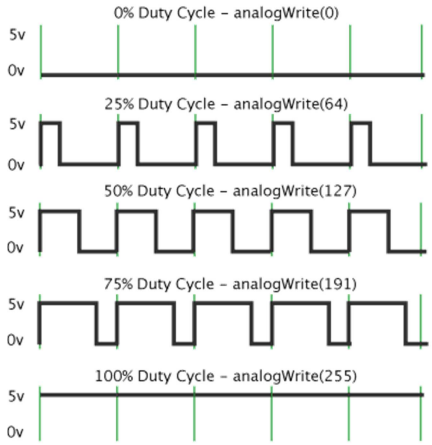


Figura 17. *PWM (Pulse Width Modulation).*

2.3. Alimentación del sistema

Para la alimentación de todos los dispositivos eléctricos del robot no se han utilizado baterías, por lo que se alimentan mediante la red eléctrica.

2.3.1. Alimentación etapa de control

El microcontrolador Arduino UNO, su rango recomendado de tensión de alimentación DC es de 7 a 12 voltios, por lo tanto se ha diseñado y construido una fuente de tensión que entrega 12 V DC a su salida, para alimentar dicha placa.

Inicialmente se ha utilizado un transformador convencional de 12 V DC y 0.5 A DC.

Figura 18. *Transformador Crovisa 12V+12V 0.5A.*

Este transformador se encarga de transformar los aproximadamente 220 V AC que hay conectados a su entrada proveniente de la red eléctrica a entregar alrededor de 12 V AC a sus dos terminales de salida.

Teniendo presente que la placa Arduino necesita 12 V pero no de alterna, sino de continua, se ha diseñado y fabricado una fuente de tensión encargada de convertir los 12 V AC que entrega a su salida el transformador, a 12 V DC aptos para poder alimentar el microcontrolador.

La fuente de tensión ha sido diseñada y construida de forma casera, sobre una placa de baquelita o también llamada de topos. En ella se han integrado todos los dispositivos electrónicos necesarios para realizar lo más estable y preciso posible dicha conversión de voltaje.

A continuación se puede ver el esquema eléctrico básico de la fuente de alimentación de 12 V DC.

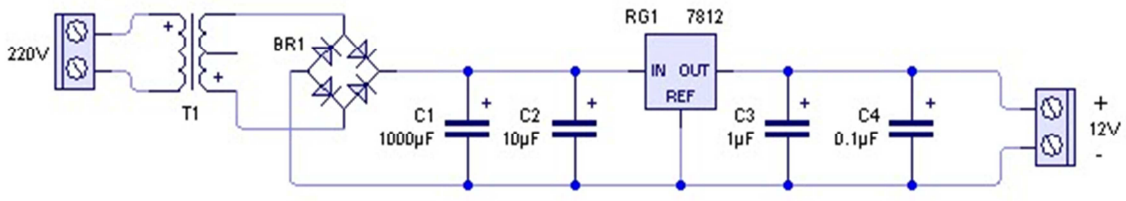


Figura 19. *Fuente de tensión 12 V DC.*

El bobinado secundario del transformador, se conecta al puente rectificador, encargado de convertir la tensión alterna del transformador a tensión continua mediante la conducción de sus diodos. Por lo tanto la tensión que se obtiene a la salida de este puente rectificador ya tiene polaridad.

La tensión en la carga que se obtiene de un rectificador es en forma de pulsos, ésta no es la clase de tensión continua que precisan la mayor parte de circuitos electrónicos. Para obtener una tensión constante lo más parecido a lo que daría una batería es necesario emplear un filtro.

De esta manera se han introducido condensadores, también llamado (condensador de aplanamiento), condensadores polarizados entre los conectores de salida del rectificador. El terminal positivo del condensador polarizado debe conectarse con la salida positiva del regulador. Este condensador debe tener una capacidad en faradios (F) igual a 5 veces la corriente a suministrar por el conversor CA CC, dividido por el índice secundario del transformador multiplicado por 1,4 veces la frecuencia.

Finalmente se ha escogido un regulador de voltaje comercial, en este caso el L7812CV, diseñado para controlar el voltaje de salida deseado, en este caso 12 V DC. El regulador tiene 3 conectores; uno común, otro para el condensador de acumulación, y otro para la salida del regulador. La salida del regulador también será la salida final del conversor CA CC.

Según la hoja de datos del fabricante del regulador, se especifica un condensador para supresión de ruido. Dicho condensador también ha sido instalado.

Al esquema eléctrico básico de la figura anterior, sobre una fuente de tensión de 12 V DC, se han introducido adicionalmente algunos dispositivos eléctricos para mejorar la protección y funcionalidad de la propia fuente de alimentación. Se ha instalado un fusible con su respectivo portafusible en el secundario del transformador o entrada a la fuente de tensión diseñada de valor de 500 mA, con la finalidad de que cuando la intensidad de corriente supere, por un cortocircuito o un exceso de carga, un determinado valor que pudiera hacer peligrar la integridad de algún dispositivo de la placa, éste deje de circular corriente, dejando abierto el circuito.

En la entrada a la fuente de tensión también se ha instalado un interruptor eléctrico de dos posiciones, con la finalidad de que el usuario pueda gobernar cuando quiere que circule corriente o no. Un led de alta luminosidad de color azul será el encargado de mostrar si el interruptor está cerrado o abierto.

Finalmente se ha utilizado un disipador de potencia o radiador sobre el regulador de tensión con la finalidad de evitar un aumento elevado de temperatura y pueda llegar a quemarse.

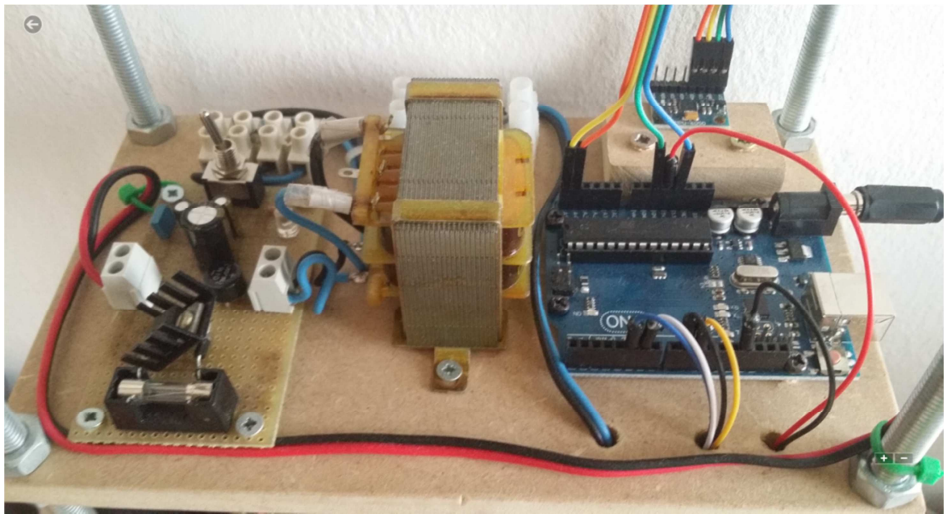


Figura 20. *Alimentación etapa de control.*

Los resultados has sido buenos, consiguiendo un valor estable y próximo a los 12 VDC para alimentar la placa Arduino UNO mediante un conector Jack. El acelerómetro/giroscopio MPU-6050 se alimenta a 3,3 V DC mediante el propio Arduino que posee una fuente de alimentación en dos de sus pines (3,3V-GND).

2.3.2. Alimentación etapa de potencia

De igual forma que para la etapa de control, la etapa de potencia también se alimenta sin la necesidad de baterías. En su caso se ha optado por una fuente de alimentación más potente, ya que el driver encargado de controlar los dos motores, necesita un valor más elevado de voltaje y corriente para su correcto funcionamiento.

En esta ocasión se ha comprado una fuente de alimentación conmutada, con una tensión de salida de 24 voltios DC y una intensidad de corriente de 2,2 Amperios, con una potencia de 50 W. Ideal para proporcionar la potencia suficiente para las características de funcionamiento del driver.

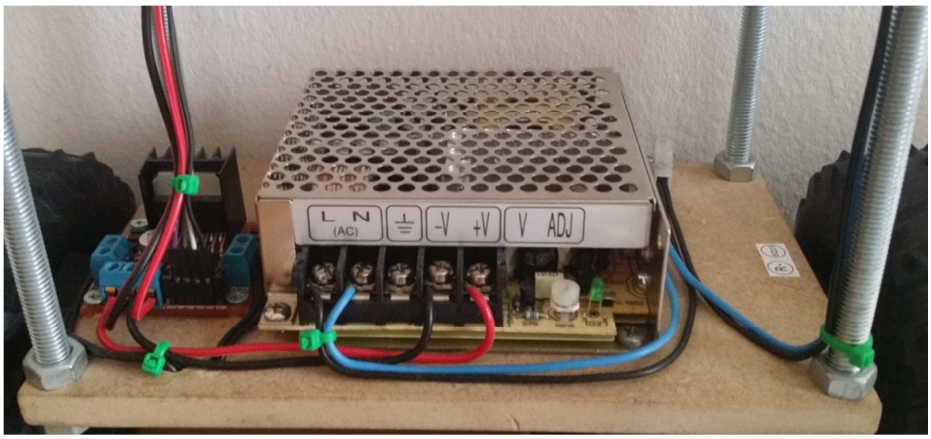


Figura 21. *Alimentación etapa de potencia.*

CAPÍTULO 3:

SOFTWARE

Se han utilizado dos tipos de software en la realización de este proyecto. El software de Arduino y la herramienta Simulink de Matlab.

3.1. Software de Arduino

La plataforma Arduino se basa en un lenguaje de programación sencillo, pero antes de empezar a usarlo es necesario instalar el software propio para la programación de Arduino (IDE). La última versión de este software la podemos descargar de la web oficial de la casa Arduino.

Figura 22. *Logotipo de Arduino Home.*

Una vez descargada, se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing (un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización) que es similar a C++. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino, debido a que Arduino usa la transmisión serial de datos soportada por la mayoría de los lenguajes mencionados. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Algunos ejemplos son: C++, Matlab, Visual Basic, Python entre otros.

Inicialmente al ser Arduino un software desconocido se empezó por lo más básico, familiarizarse con el entorno y el lenguaje de programación. Esta tarea no resultó muy compleja, por toda la información accesible a través de su página web oficial y el foro relacionado. Merecen especial interés:

- Reference: se explica con detalle las estructuras de programa y de control, sintaxis relacionada con dichas estructuras, tipos de variables (constantes, tipos de datos, conversión entre tipos) y algunas funciones de utilidad, todo ello acompañado de ejemplos.

- Examples: incluye gran cantidad ejemplos de programas.

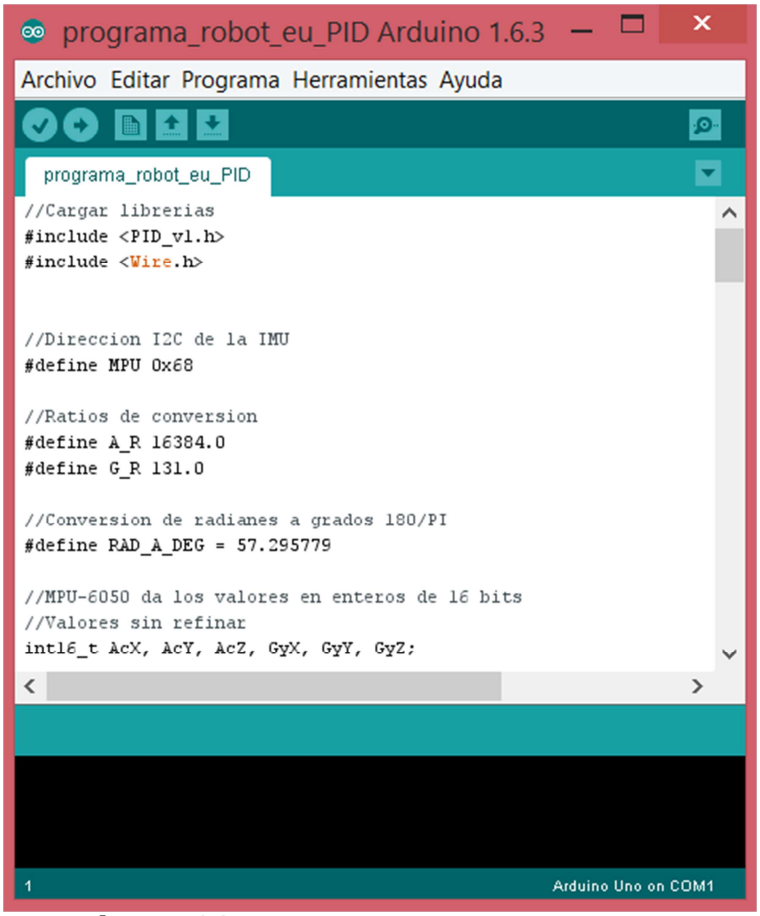


Figura 23. *Interfaz de programación Arduino.*

3.2. Simulink (Matlab)

Simulink es una herramienta de Matlab que funciona mediante un entorno de programación visual, mediante diagramas de bloque para la simulación, por lo tanto su uso es muy intuitivo y sencillo, sin necesidad de emplear lenguajes de programación complejos.

Es un entorno de programación de más alto nivel de abstracción que el lenguaje que interpreta Matlab (archivos con extensión .m). Simulink genera archivos con extensión .mdl (de "model").

Al ejecutar un modelo implementado en Simulink se genera un código en C que el ordenador reconoce y ejecuta.

Mediante Simulink se puede construir y simular modelos de sistemas físicos y sistemas de control mediante diagramas de bloques.

El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo.

Tiene una conexión directa con Matlab, pudiendo exportar los resultados obtenidos en Simulink para hacer análisis más exhaustivos y poder obtener nuevos resultados. También algunos bloques nos dan la posibilidad de incorporar algoritmos propios de Matlab.

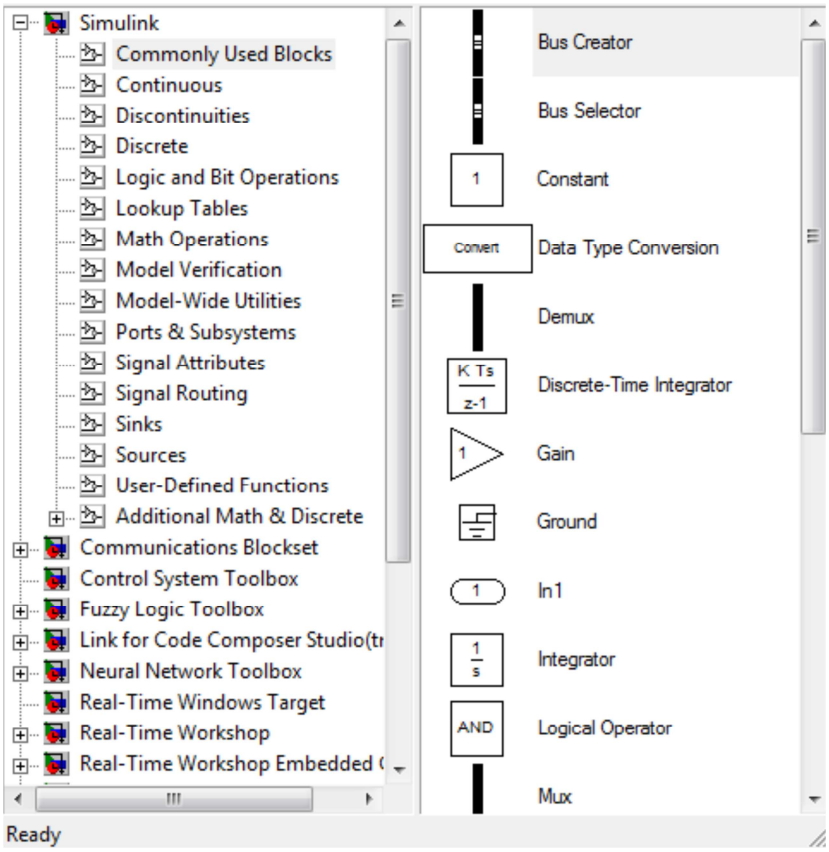


Figura 24. *Aspecto gráfico diagrama bloques en Simulink.*

Simulink ofrece la posibilidad de conectar el modelo con hardware para comprobar en tiempo real y de una manera física el funcionamiento de este. En Simulink es posible crear y simular modelos mecánicos, eléctricos, electrónicos, aeronáuticos, etc. gracias a la gran variedad de bloques de los que dispone. Estos conjuntos de bloques se encuentran agrupados en la Simulink library browser, que se despliega al ejecutar Simulink, y presenta el aspecto de la figura

Es ampliamente usado en Ingeniería Electrónica en temas relacionados con el procesamiento digital de señales (DSP), involucrando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros. También es muy utilizado en Ingeniería de Control y Robótica.

CAPÍTULO 4: OBTENCIÓN DE DATOS DEL MPU6050

Una vez conectada la IMU MPU-6050, no se consigue una lectura de valores de ángulos de inclinación directamente, por lo que habrá que calcularlos previamente, depurar esos valores y filtrar el ruido con un Filtro Complementario (Complementary Filter) para conseguir unas lecturas buenas y precisas.

Según la teoría anterior de la IMU MPU6050, se sabe que es un dispositivo formado por un acelerómetro y un giroscopio, capaz de mesurar la fuerza (aceleración) y la velocidad. Por lo tanto una IMU no mesura ángulos directamente, requiere algunos cálculos.

4.1. El acelerómetro

Como su propio nombre indica, el acelerómetro mide la aceleración, pudiéndose expresar en 3 ejes: X, Y, Z, las tres dimensiones del espacio. Sabiendo que la gravedad de la Tierra tiene una aceleración aproximadamente de 9.8 m/s^2, perpendicular al suelo, la IMU detectará esa aceleración de la gravedad terrestre, pudiendo usar esas lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y.

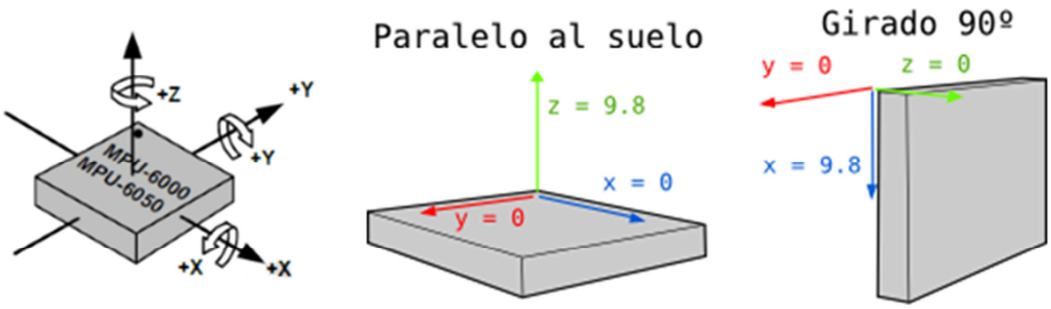


Figura 25. *Funcionamiento del MPU-6050.*

Si sabemos que la gravedad es 9.8 m/s², y sabemos el valor que dan los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la IMU. La fórmula utilizada para calcular el ángulo es:

$$Angulo\ Y = \arctan(\frac{x}{\sqrt{y^2+z^2}}) \qquad Angulo\ X = \arctan(\frac{y}{\sqrt{x^2+z^2}})$$

El eje Z en estos casos se suele ignorar.

En este proyecto, según se ha posicionado el MPU6050 en la estructura del robot, es el eje Y el único que se ha tenido en cuenta, puesto que hace referencia al eje de rotación del robot.

4.2. El giroscopio

Un giroscopio mide la velocidad angular, es decir la velocidad de rotación o el ángulo girado por una unidad de tiempo que tendrá el robot.

Si se sabe el ángulo inicial de la IMU, se podrá sumar el valor que marca el giroscopio para conocer el nuevo ángulo a cada momento. Suponiendo que una vez instalada la IMU en el robot, estando éste completamente vertical sobre sus dos ruedas la IMU marca cero grados calibrado de fábrica, si el giroscopio realiza una medida cada segundo, y marca tres en el eje X, se tiene el ángulo con esta sencilla fórmula.

$$Angulo\ Y = Angulo\ Y\ anterior + \frac{x}{\Delta t}$$

Dónde Δt es el tiempo que transcurre cada vez que se calcula esta fórmula. Y lo mismo pasa con los ejes X,Z. Sólo que al igual que con el acelerómetro se suele ignorar el eje Z.

