Universidad del País Vasco
Euskal Herriko Unibertsitatea
ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Trabajo fin de grado
Grado en Ingeniería Electrónica

Estudio de sensores IMU:
Control de equilibrio mediante un robot móvil

Autor/a:
Sergio Fernández Olivera
Directores/as:
Dr. Josu Jugo Garcia
Dr. Iñigo Arredondo López de Guereñu

Leioa, 6 de junio de 2023

Resumen

A lo largo de este documento puede verse un estudio realizado sobre unidades de medición inercial (IMU) y una aplicación experimental realizada con este tipo de sensores. Durante el estudio se han explicado los conceptos teóricos y las bases físicas relacionadas con el funcionamiento de estos sensores y la tecnología que utilizan.

Para la realización del experimento se ha utilizado una gran variedad de *Hardware* y *Software*, por lo que se han realizado comparaciones entre las diferentes posibilidades y se ha explicado el funcionamiento de los componentes principales.

El experimento realizado consiste en el control de un sistema dinámico, por lo que también se ha trabajado en teoría de control y su implementación mediante software. El sistema dinámico mencionado está formado por un balancín y dos masas, siendo una de ellas un robot.

Para la realización del experimento se han montado tanto el balancín como el robot. Además se ha realizado un diseño CAD del balancín que después ha sido impreso en 3D y un diseño electrónico de una placa PCB diseñada a medida. El diseño de la placa PCB incluye todo el *Hardware* necesario para el funcionamiento del robot, esta ira ensamblada sobre un chasis comercial.

El *Software* del robot también ha sido programado desde cero en Circuitpython.

Agradecimientos

Por su continua ayuda en relación al montaje y diseño del balancín en 3D a lo largo del proyecto al *Dr. Jorge Feuchtwanger* investigador Ikerbasque adscrito al Departamento de Electricidad y Electrónica de la Universidad del País Vasco.

Por el interés demostrado en el proyecto y su apoyo a la hora de resolver problemas relacionados con software a *Pello Usabiaga*, graduado en Ingeniería Electrónica en la Universidad del País Vasco.

Índice general

Glosario

- Protocolo I2C:

  Se trata de un protocolo de comunicación síncrono, multimaestro y multiesclavo [1]. Una de sus mayores ventajas es que utiliza tan solo dos cables, uno que envía la señal de reloj (SCL) y otro que envía la información (SDA). Mediante estos dos únicos cables puede realizarse la comunicación con múltiples dispositivos a los que comúnmente se les llama esclavos, esta comunicación viene gestionada por el dispositivo maestro. Cada dispositivo se identifica por medio de una dirección de 7 bits.

- PWM:

  En castellano "Modulación por ancho de pulso"hace referencia a un tipo de señal digital, la cual permite controlar la potencia media entregada controlando el ciclo de trabajo de dicha señal. Es un tipo de señal digital utilizada comúnmente en microcontroladores para controlar dispositivos analógicos.

- *Smart Sensors*:

  Son circuitos integrados que disponen del hardware y software necesario para realizar las mediciones, procesarlas digitalmente y enviarlas a otros dispositivos mediante el uso de protocolos de comunicación.

- MEMS:

  Son las siglas de "Micro-electromechanical Systems", son un tipo de dispositivos integrados del orden de micrómetros, estos dispositivos combinan componentes electrónicos y mecánicos

- Circuitpython:

  Circuitpython es un lenguaje de programación basado en python y diseñado para programar placas de microcontroladores de una forma sencilla.

- Protocolo MQTT:

  El protocolo *Message Queuing Telemetry Transport*, (MQTT), es un protocolo de comunicación de máquina a máquina ligero y que requiere un ancho de banda limitado. Su funcionamiento consiste en realizar publicaciones en *topics* que después pueden ser leídas por clientes subscritos al mismo *topic*, de esta forma diferentes clientes MQTT pueden escribir y leer información de un *Broker MQTT*.

Capítulo 1

Introducción

Las IMU o unidades de medición inercial, son unos dispositivos electrónicos que se utilizan para medir la aceleración, velocidad angular y la orientación de un objeto. Tienen su origen en la tecnología aeroespacial de la década de 1950. La motivación tecnológica después de la Segunda Guerra Mundial hizo posible el desarrollo de esta tecnología lo suficiente como para lograr una precisión y fiabilidad, que hizo posibles las misiones Apollo, su uso con fines militares y el uso de esta tecnología en aviones comerciales.

Sin embargo, su uso no se limita a la navegación, ya que gracias al avance tecnológico, el tamaño y el coste de estos sistemas se ha ido reduciendo cada vez más expandiendo su uso hacia un gran número de aplicaciones comerciales. Los sistemas MEMS permiten que este tipo de sensores junto con otros muchos se encuentren en un mismo circuito integrado que además de sensores puede incluir todo tipo de circuitería digital o analógica, (Smart Sensors).

Esto convierte a los sensores inerciales en una tecnología de gran importancia, por uso uso en múltiples industrias, como la automotriz, la robótica, la aeroespacial, la de realidad virtual, etc.

A lo largo de este documento se describirán los conceptos básicos necesarios para poder entender el funcionamiento de estos sensores y se realizará una aplicación experimental que utilizara un IMU.

Durante el proyecto se han utilizado diversas herramientas, entre las que se encuentran, Arduino IDE, Arduino Cloud, Adobe Visual Studio Code, Autodesk Fusion 360, Docker, Latex, Scilab, GitHub y RefWorks.

1.1. Objetivos

El objetivo de este proyecto es realizar un estudio sobre los sensores inerciales, a través del cual se pretende entender cuales son los principios que permiten el funcionamiento de estos sensores, como se trabaja con ellos y las posibilidades que ofrecen. Además se espera ser capaz de implementar un IMU en una aplicación real.

El proyecto se ha planteado desde un punto de vista práctico y experimental donde se espera poner en práctica conceptos teóricos e incluso obtener competencias electrónicas relacionadas directa o indirectamente con el tema a tratar.

Una vez entendido el funcionamiento de los sensores IMU, se realizará el montaje de un robot que incorpore uno de estos sensores y se montará un balancín, sobre el cual deberá equilibrarse el robot mediante un control automático. El sensor IMU proporciona al robot la información necesaria para que este la procese y actúe en consecuencia.

Para llevar a cabo la aplicación será necesario utilizar teoría de control para procesar los datos del IMU y conseguir que el robot actúe de una forma adecuada. Uno de los objetivos es conseguir un control que logre esto y sea capaz de controlar el robot en alguna tarea concreta.

Se quieren realizar modelos CAD mediante *Software* tanto del robot como del balancín con la intención de obtener competencias relacionadas con herramientas profesionales de este tipo y mejorar el desarrollo del proyecto.

El balancín será impreso en 3D utilizando estos modelos CAD, de esta forma se obtendrán competencias relacionadas con la impresión 3D.

Para el montaje del robot se pretende realizar un diseño PCB de la placa del circuito y mandarla a fabricar. El resultado será mejor que utilizando un placa de prototipado y se obtendrán competencias relacionadas con el diseño PCB.

Capítulo 2

IMU

Un IMU, es un dispositivo electrónico que incorpora diferentes sensores, como acelerómetros, giróscopos y en ocasiones magnetómetros, para medir la velocidad angular, aceleración y orientación de un cuerpo o sistema de referencia.

Los tres tipos de sensores mencionados anteriormente pueden ser utilizados para obtener la orientación midiendo diferentes variables físicas y realizando los cálculos necesarios, no obstante por motivos que se discutirán más adelante, en los IMU se utiliza la fusión de diferentes sensores para aumentar la precisión y fiabilidad de los datos obtenidos, ya que cada sensor tiene sus limitaciones.

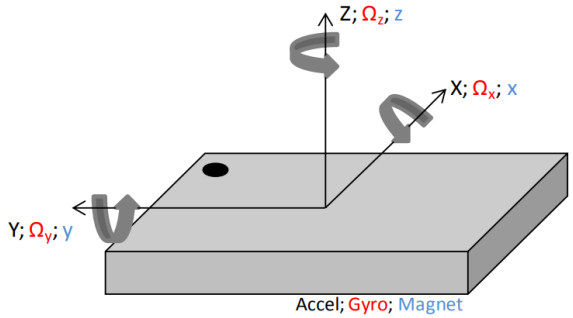

Figura 2.1: Diagrama de ejes de la unidad de medición inercial [2].

2.1. Fundamentos teóricos

2.1.1. Acelerómetros

El acelerómetro es un sensor que mide la aceleración a la que está sometido un cuerpo. Mediante el uso de tres acelerómetros es posible medir la aceleración total de un cuerpo en el espacio y por tanto la fuerza a la que está siendo sometido en cada uno de sus tres ejes.

El principio básico de funcionamiento de estos sensores consiste en un sistema masa muelle que al verse afectado por fuerzas externas sufre un desplazamiento medible. En la actualidad los acelerómetros son sistemas MEMS y aunque todos comparten el mismo principio de funcionamiento existen diferentes estructuras para estos sensores, que se diferencian entre ellas en el método utilizado para medir el desplazamiento de la masa y en el tipo de micro-mecanizado utilizado. Dos de los métodos más comunes son el piezorresistivo y el capacitivo.

En el método capacitivo la masa del sistema actúa como una de las dos placas de un condensador, al variar la distancia entre estas dos placas también lo hace la capacitancia de este, esta puede medirse mediante un circuito electrónico.

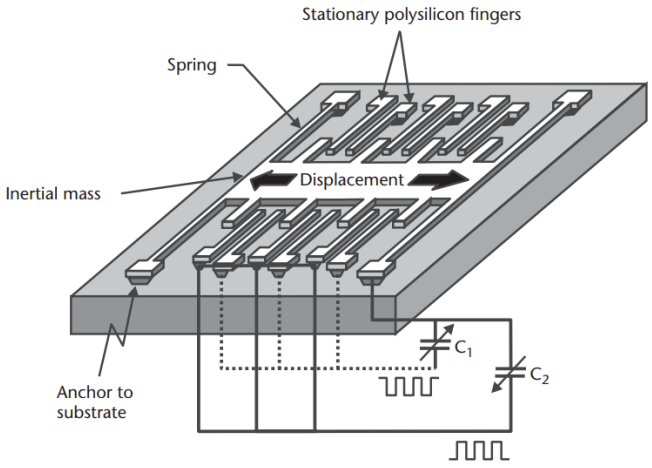

Figura 2.2: Acelerómetro capacitivo MEMS de la familia ADXL.

Un ejemplo de una de estas estructuras MEMS puede verse en la Figura 2.2, obtenida del libro *An Introduction to Microelectromechanical Systems Engineering* escrito por Nadim Maluf en 2004 [3], donde se explica de forma más exhaustiva los diferentes tipos de acelerómetros MEMS. La figura corresponde a la estructura básica de acelerómetros MEMS de la familia ADXL [4].

Para la obtención del ángulo de orientación del sensor, si el acelerómetro no se ve afectado por ninguna fuerza o aceleración externa este medirá la gravedad, puesto que es una aceleración constante, al menos en la Tierra, por tanto el eje que mida la gravedad será aquel que tenga la misma dirección que el vector de gravedad. En la mayoría de los casos la gravedad no será medida en un único eje, en su lugar se medirán sus componentes en cada uno de los 3 ejes.

Cuando el sensor y por tanto su sistema de coordenadas cambia su orientación la gravedad será medida por otro conjunto de ejes diferente. Haciendo uso de trigonometría básica es posible conocer la orientación de cada eje respecto al vector de gravedad conociendo el componente gravitatorio que mide cada eje, en la Figura 2.3 puede verse un diagrama simplificado para obtener uno de los ángulos.

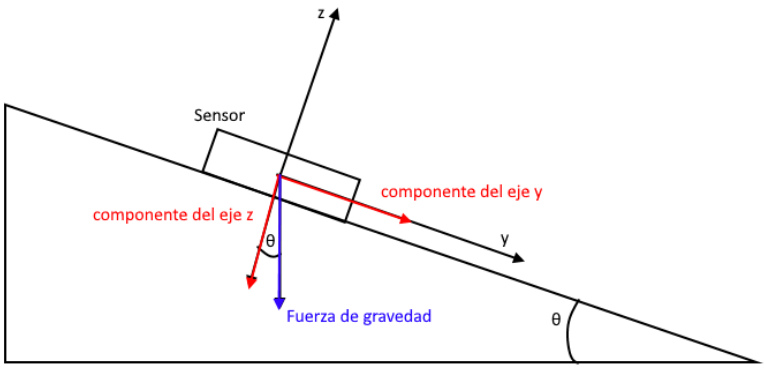

Figura 2.3: Diagrama de fuerzas simplificado.

Por tanto el ángulo θ_x puede obtenerse con la siguiente ecuación, siendo F_y y F_z las fuerzas medidas por el sensor,

$$\theta_x = \arctan\left(\frac{F_y}{F_z}\right) \tag{2.1}$$

de la misma forma se puede obtener el ángulo θ_y,

$$\theta_y = \arctan\left(\frac{F_x}{F_z}\right) \tag{2.2}$$

En el caso particular en el que el sensor rote sobre el eje perpendicular a la superficie de la Tierra la gravedad siempre será medida por ese eje, por mucho que el sensor rote. De esta forma es posible saber que el plano del sensor es el mismo que el de la superficie, pero no es posible conocer la orientación de los otros dos ejes.

En este caso un acelerómetro sólo permite obtener la inclinación del sistema de referencia en dos de sus ejes. Este es uno de los motivos por el cual los IMU incorporan tanto acelerómetros como giroscopos.

Este caso particular se suele dar comúnmente, por ejemplo en los robots que se desplazan de forma tangencial al plano de la superficie terrestre.

2.1.2. Giróscopos

Los giróscopos son sistemas mecánicos que permiten conocer la orientación de un cuerpo midiendo su velocidad angular. El principio básico consiste en un disco o rueda sobre un soporte, que gira a gran velocidad sobre su propio eje y que a causa de la conservación del momento angular mantiene la dirección de su eje de rotación constante ante las perturbaciones de su propio soporte.

