



UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA



Diseño, construcción y control de un cubo invertido

MEMORIA DE PROYECTO PRESENTADA A LA FACULTAD DE
INGENIERÍA DE LA UNIVERSIDAD DE LA REPÚBLICA POR

Francisco Valles, Francisco Pastorini

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS
PARA LA OBTENCIÓN DEL TÍTULO DE
INGENIERO ELECTRICISTA.

TUTOR

Nicolás Pérez Universidad de la República
Pablo Monzón Universidad de la República

TRIBUNAL

Álvaro Giusto IIE, Fing, Universidad de la República
Leonardo Barboni IIE, Fing, Universidad de la República
Juan Llaguno IFFI, Fing, Universidad de la República

Montevideo
jueves 14 septiembre, 2023

Diseño, construcción y control de un cubo invertido, Francisco Valles, Francisco Pastorini.

Esta tesis fue preparada en L^AT_EX usando la clase iietesis (v1.1).
Contiene un total de 156 páginas.
Compilada el jueves 14 septiembre, 2023.
<http://iie.fing.edu.uy/>

TIENES POCO TIEMPO PARA EXPLICARTE
TODOS LOS ENIGMAS QUE TE RODEAN.
L.A.S.

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Darle vida al Cubinja fue un proceso largo y de mucho trabajo en el cual no solo nosotros y los tutores fuimos parte, sino que también hubieron muchas otras personas que gracias a su apoyo pudimos sortear todas las dificultades que se nos presentaron.

Queremos primeramente agradecer a Álvaro, el tercer integrante con el que comenzamos este proyecto. Si bien su partida y abandono de la carrera nos dolió a todos, siempre se mostró dispuesto a trabajar en el proyecto, incluso después de abandonado el mismo.

Agradecemos también a nuestras familias y amigos, estas son las primeras personas en ver de cerca todo el trabajo que estamos realizando y también los primeros en apoyarnos y darnos una mano.

A nuestros tutores Pablo y Nico por guiarnos académicamente para llevar el proyecto adelante y por su disposición y entendimiento, incluso en los momentos más difíciles.

A Nicolás Finozzi por toda la ayuda y orientación que nos dio en el manejo de la impresora 3D y a Henry Figueredo por poner a disposición las impresoras del IMPI.

A Focus Ingeniería por su disposición a prestarnos componentes e instalaciones para llevar el proyecto adelante.

Finalmente, agradecer a la Universidad de la República, y especialmente, la Facultad de Ingeniería, por ser la institución que nos formó y convirtió nuestras carreras de un proyecto a una realidad.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Este trabajo presenta el desarrollo de un dispositivo que integra aspectos de la teoría del control a partir del diseño electro mecánico de un cubo robotizado. Desarrollado en el Instituto de Ingeniería Eléctrica de la Universidad de la República, este proyecto tiene como resultado final la entrega de dos dispositivos que pueden ser balanceados en una posición de equilibrio inestable.

El primer dispositivo es una versión simplificada del cubo final. Por medio de un volante de inercia acoplado a un motor, estando este acoplado a una plataforma rectangular, permite que el sistema sea balanceado en uno de sus vértices regulando el torque aplicado a un volante. Se realiza el modelado del sistema y a partir de esto se diseña e implementa una solución que pueda cumplir con el objetivo impuesto. El dispositivo sensa su posición por medio de una IMU (por sus siglas en inglés, Unidad de medición Inercial) que en comunicación con un microcontrolador aplica acciones correctivas al motor, generando el torque que permite mantener al sistema equilibrado.

El segundo dispositivo consiste en una versión en tres dimensiones del dispositivo mencionado anteriormente formando un cubo de 15x15x15cm. Tres caras de este cubo cuentan con volantes de inercia que le permiten balancearse en cada eje alterando el torque que se les aplica. Esto permite que en última instancia sea posible balancear el cubo en una de sus aristas y también en alguno de sus vértices, dejando este trabajo pendiente para futuros estudiantes que trabajen con el cubo. A su vez, el cubo es comandado internamente por un microcontrolador, es completamente autónomo, cuenta con una interfaz física para el usuario y una aplicación de control remota.

El proyecto es entregado con una serie de manuales de usuario y armado de ambos dispositivos para que el Departamento de Sistemas y Control del Instituto de Ingeniería Eléctrica lo utilice como herramienta de trabajo.

Esta página ha sido intencionalmente dejada en blanco.

Glosario

1D Refiere a una dimensión, en particular al prototipo del frame con tan solo un grado de libertad.

3D Refiere a tres dimensiones, en particular al prototipo del sistema con tres grados de libertad en formato cubo.

Active objects Es una tarea de firmware con una cola asociada para recibir datos.

Arduino Es una plataforma de fuente abierta para el diseño de prototipos electrónicos.

Back EMF Del inglés *back electromotive force* es una fuerza electromagnética que aparece en un circuito inductivo en una dirección tal que se opone a cualquier cambio de corriente en el circuito. En particular esta fuerza aparece en los motores BLDC con los que trabajamos.

BLDC Del inglés *Brushless Direct Current Motor*, motor de corriente continua sin escobillas. Se profundiza más sobre el tema en el capítulo 3.

BLE del inglés (*Bluetooth Low Energy*) Bluetooth de baja energía.

Bluetooth Protocolo de comunicaciones para redes inalámbricas de área personal.

BOOT Estado del microcontrolador en el cual espera un nuevo firmware a ser .

CURA Es una aplicación diseñada para impresoras 3D, en la que se pueden modificar los parámetros de impresión y después transformarlos a .gcode.

encoders Sistema de medición posicional del eje de un motor.

ENDER Impresora de Creality modelo ENDER, utilizada para la impresión de piezas 3D.

Flasher Es el acto de mover un programa a la memoria fija de un microcontrolador.

Glosario

Frame Se refiere en ocasiones al prototipo en 1D, que consiste en un marco cuadrado con un motor y volante de inercia, con un extremo fijo a su base por medio de un ruleman. Se detalla su modelado y funcionamiento en el capítulo 4.

Fusion 360 Software de diseño 3D con licencia de estudiante. Fue el programa utilizado a lo largo del proyecto para el diseño y visualización de piezas.

gcode Es un lenguaje de programación usado en control numérico para indicarle a una máquina qué hacer y como hacerlo, en particular es el lenguaje más común utilizado en impresión 3D para indicar los parámetros y piezas a imprimir a la impresora.

IIE Instituto Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República.

IMU del inglés (*Inertial Measurement unit*) Unidad de medición inercial.

IPC (del inglés *Inter-Process Communication*) Comunicación entre procesos.

Light blue Es una aplicación que permite al usuario conectarse a dispositivos bluetooth, en particular dispositivos que usan BLE.

LiPo Batería recargable de Litio y polímero de varias celdas.

LQR Del inglés *Linear-Quadratic Regulator*, es un tipo de controlador que busca minimizar una función de costo cuadrática que representa el desempeño del sistema. Utiliza una combinación lineal de la información del estado y del objetivo deseado para calcular la señal de control óptima.

PID Del inglés *Proportional, Integral, Derivative*, es un tipo de controlador que evalúa el error entre cierto ítem de control deseado (*Set point*) y el valor obtenido *Process variable* y aplica correcciones basadas en términos proporcionales, integrales y derivativos.

PLA Del inglés *Polyactic Acid*, es el material más común utilizado como filamento para las impresoras 3D de extrusión.

Power Budget Diagrama en el que se muestran los niveles de tensión y corriente del sistema.

PWM Del inglés *Pulse-Width Modulation*. Modifica el *duty cycle* de una señal periódica, en este caso para fijar el nivel de tensión de alimentación del BLDC.

Python Lenguaje de programación de alto nivel.

RTOS del inglés *Real Time Operating System*, Sistema operativo en tiempo real.

Rulemán Es un rodamiento radial NTN colocado en la base del prototipo 1D. Se le agregó una plataforma con orificios para acoplarlo a la base del Frame.

SPI del inglés *Serial Peripheral Interface*. Es un estándar de comunicaciones usado para transferencia de información entre circuitos integrados en equipos electrónicos.

Swing up Movimiento que permite al dispositivo pararse por sí mismo. Es tratado en profundidad en el capítulo 5.

USB Del inglés *Universal Serial Bus*, es un bus de comunicaciones estándar.

UTE La Administración Nacional de Usinas y Trasmisiones Eléctricas es la compañía estatal de producción y abastecimiento de energía eléctrica de Uruguay.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Agradecimientos	III
Resumen	V
Glosario	VII
1. Introducción	1
1.1. Motivación	1
1.2. Descripción del proyecto	2
1.3. Objetivos y especificaciones	4
1.3.1. Objetivos originales	4
1.3.2. Objetivos actualizados	4
1.4. Antecedentes y estado del arte	5
1.5. Organización del documento	6
2. Diagrama del sistema	9
2.1. Descripción general	9
2.2. Elección del microcontrolador	13
2.3. Requerimientos	14
3. Sensores y actuadores	17
3.1. Sensores	17
3.1.1. Descripción general	17
3.1.2. IMU	17
3.1.3. Selección de IMU	18
3.1.4. Estimador de posición angular	19
3.1.5. Estimador de velocidad angular	22
3.1.6. Encoder	24
3.1.7. Selección de encoder	24
3.1.8. Estimador de velocidad del volante	25
3.2. Actuadores	27
3.2.1. Descripción general	27
3.2.2. Motores BLDC	27
3.2.3. Selección de BLDC	29
3.2.4. Servo motores	31

Tabla de contenidos

4. Modelado del frame y controlador	33
4.1. Modelado del problema unidimensional	33
4.1.1. Hipótesis del modelado	33
4.1.2. Componentes y Nomenclatura	34
4.1.3. Construcción del modelo	35
4.1.4. Linealización	38
4.1.5. Ecuaciones en variables de estado	39
4.2. Identificación de parámetros	40
4.2.1. Estimación de parámetros del sistema volante + motor	40
4.2.2. Estimación de parámetros del frame	42
4.3. Controlador LQR	46
5. Swing up	49
5.1. Freno de bloqueo	51
5.2. Freno por rozamiento	52
5.3. Sistema de freno implementado - Zapata de frenado	53
6. Power Management Circuit	57
6.1. Descripción general	57
6.2. Power Budget	57
6.2.1. Alimentación de una sola celda	59
6.2.2. Alimentación de tres celdas	61
6.3. Elección de batería	62
6.4. Elección de reguladores	63
6.5. Construcción y conexión	64
7. Construcción	67
7.1. Descripción general	67
7.2. Volante de inercia	67
7.3. Unión de motor a frame	69
7.4. Frame	69
7.5. Soporte de componentes internos	70
7.6. Circuito	71
8. Diseño del software embebido	73
8.1. Descripción general	73
8.2. FreeRTOS	73
8.3. Arquitectura	75
8.3.1. Módulos diseñados	75
8.3.2. IPC: Alfred	76
8.3.3. Máquina de estados	77
8.3.4. Implementación del controlador	78
8.4. Comunicación inalámbrica	79
8.4.1. Sobre módulo BLE en el ESP32	79
8.4.2. Características	80
8.4.3. Protocolo de comunicación	80

8.4.4. Aplicación remota	81
8.5. Auxiliares	82
9. Ensayos	83
9.1. Ensayos Frame	83
9.1.1. Testeo de filtro complementario	83
9.1.2. Testeo de EMWA	84
9.1.3. Testeo encoder	85
9.1.4. Testeo Control	86
9.1.5. Testeo swing up	87
9.2. Ensayos Cubinja	90
9.2.1. Testeo encoders y motores	90
9.2.2. Detección de posición	91
9.2.3. Controlador	93
9.2.4. Monitoreo de batería	96
10. Conclusiones	97
10.1. Conclusiones finales	97
10.2. Trabajo futuro	99
A. Manual de armado	101
A.1. Herramientas	101
A.2. Precauciones y cuidados	101
A.3. Frame (1D)	103
A.3.1. Guía de partes	103
A.3.2. Armado	103
A.3.3. Cableado	104
A.4. Cubo (3D)	107
A.4.1. Guía de partes	107
A.4.2. Armado	108
A.4.3. Cableado	109
B. Guía de instalación	111
C. Manual de usuario	115
C.1. CubinjaFW y FrameFW	116
C.2. Aplicación remota	117
C.2.1. Debuggeo a través de celular y LightBlue	119
D. Protocolos de comunicación Bluetooth	121
E. Gestión de proyecto	123
E.1. Costos	123
E.2. Gestión de riesgos	123
E.3. Planificación y dedicación horaria	124

Tabla de contenidos

F. Hojas de datos	127
F.1. Microcontroladores	127
F.2. Sensores y Actuadores	127
F.3. Potencia	128
Referencias	129
Índice de tablas	132
Índice de figuras	134

Capítulo 1

Introducción

El concepto de péndulo invertido a sido utilizado a lo largo de los años como método de testeo y desarrollo de sistemas analizados desde la teoría del control. Algoritmos y metodologías de control han sido desarrollados con el fin de lograr comportamientos estables y controlados en sistemas naturalmente desbocados e inestables. Actualmente se continúan desarrollando ideas y conceptos en el entorno de este tipo de sistemas, siendo objeto de investigación activa globalmente [1].

Cubinja se desarrolla en este entorno antes mencionado, basándose en el concepto de poder controlar la estabilidad de un cubo por medio de sensores y actuadores. Este proyecto surge a partir de un proyecto desarrollado por el Instituto para Sistemas Dinámicos y Control de la universidad de Zurich en 2011 llamado *The Cubli: A Cube that can Jump Up and Balance* [2]. Este consiste en un cubo mecanizado de 15x15x15cm capaz de saltar y balancearse sobre uno de sus vértices. Esto es logrado por medio del manejo de tres volantes de inercia colocados dentro del cubo, accionados por medio de un sistema de control que se encuentra en constante escaneo de su estado de equilibrio por medio de una unidad de medición inercial (IMU). Además, el sistema es capaz de detener instantáneamente sus volantes de inercia, permitiendo el desplazamiento y saltos controlados del cubo. El concepto principal del proyecto Cubinja reside en investigar los antecedentes mencionados, además de las ramificaciones que surgen a partir de la idea original del cubo, pudiendo en última instancia reproducir el mismo desde el enfoque de los estudiantes.

1.1. Motivación

Anteriormente se mencionó muy brevemente la aplicabilidad de trabajar con sistemas de péndulo invertido. Estos se encuentran en distintas y variadas ramas de la industria, como pueden ser la robótica, aeronáutica, e incluso hoy en día hasta en artículos cotidianos. Estudiar la viabilidad de transformar estos sistemas de naturaleza inestable a una estado de estabilidad controlado implica grandes beneficios para el desarrollo de proyectos de ingeniería aplicada [3].

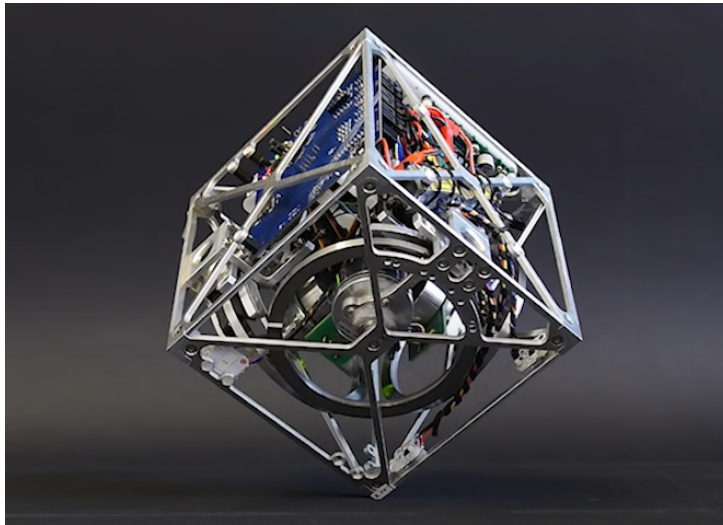


Figura 1.1: Prototipo del Cubli, extraída de video presentación de Youtube del proyecto The Cubli [2].

La elección de desarrollar el diseño y ensamblado de este prototipo reside en la oportunidad que nos brinda, como aspirantes a ingenieros, poder trabajar de cerca estos aspectos antes mencionados. El desafío que presenta involucrar diversas ramas de la ingeniería como lo son la robótica, la mecánica, los sistemas embebidos, el control y el diseño electro mecánico en general, resulta llamativo en esta etapa de la carrera.

A lo largo del proyecto los objetivos del mismo fueron evolucionando, definiendo como objetivo final entregar al Instituto de Ingeniería Eléctrica (IIE), en particular al Departamento de Sistemas y Control, un prototipo completamente definido y resuelto en todos sus aspectos, funcional, y capaz de ser utilizado por otros estudiantes con el objetivo de desarrollar sus conocimientos en el área de control.

A partir de este punto es que surge la principal motivación de desarrollar este proyecto para el grupo, poder dotar de un instrumento nuevo de trabajo al instituto. Finalmente, enfrentarse al proceso de ingeniería que implican las distintas etapas de desarrollo del mismo nos permite visualizar las dificultades y premios que uno encuentra al desempeñarse como ingeniero, que vale aclarar es nuestro objetivo final al realizar el proyecto de grado.

1.2. Descripción del proyecto

El proyecto consiste en diseñar y construir un prototipo que implemente un sistema con dinámica compleja y dotarlo con las aptitudes para que se comporte como lo desee el usuario. Se comenzará estudiando el sistema de forma unidimen-

1.2. Descripción del proyecto

sional, se construirá este primer prototipo para luego trasladar las experiencias al cubo.

Las principales tareas que se desarrollaran a lo largo del proyecto consisten en primer lugar en realizar una revisión bibliográfica de antecedentes y estado del arte. Luego se trabajará con el modelado del sistema con el fin de bosquejar una implementación del mismo. Una vez hecho esto se procede al diseño de una implementación en hardware del cubo. Se realizan las pruebas y ensayos pertinentes para verificar el correcto funcionamiento de todos los componentes del sistema, que en última instancia permitirá al usuario aplicar las leyes de control que desee con el objetivo de equilibrar el cubo. Una vez hecho esto se trabajará en la interfaz con el usuario, permitiendo la configuración y obtención de datos del sistema en forma remota. El sistema va a permitir que el usuario pueda aplicar distintas técnicas de control, modificando los distintos módulos del software de control siguiendo las indicaciones del manual de uso del cubo. A su vez el usuario podrá recolectar datos en forma de gráfico y registro de comunicación vía bluetooth e interactuar con el sistema por medio de una aplicación desarrollada en python. Por medio de esta comunicación podrá testear los distintos componentes y servicios que ofrece el cubo, así como también alterar y modificar distintos parámetros de trabajo del mismo.

A lo largo del documento se hace un seguimiento del desarrollo del proyecto, visualizando las distintas etapas del mismo. Se presenta un análisis teórico de la situación en 2 dimensiones, que luego se traslada a 3 al conformar el cubo. Se detallan la elección de componentes, tanto sensores como actuadores, diseño de software embebido y los ensayos pertinentes para corroborar los modelos dinámicos desarrollados. Además, se hace un seguimiento por las etapas de diseño e impresión de partes 3D. Es a partir del análisis teórico de la situación en 2 dimensiones que se construye un modelo en variables de estado que permite analizar la dinámica no lineal del sistema, así como sus limitaciones y parámetros específicos.

Una vez establecidas las variables del sistema en cuestión se procede a aplicar diversas estrategias de control por medio de realimentación de estados, en particular de tipo PID (*Proportional, Integral, Derivative*) y LQR (*Linear-Quadratic Regulator*), con el fin de lograr el equilibrio del sistema. Esto se hace con el objetivo de verificar el correcto comportamiento del sistema. A su vez y en paralelo con este desarrollo, se implementa el sistema de frenado que dote al prototipo con la habilidad de levantarse por si mismo.

Se entrega como producto final dos prototipos totalmente funcionales, con sus respectivos manuales de uso para que el usuario final puede utilizarlo como herramienta para sus estudios.

Se aclara al lector que los objetivos y alcances del proyecto fueron modificados a lo largo de su desarrollo, la principal razón para ello es el desistimiento de uno

Capítulo 1. Introducción

de los tres estudiantes que conformaban el equipo de estudio originalmente. Es por esto que se re estructuraron varios aspectos del proyecto en conjunto con los tutores. Lo presentado es el resultado final de este proceso. A continuación se da detalle del alcance de estas modificaciones en los objetivos del proyecto.

1.3. Objetivos y especificaciones

1.3.1. Objetivos originales

El objetivo general original del proyecto consistía en desarrollar un sistema de control sobre un cubo y así lograr que se mantenga en equilibrio parado sobre una arista o vértice. Además el cubo debería llegar a esta posición de equilibrio por medio de un movimiento de Swing up. Se deberá construir el prototipo con la caracterización de todos sus parámetros y la implementación de la comunicación entre programas de simulación, sensado y programación.

En vista del desistimiento del tercer compañero del grupo de proyecto, los objetivos y alcances del mismo fueron re estructurados y se detallan en la siguiente sección.

1.3.2. Objetivos actualizados

El objetivo principal del proyecto consiste en desarrollar la implementación de un cubo que pueda equilibrarse por sí mismo. Dentro de los objetivos particulares se encuentran el modelado unidimensional del sistema, para luego implementarlo y trasladar la implementación al problema tridimensional. El producto final debe ser autónomo, sin conexiones cableadas con su entorno. Además el mismo debe ser capaz de comunicarse en forma remota con el usuario, y contar con una interfaz que facilite esta comunicación.

Se deberá construir en primer lugar una versión simplificada del cubo en una dimensión (1D) con solo un grado de libertad, que consiste en una cara del mismo con un volante de inercia que le permita mantenerse equilibrado en uno de sus vértices. Esta versión se encontrará fija a un soporte por medio de un ruleman con el objetivo de minimizar el rozamiento del mismo y a su vez asegurar y fijar su movimiento al plano en 2 dimensiones. A esta versión se le conoce como Frame a lo largo del documento. Este sistema deberá contar con un sistema de medición inercial que le permita conocer su posición en todo momento, medición a partir de la cual por medio de un lazo de control el prototipo buscará mantenerse en equilibrio. Además, se debe implementar un sistema de frenado que le permita al prototipo partir del reposo por si mismo, deteniendo su volante de inercia en forma cuasi instantánea.

1.4. Antecedentes y estado del arte

Una vez establecidos los parámetros y estudios del sistema dinámico en 1D se procederá al armado del cubo tridimensional (3D), que consiste en tres caras donde se colocan motores de forma similar al frame y tres caras simples que sirven como apoyo para el resto de componentes del sistema, estas conforman un cubo de 15x15x15cm. El mismo debe ser autónomo en cuanto a su fuente de alimentación, y autocontenido reteniendo todas sus piezas en su interior. A su vez, su firmware permitirá la comunicación de entrada y salida con el entorno del usuario por medio de conexión USB (Universal Serial Bus) así como vía *Bluetooth* por medio de una aplicación desarrollada en *Python*.

Ambos prototipos finales deberán cumplir el objetivo principal del proyecto que es dotar al departamento de control del IIE con una herramienta versátil y fiable con la cual estudiantes futuros puedan trabajar. Su estructura modular de software permitirá total soltura al usuario a la hora de aplicar distintos métodos de control al prototipo. Por último el producto final será entregado con sus debidos manuales de usuario, ensamblado mecánico, diagramas eléctricos y especificaciones de software.

1.4. Antecedentes y estado del arte

Cubinja toma aspectos de diseño y conceptuales que en gran parte ya han sido trabajados en otros proyectos, todos ellos abarcando una gran variedad de enfoques. Existe una vasta cantidad de artículos y documentos dedicados al estudio del comportamiento de sistemas de péndulo invertido y en particular a cómo dominarlos. En este caso estamos trabajando en una modalidad del mismo problema vista desde un enfoque novedoso para lo que es el IIE, ya que no se tiene antecedentes de trabajos con el objetivo de dominar el equilibrio de un cubo pero sí en por ejemplo péndulos de furuta [4].

Existen numerosos trabajos, teóricos y prácticos, relacionados con esta área de investigación. Algunos de los primeros trabajos teóricos incluyen la tesis de grado de Roberge (1960) en MIT [5] en la que se plantea la primer solución para un sistema de péndulo invertido, pasando por sistemas con agregados de complejidad mayor tratados por Hidgon y Cannon (1963) [6] e incluso analizados en publicaciones de Ogata (1970) [7]. En cuanto al enfoque práctico de la temática existe también una vasta cantidad de documentos pertenecientes a la NASA detallando la investigación de sistemas estabilizadores de transbordadores espaciales (Charles Stark Draper, 1972 [3]), basados en análisis de problemas de péndulo invertido. [8]

Ahora bien, trasladando el problema a un prototipo de cubo tridimensional, el primer antecedente a partir del cual basamos nuestra investigación surge del proyecto *The Cubli: A Cube that can Jump Up and Balance* [2]. Este proyecto hizo de punto de partida inicial para otros variados proyectos de características similares, muchos de ellos analizados como parte de esta investigación, e incluso se tomaron

Capítulo 1. Introducción

ideas para el desarrollo del Cubinja. Algunos de estos proyectos a destacar son

- *Development of a Nonlinear Mechatronic Cube* [9], proyecto de master de la Universidad de Chalmers, Suecia.
- *Cubli - Self Balancing Cube* [10], proyecto de grado de la universidad de Aalborg, Dinamarca.
- *ReM-RC - Self Balancing Cube* [11], proyecto basado en The Cubli, Lituania.

El Cubli y la mayoría de estos proyectos giran entorno a dos problemas de control. El primero de ellos consiste en lograr mantener el cubo equilibrado en uno de sus vértices y el segundo en que el cubo se lleve a esa posición por si mismo y sin intervención de agentes externos

Para el primer problema que consiste en mantener el cubo en equilibrio en una de sus aristas o vértices existen varios enfoques para su solución. La gran mayoría de ellos consisten en algún sistema de medición inercial que por medio de giroscopios y acelerómetros entrega información en tiempo real del estado posicional (ϕ, θ, β) del sistema, accionando volantes de inercia acoplados a las caras del cubo controlados por medio de un sistema digital. Más allá de las variaciones constructivas que se plantean para el armado del cubo, relaciones peso-torque, dimensiones, interconexión eléctrica y demás, y de el análisis de la dinámica del sistema, la principal diferencia entre proyectos reside en la elección de la ley de control aplicada al cubo. Los más elegidos son sistemas de control basados en controladores de realimentación de estado, aplicando técnicas de tipo LQR y PID (por ejemplo utilizados en el proyecto The Cubli [2]) para obtener parámetros de corrección de errores.

Para el segundo problema que consiste en dotar al cubo de la posibilidad de “saltar”, las soluciones planteadas giran entorno a la detención brusca de uno o varios de los volantes de inercia del cubo. Esto ocasiona una liberación de energía muy rápida por parte del volante de inercia, de donde los métodos mas utilizados consisten en aplicar conservación de momento angular durante el impacto y conservación de la energía del sistema. A partir de estos cálculos se pueden definir los parámetros involucrados en esta transición del reposo a la posición de equilibrio.

1.5. Organización del documento

Se procede a describir los contenidos de los distintos capítulos que componen el documento a continuación.

- **Capítulo 1: Introducción.** Se introduce al lector a los temas que concierne al documento para que tenga un primer acercamiento al tema. Luego

1.5. Organización del documento

se comentan distintas temáticas y situaciones que motivaron a los estudiantes del grupo a trabajar en este proyecto. En las siguientes secciones del capítulo se exponen los primeros detalles específicos del proyecto, objetivos a cumplir, los métodos y soluciones a aplicar. Por último se hace una recopilación de antecedentes y estado del arte relacionados al estudio de sistemas de péndulo invertido, sus derivados y también comentarios de la actualidad relacionados al tema.

■ **Capítulo 2: Diagrama del sistema.**

En este capítulo se presentan los diagramas de funcionamiento interno e interfaz con el usuario de los sistemas 1D y 3D. También se enumeran los requerimientos que debe cumplir el sistema.

■ **Capítulo 3: Sensores y Actuadores.** El capítulo se divide en dos partes. Primero se centra en el trabajo relacionado a los sensores que conforman la entrada de datos del sistema, en particular la IMU y los *encoders*. Se profundiza en los métodos de obtención de parámetros del cubo aplicados para estimar la posición y velocidad angular del frame. Se presentan los métodos utilizados para estimar velocidad y aceleración del volante de inercia a partir de los *encoders* del motor. En la segunda sección del capítulo se analizan los actuadores del sistema. En primer lugar se hace un breve repaso de los principios del motor de corriente continua y se presentan características particulares de los motores BLDC presentando finalmente el proceso de elección del motor utilizado. También se presentan las características del servo motor encargado del frenado del volante de inercia.

■ **Capítulo 4: Modelado del frame y controlador.** Se definen los componentes y nomenclatura del problema unidimensional. Se construye el modelo del sistema, se obtienen sus variables de estado y se linealiza el comportamiento del mismo, teniendo la precaución de enumerar las limitaciones del modelo. Se presentan también los distintos parámetros resultantes a partir de mediciones, ensayos y cálculos que son luego utilizados a lo largo del documento. Además, se detallan los distintos acercamientos al desarrollo de un controlador, en particular aplicando el método de LQR.

■ **Capítulo 5: Swing Up.** Se detallan los distintos elementos que componen el sistema de frenado. Se analizan diferentes propuestas vistas en proyectos anteriores, se estudia el respaldo teórico de estas propuestas y por último se visualizan ventajas y desventajas de cada método de frenado. Se detalla el sistema implementado, sus características, variables que forman parte, ensayos y resultados del mismo.

■ **Capítulo 6: Power Management Circuit.** En este capítulo se detalla el proceso de diseño, testeado e implementación del sistema encargado de energizar el cubo, dotándolo de autonomía por medio de una batería LiPo recargable. En primer lugar se realiza el Power Budget del sistema para identificar sus consumos y distintos niveles de tensión, luego se detalla el proceso de

Capítulo 1. Introducción

elección de componentes, diseño del circuito principal, implementación y puesta a punto. Además, se describen los componentes que conforman el medidor de batería baja y como este notifica al usuario.

- **Capítulo 7: Construcción.** Se visualizan los aspectos constructivos de ambos prototipos, los problemas que se afrontan, el diseño de piezas, la impresión de estas piezas así como el ensamblado y puesta a punto finales. Además, en el anexo de este documento se podrá encontrar un manual detallado de ensamblado para el usuario.
- **Capítulo 8: Diseño del software embebido.** En este capítulo se profundiza en el sistema operativo del microprocesador (FreeRTOS) así como en la arquitectura de software, los módulos implementados y el diseño de un intercomunicador de procesos (IPC). También dentro de la arquitectura, se documenta el funcionamiento de la máquina de estados del sistema y por último se dan especificaciones del módulo controlador. Por otro lado, la siguiente sección detalla los protocolos de comunicación inalámbrica (BLE), la aplicación de interfaz con el usuario desarrollada en Python y los elementos auxiliares que terminan de conformar el software del cubo. Además, se recuerda que en el anexo se puede encontrar un manual del usuario donde se detallan los comandos y métodos de comunicación con el sistema operativo.
- **Capítulo 9: Ensayos.** En este capítulo se registran los ensayos realizados tanto para la versión unidimensional del problema (*frame*) como para el cubo. Se detallan ensayos de sensores, actuadores, cálculo de posición y velocidad, controlador LQR, sistema de *swing up*; además de procesar y concluir resultados de los mismos.
- **Capítulo 10: Conclusiones.** Para finalizar, en este capítulo se presentan las conclusiones, principales problemas y reflexiones relacionados al desarrollo del proyecto, y posibles mejoras y trabajo a futuro.

Capítulo 2

Diagrama del sistema

2.1. Descripción general

Como fue mencionado anteriormente en el capítulo 1, el objetivo final del proyecto consiste en entregar dos prototipos del sistema estudiado al Departamento de Sistemas y Control del IIE. En este capítulo se detalla el funcionamiento interno e interfaces con el entorno de ambos dispositivos.

En la figura 2.2 se ve un diagrama de la interfaz con el usuario y el entorno del prototipo en 1D. También se puede ver un detalle de los componentes activos del sistema, detallado más adelante. Con el objetivo de familiarizar al lector con el sistema, se aclara que este dispositivo consiste en un marco cuadrado de plástico de 15x15cm denominado Frame que cuenta con un motor BLDC acoplado y un volante de inercia. Este marco se encuentra fijado a un soporte por medio de un rulemán. Este Rulemán permite el libre movimiento en 1D del dispositivo con mínimo rozamiento en el agarre. A partir de este momento en ocasiones se denomina a este dispositivo en 1D directamente como frame.

El frame cuenta con un sensor IMU que sensa el estado posicional y velocidades del mismo, y se comunica con el microcontrolador.

Los datos son procesados y por medio de una estrategia de control se aplican acciones correctivas indicando al motor BLDC la dirección, velocidad y aceleración por medio de una PWM con la que debe responder al movimiento del frame. Este torque aplicado al volante de inercia busca estabilizar al frame en su posición de equilibrio.

Además, el dispositivo cuenta con un sistema de frenado que le permite levantarse por sí mismo desde la posición de reposo a la posición de equilibrio.

Se enumeran los componentes internos del sistema vistos en la figura 2.2:

- **Microcontrolador:** Componente principal del sistema que se encarga de comandar al mismo y establecer la comunicación remota via *bluetooth*. Se comunica por SPI con la IMU. Implementa la acción de control sobre el

Capítulo 2. Diagrama del sistema

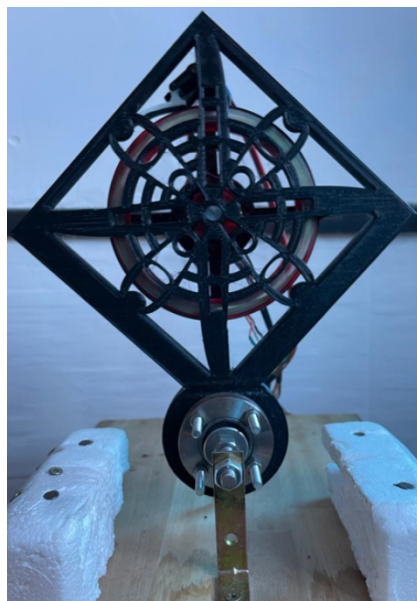


Figura 2.1: Sistema 1D.

BLDC enviándole una señal PWM y recibe información de los encoders del motor. También comanda el servo motor de frenado y un LED indicador de conexión por *Bluetooth*.

- **IMU:** Unidad de medición inercial que por medio de un acelerómetro y un giroscopio conoce el estado del frame y se lo notifica al microprocesador.
- **BLDC:** Motor acoplado al frame y al volante de inercia.
- **LED:** Led indicador de estados de conexión de bluetooth.
- **SERVO:** Servo motor encargado del frenado para hacer el Swing up.

El sistema a su vez se comunica con el usuario de distintas maneras. Permite que el mismo se conecte por USB o vía bluetooth para hacer uso de la *app* de comando del sistema. Esta permite obtener y enviar datos. El usuario también puede conectarse con su celular con el sistema por medio de la aplicación *Light-Blue* (C.2.1) para testear los componentes y funciones del sistema.

El sistema es alimentado vía USB el cual entrega 5V para energizar al micro-controlador y a la IMU. Se tienen también dos niveles de tensión de alimentación externa al sistema, en 5V para el servo motor y en 12V para el motor BLDC. Esta alimentación se entrega por medio de dos transformadores independientes, representados en la figura 2.2 por un conector de pared (220V UTE) con un transformador de baja y un rectificador de corriente.

En la figura 2.4 se ve un diagrama de la interfaz con el usuario y el entorno del prototipo en 3D. También se puede ver un diagrama de los componentes activos

2.1. Descripción general

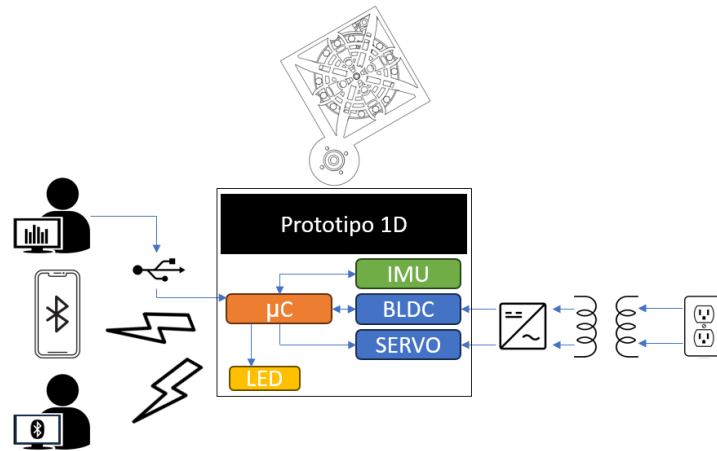


Figura 2.2: Diagrama de interfaces del sistema 1D.

del sistema, detallados más adelante en la figura 2.5. El dispositivo cubo en 3D conceptualmente funciona con los mismos principios que el sistema en 1D, pero triplicando los componentes y agregado la funcionalidad de monitoreo de nivel de batería. Este sistema cuenta con una interfaz de usuario que se puede ver en la figura 2.4. La misma cuenta con los siguientes componentes:



Figura 2.3: Sistema 3D.

- **Botón de ON/OFF:** Permite conectar y desconectar el sistema de la batería LiPo interna de alimentación.
- **Botón de BOOT:** Permite Flashear el microcontrolador.
- **Leds ON:** Indican que el sistema está encendido.

Capítulo 2. Diagrama del sistema

- **LED BT:** Led indicador de estados de conexión de bluetooth. Este mismo LED hace la función de indicador de batería baja con un parpadeo especificado en el manual del usuario.
- **Conector +12V:** Permite cargar la batería LiPo.
- **Conector USB-A:** Permite conectarse al sistema por USB.

El cubo es completamente autónomo, logrando todas sus funcionalidades independiente de cualquier fuente de energía externa. En la figura 2.4 se representa su conexión para cargar la batería sobre la derecha del diagrama. Los métodos de comunicación con el usuario son los mismos que para el prototipo anterior en 1D.

