COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO DE SISTEMA DE CONTROL DE PÉNDULO INVERTIDO ACCIONADO MEDIANTE VOLANTE DE INERCIA

Autor: Ignacio Martín de Bustamante González-Iglesias

Director: Juan Luis Zamora Macho

Madrid

Agosto de 2019

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

DISEÑO E IMPLEMETACIÓN DE UN CONTROL 3D PARA UN CUBO MEDIANTE VOLANTES DE INERCIA

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018-2019 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Ignacio Martín de Bustamante Fecha: 27/08/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Fecha: 28 / 8 / 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Ignacio Martín de Bustamante González-Iglesias

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:

DISEÑO E IMPLEMETACIÓN DE UN CONTROL 3D PARA UN CUBO MEDIANTE VOLANTES DE INERCIA, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que

pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 27 de Agosto de 2019

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

DISEÑO DE SISTEMA DE CONTROL DE PÉNDULO INVERTIDO ACCIONADO MEDIANTE VOLANTE DE INERCIA

Autor: Ignacio Martín de Bustamante González-Iglesias

Director: Juan Luis Zamora Macho

Madrid

Agosto de 2019

RESUMEN

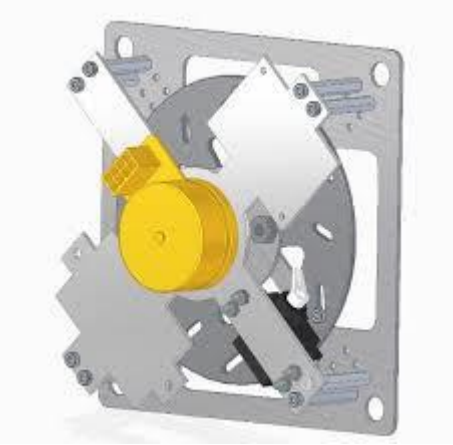
INTRODUCCIÓN Y OBJETIVOS

El proyecto se centra en el diseño de un control en el que acelerando un volante de inercia y frenándolo de golpe, permita a la cara de un cubo levantarse desde el reposo y equilibrarse sobre uno de sus vértices. El control se basa en los principios para controlar los denominados sistemas de péndulo invertido. Mediante unas ligeras modificaciones, este control podrá ser extrapolado a las otras caras del cubo, obteniendo un control 3D para así, alcanzar el fin último del proyecto: "*The Cubli: A Cube that can Jump Up and Balance.*

Los sistemas de péndulo invertido se encuentran englobados dentro de los denominados sistemas subactuados, los cuales, ofrecen un gran ahorro de energía. Esta eficiencia energética se debe a que, utilizando menos actuadores que grados de libertad existentes en el sistema, son capaces de conseguir un control robusto.

PROTOTIPO

La estructura del prototipo está basada en el prototipo de una cara realizado por la universidad ETH de Zurich. Consta de una cara cuadrada de aluminio, fijada por uno de sus vértices a un soporte que le permite girar con libertad sobre el plano vertical, y que tiene acoplada los componentes necesarios para llevar a cabo el control de sistema: motor, volante de inercia, sensores, servo y microcontrolador.



El microcontrolador elegido es una Raspberry Pi Zero W. La elección se debe a la compatibilidad con el entorno Matlab/Simulink con el que se tenía pensado trabajar, su ligero peso, pequeño tamaño y gran variedad de métodos de comunicación posibles. El motor es un Maxon EC45 FLAT Brushless de 50W y el volante de inercia un disco de mini moto. El resto de los componentes son dos sensores IMU para obtener las velocidades y aceleraciones del prototipo y un ADC para convertir la señal analógica procedente de los sensores hall del motor

a digital y así conocer la velocidad del motor y volante de inercia. La comunicación con los sensores y ADC se realiza mediante protocolo I2C.

MODELADO

Apoyándose en la mecánica lagrangiana, las ecuaciones de Euler-Lagrange y el método de los momentos generalizados, se obtienen las ecuaciones no lineales que rigen el sistema. El sistema queda definido por las siguientes variables de estado:

- θ_b = Ángulo de la diagonal del péndulo respecto a la vertical (perpendicular a la base).
- $\dot{\theta}_b$ = Velocidad angular del péndulo, a partir de ahora se denominará ω_b.
- $\dot{\theta}_w$ = Velocidad angular del volante de inercia, a partir de ahora se denominará ω_w

Posteriormente, se define el punto de operación del sistema, resultando ser aquel donde las tres variables de estado tienen valor nulo. Por último, se linealizan las ecuaciones no lineales que rigen el sistema en torno al punto de operación.