Sin embargo en la actualidad cuando se habla de forma coloquial de giróscopos se hace referencia, en realidad, a sensores de velocidad angular modernos formados por sistemas (MEMS). Este tipo de sensores es el más común y utilizado en IMUs dada su alta capacidad de integración y su bajo coste. Sin embargo la precisión de un giróscopo reside en su gran cantidad de momento angular, que depende de su tamaño, masa y velocidad de giro.

Por ello, los sensores de tamaño reducido como lo son los MEMS; no son giróscopos efectivos y en lugar de usarse para medir directamente velocidad angular, se hace uso del efecto Coriolis.

Los sensores de velocidad angular MEMS son de tipo vibratorio y tienen como núcleo una masa que vibra con cierto componente de velocidad lineal en un sistema de referencia fijo. En un sistema de referencia rotatorio como lo es la tierra, esa masa es afectada por una fuerza de Coriolis que causa la aparición un componente nuevo de velocidad, perpendicular al componente original y como resultado al nuevo componente de velocidad, la masa comienza a oscilar en un modo de frecuencia secundario. Para sistemas con alto grado de simetría, como los cilindros, la frecuencia de resonancia está degenerada en función

Este efecto es la base fundamental para realizar mediciones utilizando el efecto Coriolis, pero hay diversos tipos de estos sensores, que tienen diferentes estructuras, métodos de excitar los modos y forma de medición [3].

Un ejemplo es el sensor de velocidad angular de "Delphi Delco Electronics Systems"[5], que puede verse en la Figura 2.4,

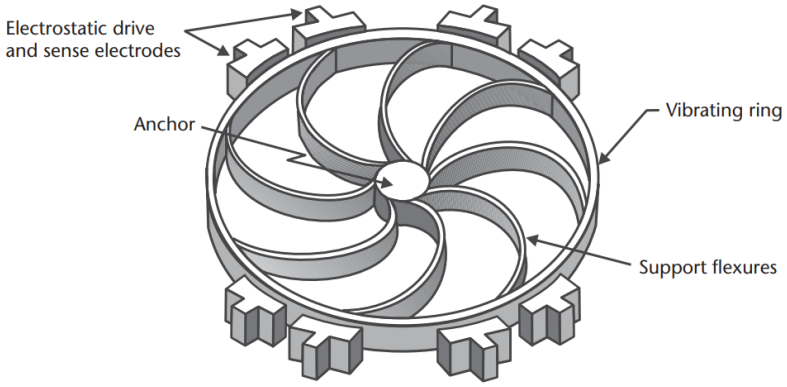

Figura 2.4: Sensor de velocidad angular Delphi Delco

El núcleo de este sensor en un anillo vibratorio anclado al substrato, este anillo se deforma al vibrar creando diferentes patrones de vibración, compuestos por nodos y antinodos, estos son los puntos que se mantienen estacionarios y los puntos de máxima deflexión respectivamente.

Cada frecuencia de resonancia tiene sus patrones de vibración característicos, dado que el anillo es un cuerpo de alta simetría este posee dos modos de resonancia degenerados de

tal forma que los nodos de un modo coinciden con los antinodos del otro.

Es posible excitar uno de los modos de resonancia mediante un circuito electrónico. Como el anillo está en vibración el efecto Coriolis excita el segundo modo de vibración dando como resultado una combinación lineal de los dos modos con una nueva combinación de nodos y antinodos. Como consecuencia se forma un patrón rotado respecto al patrón del modo principal.

Midiendo la amplitud de la deflexión y la desviación del patrón se puede obtener la velocidad angular. Además es posible realimentar en lazo cerrado las señales obtenidas con el propósito de anular el segundo modo mediante excitaciones electrostáticas, de esta forma se mantiene una vibración estacionaria y se puede obtener la velocidad angular directamente de la realimentación.

A pesar de que estos sensores miden velocidad angular, midiendo el cambio de velocidad angular en un cierto periodo de tiempo, se puede obtener mediante integración el ángulo de rotación relativo. Esto es de gran importancia ya que mediante el uso de un giróscopo no es posible conocer la orientación absoluta, solo la orientación relativa respecto a un instante anterior.

$$\theta = \int \dot{\theta} dt \tag{2.3}$$

Como sucedía con los acelerómetros, los giróscopos también tienen sus limitaciones, aparte de no poder conocer la orientación absoluta, está el hecho de que este tipo de sensores tiene una deriva en sus mediciones, un error acumulativo, que provoca que con el paso del tiempo sus mediciones sean cada vez menos fiables. Sin embargo los errores relacionados con la deriva no son críticos ya que existen métodos de corrección para este tipo de errores, se puede cuantificar esa deriva a lo largo del tiempo por lo que es posible corregirla mediante postprocesado.

2.1.3. Magnetómetros

Un magnetómetro es un dispositivo que mide la intensidad y dirección de una fuerza magnética. Pueden fabricarse microsensores de este tipo haciendo uso de la magnetorresistencia anisótropa, (AMR), como es el caso del circuito integrado HMC5883L [6], que utiliza este tipo de tecnología.

La resistividad de este tipo de materiales magnéticos depende del ángulo θ que forman el vector de imanación y la dirección de la corriente que atraviesa el material. Por tanto la resistencia de una muestra ferromagnética puede describirse de esta forma,

$$R = R_0 + \Delta R \cos\theta \tag{2.4}$$

siendo el valor de R mayor cuando las dos direcciones están alineadas y menor cuando estas son perpendiculares, (Figura 2.5).

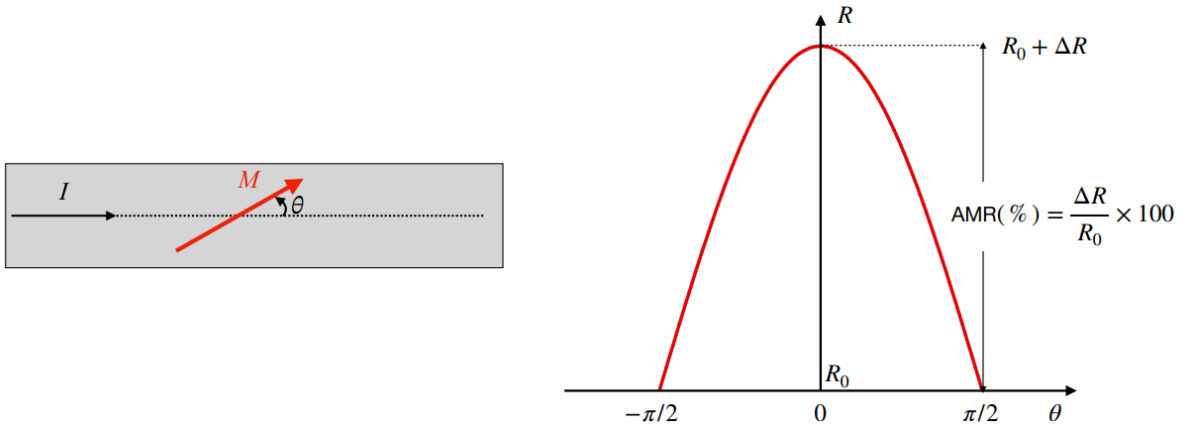

Figura 2.5: Magnetorresistencia: dependencia de la resistencia con el ángulo de la imanación [7].

Estas muestras ferromagnéticas tienen una anisotropía magnética creada durante su fabricación por lo que en ausencia de un campo magnético tienen una imanación perpendicular al eje de medición. Para su uso como sensor magnético se hace circular una corriente constante en una dirección determinada, cuando la muestra está expuesta a un campo magnético este cambia la dirección de la imanación y por tanto la resistencia de la muestra. Esa variación relativa de resistencia en la muestra es la cuantificación de la magnetorresistencia AMR y puede medirse obteniendo la tensión entre los extremos de la muestra, (Figura 2.5).

Tal y como se deduce de la ecuación, 2.4 $\theta \in [\frac{-\pi}{2}, \frac{\pi}{2}]$, con un valor de $\theta = \pi$ por ejemplo, se obtendría un valor de R equivalente a $\theta = 0$.

En el caso de tener dos campos magnéticos idénticos teniendo uno $\theta = 0$ y el otro $\theta = \pi$, no sería posible distinguir el uno del otro. Para poder distinguirlos, al utilizar este tipo de materiales como sensores se realiza una linealización alrededor de un punto de operación. Este punto de operación se obtiene polarizando el sensor mediante un campo magnético conocido, H_b.

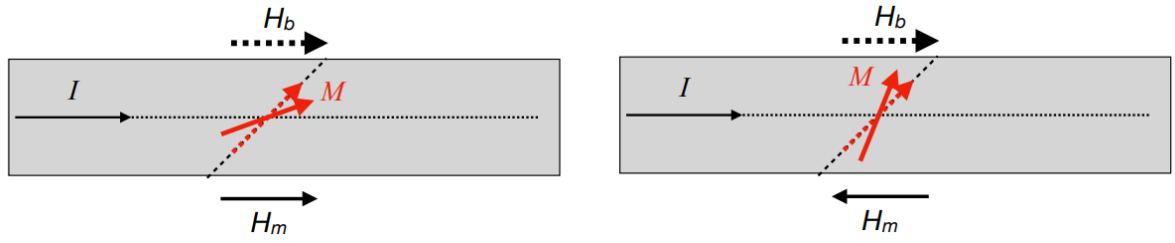

Figura 2.6: Efecto del campo a medir a partir del punto de operación [7].

De esta forma como puede verse en la Figura 2.6 pueden distinguirse dos campos H_m

idénticos en direcciones opuestas.

Si la medida de un magnetómetro no se ve alterada por ningún campo magnético externo y el dispositivo tiene la sensibilidad y resolución necesarias entonces la medida que se estará realizando será la del campo magnético terrestre, que será del orden de [25-65] μT dependiendo de posición en la que se realice la medición en el plano terrestre.

El magnetómetro puede usarse para calcular la orientación absoluta sobre el plano de la tierra, conociendo el componente del campo magnético en cada eje y utilizando trigonometría.

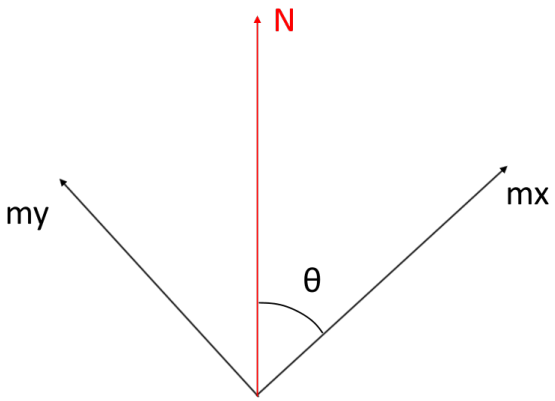

Figura 2.7: Ilustración de obtención de la orientación.

Siendo m_y y m_x medidas del campo magnético y N el norte geográfico,

$$\theta = \arctan\left(\frac{m_y}{m_x}\right) \tag{2.5}$$

2.2. Uso experimental de un IMU

Montaje del circuito

El primer paso para poder utilizar un IMU y entender cómo funciona será el montaje de un circuito básico, el propio IMU conectado únicamente a una placa Arduino Mega 2560, a través de la cual se pueden recoger datos y visualizarlos. El sensor IMU que se ha utilizado es el módulo GY-87.

Para conectar el módulo GY-87 a la placa Arduino, se conecta el pin de 5V al pin VCC_IN del módulo, el pin GND del módulo a cualquier pin GND del Arduino y por último los pines de la comunicación I2C, los pines SDA y SCL.

Descripción del modulo GY-87

El módulo GY-87 es un dispositivo de 10 grados de libertad (10DOF) formado por un MPU6050 [8] que incorpora un acelerómetro de tres ejes y un giróscopo de tres ejes en un mismo chip además de contar con un "Digital Motion Processor"(DMP) capaz de procesar algoritmos complejos para la fusión de sensores en 9 ejes.

El módulo GY-87 también cuenta con un barómetro BMP180 [9] y un magnetómetro HMC5883 [6]. Estos sensores están comunicados entre ellos usando el protocolo de comunicación I2C, siendo el chip MPU6050 el maestro que gestiona la comunicación y los otros sensores los esclavos.

Es importante tener claro que el módulo GY-87, como muchas unidades de medición inercial, no es un sensor como tal, sino un circuito electrónico diseñado con los componentes necesarios para el uso rápido y sencillo de los sensores HMC5883, BMP180 y MPU6050. Por tanto para poder entender el funcionamiento del módulo es necesario entender el funcionamiento de cada uno de los sensores, para ello se consulta la hoja de datos técnica de cada uno de ellos.

Sin embargo, aunque el uso de este tipo de módulos es relativamente sencillo gracias a las librerías de código existentes, el entendimiento completo del funcionamiento de este tipo de sensores requiere un gran conocimiento en profundidad de microcontroladores, protocolos de comunicación, programación a nivel de registro, etc. Por ello, dado que el propósito de este proyecto es el estudio de estos chips de una forma más superficial y su completo entendimiento requeriría por sí solo una gran inversión de tiempo, a lo largo de este proyecto no se analizara la estructura a nivel de registros de estos sensores.

Programa para la obtención de datos

Una vez que el circuito ya esta montado se creará un programa básico en C utilizando el Arduino IDE, este programa se encargará de realizar las siguientes tareas:

1. Realizar la comunicación I2C con el módulo GY-87:

   Para trabajar con los distintos sensores dentro del módulo se han utilizado sus respectivas librerías. Para configurar la comunicación I2C se establece la velocidad del reloj y se inicializa cada sensor asignándole su dirección I2C, para que al utilizar funciones de escritura o lectura de sus librerías puedan utilizar el protocolo I2C internamente.