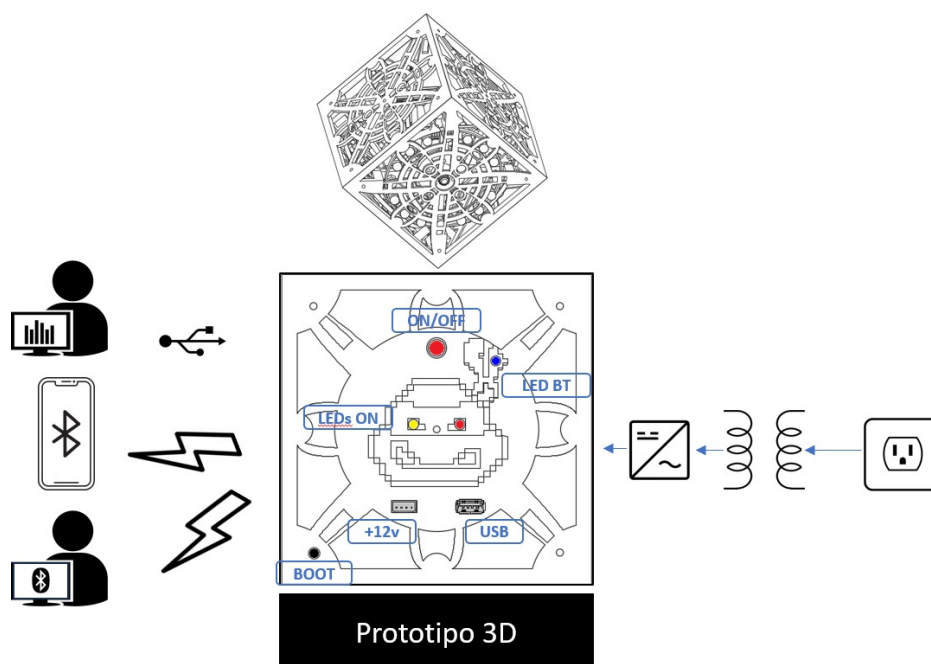


Figura 2.4: Diagrama de interfaces del sistema 3D.

En la figura 2.5 se puede ver un diagrama de los componentes internos al sistema, sus niveles de tensión y la comunicación entre sí. Se visualizan los botones de BOOT y de encendido del sistema. El circuito del divisor resistivo que lleva el voltaje de la batería a un rango aceptable para el ADC del microcontrolador para el monitoreo del nivel de carga de la misma. El circuito de gestión de potencia o PMC (del inglés *Power Management Circuit*) que a partir de 11,1V entrega 5V para la alimentación de los servo motores y 3,3V para la alimentación del microcontrolador e IMU. Los motores BLDC se alimentan directamente de la batería en 11,1V.

2.2. Elección del microcontrolador

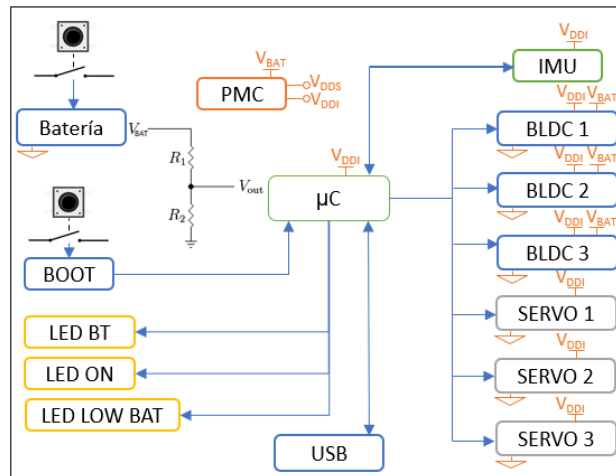


Figura 2.5: Diagrama de bloques del sistema 3D.

2.2. Elección del microcontrolador

Descripto el sistema se plantean los siguientes requerimientos para el microcontrolador:

- El microcontrolador deberá poder ser alimentado y programado vía USB.
- El microcontrolador deberá ser capaz de establecer una comunicación *bluetooth* sin necesidad de un periférico externo.
- El microcontrolador deberá tener un periférico ADC.
- El microcontrolador deberá contar con al menos un módulo SPI para comunicación con periféricos externos.
- El microcontrolador deberá contar con al menos 6 canales de PWM independientes.
- El microcontrolador deberá contar con periféricos GPIO.

Teniendo estos requisitos se paso a comparar tres alternativas distintas para el microcontrolador del sistema basado además también en los utilizados en proyectos previos como el Cubli. Se realizó la tabla comparativa que se muestra en la figura 2.6.

La primera opción analizada fue el microcontrolador utilizado por el propio *Cubli* [2], el STM32F4 [12]. Si bien este es el microcontrolador con mejores antecedentes por ser utilizado por el *Cubli* y por otros proyectos basados en este, fue descartado rápidamente debido a que el mismo no tiene un periférico *bluetooth* integrado.

Las siguientes dos opciones evaluadas fueron el *NRF52832* de Nordic Semiconductors [13] y el *ESP32 WROOVER E* de Espressif Systems [14]. Cualquiera de

Capítulo 2. Diagrama del sistema

Característica / Modelo	STM32F4	ESP32 WROVER E	NRF52832
Procesador	ARM7 Cortex-M4	ESP32-D0WD-V3	Cortex-M4
Velocidad	180MHz	Dual Core de 80 a 240 MHz	64MHz
Flash	512Kb	4Mb	512Kb
RAM	256Kb	8Mb	64k
BlueTooth	No	Si	Si
V alimentacion	3V3, usb	3v3, usb	3v3, usb
IDE	STM32CubeIDE	VScode - ESP IDF	Segger
Interfaces SPI	3	4	1
Canales PWM	17	16	8

Figura 2.6: Tabla comparativa de los microcontroladores de *STM*, *Espressif* y *Nordic Semiconductors*. Hojas de datos en apendice F.

las dos opciones cumple todos los requerimientos impuestos para el microcontrolador sin embargo se optó por el ESP32 por presentar mejores características de velocidad y memoria.

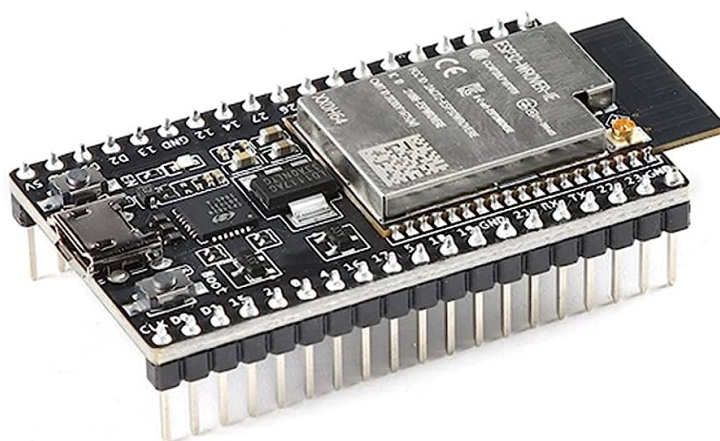


Figura 2.7: Foto del Development Kit del ESP32

El ESP32 es además un microcontrolador muy utilizado para proyectos del tipo DIY (del inglés *Do It Yourself*). Esto presenta la ventaja de tener acceso a gran cantidad de documentación, comunidad de respaldo, librerías y ejemplos que apoyan el diseño del firmware.

2.3. Requerimientos

En el desarrollo del proyecto se pretendió que el sistema Cubinja cumpla los requerimientos que se describen a continuación. Aquí se resumen estos conceptos.

Control

Ambos dispositivos deberán, en última instancia, poder ser controlados en su posición de equilibrio aplicando la ley de control que imponga el usuario.

2.3. Requerimientos

Interfaz con el usuario

El sistema deberá comunicar al usuario el estado de conexión remoto y cuando la batería deba cargarse. También deberá permitir la fácil conexión con el entorno.

Comunicaciones

El sistema embebido diseñado deberá permitir la comunicación, remota y por USB, con el procesador para que el usuario interactúe con el sistema. A su vez este deberá poder modificar los distintos parámetros del sistema, así como hacer uso de los módulos de control o de agregar o modificarlos en el caso que así lo desee.

Autonomía

El cubo deberá ser completamente autónomo, manejará todos sus niveles de tensión y consumo internamente alimentados por medio de una batería recargable. Además, debe tener una autonomía de carga de al menos treinta minutos.

Construcción

La construcción de los prototipos deberá ser robusta para garantizar la durabilidad de las partes. El cubo deberá ser montable y desmontable sin mayores inconvenientes por el usuario. A su vez, se deberá poder re hacer piezas particulares en caso de roturas. Los parámetros que alteren el rendimiento del sistema deberán ser en parte modificables por el usuario, por ejemplo el peso del volante o la fricción de las partes del sistema de frenado.

Producto final

Se deberán entregar ambos prototipos, (1D y 3D), siendo estos totalmente funcionales, habiendo comprobado el total cumplimiento de los puntos antes mencionados. Será posible testear sus componentes por separado, establecer comunicaciones de entrada y salida de datos y probar distintas estrategias de control. Se entregarán manuales de ensamblado e instalación y uso de software para el usuario.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Sensores y actuadores

3.1. Sensores

3.1.1. Descripción general

Para controlar el sistema y mantenerlo dentro de sus puntos de equilibrio es necesario estimar tres cantidades diferentes que forman el vector de estado del sistema. La posición y velocidad angular del Cubinja junto con la velocidad de los volantes de inercia. Este capítulo se enfoca en la descripción de los sensores que se utilizan con el fin de obtener un estimador del vector de estado y además cómo se manejan los datos de los mismos. Se discutirán técnicas importantes como el filtro complementario y el método de dos acelerómetros, así como el manejo de un encoder en cuadratura para el cálculo de la velocidad angular de los volantes de inercia del sistema.

3.1.2. IMU

Una Unidad de Medición Inercial (IMU, por sus siglas en inglés) es un dispositivo que combina múltiples sensores para medir y estimar la orientación, velocidad angular y aceleración de un objeto en el espacio tridimensional. Las IMUs se utilizan comúnmente en aplicaciones de navegación, control de vuelo y robótica, así como en la realidad virtual y aumentada. Gracias a su capacidad para proporcionar mediciones precisas de la orientación y la velocidad angular en tiempo real, las IMUs son una herramienta valiosa para muchos sistemas embebidos que requieren una alta precisión y fiabilidad. Usualmente una IMU cuenta con 3 tipos de sensores:

- **Giroscopio:** Es un sensor que mide la velocidad angular o la tasa de rotación de un objeto en el espacio tridimensional.
- **Acelerómetro:** Es un sensor que mide la aceleración lineal o la fuerza gravitacional en un objeto en el espacio tridimensional.

Capítulo 3. Sensores y actuadores

- **Magnetómetro:** Es un sensor que mide la fuerza y la dirección del campo magnético en un objeto en el espacio tridimensional.

El magnetómetro es un sensor opcional en una IMU. Las IMU que cuentan solo con giroscopio y acelerómetro son llamadas IMU 6DOF (seis grados de libertad) mientras que si cuentan con un magnetómetro se les llaman 9DOF.

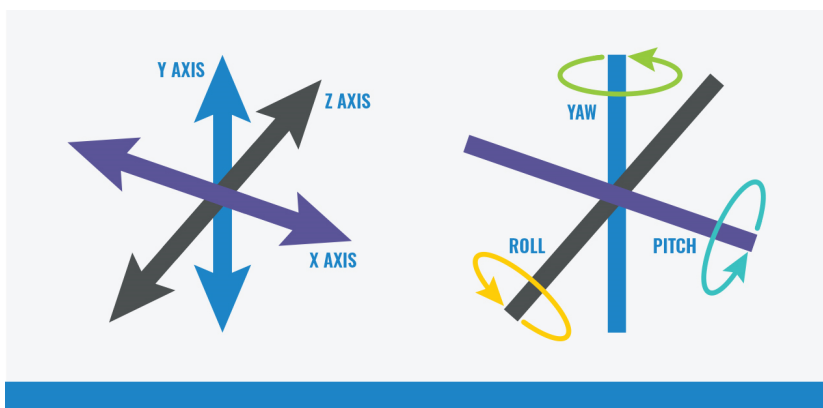


Figura 3.1: Diagrama de los ejes de medición para un acelerómetro y un giroscopio. Imagen tomada de [15].

3.1.3. Selección de IMU

La selección de la IMU se hizo en base a 3 puntos:

- Complejidad en el control del sensor
- Disponibilidad en plaza
- Facilidad en el uso de varias IMUs en simultáneo

El último punto se toma en cuenta debido a que, como se explicara en la siguiente sección, muchas técnicas de estimación de posición angular consisten en mezclar los datos de varias IMUs. Por lo tanto, poder tener varias IMUs trabajando en simultáneo es una prioridad. En base a estos puntos se decidió optar por la MPU9250 de InvenSense cuyas características principales son:

- **9DOF:** Si bien no se hará uso del magnetómetro para este proyecto, puede ser de interés para futuras aplicaciones del Cubinja en las que se desee mejorar la precisión de los estimadores de posición y velocidad angular
- **SPI:** El uso de SPI como protocolo de comunicación del sensor provee la ventaja de la velocidad en la transmisión y recepción de datos ante otros protocolos como I2C o UART
- **Alta resolución:** El ADC tiene una resolución de 16 bits para el giroscopio, acelerómetro y magnetómetro

3.1. Sensores

- **Fondo de escala configurable:** El rango de medición del giroscopio y acelerómetro pueden configurarse permitiendo una mejor resolución en base a la aplicación del sistema.
- **Probado:** La MPU9250 es una IMU altamente usada en diferentes tipos de proyectos y por lo tanto probada. La gran mayoría de los proyectos basados en el Cubli utilizan la MPU6050 que es una versión anterior al MPU9250.

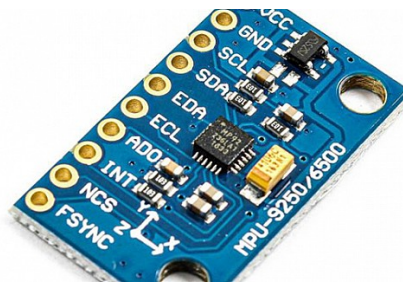


Figura 3.2: Development kit del MPU9250 utilizado en el Cubinja. Imagen tomada de [16]. Hoja de datos en F.

3.1.4. Estimador de posición angular

Muchas IMU's son capaces de procesar los datos del acelerómetro y giroscopio para devolver un ángulo de referencia del sensor. Este no es el caso del MPU9250. Con la IMU utilizada lo único que podemos obtener son los datos en crudo de los sensores.

Por esta razón se necesita implementar un algoritmo de fusión sensorial que permita combinar los datos de los sensores para adquirir una referencia de la posición del sistema.

En esta sección se presentarán los dos algoritmos investigados para la aplicación y se justifica la elección del algoritmo finalmente implementado

Método de dos acelerómetros

El método de dos acelerómetros es un método desarrollado por el propio Cubli para estimar la posición angular del sistema. Ya que un acelerómetro puede medir el vector gravedad son muy buenos sensores para el cálculo de la posición angular de un sistema en condiciones estáticas. El problema es que fuera de estas la medida del sensor no es solamente la gravedad, sino que también se encuentran magnitudes dependientes de la velocidad y aceleración angular del sistema.

Debido a esto un solo acelerómetro no puede ser utilizado fuera de condiciones estáticas, el Cubli implementa un método utilizando dos acelerómetros en el sistema para que al realizar una medida en simultáneo se puedan eliminar las componentes dependientes de la velocidad y aceleración angular y obtener una medida del vector gravedad.

Capítulo 3. Sensores y actuadores

Este método consiste en colocar dos acelerómetros en la misma diagonal como se muestra en la figura 3.3

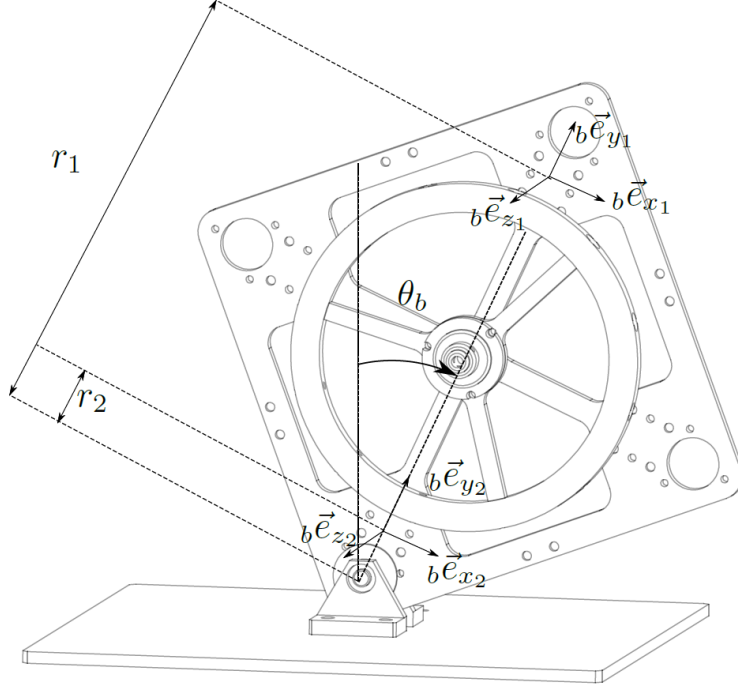


Figura 3.3: Diagrama que muestra la posición de los acelerómetros relativos al frame, los sistemas de coordenadas $b\vec{e}_i$ representan los sistemas de referencia de cada IMU. Diagrama sacado de [2].

Se puede escribir la lectura de cada acelerómetro como

$$a_i = (\vec{a}_i \cdot \vec{e}_{xi}, \vec{a}_i \cdot \vec{e}_{yi}, \vec{a}_i \cdot \vec{e}_{zi}) = (r_i \ddot{\theta}_b + g \sin \theta_b, r_i \dot{\theta}_b^2 - g \cos \theta_b, 0) \quad (3.1)$$

Por lo tanto definiendo $\mu = r_1/r_2$, y operando llegamos al siguiente resultado

$$a_1 - \mu a_2 = ((1 - \mu)g \sin \theta_b, -(1 - \mu)g \cos \theta_b, 0) := (m_x, m_y, 0) \quad (3.2)$$

Y de esta forma podemos escribir el estimador $\hat{\theta}_b$ como

$$\hat{\theta}_b = \tan^{-1}(-m_x/m_y) \quad (3.3)$$

Este método es sencillo de implementar y con un bajo costo computacional al solo tener que hacer una operación trigonométrica sobre las medidas de los sensores. El problema es que en la práctica los acelerómetros utilizados son muy susceptibles al ruido mecánico del sistema. Al encender los motores la vibración de los mismos agrega un ruido no despreciable a las medidas haciendo muy difícil el control del sistema en base a estas lecturas.

Además para el Cubinja se precisarían un total de 6 IMUs para la estimación completa del vector de estado, incrementando el costo en pines del sistema

Filtro complementario

A diferencia de los acelerómetros, los giroscopios tienen una menor sensibilidad al ruido mecánico, además existe una gran variedad de algoritmos que fusionan las lecturas de los acelerómetros con la de los giroscopios para obtener estimadores de la posición relativa del sistema, esto hace que ir por un algoritmo de este estilo en el Cubinja sea una buena alternativa al método de dos acelerómetros. En esta sección se profundiza en uno de estos algoritmos de fusión sensorial. El filtro complementario

El filtro complementario es una combinación de dos filtros diferentes: un filtro de paso bajo y un filtro de paso alto. El filtro de paso bajo se utiliza para filtrar la señal de los acelerómetros, que proporcionan información sobre la aceleración lineal del dispositivo. El filtro de paso alto se utiliza para filtrar la señal de los giroscopios, que proporcionan información sobre la velocidad angular del dispositivo.

La idea detrás del filtro complementario es que los giroscopios son buenos para medir cambios en la orientación en el corto plazo, pero se ven afectados por el ruido y principalmente por el *bias* a largo plazo.

El *bias* de un giroscopio es un error variable en el tiempo, pero que se puede asumir constante en un periodo corto de tiempo, dicho *bias* produce una lenta deriva en el ángulo estimado al integrar la velocidad angular brindada por el giroscopio.

Por otro lado, los acelerómetros son buenos para medir el vector gravedad y la aceleración lineal, pero se ven afectados por el ruido de alta frecuencia y como se vio anteriormente fuera de situaciones estáticas, la medida del vector gravedad puede ser incorrecta.

Sin embargo, en el largo plazo la medida de un acelerómetro se mantiene relativamente estable a pesar del ruido. Al combinar las mediciones de ambos sensores, se pueden obtener estimaciones precisas de la orientación y la posición del dispositivo. La idea principal será integrar la velocidad angular provista por el giroscopio y luego utilizar los datos provistos por el acelerómetro para compensar la deriva en el ángulo calculado

En la figura 3.4 puede verse un diagrama del flujo de datos de los sensores para la implementación del filtro complementario

El primer paso consiste en obtener un estimador de la posición en base a los datos del acelerómetro. En este estimador se asume que el sistema está estático y, por lo tanto, midiendo el vector gravedad se obtiene una medida de la posición angular.

Luego de esto se integran los datos del giroscopio para obtener un segundo estimador de la posición angular.

Finalmente, se combinan los datos de ambos sensores ponderándolos por una constante α que se encuentra entre 0 y 1.

Dicha constante es ajustada según la aplicación, α es básicamente una medida de la fiabilidad del acelerómetro en la medida final. Lo normal es que α se encuentre próximo a 0, ya que como se mencionó anteriormente, el acelerómetro es principalmente utilizado para compensar la deriva en el ángulo en la integración de los datos del giroscopio.

Capítulo 3. Sensores y actuadores

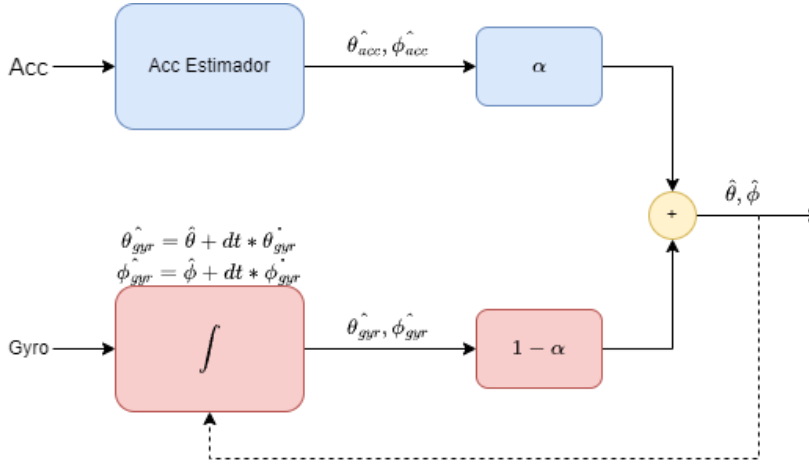


Figura 3.4: Diagrama que muestra el manejo de datos del acelerómetro y giroscopio para el método del filtro complementario. Notar que en la integral realizada se utiliza $\hat{\theta}$ y $\hat{\phi}$ y no $\hat{\theta}_{gyr}$ y $\hat{\phi}_{gyr}$

Por lo tanto, podemos escribir $\hat{\theta}$ y $\hat{\phi}$ de la siguiente forma¹:

$$\hat{\theta}_{n+1} = \alpha * \hat{\theta}_{acc} + (1 - \alpha) * (\hat{\theta}_n + dt * \dot{\theta}_{gyr}) \quad (3.4)$$

$$\hat{\phi}_{n+1} = \alpha * \hat{\phi}_{acc} + (1 - \alpha) * (\hat{\phi}_n + dt * \dot{\phi}_{gyr}) \quad (3.5)$$

en donde dt es el periodo de muestreo del sistema.

Debido a la fácil implementación del filtro complementario y a los buenos resultados obtenidos como se mostrará en el capítulo 9 se decide ir por este método para el cálculo de los estimadores de la posición angular del sistema.

$$\hat{\theta}_{acc} = \text{sen}^{-1}\left(\frac{ax}{g}\right) \quad (3.6)$$

$$\hat{\phi}_{acc} = \text{tan}^{-1}\left(\frac{ay}{az}\right) \quad (3.7)$$

$$\hat{\phi}_{gyr} = \hat{\phi} + dt * \dot{\phi}_{gyr} \quad (3.8)$$

$$\hat{\theta}_{gyr} = \hat{\theta} + dt * \dot{\theta}_{gyr} \quad (3.9)$$

3.1.5. Estimador de velocidad angular

La velocidad angular del sistema puede ser estimada directamente de los datos del giroscopio por lo tanto no hay necesidad de implementar ningún algoritmo de estimación como en el caso de la posición angular.

¹El análisis teórico del filtro complementario y todas las ecuaciones planteadas en esta sección pueden verse en [17]

Sin embargo para mejorar la condición de ruido del sistema se implementara un EMWA (del inglés Exponential Moving Weighted Average) como se explica en la siguiente sección.

Exponential Moving Weighted Avarage

El EMWA es un tipo de filtro utilizado en el procesamiento de señales y análisis de datos para suavizar series de tiempo. También se conoce como filtro de media móvil exponencial.

El EMWA es especialmente útil cuando se desea reducir el ruido en una serie de tiempo, otorgando más peso a los datos más recientes. A diferencia de la media móvil simple, donde todos los puntos tienen el mismo peso, la EMWA asigna un peso que disminuye exponencialmente a los datos históricos.

Podemos estimar las velocidades angulares del sistema de la siguiente forma¹

$$\hat{\theta}_{n+1} = (1 - k) * \dot{\theta}_{gyr} + k * \hat{\theta}_n \quad (3.10)$$

$$\hat{\phi}_{n+1} = (1 - k) * \dot{\phi}_{gyr} + k * \hat{\phi}_n \quad (3.11)$$

En donde k se denomina factor de suavizado.

Las ventajas del EMWA sobre otras técnicas de filtrado son:

- **Sensibilidad a los datos recientes:** Debido a que el EMWA asigna un peso exponencialmente decreciente a los datos históricos, da mayor importancia a los datos más recientes. Esto es útil en situaciones en las que los cambios recientes son más relevantes que los cambios pasados.
- **Menor retardo:** El EMWA tiene un menor retardo en comparación con los filtros de media móvil simple, ya que da mayor peso a los datos recientes. Esto puede ser beneficioso cuando se necesita una respuesta más rápida a los cambios en la serie de tiempo.
- **Elección del factor de suavizado:** El factor de suavizado en el EMWA puede ajustarse según las necesidades específicas. Un valor más alto de k dará más peso a los datos recientes, lo que resultará en una mayor adaptabilidad a cambios rápidos. Por otro lado, un valor más bajo de k dará más importancia a los datos históricos, lo que puede ser útil para suavizar fluctuaciones de corto plazo.

Además en la práctica tiene una implementación sencilla y de muy bajo costo computacional haciéndolo de gran utilidad para aplicaciones que requieran un manejo rápido de datos

¹Ecuaciones sacadas de [18] y [19]

3.1.6. Encoder

Para estimar la velocidad angular de los volantes de inercia se utilizaron encoders ópticos.

Los encoders ópticos son dispositivos utilizados para medir la posición, velocidad o desplazamiento angular en sistemas mecánicos. Funcionan mediante la detección de la luz emitida por una fuente de luz y su interacción con un disco o cinta codificada que contiene patrones o marcas. En la figura 3.5 puede verse un diagrama de un encoder óptico típico

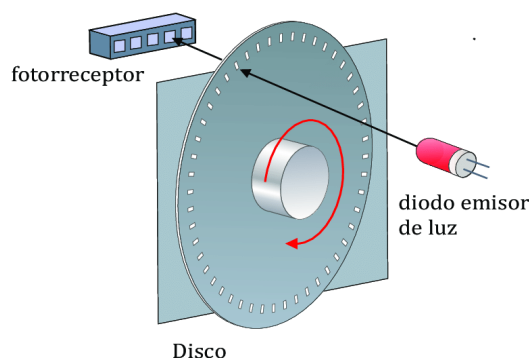


Figura 3.5: Diagrama en el que se muestran los componentes principales de un encoder óptico. Imagen sacada de [20].

El emisor de luz, como lo puede ser un LED, emite luz constantemente y gracias al disco codificado se produce un patrón óptico el cual es detectado por el foto receptor el cual genera una señal cuadrada cuyo periodo es proporcional a la velocidad angular del disco

3.1.7. Selección de encoder

Para simplificar la integración de todos los componentes del sistema se buscó un motor que ya tuviera algún sistema de detección de la velocidad de giro. Por lo tanto, los encoder del sistema serán los propios del Nidec 24H404H.

Los encoder de este motor son lo que se conocen como encoder en cuadratura, el principio de funcionamiento de estos encoders son exactamente como se describió en la sección anterior, solo que en vez de tener un emisor y un receptor tienen dos separados una distancia de exactamente $1/4$ de ranura del disco codificado. En la figura 3.6 puede verse el patrón de voltaje de salida de los dos fotorreceptores.

El defasaje de $1/4$ de ranura de los fotorreceptores produce que las señales producidas por estos estén desfasadas $1/4$ de periodo (esto es 90° si asumimos un periodo de 360° de ahí el nombre encoder en cuadratura), el posicionamiento de los sensores de esta forma produce que sin hacer modificación alguna en el disco de codificado pueda detectarse no solo la velocidad de giro sino también el sentido de giro de la misma que este dependerá si la señal B está desfasada 90° o -90° con respecto a la señal en A.

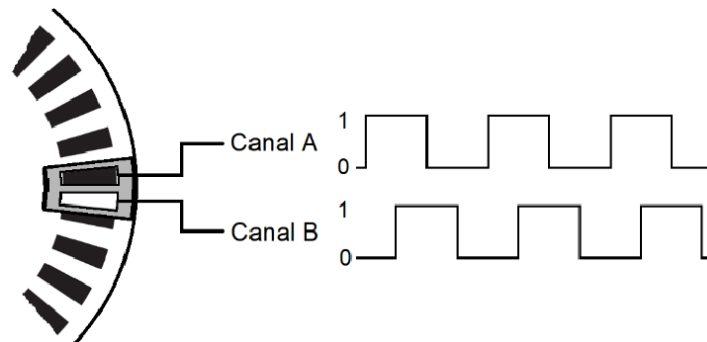


Figura 3.6: Diagrama en el que se muestra el patrón de señal en un encoder en cuadratura, puede verse que las señales de salida de los foto receptores están desfasadas un cuarto de periodo. Figura tomada de [21].

3.1.8. Estimador de velocidad del volante

Existen dos métodos típicos para estimar la velocidad del volante a partir de la señal cuadrada provista por el encoder, el método de distancia fija y el método de tiempo fijo

Método de tiempo fijo

El método de tiempo fijo consiste en contar la cantidad de pulsos que el encoder proporciona en un intervalo de tiempo fijo. Luego conociendo la relación de pulsos por vuelta del volante podemos calcular la velocidad del mismo de la siguiente forma:

$$RPM = 60NF_s/PPV \quad (3.12)$$

En donde N es la cantidad de pulsos en un periodo de muestreo, F_s es la frecuencia de muestreo y PPV es la cantidad de pulsos que se generan en una vuelta del volante. Dentro de las ventajas de este método se encuentran

- Tiempo de actualización constante, la obtención de la velocidad se hace en un tiempo fijo dado por la frecuencia de muestreo del sistema
- Facilidad de implementación. Si bien este método posee desventajas como la alta sensibilidad al ruido y la baja precisión para velocidades bajas fue el elegido debido a que para la aplicación tener un tiempo de actualización constante era fundamental para la sincronización de datos con el resto de los sensores

Capítulo 3. Sensores y actuadores

Método de distancia fija

En el método de distancia fija se calcula el periodo entre dos pulsos. Si se sabe la distancia angular entre dos rendijas consecutivas en el disco del encoder se puede calcular la velocidad midiendo el periodo del pulso.

Las ventajas principales de este método son:

- Alta precisión en el cálculo de velocidad
- Alto rechazo al ruido

La desventaja de este método es que, a diferencia del método de distancia fija, no es tan sencillo de implementar.

Puede sobrecargar al sistema a velocidades muy rápidas debido a la cantidad de interrupciones que se generarían por cada pulso.

Además a velocidades muy lentas podría generarse una latencia en el sistema debido a que el sistema debe esperar a que se complete un periodo completo antes de tomar la medida de la velocidad.

Por esta razón y por los buenos resultados arrojados por el método de tiempo que se presentaron en el capítulo 9 se descarta el método de distancia fija

3.2. Actuadores

3.2.1. Descripción general

En esta sección se detallan los aspectos relacionados al uso de actuadores del sistema 1D, trasladables al sistema 3D. Se resumen las principales características y principios de funcionamiento de los motores de corriente continua, en particular del motor sin escobillas (BLDC).

Este motor fue seleccionado para generar el torque que hace girar al volante de inercia, el cual se encuentra acoplado a su eje. El segundo actuador tratado en el capítulo es el servo motor encargado de desencadenar el evento de frenado del volante. Esta acción realiza el torque sobre el sistema que genera el levantamiento de forma controlada del mismo, llevándolo lo mas próximo posible a su punto de equilibrio.

3.2.2. Motores BLDC

Para la selección de motores se comenzó analizando las características que los mismos deben cumplir. Basados en el estudio de proyectos anteriores y previendo las distintas acciones que el mismo debe desempeñar se llegó a los siguientes requerimientos.

- Motor fiable, de rendimiento constante y confiable.
- Vida útil prolongada
- Mantenimiento nulo
- Eléctrico, con la posibilidad de alimentarlo desde una batería
- Dimensiones pequeñas
- Accesible en el mercado local o de fácil envío
- Que cumpla especificaciones físicas calculadas, en particular el torque instantáneo

El tipo de motor que mejor cumple esta serie de especificaciones y es ideal para desempeñar la tarea designada es el motor brushless DC (llamado por sus siglas BLDC.)

Comenzaremos por dar una breve explicación respecto al funcionamiento del motor de corriente continua, para luego especificar las características particulares del motor BLDC.

Para poder equilibrar el frame, se regula la corriente del motor y de esta forma la aceleración y velocidad del volante. De esta forma, conociendo los parámetros

Capítulo 3. Sensores y actuadores

del motor y del sistema podemos controlar el torque entregado por el motor para balancear el frame. Los motores DC son sistemas electromecánicos rotativos que convierten energía eléctrica en mecánica. En la figura 3.7 se pueden ver los diagramas correspondientes al funcionamiento del motor, eléctrico y mecánico.

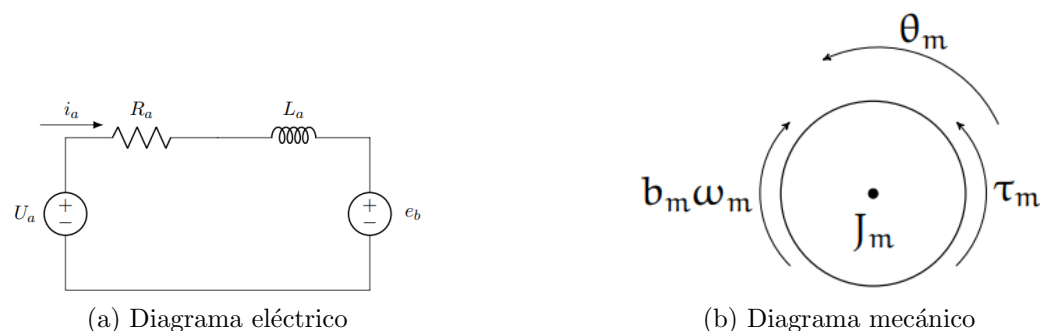


Figura 3.7: Diagramas de motor DC, extraído de [10].

Las ecuaciones que rigen el comportamiento del motor son las siguientes ¹:

$$U_a = R_a i_a + L_a \frac{di_a}{dt} + k_e \omega_m \quad (3.13)$$

Donde $k_e \omega_m = e_b$ y $k_t i_a = \tau$.

Además,

$$\tau = k_t i_a \quad (3.14)$$

$$J_m \ddot{\theta}_m = \tau - b_m \omega_m \quad (3.15)$$

De donde vemos que la relación entre la corriente de entrada del circuito y el torque del motor está dada por la constante k_t .

A continuación se detallan los parámetros y nomenclatura vistos

¹Las ecuaciones vistas en esta sección fueron tomadas de [22]

Notación	Detalle	Unidad
U_a	Tensión de alimentación	[V]
R_a	Resistencia de armadura	[Ω]
L_a	Inductancia de armadura	[H]
e_b	Voltaje contra electro motriz	[V]
k_e	Constante de voltaje	[Vs/rad]
k_t	Constante de torque	[Nms/rad]
ω	Velocidad angular del motor	[rad/s]
J_m	Inercia del motor	[kg.m ²]
b_m	Factor de fricción del motor	[Nms/rad]
τ_m	Torque del motor	[Nm]
θ_m	Posición del motor	[rad]

Los motores BLDC (Brushless DC) son un tipo de motor eléctrico que no utiliza escobillas (brushes) en el rotor, sino que en su lugar usa imanes permanentes en el rotor y un conjunto de bobinas de cobre en el estator para generar el campo magnético que produce el movimiento. Las distintas bobinas son excitadas secuencialmente por un controlador, muchas veces interno, que genera el movimiento deseado.

Algunas de sus ventajas con respecto a los motores de corriente continua convencionales son que generan mayor velocidad y potencia, por lo general tienen mayor eficiencia y menor consumo de energía. Al tener menor rozamiento interno debido a que no tiene escobillas, el nivel de ruido que genera es bajo. Por otro lado su control de velocidad, muchas veces realimentado por un encoder, es preciso y asegura una velocidad constante. A su vez, el mantenimiento de los mismos es prácticamente nulo ya que al no tener escobillas las mismas no se desgastan ni deben ser remplazadas.

Por otro lado, algunas de las desventajas de los motores BLDC pueden ser su costo, que es mayor al de un motor DC convencional y la complejidad que requiere su control electrónico, el mismo precisa de distintos componentes para asegurar el desempeño del motor.

3.2.3. Selección de BLDC

Para la elección del motor BLDC, se tuvieron en cuenta las siguientes características:

- RPM
- Peso
- Torque
- Tensión de alimentación

Capítulo 3. Sensores y actuadores

- Corriente de alimentación
- Potencia
- Dimensión
- Precio
- Accesibilidad
- Vigencia
- Tipo de control
- Fidelidad

Se hizo un estudio de antecedentes y proyectos que incluyen motores BLDC con el objetivo de visualizar opciones posibles de motores ya trabajados en aplicaciones similares. Esto permite identificar problemas existentes en distintas configuraciones, facilitando la evaluación y selección del motor más adecuado para este proyecto.