De esta manera ya se obtienen las lecturas en ángulo de giro respecto al tiempo, aunque todavía no serían valores del todo correctos y útiles.

4.3. El Filtro Complementario

Las IMU's no son dispositivos que con unas simples fórmulas trigonométricas sean capaces de ofrecer unas lecturas de ángulos con total exactitud, ya que surgen dos problemas muy importantes: el ruido y los errores.

El ruido son todas aquellas interferencias a todas aquellas señales, de origen eléctrico, no deseadas y que están unidas a la señal principal, o útil, de manera que la pueden alterar produciendo efectos que pueden ser más o menos perjudiciales.

El acelerómetro es capaz de medir cualquier ángulo, sin embargo sus lecturas son muy ruidosas y tienen un cierto margen de error.

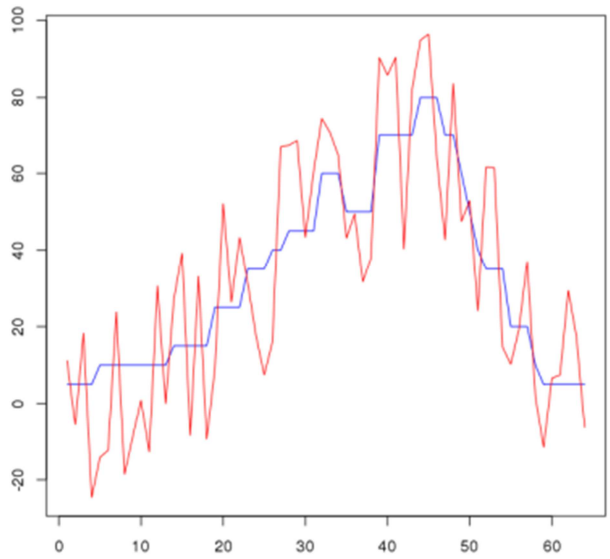


Figura 26. *Señal útil con ruido acoplado.*

El ángulo real (ideal) está marcado en azul, y las medidas reales están en rojo. Se puede afirmar que no se trata de una señal precisa.

El acelerómetro también es capaz de detectar cualquier aceleración que no sea la de la gravedad. Por lo tanto, si la IMU se mueve sin girarla, al aplicar una aceleración en otro eje, la IMU lo detectará como un cambio de rotación.

Por otra parte está el giroscopio. A diferencia del acelerómetro, da las medidas con mucha precisión, pero al realizar los cálculos del ángulo es inevitable que se produzca un pequeño error, que con el tiempo va acumulándose (drift) y acaba resultando significativo.

Ante estos errores o imperfecciones que presentan tanto el acelerómetro como el giroscopio, hay varias formas de combinar los datos de ambos para así obtener medidas más fiables y precisas.

En resumen el objetivo principal es conseguir eliminar el ruido, el drift y que el acelerómetro no cambie de ángulo al detectar otra fuerza que no sea la gravedad.

Hay distintos algoritmos, llamados filtros, que hacen esta tarea. Uno de los mejores es el famoso Filtro de Kálman, capaz de calcular el error de cada medida a partir de las medidas anteriores, eliminarlo y dar el valor real del ángulo. Sin embargo tiene un coste de procesamiento bastante elevado y bastante complejo, por lo que se ha decidido utilizar otro tipo de filtro igual de útil para el caso, más sencillo e ideal para implementar con Arduino. Se trata del Filtro Complementario o Complementary Filter.

El Filtro Complementario tiene una precisión muy buena, y consiste en una unión de dos filtros diferentes: un filtro de paso alto (High-pass Filter) para el giroscopio y un filtro paso bajo (Low-pass Filter) para el acelerómetro.

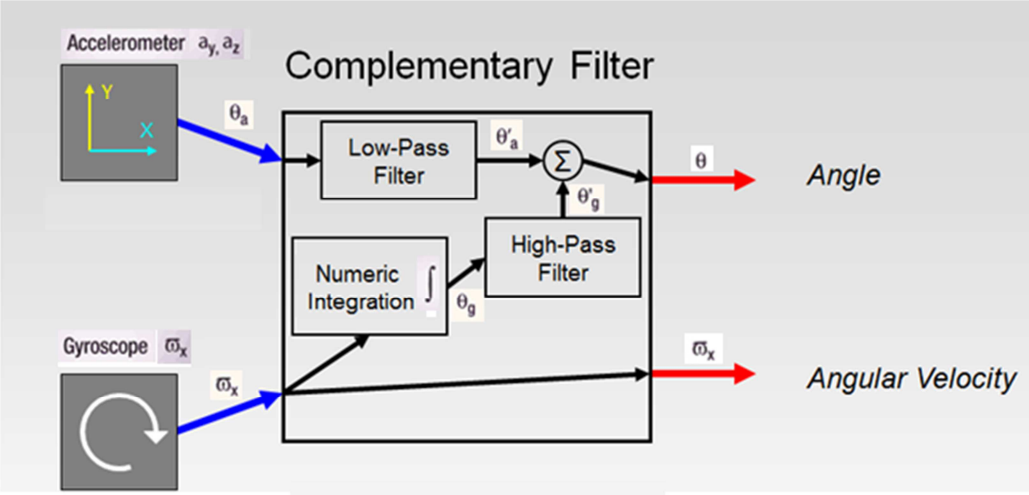


Figura 27. *Estructura Filtro Complementario.*

En el filtro paso alto (HPF), se dice que su respuesta en frecuencia se atenúan las componentes de baja frecuencia, es decir dejando pasar los valores por encima de un cierto límite, todo lo contrario que en el filtro paso bajo (LPF), que sólo permite a los que están por debajo.

La fórmula resultante de combinar (complementar, de aquí su nombre) los dos filtros es:

$$Angulo = 0.98 \cdot (Angulo + AnguloGyro \cdot 0.010) + 0.02 \cdot AnguloAccel$$

Dónde AnguloGyro es el ángulo del Giroscopio que se ha calculado previamente, mientras que AnguloAccel con el acelerómetro.

Todas estas fórmulas, introduciéndolas en el código fuente de Arduino, obtendríamos en tiempo real una lectura fiable y precisa del ángulo de giro del robot.

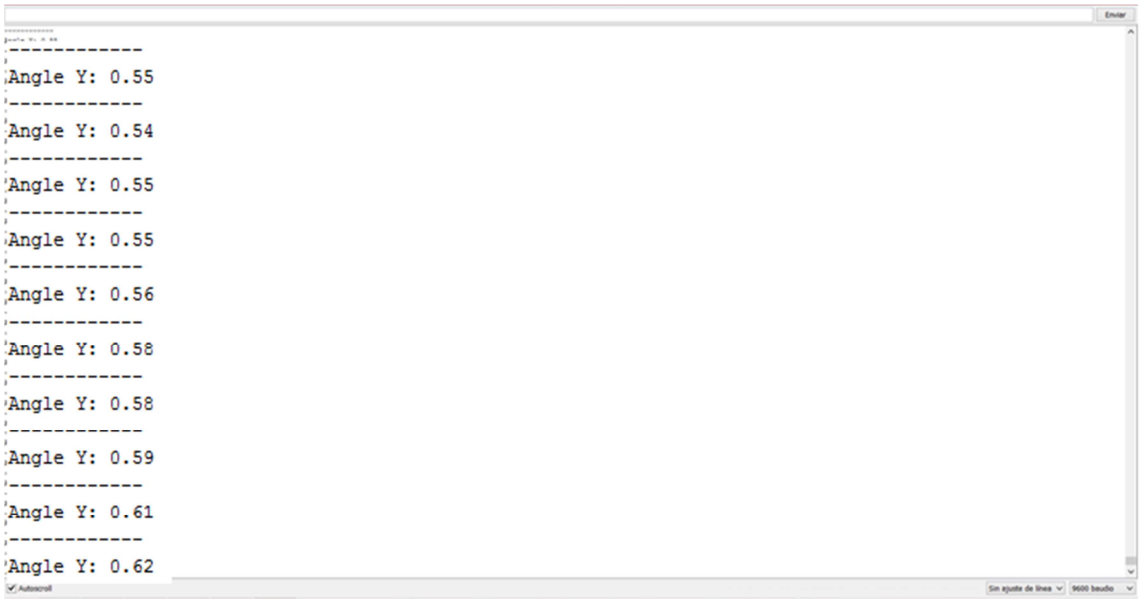


Figura 28. *Lectura ángulo de giro del robot.*

En la figura anterior, se puede ver cómo el Monitor Serie nos da una lectura en tiempo real de la posición del robot. Esta lectura se ha programado por código que se actualice cada 10 ms, y se puede ver que teniendo el robot en posición vertical da unos valores de prácticamente 0 grados de inclinación.

4.4. Comunicación I²C

El MPU-6050 utiliza el protocolo de comunicación I²C. Es un protocolo de comunicación serie diseñado por Philips que se utiliza esencialmente entre dispositivos que pertenecen al mismo circuito, por ejemplo, sensores con un microcontrolador.

Las características principales del protocolo son:

- Velocidad standard de 100Kbit/s (100kbaudios). Se puede cambiar al modo de alta velocidad (400Kbit/s)
- Configuración maestro/esclavo. La dirección del esclavo se configura con software.
- Solo se necesitan dos líneas:
  - SDA (Serial Data Line): Línea de datos.
  - SCL/CLK (Serial Clock Line): Línea de reloj, será el que marque el tiempo de RW (Lectura/Escritura)
  - Nota: Suponemos que todos los dispositivos tienen masa común, si no fuera así hay que incluir una línea de masa.

- Los comunicación siempre tiene la estructura siguiente:
  - Transmisor: Byte de datos (8 Bits)
  - Receptor: Bit llamado ACK de confirmación.

Las conexiones se realizan de la siguiente manera: SDA y SCL van a su pin correspondiente en cada dispositivo, de manera que todos quedan en paralelo.

Las líneas SDA y SCL están independientemente conectadas a dos resistores Pull-Up que se encargaran de que el valor lógico siempre sea alto a no ser que un dispositivo lo ponga a valor lógico bajo.

El tipo de comunicación es half duplex. Comunicación bidireccional por la misma línea pero no simultáneamente bidireccional.

La estructura de la comunicación básica es la siguiente:

1. START condition (Master)
2. 7 Bits de dirección de esclavo (Master)
3. 1 Bit de RW, 0 es Leer y 1 Escribir. (Master)
4. 1 Bit de Acknowledge (Slave)
5. Byte de dirección de memoria (Master)
6. 1 Bit de Acknowledge (Slave)
7. Byte de datos (Master/Slave (Escritura/Lectura))
8. 1 Bit de Acknowledge (Slave/Master (Escritura/Lectura))
9. STOP condition (Master)

Esta es la base de la comunicación pero para leer o escribir, según el dispositivo con el que se comunica el Master la comunicación tendrá una estructura específica.

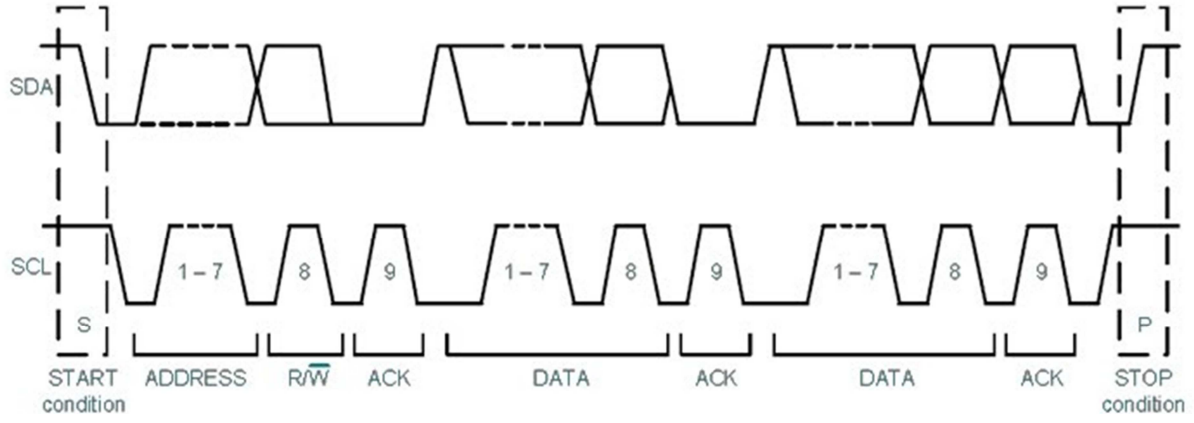


Figura 29. *Estructura básica del bus I²C.*

A nivel de programación, es bastante sencillo puesto que Arduino contiene la librería "Wire.h", para la interacción vía protocolo I²C en la que se incluyen todas las funciones necesarias.

CAPÍTULO 5:
MODELO MATEMÁTICO

Como se menciona en la introducción, se comenzará analizando el problema físico del equilibrio de un péndulo invertido, que consiste en un péndulo cuyo centro de masas se encuentra situado por encima del punto o eje de balanceo. Esta disposición dota al sistema de una inestabilidad estática, que puede ser compensada aplicando un momento o par determinado en el eje de giro.

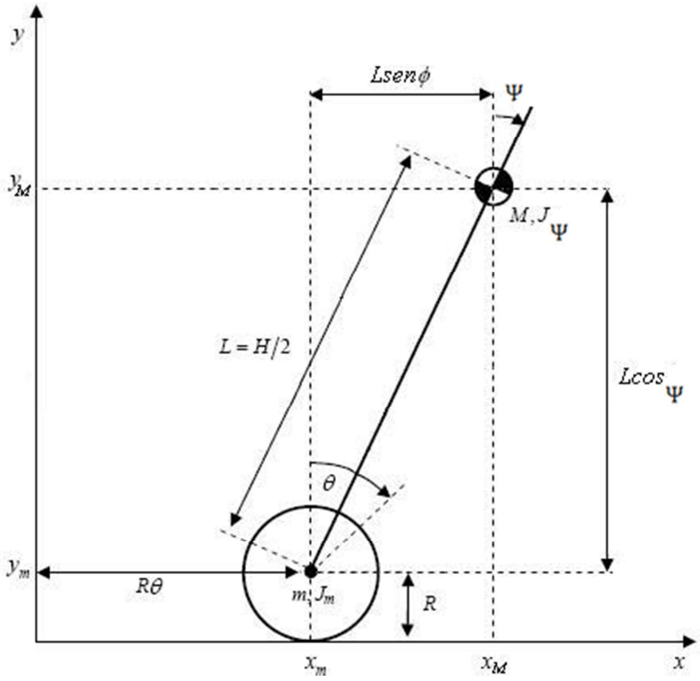


Figura 30. *Vista lateral del péndulo invertido sobre dos ruedas.*

El fundamento de este proyecto consiste de forma simplificada en tratar de conseguir y controlar dicho equilibrio, basándose en el ángulo formado con la vertical. También deberá tenerse en cuenta las aceleraciones que actúen en el robot.

El análisis de estos sistemas constituye uno de los clásicos problemas abordados en la teoría de control y en la dinámica de sistemas, es por ello tratado en diversos textos. También se suele utilizar este problema como banco de pruebas para ensayos con diferentes tipos de controles como pueden ser PID (Proporcional Integral Derivativo), sistemas de control en el espacio de estados, entre otros.

Considerando el problema como un péndulo invertido y aproximándolo mediante consideraciones como punto de balanceo fijo, y sistema de movimiento en 2 dimensiones, obtenemos como ecuación principal del comportamiento de nuestro sistema, una función en la que la aceleración angular ($\ddot{\Psi}$) es dependiente de la gravedad (g), la longitud del péndulo (L) y el seno del ángulo formado (Ψ).

$$\ddot{\Psi} = \frac{g}{L} sin\Psi$$

En esta sección se presenta el modelado matemático completo del sistema físico construido en la sección anterior. En la Figura 30, se muestra la vista lateral y el sistema de coordenadas sobre el cual se elabora el modelo matemático para el péndulo invertido sobre dos ruedas.

Se tendrán en cuenta todos los parámetros físicos del sistema, con sus respectivas unidades y breve descripción.

5.1. Modelo matemático del sistema

La descripción matemática del robot se divide en tres partes, una para el cuerpo del robot, donde su comportamiento sería equivalente al de un péndulo invertido, otra parte para las ruedas y la última para el sistema eléctrico del motor. El péndulo y las ruedas tienen 3 ecuaciones cada una, una para la dirección rotacional y dos para dirección "x" y dirección "y".

El péndulo será Ψ para el ángulo y $\dot{\Psi}$ para la velocidad angular, mientras que para las ruedas se ha considerado θ para su ángulo y $\dot{\theta}$ para su velocidad angular.

Las siguientes tablas presentan los parámetros que se utilizarán en el modelo.

Tabla 1. *Parámetros físicos del péndulo (estructura del robot).*

Parámetros	Descripción	Unidades
g	Aceleración de la gravedad	m/s^2
J_p	Inercia del péndulo (estructura robot)	$Kg \cdot m^2$
m_p	Masa del péndulo (estructura robot)	Kg
L	Distancia centro rueda al centro del péndulo	m
Ψ	Ángulo del péndulo	rad
$\dot{\Psi}$	Velocidad angular péndulo	rad/s
X_p	Dirección X del péndulo	m
Y_p	Dirección Y del péndulo	m
N_x	Fuerza entre el péndulo y rueda en dirección X	N
N_y	Fuerza entre el péndulo y rueda en dirección Y	N

Tabla 2. *Parámetros físicos de las ruedas.*

Parámetros	Descripción	Unidades
J_w	Inercia de la rueda	$Kg \cdot m^2$
m_w	Masa de la rueda	Kg
r	Radio de la rueda	m
θ	Ángulo de la rueda	rad
$\dot{\theta}$	Velocidad angular de la rueda	rad/s
X_w	Dirección X de las ruedas	m
Y_w	Dirección Y de las ruedas	m
N	Fuerza Normal del suelo sobre las ruedas	N
F	Fuerza de fricción entre el suelo y las ruedas	N

Tabla 3. *Parámetros circuito eléctrico (Motor DC).*

Parámetros	Descripción	Unidades
R_a	Resistencia eléctrica nominal	$Kg \cdot m^2$
K_t	Constante Par motor DC	Kg
K_e	Constante EMF motor DC	m
T_m	Par motor	rad
U	Voltaje entrada motor	rad/s
n	Relación de transmisión (Gear Ratio)	m

5.1.1. Modelo físico del péndulo: estructura robot

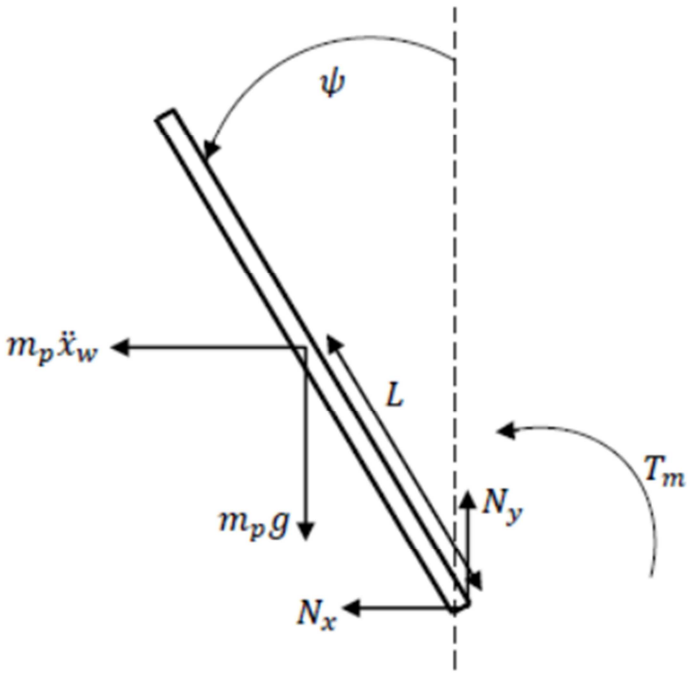


Figura 31: *Fuerzas que actúan sobre la estructura del robot (péndulo).*

Las ecuaciones del péndulo son las siguientes:

Ecuación (1), es el momento en sentido antihorario alrededor del centro de masa del péndulo.

$$T_m + m_p g L sin\Psi + m_p \ddot{x}_w L cos\Psi = J_p \ddot{\Psi} \quad \textbf{(1)}$$

Dónde: T_m: Par de los motores (N·m)

$m_p \cdot \ddot{x}_w$: (Masa x Aceleración) ⟶ Fuerza (Kg·m/s²)

$J_p \ddot{\Psi}$: Inercia péndulo x aceleración angular (rad/s²)

Las ecuaciones (2) y (3) son las fuerzas que actúan en la dirección X e Y respectivamente.

$$- N_x - m_p\ddot{x}_w = m_p\ddot{x}_p \textbf{ (2)}$$

$$N_y - m_p g = m_p\ddot{y}_p \textbf{ (3)}$$

Dónde tenemos que:

$$N_x = m_p(\ddot{x}_p + L\Psi cos\Psi - L\Psi^2 sen\Psi)$$

$$N_x = fuerza\ reactiva\ en\ la\ dirección\ de\ las\ X\ (N)$$

$$N_y = m_p g(L\ddot{\Psi} sen\Psi - L\dot{\Psi}^2 cos\Psi)$$

$$N_y = fuerza\ reactiva\ en\ la\ dirección\ Y\ (N)$$

Las aceleraciones no lineales $\ddot{x}_p$ y $\ddot{y}_p$ deben ser trasladadas desde las coordenadas X-Y del sistema a la coordenada rotacional del sistema. Las ecuaciones $\ddot{x}_p$ y $\ddot{y}_p$ quedan expresadas como:

$$X_p = -Lsen\Psi \textbf{ (4)}$$

$$Y_p = Lcos\Psi \textbf{ (5)}$$

Las expresiones anteriores se derivan para obtener las velocidades:

$$\dot{X}_p = -L\dot{\Psi} cos\Psi \textbf{ (6)}$$

$$\dot{Y}_p = -L\dot{\Psi} sen\Psi \textbf{ (7)}$$

Se volverán a derivar para obtener las aceleraciones:

$$\ddot{x}_p = -L\ddot{\Psi} cos\Psi + L\dot{\Psi}^2 sen\Psi \textbf{ (8)}$$

$$\ddot{Y}_p = -L\ddot{\Psi} sen\Psi - L\dot{\Psi}^2 cos\Psi \textbf{ (9)}$$

Para encontrar la inercia del cuerpo del robot, éste ha sido considerado como un cubo con una distribución de masa uniforme. Mediante la expresión clásica de la inercia se ha calculado:

$$J_p = \frac{1}{12} m_p w^2 + \frac{1}{3} m_p H^2 \textbf{ (10)}$$

Dónde: m_p: $masa\ robot$

w: $anchura\ robot$

H: $altura\ robot$

5.1.2. Modelo físico de las ruedas

En la siguiente figura se puede ver las fuerzas que actúan sobre las ruedas.