2. Realizar la conversión de datos a las unidades deseadas:

   En este punto ya es posible obtener datos de los sensores utilizando las librerías, estas funciones se encargan de leer los registros para almacenar en variables la información que hay en ellos. Pero para poder entender esa información se debe comprobar en qué unidades están los valores obtenidos.

En este caso la información viene dada en unidades de *Less Significant Bit / Gravity*, (LSB/g) que representa el valor del bit menos significativo por cada unidad de gravedad medida. Los registros y el ADC son de 16 bits y el rango de escala completa que se está utilizando $\pm 2g$ por lo que se puede obtener la sensibilidad en bits de la siguiente forma:

$$2^{16} = 65536 \longrightarrow \frac{65536}{4g} = 16384 \quad LSB$$

Una vez conocida la sensibilidad se realiza la conversión a unidades del sistema internacional, es decir m/s^2

$$\frac{\text{acc\_raw\_data}}{16384} \cdot g = \text{acc\_data} \quad m/s^2$$

3. Calibrar los sensores:

   Para realizar la calibración se guardan los valores de los *offsets* de cada sensor en sus respectivos registros, las librerías se encargan también de realizar la escritura, por lo que la única tarea ha realizar es conseguir los valores adecuados.

   Dado que los valores que se esperan son conocidos, se puede optar por tomar mediciones y después de determinar el valor promedio de los datos, el offset será el valor restante para alcanzar los valores deseados. Existen métodos iterativos más eficaces, como el programa que proporcionan las librerías.

   Este programa utiliza un control proporcional e integral, (PI) para encontrar el valor óptimo de los *offsets*, se coloca el sensor en una superficie horizontal, se establece como referencia el valor 0 y se obtiene el error de forma iterativa. El término proporcional sirve para reducir el efecto del ruido del sensor mientras que el término integral se va acumulando hasta corregir el error en estado estacionario. Cuando esto ocurre la aportación del término integral se usa como *offset*.

4. Abrir la comunicación con el puerto serie para mandar la información:

   Para poder visualizar los datos se inicia la comunicación con el puerto serie a 115200 baudios y se utiliza la función *Serial.print*() para escribir por el puerto la información deseada. De esta forma se pueden ver los datos a través del monitor serie del Arduino IDE o visualizarlos de forma gráfica a través del *Serial Plotter*.

Visualización e interpretación de las mediciones del IMU

Se ha creado un programa secundario en python con el objetivo de visualizar de forma más detenida la información utilizando la librería matplotlib y exportando las muestras en formato ".csv"para un análisis posterior en caso de ser necesario.

En la Figura 2.8 se muestran los datos obtenidos en el eje z del acelerómetro, siendo el eje perpendicular al plano de la tierra los valores se corresponden con la gravedad. También se puede observar un efecto de cuantificación ya que las muestras parecen estar repartidas

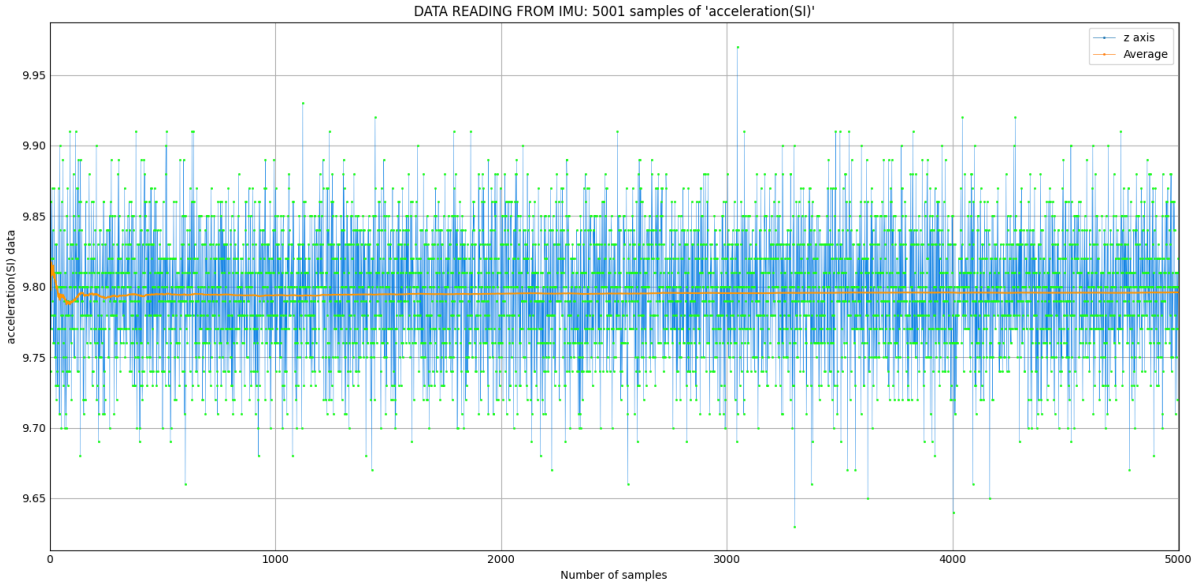

Figura 2.8: Medida del acelerómetro del módulo GY-87 en el eje z.

en líneas horizontales, con una separación entre ellas de $0{,}01 m/s^2$.

La cuantificación limita la resolución del sensor a $0{,}01 m/s^2$, en cuanto a precisión, las muestras presentan una desviación típica de $\pm 0{,}1 m/s^2$ sin embargo, esto es principalmente causado por el ruido presente en la medición y puede ser filtrado, la línea naranja representa el valor medio de las muestras obtenidas, el ruido mencionado fluctúa siempre alrededor de este valor medio de una forma aleatoria, sin alterar el valor medio.

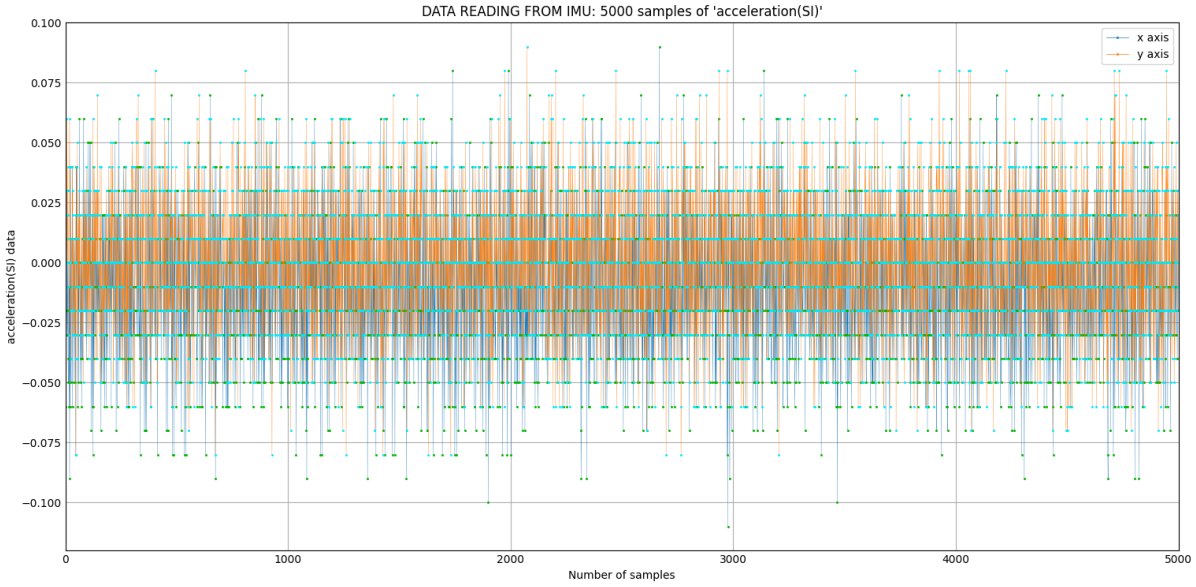

Figura 2.9: Medida del acelerómetro del módulo GY-87 en los ejes x e y.

En cuanto a la fiabilidad, aún puede mejorarse más con una calibración más exhaustiva para que la medida sea una representación fiel de la magnitud real que se quiere medir.

Aunque siendo el valor medio de las muestras $9{,}79 m/s^2$ se puede dar este valor por válido, ya que el valor de $g = 9{,}81$ representa el valor medio de la gravedad de la tierra, pero este puede variar alrededor de un 0.7 % y por tanto la medición está dentro del margen aceptable para la aplicación que se pretende realizar.

En la Figura 2.9 se muestra la medición realizada en los ejes x e y. El comportamiento del sensor es el mismo que en el eje z, la única diferencia es el valor medio medido. Esta vez los valores fluctúan alrededor del 0 ya que estos dos ejes son perpendiculares al vector de gravedad y por tanto no miden su magnitud.

Se han obtenido y se han visualizado los datos del magnetómetro y del giróscopo de la misma forma, pero dado que no se está midiendo ningún comportamiento concreto estos gráficos no aportan ningún tipo de información útil, más allá de la prueba del funcionamiento de los sensores. Para el caso de estos sensores la calibración es algo más complicada de realizar de forma manual por lo que se hizo uso del algoritmo mencionado anteriormente.

Con el propósito de comparar los resultados obtenidos con diferentes placas y lenguajes de programación se han realizado las mismas mediciones en una placa *Arduino Nano RP2040 connect*. Esta placa es compatible con el lenguaje *Circuitpython**, por lo que también se ha sustituido el lenguaje C por este, con intención de compararlos.

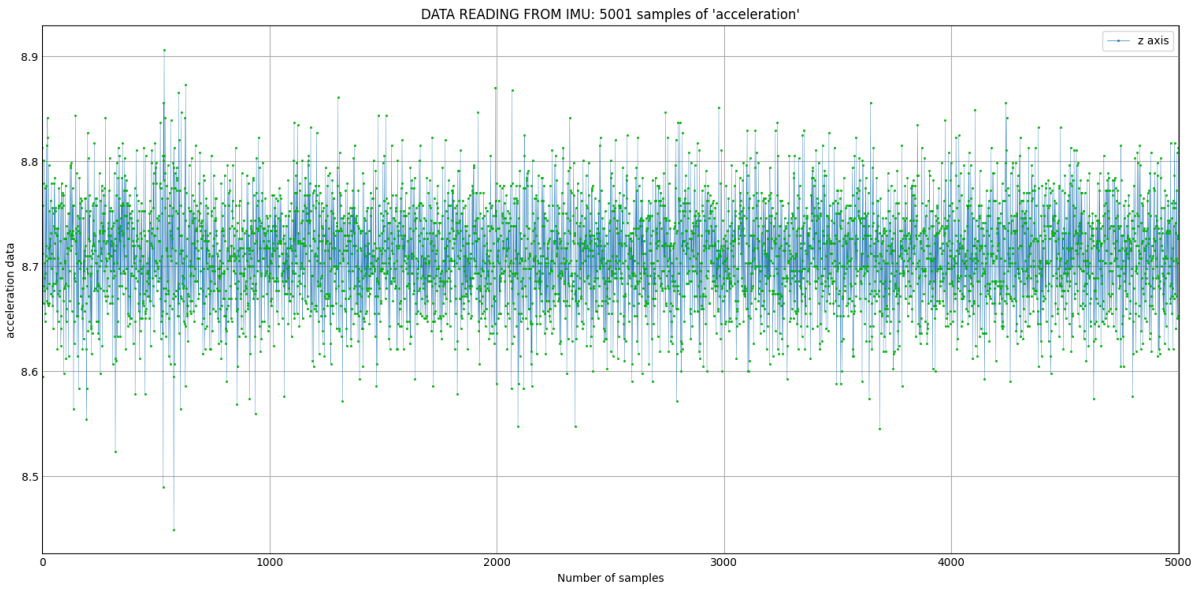

Figura 2.10: Medida del acelerómetro del módulo GY-87 en el eje z utilizando Circuitpython.

En la Figura 2.10 podemos ver una medición realizada utilizando esta nueva placa conectada al mismo sensor. A diferencia de las mediciones anteriores en este gráfico no se observa efectos de cuantificación, se puede suponer que el efecto de cuantificación tiene relación con la forma de normalizar los valores en C, ya que este efecto no se observa con los valores obtenidos directamente de los registros del sensor, y tampoco se observa

realizando los cálculos en Circuitpython.

La desviación típica sigue siendo $\pm 0{,}1 m/s^2$, pero esto no es algo crítico. En cuanto al valor de las muestras, se trata de una medición realizada antes de la calibración, por lo que su valor no es relevante. El único propósito de esta medición es comprobar si existe alguna diferencia determinante respecto a la medición realizada en la Figura 2.8.

Observaciones y problemas encontrados

A lo largo del proceso de visualización y obtención de muestras se han encontrado varios problemas y se han realizado observaciones a tener en cuenta a la hora de tomar decisiones relacionadas con el *Software* y *Hardware*. En este apartado se mencionara y explicarán los problemas y observaciones más relevantes.

Gestión de la comunicación I2C

Las medidas realizadas en el apartado anterior han sido realizadas de forma individual, sin embargo a la hora de intentar hacer uso de los 3 sensores simultáneamente surgieron errores con la comunicación I2C y la gestión de interrupciones. El problema surge al intentar usar el magnetómetro junto con el acelerómetro y giróscopo. Se ha conseguido realizar la comunicación pero los datos del magnetómetro no eran recibidos. Sin embargo si es posible hacer uso de los 3 sensores gestionando las interrupciones manualmente y al parecer sin hacer uso del procesado que realizaba el chip MPU6050.

Dado que el procesamiento del chip fusiona los 3 sensores para obtener medidas muy precisas y no usarlo conlleva tener que gestionar todo el procesado y fusión de sensores y probablemente con un resultado peor se descarto esta solución.

Dado que se también se han realizado pruebas con una placa compatible con Circuitpython, se ha decidido utilizar esta placa más moderna y hacer uso de otro tipo de librerías incluso de un sensor más moderno.