Se comenzó estudiando proyectos que tuvieran el mismo objetivo final que el nuestro, lograr balancear un cubo ya sea en una de sus aristas o vértices. Luego se procedió a estudiar proyectos con aplicaciones comparables que fueran desarrollados en el marco de Facultad de Ingeniería. Es así que profundizando en los papers e información de estos proyectos se obtuvieron marcas y modelos de motores evaluados y luego utilizados en los mismos. A continuación se presentan brevemente los resultados de esta investigación, con su consecuente resumen de características por motor visto en el cuadro 3.8, todos ellos con tensiones nominales entre 12 y 24 VDC.

- **Proyecto CUBLI [2]:** Fue tomado como referencia y camino a seguir a lo largo de todo el proyecto, por tanto se procedió a analizar las características del motor que se utiliza. Este es un BLDC EC 45 FLAT del fabricante Maxon (Suiza). Sus principales características se encuentran detalladas en el cuadro adjunto. La principal ventaja de este motor, frente a cualquier otro candidato, es que su desempeño y resultados son conocidos. Estos se verifican con el éxito rotundo y cumplimiento de objetivos del proyecto. Vemos que tanto la tensión nominal como la corriente nominal son alcanzables sin dificultad mediante el uso de una batería recargable, lo cual condice con el requerimiento de autonomía del cubo. Sus dimensiones son adecuadas, permitiendo que el cubo sea fácilmente maniobrable.
- **Proyectos Termodrón I, II y II (Prof. Rafael Canetti) [23]:** Esta serie de proyectos cuentan con un análisis detallado de motores de tipo BLDC dado que el mismo es la pieza central en un dron. Es así que se obtuvieron los datos referentes a los motores Emax GT2820, Emax MT3510 y Emax

3.2. Actuadores

GT2215/12, siendo este último el utilizado en la tercera y mejorada versión del dron armado en el proyecto por su óptima relación peso/empuje. Como se puede observar en el cuadro, las corrientes máximas que los motores pueden llegar a consumir resultan considerablemente altas cuando se los alimenta desde una batería.

Además de estos proyectos en particular que sirvieron de referencia en un principio, se encontraron otra cantidad de proyectos de menor presupuesto que utilizaban motores más en cuenta, de menor calidad pero menos costosos en comparación a los vistos anteriormente. Es así que se llega a conocer una versión simplificada del motor utilizado en el prototipo del proyecto *The Cubli* que resultó siendo el seleccionado para usar en este proyecto.

Característica / Modelo	Emax GT2820	Emax MT3510	Emax GT2215/12	Nidec 24H404H	EC 45 flat
Tensión [V]	12/24	12/24	12/24	12/24	12/24
RPM/V	850	600	905	250	259
RPM	7600	4840	7450	6000	6110
I _{max} [A]	26	15	15	3	3.86
Potencia [W]	848	166	216	25	70
Torque max [mN.m]	1500	320	420	50	149
Dimensiones [mm x mm]	46.5 x 35	41.5 x 31.8	36.5 x 28.5	52 x 45	59 x 50
Peso [g]	140	102	70	115	260
Precio [USD]	23	32	16	18	137
Control	Externo	Externo	Externo	Interno	Interno

Figura 3.8: Cuadro comparativo de características de motores BLDC. Hoja de datos en F.

El motor seleccionado es el Nidec 24H404H utilizado en el proyecto *Self Balancing Cube* de Rem-RC [11] por ser el que más se adecua a nuestras necesidades.

Como se puede apreciar en el cuadro, este motor es más liviano que el original del CUBLI, lo que nos da un cierto margen para cargar al cubo con otros pesos en caso de ser necesario. Su costo es casi ocho veces menor, las RPM alcanzables son prácticamente las mismas, dimensiones similares, y cumple una característica fundamental que es que incluye un sistema de control interno. Esto quiere decir que no precisa de un driver externo, su comando de control consiste en una PWM donde fijando su *duty cycle* se fija la tensión de alimentación del motor. Además, el driver interno cuenta con control de lazo cerrado y un *encoder* de dos canales, permitiendo evaluar la posición del motor en tiempo real, fundamental para el control realimentado del frame/cubo.

Los ensayos correspondientes al desempeño del motor se hallan en el capítulo 9, en la sección 9.2.1.

3.2.4. Servo motores

Para poder dotar al cubo o frame con la posibilidad de incorporarse desde el reposo por sí mismo (*swing up*) se investigaron en los antecedentes los distintos

Capítulo 3. Sensores y actuadores

diseños ya testeados.

La totalidad de los proyectos estudiados cuentan con algún sistema de frenado del anillo de inercia por medio de distintos métodos mecánicos accionados por un servo motor. Se procedió a analizar los mismos con el fin de seleccionar el de mejor aplicación para el proyecto. El proceso de estudio de sistemas de frenado se halla en el capítulo 5, donde se puede acompañar el proceso de selección que resulta en el sistema de frenado por zapata.

Para implementar este sistema es preciso contar con un servo motor de dimensiones pequeñas, liviano, con la velocidad de giro y fuerza suficiente para parar al volante de inercia. Tras investigar el mercado, antecedentes y proyectos similares se redujo la elección a 3 opciones, vistas en el siguiente cuadro (3.9).

Característica / Modelo	MG90S	SCS0009	TS90A
Voltaje de operación [V]	3 - 7.2	4.8 - 7.4	3 - 6
Torque max [kg x cm]	2.5	2.3	1.3
Velocidad [seg / 60 °]	0.1	0.07	0.12
Engranajes	Metal	Metal - Plástico	Plástico
Dimensiones [mm x mm x mm]	22 x 11.5 x 27	23.2 x 12 x 25.5	23.2 x 12.5 x 22
Peso [g]	13.4	18	9

Figura 3.9: Cuadro comparativo de características de servo motores. Hoja de datos en F.

El motor seleccionado para comenzar las pruebas fue el MG90S, de la marca TowerPRO. Es un motor estándar para implementaciones del estilo, trabajos en Arduino y proyectos de pequeño porte. Se encuentra en el mercado local y su costo es bajo. Permite giros de hasta 180° variando su PWM de alimentación. Su torque máximo es de 2,5[kg.cm] y su velocidad es de 0,1[seg/60°] ¹

Algunas de las experiencias que se tuvieron a la hora de trabajar con este servo motor a lo largo del proyecto indican que no es un motor de alta fidelidad ni calidad. Existen muchas copias de fabricantes ajenos a TowerPRO de menor calidad, con comportamientos erráticos, de los cuales los proveedores locales no estaban en conocimiento. Se tuvo que reemplazar varios motores por roturas inesperadas.

Más allá de estas experiencias, el servo motor desempeñó su función correctamente. Los ensayos pertinentes al sistema de frenado se pueden ver en la sección 9.1.5 del capítulo 9.

¹La unidad de velocidad (sin carga) evaluada y vista en el cuadro es la que se utiliza normalmente para medir los servo motores. Esta especificación indica el tiempo que tarda el servo en moverse 60 grados.

Capítulo 4

Modelado del frame y controlador

En este capítulo se analizará la dinámica del Frame con el objetivo de comprender su funcionamiento y lograr representarlo en un modelo con el cual se pueda realizar el diseño de un controlador que permita mantener al frame en su punto de equilibrio inestable.

Esto quiere decir que no se profundizará en el modelo al punto de llegar a un sistema que represente exactamente la dinámica del sistema, sino que se busca un modelo lo suficientemente coherente como para que el diseño del controlador cumpla su objetivo de manera aceptable.

Con este objetivo entonces se comenzó por plantear el problema físico que nos lleve al sistema de ecuaciones correspondiente. Luego se linealizó el mismo con respecto a su punto de equilibrio en el cual se pretende trabajar. En base al resultado obtenido en esta última parte se diseñó el controlador correspondiente.

4.1. Modelado del problema unidimensional

El puntapié inicial para comenzar a introducirse en el problema de control a resolver implica modelar el comportamiento del sistema. Para ello, basados en el estudio de antecedentes y en particular en los papers referenciados al proyecto *The Cubli* [2] se toma la decisión de analizar el problema en su forma unidimensional (1D).

Esto es debido a que los comportamientos de ambos prototipos, el 1D y el 3D son similares, y su fundamento y trasfondo mecánico y dinámico son comparables.

No es el objetivo del proyecto profundizar en el modelado de la dinámica 3D, sino sentar las bases para comprender correctamente el comportamiento del sistema. A continuación, a lo largo del capítulo 4 se puede seguir un desarrollo simplificado del mismo.

4.1.1. Hipótesis del modelado

El modelado se basará en las siguientes hipótesis:

Capítulo 4. Modelado del frame y controlador

- Para describir la dinámica del sistema, es suficiente considerar el acoplamiento de dos cuerpos rígidos. El primero será el frame y el segundo será el volante de inercia acoplado al motor, el cual, visto desde el eje de pivote, puede ser considerado una masa puntual en el centro del frame.
- Tanto el frame como el volante pueden ser modelados como figuras bidimensionales.
- El volante de inercia puede ser modelado como un anillo.
- El rulemán puede considerarse una masa puntual colocada exactamente en el punto de pivote.
- Las únicas fricciones en el sistema serán las del eje del motor y la del eje del rulemán, mientras que el resto serán despreciadas.
- Se modelará la fricción en el eje del motor como una fuerza proporcional a la velocidad del volante.
- Se modelará la fricción en el eje del rulemán como una fuerza proporcional a la velocidad angular del frame.
- La conexión entre el frame y el sistema volante-motor es una conexión rígida.
- Se considera que el volante está bien balanceada y solo gira en la dirección perpendicular al plano del frame.
- Todos los materiales son homogéneos.
- La base está fija y no se mueve.

4.1.2. Componentes y Nomenclatura

En esta sección se presentan los principales componentes dinámicos del sistema, así como también los sistemas de referencia y coordenadas utilizados para la realización del modelado.

El modelo unidimensional está compuesto esencialmente de dos partes, el *frame* y el *volante*. El frame consiste en un marco idealmente cuadrado en el que en el corte de sus diagonales se coloca el motor BLDC. En el eje del motor se acopla por medio de una estría dentada un anillo que funcionará como volante de inercia.

Además del motor con el anillo acoplado en su centro, el frame cuenta con un rulemán en una de sus esquinas. El mismo se coloca con el objetivo de fijar el frame a una plataforma inmóvil, minimizando el rozamiento entre el frame y el soporte. La siguiente funcionalidad del rulemán y su sujetador es restringir el movimiento del frame para que solo puede moverse en una dimensión, impidiendo que se desplace en sentido perpendicular. La implementación del sistema se puede

4.1. Modelado del problema unidimensional

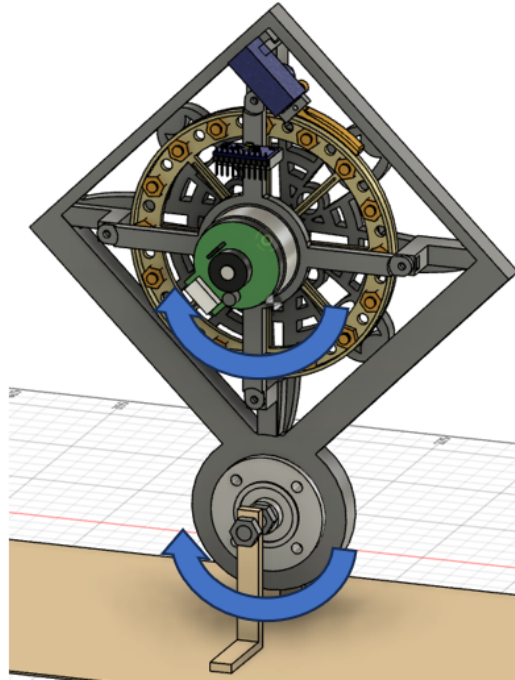


Figura 4.1: Sistema frame. Rulemán restringe el movimiento en el plano con mínimo rozamiento.

ver en la figura 4.1.

Para el modelado físico del sistema se trabajará en coordenadas polares, las cuales son las más indicadas para representar el balanceo angular con respecto a la horizontal. En el diagrama de la figura 4.2 se pueden apreciar los principales parámetros del sistema. A continuación se los detalla.

El ángulo θ indica la desviación de la posición de la vertical del frame con respecto a su posición de equilibrio, siendo $\theta = 0^\circ$ la posición de equilibrio inestable.

En la tabla 4.1 se detalla la nomenclatura y unidades de los principales parámetros del sistema, estos serán utilizados en el análisis de la dinámica del sistema a lo largo de este capítulo y en la serie de ensayos realizados al prototipo.

4.1.3. Construcción del modelo

Para la construcción del modelo en variables de estado se optó por una metodología simple y que no presenta demasiadas diferencias con respecto a estudios ya realizados sobre este sistema. Los fundamentos físicos del sistema han sido analizados en numerosos antecedentes. A continuación se presenta una recopilación y adecuación de los mismos a las necesidades de este proyecto.

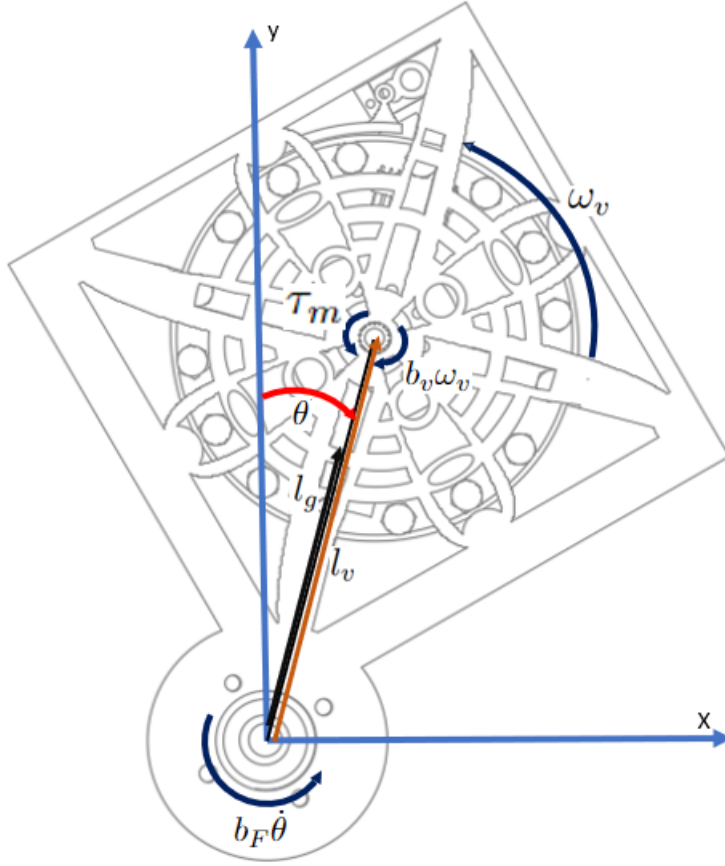


Figura 4.2: Diagrama completo del sistema frame + volante.

Comenzamos aplicando la segunda cardinal sobre el sistema frame + volante desde el origen del sistema, que se ubica en el centro del rulemán. El análisis del sistema se realiza en una dimensión para simplificar los cálculos.

Como se mencionó en las hipótesis del modelado desde el eje de rulemán podemos modelar el rígido entero como el frame + una masa puntual de valor $(m_v + m_m)$ en el centro del frame a una distancia l_v del origen.

Se obtiene entonces la ecuación 4.1

$$(I_F + (m_m + m_v)l_v^2)\ddot{\theta}(t) = -b_F\dot{\theta}(t) + \tau_G - \tau_m + b_v\omega_v \quad (4.1)$$

Con τ_G definido de la siguiente manera:

$$\tau_G = g(l_g m_F + l_v(m_v + m_m))\sin(\theta) \quad (4.2)$$

En primer lugar, vemos el aporte negativo del rozamiento en el rulemán, expresado como $-b_F\dot{\theta}(t)$ y el par realizado por el peso (τ_G). Este se puede separar en dos partes, primero el par debido al peso de la rueda y motor y otro debido al peso del frame

4.1. Modelado del problema unidimensional

Notación	Detalle	Unidad
τ_m	Torque del motor	[N m]
I_F	Inercia del frame vista desde el centro del rulemán	[m ² kg]
I_v	Inercia del volante vista desde el eje del motor	[m ² kg]
$F_{x,y}$	Fuerzas según ejes x e y centradas en el volante	[N]
g	Aceleración gravitatoria	[m/s ²]
l_g	Distancia del centro del rulemán al centro de masa del frame	[m]
l_v	Distancia del centro del rulemán al centro del volante	[m]
$\theta(t)$	Ángulo entre la vertical y la diagonal del frame	[rad]
$\dot{\theta}(t)$	Velocidad angular del frame	[rad/s]
$\omega_v(t)$	Velocidad angular del volante referenciada a sistema solidario al eje del motor	[rad/s]
$\ddot{\theta}(t)$	Aceleración angular del frame	[rad/s ²]
$\dot{\omega}_v(t)$	Aceleración angular del volante	[rad/s ²]
b_F	Coefficiente de rozamiento en el rulemán	[Nms/rad]
b_m	Coefficiente de rozamiento en eje del motor	[Nms/rad]
m_F	Masa del frame y del volante	[kg]
m_v	Masa del volante	[kg]
m_m	Masa del motor	[kg]
k_t	Constante de torque del motor BLDC	[Nm/A]

Tabla 4.1: Parámetros y variables del sistema

Además, aparecen los componentes $-\tau_m + b_v\omega_v$ siendo estas dos las fuerzas reactivas a la que realiza el motor sobre el volante.

A partir de esta primera ecuación ya podemos hacer varias observaciones, evidentemente el par debido al peso será el principal encargado de sacar al sistema de su punto de equilibrio. Un mayor rozamiento en el eje del rulemán dificultará la caída del sistema.

La observación más importante es que el torque realizado por el sistema volante + motor aparece directamente sobre la ecuación que rige el movimiento del frame. Esto significa que este será el principal encargado de restaurar el sistema (o hacer más pronta su caída) a su punto de equilibrio.

Para simplificar las cuentas definiremos las siguientes cantidades:

$$I_{tot} = I_F + (m_m + m_v)l_v^2 \quad (4.3)$$

$$K_{mgl} = g(l_g m_f + l_v(m_v + m_m)) \quad (4.4)$$

Y sustituyendo estas dos nuevas cantidades en 4.5 tenemos

$$I_{tot}\ddot{\theta}(t) = -b_F\dot{\theta}(t) + K_{mgl}\sin(\theta(t)) - \tau_m + b_v\omega_v \quad (4.5)$$

Y despejando $\ddot{\theta}(t)$

$$\ddot{\theta}(t) = \frac{K_{mgl}\sin(\theta(t))}{I_{tot}} - \frac{\tau_m}{I_{tot}} + \frac{b_v\omega_v(t) - b_F\dot{\theta}(t)}{I_{tot}} \quad (4.6)$$

La primer fracción de la ecuación resultante es debida al torque generado por el peso de los elementos. La segunda debido al torque de entrada del motor y el tercer componente debido las fricciones del rulemán y del volante en el eje del

Capítulo 4. Modelado del frame y controlador

motor.

Luego de esto pasamos a analizar el sistema motor-volante, comenzamos planteando la segunda cardinal desde un sistema solidario al eje del motor obteniendo el siguiente resultado

$$I_v(\dot{\omega}_v(t) + \ddot{\theta}(t)) = \tau_m - b_v\omega_v \quad (4.7)$$

Gracias a la simetría central del volante de inercia, el peso del mismo no realiza un torque sobre este sistema. Por esta razón el par visto desde este punto es solamente el par en el eje del motor.

Observamos que aparece la aceleración, no solo del volante, sino también del frame. Esto se debe a que el origen de este sistema de coordenadas está acelerado con una aceleración de igual valor que la del frame.

Despejando $\dot{\omega}_v(t)$ de la ecuación anterior tenemos que

$$\dot{\omega}_v(t) = \frac{\tau_m - b_v\omega_v}{I_v} - \ddot{\theta} \quad (4.8)$$

Finalmente recordamos las ecuaciones 3.13 y 3.14 en las que se plantea las dependencias eléctricas del BLDC. Sustituyendo τ_m por su valor en función de la corriente obtenemos el siguiente sistema de ecuaciones

$$\ddot{\theta}(t) = \frac{K_{mgl} \cdot \sin(\theta(t))}{I_{tot}} - \frac{k_t \cdot i}{I_{tot}} + \frac{b_v\omega_v(t) - b_F\dot{\theta}(t)}{I_{tot}} \quad (4.9)$$

$$\dot{\omega}_v(t) = \frac{k_t \cdot i - b_v\omega_v}{I_v} - \ddot{\theta} \quad (4.10)$$

$$u(t) = R_a i + L_a \frac{di}{dt} + k_e \omega_v \quad (4.11)$$

$$\tau = k_t i \quad (4.12)$$

En donde $u(t)$ es la tensión impuesta sobre el motor.

4.1.4. Linealización

Del sistema de ecuaciones presentado anteriormente vemos que la única no linealidad del mismo se encuentra en $\ddot{\theta}(t)$, ya que el mismo depende del $\sin(\theta(t))$ siendo este el único parámetro que debe ser linealizado.

Para levantar esta no linealidad se asume que el sistema estará trabajando en pequeñas oscilaciones en torno al origen. Por lo tanto, podemos asumir que $\sin(\theta) \simeq \theta$ Obteniendo el siguiente resultado para $\ddot{\theta}(t)$:

$$\ddot{\theta}(t) = \frac{K_{mgl} \cdot \theta(t)}{I_{tot}} - \frac{k_t \cdot i}{I_{tot}} + \frac{b_v\omega_v(t) - b_F\dot{\theta}(t)}{I_{tot}} \quad (4.13)$$

4.1. Modelado del problema unidimensional

Llamaremos pequeñas oscilaciones entonces a variaciones con respecto al punto de equilibrio de menos de 20 grados. Si bien esto presenta una limitación en el modelado consideramos que 20 grados es un rango suficientemente grande como para el estudio y análisis del sistema.

4.1.5. Ecuaciones en variables de estado

Para una mejor y más simple visualización del problema de control se lleva el modelado a una representación en variables de estado del tipo:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (4.14)$$

$$y(t) = Cx(t) + Du(t) \quad (4.15)$$

En este caso definimos el vector de estado como

$$x(t) = [\theta, \dot{\theta}, \omega_v]$$

Observar que si bien el sistema tiene dos grados de libertad, los cuales son la posición angular del frame y la del volante de inercia, no se toma en cuenta la posición del volante como variable de estado ya que esta no tiene ninguna energía asociada.

Además la corriente no será considerada como una variable de estado ya que la constante de tiempo del circuito RL de los motores eléctricos es mucho menor a la constante de tiempo del sistema mecánico.

Finalmente para simplificar las cuentas tomaremos como hipótesis que la tensión inducida en el motor debido a la *Back EMF* es despreciable. Bajo estas dos nuevas hipótesis podemos hacer la siguiente aproximación para la ecuación 4.11:

$$u(t) \simeq R_a \cdot i \quad (4.16)$$

Y por lo tanto podemos escribir i como:

$$i = u(t)/R_a \quad (4.17)$$

Si ahora sustituimos esta nueva definición de corriente y operamos sobre las ecuaciones 4.9 y 4.10 podemos reescribir el sistema de ecuaciones de la siguiente forma.

$$\ddot{\theta}(t) = \frac{K_{mgl}}{I_{tot}} \theta(t) - \frac{b_F}{I_{tot}} \dot{\theta}(t) + \frac{b_v}{I_{tot}} \omega_v(t) - \frac{k_t}{R_a I_{tot}} u(t) \quad (4.18)$$

$$\dot{\omega}_v(t) = -\frac{K_{mgl}}{I_{tot}} \theta(t) + \frac{b_F}{I_{tot}} \dot{\theta}(t) - \frac{b_v(I_{tot} + I_v)}{I_v I_{tot}} \omega_v(t) + \frac{k_t(I_{tot} + I_v)}{R_a I_{tot} I_v} u(t) \quad (4.19)$$

De las ecuaciones $\ddot{\theta}$ y $\dot{\omega}_v$ podemos definir la matriz A y B como a continuación

Capítulo 4. Modelado del frame y controlador

$$A := \begin{bmatrix} 0 & 1 & 0 \\ \frac{K_{mgl}}{I_{tot}} & -\frac{b_f}{I_{tot}} & \frac{b_v}{I_{tot}} \\ -\frac{K_{mgl}}{I_{tot}} & \frac{b_f}{I_{tot}} & -\frac{b_v(I_{tot}+I_v)}{I_{tot}I_v} \end{bmatrix} \quad (4.20)$$

$$B := \begin{bmatrix} 0 \\ \frac{k_t}{R_a I_{tot}} \\ \frac{k_t(I_{tot} + I_v)}{R_a I_v I_{tot}} \end{bmatrix} \quad (4.21)$$

$$C := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (4.22)$$

$$D := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.23)$$

En donde las variables que representan al estado del sistema serán $(\theta, \dot{\theta}, \omega_v)$. La entrada del sistema será la tensión impuesta al motor, ya que sobre este se implementa una estrategia de control. La salida será la posición angular del frame, ya que esta es la que se pretende controlar para mantener el equilibrio del prototipo. Se recuerda que el parámetro k_t es la constante de torque del motor BLDC, tratada en el capítulo 3, sección 3.2.2.

4.2. Identificación de parámetros

En esta sección se describirán los procesos para la identificación de parámetros del frame, se dividirá la misma en dos subsecciones. Una para la estimación de parámetros del sistema volante + motor y otra para los parámetros del frame sin el volante acoplado.

El enfoque adoptado para identificar los diversos parámetros ha sido descomponer el sistema integral en componentes más manejables. Esto implica simplificar las ecuaciones del modelo a versiones más básicas, pero conservando los mismos parámetros. De esta manera, la identificación del sistema se logra a través del análisis de los parámetros en estos sistemas más simples.

4.2.1. Estimación de parámetros del sistema volante + motor

Los parámetros a estimar de este sistema son los siguientes:

4.2. Identificación de parámetros

Parámetro	Descripción
m_v	masa del volante de inercia
m_m	masa del motor
I_v	momento de inercia del volante
l_v	distancia del volante al eje de rotación
b_v	fricción en el eje del motor

En el caso de m_v , m_m y l_v son fácilmente medibles en el sistema.

Ya que el volante de inercia se puede modelar como un anillo, el momento de inercia del volante visto desde el centro del mismo se puede calcular a partir de la siguiente ecuación: ¹

$$I_v = \frac{1}{2}m_v(r_1^2 + r_2^2) \quad (4.24)$$

Donde r_1 es el diámetro externo del volante, r_2 el diámetro interno, m_v la masa del volante, y además se desprecia la masa de los parantes radiales del volante. La masa y los radios internos y externos del volante son fácilmente medibles.

La fricción en el eje del motor no puede ser directamente medida por ninguna herramienta, para la obtención de este parámetro se realizará el ensayo que se describe a continuación.

Recordando las ecuaciones 3.15 y 3.14 tenemos que:

$$\tau = k_t i_a, I_m \ddot{\theta}_m = \tau - b_m \omega_m$$

Sustituyendo entonces la ecuación 3.14 en 3.15 obtenemos que:

$$I_m \ddot{\theta}_m = k_t i_a - b_m \omega_m$$

Por lo tanto si se alimenta el motor y se deja que el volante alcance velocidad constante se tiene que $\ddot{\theta}_m = 0$ y por lo tanto se obtiene que

$$0 = k_t i_a - b_m \omega_m \quad (4.25)$$

Conociendo entonces la constante k_t del motor y midiendo la corriente y velocidad de la rueda puede despejarse b_m .

La constante k_t puede ser calculada a partir de la hoja de datos del motor de la siguiente forma despejando de la ecuación 3.14:

$$k_t = \frac{\tau}{i_a} \quad (4.26)$$

De la hoja de datos tenemos que ante un par de 25 mNm se obtiene una corriente de 0.7A. Sustituyendo estos datos en la ecuación 4.26 tenemos que

$$k_t = 35,7 \frac{mNm}{A}$$

¹Ecuación tomada de [24].

Capítulo 4. Modelado del frame y controlador

Calculado k_t se pasa a hacer girar el motor a una velocidad conocida y se calcula el consumo de corriente para dicha velocidad. La medida de la velocidad se realiza observando la señal generada por los encoders del motor utilizando un osciloscopio. Para el consumo de corriente se coloca un amperímetro en serie con la alimentación del sistema.

Se obtuvo entonces el set de datos de la tabla 4.2

i_a [mA]	w_m [rad/seg]
16.36	55.6
25.66	86.26
39	113.21
56.6	142.8
79.4	166.13
106.4	191.44
137.1	209.44
174.9	235.41
221	258.57

Tabla 4.2: Resultados obtenidos del ensayo de medición de corriente a distintas velocidades.

Para el calculo de la fricción en el eje se aplica el método de mínimos cuadrados para hallar la recta $w_m(i)$ que mejor aproxime el set de datos. Se obtiene entonces el resultado de la figura 4.3.

Tenemos entonces la recta $w(i) = Ai$ en donde $A = 934,67$ Siendo A :

$$A = \frac{k_t}{b_m} \quad (4.27)$$

Despejando b_m se obtiene entonces que

$$b_v = 38,24 \times 10^{-6} \text{ Nms/rad}$$

Finalmente los parámetros del volante pueden verse en la tabla 4.3

Parámetro	Valor
m_v	129g
m_m	136g
I_v	$0.4 \times 10^{-3} \text{ kgm}^2$
l_v	12.59cm
b_v	$38,24 \times 10^{-6} \text{ Nms/rad}$

Tabla 4.3: Resumen de parámetros que caracterizan al volante de inercia

4.2.2. Estimación de parámetros del frame

En el caso del frame los parámetros a estimar son los siguientes:

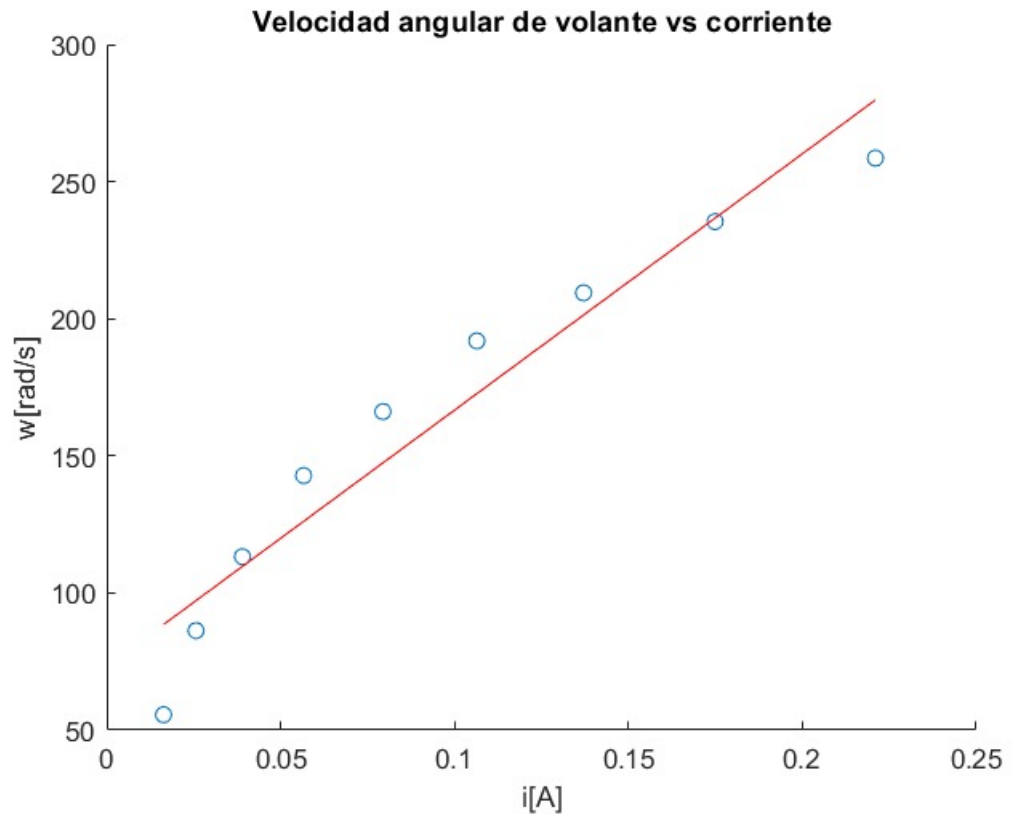


Figura 4.3: Gráfica de datos obtenidos en el ensayo de corriente con la recta que mejor se aproxima superpuesta.

Parámetro	Descripción
m_f	Masa del frame
l_g	Distancia desde el centro de masa del frame al eje de rotación
I_F	Momento de inercia del frame
b_F	Fricción en el eje de rotación

Nuevamente, todas las masas del sistema son medidas directamente, por lo cual no se requiere ningún tipo de ensayo para obtener las mismas. A su vez el parámetro l_g puede ser obtenido utilizando las herramientas de Fusion 360

Inercia del frame y rozamiento en el eje

Se le dedicará una sección al cálculo del momento de inercia del frame y rozamiento en el eje debido a la complejidad del mismo.

Si se da vuelta el frame y se lo deja oscilar con el motor trabado, el mismo se comporta de la misma forma que un péndulo simple.

Aplicando la segunda cardinal desde el eje del rulemán obtenemos el siguiente resultado

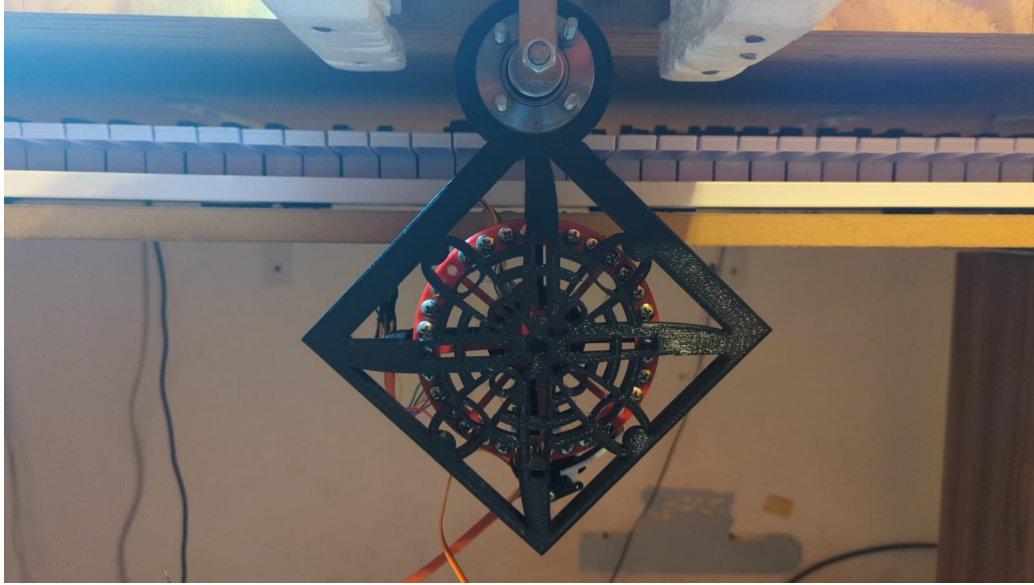


Figura 4.4: Foto del frame dado vuelta. Sobre este *setup* se realizó el ensayo para calcular el momento de inercia del frame y el rozamiento en el rulemán

$$I_{tot}\ddot{\theta} = k_{mgl}\sin(\theta) + b_F\dot{\theta} \quad (4.28)$$

Esta ecuación corresponde a la de un péndulo simple con rozamiento como era de esperarse y para resolverla y hallar los parámetros I_{tot} y b_F se aplicó el método de mínimos cuadrados no lineales.

Los mínimos cuadrados no lineales son una técnica de optimización que busca encontrar el conjunto de parámetros que minimizan la suma de las diferencias cuadradas entre los datos observados y un modelo matemático no lineal. A diferencia de los mínimos cuadrados lineales, en los que se busca ajustar una línea o un plano a los datos, los mínimos cuadrados no lineales se utilizan para ajustar modelos más complejos y no lineales a los datos.

El algoritmo generalmente utilizado para los mínimos cuadrados no lineales incluye los siguientes pasos:

1. Se selecciona un modelo matemático que se supone representa adecuadamente los datos. Este modelo incluye varios parámetros que son desconocidos.
2. Se proporciona un conjunto inicial de valores de parámetros.
3. Se calcula la suma de los cuadrados de las desviaciones entre las observaciones y las predicciones del modelo para el conjunto actual de parámetros.
4. Se actualizan los parámetros de manera iterativa para minimizar la suma de los cuadrados de las desviaciones.

4.2. Identificación de parámetros

5. Se repiten los pasos 3 y 4 hasta que se alcance un criterio de convergencia, como un cambio insignificante en los valores de los parámetros o un número máximo de iteraciones.

Para la resolución del problema de mínimos cuadrados no lineales se utilizaron las herramientas de MATLAB *Ode45* para generar las soluciones a la ecuación diferencial de segundo orden y *Lsqcurvefit* para la resolución de los mínimos cuadrados no lineales en sí.

Se dejó al sistema colgado hacia abajo y se lo hizo oscilar libre partiendo desde el reposo. Luego con *Lsqcurvefit* se obtuvo la curva que mejor se aproxima a los datos relevados en estas condiciones.

Es relevante destacar que los mínimos cuadrados no lineales, y por ende *Lsqcurvefit*, pueden encontrar soluciones óptimas locales en lugar de la solución óptima global, dependiendo del valor inicial de los parámetros. Por lo tanto, se iteró con diferentes valores iniciales de parámetros hasta encontrar la curva que mejor se aproxima a los datos relevados.

Los resultados obtenidos pueden verse en la figura 4.5

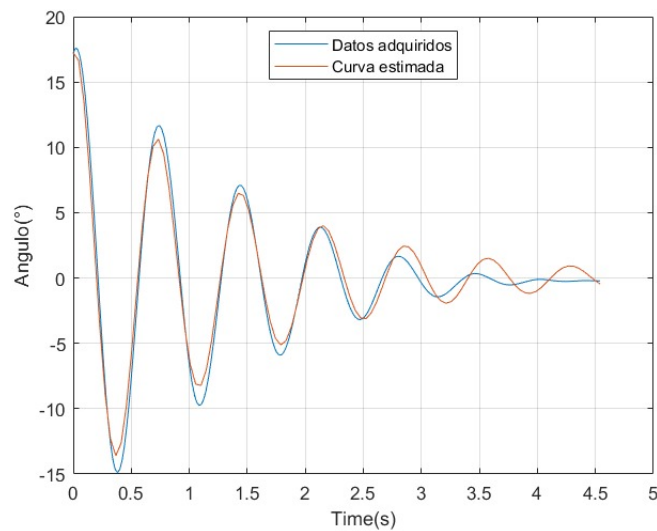


Figura 4.5: Gráfico que muestra el ángulo en función del tiempo del ensayo realizado contra el modelo teórico con los parámetros hallados.