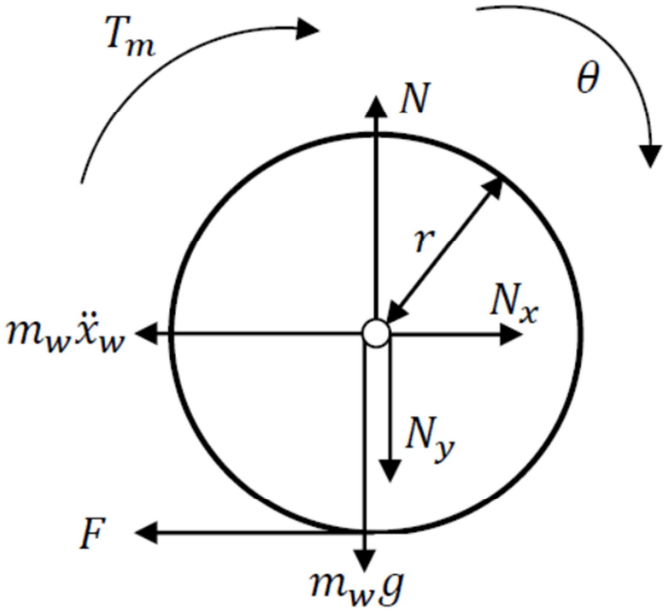


Figura 32: *Fuerzas que actúan sobre la rueda*

En este caso las ecuaciones son las siguientes:

La ecuación (11) es el momento en sentido horario referente al centro de masa de la rueda.

$$T_m + F_r = J_w\ddot{\theta} \quad \textbf{(11)}$$

Las ecuaciones (12) y (13) son las fuerzas que actúan en la dirección X e Y respectivamente:

$$N_x - m_w\ddot{x}_w - F = 0 \quad \textbf{(12)}$$

$$N - N_y - m_w g = m_w\ddot{Y}_w \quad \textbf{(13)}$$

Para trasladar desde el sistema de coordenadas X-Y al sistema de coordenadas rotacional:

$$\ddot{x}_w = \ddot{\theta}_r \quad \textbf{(14)}$$

$$\ddot{Y}_w = 0 \quad \textbf{(15)}$$

Y de igual forma que en el cuerpo del robot, pero utilizando la fórmula del círculo, la inercia se encuentra como:

$$J_w = \frac{2rsen2\alpha}{3\alpha} \quad \textbf{(16)}$$

Dónde: r: $Radio\ rueda$

α: $\acute{a}ngulo\ (el\ cual\ el\ punto\ de\ partida\ es\ 2\pi)$

5.1.3. Modelo eléctrico del circuito para los motores DC

Para el sistema se han utilizado dos pequeños motores de corriente continua (DC). Es necesaria una descripción matemática de los respectivos motores para proporcionar una relación de voltaje de entrada, la señal de control y el par entregado desde los motores. Mediante el método de las leyes de Kirchhoff para el circuito eléctrico, figura 3, se obtiene la siguiente ecuación.

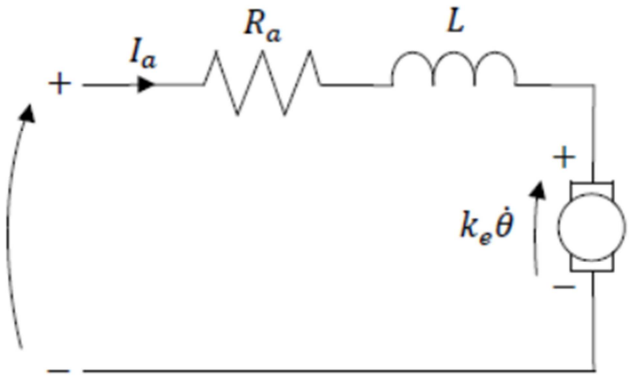


Figura 33: *Circuito eléctrico para los motores DC*

La ecuación (17) describe el motor:

$$U = R_a I_a + K_e \dot{\theta} + L \frac{di}{dt} \quad \textbf{(17)}$$

Ya que el tiempo de la constante eléctrica es mucho más pequeña que el de la mecánica, la inductancia puede ser despreciada quedando:

$$U = R_a I_a + K_e \dot{\theta} \quad \textbf{(18)}$$

La siguiente ecuación describe el par del eje:

$$T_m = nK_t I_a \quad \textbf{(19)}$$

El par en el eje necesita superar la inercia del motor, así como el amortiguamiento viscoso y la fricción del motor. Todos ellos son difíciles de estimar ya que son valores muy pequeños. Por lo tanto, la fricción y el amortiguamiento viscoso se pueden despreciar.

La expresión queda de la siguiente manera:

$$T_m = nK_t I_a \quad \textbf{(20)}$$

Las constantes del motor K_t y K_e deben ser encontradas. Si estos valores no salen en las hojas de especificaciones del motor, se pueden estimar utilizando los valores nominales del motor y haciendo simplificaciones.

Para K_t se tiene:

$$K_t = \frac{T_r}{I_{a,r}} \quad \textbf{(21)}$$

Donde T_r y $I_{a,r}$ es el par nominal y la corriente nominal de inducido. Para encontrar la constante K_e, se utiliza la ecuación (back-emf) de fuerza contraelectromotriz, y se define como:

$$V_e = wK_e \quad \textbf{(22)}$$

En el manual de especificaciones técnicas se obtiene la velocidad en rpm.

$$V_e = \frac{N \cdot 60}{2\pi} K_e \quad \textbf{(23)}$$

Y la ecuación para K_e:

$$k_e = \frac{(V_r - I_{a,r}R)60}{2\pi \cdot N_r} \quad \textbf{(24)}$$

Para la velocidad nominal, tensión y corriente. Dado que ambas constantes del motor eran estimaciones, éstas fueron las incertidumbres que pueden tener diferentes efectos en el comportamiento del robot, ya que afecta directamente al par.

Dado que los motores deben ser controlados a través de PWM, es decir por voltaje controlado, la corriente es eliminada de la ecuación de par, y la ecuación correspondiente es; haciendo uso de la ecuación (18):

$$I_a = \frac{U - K_e\dot{\theta}}{R_a} \quad \textbf{(25)}$$

Finalmente la ecuación del par para cada motor es:

$$T_m = \frac{nK_tU}{R_a} - \frac{nK_eK_t\dot{\theta}}{R_a} \quad \textbf{(26)}$$

5.1.4. Modelo linealizado

El modelo de estado queda definido como:

$$X = \begin{bmatrix} \theta \\ \Psi \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix}$$

Dónde:

θ : Ángulo rueda

Ψ : Ángulo péndulo

$\dot{\theta}$: Velocidad angular rueda

$\dot{\Psi}$: Velocidad angular péndulo

El modelo de estado linealizado será:

$$\dot{X} = Ax + Bu \quad (27)$$

$$Y = Cx + Du \quad (28)$$

Dónde:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_{32} & a_{33} & 0 \\ 0 & a_{42} & a_{43} & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ B_{13} \\ B_{14} \end{bmatrix}$$

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

$$a_{32} = \frac{gL^2 r m^2{}_p}{-L^2 r^2 m^2{}_p + J_p(J_w + r^2(m_p + m_w))}$$

$$a_{33} = \frac{-2nJ_p K_e K_t - 2LnrK_e K_t m_p}{(-L^2 r^{2m^2{}_p} + J_p\left(J_w + r^2(m_p + m_w)\right))R_a}$$

$$a_{42} = \frac{gLJ_w m_p R_a + Lr^2 m_p(gm_p + gm_w)R_a}{(-L^2 r^2 m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

$$a_{43} = \frac{-2nJ_w K_e K_t - 2nrK_e K_t((L + r)m_p + rm_w}{(-L^2 r^2 m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

$$B_{13} = \frac{2nJ_pK_t + 2LnrK_tm_p}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

$$B_{14} = \frac{2nJ_wK_t + 2nrK_t((L + r)m_p + rm_w}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

Teniendo este modelo de estado, se pueden encontrar los polos del sistema, es decir los autovalores de la matriz [A], y ver como el sistema sin ningún tipo de controlador, será un sistema totalmente inestable.

En bucle abierto los polos del sistema son los siguientes:

$$eig(A) = [0 \quad -9.2935 \quad 8.9680 \quad -0.1846]$$

Se obtiene un polo en la parte real positiva, lo que automáticamente determina que el sistema es inestable, se necesitará un modelo de control.

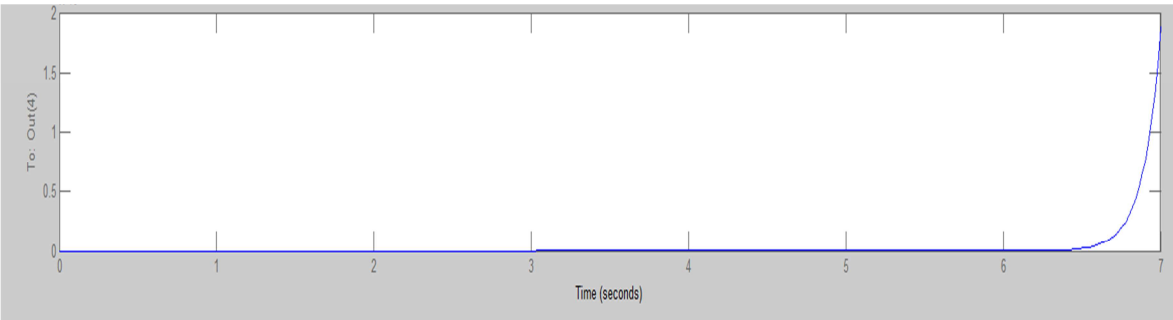


Figura 34: *Respuesta temporal del ángulo de giro Ψ del robot.*

CAPÍTULO 6: ANÁLISIS Y DISEÑO DEL SISTEMA DE CONTROL EN EL ESPACIO DE ESTADOS

Un sistema moderno complejo posee muchas entradas y muchas salidas que se relacionan entre sí de una forma complicada. Para analizar un sistema de este tipo, es esencial reducir la complejidad de las expresiones matemáticas, además de recurrir a computadoras que realicen una gran parte de los cálculos pesados que son necesarios. El enfoque en el espacio de estados para el análisis de sistemas es el más conveniente desde este punto de vista.

Mientras la teoría de control convencional se basa en la relación entrada-salida, o función de transferencia, la teoría de control moderna se basa en la descripción de las ecuaciones de un sistema en términos de n ecuaciones diferenciales de primer orden, que se combinan en una ecuación diferencial vectorial de primer orden. El uso de la notación matricial simplifica enormemente la representación matemática de los sistemas de ecuaciones. El incremento en el número de variables de estado, de entradas o de salidas no aumenta la complejidad de las ecuaciones.

6.1. Análisis del control por realimentación de estados

Este capítulo aborda el análisis y el diseño de un sistema de control en el espacio de estados basado en el método de asignación de polos con observador de estado.

El método de asignación consiste en que se colocan los polos en lazo cerrado en posiciones deseadas. Dentro de la representación del sistema en el espacio de estados, también incluye aspectos necesarios como son la controlabilidad y la observabilidad.

6.1.1. Observadores de estado

En el método de asignación de polos para el diseño de sistemas de control, se supone que todas las variables de estado están disponibles para su realimentación. Sin embargo, en la práctica no todas las variables de estado están accesibles para poder realimentarse. Entonces, se necesita estimar las variables de estado que no están disponibles. La estimación de variables de estado no medibles se denomina normalmente observación.

Un dispositivo (o un programa de computador) que estima u observa las variables de estado se llama un observador de estado, o, simplemente, un observador. Si el observador de estado capta todas las variables de estado del sistema, sin importar si algunas están disponibles por medición directa, se denomina observador de estado de orden completo. Hay ocasiones en las que un observador de este tipo no es necesario, ya que sólo se requiere la observación de las variables de estado que no se miden, pero no de aquellas que también se miden directamente. Un observador que estima menos de n variables de estado, donde n es la dimensión del vector de estado, se denomina observador de estado de orden reducido o, simplemente, un observador de orden reducido. Si el observador de estado de orden reducido es el orden mínimo posible, se denomina observador de estado de orden mínimo u observador de orden mínimo.

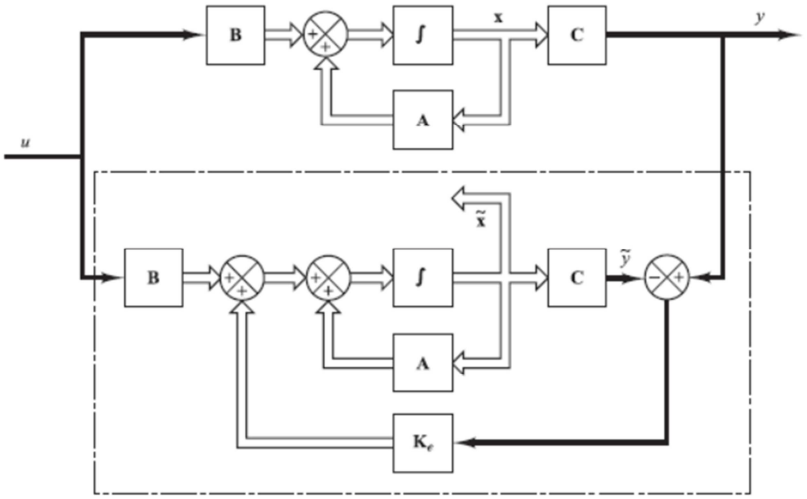


Figura 35: *Diagrama de bloque del sistema y del observador de estado de orden completo, cuando la entrada u y la salida y son escalares.*

Un observador de estado estima las variables de estado basándose en las mediciones de las variables de salida y de control. Por lo tanto, el concepto de observabilidad analizado más adelante juega un papel importante, ya que los observadores de estado pueden diseñarse si y sólo si se satisface la condición de observabilidad.

En el análisis que sigue de los observadores de estado, se utilizará la notación $\tilde{x}$ para designar el vector de estado observado. En muchos casos prácticos, el vector de estado observado $\tilde{x}$ se usa en la realimentación del estado para generar el vector de control deseado.

Sea el sistema definido mediante:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (29)$$

$$y = \mathbf{C}\mathbf{x} \quad (30)$$

El observador es un subsistema para reconstruir el vector de estado de la planta. El modelo matemático del observador es básicamente el mismo que el de la planta, salvo que se incluye un término adicional que contiene el error de estimación para compensar las imprecisiones en las matrices A y B y la falta del error inicial. El error de estimación o error de observación es la diferencia entre la salida medida y la salida estimada. El error inicial es la diferencia entre el estado inicial y el estado estimado inicial. De esta forma, se define el modelo matemático del observador como:

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e(y - \mathbf{C}\tilde{\mathbf{x}}) = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e y \quad (31)$$

donde $\tilde{x}$ es el estado estimado y $\mathrm{C}\tilde{x}$ es la salida estimada. Las entradas al observador son la salida "y" y la entrada de control "u". La matriz Ke, que se llama matriz de ganancia del observador, es una matriz de ponderación al término de corrección que involucra la diferencia entre la salida medida "y" y la salida estimada $\mathrm{C}\tilde{x}$. Este término corrige de forma continua la salida del modelo y mejora el comportamiento del observador.

6.1.2. Observador de orden mínimo

Dentro de los observadores de estado, los observadores de orden completo se diseñaron para reconstruir todas las variables de estado. En la práctica, algunas de las variables de estado se pueden medir con precisión. Tales variables de estado medidas con precisión no necesitan estimarse.

Supóngase que el vector de estado **x** es un vector de dimensión n y que el vector de salida **y** es un vector de dimensión m medible. Como las m variables de salida son combinaciones lineales de las variables de estado, no necesitan estimarse m

variables de estado, sino sólo n - m variables de estado. Así, el observador de orden reducido se vuelve un observador de (n - m)-ésimo orden. Tal observador de (n - m)-ésimo orden es el observador de orden mínimo. La siguiente figura muestra el diagrama de bloques de un sistema con un observador de orden mínimo.

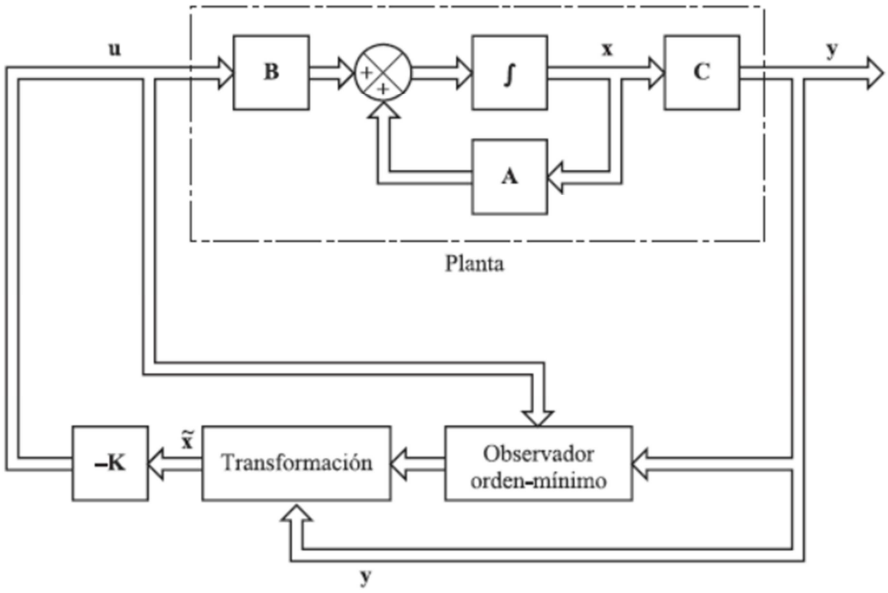


Figura 36: *Sistema de control realimentado con estado observado con un observador de orden mínimo.*

Para presentar la idea básica del observador de orden mínimo, sin complicaciones matemáticas innecesarias, se expondrá el caso en el que la salida es un escalar (es decir, m = 1) y se derivará la ecuación de estado para el observador de orden mínimo. Sea el sistema:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (32)$$

$$y = \mathbf{C}\mathbf{x} \quad (33)$$

donde el vector de estado **x** se particiona en dos partes x_a (un escalar) y x_b [un vector de dimensión (n-1)]. Aquí la variable de estado x_a es igual a la salida y de modo que se mide directamente, y x_b s la parte no medible del vector de estado. De esta forma, el estado particionado y las ecuaciones de salida son:

$$\left[\begin{array}{c} \dot{x}_a \\ \hline \dot{\mathbf{x}}_b \end{array}\right] = \left[\begin{array}{c|c} A_{aa} & \mathbf{A}_{ab} \\ \hline \mathbf{A}_{ba} & \mathbf{A}_{bb} \end{array}\right] \left[\begin{array}{c} x_a \\ \hline \mathbf{x}_b \end{array}\right] + \left[\begin{array}{c} B_a \\ \hline \mathbf{B}_b \end{array}\right] u \quad (34)$$

$$y = [1 \mid 0] \left[\begin{array}{c} x_a \\ \hline \mathbf{x}_b \end{array}\right] \quad (35)$$

donde A_{aa} = escalar

$\mathbf{A}_{ab}$ = matriz de $1 \times (n-1)$

$\mathbf{A}_{ba}$ = matriz de $(n-1) \times 1$

$\mathbf{A}_{bb}$ = matriz de $(n-1) \times (n-1)$

B_a = escalar

$\mathbf{B}_b$ = matriz de $(n-1) \times 1$

A continuación se presentará un método para diseñar un observador de orden mínimo. El procedimiento de diseño se simplifica si se utiliza la técnica de diseño desarrollada para el observador de estado de orden completo.