Hardware y librerías obsoletas

Se siguieron haciendo pruebas con diferentes combinaciones de hardware con el objetivo de encontrar la mejor opción. Entre otros con el Arduino Nano RP2040 connect, el que a partir de ahora será referenciado como Arduino Nano. Esta placa soporta Circuitpython pero también es posible su programación en C como cualquier Arduino convencional. El uso de Circuitpython es muy sencillo y resulta muy familiar por lo que su uso es preferible al uso de C, pero a pesar de que existen librerías de MPU6050 para Circuitpython, no se consiguieron librerías que funcionasen bien para los sensores BMP180 y HMC5883, pues son librerías pensadas para Raspberry-s y no para Circuitpython, se intentó crear una librería propia combinando las 3 pero no hubo éxito. La idea era utilizar las partes funcionales de las librerías encontradas, ya que el desarrollo de estas librerías desde cero, se encuentra fuera de los objetivos de este proyecto y consumiría demasiado tiempo.

Como solución se intentó de nuevo volver a usar el lenguaje de programación C, pero al intentar cargar el código previamente utilizado resultó no ser compatible. La arquitectura interna del Arduino Nano es diferente a la del Arduino Mega, este último tiene un núcleo AVR, propio de los Arduinos clásicos, mientras que el Arduino Nano tiene un núcleo "Mbed OS". Por tanto el problema se encuentra en las librerías utilizadas en el código, las cuales solo son compatibles con los núcleos AVR, encontrar una librería moderna para estos sensores en concreto fue imposible, ya que están obsoletos y por tanto sus librerías no se mantienen actualizadas.

Puesto que este sensor no había sido elegido por ningún motivo en particular y muchos de los problemas están relacionados con sus incompatibilidades, se optó por una solución rápida y se decidió sustituir este sensor por otro más moderno, *Adafruit 9-DOF Absolute Orientation IMU* [2], el cual facilitara el trabajo.

Circuitpython

Circuitpython es un lenguaje de programación simplificado y basado en Python, pensado para su uso sencillo, lo cual es algo positivo. Sin embargo esto hace que sus librerías sean muy simples y en la mayoría de los casos no implementan todas las funciones que el hardware tiene. Además muchas de las librerías están en formato ".mpy", un formato pre-compilado para ocupar un espacio reducido al ser guardado en la memoria de un Arduino, las librerías son cargadas por el Arduino pero su archivo no se puede abrir de una forma convencional, por lo que sin el archivo "*.py*" completo no se puede acceder al fichero para consultar el código, al menos de forma inmediata.

En las ocasiones en las que se desconoce la función de algún método, no poder revisar la librería complica su entendimiento. Esto es especialmente molesto cuando existen diferentes versiones de la misma librería con sus métodos actualizados, ya que al buscar documentación en la red sobre alguna función concreta se obtiene información oficial que es correcta, pero pertenece a versiones diferentes de la misma librería.

Capítulo 3

Aplicación experimental

Como aplicación real de un sensor IMU se ha propuesto el montaje de un sistema dinámico compuesto principalmente por un balancín y un robot. Este irá dotado del sensor IMU y tendrá que valerse de las mediciones de este para estabilizar el sistema.

Para ello se diseñará el robot con los componentes que se crean necesarios y se implementará un sistema de control en lazo cerrado, donde se usará el IMU para medir la variable que se usará como realimentación. Los motores del robot serán los actuadores del sistema y tendrán la tarea de llevar el sistema al punto deseado para lograr el equilibrio.

El balancín será diseñado mediante el software Fusión 360 y será impreso en 3D.

3.1. Modelo teórico

A continuación se presenta el modelo matemático del sistema dinámico que se pretende controlar. El sistema está formado por un balancín y dos masas como puede verse en la Figura 3.1, también están representados dos sistemas de referencia, el sistema de referencia del balancín, S^*, y el sistema de referencia fijo, S.

La masa m_2 representa una masa fija a una distancia d, mientras que la masa m_1 representa la masa del robot a una distancia x del punto de equilibrio del sistema x_e. La posición absoluta del robot, $\vec{x'}$, en el sistema S^* está determinada por la siguiente ecuación,

$$\vec{x'} = \vec{x_e} + \vec{x} \tag{3.1}$$

sin embargo para el análisis de la dinámica la variable de interés es x.

Los dos sistemas de referencia tienen el mismo punto de referencia pero el sistema S^* rota respecto a S. En la ecuación 3.2 aparecen los vectores de posición de cada una de las masas, obtenidos en el sistema S^*.

$$\begin{cases} \vec{r_1} = x_e\hat{i} + \xi\hat{j} \\ \vec{r_2} = -d\hat{i} + \xi\hat{j} \end{cases} \tag{3.2}$$

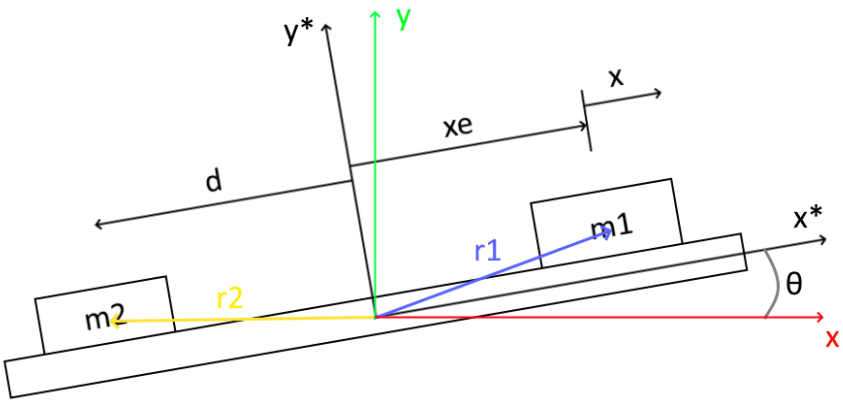

Figura 3.1: Diagrama del balancín y las masas.

ξ es la variable que hace referencia a la mitad del grosor del balancín más la mitad del grosor de las masas, sin embargo esta variable es despreciable en comparación al largo del balancín, dando lugar a la siguiente ecuación,

$$\begin{cases} \vec{r_1} \approx x_e \hat{i} \\ \vec{r_2} \approx -d\hat{i} \end{cases} \tag{3.3}$$

De esta manera los vectores quedan compuestos solo por el componente $\hat{i}$ y el cambio de sistema de referencia se vuelve más simple.

Para obtener la ecuación dinámica del sistema se calcula la suma de los torques que sufre el balancín. Para ello se ha planteado la ecuación 3.4 siendo I la inercia de todo el conjunto rotatorio y b la fricción.

$$\sum \vec{M} = I\ddot{\theta} + b\dot{\theta} \tag{3.4}$$

La inercia del sistema completo está descrita por la siguiente ecuación donde m_B es la masa del propio balancín y L su longitud.

$$I = \frac{1}{12} m_B L^2 + m_1 {x_e}^2 + m_2 d^2 \quad ; \quad \begin{cases} m_B = 667g \pm 1g \\ m_1 = 251g \pm 1g \end{cases} \tag{3.5}$$

Aplicando a la ecuación 3.4 los vectores de posición de las masas 3.3 en el sistema de referencia S se obtiene,

$$x_e m_1 g \cos\theta - d m_2 g \cos\theta = I\ddot{\theta} + b\dot{\theta} \tag{3.6}$$

Resolviendo la ecuación 3.6 para el caso $\dot{\theta} = 0$ y $\ddot{\theta} = 0$ se obtiene la posición de equilibrio x_e y se reescribe la ecuación 3.6 en función de x.

$$\ddot{\theta} = \frac{g}{I} \cos\theta m_1 x - \frac{b}{I}\dot{\theta} \quad ; \quad \vec{x_e} = \frac{m_2}{m_1}\vec{d} \tag{3.7}$$

La ecuación obtenida es una ecuación diferencial no lineal pero se han asumido oscilaciones pequeñas y se ha linealizado la ecuación en torno al punto de equilibrio x_e.

$$\ddot{\theta} = \frac{g}{I} m_1 x - \frac{b}{I}\dot{\theta} \tag{3.8}$$

Como resultado se obtiene la ecuación diferencial de segundo orden linealizada 3.8, esta es la ecuación dinámica que representa el comportamiento del sistema, la variable del sistema a controlar es θ, el ángulo del balancín, para ello se realimenta dicha variable utilizando el sensor IMU para su obtención y mediante el control realizado se modifica el valor de x hasta lograr el equilibrio ($\theta = 0$).

Este modelo es útil para entender el comportamiento del sistema, no obstante es un modelo incompleto, ya que faltan los modelos de los actuadores y sensores del control:

- El sensor que se utilizará para medir θ no será ideal por lo que debe tenerse en cuenta como un elemento más del control, el cual puede añadir ruido, retardo, etc.
- Los motores que se utilizaran como actuadores tampoco serán ideales, estos definirán prácticamente la dinámica del robot, en función de la tensión aplicada a los motores.

3.2. Robot

En este apartado se presenta el robot montado para el experimento, en la Figura 3.2 puede verse la versión final, el robot está montado en un chasis Zumo de la marca Pololu con motores "50:1 Micro Metal Gearmotor HP 6V"de la misma marca.

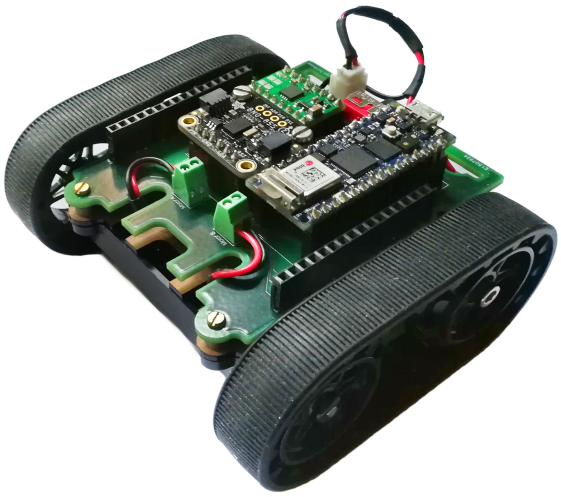

Figura 3.2: Versión terminada del robot montado.

El circuito electrónico esta compuesto por el microcontrolador Arduino Nano RP2040 connect, el modulo controlador de motores DRV8833 de la marca Pololu y el sensor Adafruit 9-DOF Absolute Orientation IMU, todos los módulos junto con los conectores han sido montados en una placa PCB y alimentados con cuatro celdas AA de NiMH. Esta placa ha sido diseñada a medida y va anclada al chasis del robot.

3.2.1. Componentes de alto nivel

El circuito electrónico es una parte fundamental del robot, ya que es el encargado de controlar todas las funciones del robot.

Para ello posee un microcontrolador programable que lee los datos del IMU y actúa en consecuencia comunicándose mediante un conjunto de señales con el controlador de motores. Este se encarga de mandar la corriente necesaria a los motores en función de las órdenes del microcontrolador.

Este apartado se describe con detalle el Hardware utilizado para crear el circuito electrónico.

Arduino Nano RP2040 connect

Tras haber obtenido mediciones con el Arduino Mega 2560 y con esta placa, se ha decidido continuar el proyecto utilizando esta. Aunque es cierto que las mediciones con ambas placas son muy similares ya que se usa el mismo sensor, el código en Circuitpython resulta más simple de usar y al no tener cuantificación podemos obtener una mayor resolución.

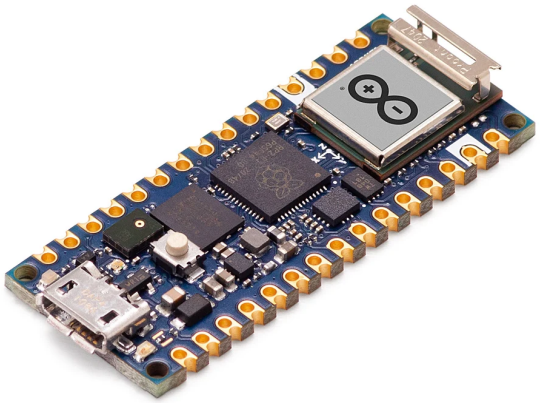

Figura 3.3: Imagen de la placa Arduino Nano RP2040 connect [10].

Sin embargo esto no es algo crítico y el motivo principal por lo que hemos preferido esta placa frente al Arduino Mega es su conectividad y su tamaño reducido. Esta placa dispone de un módulo Wifi y Bluetooth, lo que permite su control y monitorización a distancia. Tanto su tamaño como su conectividad serán muy útiles a la hora de implementar el circuito en el robot ya que el circuito podrá ser muy compacto y de peso muy reducido, además este tendrá la capacidad de conectarse a una red Wifi pudiendo así ser controlado de forma remota o mantener una comunicación constante.

Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055

Este módulo de Adafruit hace uso del chip BNO055 [2], el cual es un IMU que utiliza la tecnología de fusión de sensores para obtener información de orientación absoluta. Este

módulo ha sido elegido como una alternativa mejor al módulo probado anteriormente, es un sensor muy popular en la comunidad para el cual se puede encontrar mucha documentación y librerías de Circuitpython actualizadas. Además su diseño con agujeros de montaje permite que sea anclado a una placa de circuito, logrando así evitar en gran medida movimientos y vibraciones indeseadas.

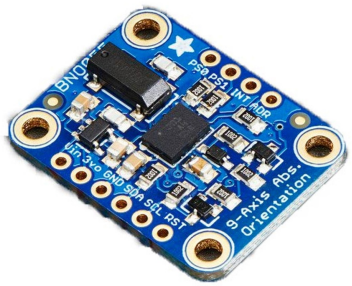

Figura 3.4: Modulo Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout.

Controlador de motores DRV8833 de Pololu

Para poder controlar la velocidad y la dirección de rotación de los motores es necesario un controlador de motores. Un motor puede girar a ambas direcciones alimentándolo con una polarización u otra, el controlador permite cambiar esa polarización sin necesidad de cambiar los cables, gracias a un circuito llamado puente en H compuesto por transistores y en la mayoría de los casos por diodos de protección [11].