Vemos como la curva estimada se aproxima muy bien a los datos relevados. Por lo tanto podemos considerar que el método fue exitoso.

Los valores de I_{tot} y b_f son:

$$I_{tot} = 6,7 \times 10^{-3} \text{kgm}^2, b_f = 7,2 \times 10^{-3} \text{Nms/rad}$$

De la ecuación 4.3 podemos despejar el valor de I_F

Capítulo 4. Modelado del frame y controlador

$$I_F = I_{tot} - l_v^2(m_v + m_m) \quad (4.29)$$

Obteniendo entonces $I_F = 2,2 \times 10^{-3} \text{kgm}^2$.

En resumen los parámetros del frame quedan como se muestra en la tabla 4.4

Parámetro	Valor
m_F	192g
l_g	9.42cm
I_F	$2,2 \times 10^{-3} \text{kgm}^2$
b_F	$8,8 \times 10^{-3} \text{Nm/s/rad}$

Tabla 4.4: Resumen de parámetros que caracterizan al volante de inercia.

4.3. Controlador LQR

¹En esta sección se presentará el controlador diseñado para el testeo de la funcionalidad del sistema frame (1D).

El controlador diseñado es un controlador LQR (Regulador Cuadrático Lineal). Esta es una técnica de control óptimo basada en la representación de variables de estado de un sistema que se utiliza para determinar una ley de control que minimiza una función de costo cuadrática.

Esta función de costo cuadrático es, típicamente, una suma ponderada del error cuadrático en los estados del sistema y el costo del controlador. El propósito de la función de costo cuadrático es cuantificar la calidad de una solución de control particular con respecto a los objetivos del sistema.

El control LQR tiene la estructura que se muestra en la figura 4.6, la misma se conoce como *Fullstate Feedback Controller* (Controlador realimentador de Estado Completo en español). Este tipo de controlador es particularmente útil cuando se desea controlar sistemas dinámicos complejos, donde todas las variables de estado son importantes para el comportamiento del sistema.

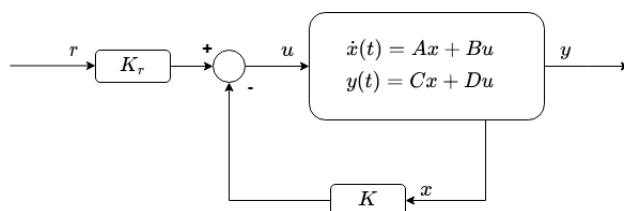


Figura 4.6: Diagrama que muestra la estructura de un controlador de estado completo.

¹El análisis teórico de esta sección y las ecuaciones planteadas se hacen en base a [25]

4.3. Controlador LQR

La estructura *Fullstate Feedback* no es una estructura única de los controladores LQR, existen otro tipo de técnicas de control como el *PolePlacement* que también se basan en la realimentación de todos los estados.

Lo que caracteriza al controlador LQR es la forma en la que se selecciona la matriz K . Se busca la matriz que optimice ciertas características del lazo cerrado importantes para el sistema, más específicamente qué tan bien se comporta el sistema y qué tanto esfuerzo requiere alcanzar esa performance. Esto es qué tan lejos se está del estado objetivo y qué tan intensa tiene que ser la magnitud de entrada para alcanzarlo.

Lo que se busca entonces es encontrar una solución K que minimice la función de costo J definida de la siguiente forma:

$$J = \int_0^{\infty} (x^T Q x + u^T R u) dt \quad (4.30)$$

Con u definido como

$$u = -Kx \quad (4.31)$$

Las matrices Q y R son las matrices de ponderación de estado y control respectivamente, la elección de estas matrices está sujeta al sistema en sí.

Generalmente, la matriz Q es una matriz diagonal en la cual los valores de su diagonal ponderan qué tan importante es que el error del respectivo estado sea cercano a 0. En el caso de R es similar, pero pondera la importancia del estado sobre la acción de control.

Para la resolución de este problema se aplicará la herramienta de MATLAB, *lqr* la cual teniendo el sistema correctamente modelado en variables de estado retorna la matriz K . Utilizando la matriz identidad para Q y R se obtuvo el siguiente resultado para K

$${}^1K = [23,55 \quad 2,69 \quad 0,094]$$

El resultado de K es coherente con lo esperado, la realimentación de mayor peso se da en θ ya que es la variable de estado más importante del sistema, luego de esta le sigue $\dot{\theta}$.

También es coherente el bajo valor de la realimentación de ω_v ya que el principal propósito de la realimentación de esta variable es que el disco no gire infinitamente. Esto es debido a que, por como se ve en la ecuación 4.7, la acción de control que ejerce el motor sobre el sistema está en los cambios de velocidades del volante y no en la velocidad en sí. Por esta razón es necesario que el volante tienda a frenarse ya que este no puede acelerar infinitamente y esta limitado principalmente por el rango de alimentación del sistema.

Los resultados prácticos de este controlador se verán en el capítulo 9.

¹Para el calculo de K se asume que la posición angular es medida en grados, la velocidad angular en grados/seg y la velocidad del volante en RPM

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Swing up

El concepto de swing up es utilizado en el contexto del control y dinámica de sistemas y refiere a una acción diseñada con el fin de trasladar un sistema desde un estado de reposo hacia un estado de movimiento o balanceo. En el caso del proyecto, consiste en llevar el cubo desde un estado de reposo hacia un equilibrio inestable de balanceo por sus propios medios, es decir sin la aplicación de fuerzas externas al sistema.

Como se menciona en el capítulo 4, $\tau_m + b_v\omega_v$ representa el torque que el sistema volante + motor ejerce sobre el frame. Tomando la ecuación 4.7 y dado que el frame se encuentra en reposo, viendo la ecuación 4.5 tenemos que

$$I_v\dot{\omega}_v = K_{mgl}\cdot\sin(\theta) \quad (5.1)$$

Con K_{mgl} definido en la ecuación 4.4. Para poder llevar a cabo la acción de *swing up* se debe lograr que:

$$I_v\dot{\omega}_v > K_{mgl}\cdot\sin(\theta) \quad (5.2)$$

Por lo tanto, se debe generar una aceleración en el volante de inercia lo suficientemente grande para lograr que el frame o cubo pueda salir del reposo en forma controlada. El objetivo es que esta acción levante al prototipo y lo deje en una posición lo suficientemente cercana a la de equilibrio para proceder a activar el control de balanceo. Esto se puede lograr generando un cambio abrupto en la velocidad del volante de dos formas distintas.

La primera de ellas implica detener de forma instantánea la rotación del volante, de forma que la energía disipada al momento del choque no destruya el sistema y el torque generado al frenar el volante levante el dispositivo. La segunda opción consiste en implementar un sistema de frenado por fricción que introduzca un torque interno al sistema que sea suficiente para lograr el objetivo.

Capítulo 5. Swing up

En definitiva, la desaceleración del volante de inercia se debe dar lo suficientemente rápido como para que el torque que genera este en el frame venza el torque generado por el peso del sistema.

A continuación se desarrolla el razonamiento presentado en el *paper* del proyecto *The Cubli* [2] en referencia a la implementación del *swing up* frenando el volante de inercia.

Si asumimos una colisión instantánea, perfectamente inelástica, es decir con un coeficiente nulo de restitución y por ende sin retroceso de la rueda, se puede calcular la velocidad angular ω_v necesaria para lograr el *swing up* de la siguiente forma. Aplicando conservación del momento angular al instante del impacto se tiene que

$$I_v \omega_v = (I_v + I_f + m_v l_v^2) \dot{\theta} \quad (5.3)$$

Dónde I_v es el momento de inercia del disco respecto al centro de este, I_f el momento de inercia del Frame respecto del vértice apoyado, l_v distancia del centro del volante al vértice apoyado y m_v la masa del volante. Esto quiere decir que el momento angular del volante = momento angular del frame + volante respecto al vértice.

Por lo visto en el capítulo 4 la inercia del volante puede ser calculada como:

$$I_v = \frac{1}{2} m_v (r_1^2 + r_2^2) \quad (5.4)$$

Donde r_1 es el diámetro externo del volante, r_2 el diámetro interno, m_v la masa del volante, y además se desprecia la masa de los parantes radiales del volante ([24]).

Por otro lado, planteando la conservación de la energía en el impacto ($\theta = \pi/4$)¹ hasta que el frame llega al ángulo de equilibrio ($\theta = 0$) tenemos que la energía cinética inicial del sistema frame + volante = aumento de la energía potencial gravitatoria de frame + volante.

$$\frac{1}{2} (I_v + I_f + m_v l_v^2) \dot{\theta}^2 = (m_f l_f + m_v l_v) g \left(1 - \frac{1}{\sqrt{2}}\right) \quad (5.5)$$

Eliminando $\dot{\theta}$ de 5.3 y 5.5 se obtiene que la velocidad de giro necesaria del volante es tal que

$$\omega_v^2 = (2 - \sqrt{2}) \frac{(I_v + I_f + m_v l_v^2)}{I_v^2} (m_f l_f + m_v l_v) g \quad (5.6)$$

¹En el capítulo 4 se aclara que la posición θ del frame en reposo es próxima a $\pi/4$, dada la geometría del frame cuadrado este el ángulo que se forma entre las aristas y la diagonal.

5.1. Freno de bloqueo

Esta velocidad es el punto de partida a partir del cual se deben realizar pruebas variando la velocidad del volante para encontrar la velocidad óptima de giro. Por lo visto en el *paper*, es probable que exista una diferencia entre el valor de ω_v calculado y el obtenido mediante iteración de pruebas. Esta desviación se debe a la imposibilidad de lograr un choque inelástico entre las piezas y la diferencia de valores reales vs teóricos que pueden llegar a tener los distintos parámetros. El frenado ideal consiste en alcanzar el punto de equilibrio con $\dot{\theta} = 0$, donde una vez alcanzada la posición de $\theta = 0$ se activa el control lineal de equilibrio.

5.1. Freno de bloqueo

El primer método mencionado fue utilizado en el proyecto The Cubli [2], se procede a describir su funcionamiento y características principales.

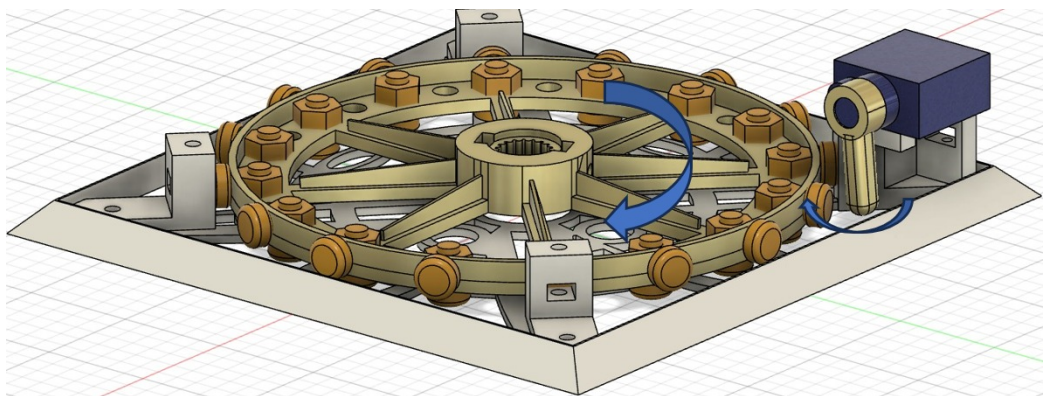


Figura 5.1: Sistema de frenado por bloqueo del volante. El servo motor desplaza su brazo hasta trabar el giro del volante al entrar en contacto con los tornillos de la cara externa del volante.

El sistema de frenado que se muestra en la figura 5.1 consiste de un servo motor con una vara anexada, que al desplazarse entra en contacto directo con una serie de tornillos colocados de forma perpendicular al volante de inercia. Al accionar el sistema el servo motor traba el desplazamiento de la rueda en un tiempo muy corto, generando el efecto deseado.

En primer lugar, esta acción de frenado libera una gran cantidad de energía en forma casi instantánea, que debe ser en gran parte disipada por el servo motor y su vara de frenado en sí. También se disipa energía en el punto que entra en contacto con el servo y su fijación al eje del motor de giro, además de en el soporte que acopla el servo motor al frame. Esto implica que los materiales de armado de estas piezas deben ser de gran durabilidad, de forma de extender la vida útil del equipo y del sistema.

Capítulo 5. Swing up

Si bien este método es directo, y que su desempeño y resultados son validados por los desarrolladores del proyecto The Cubli, su aplicación no es apta para el diseño implementado en este proyecto.

En primer lugar, las cualidades de los materiales que participan del sistema de frenado que seleccionamos no son las adecuadas para soportar golpes instantáneos de las características mencionadas. El material utilizado en las impresiones 3D del sistema (PLA) no tienen la densidad ni robustez suficiente para soportar un frenado en seco. Más aún considerando que el frenado en sí se daría entre la herramienta del servo motor y tornillos fijados al volante. Pensamos que estos tornillos destrozarían el volante al momento del impacto dada la violencia del choque. Además, un servo motor de las características dimensionales y de precio que buscamos es poco probable que pueda contar con engranajes metálicos de gran durabilidad, aptos para este tipo de sistemas de frenado. En definitiva el método de frenado fue descartado por la incidencia directa que puede tener en la vida útil del sistema en general.

5.2. Freno por rozamiento

El segundo método de frenado estudiado, que optamos por implementar en el sistema, consiste en el frenado del volante de inercia por medio de fricción por rozamiento. Para ello estudiamos los siguientes sistemas de frenado. Ambos consisten en desacelerar el volante con la suficiente rapidez para que el torque generado en esta acción de frenado le permita al frame levantarse ($\theta = 0$). En primer lugar se analizaron cuestiones constructivas de cada método para luego estudiar su viabilidad y parámetros con respecto al sistema.

El primer sistema de frenado por fricción consiste en una banda de goma de frenado, situada al rededor del volante de inercia y accionada por un servo motor. Al accionarse el frenado, el servo motor tensa la banda y esta entra en contacto con la pista exterior del volante. La fricción generada entre ambas partes genera la desaceleración buscada. Una posible implementación del sistema se puede ver en la figura 5.2.

Este sistema de frenado no genera el impacto violento del sistema visto anteriormente, lo cual condice con las condiciones de diseño que resultan de un primer análisis. Al ser un sistema de frenado por fricción, la misma genera desgaste en las piezas involucradas, es decir en la banda de goma y en la pista de frenado exterior al volante de inercia. La desventaja de este sistema consiste en la complejidad prevista en la instalación de la banda de frenado, ya que el material de la misma debe volver a su lugar de reposo una vez realizado el frenado para permitir el libre giro del volante. Para subsanar este problema se puede colocar una banda metálica de mayor rigidez en la cara externa de la banda, que permita repetir el frenado sin deformarse volviendo a su forma y posición originales luego de cada frenado. La dificultad de lograr el correcto funcionamiento de este método hizo que el mismo

5.3. Sistema de freno implementado - Zapata de frenado

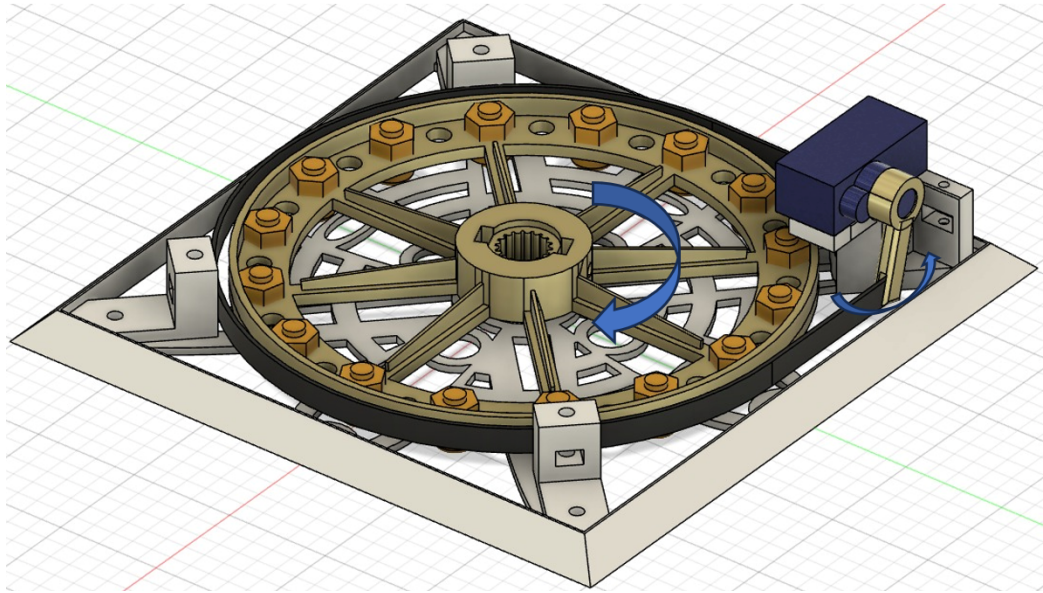


Figura 5.2: Sistema de frenado por tensado de banda. El servo desplaza su brazo y tensa la banda para frenar el volante.

fuera descartado.

5.3. Sistema de freno implementado - Zapata de frenado

El sistema implementado consiste en una zapata de frenado que al ser accionada por el servo motor entra en contacto con la pista externa del volante de inercia, desacelerando la misma. Las variantes y parámetros implicados en este sistema a diseñar son las siguientes:

- Servo motor
- Re diseño del frame
- Re diseño del volante
- Diseño de zapata
- Elementos friccionantes

Una vez analizados estos aspectos, hubo que calcular y calibrar la velocidad del volante y la fuerza aplicada por el servo al frenado.

Con respecto a la elección del motor de frenado, en vista de que las posiciones de descanso y de activación del freno son requieren de una alta precisión, la elección natural es un servo motor. El mismo debe ser de dimensiones lo más reducidas posibles, ser liviano, y contar con la fuerza y velocidad suficiente para

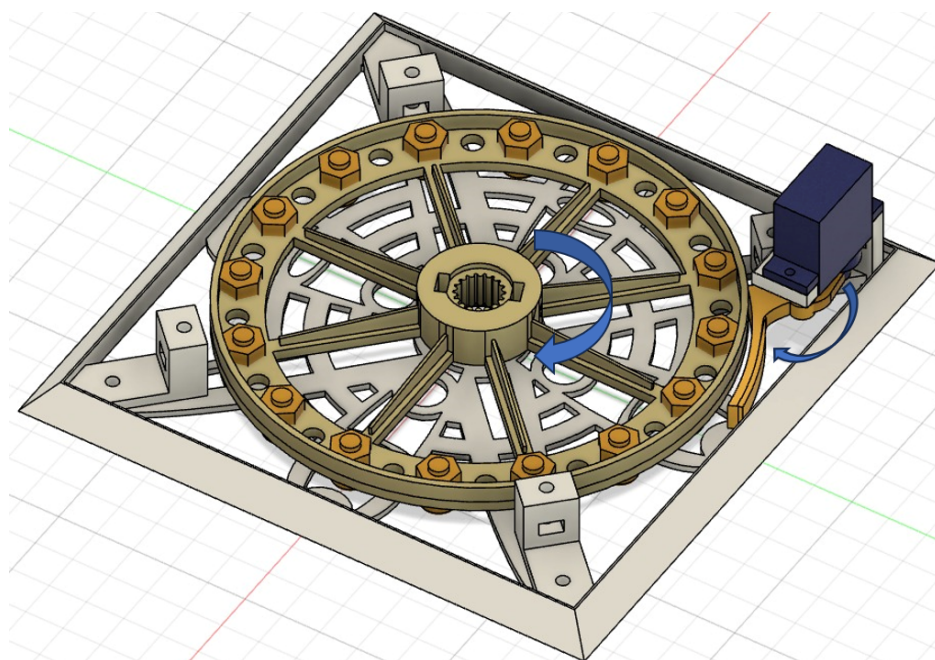


Figura 5.3: Sistema de frenado por zapata implementado. El servo desplaza la zapata y esta frena el volante.

lograr el objetivo. El proceso de elección del servo motor se puede ver en la sección 3.2.4 del capítulo 3. Una vez seleccionado el servo motor **MG90S** se tomaron sus dimensiones para realizar el diseño del mismo en 3D y adaptar el frame para su colocación.

Dada la fragilidad del material plástico PLA de impresión, se buscó implementar un soporte para el motor que sea robusto y macizo en lo posible, su diseño se puede ver en la figura 5.4.

Una vez solucionada la colocación y ubicación del servo, se colocó la pista externa de frenado al volante de inercia (ver figura 5.6). Luego se diseñó una zapata de frenado en base a la pista de frenado del volante (figura 5.5).

Debido al poco rozamiento entre las dos superficies se toma la decisión de aplicar elementos que añadan fricción al frenado. Para esto se coloca cinta doble faz gruesa en la zapata y cinta aisladora en la pista de frenado. Ambos elementos se irán desgastando a medida que se hace uso del sistema de frenado, como en cualquier freno de este tipo. Se deberá evaluar en función del rendimiento y éxito del swing up, el estado de las cintas de frenado y si hubiera que remplazar las mismas.

Una vez implementado el sistema y sus componentes, se debe evaluar las posiciones que debe adoptar el servo motor cuándo el volante está girando a cierta velocidad y la posición cuando se quiere frenar el volante. Estas posiciones se definen como ángulos del servo tal que

5.3. Sistema de freno implementado - Zapata de frenado

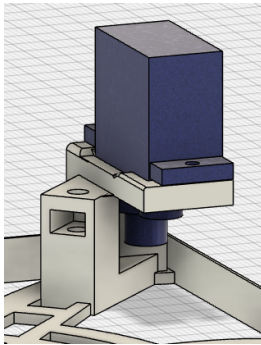


Figura 5.4: Detalle de soporte de servo motor de frenado.

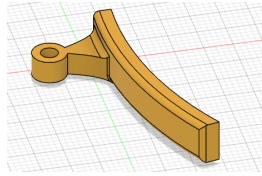


Figura 5.5: Zapata de frenado.

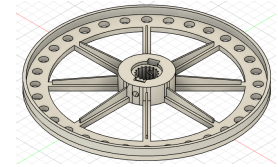


Figura 5.6: Volante de inercia con pista exterior de frenado.

- α_r es el ángulo de reposo del servo motor. La zapata de frenado se encuentra en esta posición en todo momento que el volante esté girando libremente.
- α_f es el ángulo en el que se posiciona el servo motor al momento de frenar el volante. Cuanto mayor sea este ángulo, mayor será la fuerza aplicada por el servo al volante.

A su vez, resta definir la velocidad de giro del volante al momento del frenado (δ_{su}) junto con el tiempo de asentado de esta velocidad y el tiempo en el que el freno está activo (T_f), es decir en el que el servo motor se encuentra en α_f .

Los ensayos realizados y resultados obtenidos del sistema de frenado pueden verse en el capítulo 9, sección 9.1.5.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Power Management Circuit

6.1. Descripción general

En este capítulo se describirán los criterios elegidos para la elección de los componentes pertenecientes al Circuito de Gestión Energética (*Power Management Circuit*, PMC).

El Cubinja tiene una gran cantidad de componentes que se alimentan a distintos niveles de tensión. Además la tarea de mover motores siempre presenta un gran desafío a lo que consumo refiere.

Por esta razón se comenzará haciendo un análisis de consumo de todas las partes del sistema para dimensionar la capacidad de corriente que cada fuente de tensión debe tener.

Se propondrán entonces dos diferentes topologías para el PMC y se presentará la elección de la más adecuada en función de la complejidad del circuito y de la cantidad y disponibilidad de los componentes del mismo.

Si bien se desarrolla la elección de batería en una sección de este capítulo se tomará como primer requerimiento del PMC que sea alimentado desde una batería LiPo. Se toma esta decisión debido a la gran disponibilidad de baterías de este tipo junto con sus cargadores en plaza y sus altas tasas de descarga, permitiendo entregar grandes cantidades de corriente para capacidades no tan altas, lo que significa baterías de tamaño reducido.

6.2. Power Budget

La primera sección de este capítulo será la presentación del Power Budget (presupuesto de potencia en español). El Power Budget es un diagrama que se utiliza para detallar y calcular la distribución de energía en un sistema embebido. Este diagrama presenta información sobre cuánta energía se espera que consuma cada componente individual del sistema, desde los sensores y actuadores hasta el microcontrolador y el resto de periféricos.

Los principales dos requerimientos de cualquier PMC son primeramente los niveles de voltaje que el mismo debe proveer y la corriente que se le consumirá a

Capítulo 6. Power Management Circuit

cada fuente de tensión

Para el diseño de estos dos requerimientos se recurren a las hojas de datos de cada dispositivo y se verifica a qué nivel de tensión debe alimentarse cada uno de ellos y cuál es el máximo consumo a este nivel. El diagrama de la figura 6.1 resume la información recolectada.

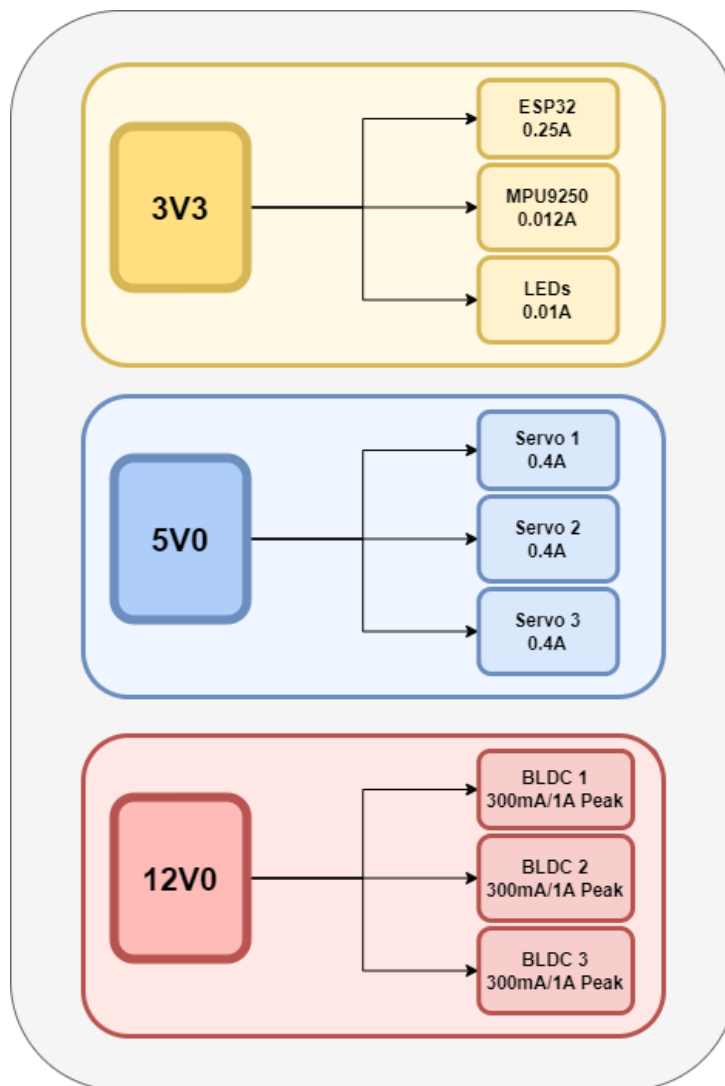


Figura 6.1: Diagrama que muestra los tres niveles de tensión requeridos por el sistema junto con los correspondientes componentes alimentados y su respectivo consumo.

La fuente de tensión de 3.3V deberá alimentar al microcontrolador, a la IMU y a los LEDs. El consumo total de estos 3 periféricos suma un total de 0,272A y esta es la corriente que la fuente de 3.3V debe ser capaz de proveer como mínimo.

El único sistema que requiere una alimentación de 5V son los servomotores. A lo que consumo respecta cada uno de estos consume 0,4A, sin embargo, el sistema diseñado no requerirá del uso simultáneo de los servomotores en ningún caso, por

lo tanto, la fuente de 5V debe poder entregar como mínimo 0.4A, esto es activar un solo servomotor a la vez

Finalmente, la fuente de 12V debe ser capaz de alimentar a los tres BLDC´s del sistema. De la hoja de datos se obtuvo que la corriente de arranque del motor a 12V puede llegar a ser de hasta 1A. El consumo estático se midió directamente del sistema, ya que el mismo depende de la carga que este tenga que mover, se montó el volante de inercia en el eje del motor y se lo hizo girar a su máxima velocidad obteniendo un consumo estático de 300mA.

En el caso de los motores de continua es esperable que haya casos en los que todos los motores tengan que estar girando en simultáneo, por lo tanto, la fuente de 12V deberá ser capaz de soportar picos de hasta 3A y consumos estáticos de 0,9A.

6.2.1. Alimentación de una sola celda

La primera topología de PMC propuesta busca colocar una batería con un peso y tamaño reducido. Por lo tanto, en este primer circuito la fuente principal de alimentación será una batería LiPo de una sola celda. Puede verse dicha topología en la figura 6.2.

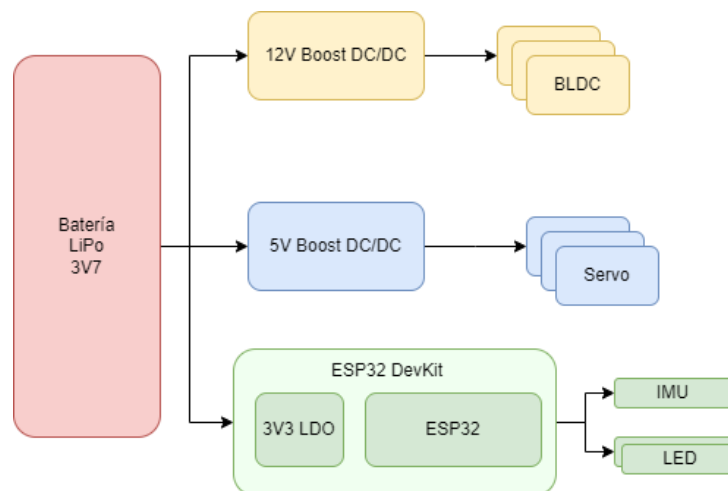


Figura 6.2: Power Budget sin consumos de corriente para topología alimentada por una sola celda de 3V7.

La primera observación a hacer es que no es necesario agregar un regulador de 3.3V al sistema debido a que el propio *development kit* (DK) (*datasheet* en [26]) del ESP32 ya cuenta con un regulador lineal (LDO) integrado.

Dicho LDO es un AMS1117. Este regulador no solo alimenta al ESP32 internamente sino que está disponible su salida en los pines de 3.3V del DK permitiendo alimentar otros sistemas desde el mismo.

El rango de entrada de este LDO es hasta 15V, esto significa que se podría conectar el DK del ESP32 directo desde la batería o en serie con cualquiera de

Capítulo 6. Power Management Circuit

los otros dos reguladores; sin embargo, se recuerda que la eficiencia de un LDO se puede calcular como en la fórmula 6.1 [27].

$$LDO_{eff} = \frac{V_{out}I_{out}}{V_{in}(I_{out} + I_q)} \quad (6.1)$$

En donde V_{out} e I_{out} son el voltaje y corriente de salida respectivamente, V_{in} es el voltaje de entrada del regulador e I_q es la corriente de *quiescent* o de consumo estático del regulador.

Vemos que la eficiencia es inversamente proporcional al voltaje de entrada, por lo tanto, cuanto mayor sea el voltaje de entrada más ineficiente será el sistema, aumentando así las pérdidas innecesariamente

La segunda observación es que esta topología necesita un total de 2 reguladores switcheados de tipo Boost que eleven el voltaje a los niveles deseados. Los reguladores switcheados tienen el inconveniente de ser circuitos complejos, lo que significa mayor tamaño y posibilidades de fallas. Además de esto introducen ruido de alta frecuencia en el sistema, perjudicando así la lectura de señales analógicas, entre otras cosas.

Esto no es considerado un inconveniente en el sistema, ya que el Cubinja cuenta con el espacio necesario para estos dos circuitos y la única señal analógica a medir es el voltaje de la batería del sistema, haciendo que este ruido sea despreciable para la medida.

El único inconveniente teórico de este sistema es que el uso de elevadores de tensión aumenta el consumo de corriente directa desde la batería. Tomando como ejemplo los 3A consumidos por el regulador de 12V y tomando una transferencia de energía sin pérdidas podemos plantear lo siguiente

$$P_{in} = P_{out} \quad (6.2)$$

Por lo tanto,

$$V_{in}I_{in} = V_{out}I_{out} \quad (6.3)$$

Y despejando I_{in}

$$I_{in} = \frac{V_{out}I_{out}}{V_{in}} \quad (6.4)$$

Si tomamos como voltaje de entrada 3.7V y de salida 12V a una corriente de 3A a la entrada se tendría 9.7A del lado de la batería, lo cual es un valor muy grande incluso para una batería LiPo. Esto provoca que la batería a introducir en el sistema deba tener una capacidad considerable para poder suministrar esta corriente.

Si bien esto no es ideal, la alta tasa de descarga de las baterías LiPo hace que puedan manejarse estos niveles de corrientes con baterías de un tamaño compacto.

Aunque en teoría esta topología podría funcionar, no se pudo conseguir un regulador boost capaz de soportar picos de hasta 3A. Solo se pudieron conseguir reguladores capaces de entregar picos de hasta 2A como máximo, haciendo que se precise un regulador por motor, como se muestra en la figura 6.3.

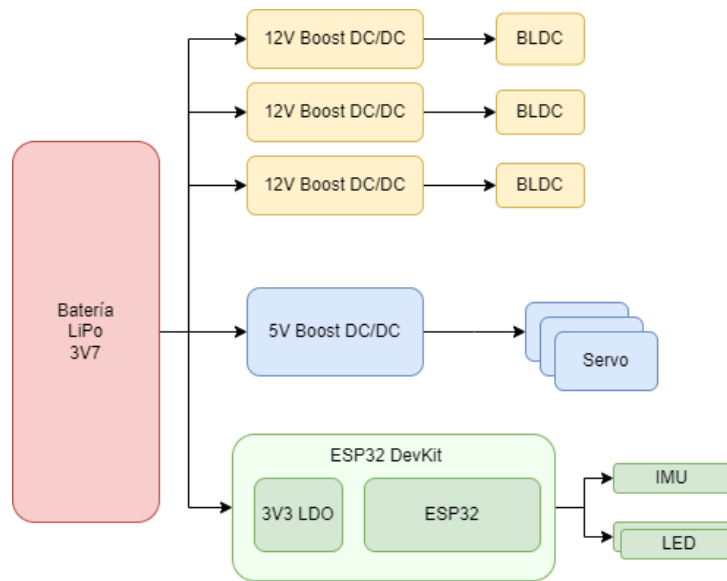


Figura 6.3: Diagrama de bloques de PMC para topología de una sola celda utilizando un regulador boost para cada motor

Por este motivo esta topología fue descartada, ya que la conexión y posicionado de 4 reguladores *switcheados* en el sistema presentaba un exceso de dificultad frente a otras posibilidades de PMC.

6.2.2. Alimentación de tres celdas

La topología de tres celdas fue la elegida para el PMC del Cubinja. Esta topología busca eliminar la necesidad de un regulador *switcheados* del tipo *boost* para alimentar los motores colocando una batería de hasta tres celdas. Estas baterías tienen un voltaje nominal de 11.1V y llegan hasta 12.6V cuando están cargadas totalmente. El Power Budget de este PMC se puede ver en la figura 6.4.

Este circuito solo requiere el uso de un regulador *switcheados* del tipo *buck* para generar los 5V de alimentación del servo. Observar que, por la justificación de eficiencia que se dio en la sección anterior, se conecta el LDO del DK del ESP32 en serie con el regulador *switcheados* de 5V y no directamente desde la batería.

Esta topología presenta el inconveniente de la necesidad de una batería de tres celdas, estas tienen un tamaño considerable haciendo que el diseño mecánico sea más complejo. El parámetro más restrictivo en la elección de una batería para el sistema es su masa. A pesar de ello se logró encontrar en plaza baterías con una masa muy pequeña en comparación con el resto del sistema.

El otro punto negativo de esta topología es que los motores no estarán alimentados por una fuente regulada, haciendo que la tensión máxima de control no sea constante.

Si bien esto último es un problema, se considerará que una caída de hasta 1V del nivel nominal de la batería no es perjudicial para el control y manejo de los

Capítulo 6. Power Management Circuit

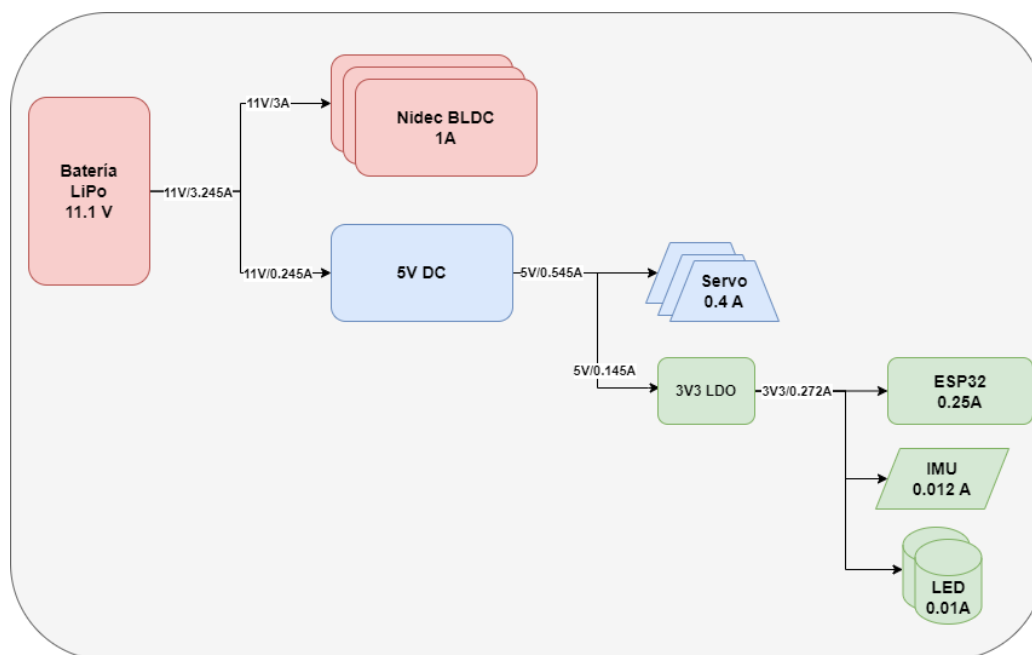


Figura 6.4: Power budget del Cubinja, todos los pasajes de corriente aguas arriba se asumen con 100% de eficiencia

motores. Por lo tanto, se monitoreará este valor y se informará al usuario cuando el nivel de la batería haya salido del rango permitido.