Compárese la ecuación de estado para el observador de orden completo con la del observador de orden mínimo. La ecuación de estado para el observador de orden completo es:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (36)$$

y la «ecuación de estado» para el observador de orden mínimo es:

$$\dot{\mathbf{x}}_b = \mathbf{A}_{bb}\mathbf{x}_b + \mathbf{A}_{ba}x_a + \mathbf{B}_b u \quad (37)$$

La ecuación de salida para el observador de orden completo es:

$$y = \mathbf{C}\mathbf{x} \quad (38)$$

y la «ecuación de salida» para el observador de orden mínimo es:

$$\dot{x}_a - A_{aa}x_a - B_a u = \mathbf{A}_{ab}\mathbf{x}_b \quad (39)$$

El diseño del observador de orden mínimo se realiza del modo siguiente. Primero, obsérvese que la ecuación del observador para el observador de orden completo se obtuvo a partir de la Ecuación (31), la cual se repite aquí:

$$\dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\tilde{\mathbf{x}} + \mathbf{B}u + \mathbf{K}_e y \quad (40)$$

Así, haciendo las sustituciones de la Tabla 4 en la Ecuación (40), se obtiene:

$$\dot{\tilde{\mathbf{x}}}_b = (\mathbf{A}_{bb} - \mathbf{K}_e\mathbf{A}_{ab})\tilde{\mathbf{x}}_b + \mathbf{A}_{ba}x_a + \mathbf{B}_b u + \mathbf{K}_e(\dot{x}_a - A_{aa}x_a - B_a u) \quad (41)$$

donde la matriz de ganancias del observador de estado K*e* es una matriz de (*n*-1)x1. En la Ecuación (41), obsérvese que para estimarse $\tilde{X}b$ necesita la derivada de X_a.

Tabla 4. *Lista de sustituciones necesarias para escribir la ecuación del observador para el observador de estado de orden mínimo.*

Observador de estado de orden completo	Observador de estado de orden mínimo
$\tilde{\mathbf{x}}$	$\tilde{\mathbf{x}}_b$
$\mathbf{A}$	$\mathbf{A}_{bb}$
$\mathbf{B}u$	$\mathbf{A}_{ba}x_a + \mathbf{B}_b u$
y	$\dot{x}_a - A_{aa}x_a - B_a u$
$\mathbf{C}$	$\mathbf{A}_{ab}$
$\mathbf{K}_e$ (matriz $n \times 1$)	$\mathbf{K}_e$ [matriz $(n-1) \times 1$]

La ecuación característica para el observador de orden mínimo se expresa de la siguiente manera:

$$|s\mathbf{I} - \mathbf{A}_{bb} + \mathbf{K}_e\mathbf{A}_{ab}| = (s - \mu_1)(s - \mu_2)\cdots(s - \mu_{n-1}) \quad (41)$$

Donde $\mu_1, \mu_2, \ldots, \mu_{n-1}$ son valores propios deseados para el observador de orden mínimo. La matriz de ganancias del observador K_e se determina seleccionando primero los valores propios deseados para el observador de orden mínimo (es decir, colocando las raíces de la ecuación característica, Ecuación (41), en las posiciones deseadas) y después empleando el procedimiento desarrollado para el observador de orden completo con las modificaciones adecuadas.

6.1.3. Diseño control por observador de orden mínimo

En este caso se ha realizado siguiendo la teoría anterior, un diseño de control por observador de orden mínimo, donde únicamente se va a tomar como variable de estado medible el ángulo del péndulo, dejando las otras variables de estado a observar.

El ángulo del péndulo sería una variable fácil de medir en tiempo real mediante la IMU MPU-6050, mientras que para las otras variables de estado del sistema al ser observadas, se evita tener que instalar más sensores para ser medidas, con lo cual se simplifica el sistema a nivel hardware.

Según el modelo de estado presentado anteriormente tenemos:

$$X = \begin{bmatrix} \theta \\ \Psi \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix}$$

- θ: Ángulo rueda
- Ψ: Ángulo péndulo
- $\dot{\theta}$: Velocidad angular rueda
- $\dot{\Psi}$: Velocidad angular péndulo

En este caso se puede simplificar el modelo de estado inicial, ya que el ángulo θ de las ruedas no es de interés y puede ser suprimida de la ecuación de estado del controlador. Las nuevas variables de estado eliminando θ queda:

$$X = \begin{bmatrix} \Psi \\ \dot{\theta} \\ \dot{\Psi} \end{bmatrix}$$

Esto produce un nuevo modelo de espacio de estados con 3 estados.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x}$$

Dónde:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ B_{21} \\ B_{31} \end{bmatrix} \quad C = [1 \quad 0 \quad 0] \quad D = [0]$$

$$a_{21} = \frac{gL^2rm^2{}_p}{-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w))} \qquad a_{22} = \frac{-2nJ_pK_eK_t - 2LnrK_eK_tm_p}{(-L^2r^{2m^2{}_p} + J_p\left(J_w + r^2(m_p + m_w)\right))R_a}$$

$$a_{31} = \frac{gLJ_wm_pR_a + Lr^2m_p(gm_p + gm_w)R_a}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a} \qquad a_{32} = \frac{-2nJ_wK_eK_t - 2nrK_eK_t((L + r)m_p + rm_w}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

$$B_{21} = \frac{2nJ_pK_t + 2LnrK_tm_p}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a} \qquad B_{22} = \frac{2nJ_wK_t + 2nrK_t((L + r)m_p + rm_w}{(-L^2r^2m^2{}_p + J_p(J_w + r^2(m_p + m_w)))R_a}$$

Sustituyendo los valores en las expresiones de las matrices anteriores tenemos:

$$A=\begin{bmatrix} 0 & 0 & 1 \\ 204.6619 & -0.5101 & 0 \\ 83.2847 & -0.1324 & 0 \end{bmatrix} \qquad B=\begin{bmatrix} 0 \\ 7.1057 \\ 1.8443 \end{bmatrix} \qquad C=[1 \quad 0 \quad 0] \qquad D=[0]$$

Por lo tanto tenemos:

- Variables de estado del sistema:

$$\begin{cases} \Psi\text{: Ángulo péndulo} \\ \dot{\theta}\text{: Velocidad angular rueda} \\ \dot{\Psi}\text{: Velocidad angular péndulo} \end{cases}$$

- Variables de estado a observar:

$$\begin{cases} \dot{\theta}\text{: Velocidad angular rueda} \\ \dot{\Psi}\text{: Velocidad angular péndulo} \end{cases}$$

Es decir, la única variable no observable y por lo tanto medible, será la del ángulo del péndulo o cuerpo del robot, de aquí que se denomine observador de orden mínimo.

Siguiendo los pasos del algoritmo de control del observador de orden mínimo queda:

n; (número de variables que tiene el sistema) (Orden del sistema)

m; (número de estados medidos)

i = (n-m); (estados observados) (Orden del Observador)

i equivale al orden de un vector del observador que llamaremos L, y quedará expresado matricialmente como (n-m x1).

Con lo cual tenemos:

n=3; las tres variables de estado que forman el sistema ($\Psi, \dot{\theta}, \dot{\Psi}$)

m=1; la única variable de estado medida (Ψ)

i = (n-m) = (3-1) = 2; Orden del observador

Orden del vector L del observador (n-mx1) = (2x1) = $\begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$ = L

Mediante la fórmula (34) y (35) se obtendrá las matrices equivalentes para el sistema de control de observador de orden mínimo.

$$\begin{bmatrix} \dot{x}_a \\ \hline \dot{\mathbf{x}}_b \end{bmatrix} = \left[\begin{array}{c|c} A_{aa} & \mathbf{A}_{ab} \\ \hline \mathbf{A}_{ba} & \mathbf{A}_{bb} \end{array}\right] \begin{bmatrix} x_a \\ \hline \mathbf{x}_b \end{bmatrix} + \begin{bmatrix} B_a \\ \hline \mathbf{B}_b \end{bmatrix} u$$

$$y = [1 \;\vdots\; \mathbf{0}] \begin{bmatrix} x_a \\ \hline \mathbf{x}_b \end{bmatrix}$$

Equivalencias: A= A_{bb}, B = A_{ba}, C= A_{ab}, D=A_{aa}.

Por lo tanto las dimensiones de las submatrices queda:

$A_{aa}(mxm)$ (1x1) $B_a(mx1)$ (1x1)

$A_{ab} = (mxn - m)$ (1x2) $B_b(n - mx1)$ (2x1)

$A_{ba}(n - mxm)$ (2x1) $C_a(1xm)$ (1x1)

$A_{bb}(n - mxn - m)$ (2x2) $C_b(1xn - m)$ (1x2)

Las matrices resultantes son:

A = $A_{bb} = \begin{bmatrix} -0.5101 & 0 \\ -0.1324 & 0 \end{bmatrix}$ **B** = $A_{ba} = \begin{bmatrix} 204.6619 \\ 83.2847 \end{bmatrix}$ **C** = $A_{ab} = [0 \quad 1]$ **D** = $A_{aa} = [0]$

$$B_a = [0] \qquad B_b = \begin{bmatrix} 7.1057 \\ 1.8443 \end{bmatrix}$$

Ahora se calculará el vector de control L del observador mínimo, aunque antes habrá que diseñar el regulador (planta) del sistema, para poder después fijar los polos del observador con una dinámica más rápida con los polos de la planta del sistema.

- Diseño del regulador de la planta.

Siendo la planta: A= $\begin{bmatrix} 0 & 0 & 1 \\ 204.6619 & -0.5101 & 0 \\ 83.2847 & -0.1324 & 0 \end{bmatrix}$

Mediante MATLAB conocemos los polos del sistema con la instrucción eig(A).

Los polos de la planta son los siguientes:

$$\text{eig (A)} = [-9.2935 \quad 8.9680 \quad -0.1846]$$

Evidentemente al no haber ningún control, la planta del sistema sale totalmente inestable.

Se realizará un control de la planta con las siguientes especificaciones:

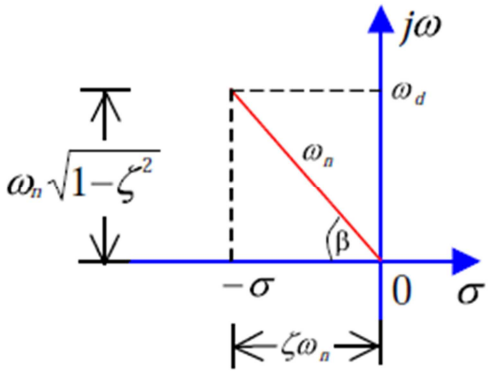
$$T_s(2\%) = 0.5s; \quad \xi = 0.7;$$

Situando los polos dominantes de lazo cerrado en el plano S de forma que el sistema tenga una respuesta transitoria con un tiempo de estabilización de 20 s y un coeficiente de amortiguamiento igual a 0.7.

$$T_s\,(2\%) = \frac{4}{\xi \cdot \omega_n} = 0.5 \longrightarrow 4 = 0.5 \cdot (0.7 \cdot \omega_n) \longrightarrow \omega_n = {}^{80}/_{7}$$

$$\sigma = \xi \cdot \omega_n = 0.7 \cdot {}^{80}/_{7} = 8$$

$$\omega_d = \omega_n\sqrt{1-\xi^2} = 8.16$$



Los polos dominantes en esas condiciones son:

$$S_1 = (-8 + 8.16j) \quad S_2 = (-8 - 8.16j)$$

Al ser un sistema de tercer orden, se deberá de poner otro polo. El polo se situará cinco veces más lejano del eje imaginario, para que no afecte mucho a la dinámica del sistema.

$$S_3 = -40$$

Por lo tanto la ecuación característica del sistema queda:

$$(s + 8 + 8.16j)(s + 8 - 8.16j)(s + 40) = 0$$

Quedando el polinomio: $s^3 + 56s^2 + 770.6s + 5223.424 = 0$

Finalmente se obtiene mediante la instrucción (acker) en MATLAB el vector K de ganancias de realimentación del vector de estado del sistema en anillo cerrado.

$$K = [462.9674 \quad -24.4427 \quad 124.2570]$$

Una vez se tiene el diseño del regulador de la planta, ya se puede realizar el diseño del observador.

Los polos del observador se fijarán más rápidos que los polos del regulador de la planta, ya que se necesita que se anticipe a éste siendo su dinámica más rápida. Se situarán de tal manera que el observador sea cinco veces más rápido, por lo tanto:

Polos 5 veces más rápidos:

$$S_1 = (-40 + 40.8j) \quad S_2 = (-40 - 40.8j)$$

Esos serán los polos deseados del observador, y se sabe que serán cinco veces más rápida su respuesta ya que los polos se han situado cinco veces más lejanos del eje imaginario con respecto a los polos de la planta.

Por lo tanto la ecuación característica del sistema queda:

$$(s + 40 + 40.8j)(s + 40 - 40.8j) = 0$$

Quedando el polinomio: $s^2 + 80s + 3264.64 = 0$ (ec1)

Ahora se calculará el polinomio característico del observador según la expresión:

$$|SI - A_{bb} + LA_{ab}| = 0$$

Quedando el siguiente polinomio con L_1 y L_2 como incógnitas.

$$s^2 + (0.5101 + L_2)s + 0.5101L_2 - 0.1324L_1 = 0 \quad \text{(ec2)}$$

Finalmente mediante la (ec1) y la (ec2) obtendremos el vector L del observador.

$$80 = 0.5101 + L_2$$

$$3264.64 = 0.5101L_2 - 0.1324L_1$$

Vector L del observador: L = $\begin{bmatrix} L_1 \\ L_2 \end{bmatrix} = \begin{bmatrix} -24351.15 \\ 79.50 \end{bmatrix}$

Y así quedaría diseñado el modelo de estado por observador de orden mínimo, únicamente midiendo el ángulo de giro del robot y observando el resto de variables.

CAPÍTULO 7:

CONTROL PID

7.1. Introducción

Un controlador PID es un mecanismo de control por realimentación ampliamente usado en sistemas de control industrial. Este calcula la desviación o error entre un valor medido y un valor deseado. El control PID es con diferencia el algoritmo de control más común, siendo utilizado en el 95% de los lazos de control que existen en la industria.

El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional depende del error actual. El Integral depende de los errores pasados y el Derivativo es una predicción de los errores futuros. La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control.

Considerando un lazo de control de una entrada y una salida (SISO) de un grado de libertad:

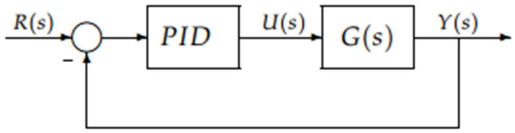


Figura 37: *Diagrama de bloques.*

7.2. Acción proporcional

• La parte proporcional consiste en el producto entre la señal de error y la constante proporcional K_p para lograr que el error en estado estacionario se aproxime a cero.

La fórmula del proporcional está dada por: $P_{sal} = K_p e(t)$

7.3. Acción integral

• El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el modo proporcional. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El error es integrado, lo cual tiene la función de promediarlo o sumarlo por un período determinado; Luego es multiplicado por una constante Ki.

La fórmula del integral está dada por: $I_{sal} = K_i \int_0^t e(t)\,dt$

7.4. Acción derivativa

• La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error; (si el error es constante, solamente actúan los modos proporcional e integral). El error es la desviación existente entre el punto de medida y el valor consigna, o "Set Point". La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente.

Se deriva con respecto al tiempo y se multiplica por una constante D y luego se suma a las señales anteriores (P+I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor derivativa corresponde a un cambio más rápido y el controlador puede responder acordemente.

La fórmula del derivativo está dada por: $D_{sal} = K_d \frac{d_e}{d_t}$

La salida de estos tres términos, el proporcional, el integral, y el derivativo son sumados para calcular la salida del controlador PID. Definiendo y (t) como la salida del controlador, la forma final del algoritmo del PID es:

$$y(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{d_e}{d_t}$$

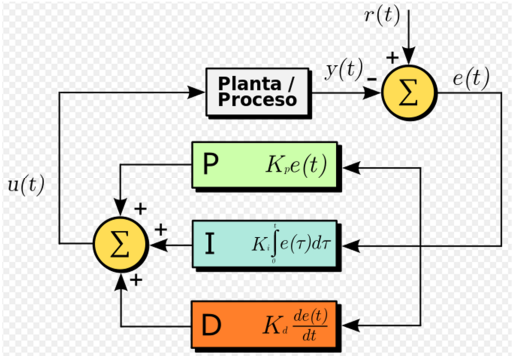


Figura 38: *Diagrama de bloques de un controlador PID en un lazo realimentado.*

7.5. Ajuste de parámetros del PID

El objetivo de los ajustes de los parámetros PID es lograr que el bucle de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones; se tiene que lograr la mínima integral de error. Si los parámetros del controlador PID (la ganancia del proporcional, integral y derivativo) se eligen incorrectamente, el proceso a controlar puede ser inestable.

Ajustar un lazo de control significa ajustar los parámetros del sistema de control a los valores óptimos para la respuesta del sistema de control deseada. El comportamiento óptimo ante un cambio del proceso o cambio del "setpoint" varía dependiendo de la aplicación. Generalmente, se requiere estabilidad ante la respuesta dada por el controlador, y este no debe oscilar ante ninguna combinación de las condiciones del proceso y cambio de "setpoints".

7.6. Obtención de parámetros del PID

Hay varios métodos para ajustar un lazo de PID. En este caso los parámetros de ajuste del controlador PID utilizado para estabilizar el sistema del robot, se han encontrado gracias a una herramienta de Simulink (PID tune).

Una vez definida la función de transferencia del modelo de la planta que describe nuestro sistema, la herramienta (PID tune) encuentra los parámetros de ajuste del PID que más se ajustan a la estabilización de dicha planta.

Inicialmente mediante el modelo de estado que tenemos ya definido obtenemos la función de transferencia del sistema.

$$G(s) = \frac{1.8443s^2}{s^4 + 0.5101s^3 - 83.2847s^2 - 15.3868s}$$

El siguiente paso es construir el diagrama de bloques en el entorno de Simulink.