Para este robot se ha elegido el módulo controlador de motores DRV8833 [11] de Pololu, este módulo utiliza el chip DRV8833 e incluye protecciones contra el exceso de corriente, tensión o temperatura. Estas protecciones harán que el circuito se bloquee y deje de funcionar de forma preventiva.

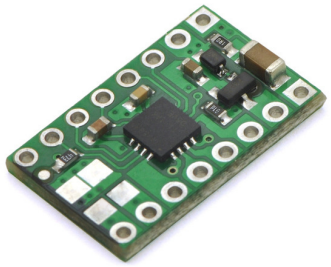

Figura 3.5: Controlador de motores DRV8833 de Pololu

Tiene un rango de tensión de operación de 2,7 V a 10,8 V y es capaz de proporcionar 1,2 A de forma continua a dos motores simultáneamente, lo que es suficiente para un ligero robot, es capaz de controlar dos motores DC y ocupa un espacio muy reducido.

En la Figura 3.6 se muestra un esquema simplificado del funcionamiento del controlador de motores para un solo motor. La tensión aplicada al motor es una fracción de la tensión de alimentación del DRV8833. Esta fracción es controlada por el ciclo de trabajo de la señal PWM. En este caso el controlador encargado de enviar y leer las señales será un Arduino

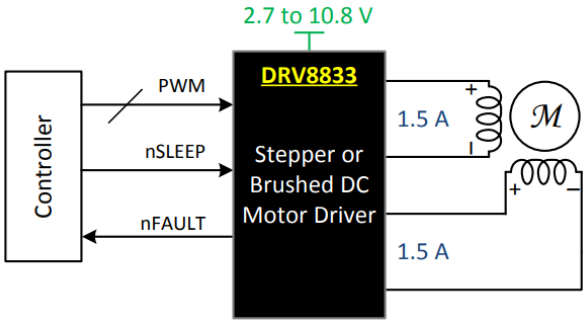

Figura 3.6: Esquema simplificado del circuito controlador de motores, obtenido de la hoja de datos técnica de Texas Instruments Incorporated [11].

que generará la señal PWM que será enviada al controlador de motores, controlara la señal nSLEEP que habilita o deshabilita el funcionamiento y leerá la señal nFAULT que nos indica si hay errores.

xIN1	xIN2	FUNCTION
PWM	0	Forward PWM, fast decay
1	PWM	Forward PWM, slow decay
0	PWM	Reverse PWM, fast decay
PWM	1	Reverse PWM, slow decay

Figura 3.7: Tabla lógica para el control de motores [11].

En la figura 3.7 se muestra la tabla lógica para el control de los motores, cada motor se controla mediante dos señales lógicas que determinan la dirección de giro del motor o si el motor está parado. Estas dos señales suelen ser binarias, sin embargo este controlador permite usar directamente la señal PWM como señal de control, controlando también la velocidad.

Motores "50:1 Micro Metal Gearmotor HP 6V"de Pololu

Se han elegido estos motores de la marca Pololu ya que los motores deben de tener unas dimensiones concretas para encajar en el chasis, como su nombre indica funciona a 6V y ofrecen una velocidad angular de 590 RPMs. Se han necesitado dos de ellos, uno a cada lado.

Pololu ofrece una amplia gama de motores con diferentes prestaciones en cuanto a velocidad y torque. Dado que el robot es pequeño se ha priorizado la velocidad a la fuerza, con intención de obtener una respuesta más rápida por parte del robot.

Celdas de alimentación AA de NiMH

Para alimentar el robot se han utilizado cuatro celdas AA de NiMH, ya que el chasis incluye la caja para insertar cuatro baterías de este tipo. Estas tienen un valor nominal de 1,2 V y una capacidad de 2200 mAh, se espera que cada celda totalmente cargada proporcione aproximadamente 1,5 V, que en total son 6 V. En cuanto a la capacidad total

se mantiene en 2200 mAh, ya que las celdas están conectadas en serie.

Las baterías son capaces de proporcionar 2,2 A de forma continua durante 1 hora pero su duración depende del consumo del robot, podría durar 2 horas si consumiese 1,1 A o 30 minutos si su consumo fuera de 4,4 A.

Para cargar el paquete de baterías es necesario realizar una carga balanceada, para que cada celda quede totalmente cargada y la tensión entre ellas esté equilibrada. Esto supone un mayor rendimiento y alargar la vida útil de las baterías [12].

Por suerte la carga balanceada de las baterías de NiMH es muy sencilla [13], ya que este tipo de baterías disipa en forma calor el exceso de energía, por lo que si una de las celdas se carga antes que otra, empezará a disipar energía mientras el resto de celdas continúa con su carga.

Hay muchas formas de cargar las baterías y detectar cuando están totalmente cargadas, en este caso se ha utilizado una carga estándar a una velocidad de carga de C/10. Esto consiste en cargar las baterías con una corriente equivalente a su capacidad total entre 10, a este ritmo de carga las baterías son capaces de disipar la energía en exceso en forma de calor de forma segura sin dañar las baterías. La carga suele durar entre 14 y 16 horas de media, ya que se le añade un tiempo extra a las 10 horas que tardaría de forma ideal debido a que el rendimiento de la transferencia de energía no es del 100 %.

3.2.2. Prototipo

Una vez seleccionados los componentes se ha realizado un prototipo del circuito para comprobar que su funcionamiento es el correcto y empezar a realizar pruebas de control con el robot antes de montar la versión final.

El esquema del circuito puede verse en la Figura 3.8, para el montaje del prototipo no se han tenido en cuenta los 32 pines de conexión que se encuentran fuera del bloque "$MAIN$".

Tras probar cada componente por separado en una placa de pruebas y verificar su correcto funcionamiento se ha realizado la misma prueba con todo el circuito completo. Se ha verificado de nuevo su correcto funcionamiento y se ha soldado el prototipo en una placa de prototipado para después poder realizar el montaje en el chasis.

En la Figura 3.9 pueden verse las soldaduras y conexiones del prototipo. A pesar de que en las pruebas realizadas el circuito ha funcionado correctamente, tras haber sido soldado en la placa este dejo de funcionar.

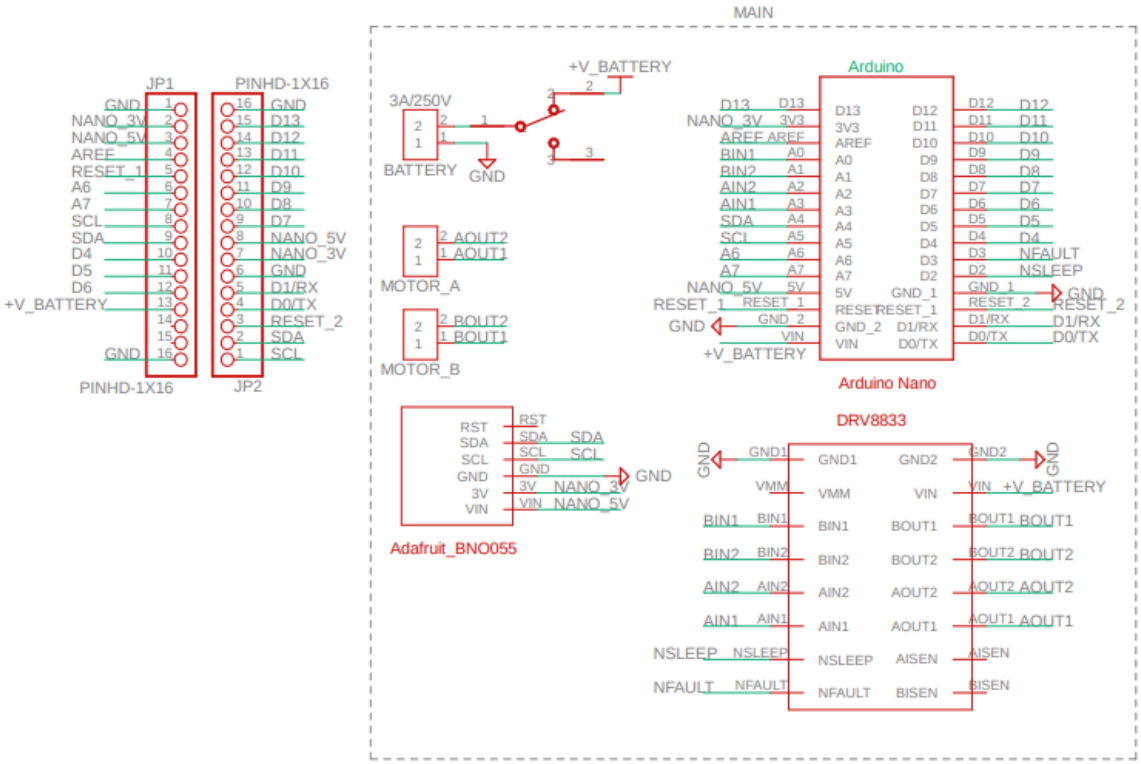

Figura 3.8: Esquema eléctrico del circuito.

Con intención de encontrar el error se han realizado varias comprobaciones básicas, como revisar que el cableado sea correcto y comprobar una a una cada conexión con un multímetro en busca de cortocircuitos. Se ha determinado que el error podía haber sido causado por alimentación insuficiente, picos de tensión o incluso interferencias.

Finalmente tras descartar el resto de causas posibles se ha llegado a la conclusión de que el error estaba relacionado con las interferencias. El error fue detectado casualmente tras volver a realizar pruebas en una placa de pruebas, al realizar contacto con uno de los pines se ha observado que el Arduino repentinamente deja de funcionar y se apaga por un momento, este pin parece ser el pin de reset, por lo que el fallo cobra sentido.

Como puede apreciarse en la Figura 3.9, esta placa está distribuida en filas de cobre lo que puede estar actuando en forma de antena, la longitud es una magnitud crítica que determina la frecuencia que puede emitir o recibir una antena, si la longitud de una antena se disminuye esa frecuencia aumenta, por lo que como solución rápida se ha optado por perforar uno de los orificios de la placa hasta eliminar la conexión, de esta forma se ha dividido la pista de cobre del pin de reset en dos.

Esto soluciona el error y el circuito empieza funcionar correctamente, por lo que se puede deducir que la causa del error era una interferencia que estaba causando que el Arduino se resetease constantemente. Los pines del Arduino cuentan con condensadores que actúan como filtros para ciertas frecuencias, pero la frecuencia que estaba causando las interfe-

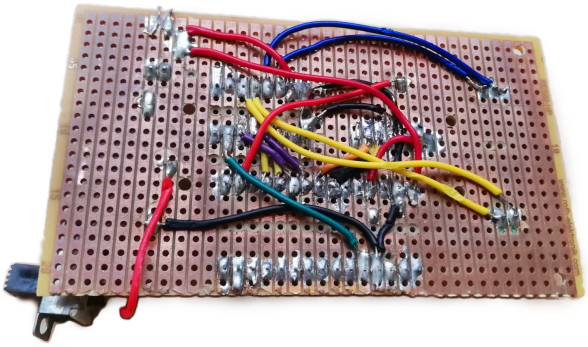

Figura 3.9: Vista inferior del prototipo del circuito.

rencias estaba por debajo de la frecuencia de corte del filtro por lo que no estaba siendo filtrada.

La capacidad de los equipos electrónicos para funcionar en presencia de interferencias de otros equipos cercanos es una característica más de estos y se denomina *ElectroMagnetic Compatibility*, (EMC).

Una vez solucionado el error, se ha montado el prototipo en el chasis para comenzar con las pruebas de control, el montaje puede verse en la Figura 3.10, el Arduino y los módulos se conectan en los conectores que pueden verse.

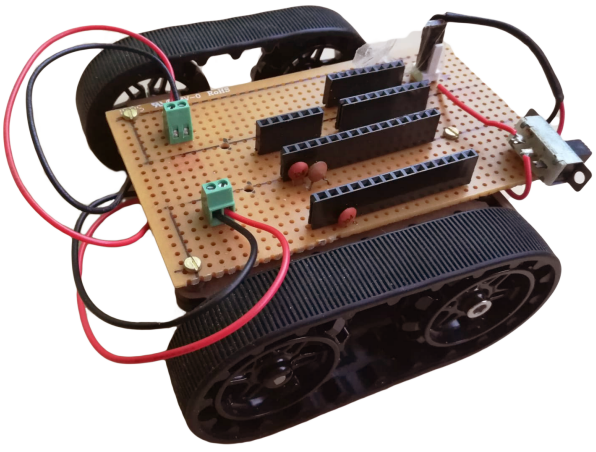

Figura 3.10: Montaje del prototipo.

3.2.3. Diseño PCB

Después de comprobar el correcto funcionamiento del prototipo, se ha diseñado una PCB de la versión final del circuito, para ello se ha utilizado el Software Fusion 360. Este software nos permite realizar un diseño electrónico y un diseño mecánico en el mismo entorno, esto resulta muy útil ya que es posible diseñar la placa de circuito impreso a la medida exacta para que se adapte perfectamente en el chasis del robot.

Creación de componentes

El primer paso para crear una placa PCB es el crear el esquema del circuito eléctrico, el esquema puede verse en la Figura 3.8, en el esquema pueden verse los diferentes componentes y sus conexiones.

Dado que los módulos que se utilizan en el robot no son componentes genéricos que vienen incluidos en las librerías se crearán los componentes nuevos en una librería propia. De esta manera es posible crear cualquier componente personalizado. Para ello es necesario crear un símbolo y una huella, también es posible adjudicar un modelado CAD al componente de forma opcional.

1. Creación del símbolo:

   El símbolo es la figura que aparece en el esquema eléctrico representando el componente, no hay requisitos en cuanto a la forma, pero sí debe incluir los pines del componente para poder realizar las conexiones.

2. Creación de la huella:

   La huella es la representación del componente en la placa de circuito impreso, en este caso la forma de la figura es más importante y debe representar las dimensiones del componente, de esta forma conociendo el espacio que ocupa cada componente su posicionamiento en la placa puede ser realizado de forma sencilla sin que estos se solapen entre ellos.