Elegida la topología del PMC se pasa a la elección de componentes del mismo.

6.3. Elección de batería

El primer componente a elegir es la batería del sistema. Como se explicó en la introducción al capítulo se optó por una batería de tres celdas del tipo LiPo.

En la sección anterior se mencionó que el voltaje nominal de estas baterías es de 11.1V, pero alcanzan un voltaje de hasta 12.6V cuando están completamente cargadas, esto le da al controlador un rango de funcionamiento suficientemente bueno para cumplir su función.

El criterio de elección de la batería fue que la misma tuviera una masa reducida. Esto se traduce a una capacidad baja pero suficiente como para entregar la corriente necesaria y tener un tiempo de uso prolongado.

Además, debido a que el circuito de carga de la batería no se incluye en el PMC del sistema, la misma debe contar con un sistema de carga externo al Cubinja.

Con estos dos objetivos presentes se eligió la batería de tres celdas de *HOBBYHUB*. Esta batería tiene una masa de 183g lo cual es un valor aceptable para el sistema.

La batería de *HOBBYHUB* es una batería de 2200mAh y con una tasa de descarga de 40C, esto es de hasta 40 veces su capacidad, logrando entregar corriente de hasta 88A.

De la figura 6.1 suponiendo que el sistema se encuentra con todos sus motores

6.4. Elección de reguladores



Figura 6.5: Foto de la batería seleccionada para el Cubinja.

girando en simultáneo a máxima velocidad, podemos estimar el consumo estático del Cubinja con 1A. Podemos calcular un estimativo del tiempo de duración de la batería como se muestra en la ecuación 6.5

$$T_{Duracion} = \frac{Capacidad}{Consumo} \quad (6.5)$$

Esto termina dando un tiempo de aproximadamente 2.2Hs de uso antes de que la batería se agote por completo, considerando la batería totalmente descargada cuando el voltaje en vacío de la misma es de 9V. Dado que se busca lograr una autonomía de 30 minutos, este tiempo de funcionamiento se considera satisfactorio

6.4. Elección de reguladores

El siguiente componente a seleccionar es el regulador de 5V. Como se mencionó anteriormente este es el único regulador a seleccionar debido a que el DK del ESP32 ya tiene adentro suyo un regulador lineal de 3,3V capaz de alimentar al resto de periféricos que requieren esta tensión.

Como se describe en la figura 6.4, el principal requerimiento para este regulador es que sea capaz de suministrar hasta 0.545A. Además el mismo debe soportar un rango de entrada de hasta 12.6V.

Con estos dos objetivos en mente se seleccionó el regulador LM2596 (*datasheet* en [28]) de Texas Instruments. Este regulador es switchado de tipo *buck* (*step-down*) con un rango de entrada de hasta 40V y una capacidad de corriente de hasta 3A cumpliendo con los dos requisitos principales de este sistema.



Figura 6.6: Foto del DK del regulador LM2596 (hoja de datos en F).

Como se muestra en la figura 6.6 es un regulador con una baja cantidad de componentes extra, lo que significa un DK de tamaño reducido.

Por estas razones y por su gran disponibilidad en plaza, se utilizará el DK del LM2596 para suministrar los 5V al sistema.

6.5. Construcción y conexión

Se pueden ver los componentes que forman parte del diseño del PMC en la figura 6.7. En azul vemos el DC-DC LM2596 y en negro el ESP32 DK.

En la figura 6.8 se ven los puntos notables del circuito en donde se entregan los distintos niveles de tensión. La placa se conecta directamente a la batería LiPo y por medio de un switch de encendido se energiza a los motores y resto de los componentes. Desde el nivel de tensión de la batería se alimenta el DC-DC que entrega los 5V para a los servo motores. Por otro lado, en el ESP32 DK se ven los pines de entrada y salida de interés. En modo autónomo, es decir cuando el cubo se encuentra alimentado únicamente por la batería, el microcontrolador es alimentado por el DC-DC por su pin de 5V. Se cuenta con un pin denominado *5VOFF* en el diagrama con el cual se desconecta la alimentación de 5V del ESP32, para el caso en el que la placa este alimentada vía USB. El resto de los conexionados de la placa se pueden ver en el capítulo 7.

6.5. Construcción y conexión

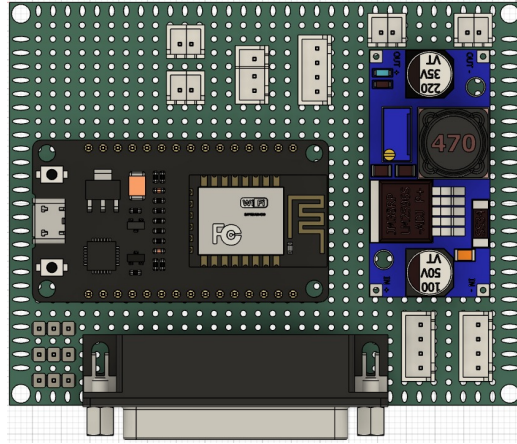


Figura 6.7: Componentes.

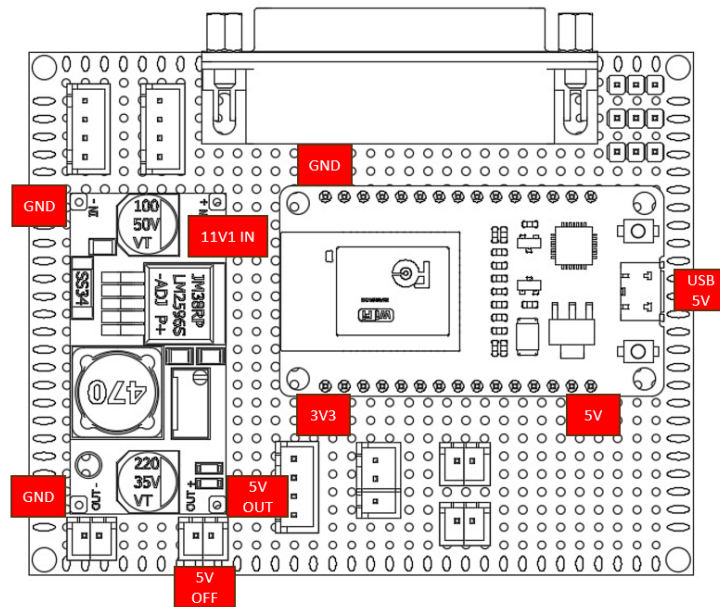


Figura 6.8: Niveles de tensión del circuito.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Construcción

7.1. Descripción general

La totalidad de las piezas fueron diseñadas en el software de diseño Fusion 360 e impresas en 3D en PLA con una impresora ENDER, tomando como referencia de diseño algunos proyectos anteriores [2] [11]. Las dimensiones generales de los componentes del cubo fueron tomadas de proyectos anterior, en particular del proyecto The Cubli. Se estudiaron proyectos que surgen de adaptar el Cubli para imprimir las piezas en 3D para verificar y validar el comportamiento del material, ya que en el proyecto original la mayoría de las piezas fueron fabricadas en aluminio y no en plástico.

El proceso de diseño consistió en validar el armado y ensamblado de piezas en el visor 3D de Fusion 360 previo a su impresión. Componentes comprados como el BLDC, servo motor de frenado, IMU, ESP32, conversor DC-DC, tornillos, tuercas y rulemanes fueron relevados con calibre, diseñados y posicionados en el software. Una vez validado el diseño se generaron los archivos en formato .gcode necesarios para la impresora 3D con el software CURA. Parte del iterado del proceso de impresión consistió en la experimentación con distintas densidades de impresión, modificando así la rigidez y peso de las piezas. Se recuerda que la densidad del PLA es de $1,24g/cm^3$.

7.2. Volante de inercia

Esta pieza se encuentra acoplada al eje del BLDC por medio de una polea dentada fijada al mismo y una estría diseñada a medida. A su vez cuenta con dos tornillos prisioneros con tuerca que impiden el deslizamiento en sentido perpendicular de la pieza. En un principio se optó por mecanizar una muesca en el eje del motor y fijar un prisionero en el volante que se clave en la muesca. Este método presentó graves problemas de deslizamiento al revolucionar el motor, por tanto fue sustituido por el antes mencionado.

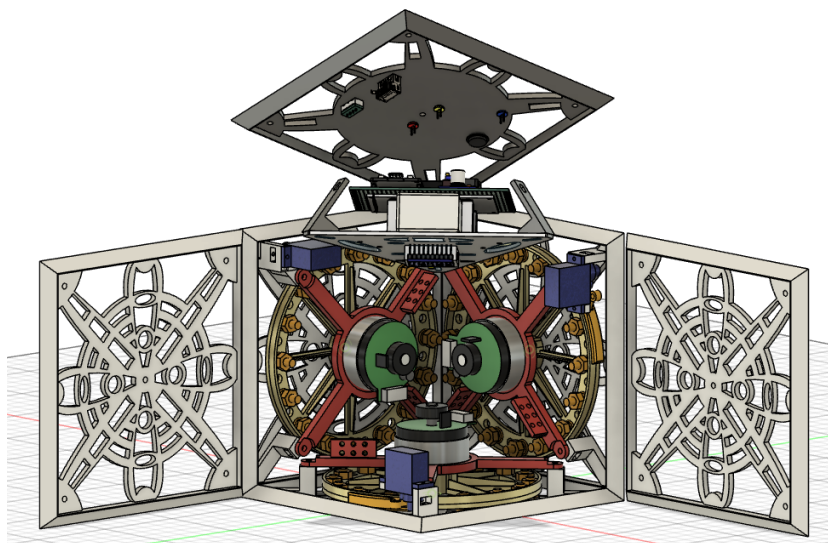


Figura 7.1: Vista completo de componentes internos del cubo.

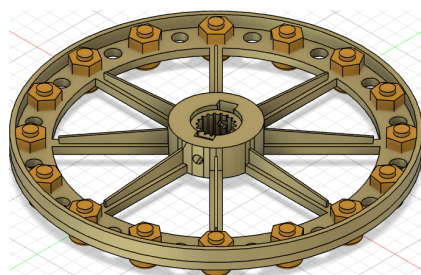


Figura 7.2: Volante de inercia, con peso agregado.

Parámetro de volante	Medida
Diámetro exterior	61.20 mm
Diámetro interior	48.66 mm
Altura pista	7.30 mm

El volante cuenta con orificios de 5.30mm de diámetro dónde se colocan tornillos de 5x8mm que son fijados con tuercas. De esta forma se le agrega peso al volante. Mientras mayor sea el peso mayor será la inercia rotacional del volante, por tanto mayor deberá ser el torque aplicado por el motor para cambiar la aceleración del volante y mayor será la reacción aplicada al frame para estabilizarlo en su posición de equilibrio.

Por último se agregó una pista de frenado en la cara externa del volante de 1.70mm de ancho para que la zapata de frenado entre en contacto con esta pista a la hora de desencadenar el swing up del frame/cubo. Esta pista a su vez fue

recubierta con cinta aisladora para aumentar su fricción.

7.3. Unión de motor a frame

Se diseña un soporte para motor en forma de cruz. Sus cuatro puntas se apoyan en cuatro descansos colocados en el frame correspondiente y se atornilla a los mismos con tornillos de 3x10mm con tuerca. El motor se atornilla a la cruz con tornillos 3x4mm. A lo largo del proyecto se ven dos prototipos de cruces. El primero de ellos tiene el agregado de un soporte para IMU, ya que en la versión 1D que consiste en solamente un frame la IMU del sistema se coloca en este punto. La segunda versión, utilizada en el cubo, no tiene soporte para IMU y sí tiene soportes para poder sujetar con precintos los cables del motor y componentes internos.

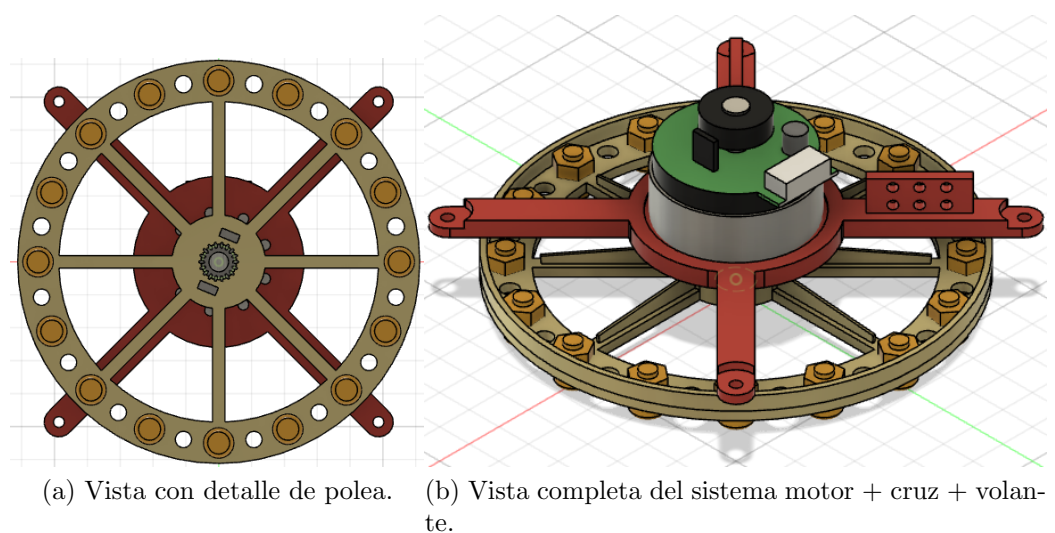


Figura 7.3: Ensamblado de volante de inercia.

7.4. Frame

El frame es un cuadrado de 150x150x7mm. En su versión más simplificada sirve solamente de soporte estructural del cubo (7.4). Para poder sostener el sistema de motor más volante se colocan cuatro descansos para acoplar la cruz del motor. A su vez se tiene también un soporte robusto para el servo motor (7.5). El frame diseñado para la versión 1D del sistema se puede ver en la figura 4.1. El tercer formato del frame presenta los siguientes detalles:

Capítulo 7. Construcción

- Conector USB A hembra: Permite la comunicación con el microprocesador ESP32.
- Conector macho para carga de batería lipo.
- LEDs rojo y amarillo de encendido
- LED azul indicador de conexión bluetooth pendiente y de batería baja
- Botón de encendido del sistema.

Un detalle no menor es que los bordes de los frames son a 45° para poder acoplarlos entre sí y que formen un ángulo recto. La unión entre dos caras contiguas se hace en dos puntos con cinta aisladora y es reforzada en las esquina por medio de esquineros atornillados (ver figura 7.7).

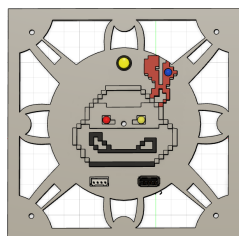


Figura 7.4: Frame con conexiones y leds.

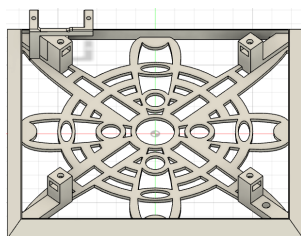


Figura 7.5: Frame con motor y volante.

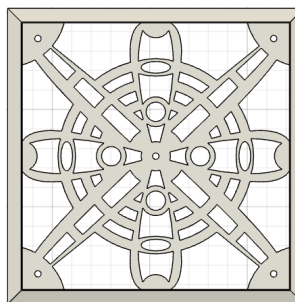


Figura 7.6: Frame simple.

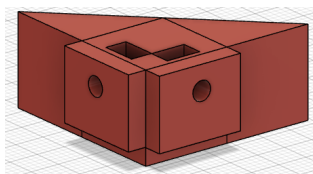
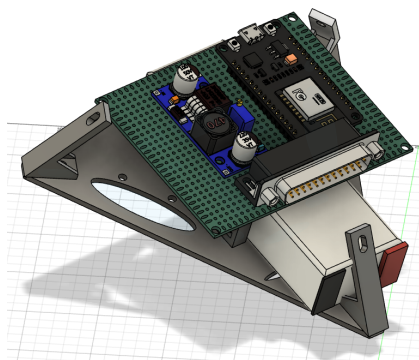


Figura 7.7: Esquinero para refuerzo de unión de caras.

7.5. Soporte de componentes internos

En la figura 7.8a se puede ver el conjunto de componentes que son interiores al cubo. Lo visto consiste en una plataforma en forma de triángulo con tres soportes que van atornillados con tornillos 3x4mm y tuerca a las caras (frames) simples del cubo. Este soporte sostiene la batería de 11.1V con dos topes regulables. También hace de base para la colocación del circuito de control. A su vez, como se puede observar en la vista (b) de la figura 7.8a, la IMU se coloca en la parte posterior del soporte triangular.



(a) Vista superior.

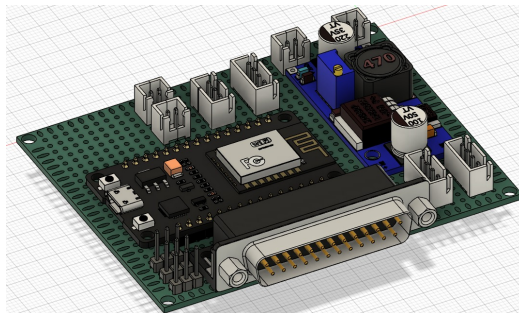


(b) Vista inferior.

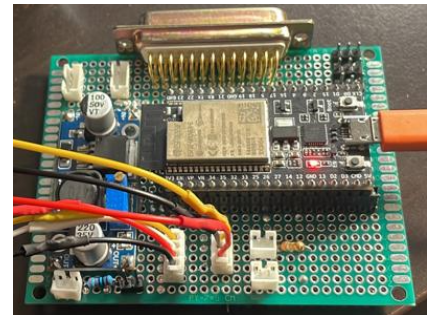
Figura 7.8: Soporte de batería, placa e IMU.

7.6. Circuito

En las figuras 7.9a y 7.9b se pueden ver el diseño 3D de la placa de control y el resultado final una vez soldados y conectados todos los componentes. Los elementos a destacar son:



(a) Vista de modelo 3D.



(b) Foto del circuito completo.

Figura 7.9: Vistas del circuito.

- Conector DB25: Contiene las conexiones de los BLDC (A, B y C). Fue colocado para facilitar el desarmado del cubo ya que junto con los cables de los servos son el único cableado que une la mitad del cubo que tiene los frames con motor a la mitad del cubo que tiene el comando del sistema.
- Conectores varios, se pueden ver en detalle en la figura 7.10.
- En azul se ve el DC-DC LM2596 que provee al circuito con 5V para la alimentación del ESP32 y los servos a partir de la entrada de 11V1 de la batería.

Capítulo 7. Construcción

- El ESP32 DK cuenta con un conector USB para comunicarse y poder cargar programas nuevos en el sistema operativo.

Como fue mencionado anteriormente en la figura 7.10 se pueden ver los puntos de conexión de la placa. El cableado de los motores con el conector DB25 se puede ver en el cuadro de la figura 7.11. Para mayor comprensión del código de colores del cableado de los motores se puede usar la figura 7.12 de referencia.

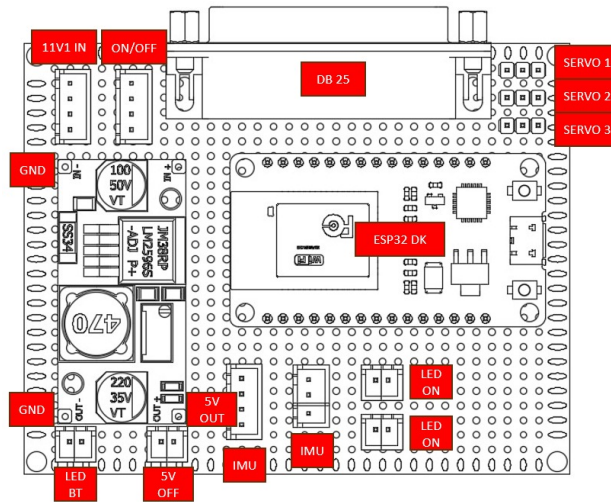


Figura 7.10: Layout de borneras y placas.

	1	2	3	4	5	6	7	8	9	10	11	12	13
arriba													
motor_color		M3_verde	M2_verde	M1_verde	M1_naranja	M1_amarillo	M1_negro	M1_azul	M1_rojo	M2_rojo	M2_amarillo	M3_negro	M3_naranja
ESP32		PIN 27	PIN 15	PIN 5	PIN 19	PIN 18	PIN 35	GND	PIN 34	PIN 39	PIN 0	PIN 25	PIN 12
abajo	14	15	16	17	18	19	20	21	22	23	24	25	shield
motor_color	M3_blanco	M3_rojo	M2_violeta	M3_amarillo	M2_negro	M2_azul	M1_blanco	M1_violeta	M3_violeta	M3_azul	M2_naranja	M2_blanco	
ESP32	+3V3	PIN 26	+12V	PIN 14	PIN 32	GND	+3V3	+12V	+12V	GND	PIN 2	+3V3	

Figura 7.11: Conexión interna de motores A, B y C con conector DB25.

+12v	violeta
GND	azul
BRAKE	verde
PWM	naranja
CW/CCW	amarillo
+3V3	blanco
CANAL A	rojo
CANAL B	negro

Figura 7.12: Diagrama de colores y funcionalidades de motores BLDC.

Capítulo 8

Diseño del software embebido

8.1. Descripción general

El presente capítulo detalla la arquitectura de firmware diseñada para el Cubinja, abarcando primeramente el uso de freeRTOS y las herramientas que este provee. Luego se describe un panorama más general de la arquitectura, siguiendo por las tareas que hacen al sistema.

Finalmente se presentan las funcionalidades que el sistema presenta al usuario via *Bluetooth* junto con los protocolos de comunicación implementados y la aplicación en *Python* que a este se le provee

8.2. FreeRTOS

FreeRTOS es un sistema operativo en tiempo real de código abierto ampliamente utilizado en el desarrollo de firmware embebido. Una de sus características principales es su sistema de gestión de tareas basado en prioridades, junto con un potente scheduler, lo cual ofrece una solución eficiente y versátil para la gestión de tareas concurrentes en sistemas embebidos. Esto lo convierte en una opción popular en aplicaciones de Internet de las cosas (IoT) y otros sistemas embebidos.

FreeRTOS ofrece una serie de características que son beneficiosas para el desarrollo de firmware. Algunas de estas características incluyen

- **Scheduler:** FreeRTOS proporciona un componente llamado scheduler. Este scheduler decide cuándo y durante cuánto tiempo se ejecutan las diferentes tareas en base a un sistema de prioridades, administrando de esta forma los recursos de la CPU.
- **Semáforos:** FreeRTOS proporciona semáforos binarios y de conteo, que son mecanismos de sincronización que permiten la comunicación y la coordinación entre tareas. Los semáforos se utilizan para controlar el acceso a recursos compartidos, prevenir condiciones de carrera y garantizar la correcta ejecución de tareas en un entorno multitarea.

Capítulo 8. Diseño del software embebido

- **Colas:** FreeRTOS ofrece colas que permiten la transferencia de datos entre tareas de forma eficiente y segura. Las colas son utilizadas para la comunicación y la transferencia de información entre tareas, lo que facilita la implementación de sistemas distribuidos y la coordinación de tareas en tiempo real.
- **Notificaciones:** FreeRTOS ofrece mecanismos de notificación que permiten a las tareas comunicarse y sincronizarse de manera eficiente. Las notificaciones son utilizadas para señalar eventos, despertar tareas y coordinar la ejecución de tareas en un entorno multitarea.
- **Active objects:** Los *active objects* en FreeRTOS son una combinación de una tarea y una cola, lo cual permite implementar patrones de diseño habituales de manera eficiente y organizada. Esto permite la implementación de sistemas distribuidos y la prevención de problemas de datos compartidos, asegurando una ejecución confiable de las tareas en un entorno multitarea.

FreeRTOS es una excelente elección para utilizar con ESP32, un microcontrolador ampliamente utilizado en aplicaciones IoT. La combinación de FreeRTOS con ESP32 ofrece un entorno de desarrollo robusto y eficiente para implementar firmware embebido en sistemas IoT, permitiendo una gestión avanzada de tareas, sincronización de eventos y optimización de recursos en tiempo real.

Aunque FreeRTOS ofrece numerosos beneficios, también es importante tener en cuenta algunas posibles desventajas, como la curva de aprendizaje inicial, ya que puede requerir una comprensión detallada de la programación multitarea y el uso de sus características avanzadas. Además, la gestión de memoria y recursos puede ser más compleja en un entorno multitarea, lo que demanda un diseño cuidadoso y una planificación adecuada para evitar problemas de rendimiento y errores de programación.

En resumen, FreeRTOS provee una gran cantidad de herramientas que permitirá al firmware del Cubinja mantenerse modular y organizando, con todos sus procesos bien definidos y ordenados, permitiendo de esta forma un fácil entendimiento del mismo con conocimientos básicos de RTOS y por lo tanto mantenible y escalable.

8.3. Arquitectura

Por las ventajas que se mencionaron en la sección anterior la arquitectura diseñada se basa fuertemente en el uso de *Active objects*, se crea un *active object* diferente para cada funcionalidad del Cubinja con el objetivo de mantener un código modular.

8.3.1. Módulos diseñados

En la figura 8.1 se presenta un diagrama de bloques de la estructura del firmware diseñado. Se describen brevemente los módulos de firmware desarrollados:

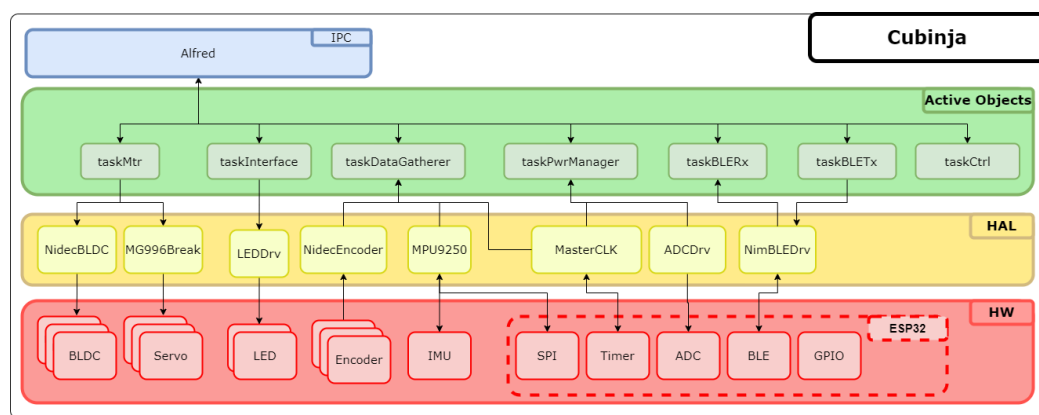


Figura 8.1: Diagrama de bloques del firmware diseñado, separado en cuatro niveles de abstracción diferentes según su jerarquía. En el nivel más bajo se encuentra el hardware, englobando a todos los componentes del cubinja. Luego se encuentra la HAL del sistema utilizada luego más arriba por todos los active objects para poder interactuar con cada componente del hardware. Finalmente en la sección superior se encuentra Alfred, la sección de código más abstraída del hardware que cumple el rol de IPC, siendo el encargado de la comunicación entre todos los *active objects*

- **HAL:** Todos los integrantes de la HAL proveen una API para el manejo de todos los periféricos HW del Cubinja con el objetivo de abstraer a la aplicación del bajo nivel. Dentro de la HAL encontramos los drivers de
 - Nidec BLDC
 - MG996 Servo motores
 - LEDs
 - Nidec BLDC Encoders
 - IMU MPU9250
 - ESP32 Hardware Timers
 - ESP32 ADC
 - ESP32 *Bluetooth* module

■ **Active objects:**

- **taskMtr:** Tarea que se encarga de setear el duty Cycle solicitado en alguno de los tres motores BLDC y además comenzar el proceso de frenado mediante el manejo de los servomotores
 - **taskInterface:** Tarea se encarga de manejar los LEDs indicadores de conexión BLE y batería baja
 - **taskDataGatherer:** Tarea generadora de datos que se encarga de muestrear al MPU9250 y los encoders de los tres motores BLDC. Luego de muestrearlos los procesa para obtener los estimadores correspondientes de cada variable de estado del sistema
 - **taskPwrManager:** Tarea que se encarga de la lectura del ADC para adquirir el nivel de batería del sistema
 - **taskBLERx:** Tarea que se encarga del procesamiento de todos los mensajes recibidos via *Bluetooth* por el ESP32
 - **taskBLETx:** Tarea que se encarga de armar los paquetes correspondientes a ser transferidos vía *Bluetooth* a la aplicación principal
 - **taskCtrl:** Tarea consumidora de los datos proporcionados por Task-DataGatherer, tiene el objetivo de controlar la máquina de estados del sistema e implementar el lazo de control según el estado en el que el sistema se encuentre.
- **Alfred:** Este objeto se encarga de implementar el IPC del sistema, almacenando todas las credenciales de freeRTOS de cada active object del sistema, así es como cada tarea al obtener una instancia de Alfred es capaz de comunicarse con cualquier otra tarea de manera ordenada con las otras tareas a través de el.

8.3.2. IPC: Alfred

Como se menciona en la introducción de esta sección uno de los objetivos del firmware es brindarle a futuros usuarios una plataforma de trabajo de fácil acceso, intuitiva y flexible a modificaciones.

Por esta razón es fundamental la correcta modularización del sistema, de esta forma se logrará un código sencillo de mantener, escalar y depurar.

Por lo tanto, con el objetivo de disminuir la dificultad y dependencia de cada tarea entre si se diseñó un objeto IPC al que se le llamo Alfred.

Alfred consiste en una clase del tipo *singleton*, esto significa que hay un solo objeto Alfred en todo el sistema y cada vez que un active object obtiene una instancia de Alfred esta obteniendo una instancia del mismo objeto ya que es único.

El contenido de Alfred es una lista con los handlers de cada active object del sistema. Como se muestra en la figura 8.2 al instanciar una nueva tarea en el

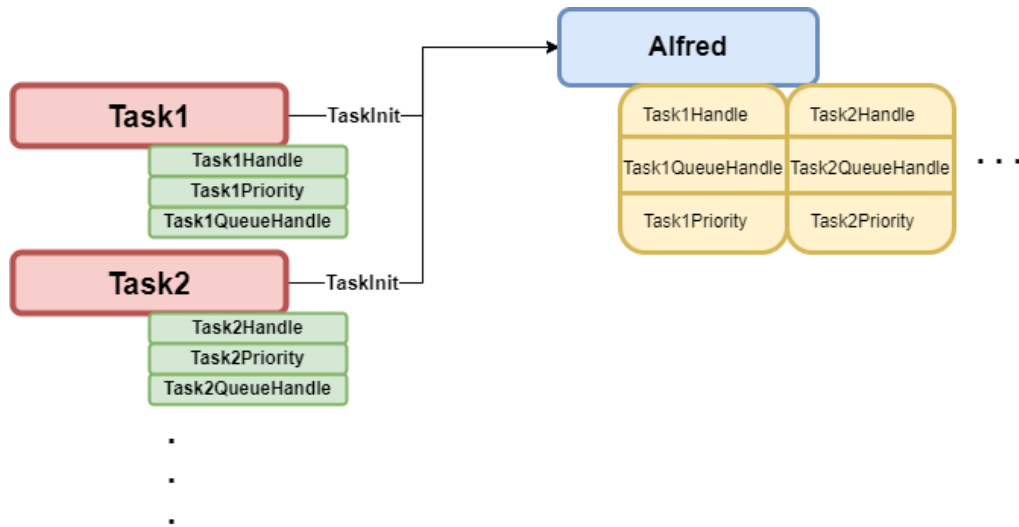


Figura 8.2: Diagrama que muestra como al inicializar una tarea nueva esta adjunta todos sus handlers a Alfred y este arma una lista con los handlers para cada tarea.

sistema esta comienza por pasarle todos los parámetros de la misma a Alfred que los almacena en forma de lista.

Una vez guardado los parámetros de cada tarea, Alfred brinda métodos a cada active object que le permite a cada tarea encolar mensajes, notificar o señalar a otras tareas sin tener que acceder directamente a ningún objeto de la otra tarea, evitando así fallas por datos compartidos o dependencias ocultas.

De esta forma, Alfred funciona como un “mayordomo” para los active objects del sistema, haciendo que estas no tengan que ocuparse de nada más que sus tareas y toda la comunicación entre ellas sea resuelto por él.

8.3.3. Maquina de estados

En la figura 8.3 puede verse los estados posibles en los cuales el Cubinja se puede encontrar y la tarea taskCtrl se encarga de controlar en base a los datos de la posición angular del sistema brindados por taskDataGatherer.

El sistema inicia en el estado de calibración en el cual se chequea la comunicación con el MPU9250 y además se hace una medición de la posición y offset iniciales en todos los ejes del giroscopio, una vez terminado este proceso el siguiente estado será de equilibrio o reposo según el valor de la posición inicial.

Se considera al sistema en reposo cuando este se encuentra muy lejos de cualquiera de sus puntos de equilibrio. En este estado se apaga el controlador, apagando los motores y dejando de insertar nuevos valores a la entrada de los mismos.

En caso de que el sistema se encuentra próximo a alguna de sus posiciones de equilibrio se considera que el sistema está en el estado de equilibrio. En este estado de funcionamiento el controlador está encendido solicitando nuevos valores a la entrada del motor a una frecuencia f_s .

Este estado posee 4 sub estados, uno por cada punto de equilibrio, esto es

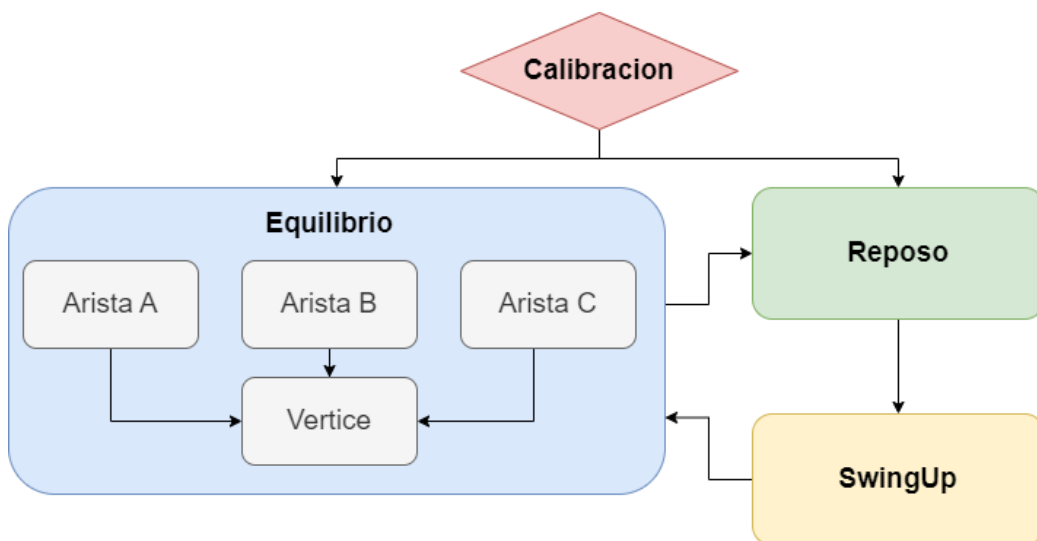


Figura 8.3: Diagrama de estados del sistema, se muestran los 4 estados posibles del sistema y la transición entre ellos. La única entrada de esta maquina de estados son los estimadores de la posición angular del mismo.

debido a que la acción del controlador es diferente para cada punto de equilibrio, por lo cual no se puede considerar un mismo estado para todos los puntos de equilibrio.

El último estado en el que el sistema se puede encontrar es el de SwingUp, en este estado se está preparando el sistema para realizar un salto. En este estado taskMtr setea un voltaje de entrada del correspondiente motor, espera a alcanzar una velocidad objetivo y se accionan los servomotores, en este instante el sistema transiciona al estado de equilibrio. Durante todo este proceso el controlador se encuentra apagado y se ignoran los datos de taskDataGatherer.

Observar que las tareas interface, powerManager, BLERx y BLETx no interfieren en el funcionamiento de la máquina de estados, ya que los eventos que estas tareas manejan son asíncronos e independientes de la posición del Cubinja

8.3.4. Implementación del controlador

La acción de control se implementa entre tres tareas, taskDataGatherer, taskCtrl y taskMtr. En la figura 8.4 se muestra el flujo de datos de la acción de control. El sistema comienza con MasterClk notificando a taskDataGatherer con una frecuencia igual a f_s .

Recibida la notificación taskDataGatherer realiza una lectura del MPU9250 y a los Encoders, luego ejecuta los algoritmos explicados en el capítulo 3 para obtener así un estimador de cada variable de estado del sistema para luego empaquetarlo y encolarlo hacia taskCtrl

TaskCtrl corre la máquina de estados descrita en la sección anterior y en base al estado del sistema ejecuta la acción de control pertinente, hecho el cálculo de los nuevos valores de voltaje de control la tarea empaqueta el dato y lo encola a

8.4. Comunicación inalámbrica

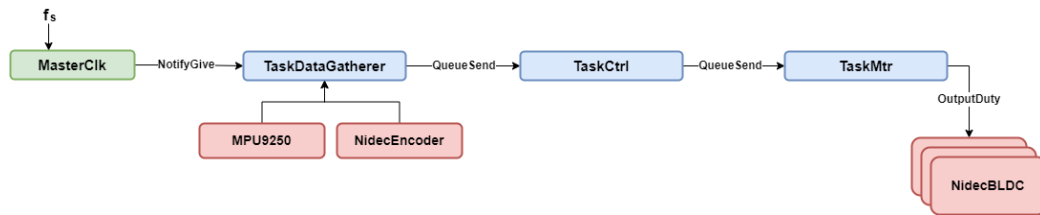


Figura 8.4: Diagrama que muestra el flujo de datos en el sistema en la acción de control, comenzando por taskDataGatherer, siguiendo a taskCtrl y finalizando en TaskMtr.

la tarea taskMtr

La tarea taskMtr al recibir un paquete de datos del controlador simplemente setea a la entrada de cada motor BLDC el duty cycle pertinente para cada entrada de voltaje recibida.