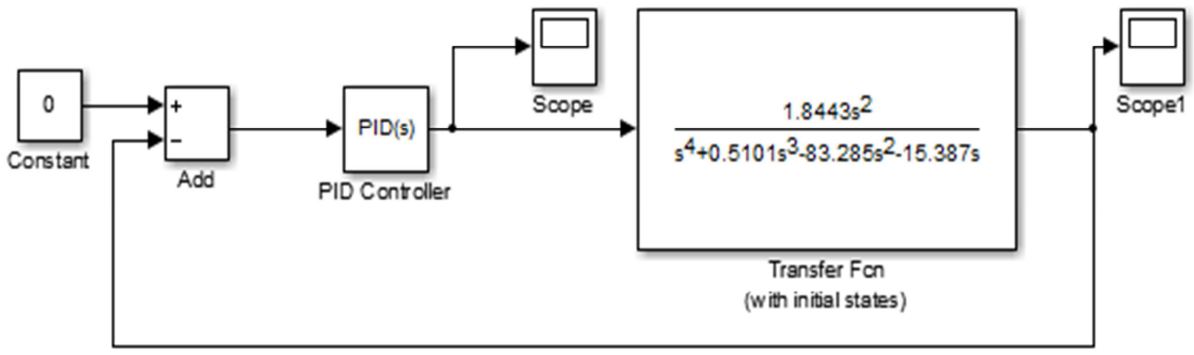


Figura 39: *Diagrama de bloques del sistema. (Controlador PID + Planta)*

En la figura 39, está representado el diagrama de bloques del sistema, con la planta y su regulador PID que será el encargado de estabilizar el sistema. La entrada al sistema es un valor constante de valor igual a cero, que representa la consigna, es decir al valor en grados de inclinación que el robot debe tener. Los cero grados se entiende que serán los que leerá la IMU MPU-6050 cuando el robot esté completamente vertical sobre sus ruedas respecto a la superficie de apoyo. La realimentación del sistema o lo que es lo mismo, la salida de la planta vuelve al sumador y se resta al valor de entrada y la salida de este sumador indica el error que existe. Este error es el que entra al controlador, encargándose éste de que ese error vaya disminuyendo hasta llegar a convertirse en cero.

Dentro del bloque PID Controller tenemos la oportunidad de encontrar los parámetros de ajuste de forma automática. Dentro de las opciones de este bloque se escoge la forma paralela del controlador PID, debido a que el algoritmo PID de Arduino trabaja de esta forma.

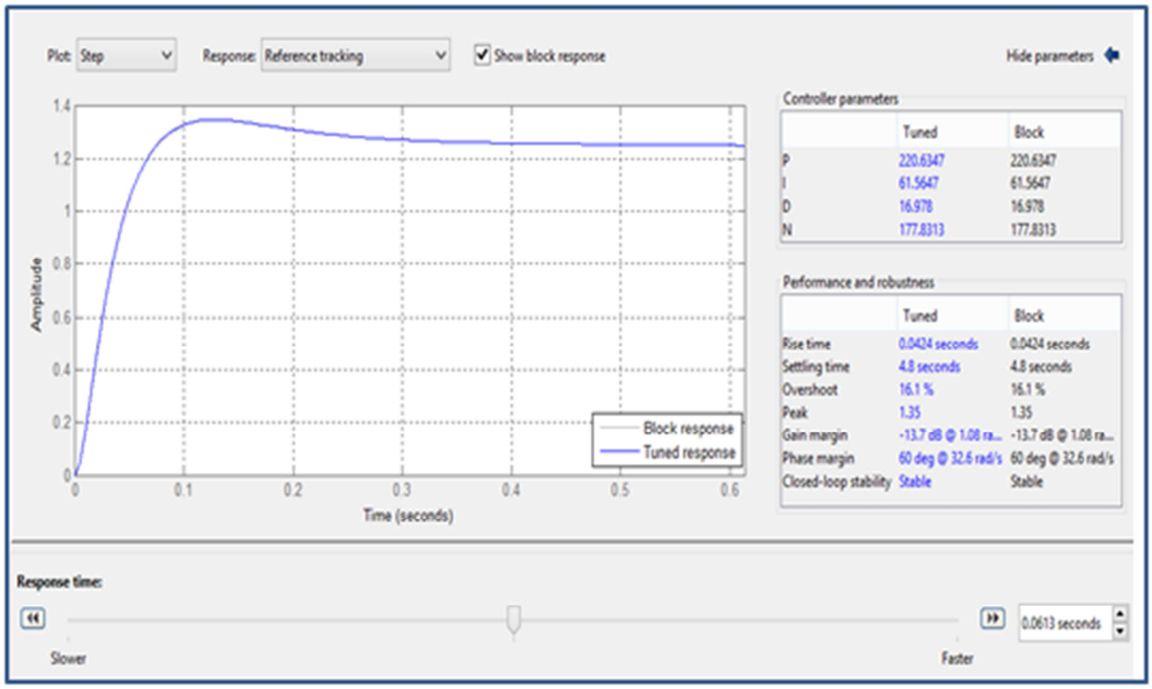


Figura 40: *PID Tuner.*

Aquí nos presenta los diferentes parámetros del controlador PID, para un tiempo de respuesta determinado. Si se determina un tiempo de respuesta mucho menor al que hay por defecto producirá un aumento de la constante proporcional es decir un aumento en la ganancia, para tratar de llegar con mayor rapidez a la estabilización del sistema.

A continuación se ven algunas gráficas sobre la respuesta del sistema en el tiempo, con los parámetros de ajuste ya introducidos en el controlador PID.

En el siguiente gráfico, se ha partido desde una condición inicial de +5 grados de inclinación con respecto al cero (posición vertical robot), y se ve como rápidamente el sistema se estabiliza en el cero o posición de equilibrio, manteniéndose en el tiempo.

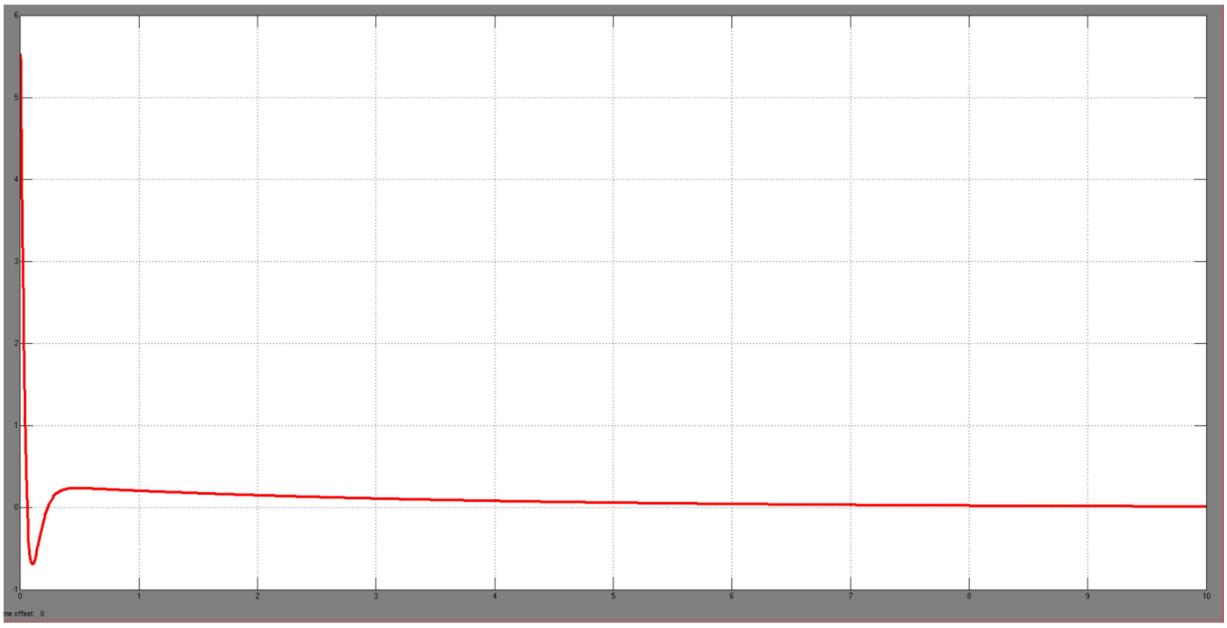


Figura 41: *Simulación sistema. (Condición inicial +5 grados).*

Mientras que en este caso se ha partido desde una posición inicial de -3 grados de inclinación respecto a la posición de equilibrio.

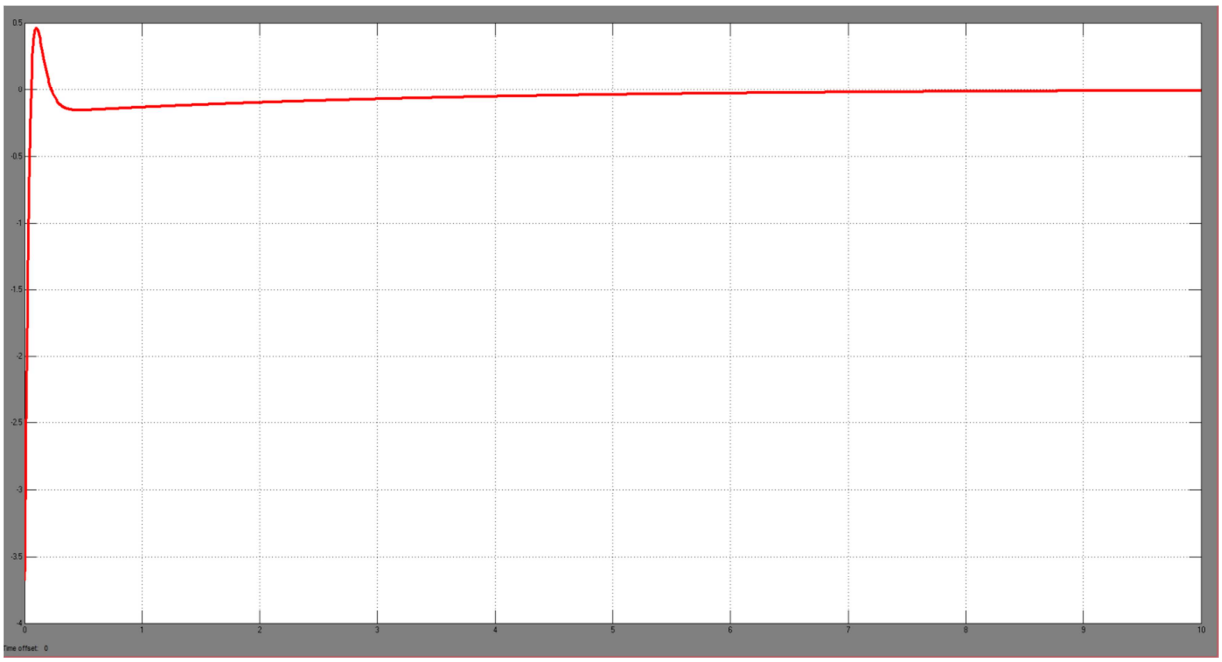


Figura 42: *Simulación sistema. (Condición inicial -3 grados).*

CAPÍTULO 8: PROGRAMA

8.1. Código fuente

En el capítulo 3, se especificó que el programa principal o código fuente se ha realizado en el software de Arduino.

A continuación se presentará el código realizado, con una breve explicación.

```
//Cargar librerias
#include <PID_v1.h>
#include <Wire.h>


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

//Ratios de conversion
#define A_R 16384.0
#define G_R 131.0

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

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

//Angulos
float Acc[2];
float Gy[2];
float Angle[2];

// pines del driver
int IN1=9;
int IN2=10;
int IN3=6;
int IN4=5;
int val;
```

Las dos primeras líneas incluyen la librería PID necesaria para el control y Wire necesaria para la interacción vía protocolo de comunicación I^2C.

#define MPU 0x68 es la dirección I^2C de la IMU.

Los ratios de conversión son los especificados en la documentación. Se deberá dividir los valores que dan el Giroscopio y el Acelerómetro entre estas constantes para obtener un valor coherente. Por otro lado, RAD_A_DEG es la conversión de radianes a grados.

La IMU da los valores en enteros de 16 bits. Como Arduino los guarda en menos bits, hay que declarar las variables que almacenarán los enteros provenientes de la IMU como un tipo de enteros especiales. int16_t AcX, AcY, AcZ, GyX, GyY son, pues, los raw_values de la IMU (valores en bruto).

Seguidamente hay tres arrays (Acc [], Gy [], Angle []) que guardan el ángulo X, Y del Acelerómetro, el Giroscopio y el resultado del Filtro respectivamente. [0] se corresponde a X. [1] a Y. Después se declaran los pines del driver L298N.

```
//CONFIGURACIÓN DEL LAZO PID
//Definir las variables que vamos a conectar
double sp,setpoint, input, output, in, out;

//Definir los parametros de ajuste
double Kp=220.6347, Ki=61.5647, Kd=16.978;

//Especificar los vinculos y los parámetros de ajuste
PID robot(&in,&out,&sp,Kp,Ki,Kd,DIRECT);
```

Se definen las variables del controlador PID, y se introducen los parámetros de ajuste calculados mediante PID Tune en Simulink.

PID () crea el controlador PID vinculado a la especificada entrada, salida y consigna. El algoritmo PID está en forma paralela.

```
void setup()
{

Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(9600);


//Configuración pines del driver
pinMode (IN1,OUTPUT); //Input 1 conectada al pin 9
pinMode (IN2,OUTPUT); //Input 2 conectada al pin 10
pinMode (IN3,OUTPUT); //Input 3 conectada al pin 6
pinMode (IN4,OUTPUT); //Input 4 conectada al pin 5


//CONFIGURACION LAZO PID
delay(3000);
Serial.println(sp);
setpoint = Angle[1]; //variable entrada PID
sp=0;      //consigna PID
robot.SetMode(AUTOMATIC);  //enciende el PID
robot.SetSampleTime(20);
robot.SetOutputLimits(-255,255);
}
```

La función setup es la siguiente:

Se inicia la comunicación por I^2C con el dispositivo MPU, y se "actica" enviando el comando 0. Posteriormente se inicia el puerto serie para ver los resultados.

En la configuración de los pines del driver, se establecen como salida.

En la configuración del Lazo PID, inicialmente se establece un retardo de 3 ms para que lea el valor del sensor y pueda hacer el control del ángulo.

Se guarda la variable Angle [1] en la variable setpoint, se inicializa la variable sp a 0.

Robot.SetSampleTime(20); se modifica los valores de SetSampletime de la librería a 20 ms para conseguir una lectura más rápida, ya que por defecto trabaja a 100 ms.

Robot.SetMode(AUTOMATIC); la librería estará cada 20 ms haciendo la operación.

Robot.SetOutputLimits(-255,255); se crean estos límites para la salida PWM en ambos sentidos.

```
void loop()
{
   //Leer los valores del Acelerometro de la IMU
   Wire.beginTransmission(MPU);
   Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6 registros
   AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   AcY=Wire.read()<<8|Wire.read();
   AcZ=Wire.read()<<8|Wire.read();

    //Se calculan los angulos Y
   Acc[1] = atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2) + pow((AcZ/A_R),2)))*RAD_TO_DEG;


   //Leer los valores del Giroscopio
   Wire.beginTransmission(MPU);
   Wire.write(0x43);
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,4,true); //A diferencia del Acelerometro, solo se piden 4 registros
   GyX=Wire.read()<<8|Wire.read();
   GyY=Wire.read()<<8|Wire.read();

   //Calculo del angulo del Giroscopio
   Gy[1] = GyY/G_R;

   //Aplicar el Filtro Complementario
   Angle[1] = 0.98 *(Angle[1]+Gy[1]*0.010) + 0.02*Acc[1];
```

El void loop suele ser la parte del programa más compleja. En él se leen y se guardan los datos de la IMU, se calcula el ángulo, se aplica el filtro complementario y se configura el control de lazo PID.

```
  //Configuracion Lazo de Control PID
  input= Angle[1];
  val = setpoint - input;
  in = val;
  robot.Compute();
  if(abs(val)>0){
    if(val>0){
      sal=5;
      sald=10;
     digitalWrite(6,LOW);
     digitalWrite(9,LOW);
    }

    if(val<1)
    {
     sal=6;
     sald=9;
     digitalWrite(5,LOW);
     digitalWrite(10,LOW);
    }
    output = abs(out);
    analogWrite(sal,output);
    analogWrite(sald,output);
    }else{
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
    }

  //Mostrar los valores por consola
  Serial.print("Angle Y: "); Serial.print(Angle[1]); Serial.print("\n------------\n");

  delay(10); //Nuestra dt sera, pues, 0.010, que es el intervalo de tiempo en cada medida
 }
```

En input se guarda la lectura del ángulo calculado y filtrado.

val = Setpoint – input; se guarda la diferencia entre el valor deseado y nuestra entrada.

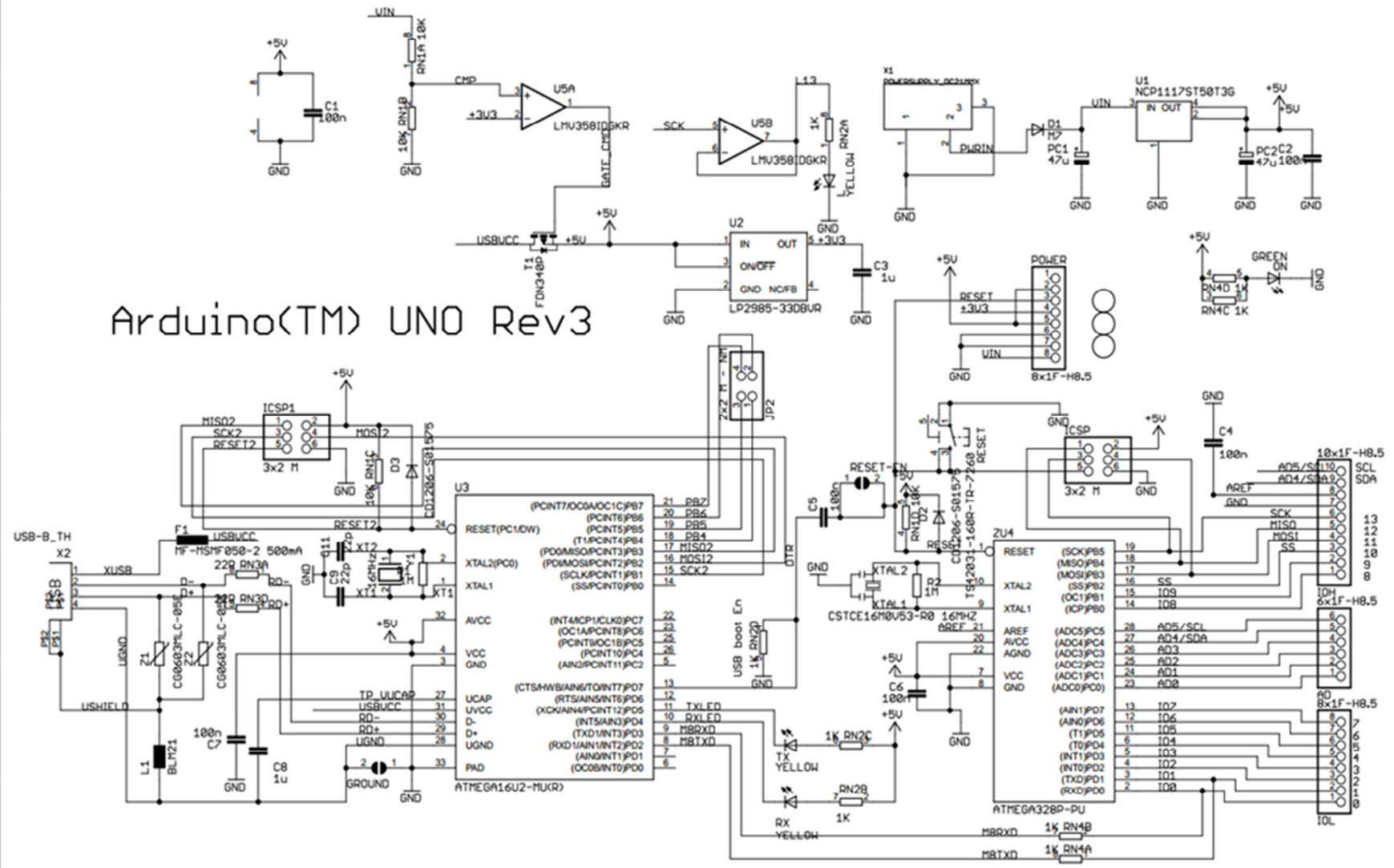
in = val; el valor de la variable val se guarda en la variable double in, la cual se irá a la librería robot.Compute(); para que realice la operación y genere una salida.

Con la instrucción if – else se controla el sentido del motor.

output = abs(out); se genera la variable salida, con el valor absoluto de out. Por el motivo que se ha programado los límites -255 a 255, por lo que cuando esté inclinado hacia un lado se enviará una señal negativa y ese detalle por cuestiones de programación no es posible, puesto que el microcontrolador no puede mandar ninguna salida PWM de valor negativo.