   Más importante aún es la representación de los pines, que indica la ubicación de los orificios o pads de la PCB, es necesario que sean colocados con el espaciado correcto. Esta información puede encontrarse en las hojas de datos técnicas de cada componente un ejemplo de estas dimensiones puede verse en la Figura 3.11, además es necesario vincular cada pad con el pin del componente correspondiente.

3. Vincular modelado CAD:

   Por último, es posible realizar un modelo 3D o vincular un modelo CAD existente y fiel con la realidad en cuanto a dimensiones. Este modelo debe ser alineado con la huella del componente para que se visualice de forma correcta.

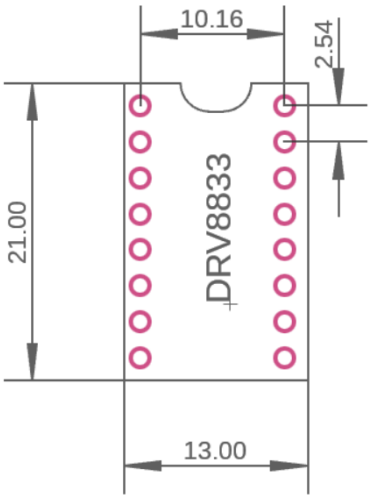

Figura 3.11: Dimensiones de la huella del módulo DRV8833 en milímetros.

Una vez se tenga el esquema eléctrico con todos los componentes y sus huellas, es posible comenzar con la distribución de los componentes en la placa.

Organización de los componentes en la placa

De forma predeterminada todas las huellas de los componentes aparecen sin conectar en una placa con dimensiones cuadradas. Para diseñar una PCB a medida se ha utilizado un archivo de dibujo "*.dxf*" del chasis del robot. Este archivo contiene información sobre los orificios y las dimensiones de la placa de montaje del chasis, que es donde la PCB irá montada.

Con la ayuda del archivo mencionado se define el contorno de la placa y se colocan dentro los componentes de forma que el IMU esté lo más centrado posible en la placa. La posición del resto de componentes se determina en función de varios factores: la accesibilidad en el caso del interruptor o el puerto USB del Arduino y la cercanía a sus conexiones en el caso de los conectores de los motores. En general se tienen en cuenta las conexiones que se deben realizar a la hora de orientar o posicionar cada componente, pues hay que tener en cuenta que las pistas no pueden cruzarse y se debe intentar conectar todo de la forma más óptima posible, es decir con las pistas más cortas o más directas posibles.

Enrutado de pistas

Las placas pueden tener múltiples capas de enrutado, en este caso se ha realizado el diseño utilizando tan solo las dos capas básicas, la superior e inferior. El enrutado se puede hacer en cualquiera de estas capas, incluso se puede pasar entre ellas utilizando vías para evitar así que las pistas se crucen.

Antes de empezar a enrutar se debe tener en cuenta la cantidad de corriente que pasará

por cada pista para que esta no se queme. La mayoría de pistas son digitales por lo que pasara poca corriente por ellas, estas serán de 0.3 mm de grosor, sin embargo las pistas de alimentación serán de 1 mm de grosor y las pistas de los motores de 0.5 mm cada una.

El proceso de enrutado puede hacerse de forma manual o automática pero para obtener un mejor resultado se hará de forma manual.

Por último, tras haber realizado el enrutado se convierten la capa superior e inferior en planos de tierra, esto hace que todas las tierras sean comunes y reduce los efectos de las interferencias. En la Figura 3.12 puede verse un modelo CAD del resultado de la PCB,

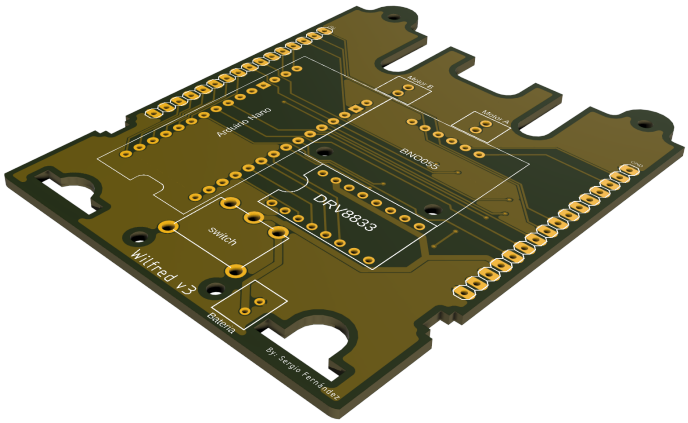

Figura 3.12: Modelo CAD de la placa de circuito impreso.

pueden apreciarse los pads de soldadura y la organización de los componentes gracias a la serigrafía. En los dos extremos laterales de la placa pueden verse 16 conexiones para pines a cada lado, esto ha sido diseñado con la intención de incluir un *hat* en el futuro. Esto permite expandir la placa PCB actual añadiendo un segundo piso con *Hardware* adicional, sin necesidad de modificar el diseño PCB. Por último en la Figura 3.13 se muestra un

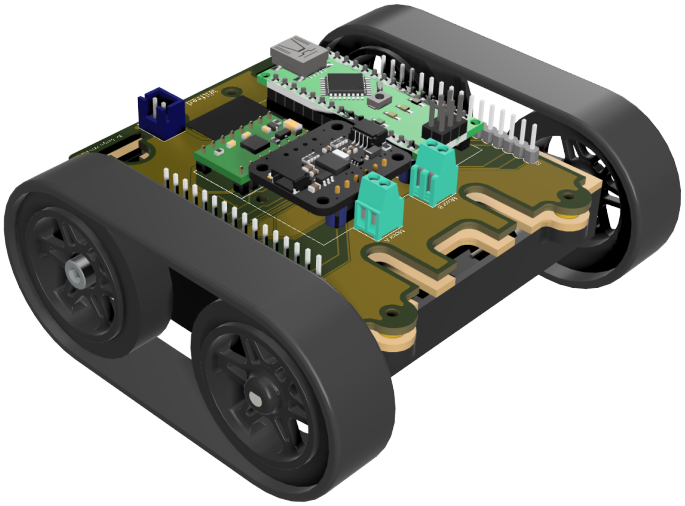

Figura 3.13: Modelo CAD de la versión final del robot.

renderizado de CAD del resultado final del robot, la placa PCB con todos los componen-

tes soldados se encuentra atornillada al chasis sobre unos espaciadores impresos en 3D. Mediante este renderizado es posible comprobar que los elementos sobre la PCB no estén superpuestos además de visualizar el resultado antes de su fabricación.

3.2.4. Software

En este apartado se hablará en detalle del *Software* y código utilizado para implementar las funciones del robot. Para el desarrollo del código y la programación en *Circuitpython* del robot se ha utilizado el programa *Visual Studio Code*.

Circuitpython y Python

Tras haber utilizado C y Circuitpython, se ha decidido continuar usando Circuitpython y Python. Esta decisión no ha sido tomada basándose en ningún aspecto técnico concreto, pero se ha observado que la facilidad de estos lenguajes acelera el proceso de programación, ofreciendo más flexibilidad y margen a la hora de realizar programas más complejos.

El código que ha de ser ejecutado en el Arduino se ha programado en Circuitpython usando librerías de Adafruit en su gran mayoría u otras proporcionadas por la comunidad, en cambio el resto de programas, que estarán alojados en el PC, han sido programados en Python convencional, usando cualquier librería disponible, sin reparar en el espacio de memoria que estas ocupan.

Código

El robot es capaz de realizar tanto movimientos básicos como movimientos más complejos controlados en lazo cerrado por el IMU. Además es capaz de conectarse a un Broker MQTT mediante Wifi para comunicarse con otros dispositivos. Esto permite que pueda ser controlado de forma inalámbrica a través de comandos e incluso recibir datos del robot en tiempo real.

Para poder implementar estas funciones se han creado una serie clases, *Robot()*, *Motor()*, *PID_Controller()*, *MQTT_Manager()*, *Command_Handler()*, etc.

A continuación se explican las funciones más relevantes, para ello se ha dividido el código en tres grupos principales, el código encargado del control del robot, el encargado de gestionar la comunicación MQTT y el encargado del controlador PID.

Control del Robot

Las clases encargadas de gestionar los movimientos y las funciones del robot son las siguientes:

- *Motor():*

Esta clase hace de interfaz con el controlador de motores DRV8833, permitiendo controlar la potencia y dirección de rotación de los motores, al definir un objeto Motor() se le asignan sus dos pines de control mediante los cuales se envían las señales PWM de 50 KHz. Cada motor tiene una variable normalizada de potencia, cuando a un motor se le ordena moverse, una función se encarga de realizar la conversión de esta variable a un valor entre 0 y 65535 y se utiliza para determinar el duty_cycle de la señal PWM.

- *Robot():*

  En esta clase se definen las funciones encargadas de los movimientos del robot que utilizan para ello los objetos Motor(), estas funciones se encargan de hacer girar cada motor en la dirección correcta para realizar el movimiento deseado y aplican la potencia que reciben como parámetro. Un objeto Robot() tiene varios atributos, entre ellos el objeto IMU y el objeto MQTT, esta clase también es la encargada de calibrar el IMU y realizar la conexión MQTT.

Comunicación MQTT

La comunicación MQTT es una de las funciones más importantes del robot, ya que es la que se usa para mandar comandos de control al robot y para recibir información en tiempo real de este. El robot tiene funciones que se encargan de enviar la información en un formato concreto que después es interpretado desde otro dispositivo para realizar los gráficos en vivo.

Cuando el robot está ejecutando en bucle el algoritmo encargado de estabilizarlo, cada bucle se envía mediante MQTT un conjunto de variables, entre ellas, la señal de control, la señal de error, el ángulo medido, etc.

Para gestionar esta comunicación se utilizan dos clases:

- *MQTT_Manager():*

  En esta clase se configura el *socket* utilizado para la comunicación, para ello se le pasa al *socket* el objeto *esp* el cual se controla mediante protocolo *spi*, después se configura el cliente *MQTT* otorgándole el *Broker* al que debe conectarse y su puerto. Una vez realizada la configuración se definen las funciones que realizan las operaciones básicas, como conectarse, suscribirse a un *topic*, mandar mensajes o publicar mensajes.

  Por último se definen las funciones que deben ser llamadas cada vez que ocurre algún evento relacionado con alguna operación, se define la función *on_message* que es llamada cada vez que se recibe un mensaje, esta función coge el *topic* y el mensaje recibidos y se los pasa como argumentos a otra clase llamada *Command_Handler* para que esta los gestione.

- *Command_Handler:*

  Esta clase es la clase encargada de recibir los mensajes y ver si se corresponden con alguno de los comandos definidos. En caso de que así sea ejecuta las funciones del robot que le corresponden a cada comando.

  Los comandos también incluyen los argumentos necesarios para cada función, por lo que esta clase también se encarga de comprobar que estos sean los correctos y llama a las funciones utilizando estos argumentos, en caso de no recibir argumentos utiliza argumentos predefinidos en caso de ser posible, o manda un mensaje informando de que el comando recibido es erróneo.

Controlador PID

Para controlar la dinámica del sistema se usa un controlador PID [14] en lazo cerrado, el PID consta de tres partes, la proporcional, C_p, la integral, C_i, y la derivativa, C_d. La contribución de cada parte viene descrita por las siguientes ecuaciones,

$$C_p = K_p e(t) \tag{3.9}$$

$$C_i = K_i \int e(\tau) d\tau \tag{3.10}$$

$$C_d = K_d \frac{de(t)}{dt} \tag{3.11}$$

donde $e(t)$ es el error del sistema, que se define como $e(t) = y(t) - r(t)$, siendo $y(t)$ la salida del sistema realimentada y $r(t)$ la referencia a seguir.

Las ganancias del PID se definen de la siguiente manera,

$$K_p = K \quad ; \quad K_i = \frac{K}{T_i} \quad ; \quad K_d = KT_d \tag{3.12}$$

La señal de control, u, que se obtiene a la salida del PID es la suma de las contribuciones de cada parte.

$$u = C_p + C_i + C_d \tag{3.13}$$

Para poder diseñar un control útil y eficaz en una aplicación real y no lineal se han hecho ciertas modificaciones al PID básico. Se ha implementado un mecanismo *antiwindup* y un filtrado de consigna para reducir el ruido de la parte derivativa. El controlador realizado puede verse en la Figura 3.14.

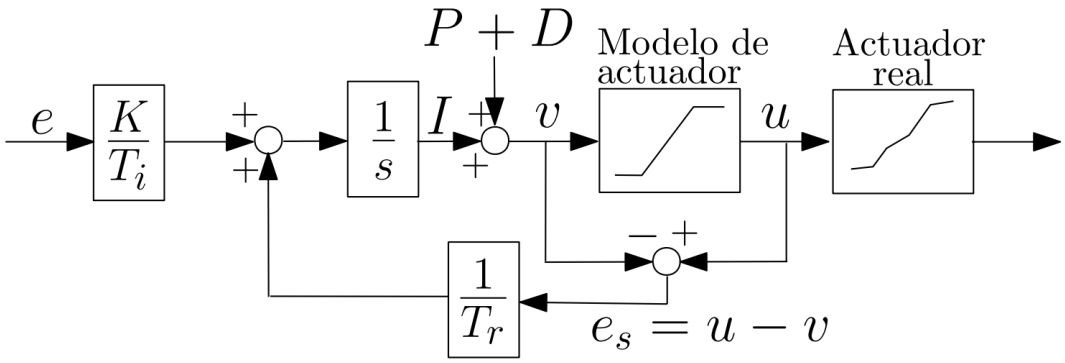

Figura 3.14: Diagrama de bloques del controlador realizado [15].

La función del mecanismo *antiwindup* consiste en reducir la parte integral del PID en caso de que el controlador esté saturado ya que de lo contrario el término integral acumulado afectaría de forma negativa volviendo el controlador más lento.