8.4. Comunicación inalámbrica

En esta sección se amplía sobre el uso de *Bluetooth Low Energy* (BLE) como enlace de comunicación inalámbrica con el Cubinja. La tecnología BLE es una tecnología inalámbrica que permite la comunicación de baja potencia entre dispositivos cercanos. A diferencia de *Bluetooth* clásico, o WiFi, BLE consume mucha menos energía, lo que lo hace ideal para dispositivos alimentados por baterías o que necesitan una larga duración de la batería.

Utilizaremos BLE para establecer una comunicación inalámbrica eficiente y en tiempo real con otros dispositivos. Una de las ventajas de utilizar BLE es la facilidad con la que se puede establecer un servicio UART, lo que permite a los dispositivos conectados comunicarse a través de un puerto serie virtual.

Otra ventaja de BLE es la facilidad para desarrollar aplicaciones utilizando lenguajes populares como *Python*. Esto hace que el desarrollo de una aplicación para el monitoreo y control desde una PC sea sencillo.

8.4.1. Sobre módulo BLE en el ESP32

El stack de protocolo *Bluetooth* del ESP32 está diseñado para cumplir con las especificaciones del estándar *Bluetooth* y BLE y ofrece soporte para los perfiles GATT (Generic Attribute Profile) y GAP (Generic Access Profile), fundamentales para el desarrollo de aplicaciones *Bluetooth* y BLE. Para el manejo de este stack se cuenta con dos librerías diferentes *Bluedroid* y *NimBLE*.

NimBLE es una librería de código abierto diseñada específicamente para el ESP32 que ofrece un menor consumo de energía, un menor uso de memoria y una mayor flexibilidad en la configuración de dispositivos BLE. Esto la hace ideal para aplicaciones de baja potencia y distancia corta. *NimBLE* también ofrece soporte para los perfiles GATT y GAP, y es compatible con la mayoría de los dispositivos BLE del mercado.

Capítulo 8. Diseño del software embebido

En cambio *Bluedroid* ofrece soporte para *Bluetooth* clásico y BLE, pero no está optimizada para el bajo consumo de energía y no es tan eficiente en el uso de la memoria como *NimBLE*, por esta razón se utilizará *NimBLE* para el manejo del periférico BLE del ESP32

8.4.2. Características

En BLE, los datos y las funcionalidades se organizan en unidades estructuradas conocidas como Servicios y Características.

Un Servicio en BLE es una entidad lógica que agrupa funcionalidades relacionadas que un dispositivo puede ofrecer. Se define mediante un UUID (Identificador Único Universal) y sirve como un contenedor de alto nivel que agrupa Características relacionadas. Cada servicio representa un conjunto específico de comportamientos o capacidades que un dispositivo periférico ofrece a los dispositivos centrales. Los servicios permiten una organización estructurada de las funcionalidades y datos que un dispositivo ofrece, lo que facilita la interoperabilidad y el descubrimiento de servicios en dispositivos BLE.

Una característica en BLE, que se encuentra dentro de un servicio, es una entidad más granular que representa un dato específico o una funcionalidad que el dispositivo ofrece.

En el caso del Cubinja solo se cuenta con un servicio llamado UART service el cual tiene dos características, TX para transmisión de datos y RX para recepción de datos.

El UART service en BLE proporciona una manera sencilla y efectiva de lograr una comunicación inalámbrica entre dispositivos. Las principales ventajas del mismo son:

- **Comunicación bidireccional:** El servicio UART permite una comunicación bidireccional entre el dispositivo BLE y otro dispositivo conectado, lo que facilita el intercambio de datos en ambas direcciones.
- **Simplicidad:** Al emular una interfaz de comunicación serie (UART), simplifica el proceso de envío y recepción de datos, ya que se puede desarrollar como una conexión serie tradicional
- **Acceso remoto al sistema:** El servicio UART BLE puede permitir el acceso remoto a los datos del dispositivo, lo que es útil para aplicaciones de monitoreo y control

El único inconveniente del UART service es que al emular una interfaz serial convencional no provee acceso directo a datos específicos si no que es responsabilidad del software embebido la correcta interpretación de los mensajes recibidos y transmitidos.

8.4.3. Protocolo de comunicación

Debido a que la comunicación remota con el Cubinja se hace a través de un servicio que emula una interfaz de comunicación serial, se creó un protocolo de

8.4. Comunicación inalámbrica

comunicación para la correcta decodificación de cada mensaje recibido y transmitido por el sistema. Como se muestra en la figura 8.5 cada mensaje cuenta con 3 campos Opcode, Length y Payload



Figura 8.5: Diagrama que muestra la estructura de cada mensaje BLE recibido y transmitido en el sistema Cubinja.

En el campo Opcode de largo un byte se determina la funcionalidad o dato del Cubinja al cual se está intentando acceder.

El campo Length también de largo un byte se determina el largo del payload del mensaje, este campo cobra gran importancia en la transmisión de datos hacia la app, la cantidad de bytes a transmitir no será siempre la misma sino que dependerá del tipo de mensaje.

El campo Payload consta de N bytes de largo, siendo N el valor en el campo Length. De esta forma, al leer el campo opcode de cada mensaje se sabe a qué funcionalidad del cubinja se está intentando acceder y, por lo tanto, se podrá decodificar el mensaje enviado en el campo payload de la forma correcta

8.4.4. Aplicación remota

Para el control del Cubinja de manera remota se desarrolló una aplicación de computadora en *Python* basada en la librería Bleak. Bleak es una librería de *Python* para interactuar con dispositivos BLE, que proporciona una interfaz unificada para encontrar, conectar y comunicarse con estos dispositivos en múltiples plataformas, incluyendo Windows, MacOS y Linux. Al ser una librería asíncrona, permite operaciones no bloqueantes, mejorando así la eficiencia de la aplicación al permitir la ejecución de otras tareas durante la realización de operaciones BLE. La librería cuenta con una amplia documentación y una comunidad de desarrolladores activa, lo que facilita su utilización y mantenimiento. Su naturaleza multiplataforma y su facilidad de uso hacen de Bleak una opción poderosa y flexible para el desarrollo de aplicaciones BLE en *Python*. La aplicación es controlada a través de la entrada estándar (teclado) de la computadora y permite el control de las siguientes funcionalidades del Cubinja:

- Adquisición de datos de sensores
- Adquisición de datos del controlador
- Comienzo de *Swing Up*
- Modificación de parámetros de control
- Testeo de Motores

Capítulo 8. Diseño del software embebido

- Testeo de Servo Motores
- Adquisición de nivel de batería

8.5. Auxiliares

El sistema además posee dos tareas auxiliares que corren continuamente independiente del estado del sistema `taskInterface` y `taskPowerManager`.

La tarea **`taskInterface`** controla el LED azul del Cubinja, esta enciende y apaga el LED según si se estableció una conexión BLE con el Cubinja y titila en caso de que el Cubinja se encuentre realizando un streaming de datos.

La tarea **`taskPowerManager`** monitorea el nivel de batería del sistema¹ y se comunica con la tarea `taskMtr` para que esta lo mantenga actualizado a la hora del seteo del `dutyCycle` para los motores. Además, con el fin de preservar la vida útil de la batería, apaga el controlador cuando la batería está demasiado baja, obligando al usuario a cargarla para poder continuar usando al Cubinja

¹Como se explicó en el capítulo 2 no se monitorea directamente el voltaje de la batería debido a que esta supera el rango de trabajo del ADC. Se monitorea el voltaje en un divisor resistivo alimentado por la batería

Capítulo 9

Ensayos

En este capítulo se describirán la serie de ensayos realizados tanto sobre el frame como sobre el Cubinja para el testeado de cada componente del sistema.

Los ensayos son diseñados con el objetivo de estudiar modularmente cada componente y luego la interacción entre ellos. Para la adquisición de datos del sistema se hará uso de la aplicación BLE diseñada en python testeando de esta forma además que el sistema es controlable de forma remota.

9.1. Ensayos Frame

Todos los ensayos del frame son realizados adquiriendo datos de manera remota con la aplicación principal de python para este sistema.

A su vez también el firmware para todos los ensayos es el principal del sistema sin ninguna modificación.

9.1.1. Testeo de filtro complementario

Para testear que el filtro complementario está funcionando correctamente se realizaron dos ensayos. Primero se hará una prueba estática para medir la precisión del estimador y luego se hará una prueba dinámica para verificar que el estimador sigue siendo correcto a pesar de que el sistema se encuentre en movimiento.

Para el ensayo estático se inclina el frame a un ángulo conocido y se observa la medida devuelta por el filtro complementario. Se realizó esto para un total de 7 ángulos distintos y se obtuvieron los resultados que se muestran en la tabla 9.1.

Vemos que para condiciones estáticas el filtro complementario se comporta de manera precisa, con un error de menos de 2% en el peor caso.

Se procede a realizar el testeado para condiciones dinámicas, para esto se hará oscilar al sistema entre dos ángulos conocidos a diferentes velocidades y se chequeará que la lectura de los mismos no cambia en función de esto.

Como se muestra en la figura 9.1 se hizo oscilar el sistema entre -20 y 20 grados a 4 velocidades diferentes obteniendo diferencias no apreciables entre cada iteración por lo que podemos concluir que el filtro complementario funciona correctamente.

Capítulo 9. Ensayos

Referencia (°)	Estimación(°)	Error relativo
-45	-44.2	1.7 %
-30	-29.8	0.6 %
-15	-14.9	0.7 %
0	0	0 %
15	15.2	1.3 %
30	29.7	1 %
45	45.8	1.8 %

Tabla 9.1: Resultados de test del filtro complementario, se muestra el estimador obtenido contra una posición de referencia

Se destaca además que todos los testeos fueron hechos con el volante girando a velocidad constante para evaluar la vibración introducida por el mismo.

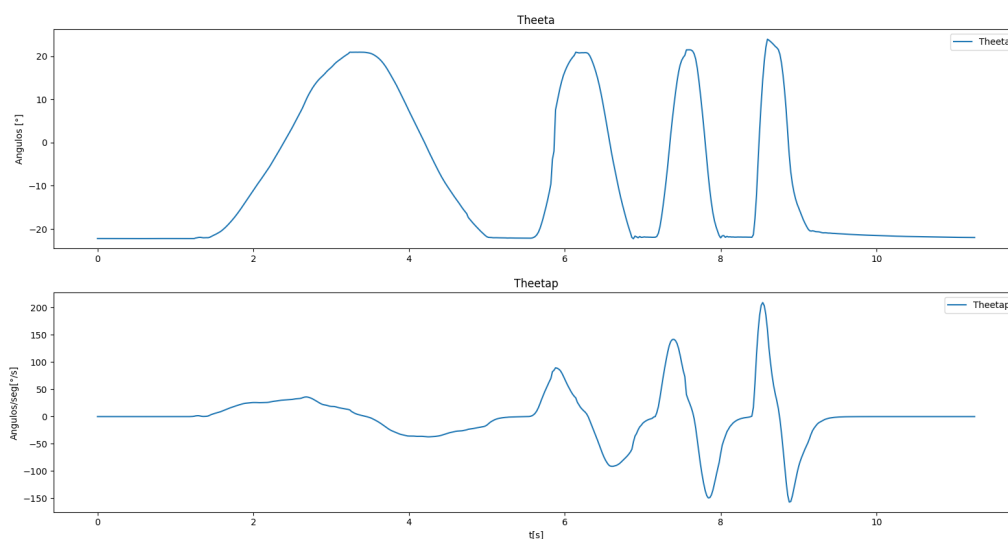


Figura 9.1: En la gráfica de arriba podemos ver la variación del ángulo a lo largo del tiempo y en la de abajo la velocidad.

9.1.2. Testeo de EMWA

Para el testeo del estimador de la velocidad angular del sistema se comparará la salida del EMWA contra la derivada de la posición angular del sistema.

En la figura 9.2 podemos ver ambos estimadores solapados.

9.1. Ensayos Frame

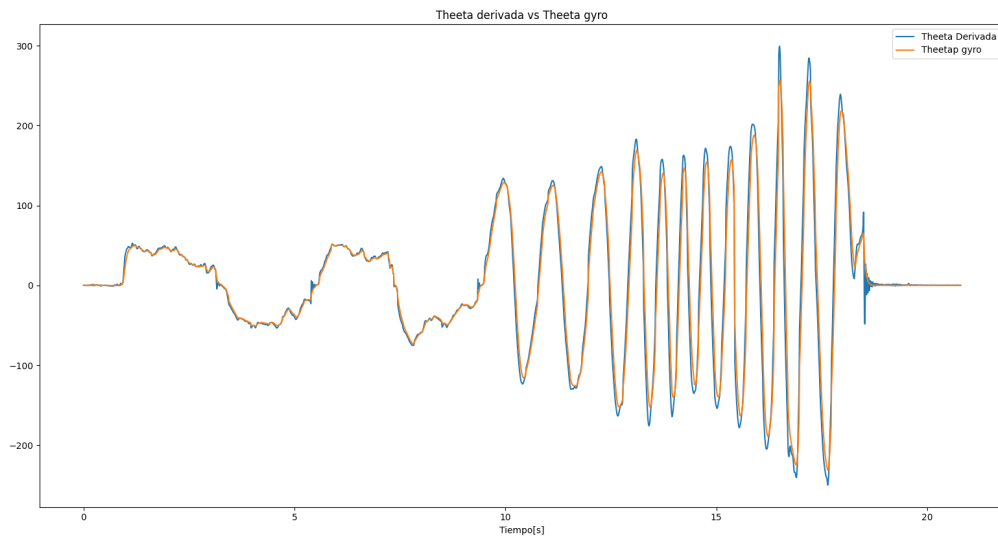


Figura 9.2: Gráfica de derivada del estimador angular vs salida del EMWA, ambos medidos en grados por segundo. En azul la derivada de la posición angular y en naranja los datos del giroscopio pasados por el EMWA.

Vemos que los datos del giroscopio siguen muy bien a la derivada de la posición angular pero de una forma más suave, podemos concluir entonces que dicho estimador funciona de manera correcta.

9.1.3. Testeo encoder

Para verificar que el sensado del encoder y el algoritmo de estimación de velocidad funciona correctamente se alimenta con distintos valores de tensión al motor y se ejecuta dicho algoritmo mientras se monitorea el encoder con un osciloscopio.

Los resultados obtenidos pueden verse en la tabla 9.2

Periodo encoder (μs)	Velocidad de referencia (RPM)	Velocidad estimada (RPM)	Error relativo
1830	327.87	322.5	1.63 %
934	645	637.5	1.16 %
587	1022.14	1012.5	0.94 %
402	1492	1480	0.8 %
331	1812.68	1807.5	0.3 %
223	2690.58	2687	0.1 %

Tabla 9.2: Resultado del ensayo sobre el encoder utilizando el método de tiempo fijo

Puede observarse que el resultado devuelto por el estimador es coherente con el período medido de los encoder, con un error relativo menor a 2 % en el peor de los casos, por lo cual podemos concluir que el algoritmo estimador de velocidad del volante funciona correctamente

9.1.4. Testeo Control

Para el testeo del controlador LQR diseñado se *setean* las constantes calculadas en el sistema y se lo lleva a su punto de equilibrio mientras se monitorea tanto su posición angular como el *duty cycle* de la PWM. impuesta sobre el motor.

Recordamos del capítulo 4 que el valor de las mismas son:

$$K = [23,55 \quad 2,69 \quad 0,094]$$

Se obtuvieron entonces los resultados de la figura 9.3.

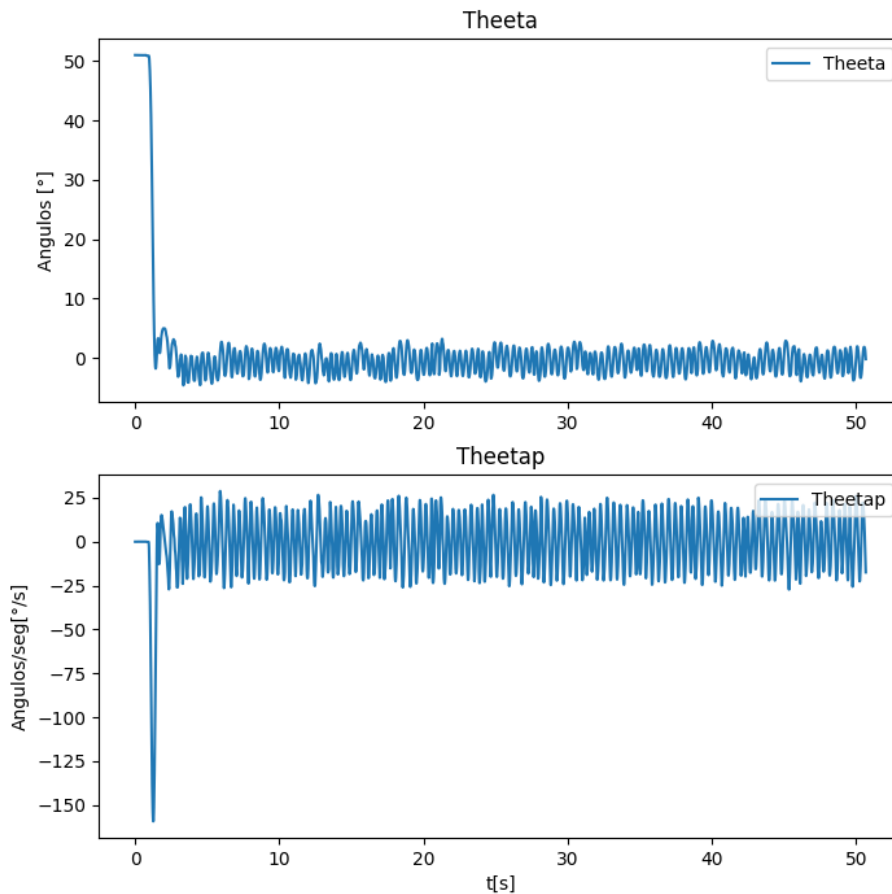


Figura 9.3: Posición y velocidad angular en función del tiempo con el controlador encendido

Los resultados del controlador son satisfactorios, el frame logra mantenerse próximo a su posición de equilibrio por tiempo indefinido sin caerse. Puede verse

9.1. Ensayos Frame

una pequeña oscilación constante con respecto al equilibrio pero estas no son lo suficientemente grandes como para que el sistema no se pueda recuperar.

Si bien el Frame logra mantenerse parado podemos ver que la velocidad se mantiene oscilando constantemente en valores significativamente altos, como se muestra en la figura 9.4, esto provoca que sobre el motor se estén imponiendo voltajes elevados muy frecuentemente haciendo el controlador muy agresivo.

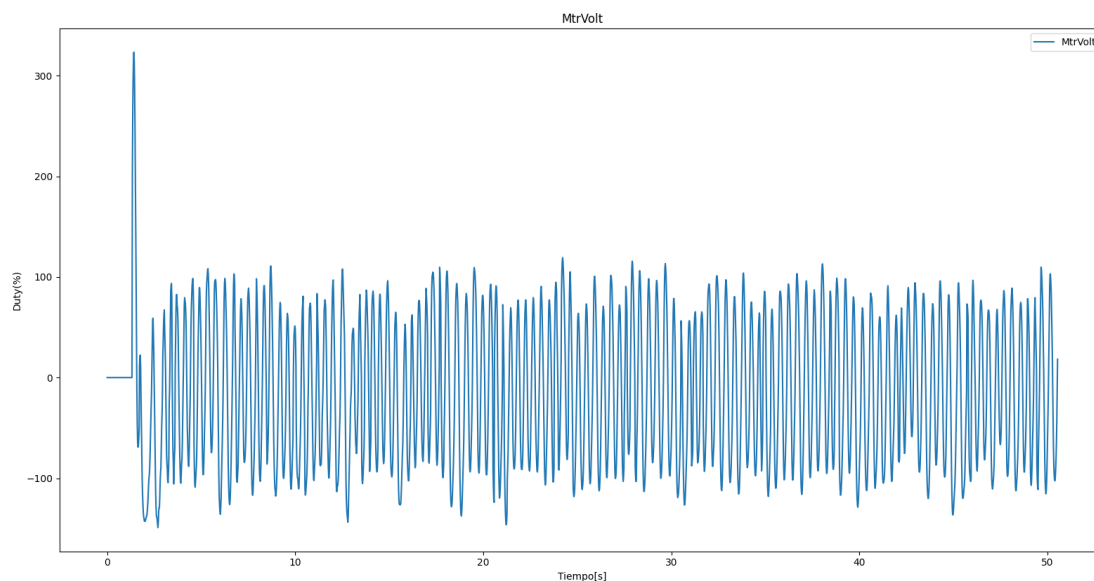


Figura 9.4: Gráfica del duty cycle de la PWM impuesta sobre el motor en función del tiempo. Se puede observar que en algunos puntos se solicita un duty cycle incluso mayor a 100 %. Esto significa que se solicita un voltaje mayor incluso al que se puede imponer, para estos valores se impone un duty cycle de 100 %.

9.1.5. Testeo swing up

Para los ensayos del *swing up* es necesario caracterizar cuatro parámetros importantes: el ángulo de reposo del servo α_r , ángulo de frenado α_f , tiempo de actuación del freno T_f y *duty cycle* para la alimentación del volante en el *swing up* δ_{su} .

La forma de obtener estos parámetros consiste en ensayar e iterar el frenado según la performance del proceso de *swing up*.

Primero se ensaya la posición de reposo y frenado del servo motor, es importante tener en cuenta que al definir la posición de frenado del servo se define la fuerza aplicada por el mismo. Se debe tener precaución ya que el servo motor no cuenta con un sistema de alerta de sobreesfuerzo y en caso de bloquear su desplazamiento los engranajes metálicos del mismo ceden ante la pista de frenado, inutilizando el motor.

Capítulo 9. Ensayos

De esta forma los ángulos de reposo y frenado se definieron con los siguientes valores:

$$\begin{aligned}\alpha_r &= 40^\circ \\ \alpha_f &= 65^\circ\end{aligned}$$

Una vez definidas las posiciones del servo, se realiza un barrido de aumento progresivo de la tensión de alimentación (PWM) del BLDC y por tanto de la velocidad del volante de inercia. El pseudo código que gobierna la implementación del sistema de frenado puede verse a continuación:

Algorithm 1 SWING UP

Data:PWM_BLDC = δ_{su} REST_ANGLE = α_r BRAKE_ANGLE = α_f BRAKE_t = T_f

BRAKE_stand_by = 5s

begin

set_brake_position = REST_ANGLE

set_BLDC_PWM = PWM_BLDC

delay(BRAKE_stand_by)

set_brake_position = BRAKE_ANGLE

*BLDC*internalbrake \leftarrow ON delay(BRAKE_t)

set_brake_position = REST_ANGLE

*BLDC*internalbrake \leftarrow OFF

El mismo consiste en dejar al servo motor en su posición de reposo todo el tiempo hasta que se comienza el proceso de *swing up*. Una vez que llega la orden del mismo se alimenta al motor con una PWM con el duty cycle δ_{su} y se espera un tiempo BRAKE.StandBy de 5 segundos para darle tiempo al volante a que se asiente en su velocidad objetivo. Este tiempo se estima en base a pruebas, llegando a la conclusión de que 5 segundos es tiempo suficiente para que el volante alcance su velocidad final.

Una vez transcurrido este tiempo se activa el freno interno del BLDC a la misma vez que se lleva al servo motor a su posición de frenado. El sistema se encontrará en este estado durante un tiempo igual a T_f , transcurrido este tiempo se libera el freno del BLDC y se lleva al servo motor a su posición de reposo.

Para definir δ_{su} se realiza un barrido de valores hasta lograr un *swing up* exitoso. Se modifica la velocidad angular del volante progresivamente y se observa el comportamiento del sistema. La prueba consiste en verificar a partir de qué velocidad de giro del volante el sistema se levanta y cae en su posición de reposo

9.1. Ensayos Frame

opuesta. Esto implica que la velocidad óptima que aproxima el sistema a la posición de reposo al frenar el volante se encuentra entre las velocidades en la que el sistema vuelve a caer a su posición de partida y que el sistema pasa la posición de equilibrio y cae hacia el otro lado. Los resultados de la prueba se pueden ver en la tabla 9.5.

RPM	δ_{su}	Resultado test	Posición final frame
231	10 %	FALLA	-45°
462	20 %	FALLA	-45°
693	30 %	FALLA	-45°
924	40 %	FALLA	-45°
1155	50 %	FALLA	-45°
1386	60 %	FALLA	-45°
1617	70 %	FALLA	-45°
1848	80 %	ÉXITO	45°
2079	90 %	ÉXITO	45°
2310	100 %	ÉXITO	45°

Tabla 9.3: Resultado de ensayos de frenado para el swing up

Finalmente, el último parámetro a definir es T_f , se busca que este sea el menor valor posible para lograr liberar el volante e iniciar la acción de control una vez que el frame se encuentra en una posición cercana a la posición de equilibrio. Para esto se definió un $T_f = 500ms$, el resultado fue logrado a partir de ensayos e iteraciones del frenado.

Los resultados de la experiencia indican que la combinación de parámetros ideal para generar el *swing up* del *frame* con el fin de encadenar el movimiento con el balanceo por control lineal es la siguiente:

$$\begin{aligned}\alpha_r &= 40^\circ \\ \alpha_f &= 65^\circ \\ 70\% &< \delta_{su} < 80\%. \\ T_f &= 500ms\end{aligned}$$

Se deberá variar $\Delta\delta_{su}$ en pequeños incrementos a medida que se perciba el desgaste de los elementos de fricción en el sistema de frenado. De esta forma se puede subsanar en cierta medida el desgaste de las piezas. Una vez que no se pueda alcanzar el resultado deseado se deberá realizar el cambio de cintas de frenado conforme se detalla en el manual del usuario (A).

9.2. Ensayos Cubinja

Nuevamente todos los ensayos son realizados adquiriendo datos de manera remota pero ahora usando la aplicación principal de python del Cubinja.

A su vez también el firmware para todos los ensayos es el principal del Cubinja sin ninguna modificación.

Para el testeo de sensores esta vez no se busca evaluar su precisión ya que los algoritmos implementados son los mismos que para el Frame y por lo tanto su precisión ya fue verificada. En cambio para los ensayos del Cubinja se busca verificar la funcionalidad de cada componente y la coherencia de los datos de los mismos.

9.2.1. Testeo encoders y motores

Para verificar que todos los motores y encoders del sistema se encuentran funcionando correctamente se envía el comando de test de motores para cada uno de ellos desde la aplicación central y se verifica que las velocidades alcanzada por cada motor sea la esperada según lo medido en sus encoders por un osciloscopio. Se obtiene entonces el resultado de la figura 9.5.

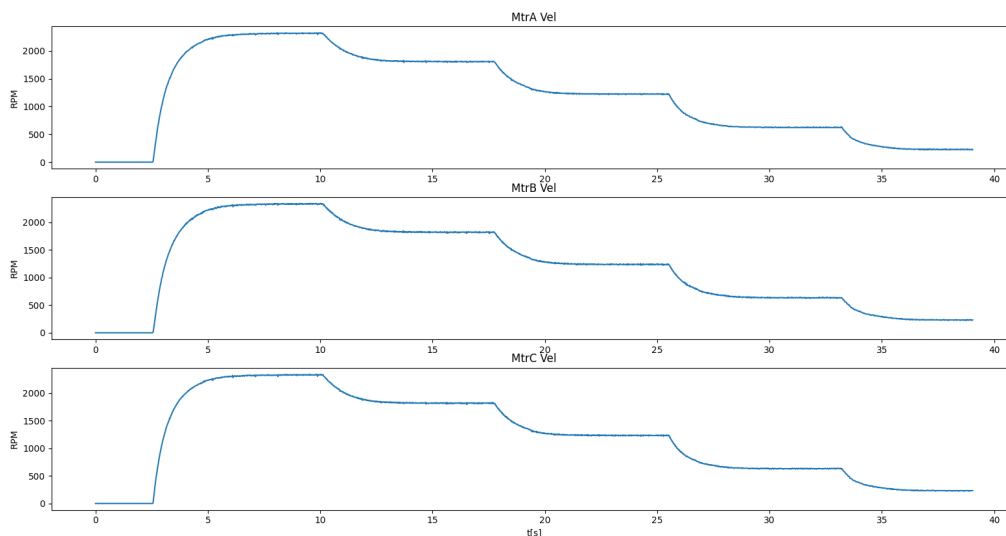


Figura 9.5: Gráfica obtenida desde la aplicación BLE en la que se ve la variación de la velocidad de los tres motores del Cubinja en función del tiempo.

Vemos que los tres motores responden exitosamente a los cambios en la PWM establecida en sus señales de control, por lo que podemos validar que cada motor del sistema es controlable.

Las lecturas de los encoders devolvieron los valores de la tabla 9.4 para cada velocidad.

9.2. Ensayos Cubinja

Duty	Velocidad estimada(RPM)	Velocidad estimada(RPM)	Error relativo
100	2330	2315	0.6 %
75	1747.5	1770	1.3 %
50	1165	1220	4.7 %
25	582.5	607	4.2 %
10	233	232	0.4 %

Tabla 9.4: Resultado de ensayos sobre encoders y motores. Nuevamente la velocidad estimada se obtiene a partir del monitoreo de los encoders con un osciloscopio.

Los valores estimados por el encoder dan muy próximos a los estimados con un error máximo de 4.7% en el peor caso. Por lo tanto, podemos concluir que los encoders están funcionando correctamente.

9.2.2. Detección de posición

En el caso del cubo entero son dos los ángulos de interés, θ y ϕ y ambos también estimados a partir del filtro complementario. Un diagrama representativo puede ser visto en la figura 9.6.

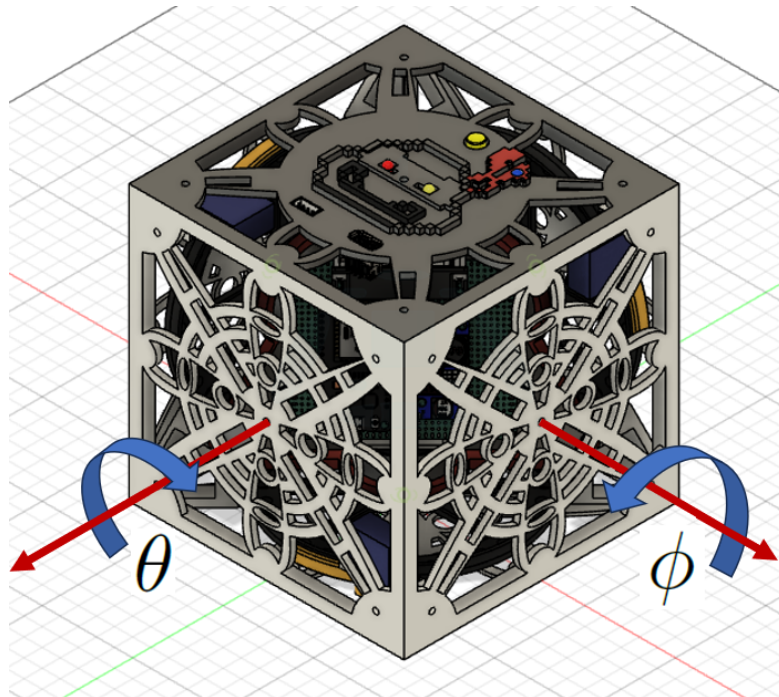


Figura 9.6: Diagrama representativo de los ángulos de interés del sistema cubo. Estos son el pitch y el roll del cubo

Estos ángulos no solo son utilizados en el lazo de control, sino que también comandan la máquina de estados que se describió en la sección 8.3.3 por lo tanto

Capítulo 9. Ensayos

se busca verificar que el sistema es capaz de detectar en que posición de equilibrio se encuentra en base θ y ϕ .

Para esto primero se lleva al Cubinja a sus 4 posiciones de equilibrio correspondientes y se registran los ángulos correspondientes de cada posición. Los resultados se pueden ver en la tabla 9.5.

	Vertice	Arista A	Arista B	Arista C
θ	2.8°	1.43°	32.8°	-31.32°
ϕ	2.7°	-34.5°	21.77°	21.8°

Tabla 9.5: Ángulos que muestran la posición de la IMU en cada punto de equilibrio

Para verificar que el sistema reconoce sus posiciones de equilibrio luego de ajustado los ángulos de cada una se escribe un programa que haga girar el motor correspondiente a velocidad constante cuando el sistema se encuentre en dicha posición. Esto es girar todos los motores cuando se encuentra en el vértice, el motor A cuando se encuentre en la arista A y así.

Se muestran los resultados obtenidos para la arista A y el vértice en las figuras 9.7 y 9.8.

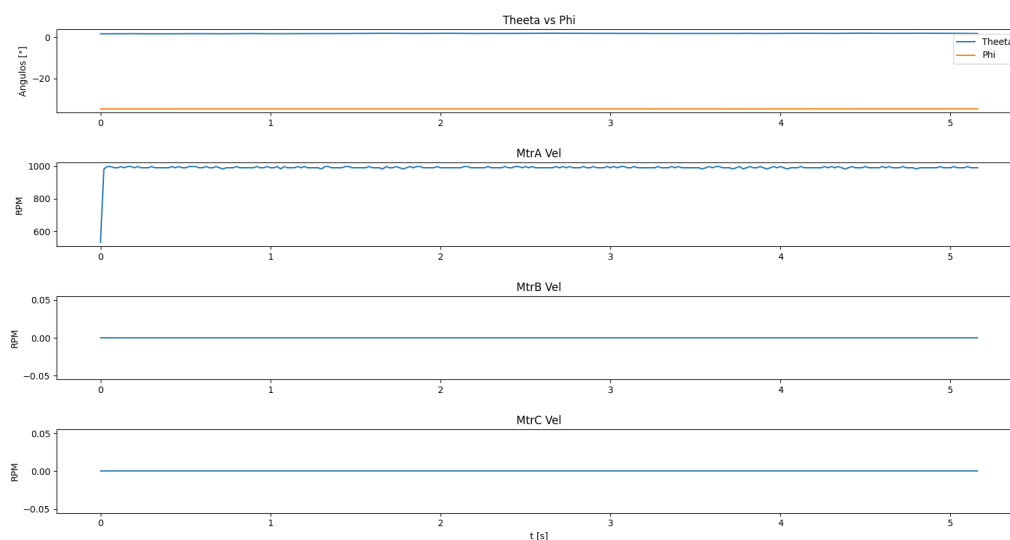


Figura 9.7: Gráficas donde se observan la velocidad de los motores junto a los ángulos θ y ϕ . Puede verse cómo en $\theta \approx 2^\circ$ y $\phi \approx -35^\circ$, los motores B y C permanecen quietos mientras el Motor A alcanza una velocidad de casi 1000 RPM.

Puede verse como al colocar al cubinja en una posición de equilibrio el motor que se mueve es el correcto, lo mismo sucede para las aristas B y C. Por lo tanto queda verificado que el algoritmo para detectar la posición relativa del Cubinja funciona correctamente.

9.2. Ensayos Cubinja

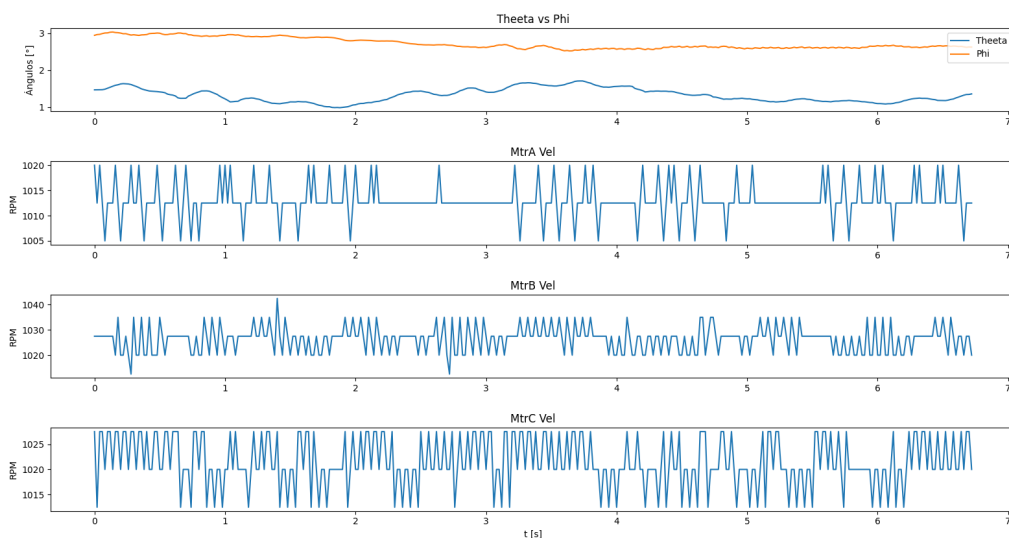


Figura 9.8: Gráficas donde se observan la velocidad de los motores junto a los ángulos θ y ϕ . Puede verse cómo en $\theta \approx 0$ y $\phi \approx 0$, todos los motores alcanzan una velocidad de casi 1000 RPM.

9.2.3. Controlador

Si bien no se desarrolló un controlador para el Cubinja se realiza una serie de ensayos para verificar la comunicación y el correcto manejo de datos por la tarea encargada de implementar un lazo de control.

Para la verificación de que esta funciona correctamente se efectúan tres ensayos sobre la arista A.

Primero se ensaya el control sobre la variable de estado θ , para esto desde la aplicación se setean todas las constantes de control a 0 menos K_{theta} la cual se setea a un valor arbitrario, 5 para esta prueba.

Luego se lleva al cubo a su punto de equilibrio y se lo desplaza suavemente al rededor de este mientras se monitorea el voltaje solicitado a los motores.

Como se muestra en la figura 9.9, el voltaje impuesto sobre el motor es cada vez más grande a medida que el Cubinja se aleja de la posición de equilibrio, haciendo que el mismo se mueva cada vez más rápido a medida que crece θ .