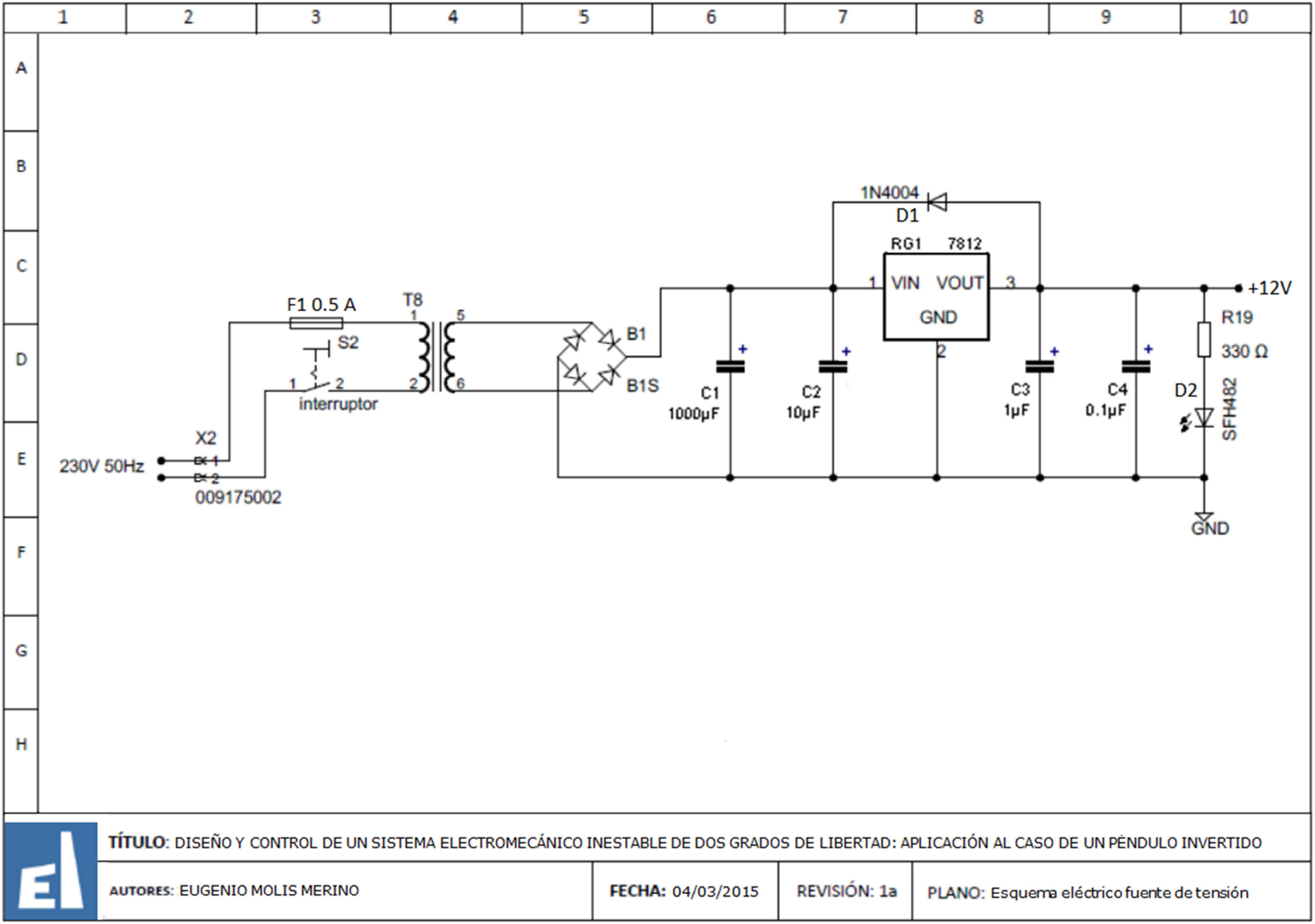
Finalmente se manda imprimir por pantalla en tiempo real el ángulo de giro.

CAPÍTULO 9: ESQUEMAS ELÉCTRICOS Y ESPECIFICACIONES



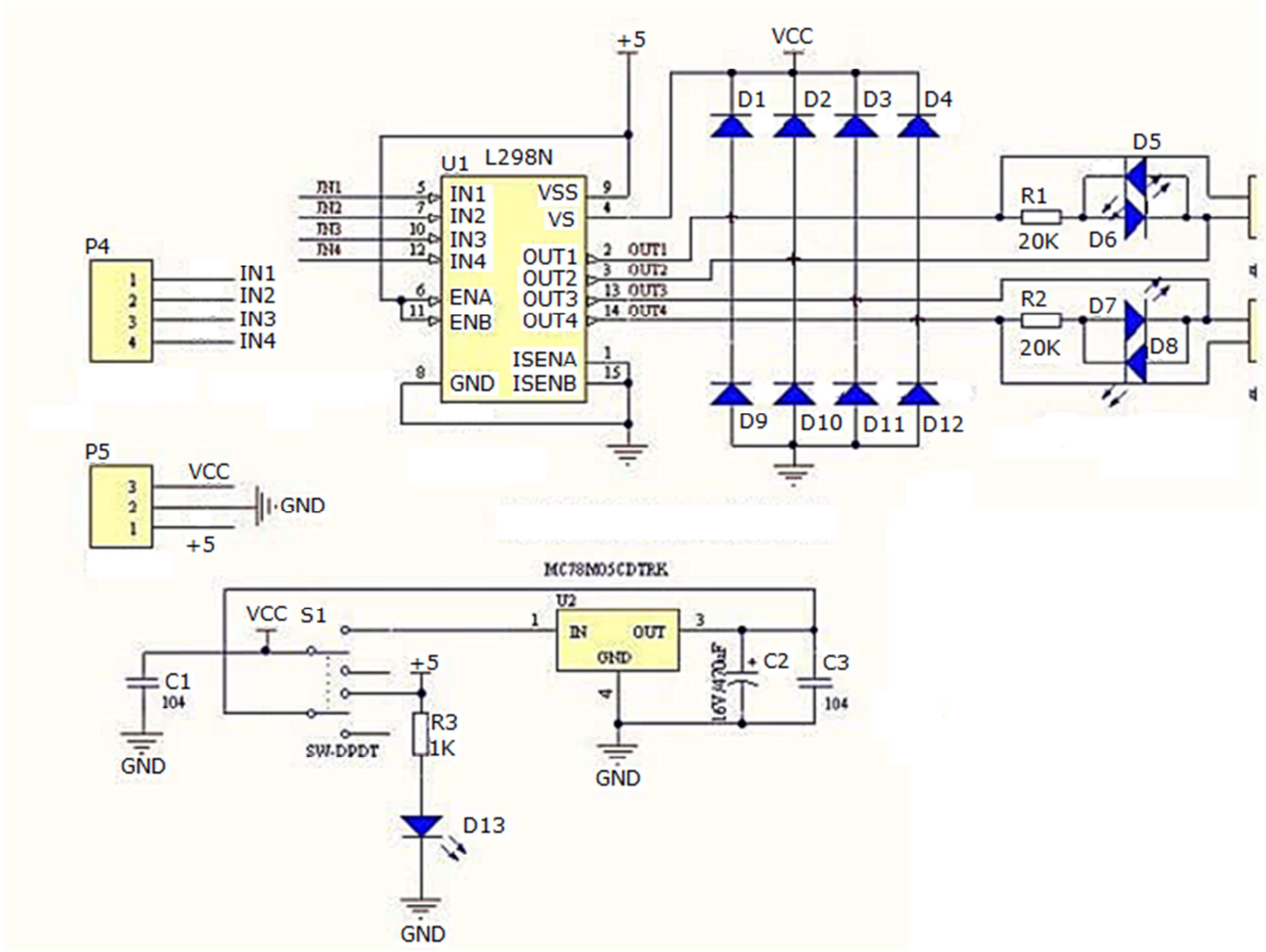
TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO

AUTORES: EUGENIO MOLIS MERINO	FECHA: 04/03/2015	REVISIÓN: 1a	PLANO: Esquema eléctrico Arduino UNO

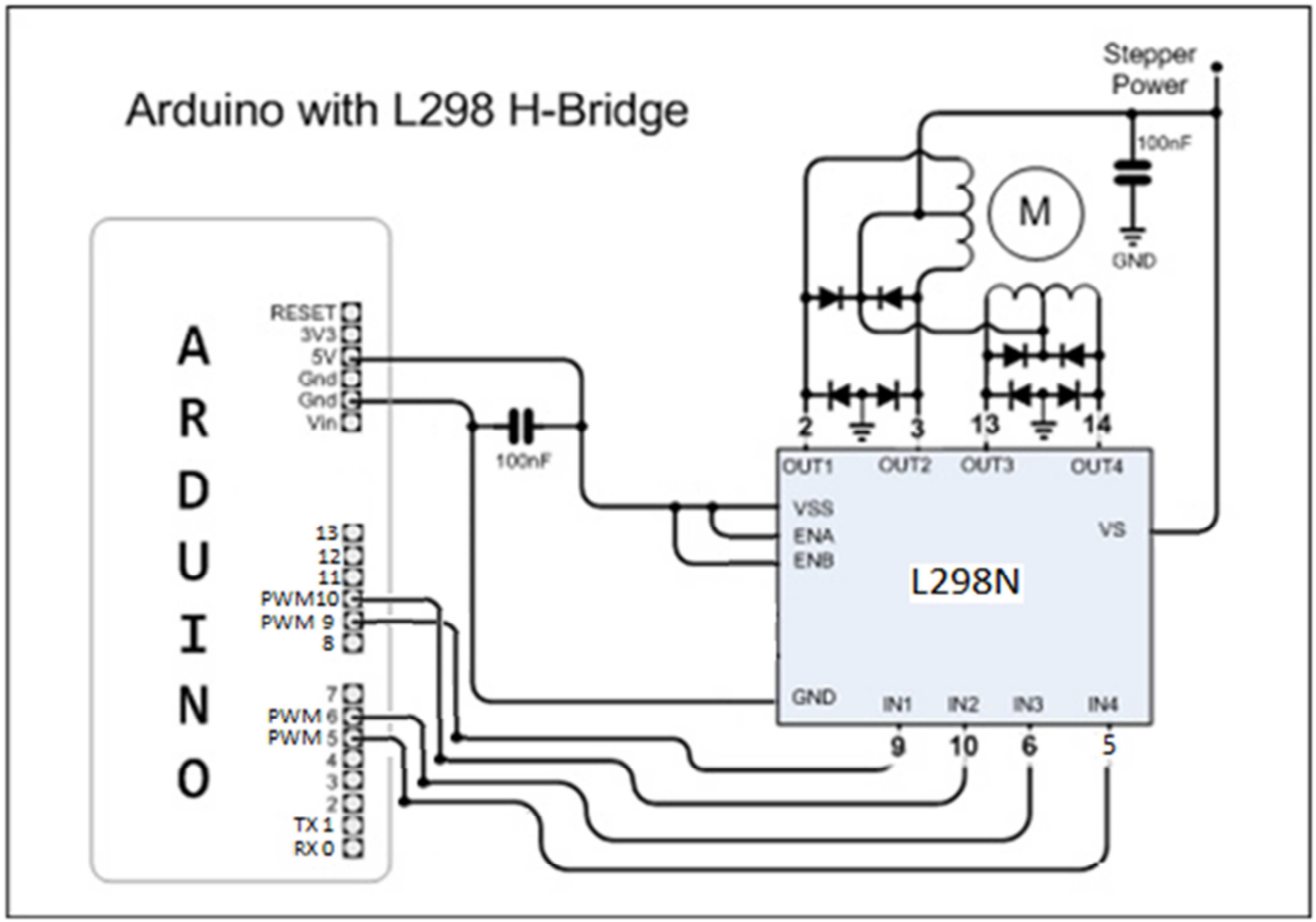


TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO

AUTORES: EUGENIO MOLIS MERINO	FECHA: 04/03/2015	REVISIÓN: 1a	PLANO: Esquema eléctrico fuente de tensión

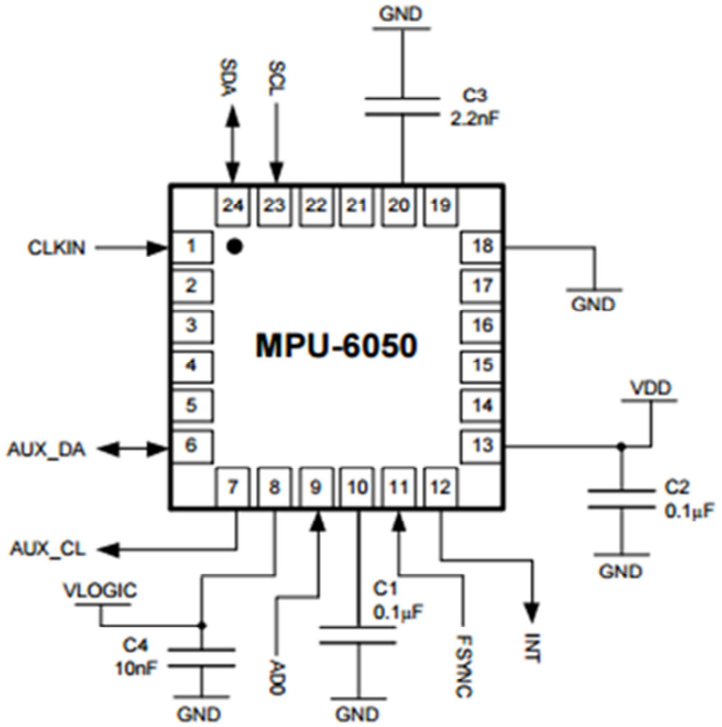


TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÈNDULO INVERTIDO			
AUTORES: EUGENIO MOLIS MERINO	**FECHA**: 15/03/2015	REVISIÓN: 1a	PLANO: Esquema eléctrico driver L298N



TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO

AUTORES: EUGENIO MOLIS MERINO	FECHA: 23/03/2015	REVISIÓN: 1a	PLANO: Esquema conexión Arduino + Driver L298N



Component	Label	Specification	Quantity
Regulator Filter Capacitor (Pin 10)	C1	Ceramic, X7R, 0.1µF ±10%, 2V	1
VDD Bypass Capacitor (Pin 13)	C2	Ceramic, X7R, 0.1µF ±10%, 4V	1
Charge Pump Capacitor (Pin 20)	C3	Ceramic, X7R, 2.2nF ±10%, 50V	1
VLOGIC Bypass Capacitor (Pin 8)	C4*	Ceramic, X7R, 10nF ±10%, 4V	1

* MPU-6050 Only.

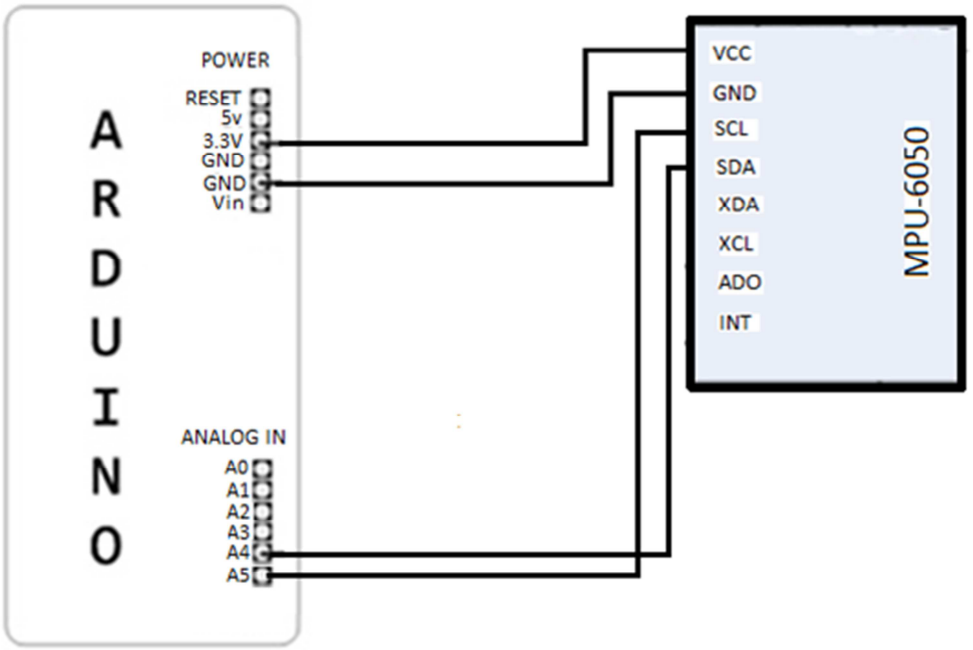
TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO

AUTORES: EUGENIO MOLIS MERINO

FECHA: 25/03/2015

REVISIÓN: 1a

PLANO: Esquema operación de la IMU MPU-6050



TÍTULO: DISEÑO Y CONTROL DE UN SISTEMA ELECTROMECÁNICO INESTABLE DE DOS GRADOS DE LIBERTAD: APLICACIÓN AL CASO DE UN PÉNDULO INVERTIDO

AUTORES: EUGENIO MOLIS MERINO

FECHA: 01/04/2015

REVISIÓN: 1a

PLANO: Esquema conexión Arduino + MPU-6050

CAPÍTULO 10: PLIEGO DE CONDICIONES

10.1. Pliego de condiciones

Las condiciones técnicas principales han sido la de construir la estructura de un robot sobre dos ruedas, con un tamaño reducido capaz de poder ver su funcionalidad en cualquier espacio. El tamaño de las ruedas son medidas generosas tanto de diámetro como de anchura, para conseguir mecánicamente una mayor estabilidad.

Se han utilizado materiales económicos y no muy pesados, para tener una mayor facilidad de transporte. Siguiendo con las condiciones económicas, se han realizado estudios para determinar que dispositivos eléctricos se ajustaban mejor a las necesidades del proyecto teniendo en cuenta su precio.

Este equipo ha de tener un control PID, siendo más sencillo de implementar que otros sistemas de control, aun así se realiza el estudio sobre el control mediante realimentación de variables de estado.

CAPÍTULO 11: TEMPORIZACIÓN

11.1. Temporización

A continuación se expone el diagrama de Gantt del proyecto. Se detalla el número de actividades principales realizadas, y su duración expresada en semanas.

1. Diseño mecánico.
2. Mecanizado estructura.
3. Diseño eléctrico.
4. Instalación eléctrica.
5. Modelado del sistema y desarrollo algoritmo de control.
6. Programación Simulink/Arduino.
7. Simulación.
8. Pruebas.
9. Documentación escrita.

Tabla 5. *Cronograma de las actividades.*

	Año 2014/2015	JULIO				AGOSTO				SETIEMBRE				OCTUBRE				NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO				ABRIL			
Nº	ACTIVIDAD/SEMANA	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	DISEÑO MECÁNICO																																								
2	MECANIZADO ESTRUCTURA																																								
3	DISEÑO ELÉCTRICO																																								
4	INSTALACIÓN ELÉCTRICA																																								
5	MODELAMIENTO DEL SISTEMA Y DESARROLLO ALGORITMO DE CONTROL																																								
6	PROGRAMACION SIMULINK/ARDUINO																																								
7	SIMULACIÓN																																								
8	PRUEBAS																																								
9	DOCUMENTACIÓN ESCRITA																																								

CAPÍTULO 12: PRESUPUESTO

12.1. Presupuesto

En este apartado se presenta el coste de todos los componentes tanto mecánicos como eléctricos necesarios para la realización de este prototipo, así como el coste de la mano de obra.

Los grupos en los que se han dividido los costes son:

- Coste de elementos mecánicos y eléctricos.
- Coste del personal involucrado en el proyecto, o mano de obra.

Los costes están expresados en Euros a no ser que se especifique lo contrario. Las cantidades económicas se expresarán con dos decimales, redondeando su valor en el caso que sea necesario.

12.2. Mano de obra

En el coste de la mano de obra, no se tiene en cuenta todo el tiempo de investigación, pruebas o ensayos realizados para su construcción y funcionamiento final, de esta manera únicamente se ha estimado un tiempo total de 40 horas.

Se especifica el número de personas que han participado en el proyecto, en este caso siendo el de un único operario.

El salario por hora ha sido definido por un valor aproximado de lo que cobra en la actualidad un ingeniero electrónico.