Para que el mecanismo funcione de forma correcta se ha de elegir una ganancia de error adecuada, un valor típico es el siguiente [15],

$$T_r = \sqrt{T_i T_d} \tag{3.14}$$

Como puede verse en la Figura 3.14 se ha utilizado un modelo de los actuadores que caracteriza las no linealidades de los motores, es decir la zona muerta y la saturación. Estas no linealidades han sido medidas previamente mediante pruebas experimentales.

En cuanto al filtrado de consigna, se ha implementado un filtro RC de forma digital. Para ello se ha obtenido la función de transferencia del filtro y después mediante el método *Zero-order Holder*, (ZOH), se ha discretizado la función.

$$C_d[k] = C_d[k-1]e^{-2\pi T f_c} + C_{din}[k-1](1 - e^{-2\pi T f_c}) \tag{3.15}$$

$$f_c = \frac{1}{2\pi RC} \tag{3.16}$$

En la ecuación 3.15 puede verse el resultado, la frecuencia de corte puede establecerse calculando los valores R y C y el periodo de muestreo T.

A continuación en el Listing 3.1 se muestra la función que se encarga de estabilizar el robot en el balancín. En el código pueden apreciarse los parámetros que recibe esta función, entre ellos se encuentran las ganancias del PID, el ángulo de referencia y la tiempo de duración del algoritmo. Este fragmento de código programado en Python se muestra con el objetivo de proporcionar una idea orientativa del funcionamiento del algoritmo. Es por

ese motivo que solo se muestra este fragmento y no el codigo completo.

```
    def stabilize(self, Kp=0, Kd=0, Ki=0, a=0.9, ref_angle=0, end_t=20):
        data = ""
        sensor_f = PID_Filter(5, 0.005)
        ii = 0
        ii_rate = 50
        t0 = t = time.monotonic()
        angle_measure = self.IMU.euler[2]
        angle_measure = check_angle(angle_measure)
        controller = PID_Controller(Kp, Ki, Kd, ref_angle, angle_measure
            , a)
        while(t-t0) < end_t:
            prev_t = t
            t = time.monotonic()
            dt = t - prev_t

            if (ii-1) % ii_rate != 0:
                pid = controller.update(angle_measure, dt)

            if pid <= 0:
                self.move_forward(-pid)
            elif pid > 0:
                self.move_backward(pid)
            data += str(pid) + ":" + str(angle_measure) + "/"

            if ii % ii_rate == 0 and ii != 0:
                data = data[0:len(data)-1]
                self.mqtt.publish(data,"data")
                data = ""

            ii += 1
            angle_measure = self.IMU.euler[2]
            angle_measure = sensor_f.get_value(angle_measure)
            angle_measure = check_angle(angle_measure)

        # Al acabar se mandan las muestras restantes
        data = data[0:len(data)-1]
        self.mqtt.publish(data,"data")
        self.standby()
        return True
```

Listing 3.1: Funcion encargada de estabilizar el robot en el balancín.

3.3. Balancín

El balancín donde debe equilibrarse el robot es uno de los elementos esenciales en el experimento, con intención de tener un control total y libertad sobre el diseño este se ha impreso en 3D, el resultado final del balancín puede verse en la Figura 3.15

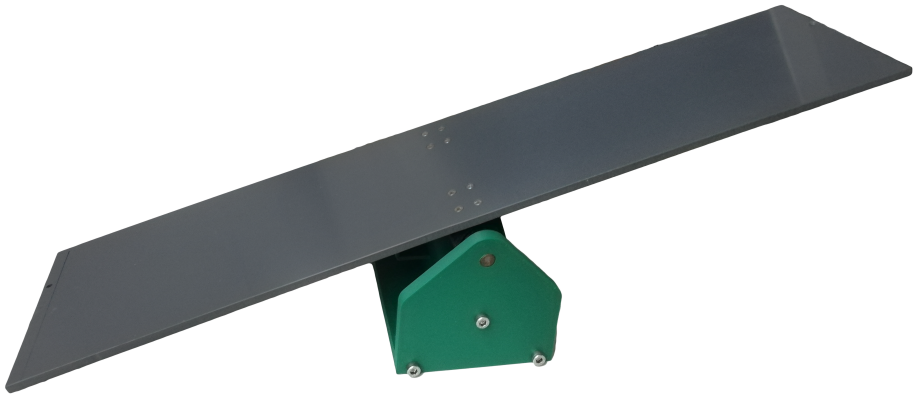

Figura 3.15: Imagen real del balancín construido.

Las dimensiones del balancín se han diseñado teniendo en cuenta el robot. Este tiene unos 10 cm de ancho, por lo que el balancín tiene un ancho de 12 cm, dejando 1 cm de margen a cada lado. El robot debe ser capaz de mantener una trayectoria relativamente recta, por lo que no debería salirse.

El plano del balancín es una plancha de plástico de 550 mm de largo, dadas sus dimensiones este elemento no ha sido impreso en 3D. Tampoco lo ha sido el eje de acero de 9.98 mm.

3.3.1. Desarrollo del modelo

Modelos iniciales del balancín

Como ocurre en cualquier diseño se empezaron a diseñar desde cero posibles modelos que poco a poco fueron evolucionando, en la Figura 3.16 se muestra el primer diseño realizado con Fusion 360.

El balancín incorporará un potenciómetro o encoder para medir su propio ángulo, de esta forma se tendrá una segunda medición del ángulo para contrastar con las mediciones del robot. Para el montaje de estos sensores en el balancín se quiere diseñar una transmisión que multiplique la rotación del balancín para así aumentar la resolución de las mediciones realizadas y ser capaces de medir variaciones de ángulos más pequeñas. Esto es importante ya que si el sistema funciona bien, el balancín se mantendrá en ángulos pequeños que tendremos que ser capaces de medir.

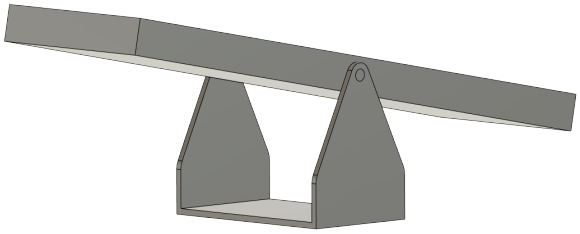

Figura 3.16: Primer modelo de la balanza.

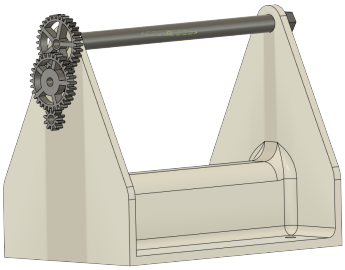

Figura 3.17: Soporte de balanza con engranajes.

El soporte del balancín se fue modificando para incorporar esta transmisión. En la Figura 3.17 se muestra uno de los modelos, que utiliza una engranajes para multiplicar por 9 la rotación del balancín. De esta forma si limitamos la rotación del balancín de forma que $\theta \in [-10, 10]^\circ$ esta rotación corresponderá a media vuelta del potenciómetro.

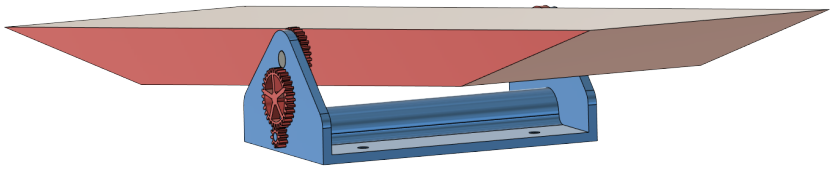

Figura 3.18: Modelo de altura reducida.

El diseño se modificó nuevamente reduciendo la altura del soporte para que el balancín se apoyase en la superficie limitando su rotación, como se muestra en la Figura 3.18, sin embargo este diseño fue descartado ya que la transmisión atravesaba una de las paredes del soporte, lo que causaría fricción extra y complicaría el diseño.

Observaciones en los modelos

Hay que tener en cuenta varios factores críticos a la hora de realizar los diseños:

1. **Resolución y tolerancias:**

   A la hora de realizar los diseños todo encaja perfectamente y se diseñan los objetos con las medidas exactas, pero a la hora de la impresión hay que tener en cuenta la resolución de la impresora y las tolerancias, el plástico al calentarse y enfriarse sufre ciertas dilataciones, por lo que se debe tener especial cuidado a la hora de diseñar partes móviles o partes que deban encajar entre ellas. Por otro lado, en los softwares

de edición se tiende a perder la noción de las medidas y es fácil diseñar partes con un grosor milimétrico sin ser consciente de lo realmente fino que es.

2. **Fricción:**

   Es conveniente minimizar la fricción del balancín todo lo posible, la fricción que genera un potenciómetro puede parecer pequeña pero al estar conectado a una transmisión esa fricción también se multiplicara, por lo que es algo a tener en cuenta, junto con la fricción de los engranajes y sus ejes.

3. **Tiempo de impresión:**

   El proceso de impresión es un proceso relativamente lento, durante el cual pueden ocurrir errores, aun siendo errores pequeños estos pueden llevar a la impresora a un mal funcionamiento causando que ésta se bloquee o que la pieza impresa quede inservible. Es por esto que durante una impresión se suele vigilar la impresora para poder suspender la impresión a tiempo en caso de fallo y reiniciar la impresión.

   Las partes de este diseño son de un tamaño considerable y el tiempo de impresión asciende a 8 horas para el soporte. Es por eso que conviene dividir el modelo en partes más pequeñas que faciliten su impresión.

Cambios en el modelo

Tras realizar estas observaciones se ha decidido realizar varios cambios,

- Sustituir los engranajes por poleas, que cumplen la misma función y no requieren de tanta precisión como los engranajes para encajar.

- Reforzar la estructura del soporte y dividirlo en varias partes. Estas partes se unirán tras la impresión usando tornillos de 5 mm.

- Utilizar un eje de acero de 9,98 mm en lugar de uno impreso en 3D, este encaja a presión y es perfectamente cilíndrico reduciendo la fricción con el plano del balancín.

- No imprimir el plano del balancín por sus grandes dimensiones, en su lugar se usará una plancha de teflón ligero de 550 mm de largo.

- Debido al pequeño grosor del plano en lugar de ser perforado para introducir el eje, será atornillado con tornillos de 5 mm a unos anclajes. Estas piezas unirán el plano con el eje permitiendo que rote a través del eje, para ello se tendrán en cuenta las tolerancias.

- Usar un *encoder* multivuelta ya que el eje de estos rota con menor fricción que el de un potenciómetro convencional. También se ha diseñado un pequeño soporte para el montaje del *encoder* en la parte interior del soporte del balancín.

Modelo final

Finalmente, se tomó la decisión de utilizar engranajes en lugar de poleas, el soporte ya estaba imprimido por lo que se diseñó el sistema de engranajes para utilizar los mismos orificios. Por ese mismo motivo el sistema de engranajes consta de dos etapas, como resultado se obtuvo una amplificación del movimiento aproximadamente de unas 25 veces el movimiento del balancín. El resultado puede verse en la Figura 3.19.

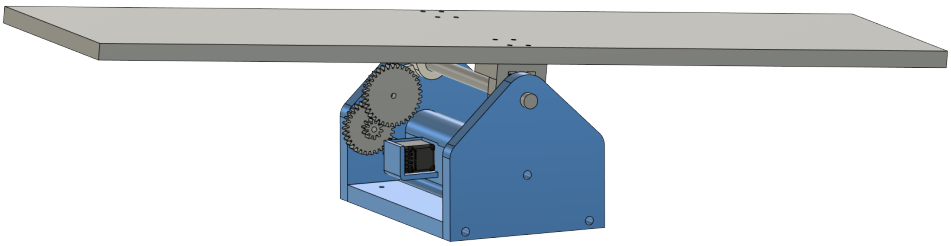

Figura 3.19: Modelo final con transmisión de engranajes.

Errores de la versión final

La impresión de los engranajes fue exitosa y su calidad era buena, se realizó el montaje y se engraso el sistema. Por desgracia, esta versión del balancín tampoco tuvo éxito, el sistema de engranaje amplificaba demasiado la fricción del encoder por lo que el balancín se movía con dificultad, el gran torque necesario para mover algunos de los engranajes, sumado a las pequeñas holguras del sistema provocaba que los engranajes se atascasen.

En el diseño se tuvieron en cuenta las holguras y tolerancias, pero esto no fue suficiente. Pese a que el sistema desconectado del *encoder* giraba bien al conectar todos los elementos se atascaba en algunos ángulos concretos.

Conclusiones del diseño

Los errores mencionados pueden solucionarse optimizando aún más el diseño y modificando varias de las piezas, pero el análisis y diseño de componentes mecánicos no es una tarea trivial y dado que ya se ha invertido demasiado tiempo en el diseño y no es la prioridad de este proyecto se ha decidido no continuar con el proceso. El balancín es funcional sin el sistema de transmisión, por lo que finalmente se usara sin el.

3.4. Resultados experimentales

En la sección 3.2.4 se han mostrado tanto el controlador PID implementado como el modelo teórico de la dinámica del sistema, sin embargo para poder controlar el sistema es necesario sintonizar el PID ya que sin una buena elección de los valores K_p, K_i y K_d

la dinámica del sistema será inestable.

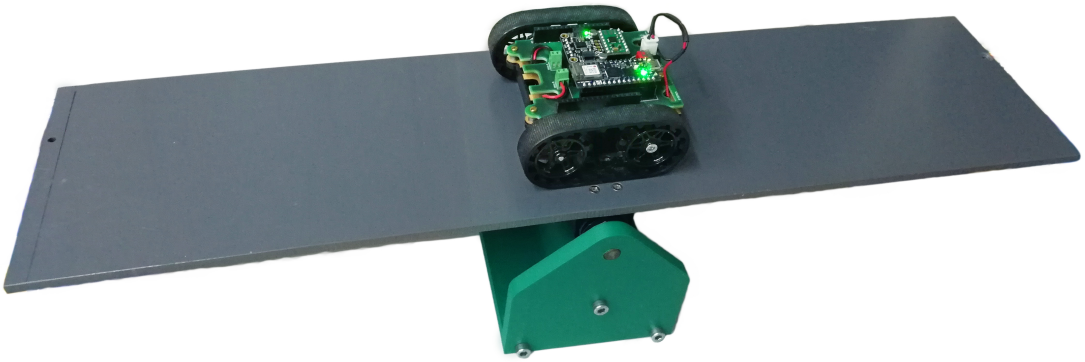

Figura 3.20: Sistema robot balancín.