Capítulo 9. Ensayos

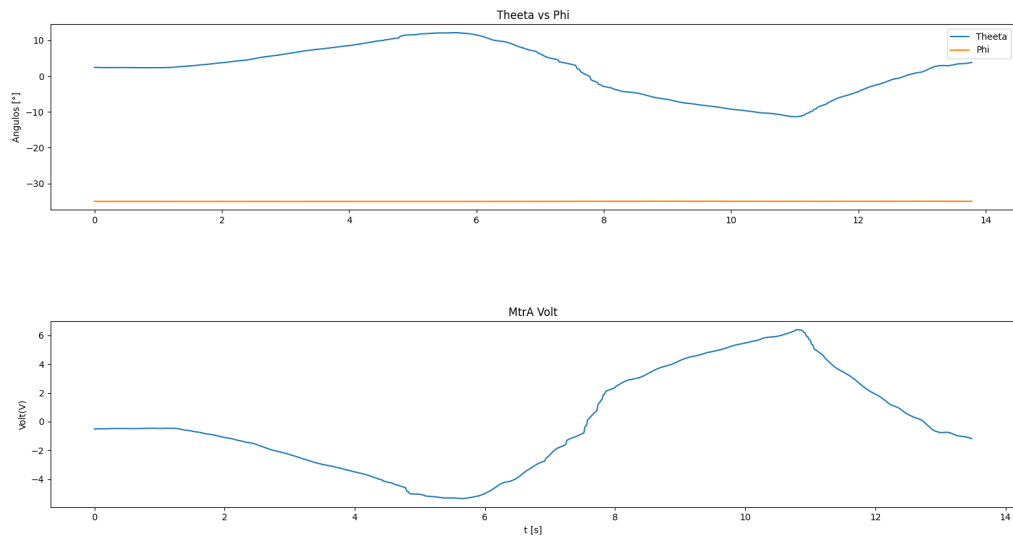


Figura 9.9: Gráfico del ensayo de control sobre θ . Se puede ver cómo al aumentar la diferencia con respecto a su posición de equilibrio, cada vez es mayor el voltaje impuesto al motor.

Se repite el mismo procedimiento, pero ahora seteando todas las constantes de control a 0 menos K_{θ} que controla la variable de estado $\dot{\theta}$.

Nuevamente, se lleva al cubo a su punto de equilibrio y ahora se lo hace oscilar con respecto a su punto de equilibrio a diferentes velocidades, obteniendo los resultados de la figura 9.10.

9.2. Ensayos Cubinja

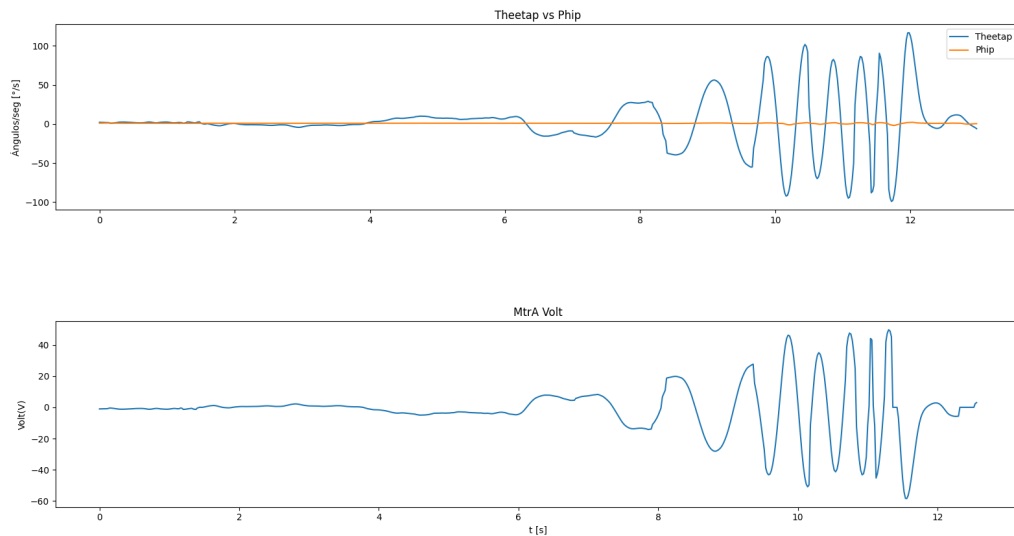


Figura 9.10: Gráfico del ensayo de control sobre $\dot{\theta}$. Se puede ver cómo a mayor velocidad, mayor el voltaje impuesto sobre el motor.

Los resultados de este ensayo son también satisfactorios, puede observarse como al aumentar la velocidad con la que se hace oscilar al Cubinja mayor es la fuerza que el motor realiza sobre el sistema.

Finalmente, se ensaya el control sobre ω , para esto se setean todas las constantes de control a 0 salvo por K_{MtrA} . Nuevamente, se lleva al sistema a su punto de equilibrio y esta vez se hace girar manualmente el volante a diferentes velocidades. En la figura 9.11 puede apreciarse el resultado.

El resultado es nuevamente satisfactorio, al intentar girar el volante el sistema establece un voltaje sobre los motores que frenan dicho movimiento impidiendo así que la rueda gire infinitamente.

Capítulo 9. Ensayos

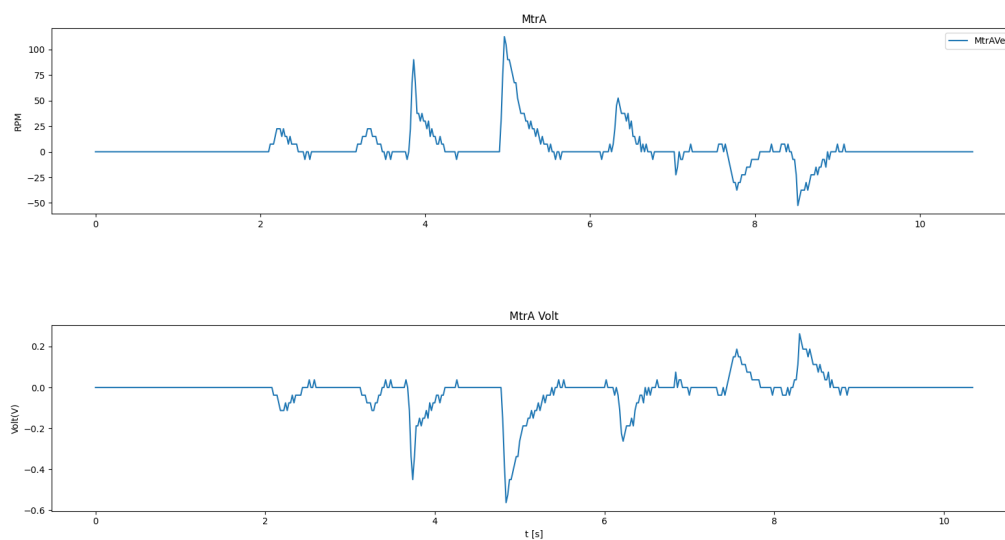


Figura 9.11: Gráfico del ensayo de control sobre ω_f . Se observa cómo al aumentar la velocidad del volante, el sistema establece un voltaje que se opone a dicho movimiento.

9.2.4. Monitoreo de batería

Para el testeo de la lectura de nivel de batería se sustituye la batería por una fuente de tensión variable y se lee el valor del ADC a diferentes niveles de tensión. Los resultados obtenidos se pueden ver en la tabla 9.6

Voltaje impuesto(V)	Voltaje medido(V)	Error relativo
12.6	12.71	0.8 %
12	12.2	1.6 %
11.5	11.7	1.7 %
11	11.1	0.9 %
10.5	10.6	1 %
10	10.1	1 %

Tabla 9.6: Resultados de ensayos de medición de batería

La tensión leída está dentro de un rango aceptable en comparación con la tensión seteada en la fuente de tensión, con un error máximo de menos de 2 % por lo cual queda validada la función de monitoreo remoto del nivel de tensión en la batería.

Capítulo 10

Conclusiones

10.1. Conclusiones finales

Se completó con éxito el sistema Cubinja. Las etapas establecidas para el proyecto se cumplieron de buena manera. La revisión bibliográfica de antecedentes y estado del arte rindió sus frutos, y permitió elaborar el modelado del sistema en 1D para bosquejar la implementación del sistema. Los objetivos de diseño de software y hardware, tanto para controlar el cubo y el frame como para establecer una interfaz de comunicación interactiva con el usuario fueron alcanzados satisfactoriamente. Se puede afirmar que el sistema diseñado cumple con todos los requisitos propuestos post re estructuración del proyecto, definidos una vez que desiste el tercer integrante y el grupo pasa a conformarse por dos estudiantes.

En particular se diseñó e implementó el software de sistema embebido con arquitectura basada en *Active objects* sobre el sistema operativo *FreeRTOS* en forma exitosa. Se comprobó el correcto funcionamiento de todas las tareas del código y funcionalidades del sistema. Además, el código final resultó presentado en forma modular para su fácil comprensión y testeo, dejando así la posibilidad de ser modificado por el usuario. Se implementó una interfaz de comunicación por medio de una app en Python con la cual se verificó la entrada y salida de datos del sistema, pudiendo graficar distintos parámetros y datos recolectados por el microcontrolador. El funcionamiento del registro de mensajes de alerta y comunicación se implementó en forma exitosa, entregando el estado del cubo en tiempo real.

Se implementaron y verificaron una serie de testeos para distintos componentes (servo motores y BLDC) y acciones del sistema (*swing up*) en forma exitosa. Se implementó también un protocolo de comunicación via la aplicación Light blue por celular (conexión bluetooth) donde se pueden modificar parámetros y evaluar el desempeño del sistema en forma ágil, sin computadoras ni cableado de por medio.

Se diseñó e implementó la interfaz física de comunicación con el usuario que le permite las siguientes funcionalidades:

- Conectar y desconectar la energización de 11.1V del sistema.

Capítulo 10. Conclusiones

- Conocer el estado de carga de la batería por medio de un indicador LED.
- Conocer el estado de conectividad bluetooth del sistema.
- Cargar un nuevo programa en el microcontrolador via USB.
- Cargar la batería LiPo interna al sistema por un conector de fácil acceso.

Constructivamente se diseñaron (tomando como punto de partida diseños de proyectos anteriores), imprimieron y ensamblaron las piezas estructurales de ambos prototipos. Se comprobó la robustez en ambos sistemas. Se verificó la facilidad de armado y desarmado del sistema en caso de que el usuario deba hacerle modificaciones o ajustes internos. Se resolvió el agregado de peso al volante de inercia en forma sencilla con la adición de tornillos y tuercas, permitiendo al usuario alterar la inercia del volante.

Se valida el diseño del PMC del cubo, comprobando la autonomía de funcionamiento del mismo en todos sus niveles de tensión. No se llega a recopilar datos suficientes como para estimar la duración de la carga con el sistema en funcionamiento en régimen. Aún así, se verifica que el requisito de treinta minutos del sistema funcionando alimentado por la batería se cumple.

Se comprobó el correcto funcionamiento del sistema al aplicarle la ley de control que desee el usuario. Al trabajar con un controlador por realimentación de estados seteando los parámetros de control obtenidos por el método de LQR, los comportamientos y movimientos del volante en función del movimiento del sistema (medido por la IMU) son los esperados.

El movimiento de *swing up* se logró de forma satisfactoria, pudiendo elevar el sistema con la acción de frenado del volante para aproximar al dispositivo a la zona de equilibrio. Debido al desgaste que sufren los encintados de la zapata y el volante al frenar, resultó difícil comprobar la vida útil de los elementos del sistema de frenado.

En el prototipo en 1D se validó el funcionamiento del sistema completo. Se logró equilibrar el frame en su posición de equilibrio aplicando los valores obtenidos por el método de LQR al controlador. Su comportamiento es el esperado y el equilibrio se logra por el tiempo que desee el usuario. Además, el sistema en la posición de equilibrio soporta pequeñas perturbaciones externas sin caer.

Habiendo comprobado que se puede lograr el equilibrio en el sistema 1D se da por supuesto que el cubo 3D puede ser balanceado en una arista. Se hacen pruebas post entrega de este documento para validar la suposición. No se llega a comprobar el equilibrio en un vértice ni la secuenciación de frenados de volantes para lograr el *swing up* al vértice. Queda como desafío para futuros estudiantes.

El proyecto realizado resultó ser una experiencia enriquecedora ya que involucró varias áreas de la ingeniería eléctrica que resultaron ser de gran interés para el grupo. El diseño de un sistema electro mecánico, realizado del modelado físico, selección de componentes adecuados, diseño e impresión 3D, desarrollo del sistema embebido e integración de todos estos cumplió con todas las etapas de un proyecto de ingeniería. A su vez, se generaron conocimientos en el manejo de un proyecto

íntegro: se gestionaron tiempos y dedicación de recursos, compras y gastos y resolución de problemas no previstos de buena manera. En particular, se pudo reestructurar el proyecto una vez perdido un tercio de la mano de obra del grupo y esto se hizo en forma satisfactoria.

10.2. Trabajo futuro

Como trabajo a futuro surge en primer lugar, el continuar trabajando con el prototipo del cubo para lograr equilibrarlo en una arista. Este equilibrio es factible dado que se pudo equilibrar el prototipo 1D en forma satisfactoria por el tiempo deseado. Trasladar el concepto al equilibrio del cubo en una arista solamente implica un mayor peso del sistema y un punto de apoyo distinto.

Una vez comprobado este punto se podría comenzar a trabajar persiguiendo el equilibrio del cubo en un vértice y también llevarlo a la posición de equilibrio por medio del *swing up*.

No se pudo hacer una medición del tiempo de autonomía que tiene el sistema alimentado desde la batería, más allá de haber comprobado el correcto funcionamiento del sistema al momento de la mayor demanda de amperaje posible, con todos los actuadores encendidos a máxima potencia y también habido superado el mínimo tiempo de carga de treinta minutos. Establecer un tiempo estimado de carga sería útil para el usuario, dado que la solución existente al momento consiste en cargar el sistema una vez accionada la alarma de batería baja.

A futuro queda pendiente también elaborar una interfaz gráfica para la aplicación de *python*, de forma de que sea más amigable para el usuario de lo que es ahora. Se recuerda que la comunicación es a través de la entrada estándar de teclado en la terminal del programa. Esta *app* podría contar con una actualización que permita monitorear el controlador y los sensores en tiempo real, pudiendo de esta forma por ejemplo graficar la posición del sistema también en tiempo real.

Por último, y en vista a la complejidad que presentó el cableado y soldado de componentes en el circuito eléctrico hecho a mano, surge la posibilidad de diseñar un PCB con posterior fabricación a medida por encargo.

Independientemente de los trabajos futuros propuestos, el sistema alcanzado logra cumplir con todos los objetivos establecidos y brinda una herramienta poderosa a futuros estudiantes que quieran aceptar el desafío de equilibrar el CUBINJA.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Manual de armado

A.1. Herramientas

A continuación se detallan las herramientas que serán de uso para el armado y desarmado de los prototipos.

- Destornillador phillips
- Cinta asiladora
- Pinza fina
- Pinza gruesa
- Trincheta
- Tijera
- Cinta doble faz

A.2. Precauciones y cuidados

Ambos dispositivos cuentan con partes móviles girando a revoluciones considerables. Dado el peso y dureza de las partes, se debe tener especial cuidado al operar los dispositivos mientras se encuentran encendidos. Se recomienda no acercar manos ni dedos al sistema mientras se encuentra operando. En particular no se debe tratar de frenar los volantes de inercia con las manos, siempre es preferible activar el sistema de frenado.

El sistema cuenta con varios niveles de tensión, todos ellos de bajo voltaje. Todos sus bornes se encuentran aislados eléctricamente. Aún así se recomienda no trabajar con la circuitería del sistema estando este energizado para evitar descargas.

Se pudo comprobar que los motores alcanzan temperaturas elevadas con el sistema operando en régimen. Esto es debido a la exigencia del comportamiento de cambio de dirección constante que se les exige. Se recomienda no tocar los BLDC con el

Apéndice A. Manual de armado

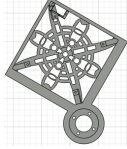
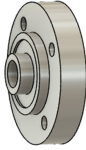

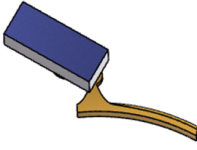
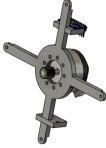
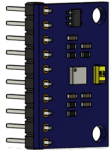
sistema operando ya que se podrían sufrir quemaduras leves.

Se aclara que siempre existe la posibilidad de falla del sistema en la que se desprendan partes en movimiento a altas velocidades. Esto podría ocasionar lesiones. Se recomienda no acercarse a la cara, en particular la vista, al sistema en funcionamiento. Dado que el peso de ambos dispositivos es considerable y en vista de que sus caídas se dan en forma repentina existe el riesgo de atrapamiento o aplastamiento de manos y dedos en la zona de operación de los dispositivos. Se recomienda precaución a la hora de preparar el lugar de trabajo.

Respecto a los cuidados del sistema, se aclara al usuario que en caso de sufrir roturas de piezas plásticas del sistema estas pueden ser reimpresas sin dificultad con los archivos *.gcode* o *.stl* del proyecto. En caso de rotura de componentes eléctricos, todos ellos pueden ser conseguidos fácilmente y a precios accesibles, muchos de ellos incluso en plaza.

A.3. Frame (1D)

A.3.1. Guía de partes

Componente	Partes	Imagen
Frame	4x Tornillos 3x10mm 4x Tuercas	
Rulemán	4x Tornillos 5x40	
Volante de inercia	Volante 16x Tornillos 5x8mm* 16x Tuercas* Cinta de frenado 2x Tornillos 3x6mm 2x Tuercas	
Freno	Servo motor Zapata Cinta de frenado Brazo servo 1x Tornillo 3x4mm 3x Tornillos 2x4mm	
Cruz motor	Cruz BLDC IMU 3x Tornillo 3x4mm	
IMU	2x Tornillos 3x4mm	

*La cantidad y distribución de los tornillos de peso puede variar a elección

A.3.2. Armado

Previo a detallar el armado y desarmado del prototipo se hacen algunas aclaraciones. Al trabajar con partes de plástico en conjunto con piezas de metal se debe tener especial precaución al ajustar. Fuerza excesiva puede hacer ceder el plástico, inutilizando la pieza.

- Colocar el rulemán dentro de la cavidad del soporte del frame. Alinear los orificios. Colocar los cuatro tornillos 5x40mm y ajustarlos con tuerca.

Apéndice A. Manual de armado

- Presentar el BLDC en la cruz, alinear los orificios y fijarlo con tres tornillos 3x4mm. Colocar la IMU en su plataforma y atornillarla con dos tornillos 3x4mm.
- Colocar los tornillos que hacen de peso 5x8mm con tuerca en el volante. Colocar la cinta de frenado en la pista exterior. Colocar el volante en la polea del BLDC y fijar su posición con dos tornillos 3x6mm con tuercas.
- Presentar el sistema cruz motor en los soportes del frame con la IMU del lado opuesto al rulemán. Fijar la cruz al frame con cuatro tornillos 3x10mm con tuerca.
- Colocar la cinta de frenado en la zapata y atornillarle el brazo con un tornillo de 2x4mm. Atornillar el brazo al servo motor con un tornillo de 3x4mm. Fijar el servo motor al frame con dos tornillos 2x4mm. Esta cinta de frenado en el volante y la zapata puede ser sustituida conforme se desgasta, según los resultados que vea el usuario al realizar el *swing up*.
- Fijar el frame a la base pasando un tornillo tipo esparrago a través de su centro, fijando la posición del frame y del esparrago en los soportes con tuercas.
- Se recomienda colocar soportes que amortigüen las caídas del dispositivo para evitar golpes innecesarios.
- Colocar la protoboard con el ESP32 lo suficientemente próximo al dispositivo para poder cablear los componentes sin dificultades.

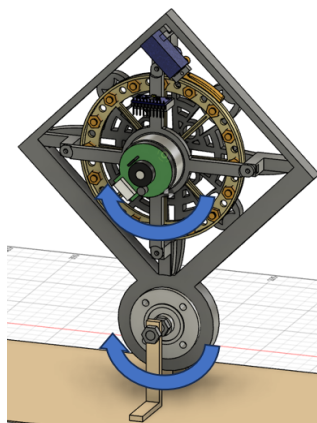


Figura A.1: Prototipo del frame completo.

A.3.3. Cableado

Los elementos a cablear son el BLDC, el servo motor y la IMU, a sus respectivas alimentaciones y al microcontrolador. En la figura A.2a se puede ver el

A.3. Frame (1D)

detalle de conexión de los cables del BLDC. La alimentación de 12V se toma de un transformador externo al sistema. Lo mismo con la alimentación de 5V para el servo motor. La alimentación del ESP32 se toma del conector USB. El resto de los cables se conectan a los pines correspondientes del microcontrolador. Se procede de igual forma con el cableado de la IMU, el servo motor y el LED de indicación de estado de conexión *bluetooth*. El detalle de pines del se puede ver en la figura A.2b.

+12v	violeta
GND	azul
BRAKE	verde
PWM	naranja
CW/CCW	amarillo
+3V3	blanco
CANAL A	rojo
CANAL B	negro

(a) Diagrama de colores y funcionalidades de motores BLDC.

```
//-----IMU PIN OUT-----
#define PIN_IMU_CS           GPIO_NUM_3
#define PIN_SCLK             GPIO_NUM_23
#define PIN_MOSI             GPIO_NUM_22
#define PIN_MISO             GPIO_NUM_21

//-----MOTOR PIN OUT-----

#define MTR_PULSE            GPIO_NUM_19
#define MTR_CW_CCW          GPIO_NUM_18
#define MTR_BRAKE            GPIO_NUM_5

//-----ENCODER PIN OUT-----
#define A_CHANN_PIN          GPIO_NUM_34
#define B_CHANN_PIN          GPIO_NUM_35

//-----SERVO PIN OUT-----
#define SERVO_PULSE          GPIO_NUM_4

//-----SERVO PIN OUT-----
#define BLUE_LED              GPIO_NUM_NC
```

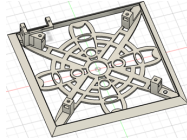
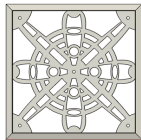
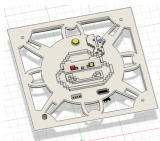

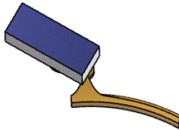
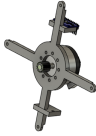

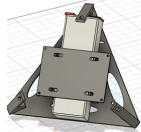
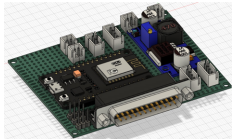
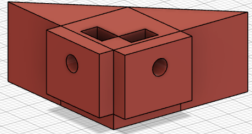
(b) Mapa de pines del frame.

Figura A.2: Cubo 3D.

Apéndice A. Manual de armado

A.4. Cubo (3D)

A.4.1. Guía de partes

Componente	Partes	Imagen
3x Frame motor	4x Tornillos 3x10mm 4x Tuercas	
2x Frame simple		
1x Frame interfaz	3x LED 2x Botón 1x USB-A hembra 1x Conector batería	
3x Volante de inercia	Volante 16x Tornillos 5x8mm* 16x Tuercas* Cinta de frenado 2x Tornillos 3x6mm 2x Tuercas	
3x Freno	Servo motor Zapata Cinta de frenado Brazo servo 1x Tornillo 3x4mm 3x Tornillos 2x4mm	
3x Cruz motor	Cruz 3x Tornillos 3x4mm BLDC 3x Tornillo 3x4mm	
IMU	2x Tornillos 3x4mm	
Soporte interno	14x Tornillos 3x6mm 14x Tuercas Batería LiPo	
Placa	LM2596 ESP32 DK Conector DB25 4x Tornillos 3x6mm 4x Tuercas Cinta doble faz	
4x Esquinero	3x Tornillos 3x6mm 3x Tuercas	

*La cantidad y distribución de los tornillos de peso puede variar a elección

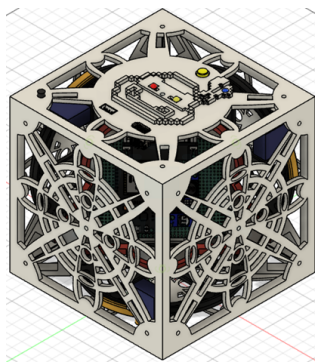
A.4.2. Armado

Previo a detallar el armado y desarmado del prototipo se hacen algunas aclaraciones. Al trabajar con partes de plástico en conjunto con piezas de metal se debe tener especial precaución al ajustar. Fuerza excesiva puede hacer ceder el plástico, inutilizando la pieza. Cada unidad de componente contiene la cantidad de partes vista en la lista anterior.

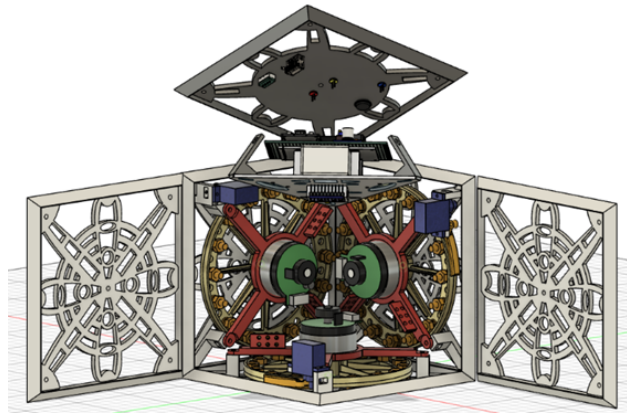
- Presentar el BLDC en la cruz, alinear los orificios y fijarlo con tres tornillos 3x4mm. Repetir para los tres motores.
- Colocar los tornillos que hacen de peso 5x8mm con tuerca en el volante. Colocar la cinta de frenado en la pista exterior. Colocar el volante en la polea del BLDC y fijar su posición con dos tornillos 3x6mm con tuercas. Repetir para los tres volantes.
- Presentar el sistema cruz motor en los soportes del frame motor. Fijar la cruz al frame con cuatro tornillos 3x10mm con tuerca. Repetir para los tres lados.
- Colocar la cinta de frenado en la zapata y atornillarle el brazo con un tornillo de 2x4mm. Atornillar el brazo al servo motor con un tornillo de 3x4mm. Fijar el servo motor al frame con dos tornillos 2x4mm. Repetir para los tres servo motores.
- Colocar los topes del soporte interior en los vértices del triángulo base con tres tornillos 3x6mm con tuerca. Presentar la batería en su posición. Colocar los topes de batería con cuatro tornillos 3x6mm con tuerca. Colocar la base de la placa eléctrica con cuatro tornillos 3x6mm con tuerca. Fijar la placa eléctrica a su base con cuatro tornillos 3x6mm con tuerca, agregando cinta doble faz para pegarla en su posición. Colocar la IMU en su posición con dos tornillos 3x4mm.
- Unir los tres frames con motor entre sí, fijándolos en su posición para que formen un ángulo recto y unir las caras con cinta aisladora. En este punto se debería tener medio cubo armado.
- Colocar los componentes de la interfaz con el usuario en su posición.
- Unir los tres frames restantes entre sí, fijándolos en su posición para que formen un ángulo recto y unir las caras con cinta aisladora.
- Conectar los componentes de la interfaz con el usuario en sus posiciones de la placa eléctrica.
- Colocar los conectores de los BLDC al cable DB25 y conectar el cable a la placa eléctrica.

A.4. Cubo (3D)

- Fijar el soporte interno en su posición, con un apoyo en cada una de las caras usando tres tornillos 3x6mm con tuerca. En este punto se debería tener la segunda mitad del cubo armada.
- Colocar los esquineros en posición para reforzar las uniones en caso de notar falta de rigidez en las uniones con cinta entre las caras.
- Unir las dos mitades del cubo con cinta aisladora.



(a) Cubo completo.



(b) Vista interior del cubo completo.

Figura A.3: Cubo 3D.

A.4.3. Cableado

Los detalles del funcionamiento del circuito se pueden revisar a lo largo del documento, en particular en los capítulos 2, 6 y 7. En la figura A.2a se puede ver el detalle de conexión de los cables del BLDC. Los motores A, B y C se cablean al conector DB25 según el detalle de la figura A.4. El conexionado de pines se puede llevar a cabo siguiendo el detalle de mapa de pines de la figura A.5 que fue tomado del firmware (*pinMap.h*).

arriba	1	2	3	4	5	6	7	8	9	10	11	12	13
motor_color		M3_verde	M2_verde	M1_verde	M1_naranja	M1_amarillo	M1_negro	M1_azul	M1_rojo	M2_rojo	M2_amarillo	M3_negro	M3_naranja
ESP32		PIN 27	PIN 15	PIN 5	PIN 19	PIN 18	PIN 35	GND	PIN 34	PIN 39	PIN 0	PIN 25	PIN 12
abajo	14	15	16	17	18	19	20	21	22	23	24	25	shield
motor_color	M3_blanco	M3_rojo	M2_violeta	M3_amarillo	M2_negro	M2_azul	M1_blanco	M1_violeta	M3_violeta	M3_azul	M2_naranja	M2_blanco	
ESP32	+3V3	PIN 26	+12V	PIN 14	PIN 32	GND	+3V3	+12V	+12V	GND	PIN 2	+3V3	

Figura A.4: Cableado del conector DB25 con los BLDC A, B y C.

Apéndice A. Manual de armado

```
//-----IMU PIN OUT-----
#define PIN_IMU_CS           GPIO_NUM_3
#define PIN_SCLK             GPIO_NUM_23
#define PIN_MOSI             GPIO_NUM_22
#define PIN_MISO             GPIO_NUM_21

//-----MOTOR PIN OUT-----

#define A_MTR_PULSE          GPIO_NUM_19
#define A_MTR_CH_CCM        GPIO_NUM_18
#define A_MTR_BRAKE         GPIO_NUM_5

#define B_MTR_PULSE          GPIO_NUM_2
#define B_MTR_CH_CCM        GPIO_NUM_0
#define B_MTR_BRAKE         GPIO_NUM_15

#define C_MTR_PULSE          GPIO_NUM_12
#define C_MTR_CH_CCM        GPIO_NUM_14
#define C_MTR_BRAKE         GPIO_NUM_27

//-----ENCODER PIN OUT-----
#define A_A_CHANM_PIN        GPIO_NUM_35
#define A_B_CHANM_PIN        GPIO_NUM_34

#define B_A_CHANM_PIN        GPIO_NUM_32
#define B_B_CHANM_PIN        GPIO_NUM_39

#define C_A_CHANM_PIN        GPIO_NUM_25
#define C_B_CHANM_PIN        GPIO_NUM_26

//-----SERVO PIN OUT-----
#define A_SERVO_PULSE        GPIO_NUM_4
#define B_SERVO_PULSE        GPIO_NUM_13
#define C_SERVO_PULSE        GPIO_NUM_33

//-----SERVO PIN OUT-----
#if DEBUG_MODE
    #define BLUE_LED          GPIO_NUM_NC
#else
    #define BLUE_LED          GPIO_NUM_1
#endif

#define YELLOW_LED           GPIO_NUM_NC

//-----ADC CHANN-----
#define ADC_CHAN              ADC1_CHANNEL_0 //GPIO 36
```

Figura A.5: Mapa de pines del cubo.

Apéndice B

Guía de instalación

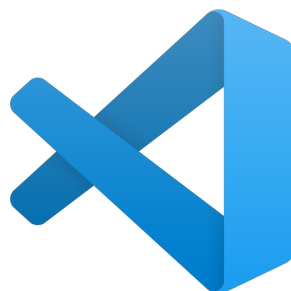
En este anexo se hará un breve tutorial de como instalar las herramientas utilizadas para compilar y flashear tanto al Cubinja como al frame.

El firmware del Cubinja fue desarrollado utilizando Visual Studio code(VSCode) como IDE.

VSCode es un editor de código fuente altamente popular y ampliamente utilizado desarrollado por Microsoft. Es un entorno de desarrollo ligero y altamente personalizable que brinda una amplia gama de características y extensiones para mejorar la productividad de los desarrolladores. En particular para el Cubinja resulta de gran utilidad su fácil interacción con el framework de desarrollo para el ESP32 creado por Espressif y con Python.

También se instalará ESP-IDF (Espressif IoT Development Framework) es un framework de desarrollo de código abierto creado por Espressif Systems. Está diseñado específicamente para el desarrollo de firmware y aplicaciones embebidas sobre sus microcontroladores ESP.

Finalmente la ultima herramienta a instalar será *Python* junto con las librerías necesarias para hacer uso de la aplicación para el manejo remoto del Cubinja.

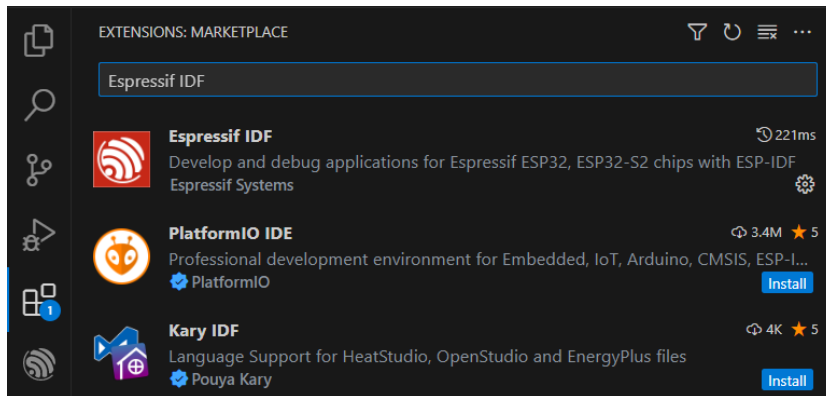


Se presenta entonces esta guía para la instalación de estas tres herramientas.

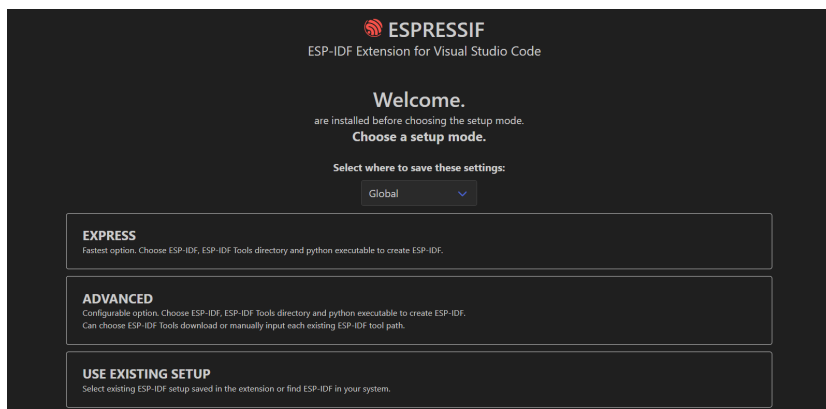
1. Descargar e instalar VS Code de la pagina oficial del mismo: <https://code.visualstudio.com/>

Apéndice B. Guía de instalación

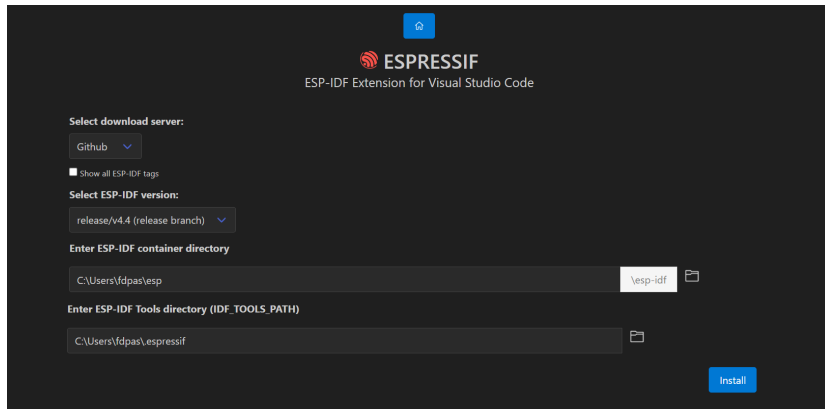
2. Para la instalación del ESP-IDF y su SDK lo haremos directo desde VS Code, para esto en la pestaña de extensiones de VS Code buscar *Espressif IDF*



3. Abrir la extensión descargada y en la pestaña de setup elegir *Express*



4. En la pestaña de configuraciones seleccionar la versión 4.4.1 del ESP-IDF, esta fue la utilizada para el desarrollo del Cubinja, no se asegura compatibilidad con versiones mas actualizadas del mismo. El resto de las opciones se recomienda dejarlas por defecto



5. Pasamos entonces a la instalación y configuración de *Python* para VS Code, primeramente descargar e instalar *Python* de la pagina oficial www.python.org
6. Como en los pasos anteriores, abrir la pestaña extensiones de VS Code y ahora instalar la extensión *Python*.
7. Finalmente pasamos a instalar las librerías de *Python* necesarias para ejecutar la aplicacion principal. Abrir la consola de Windows y ejecutar el comando: `pip install bleak`
8. En la consola ejecutar: `pip install scipy`

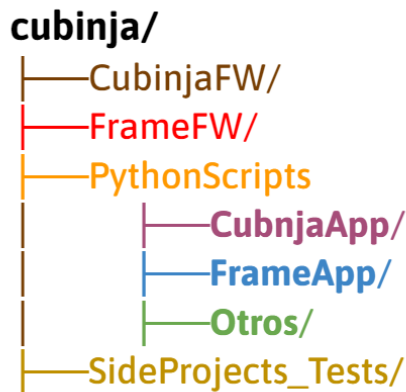
Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Manual de usuario

En este anexo se presenta una breve descripción de como poner en marcha y modificar al sistema, tanto para compilar y flashear un nuevo firmware como para la aplicación remota.

El repositorio del Cubinja tiene la siguiente arquitectura de carpetas



En las carpetas CubinjaFW y FrameFW se encuentra el firmware principal de cada sistema. Para abrir cada uno de los proyectos dentro de VS Code debe hacerse clic en la pestaña *file* y en *open folder* y seleccionar una de estas carpetas. No es necesario hacer ninguna configuración extra en VS Code, la extensión Espressif IDF reconoce instantáneamente cuando se encuentra en una carpeta con el proyecto de cada sistema.

La carpeta PythonScripts contiene 3 subcarpetas: CubinjaApp, FrameApp y otros. En CubinjaApp y FrameApp se encuentran las aplicaciones principales de Python para la comunicación remota con los sistemas. En otros se encuentran otros scripts con pruebas intermedias que se realizaron en el proceso.

La carpeta SideProjects_Tests tiene otros Firmwares que fueron realizados en el proceso para testeo de funcionalidades en particular.

C.1. CubinjaFW y FrameFW

Se dará la guía para ambos FW en la misma sección ya que se buscó que ambos proyectos en su diseño fueran prácticamente análogos.

Ambos proyectos cuentan con un archivo de configuración llamados *CubinjaConfig.h* y *FrameConfig.h* respectivamente.