Tabla 6. *Coste final de la mano de obra.*

Mano de obra			
Salario ingeniero electrónico	**CANTIDAD (HORAS)**	**Número operarios**	**COSTE (€)**
20€/Hora	40	1	**800,00**
		Coste total mano de obra (€)	**800,00**
		I.V.A.(21%) (€)	**168,00**
		TOTAL (€)	**968,00**

12.3. Presupuesto final

En la siguiente tabla se presenta el presupuesto definitivo, conteniendo en primer lugar la lista de material utilizado en la mecanización, posteriormente los elementos eléctricos principales, seguido de los otros componentes eléctricos y finalmente la mano de obra.

Tabla 7. *Tabla de presupuesto completo.*

COMPONENTES	CANTIDAD	PRECIO UNITARIO (€)	PRECIO TOTAL (€)
Lista de material mecanización			
Tablero madera prensada 0.5x0.5M	1	4,50	4,50
Varilla roscada Zinc Diam 8mm 1 M	2	1,18	2,36
Tornillo Avell A.cromo CR3 3x12	40	0,03	1,20
Tuerca Hexagonal A.Cincado D.8	20	0,11	2,20
Tuerca ciega Hexagonal A.Cincado D. 8	4	0,22	0,88
Tuerca Hexagonal A.Cincado D.3	8	0,10	0,80
Pieza en forma de L material Aluminio	2	2,00	4,00
Rueda Diam 12 cm anchura 5 cm	2	9,99	19,99
Componente eléctrico principal			
Placa Ardunio UNO REV.3	1	17,00	17,00
Driver doble puente H L298N	1	10,90	10,90
IMU MPU-6050	1	5,60	5,60
Fuente Aliment. 24V 40W 1.6	1	33,00	33,00
Transformador Crovisa 12V 0.5A	1	12,55	12,55
Motor Amax 26 + GS38	2	200,50	401,00
Lista resto material eléctrico			
Regleta C.I. 2 contactos	2	0,33	0,66
Placa Baquelita CT-4	1	2,40	2,40
Fusible 0.5A 5x20 C	1	0,06	0,06
Portafusible Pinza C/I.5	1	0,13	0,13
Interruptor 2/P 1/C	1	0,53	0,53
Puente B-380 C-1500 Redondo	1	0,25	0,25
C.Electrolítico 1mF. 63V 10	1	0,08	0,08
Condensador MKT 100K 100V	1	0,12	0,12
C.Electrolitico 10 mF. 25V	1	0,09	0,09
C.electrolitico 1000 mF. 25V	1	0,18	0,18
C/Integrado L-7812-CV	1	0,34	0,34
Resistencia 330 Ohm 1/4W	1	0,03	0,03
Led Azul Alta Lumin. 12.	1	0,50	0,50
Radiador 2906.6 15x19x10 T0	1	0,45	0,45
Jack 2.1mm Largo CD004H	1	1,02	1,02
Cable 2X0.5mm Bicolor R/N WIR90	1	0,21	0,21
Cable Arduino UNO hembra-hembra	8	0,13	1,00
Cable Arduino UNO macho-macho	6	0,07	0,42
Regleta conexión 10 contactos 10777/1.5	1	1,73	1,73
Cable conexión 0.50 N 1m	2	0,23	0,46
		Coste total componentes (€)	526,64
		I.V.A.(21%) (€)	110,59
		TOTAL (€)	637,23

Mano de obra			
Salario ingeniero electrónico	**CANTIDAD (HORAS)**	**Número operarios**	**COSTE (€)**
20€/Hora	40	1	800,00
		Coste total mano de obra (€)	800,00
		I.V.A.(21%) (€)	168,00
		TOTAL (€)	968,00

Coste total proyecto (€)	1326,64
I.V.A.(21%) (€)	278,59
TOTAL (€)	1605,23

El presupuesto final del proyecto asciende a 1605,23 euros.

Datos a destacar del coste total:

- El 21% corresponde a impuestos, siendo el importe correspondiente al IVA, haciendo un total de 278,59 euros.

- El coste total de todo el material relacionado con el proyecto asciende a 637,23 euros.

- El coste total de la mano de obra asciende a 968,00 euros.

CAPÍTULO 13: CONCLUSIÓN

13.1. Conclusión

La elaboración de este proyecto ha supuesto una manera muy interesante y entretenida de poder realizar el control de un sistema inestable por software mediante simulación, y tener la oportunidad de trasladar ese estudio sobre un robot físico, viendo su comportamiento en tiempo real.

Este proyecto además, ha resultado muy gratificante y útil a la hora de tener que diseñar tanto mecánicamente como eléctricamente un elemento desde cero. También ha servido para volver a tener una toma de contacto directa con la física, puesto que para el diseño mecánico, se ha tenido que tener en cuenta algunas leyes físicas, y ponerlas en práctica en el algoritmo matemático utilizado para su posterior control de estabilidad.

También cabe destacar, los conocimientos adquiridos con el microcontrolador utilizado, en este caso la plataforma Arduino UNO, descubriendo una casa muy importante en Europa sobre la distribución de hardware libre junto a su entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares y destacando su programación en código abierto ideal para iniciarse en el mundo de la electrónica para todos los estudiantes.

Por otro lado, ha dado la oportunidad de poder conocer dispositivos electrónicos que hasta el momento se desconocían, como la IMU MPU-6050, investigando en profundidad su complejo funcionamiento, tanto del acelerómetro y giroscopio.

A través de la IMU MPU-6050, se ha debido de investigar y como consecuencia aprender a adquirir lecturas en grados de inclinación mediante cálculos trigonométricos así como ver la funcionalidad del Filtro Complementario para obtener lecturas mucho más fiables y óptimas.

En el campo de la alimentación, también ha resultado muy interesante, el diseño y posterior construcción mediante soldadura de estaño la fuente de tensión de 12 V DC.

En resumen, ha resultado muy gratificante ver los resultados finales, y un acierto el decidir hacer un proyecto con elementos físicos presentes, teniendo la oportunidad de poder trasladar a un robot real los estudios realizados en un ordenador, interactuando y comunicando con los diferentes elementos que lo constituyen.

13.2. Futuras mejoras

Se han pensado en una serie de futuras mejoras, algunas de ellas incrementando las funcionalidades del robot y otras simplemente mejorando algunas prestaciones ya presentes.

Las posibles mejoras se citan a continuación:

- Emplear el algoritmo del Filtro de Kalman, en la obtención del ángulo de inclinación. Con este sistema en sustitución del Filtro Complementario se ganaría algo de rapidez a la hora de obtener el ángulo, ganando en este caso tiempo para la regulación.

- Implementar otros algoritmos de control, como podría ser llevar a cabo el control por realimentación de variables de estado.

- Diseñar y construir la estructura con otro tipo de material, por ejemplo de metacrilato, siendo un material resistente y no muy pesado.

- Sustituir la alimentación de fuentes de alimentación por baterías, consiguiendo un robot con mayor autonomía e independencia.

- Utilizar una conexión inalámbrica para dirigir el robot y que se mueva siguiendo el recorrido deseado por el usuario.

CAPÍTULO 14: BIBLIOGRAFÍA

Arduino Modules - L298N Dual H-Bridge Motor Controller, (2014-09-14), url: http://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/?lang=es

Arduino web site, version dated (2014-07-15), url: http://www.arduino.cc/

Brogan, W. L., Modern Control Theory. Upper Saddle River, NJ: Prentice Hall, 1985.

Cannon, R., *Dynamics of Physical Systems*. New York: McGraw-Hill Book Company, 1967.

Campbell, D. P., *Process Dynamics*. New York: John Wiley & Sons, Inc., 1958.

Craig, J. J., *Introduction to Robotics*, *Mechanics and Control*. Reading, MA: Addison Wesley Publishing Company, Inc., 1986.

Cunningham, W. J., *Introduction to Nonlinear Analysis*. New York: McGraw-Hill Book Company, 1958.

Dorf, Richard C.; Bishop, Robert H. Sistemas de control moderno. 10ª ed. Madrid [etc.]: Prentice Hall, cop. 2005. ISBN 8420544019.

Evans, W. R., «Control System Synthesis by Root Locus Method», *AIEE Trans Part II*, 69 (1950), pp. 6-9.

Evans, W. R., «The Use of Zeros and Poles for Frequency Response or Transient Response», *ASME Trans*.76 (1954), pp. 1135-44.

Guía IMU MPU-6050 (Acelerómetro y Giroscopio), (2014-10-15),url: http://www.starlino.com/imu_guide.html

Katsuhiko Ogata. "Ingeniería de control moderna". Pearson Educación, S.A., Madrid, (2010): 722-786. ISBN: 978-84-8322-660-5.

Kuo, Benjamin C. Sistemas automáticos de control. 9ª ed. México: Compañía Editorial Continental, 1991. ISBN 9682611393.

Luenberger, D. G., «An Introduction to Observers», *IEEE Trans*. *Automatic Control*, AC-16 (1971), pp.596-602.

MathWorks, Inc., *The Student Edition of MATLAB*, version 5. Upper Saddle River, NJ: Prentice Hall, 1997.

Ogata, K., *Designing Linear Control Systems with MATLAB*. Upper Saddle River, NJ: Prentice Hall, 1994.

Ogata, K., *Discrete-Time Control Systems*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1995.

Ogata, K., *System Dynamics*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2004.
Ogata, K., *MATLAB for Control Engineers*, 4th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2008.

S. Crandall, D. Karnopp, E. Kurtz, and D. Pridmore-Brown. Dynamics of Mechanical and Electromechanical Systems. Krieger Publishing Company, Malabar, Florida, 1968.

Teoría Puente en H. (2014-08-10), url: http://www.taringa.net/post/ciencia-educacion/8662220/Que-es-y-como-funciona-un-puente-H.html

Tutorial: Uso de Driver L298N para motores DC y paso a paso con Arduino. (2014-07-14), url: http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/

Wikipedia. Modulación por ancho de pulso, versión (2014-09-03), url: http://en.wikipedia.org/wiki/pulse-width-modulation, 2012.

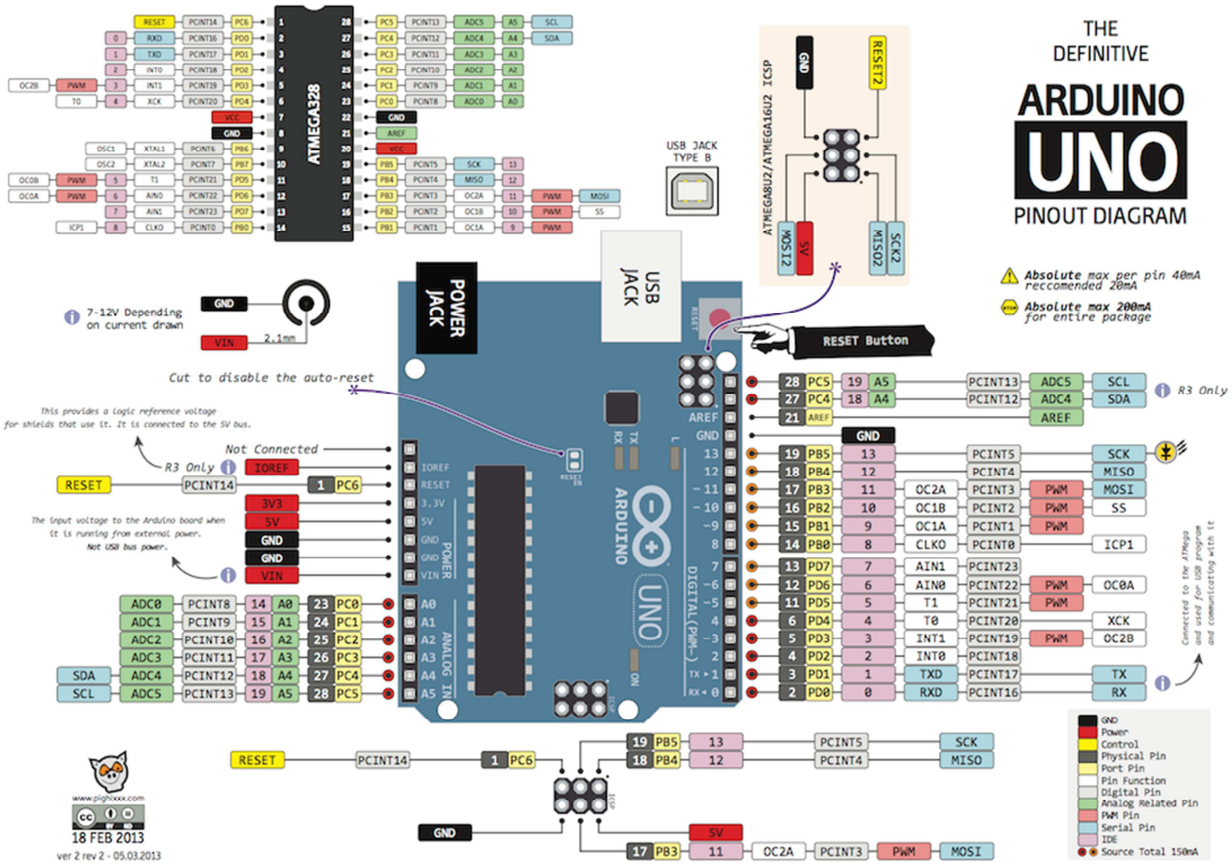
Yorihisa Yamamoto. NXTway-GS Model-Based Design - Control of selfbalancing two-wheeled robot built with LEGO Mindstorms NXT. 2009.

CAPÍTULO 15:
ANEXOS

15.1 Datos técnicos Arduino

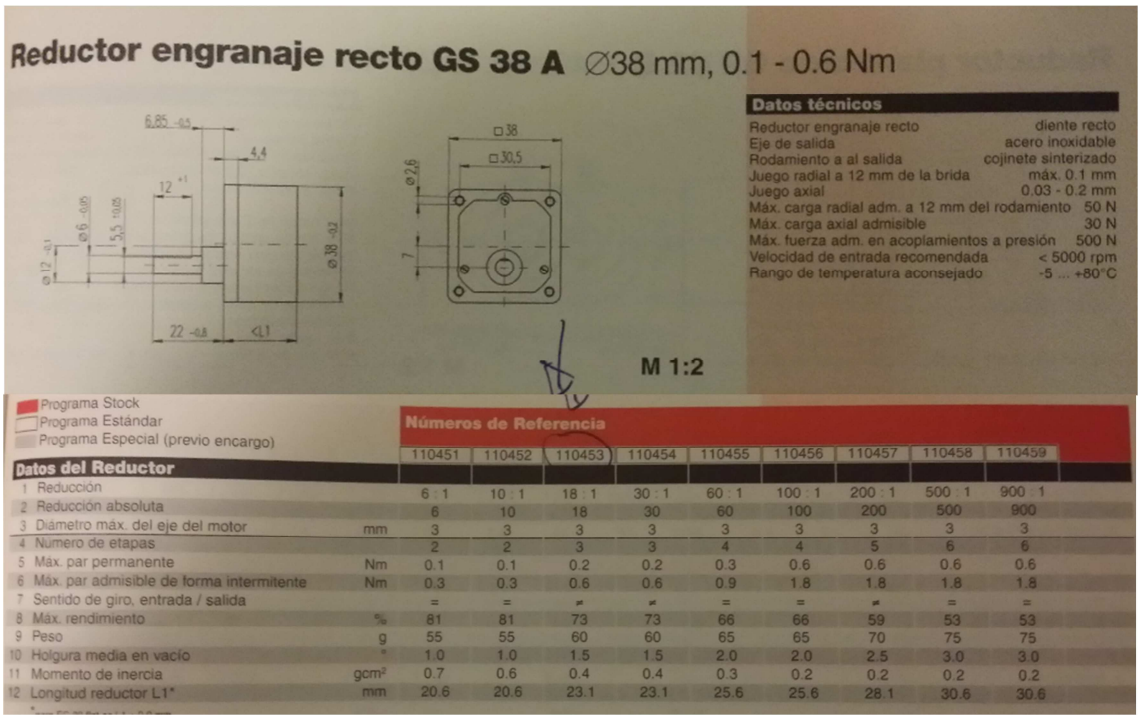
Resumen de características Técnicas

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz



15.2 Datos técnicos Reductor (de catálogo)

Reductor engranaje recto GS 38 A



Reductor engranaje recto GS 38 A ∅38 mm, 0.1 - 0.6 Nm

Datos técnicos

Reductor engranaje recto	diente recto
Eje de salida	acero inoxidable
Rodamiento a al salida	cojinete sinterizado
Juego radial a 12 mm de la brida	máx. 0.1 mm
Juego axial	0.03 - 0.2 mm
Máx. carga radial adm. a 12 mm del rodamiento	50 N
Máx. carga axial admisible	30 N
Máx. fuerza adm. en acoplamientos a presión	500 N
Velocidad de entrada recomendada	< 5000 rpm
Rango de temperatura aconsejado	-5 ... +80°C

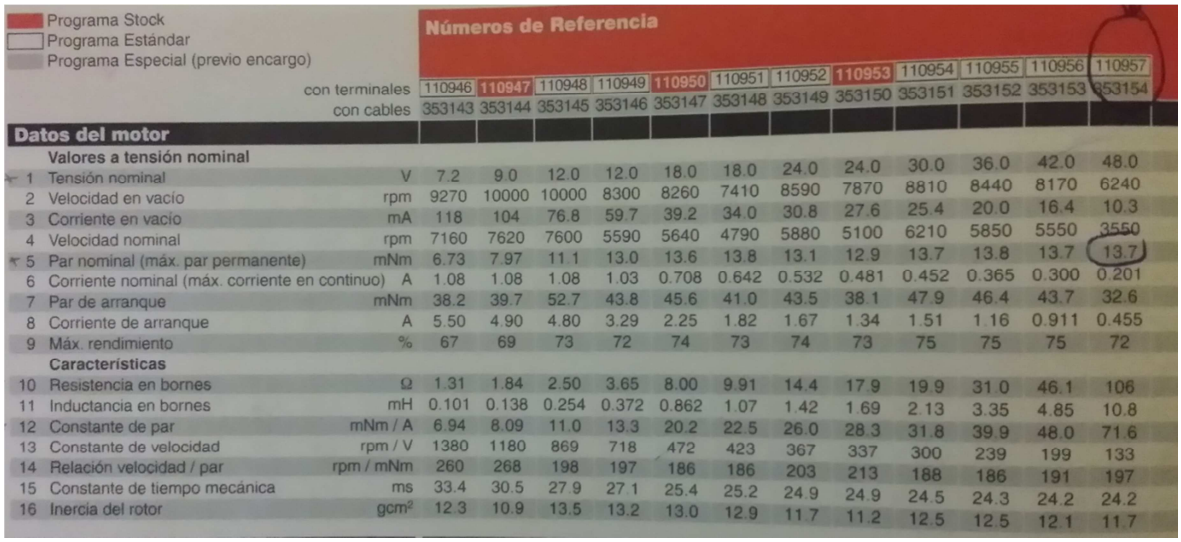
M 1:2

Programa Stock
Programa Estándar
Programa Especial (previo encargo)

Datos del Reductor		Números de Referencia								
		110451	110452	110453	110454	110455	110456	110457	110458	110459
1 Reducción		6 : 1	10 : 1	18 : 1	30 : 1	60 : 1	100 : 1	200 : 1	500 : 1	900 : 1
2 Reducción absoluta		6	10	18	30	60	100	200	500	900
3 Diámetro máx. del eje del motor	mm	3	3	3	3	3	3	3	3	3
4 Número de etapas		2	2	3	3	4	4	5	6	6
5 Máx. par permanente	Nm	0.1	0.1	0.2	0.2	0.3	0.6	0.6	0.6	0.6
6 Máx. par admisible de forma intermitente	Nm	0.3	0.3	0.6	0.6	0.9	1.8	1.8	1.8	1.8
7 Sentido de giro, entrada / salida		=	=	≠	≠	=	=	≠	=	=
8 Máx. rendimiento	%	81	81	73	73	66	66	59	53	53
9 Peso	g	55	55	60	60	65	65	70	75	75
10 Holgura media en vacío	°	1.0	1.0	1.5	1.5	2.0	2.0	2.5	3.0	3.0
11 Momento de inercia	gcm²	0.7	0.6	0.4	0.4	0.3	0.2	0.2	0.2	0.2
12 Longitud reductor L1*	mm	20.6	20.6	23.1	23.1	25.6	25.6	28.1	30.6	30.6

15.3 Datos técnicos Motor DC (de catálogo)

Maxon A-max26, 6 vatios



Programa Stock
Programa Estándar
Programa Especial (previo encargo)

Datos del motor		Números de Referencia											
con terminales		110946	110947	110948	110949	110950	110951	110952	110953	110954	110955	110956	110957
con cables		353143	353144	353145	353146	353147	353148	353149	353150	353151	353152	353153	353154
Valores a tensión nominal													
1 Tensión nominal	V	7.2	9.0	12.0	12.0	18.0	18.0	24.0	24.0	30.0	36.0	42.0	48.0
2 Velocidad en vacío	rpm	9270	10000	10000	8300	8260	7410	8590	7870	8810	8440	8170	6240
3 Corriente en vacío	mA	118	104	76.8	59.7	39.2	34.0	30.8	27.6	25.4	20.0	16.4	10.3
4 Velocidad nominal	rpm	7160	7620	7600	5590	5640	4790	5880	5100	6210	5850	5550	3550
5 Par nominal (máx. par permanente)	mNm	6.73	7.97	11.1	13.0	13.6	13.8	13.1	12.9	13.7	13.8	13.7	13.7
6 Corriente nominal (máx. corriente en continuo)	A	1.08	1.08	1.08	1.03	0.708	0.642	0.532	0.481	0.452	0.365	0.300	0.201
7 Par de arranque	mNm	38.2	39.7	52.7	43.8	45.6	41.0	43.5	38.1	47.9	46.4	43.7	32.6
8 Corriente de arranque	A	5.50	4.90	4.80	3.29	2.25	1.82	1.67	1.34	1.51	1.16	0.911	0.455
9 Máx. rendimiento	%	67	69	73	72	74	73	74	73	75	75	75	72
Características													
10 Resistencia en bornes	Ω	1.31	1.84	2.50	3.65	8.00	9.91	14.4	17.9	19.9	31.0	46.1	106
11 Inductancia en bornes	mH	0.101	0.138	0.254	0.372	0.862	1.07	1.42	1.69	2.13	3.35	4.85	10.8
12 Constante de par	mNm / A	6.94	8.09	11.0	13.3	20.2	22.5	26.0	28.3	31.8	39.9	48.0	71.6
13 Constante de velocidad	rpm / V	1380	1180	869	718	472	423	367	337	300	239	199	133
14 Relación velocidad / par	rpm / mNm	260	268	198	197	186	186	203	213	188	186	191	197
15 Constante de tiempo mecánica	ms	33.4	30.5	27.9	27.1	25.4	25.2	24.9	24.9	24.5	24.3	24.2	24.2
16 Inercia del rotor	gcm²	12.3	10.9	13.5	13.2	13.0	12.9	11.7	11.2	12.5	12.5	12.1	11.7

15.4 Datos técnicos MPU-6050 (de catálogo)

Características eléctricas: especificaciones del giroscopio.