Existen diversos métodos de sintonía de PIDs [15], como los de Ziegler-Nichols las reglas de sintonía MIGO o las de AMIGO pero para sintonizar el sistema se decidió utilizar el método de prueba y error, ya que los motores no están totalmente caracterizados y no se dispone de la función de transferencia del sistema completo.

Después de un largo proceso de sintonía no se obtuvieron resultados en cuanto a la estabilidad. El robot es incapaz de estabilizarse en el balancín. En alguna ocasión se consiguió estabilizar el sistema, pero se determinó que estos sucesos no eran fruto de un control exitoso sino de un conjunto de condiciones que se habían dado, como la posición inicial del robot.

Durante el experimento se han presentado diferentes dificultades a la hora de realizar el control:

- Retardo en la respuesta:

  El robot parece tener dificultades a la hora de seguir los movimientos bruscos del balancín, esto indica que el robot presenta un retardo a la hora de responder a estos movimientos, especialmente cuando se encuentra cerca del punto de equilibrio y debe realizar pequeñas correcciones rápidamente.

- No linealidades:

  Los motores presentan un comportamiento no lineal, esto se puede evitar linealizando su comportamiento y trabajando en la zona linealizada.

  Sin embargo durante la aplicación se ha comprobado que en este caso particular delimitar esa zona es una tarea compleja, ya que el control que se debe realizar requiere unos movimientos muy leves y finos de los motores, esto implica que los motores estén trabajando con tensiones bajas en el límite entre la zona muerta y la zona lineal.

Una caracterización de la zona muerta muy ajustada permite controlar el motor a una velocidad más lenta y realizar movimientos más leves cerca del punto de equilibrio, sin embargo en ocasiones alguno de los motores puede encontrarse en la zona muerta y no responder como debería y dado que los dos motores no funcionan en lazo cerrado y no son totalmente idénticos se aprecia un pequeño desvió en la dirección.

Por el contrario, delimitar esta zona muerta con demasiado margen provoca que para valores de tensiones muy bajos los motores giren a una velocidad demasiado alta.

- Periodo de muestreo:

  El periodo de muestreo del robot a la hora de medir el ángulo está limitado por la velocidad de procesamiento del Arduino, este ejecuta el algoritmo en un bucle por lo que el tiempo que tarda en realizar ese bucle es el periodo de muestreo.

  Se ha optimizado el código todo lo posible para que el periodo sea el mínimo, sin embargo alguna operaciones de procesado como el filtrado del término derivativo o el envió de datos mediante MQTT que se realizan dentro del bucle aumenta este periodo.

  En el caso del filtro, esto supone tener que buscar un compromiso entre el procesado de la señal y la velocidad de muestreo pero en cualquier caso limita el rendimiento del control.

- Aceleración brusca:

  En algunas ocasiones se ha intentado conseguir una respuesta más rápida por parte del robot aumentando el término proporcional o el derivativo, como consecuencia, su respuesta ante errores bruscos causaba una gran aceleración provocando que el robot se inclinase debido a esta. Además el hecho de que el robot se incline conlleva que comienza a acelerar aún más para corregir esa inclinación.

  Esta situación inutiliza por completo el control ya que el ángulo medido por el sensor del robot deja de ser el ángulo del balancín y el robot se desestabiliza por completo.

Las dificultades mencionadas añaden complejidad al control del sistema, sin embargo aun con estas dificultades el sistema podría ser controlado si la dinámica del balancín fuera más lenta, por ejemplo si las proporciones de masas del balancín y el robot fueran diferentes a las actuales (ecuación 3.17). En la configuración actual el balancín se mueve con demasiada facilidad ante cualquier perturbación leve.

$$\begin{cases} m_B = 667g \pm 1g \\ m_R = 251g \pm 1g \end{cases} \tag{3.17}$$

Durante el montaje del balancín se ha tenido especial cuidado en reducir la fricción del balancín con el eje, pero si esta fricción fuera mayor o la masa del balancín fuera mucho mayor que la del robot este tendría menos dificultades a la hora de mantener el balancín

estable en el punto de equilibrio.

Pese no haber sido capaces de controlar el sistema, como resultado del experimento se ha obtenido un robot programable con Circuitpython totalmente funcional, además gracias al *hat* que se implementó en la placa PCB las funciones y el hardware del robot son totalmente expandibles para futuras aplicaciones.

El sistema robot/balancín creado también puede ser utilizado en futuros experimentos o con intención de mejorar el control realizado, pues existen gran cantidad de técnicas que pueden ser utilizadas para mejorar el control pero que por motivos de tiempo no se han explorado a lo largo de este estudio.

El control mencionado en la sección 3.2.4 está implementado mediante código en el robot. Esto permite utilizar el controlador con diferentes parámetros y variables de referencia, por lo que se ha utilizado el control para realimentar el ángulo de orientación del eje z (ver Figura 2.1) y se ha obtenido un resultado positivo. El robot es capaz de mantener una trayectoria en cierta dirección y corregir la trayectoria ante cualquier tipo de perturbación, el control proporciona a cada motor la potencia necesaria para corregir la desviación.

Capítulo 4

Conclusiones

A lo largo del estudio se ha tratado un gran número de temas de forma teórica o incluso práctica, muchos de estos conceptos como la obtención de medidas de un sensor IMU o el control mediante PID de un sistema son conceptos teóricos sencillos, sin embargo aplicar estos conceptos en la práctica de forma funcional es una tarea más compleja. En general como se ha podido comprobar esto ocurre con la mayoría de conocimientos, que en un principio parecen sencillos, pero al profundizar más en ellos para entender su completo funcionamiento aparecen dificultades técnicas o conceptos más complejos. Durante el uso experimental del IMU, cuando se realizaron las primeras mediciones se encontró la necesidad de conocer las características de cada sensor dentro del módulo GY-87. Esto fue necesario para poder interpretar lo que se estaba midiendo inicialmente y conocer la escala de las medidas, para poder entender esto fue necesario entender las diferentes configuraciones que tienen los sensores, ya que estos pueden trabajar en diferentes rangos y con diferentes resoluciones. Los sensores vienen configurados de forma predeterminada pero esta puede no ser la más adecuada para todas las aplicaciones.

Tras realizar varias pruebas y haber encontrado varios inconvenientes con el módulo GY-87 se decidió cambiar este por un módulo que utilizaba el *IC* BNO055 [2]. Con este nuevo sensor fue necesario un nuevo estudio de sus características, sin embargo el estudio anterior fue de gran ayuda ya que el procedimiento a realizar es el mismo.

Una gran parte de este proyecto se ha basado en tomar decisiones y realizar diseños, en un principio al tener tanta libertad esto puede parecer una tarea trivial pues se puede elegir entre un gran número de opciones. Sin embargo esta tarea es de las más educativas y de las que más tiempo conllevan, ya que cuando se pretende cumplir un objetivo concreto o se quieren ciertas especificaciones, se deben comparar posibilidades, lo que conlleva a realizar un estudio de las características de cada elemento a elegir, con el fin de determinar qué opción es la más adecuada para cada caso concreto. Esto ha sido notorio durante el desarrollo de la aplicación del IMU donde se han tenido que realizar diversos diseños CAD hasta lograr la versión final del balancín y del robot. Finalmente, se han obtenido diseños satisfactorios del balancín y del robot, el balancín funciona como se esperaba y el robot es totalmente funcional, programable y expandible.

Durante la realización del diseño ha sido necesario tener que documentarse sobre aspectos o tecnologías que hasta el momento se desconocían o no se entendían por completo. Este ha sido el caso de la selección del hardware del robot y de la selección de protocolo de comunicación, inicialmente se utilizaron otros tipos de comunicación basados en Servidores Web y en peticiones *HTTP*, pero finalmente se decidió seguir con el protocolo *MQTT* debido a su mayor velocidad de transmisión.

La aplicación real se llevó a cabo sin lograr el equilibrio del sistema, pero el desarrollo de esta aplicación ha dado como resultado el estudio de una gran variedad de protocolos de comunicación, hardware, métodos de control, etc. Por lo que se puede concluir que tanto el estudio realizado como el desarrollo de esta aplicación han sido fructíferos. Para el diseño del control se observaron los efectos no lineales y no ideales de forma experimental y en base a las mediciones se modificó el controlador PID hasta obtener el control mencionado en la sección 3.2.4, tras la implementación del filtrado de la parte derivativa se observó una mejora en la señal de control. Esta tenía una forma menos ruidosa. Por otro lado al implementar el *antiwindup* se noto una mejora en la respuesta del control en las ocasiones en las que este se encuentra saturado. Esta última modificación actuaba rara vez, ya que durante las pruebas experimentales la mayoría del tiempo se trabaja con potencias pequeñas y los motores se encontraban lejos de saturarse.

Se ha comprobado que en un sistema real la velocidad de muestreo cobra un papel importante llegando a determinar el éxito o el fracaso del funcionamiento de un controlador. Es por eso que a la hora de montar un sistema es importante una selección correcta del hardware.

La aplicación del IMU ha sido realizada con éxito teniendo en cuenta que se han obtenido un gran número de conocimientos gracias a las observaciones realizadas. Por otra parte, se espera que la aplicación sea de utilidad para futuros experimentos. Pese a que el robot no es capaz de estabilizarse en el balancín el control PID funciona de forma correcta. Esto se ha podido comprobar en otras pruebas experimentales realizadas, donde se hizo al robot mantener una trayectoria recta. El PID controla la potencia de cada motor de forma correcta para que el robot mantenga la orientación inicial y el control responde bien ante perturbaciones relativamente altas.

Bibliografía

[1] I. Lizarraga, "Apuntes de la asignatura instrumentación 2." Facultad de Ciencia y Tecnología. UPV/EHU.

[2] *BNO055 Intelligent 9-axis absolute orientation sensor*, Bosch Sensortec GmbH, 2016, rev. 1.4.

[3] N. Maluf, *An Introduction to Microelectromechanical Systems Engineering*, 2nd ed. Artech House, June 2004.

[4] K.-L. Chau, S. Lewis, Y. Zhao, R. Howe, S. Bart, and R. Marcheselli, "An integrated force-balanced capacitive accelerometer for low-g applications," in *Proceedings of the International Solid-State Sensors and Actuators Conference - TRANSDUCERS '95*, vol. 1, 1995, pp. 593–596.

[5] S. Chang, M. Chia, P. Castillo-Borelley, W. Higdon, Q. Jiang, J. Johnson, L. Obedier, M. Putty, Q. Shi, D. Sparks, and S. Zarabadi, "An electroformed cmos integrated angular rate sensor," *Sensors and Actuators A: Physical*, vol. 66, no. 1, pp. 138–143, 1998.

[6] *3-Axis Digital Compass IC HMC5883L*, Honeywell International Inc., 2013, rev. E.

[7] A. G. Arribas, "Apuntes de la asignatura sensores y actuadores." Facultad de Ciencia y Tecnología. UPV/EHU.

[8] *MPU-6000 and MPU-6050 Product Specification*, InvenSense Inc., 2013, rev. 3.4.

[9] *BMP180 Digital pressure sensor*, Bosch Sensortec GmbH, 2013, rev. 2.5.

[10] *Arduino Nano RP2040 Connect*, Arduino S.r.l, 2020, rev. 0.1.

[11] *DRV8833 Dual H-Bridge Motor Driver*, Texas Instruments, 2011, rev. E.

[12] B. Hariprakash, A. Shukla, and S. Venugoplan, "Secondary batteries – nickel systems — nickel–metal hydride: Overview," in *Encyclopedia of Electrochemical Power Sources*, J. Garche, Ed. Amsterdam: Elsevier, 2009, pp. 494–501.

[13] E. Lemaire-Potteau, M. Perrin, and S. Genies, "Batteries — charging methods," in *Encyclopedia of Electrochemical Power Sources*, J. Garche, Ed. Amsterdam: Elsevier, 2009, pp. 413–423.

[14] K. J. Åström and T. Hägglund, *PID Controllers: theory, design, and tuning*, 2nd ed. ISA - The Instrumentation, Systems and Automation Society, 1995.

[15] I. Sagastabeitia, "Apuntes de la asignatura control automático 2." Facultad de Ciencia y Tecnología. UPV/EHU.

[16] G. F.Franklin, J. Powell, and A. Emani-Naeini, *Feedback Control of Dynamic Systems*, 8th ed.

[17] T. Kibble and F. H. Berkshire, *Classical Mechanics*, 5th ed. Singapore: World Scientific Publishing Company, Jun 03, 2004.

[18] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, "Control system design," Valparaíso, January 2000.

[19] J.-D. Warren, J. Adams, and H. Molle, "Arduino robotics," 2011.

[20] K. R. Britting, *Inertial navigation systems analysis.* Boston [u.a.]: Artech House, 2010.

[21] N. Yazdi, F. Ayazi, and K. Najafi, "Micromachined inertial sensors," *Proceedings of the IEEE*, vol. 86, no. 8, pp. 1640–1659, 1998.

[22] J. Söderkvist, "Micromachined gyroscopes," *Sensors and Actuators A: Physical*, vol. 43, no. 1, pp. 65–71, 1994.

[23] C. H. J. Fox, "The dynamics of a vibrating cylinder gyro with imperfection," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 210, no. 5, pp. 453–464, 1996.

[24] R. Langdon, "The vibrating cylinder gyroscope," *The Marconi Review*, 1982.