En *CubinjaConfig.h* podemos editar los siguientes parámetros del sistema:

- Frecuencia de muestreo de sensores y acción de control
- Activar o desactivar modo debug para poder ver los logs del ESP al estar conectado via USB.
- Activar o desactivar lectura de batería para poder utilizar el sistema con la batería desconectada.
- Activar o desactivar debuggeo de mensajes de transmisión BLE
- Configuración de sampleo de voltaje de batería
- Configuración de parametros iniciales de controladores

En el caso de *FrameConfig.h* es muy similar pero con menos opciones de configuración:

- Frecuencia de muestreo de sensores y acción de control
- Activar o desactivar debuggeo de mensajes de transmisión BLE
- Configuración de parámetros iniciales de controladores

En cuanto a las configuraciones del hardware ambos proyectos tienen un archivo *PinMap.h* para modificar la configuración de pines de cada sistema.

Para la modificación del lazo de control se debe acceder a *ControllerStateMachine.h*, dentro de este archivo tenemos los métodos *CtrlAction*. Estos métodos son los que se corren al recibir un nuevo dato de parte de la tarea *taskDataGatherer*, en el caso del Cubinja tenemos 4 métodos distintos, uno para cada punto de equilibrio y en el caso del Frame uno solo.

De la figura C.1 vemos que este método recibe como parámetro un puntero a void llamado *data*, en este dato se pasa como parámetro el vector de estado del sistema para ejecutar la acción de control.

Finalmente para la compilar y flashear el firmware podemos hacerlo también directo desde VS Code utilizando la barra de tareas brindada por la extensión de Espressif, la misma se puede ver en la figura C.2

En esta barra de herramientas se encuentran las siguientes funcionalidades

1. Selección de puerto de comunicación para flasheo y monitoreo de ESP32, asegurarse que el puerto COM seleccionado sea el correspondiente a la ESP conectada. Distintos ESP pueden tener distintos puertos COM
2. Compilación del proyecto


```

//----- Accion de control para AristaA -----
IPC_protocol::Mtr_message_payload_t ControllerStateMachine::AristaA::CtrlAction(void* data){
char tag[] = "[AristaAST]";
IPC_protocol::Mtr_message_payload_t mMtrMessage;
Controller::State_Vector_t* mStateVector = (Controller::State_Vector_t*)data;

mMtrMessage.MtrAVolt = (FeedbackParams.KTheeta) * (TheetaData.getError(mStateVector->AngleData.Theeta))
+ (FeedbackParams.KTheetap) * (TheetapData.getError(mStateVector->AngleData.Theetap))
+ (FeedbackParams.KPhi) * (PhiData.getError(mStateVector->AngleData.Phi))
+ (FeedbackParams.KPhip) * (PhiData.getError(mStateVector->AngleData.Phip))
+ (FeedbackParams.KMtrA) * (MtrAVelData.getError(mStateVector->MtrVel.MtrAVel))
+ (FeedbackParams.KMtrB) * (MtrAVelData.getError(mStateVector->MtrVel.MtrBVel))
+ (FeedbackParams.KMtrC) * (MtrAVelData.getError(mStateVector->MtrVel.MtrCVel));

mMtrMessage.MtrBVolt = 0;
mMtrMessage.MtrCVolt = 0;
ESP_LOGD(tag, "MtrAVolt = %f MtrBVolt = %f MtrCVolt = %f", mMtrMessage.MtrAVolt, mMtrMessage.MtrBVolt, mMtrMessage.MtrCVolt);

return mMtrMessage;
}

```

Figura C.1: Captura del método *CtrlAction* para la *AristaA* en *ControllerStateMachine.h* del *CubinjaFW*



Figura C.2: Barra de tareas de Espressif IDF para el control y flasheo del ESP32

3. Flasheo del proyecto
4. Monitoreo del log del firmware

Por lo tanto para cargar y monitorear un programa al ESP se seleccionan todos esos ítem en ese orden, primero se chequea que se este tabajando con el puerto COM correcto, luego se compila, finalizada la compilación del proyecto se flashea y opcionalmente se monitorea el programa cargado.

Es importante tener las siguientes consideraciones antes de flashear:

1. En el caso del *Cubinja* mantener el sistema apagado antes de flashear, esto es debido a que el development kit del ESP32 produce señales PWM en algunos de sus pines mientras se da el proceso de flasheo pudiendo llegar a activar alguno de los motores si el sistema no se mantiene apagado.
2. Mantener apretado el botón de boot al flashear, el ESP controla que uno de sus pines esté en nivel bajo al momento de bootear, este botón conecta dicho pin directo a tierra permitiendo que el ESP pueda entrar en modo boot siempre.

C.2. Aplicación remota

Para hacer uso de la aplicación remota abrir en VS Code *MainApp.py* en la carpeta *CubinjaApp* o *MainAppFrame.py* en la carpeta *FrameApp*.

Para ejecutar la aplicación hacer clic en *run code*, esta opción se encuentra en la esquina superior derecha como se muestra en la figura C.3.

Apéndice C. Manual de usuario

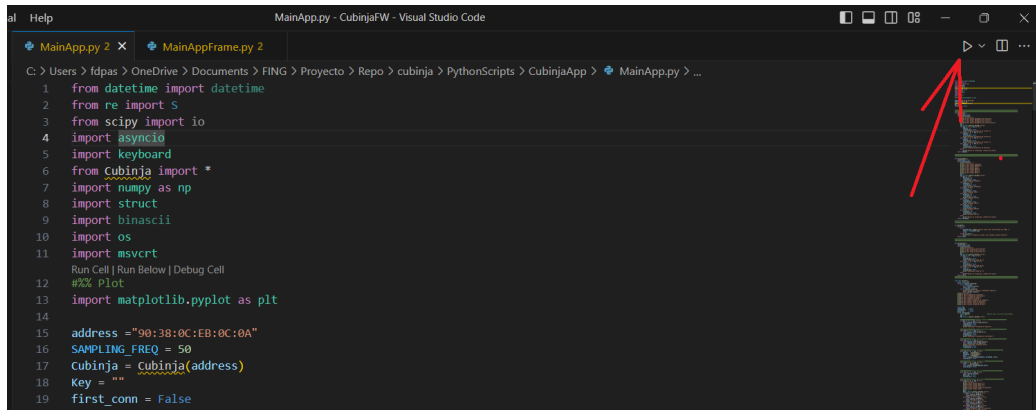


Figura C.3: Captura de pestaña de ejecución del programa principal en PC

Al dar comienzo a cualquiera de las dos aplicaciones se comenzara primero a buscar al ESP32 en la red *Bluetooth* y una vez encontrado y emparejado con el mismo se mostrara la interfaz de usuario de la figura C.4 para poder acceder a las funcionalidades del Cubinja o del Frame.

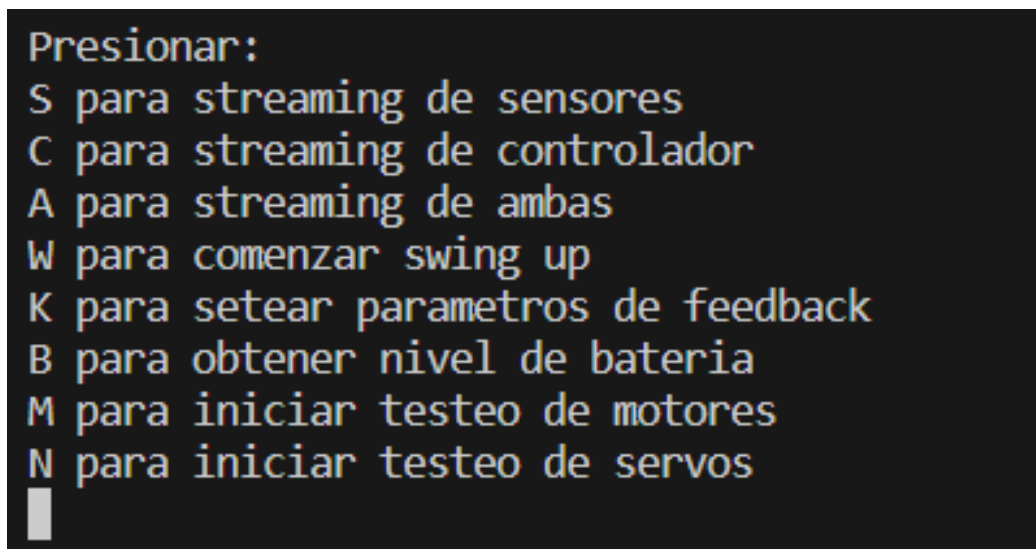


Figura C.4: Interfaz de usuario de la aplicación principal del Cubinja

El único parametro configurable en las aplicaciones son el address del dispositivo y la frecuencia de muestreo del sistema, es importante que esta ultima coincida con la frecuencia de muestreo del Cubinja ya que de lo contrario la representación temporal en las funcionalidades de adquisición y ploteo de datos sera incorrecta.

Para el address del dispositivo por defecto estarán configuradas para los sistemas entregados pero en caso de cambiar el microcontrolador esta dirección debe ser reconfigurada ya que depende de la dirección MAC de cada ESP

En caso de que la address este mal configurada la propia aplicación imprimirá en la terminal los dispositivos hallados en la red junto con su dirección como se

C.2. Aplicación remota

muestra en la figura C.5, la dirección encontrada debe coincidir con la configurada en la app.

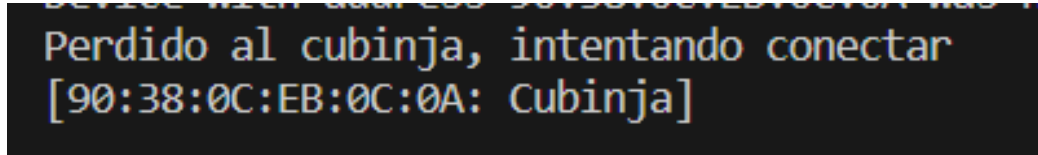


Figura C.5: Captura de la terminal de la aplicación, en la misma se puede ver que se encontró a un dispositivo con nombre Cubinja y dirección 90:38:0C:EB:0C:0A

C.2.1. Debuggeo a través de celular y LightBlue

Antes de la aplicación de computadora en *Python* primero se utilizó la aplicación de celular *LightBlue* para el testeo de la comunicación con el ESP32 y la correcta codificación y decodificación de los mensajes del protocolo BLE.



Figura C.6: Logo de la aplicación LightBlue

LightBlue es una aplicación que proporciona una interfaz intuitiva y fácil de usar para controlar y gestionar dispositivos Bluetooth desde un dispositivo móvil. Está diseñada para simplificar la conexión y la interacción con dispositivos Bluetooth, permitiéndole al usuario comunicarse con estos de manera eficiente.

Para el desarrollo del Cubinja resultó de gran utilidad debido a la facilidad para enviar y recibir datos a la hora de configurar distintos componentes. En particular, se utilizó esta conexión a la hora de testear y calibrar el sistema de frenado y *swing up*. Por medio de un código escrito en base hexadecimal se le indica al sistema la posición de descanso y frenado del servo motor, así como también la velocidad del BLDC y el tiempo en el que debe transcurrir el frenado.

Apéndice C. Manual de usuario

Estas funcionalidades antes mencionadas luego fueron abarcadas y sustituidas por la aplicación desarrollada en python dado que esta facilita en mayor medida la comunicación con el usuario.

Apéndice D

Protocolos de comunicación Bluetooth

En este anexo se detallan los protocolos de comunicación del sistema y los diferentes mensajes que pueden ser enviados o recibidos vía Bluetooth o para la comunicación interna entre los diferentes procesos.

Como se menciona en el capítulo 8 los mensajes BLE tienen la forma de la figura D.1

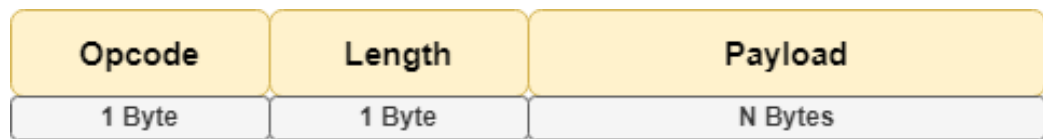


Figura D.1: Protocolo de comunicación BLE

Los mensajes pertenecientes a el protocolo de comunicación son los siguientes:

Apéndice D. Protocolos de comunicación Bluetooth

Nombre	Opcode (HEX)	Length (Bytes)	Descripción
SensorsPacket	0x00	28	Paquete de datos de sensores, su payload es el vector de estado del sistema.
CtrlPacket	0x01	13	Paquete de datos de control, su payload es el voltaje impuesto sobre cada motor y el estado de sub estado de equilibrio
StartSensorsStream	0x10	0	Mensaje que inicializa el stream de datos de sensores
StopSensorsStream	0x11	0	Mensaje que detiene el stream de datos de sensores
PIDParametersSet	0x41	6	Mensaje utilizado para configurar las constantes de realimentación del sistema
PIDParametersGet	0x42	6	Mensaje utilizado para leer las constantes de realimentación del sistema
StartSwingUp	0x50	3	Mensaje que comienza el protocolo de swing up del Cubinja, en su payload se encuentra sobre que volante y a que velocidad se aplica el frenado
ServoTest	0x52	1	Mensaje que inicia el testeo de un servo motor, en su payload se selecciona cual
BattLvl	0x60	4	Mensaje utilizado para comunicar el nivel de batería del sistema
MtrTest	0x70	1	Mensaje utilizado para iniciar el testeo de un motor, en el payload se especifica cual

Tabla D.1: Tabla de opciones de mensajes en el protocolo BLE, se muestra el opcode utilizado para decodificar cada mensaje junto con el largo en bytes del mismo y una breve descripción de la funcionalidad de cada uno de estos mensajes

Apéndice E. Gestión de proyecto

de ambos dos (1D y 3D). Se lo protegió contra golpes bruscos. La rotura de piezas plásticas no implica un riesgo mayor debido a que todas ellas pueden ser reimpresas. En cuanto a los componentes, se cuenta con repuestos para todos ellos.

La inestabilidad de costos no resultó ser un problema en el desarrollo del proyecto, los componentes y materiales utilizados son productos estándar en el mercado local e internacional y mantienen un precio estable. El manejo de importaciones no resultó ser un problema dado a los bajos costos de materiales importados. Además estos no contienen materiales con manejo especial ni tienen un peso excesivo.

La incertidumbre en el modelado resultó un punto de discusión a lo largo del desarrollo del proyecto. En última instancia se alcanzaron resultados que prueban la validez del modelado. Aplicando el método de LQR al control por realimentación de estados utilizando los datos del modelado se pudo lograr el equilibrio del prototipo 1D con los valores obtenidos, sin ajustes.



Figura E.2: Análisis de riesgos

Como pieza central de este análisis surge que nunca se tuvo en cuenta la posibilidad de baja de un integrante. Esto ocurrió y la totalidad del proyecto debió ser reestructurado. A pesar de perder un tercio de la mano de obra en la mitad del transcurso del proyecto con la ayuda de los tutores se resolvió ajustar los objetivos y se pudo continuar de buena manera, alcanzando buenos resultados.

E.3. Planificación y dedicación horaria

A lo largo del proyecto se fue modificando la planificación y plazos, este proceso y evolución se puede ver en las figuras E.3, E.4, E.5 y E.6

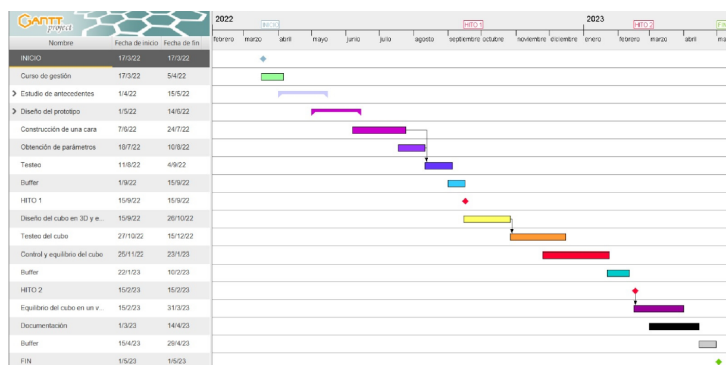


Figura E.3: Diagrama de gantt original del proyecto.

E.3. Planificación y dedicación horaria

En la figura E.3 se puede ver el diagrama de gantt original entregado en el plan de proyecto correspondiente a la tarea cinco del curso de gestión del proyecto.

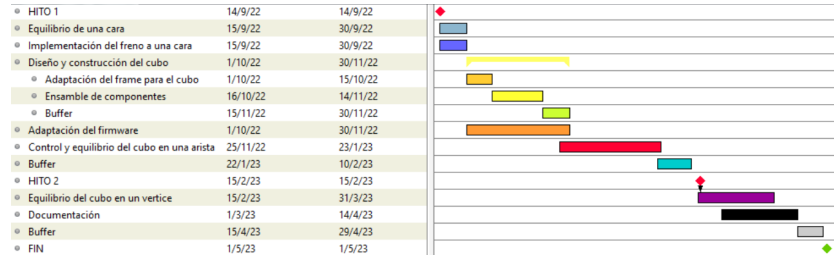


Figura E.4: Diagrama de gantt realizado de cara al hito 2 del proyecto.

En la figura E.4 se puede ver el diagrama de gantt una vez realizado el hito 1 del proyecto. Se logró cumplir con los tiempos y tareas establecidas de cara al hito 1. Se agrega tarea de implementación del sistema de frenado.

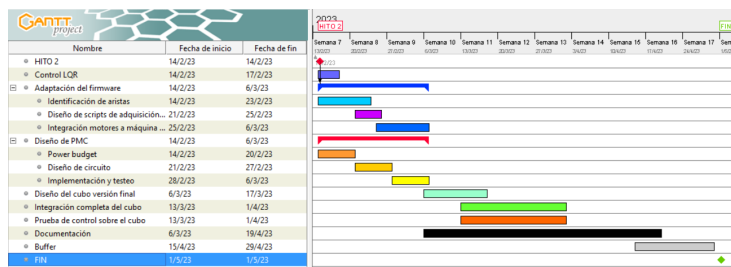


Figura E.5: Diagrama de gantt realizado de cara a la entrega final del proyecto. En este punto el grupo de trabajo ya se conforma por dos estudiantes.

En la figura E.5 se puede ver el diagrama de gantt una vez presentado el hito 2, ya contando con solo dos estudiantes en el grupo. Los cambios principales con respecto al plan original consisten en asegurar la entrega de un producto final completo, tanto en hardware como software, que permita trabajar con el sistema a futuros estudiantes. En vista de la perdida de mano de obra se deja como extra y no como requerimiento el equilibrio del cubo.

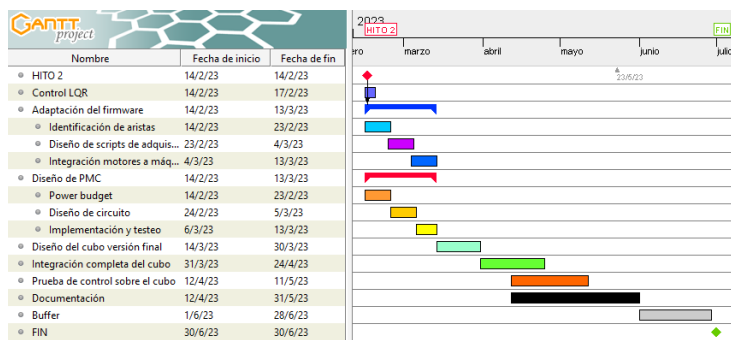


Figura E.6: Diagrama de gantt realizado de cara a la petición de prorroga para la entrega.

Apéndice E. Gestión de proyecto

En la figura E.6 se puede ver el diagrama de gantt de cara a la fecha de finalización original del proyecto. Proyectando los tiempos que tomaron las tareas trabajadas se tomó la decisión de solicitar un tiempo de prórroga de dos meses, totalizando el proyecto para fines de junio.

Apéndice F

Hojas de datos

F.1. Microcontroladores

Los datos relacionados al microcontrolador ESP32 son accesibles en la siguiente bibliografía [14].

Los datos relacionados al *development kit* del microcontrolador ESP32 son accesibles en la siguiente bibliografía [26].

Los datos relacionados al microcontrolador nRF52832 son accesibles en la siguiente bibliografía [13].

Los datos relacionados al microcontrolador STM32F4 son accesibles en la siguiente bibliografía [13].

F.2. Sensores y Actuadores

Los datos relacionados a la Unidad de Medición Inercial MPU-9250 son accesibles en la siguiente bibliografía [29].

Los datos relacionados al servo motor MG90S son accesibles en la siguiente bibliografía [30].

Los datos relacionados al servo motor SCS0009 son accesibles en la siguiente bibliografía [31].

Los datos relacionados al servo motor TS90 son accesibles en la siguiente bibliografía [32].

Los datos relacionados al motor BLDC Nidec 24H404H son accesibles en la siguiente bibliografía [33].

Apéndice F. Hojas de datos

Los datos relacionados al motor BLDC Maxon EC 45 Flat son accesibles en la siguiente bibliografía [34].

F.3. Potencia

Los datos relacionados al convertidor de voltaje DC-DC LM2596 de tipo *step-down* son accesibles en la siguiente bibliografía [28].

Referencias

- [1] M. Gajamohan, “The cloud, paper planes, and the cube,” *ETH Zurich library*, 2014. [En línea]. Disponible en: <https://www.research-collection.ethz.ch/handle/20.500.11850/92139>
- [2] M. Gajamohan, M. Merz, I. Thommen, and R. D’Andrea, “The cubli: A cube that can jump up and balance,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3722–3727.
- [3] McKern, Brown, Dove, Gilmore, Landey, Musoff, Amand, and Vincent, “Space shuttle avionics a redundant imu on-board checkout and redundancy management system,” *DSR Project 55-46700, by the Marshall Spaceflight Center of the National Aeronautics and Space Administration (NASA) through Contract NAS 8-27624*, 1972.
- [4] A. Bellati, F. Cancela, and N. Pérez, “Construcción y control del péndulo de furuta,” *Tesis de grado, IIE, Facultad de Ingeniería, Universidad de la República, Uruguay*, p. 175, 2021. [En línea]. Disponible en: <https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/26548?mode=full>
- [5] J. K. Roberge, “The mechanical seal,” *Bachelor’s thesis, Massachusetts Institute of Technology, Cambridge*, 1960.
- [6] D. T. Higdon and R. H. Cannon, “On the control of unstable multiple-output mechanical systems,” *ASME Publication 63-WA-148, American Society of Mechanical Engineers, New York*, 1963.
- [7] K. Ogata, “Modern control engineering,” *Prentice-Hall, Englewood Cliffs*, pp. 277–279, 1970.
- [8] K. H. Lundberg and T. W. Barton, “History of inverted-pendulum systems,” *Massachusetts Institute of Technology, Cambridge, Mass. 02139*, pp. 277–279, 2009. [En línea]. Disponible en: <http://web.mit.edu/klund/www/keeling/Pendulum2009.pdf>
- [9] E. BJERKE and B. PEHRSSON, “Development of a nonlinear mechatronic cube,” *Department of Signal and Systems Division of Automatic control, Automation and Mechatronics CHALMERS UNIVERSITY OF TECHNOLOGY Gothenburg, Sweden*, 2016.

Referencias

- [10] J. S. Knudsen, L. Røjkjær-Jensen, and M. J. Riisager, “Cubli - self balancing cube,” *Aalborg University Department of Electronic Systems Fredrik Bajers Vej 7B DK-9220 Aalborg*, 2020.
- [11] R. RC, “Rem-rc - self balancing cube,” 2022.
- [12] *STM32F405xx STM32F407xx Datasheet*, STMicroelectronics, 2020, dS8626 Rev 9. [En línea]. Disponible en: <https://www.st.com/resource/en/datasheet/stm32f405rg.pdf>
- [13] *nRF52832 Datasheet*, Nordic Semiconductors, 2023, version 2.4. [En línea]. Disponible en: <https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52832-product-brief.pdf>
- [14] *ESP32 Series Datasheet*, Espressif Systems, 2023, version 4.2. [En línea]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- [15] S. Harris, “Inertial measurement unit, an introduction,” *Article for Advanced Navigation*, 2023. [En línea]. Disponible en: <https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/>
- [16] FlyRobo, “Interfacing mpu-9250 9-dof sensor with arduino,” *FlyRobo Blog*, 2023. [En línea]. Disponible en: <https://www.flyrobo.in/blog/mpu9250-arduino>
- [17] P. Lab. Complementary filter - sensor fusion 2 - phil’s lab 34. Youtube. [En línea]. Disponible en: https://www.youtube.com/watch?v=BUW2OdAtzBw&t=215s&ab_channel=Phil%E2%80%99sLab
- [18] C. Team, “Exponentially weighted moving average (ewma) a quantitative or statistical measure used to model or describe a time series,” *Article for CFI*, 2021. [En línea]. Disponible en: <https://corporatefinanceinstitute.com/resources/capital-markets/exponentially-weighted-moving-average-ewma/>
- [19] C. Lane. Exponentially weighted moving average or exponential weighted average — deep learning. Youtube. [En línea]. Disponible en: https://www.youtube.com/watch?v=XV1f_srZg_E&ab_channel=CodingLane
- [20] J. A. C. Álvarez, “Diseño de un robot autobalanceado,” *Universidad Tecnológica de la Habana*, 2019. [En línea]. Disponible en: https://www.researchgate.net/publication/348993028_Disenio_de_un_Robot_Autobalanceado
- [21] A. D. S.-G. y Javier Perez-Holguin, “Modulo ip basado en fpga para la decodificación de encoders de cuadratura,” *Universidad Pedagógica y Tecnológica de Colombia Escuela de Ingeniería Electrónica (Sogamoso)*, 2020. [En línea]. Disponible en: https://www.researchgate.net/publication/348374766_Modulo_IP_basado_en_FPGA_para_la_decodificacion_de_encoders_de_cuadratura

- [22] F. E.F. and M. M.A.S., “Block diagrams of electromechanical systems,” *Power Conversion of Renewable Energy Systems*. Springer, Boston, MA, 2011.
- [23] M. Amado, G. Fisher, and K. Sosa, “Termodron iii : Dron autónomo de reconocimiento por termografía,” *Tesis de grado. Universidad de la República (Uruguay). Facultad de Ingeniería.*, 2021. [En línea]. Disponible en: <https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/26959?mode=full>
- [24] P. B. Shoshany, “Moments of inertia, physics department, brock university, st. catharines, ontario,” 1996. [En línea]. Disponible en: <https://www.physics.brocku.ca/fun/NEWT3D/PDF/MOMINERT.PDF>
- [25] R. Tedrake, *Underactuated Robotics*, 2023. [En línea]. Disponible en: <https://underactuated.csail.mit.edu>
- [26] *ESP32-WROVER-E ESP32-WROVER-IE Datasheet*, Espressif Systems, 2023, version 1.8. [En línea]. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf
- [27] K. Marasco, “How to successfully apply low-dropout regulators,” *Analog Dialogue*, 2009. [En línea]. Disponible en: <https://www.analog.com/en/analog-dialogue/articles/applying-low-dropout-regulators.html>
- [28] *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator datasheet (Rev. G)*, Texas Instruments, Noviembre 1999, revisado marzo 2023, sNVS124G. [En línea]. Disponible en: <https://www.ti.com/lit/ds/snvs124g/snvs124g.pdf?ts=1689010427166>
- [29] *MPU-9250 Product Specification*, InvenSense Inc., 2016, revision 1.1. [En línea]. Disponible en: <https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>
- [30] *MG90S Metal Gear Servo Datasheet*, Tower PRO. [En línea]. Disponible en: https://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf
- [31] *SCS0009 Datasheet*, Feetech. [En línea]. Disponible en: <https://abra-electronics.com/electromechanical/motors/servo-motors-feetech/scs0009.html>
- [32] *TS/GS90 Datasheet*, Tower PRO. [En línea]. Disponible en: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
- [33] *CMC 24H series*, Nidec. [En línea]. Disponible en: <https://www.nidec.com/en/product/search/category/B101/M102/S100/NCJ-24H-24-01/>
- [34] *EC 45 flat Datasheet*, Maxon, 2019. [En línea]. Disponible en: https://www.maxongroup.com/medias/sys_master/root/8833813184542/19-EN-264.pdf

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

4.1. Parámetros y variables del sistema	37
4.2. Resultados obtenidos del ensayo de medición de corriente a distintas velocidades.	42
4.3. Resumen de parámetros que caracterizan al volante de inercia . . .	42
4.4. Resumen de parámetros que caracterizan al volante de inercia. . .	46
9.1. Resultados de test del filtro complementario, se muestra el estimador obtenido contra una posición de referencia	84
9.2. Resultado del ensayo sobre el encoder utilizando el método de tiempo fijo	85
9.3. Resultado de ensayos de frenado para el swing up	89
9.4. Resultado de ensayos sobre encoders y motores. Nuevamente la velocidad estimada se obtiene a partir del monitoreo de los encoders con un osciloscopio.	91
9.5. Ángulos que muestran la posición de la IMU en cada punto de equilibrio	92
9.6. Resultados de ensayos de medición de batería	96
D.1. Tabla de opciones de mensajes en el protocolo BLE, se muestra el opcode utilizado para decodificar cada mensaje junto con el largo en bytes del mismo y una breve descripción de la funcionalidad de cada uno de estos mensajes	122

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

1.1. Prototipo del Cubli, extraída de video presentación de Youtube del proyecto The Cubli [2].	2
2.1. Sistema 1D.	10
2.2. Diagrama de interfaces del sistema 1D.	11
2.3. Sistema 3D.	11
2.4. Diagrama de interfaces del sistema 3D.	12
2.5. Diagrama de bloques del sistema 3D.	13
2.6. Tabla comparativa de los microcontroladores de <i>STM</i> , <i>Espressif</i> y <i>Nordic Semiconductors</i> . Hojas de datos en apendice F.	14
2.7. Foto del Development Kit del ESP32	14
3.1. Diagrama de los ejes de medición para un acelerómetro y un giroscopio. Imagen tomada de [15].	18
3.2. Development kit del MPU9250 utilizado en el Cubinja. Imagen tomada de [16]. Hoja de datos en F.	19
3.3. Diagrama que muestra la posición de los acelerómetros relativos al frame, los sistemas de coordenadas $b\vec{e}_i$ representan los sistemas de referencia de cada IMU. Diagrama sacado de [2].	20
3.4. Diagrama que muestra el manejo de datos del acelerómetro y giroscopio para el método del filtro complementario. Notar que en la integral realizada se utiliza $\hat{\theta}$ y $\hat{\phi}$ y no $\hat{\theta}_{gyr}$ y $\hat{\phi}_{gyr}$	22
3.5. Diagrama en el que se muestran los componentes principales de un encoder óptico. Imagen sacada de [20].	24
3.6. Diagrama en el que se muestra el patrón de señal en un encoder en cuadratura, puede verse que las señales de salida de los foto receptores están desfasadas un cuarto de periodo. Figura tomada de [21].	25
3.7. Diagramas de motor DC, extraído de [10].	28
3.8. Cuadro comparativo de características de motores BLDC. Hoja de datos en F.	31
3.9. Cuadro comparativo de características de servo motores. Hoja de datos en F.	32
4.1. Sistema frame. Rulemán restringe el movimiento en el plano con mínimo rozamiento.	35

Índice de figuras

4.2.	Diagrama completo del sistema frame + volante.	36
4.3.	Gráfica de datos obtenidos en el ensayo de corriente con la recta que mejor se aproxima superpuesta.	43
4.4.	Foto del frame dado vuelta. Sobre este <i>setup</i> se realizó el ensayo para calcular el momento de inercia del frame y el rozamiento en el rulemán	44
4.5.	Gráfico que muestra el ángulo en función del tiempo del ensayo realizado contra el modelo teórico con los parámetros hallados. . .	45
4.6.	Diagrama que muestra la estructura de un controlador de estado completo.	46
5.1.	Sistema de frenado por bloqueo del volante. El servo motor desplaza su brazo hasta trabar el giro del volante al entrar en contacto con los tornillos de la cara externa del volante.	51
5.2.	Sistema de frenado por tensado de banda. El servo desplaza su brazo y tensa la banda para frenar el volante.	53
5.3.	Sistema de frenado por zapata implementado. El servo desplaza la zapata y esta frena el volante.	54
5.4.	Detalle de soporte de servo motor de frenado.	55
5.5.	Zapata de frenado.	55
5.6.	Volante de inercia con pista exterior de frenado.	55
6.1.	Diagrama que muestra los tres niveles de tensión requeridos por el sistema junto con los correspondientes componentes alimentados y su respectivo consumo.	58
6.2.	Power Budget sin consumos de corriente para topología alimentada por una sola celda de 3V7.	59
6.3.	Diagrama de bloques de PMC para topología de una sola celda utilizando un regulador boost para cada motor	61
6.4.	Power budget del Cubinja, todos los pasajes de corriente aguas arriba se asumen con 100 % de eficiencia	62
6.5.	Foto de la batería seleccionada para el Cubinja.	63
6.6.	Foto del DK del regulador LM2596 (hoja de datos en F).	64
6.7.	Componentes.	65
6.8.	Niveles de tensión del circuito.	65
7.1.	Vista completo de componentes internos del cubo.	68
7.2.	Volante de inercia, con peso agregado.	68
7.3.	Ensamblado de volante de inercia.	69
7.4.	Frame con conexiones y leds.	70
7.5.	Frame con motor y volante.	70
7.6.	Frame simple.	70
7.7.	Esquinero para refuerzo de unión de caras.	70
7.8.	Soporte de batería, placa e IMU.	71
7.9.	Vistas del circuito.	71
7.10.	Layout de borneras y placas.	72

7.11. Conexión interna de motores A, B y C con conector DB25.	72
7.12. Diagrama de colores y funcionalidades de motores BLDC.	72
8.1. Diagrama de bloques del firmware diseñado, separado en cuatro niveles de abstracción diferentes según su jerarquía. En el nivel mas bajo se encuentra el hardware, englobando a todos los componentes del cubinja. Luego se encuentra la HAL del sistema utilizada luego mas arriba por todos los active objects para poder interactuar con cada componente del hardware. Finalmente en la sección superior se encuentra Alfred, la sección de código mas abstraída del hardware que cumple el rol de IPC, siendo el encargado de la comunicación entre todos los <i>active objects</i>	75
8.2. Diagrama que muestra como al inicializar una tarea nueva esta adjunta todos sus handlers a Alfred y este arma una lista con los handlers para cada tarea.	77
8.3. Diagrama de estados del sistema, se muestran los 4 estados posibles del sistema y la transición entre ellos. La única entrada de esta maquina de estados son los estimadores de la posición angular del mismo.	78
8.4. Diagrama que muestra el flujo de datos en el sistema en la acción de control, comenzando por taskDataGatherer, siguiendo a taskCtrl y finalizando en TaskMtr.	79
8.5. Diagrama que muestra la estructura de cada mensaje BLE recibido y transmitido en el sistema Cubinja.	81
9.1. En la gráfica de arriba podemos ver la variación del ángulo a lo largo del tiempo y en la de abajo la velocidad.	84
9.2. Gráfica de derivada del estimador angular vs salida del EMWA, ambos medidos en grados por segundo. En azul la derivada de la posición angular y en naranja los datos del giroscopio pasados por el EMWA.	85
9.3. Posición y velocidad angular en función del tiempo con el controlador encendido	86
9.4. Gráfica del duty cycle de la PWM impuesta sobre el motor en función del tiempo. Se puede observar que en algunos puntos se solicita un duty cycle incluso mayor a 100 %. Esto significa que se solicita un voltaje mayor incluso al que se puede imponer, para estos valores se impone un duty cycle de 100 %.	87
9.5. Gráfica obtenida desde la aplicación BLE en la que se ve la variación de la velocidad de los tres motores del Cubinja en función del tiempo.	90
9.6. Diagrama representativo de los ángulos de interes del sistema cubo. Estos son el pitch y el roll del cubo	91
9.7. Gráficas donde se observan la velocidad de los motores junto a los ángulos θ y ϕ . Puede verse cómo en $\theta \approx 2^\circ$ y $\phi \approx -35^\circ$, los motores B y C permanecen quietos mientras el Motor A alcanza una velocidad de casi 1000 RPM.	92

Índice de figuras

9.8.	Gráficas donde se observan la velocidad de los motores junto a los ángulos θ y ϕ . Puede verse cómo en $\theta \approx 0$ y $\phi \approx 0$, todos los motores alcanzan una velocidad de casi 1000 RPM.	93
9.9.	Gráfico del ensayo de control sobre θ . Se puede ver cómo al aumentar la diferencia con respecto a su posición de equilibrio, cada vez es mayor el voltaje impuesto al motor.	94
9.10.	Gráfico del ensayo de control sobre $\dot{\theta}$. Se puede ver cómo a mayor velocidad, mayor el voltaje impuesto sobre el motor.	95
9.11.	Gráfico del ensayo de control sobre ω_f . Se observa cómo al aumentar la velocidad del volante, el sistema establece un voltaje que se opone a dicho movimiento.	96
A.1.	Prototipo del frame completo.	104
A.2.	Cubo 3D.	105
A.3.	Cubo 3D.	109
A.4.	Cableado del conector DB25 con los BLDC A, B y C.	109
A.5.	Mapa de pines del cubo.	110
C.1.	Captura del método <i>CtrlAction</i> para la <i>AristaA</i> en <i>ControllerStateMachine.h</i> del <i>CubinjaFW</i>	117
C.2.	Barra de tareas de Espressif IDF para el control y flasheado del ESP32	117
C.3.	Captura de pestaña de ejecución del programa principal en PC	118
C.4.	Interfaz de usuario de la aplicación principal del <i>Cubinja</i>	118
C.5.	Captura de la terminal de la aplicación, en la misma se puede ver que se encontró a un dispositivo con nombre <i>Cubinja</i> y dirección 90:38:0C:EB:0C:0A	119
C.6.	Logo de la aplicación <i>LightBlue</i>	119
D.1.	Protocolo de comunicación BLE	121
E.1.	Gastos del proyecto	123
E.2.	Análisis de riesgos	124
E.3.	Diagrama de gantt original del proyecto.	124
E.4.	Diagrama de gantt realizado de cara al hito 2 del proyecto.	125
E.5.	Diagrama de gantt realizado de cara a la entrega final del proyecto. En este punto el grupo de trabajo ya se conforma por dos estudiantes.	125
E.6.	Diagrama de gantt realizado de cara a la petición de prórroga para la entrega.	125

Esta es la última página.
Compilado el jueves 14 septiembre, 2023.
<http://ie.fing.edu.uy/>