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/g	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

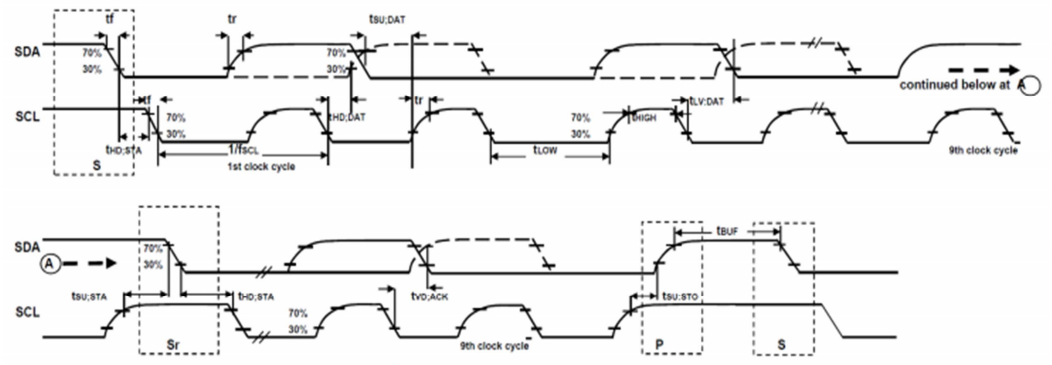
1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus



I²C Bus Timing Diagram

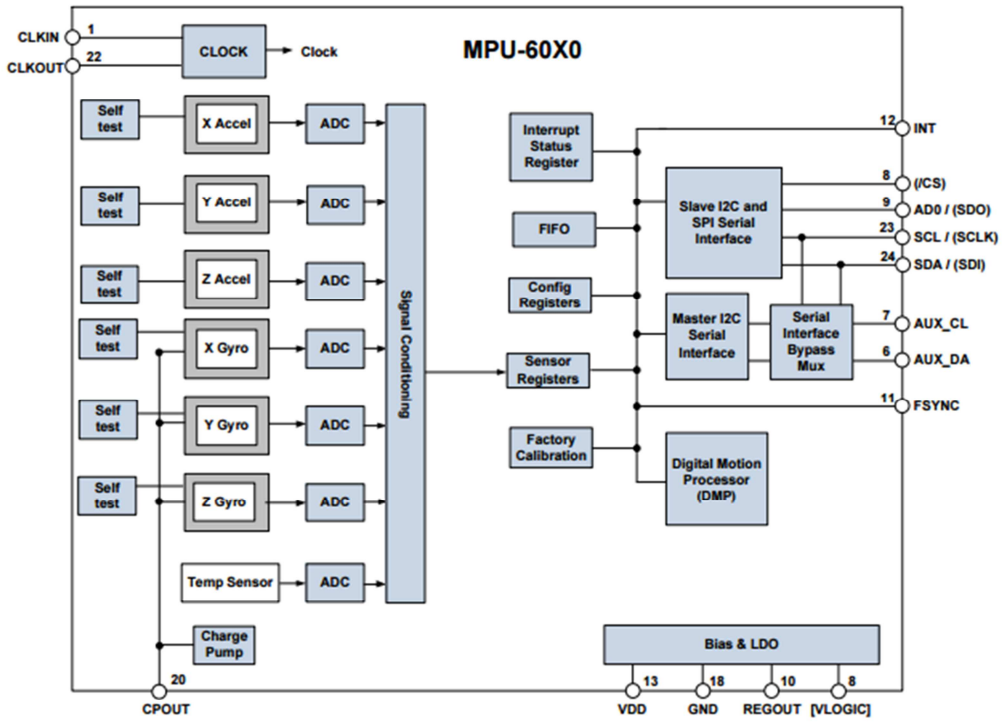
InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013

6.9 Absolute Maximum Ratings

Stress above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level (MPU-6050)	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 250V (MM)
Latch-up	JEDEC Class II (2),125°C ±100mA

7.5 Block Diagram



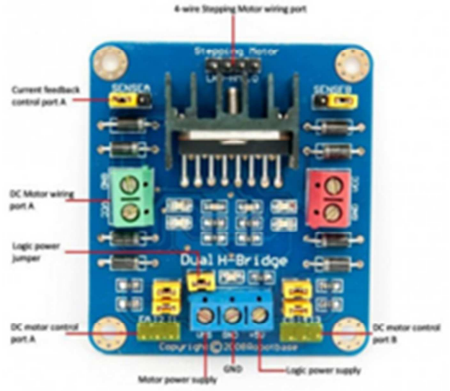
Note: Pin names in round brackets ( ) apply only to MPU-6000
Pin names in square brackets [ ] apply only to MPU-6050

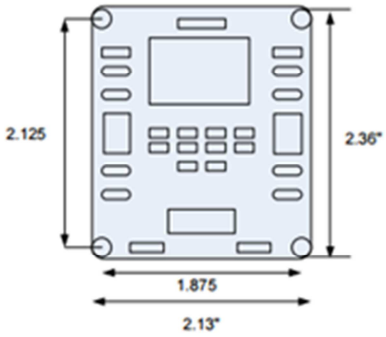
15.5 Datos técnicos driver L298 H-bridge (de catálogo)

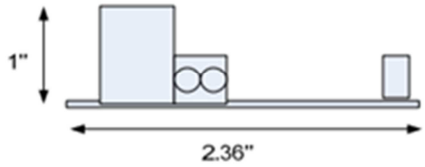
Gobotics.com support@gobotics.com

L298 Dual H-Bridge Motor Driver

Technical data sheet







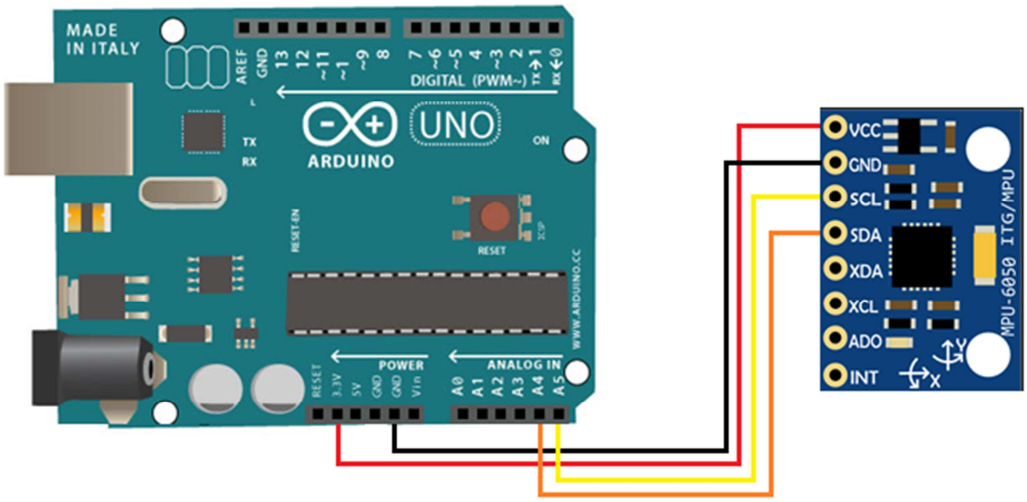
Motor Driver Features:

Compact and Light Weight
High Capacity Heat Sink
Motor Direction LEDs (indicates direction of motor)
Current Feedback for both Ports
Four Pull Up Resistor Switch
Four Standard Mounting Holes

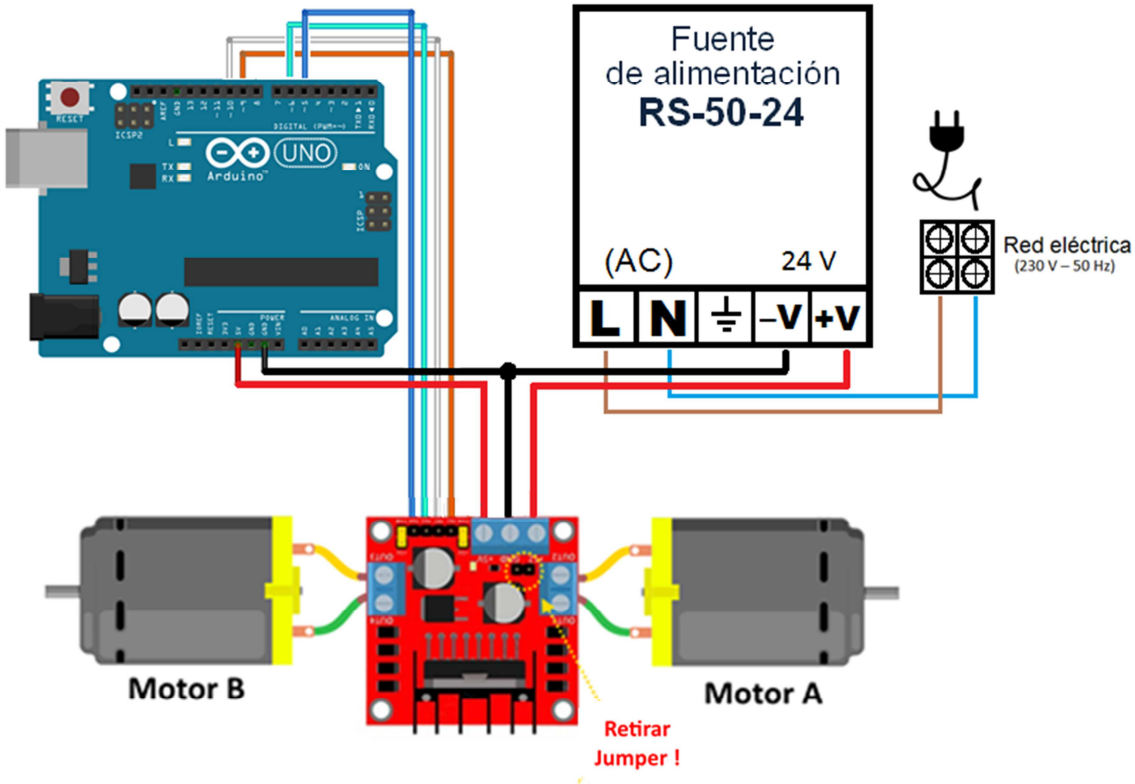
Motor Driver Specifications:

Driver Voltage:	5VDC-46VDC
Driver Peak Current:	2A
Logic Voltage:	5VDC
Logic Power Output:	5VDC at 1A max
Logic Current:	0-36 mA
Logic Levels:	Low: -.3V to 1.5V ; High: 2.3V to VCC
Interface:	TTL
Interface connectors:	Terminal blocks, .100 headers
Dimensions:	2.36" x 2.13" x 1" (not including connector)
Weight:	48 grams
Mounting:	4 corner holes, 0.125" diameter

15.6 Conexión Arduino + MPU-6050



15.7 Conexión Arduino + Etapa de potencia



IN1: 9 (digital PWM Arduino)
IN2: 10 (digital PWM Arduino)
IN3: 6 (digital PWM Arduino)
IN4: 5 (digital PWM Arduino)

15.8 Programa Arduino

```
//Cargar librerias
#include <PID_v1.h>
#include <Wire.h>

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

//Ratios de conversion
#define A_R 16384.0
#define G_R 131.0

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

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

//Angulos
float Acc[2];
float Gy[2];
float Angle[2];

// pines del driver
int IN1=9;
int IN2=10;
int IN3=6;
int IN4=5;
int val, sal, sald;

//CONFIGURACIÓN DEL LAZO PID
//Definir las variables que vamos a conectar
double sp,setpoint, input, output, in, out;

//Definir los parametros de ajuste
double Kp=220.6347, Ki=61.5647, Kd=16.978;

//Especificar los vinculos y los parámetros de ajuste
PID robot(&in,&out,&sp,Kp,Ki,Kd,DIRECT);


void setup()
{

Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B);
Wire.write(0);
```

```
Wire.endTransmission(true);
Serial.begin(9600);


//Configuración pines del driver
pinMode (IN1,OUTPUT); //Input 1 conectada al pin 9
pinMode (IN2,OUTPUT); //Input 2 conectada al pin 10
pinMode (IN3,OUTPUT); //Input 3 conectada al pin 6
pinMode (IN4,OUTPUT); //Input 4 conectada al pin 5


//CONFIGURACION LAZO PID
delay(3000);
Serial.println(sp);
setpoint = Angle[1]; //variable entrada PID
sp=0;      //consigna PID
robot.SetMode(AUTOMATIC);  //enciende el PID
robot.SetSampleTime(20);
robot.SetOutputLimits(-255,255);
}

void loop()
{
  //Leer los valores del Acelerometro de la IMU
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6 registros
  AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();

   //Se calculan los angulos Y
  Acc[1] = atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2) +
pow((AcZ/A_R),2)))*RAD_TO_DEG;


  //Leer los valores del Giroscopio
  Wire.beginTransmission(MPU);
  Wire.write(0x43);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,4,true); //A diferencia del Acelerometro, solo se piden 4
registros
  GyX=Wire.read()<<8|Wire.read();
  GyY=Wire.read()<<8|Wire.read();

  //Calculo del angulo del Giroscopio
  Gy[1] = GyY/G_R;

  //Aplicar el Filtro Complementario
  Angle[1] = 0.98 *(Angle[1]+Gy[1]*0.010) + 0.02*Acc[1];
```

```
  //Configuracion Lazo de Control PID
  input= Angle[1];
  val = setpoint - input;
  in = val;
  robot.Compute();
  if(abs(val)>0){
    if(val>0){
      sal=5;
      sald=10;
     digitalWrite(6,LOW);
     digitalWrite(9,LOW);
    }

    if(val<1)
    {
     sal=6;
     sald=9;
     digitalWrite(5,LOW);
     digitalWrite(10,LOW);
    }
    output = abs(out);
    analogWrite(sal,output);
    analogWrite(sald,output);
    }else{
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
    }

  //Mostrar los valores por consola
  Serial.print("Angle Y: "); Serial.print(Angle[1]); Serial.print("\n----------\n");

  delay(10); //Nuestra dt sera, pues, 0.010, que es el intervalo de tiempo en cada
medida

}
```

15.9 Desarrollo en MATLAB

```
clc
clear all
%%MODELO MATEMATICO ROBOT EQUILIBRISTA
%%MODELO DE ESTADO CON LAS 3 VARIABLES DE ESTADO:
%%[X1 = ángulo péndulo; X2 = vel. angular rueda; X3 = vel. angular péndulo]

%%TODAS LAS VARIABLES DEL SISTEMA

g=9.81;       %% Constante gravitacional [m/s^2]
L=0.18;       %% Longitud entre el eje de las ruedas y el centro de masas del robot (CoM)[m]
r=0.06;     %% Radio de las ruedas [m]
mp=2.15;      %% Masa cuerpo robot [Kg]
mw=0.09;      %% Masa ruedas [Kg]
Jp=0.1026445833;    %% Momento de inercia del cuerpo del robot respecto el centro de masa (CoM)[Kg*m^2]
Jw=0.001385096159; %% Momento de inercia de las ruedas [Kg*m^2]
Ra=106;       %% Resistencia motor electrico [Ohms]
Kt=0.0716;    %% Constante Motor (constante de par) [Nm/A]
Ke=0.07179;   %% Constante de f.c.e.m (fuerza contra electromotriz)(BEMF) [Vs/rad] = [V/rpm]
n=18;         %% Reducción motor (Reductor)


%%definición ecuaciones PLANTA matriz A y B

a21=(g*L^2*r*mp^2)/(-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)));
a22=(-2*n*Jp*Ke*Kt-2*L*n*r*Ke*Kt*mp)/((-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)))*Ra);
a31=(g*L*Jw*mp*Ra+L*r^2*mp*(g*mp+g*mw)*Ra)/((-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)))*Ra);
a32=(-2*n*Jw*Ke*Kt-2*n*r*Ke*Kt*((L+r)*mp+r*mw))/((-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)))*Ra);

b21=(2*n*Jp*Kt+2*L*n*r*Kt*mp)/((-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)))*Ra);
b31=(2*n*Jw*Kt+2*n*r*Kt*((L+r)*mp+r*mw))/((-L^2*r^2*mp^2+Jp*(Jw+r^2*(mp+mw)))*Ra);

%%MODELO PLANTA

A=[0 0 1;a21 a22 0;a31 a32 0];
B=[0; b21; b31];
C=[1 0 0];
D=[0];


%Polos de la PLANTA

POLOS_PLANTA=eig(A)

% CONTROLABILITAT SISTEMA:

A=[0 0 1;a21 a22 0;a31 a32 0];
B=[0; b21; b31];

Mc = ctrb(A,B);

if rank(Mc)<length(A)
    disp('no controlable')
    disp(['el numero de variables destat no controlables es ',num2str(length(A)-rank(Mc))]);
else
    disp('controlable')
end;
```

```
% CONDICION DE OBSERVABILITAT:

A=[0 0 1;a21 a22 0;a31 a32 0];
C=[1 0 0];

Mo = obsv(A,C);

if rank(Mo)<length(A)
    disp('no observable')
    disp(['el numero de variables destat no observables es ',num2str(length(A)-rank(Mo))]);
else
    disp('observable')
end;

%DISEÑO DEL REGULADOR
%Fijar los polos del regulador por realimentación del vector de estado
polos_regulador=[-8+8.16j -8-8.16j -40];
% Ecuación caracteristica del sistema (planta) para las condiciones asignadas
polinomio_caracteristico_regulador=poly([-8+8.16j -8-8.16j -40])

%Calculamos los coeficientes de realimentación de estado (vector K)
K=acker(A,B,polos_regulador)

%Calculo Funcion de Transferencia de la planta
[num,den] = ss2tf(A,B,C,D)
```