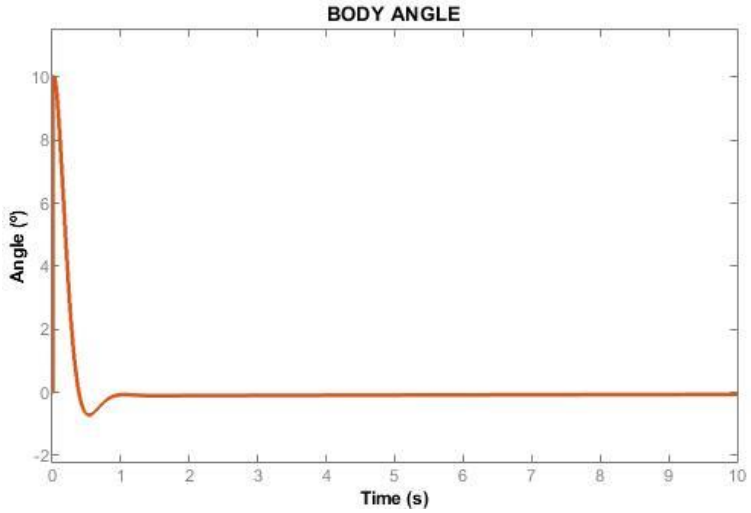
DISEÑO DEL CONTROL

Consiste en un control por realimentación de estados donde el diseño del control se realiza sobre el modelado del sistema. Se emplea la configuración Butterworth de tercer orden, que consiste en colocar dos polos complejos y uno real; todos con el mismo módulo. Dichos polos, tienen que cumplir con las necesidades del sistema. Deben proporcionar tanto un sobrepaso pequeño, como la rapidez suficiente para que no permitan al sistema superar el ángulo límite de recuperación.

Es necesario tener claro que la estimación de las variables de estado es fundamental ya que las salidas solo dependen de la entrada aplicada y de las propias variables de estado. La estimación del ángulo de inclinación es la más complicada. Se obtiene mediante el uso de dos IMU colineales en la diagonal de la cara y a partir de las medidas de sus respectivos acelerómetros del eje X e Y. Las estimaciones de las velocidades son prácticamente directas ya que la velocidad del volante de inercia es proporcionada por los sensores hall del motor y la velocidad del cuerpo por el giróscopo en el eje Z de las IMUs.

IMPLANTACIÓN DEL CONTROL

La implantación del control se realiza con la herramienta Simulink. Antes de realizar la implantación, es necesario probar el control sobre el modelo generado del sistema en simulación. De no hacerlo, el hardware podría sufrir daños. Soltando el cuerpo desde un ángulo de 10^{o} se obtuvo el siguiente resultado de simulación en el control de ángulo:



Debido a las pocas entradas y a que los estados son secuenciales, resulta muy útil a la hora de implantar el control una máquina de estados o de Moore para unir los controles de salto y balanceo. Los estados de la máquina de estados son los siguientes:

1) STOP: Se trata del estado inicial en donde todo está parado.
2) CALIBRATION: Durante este estado se están calibrando los sensores para que en las etapas posteriores funcionen adecuadamente
3) ACCELERATION: Se activa el motor y este empieza a acelerarse hasta alcanzar la velocidad fijada para realizar el salto.
4) BREAK: Al llegar a este estado se activa el servo para frenar el disco de forma brusca y generar el par que eleve la cara.
5) BALANCE: Comienza el control de balanceo que intentará mantener el sistema en el punto de operación.

RESULTADOS Y CONCLUSIONES

El control de balanceo resultó ser deficiente e incapaz de mantener al prototipo en el punto de operación debido a las vibraciones producidas por el volante de inercia. A pesar de ello, se adaptaron todos los componentes necesarios del sistema al prototipo y se mejoró su estabilidad. También, se consiguió la estimación de todas las variables de estado de manera simultánea y se preparó el control de salto para poder llevarse a cabo en futuros desarrollos.

ABSTRACT

INTRODUCTION AND OBJECTIVES

The project focuses on the design of a control that accelerates a flywheel and brakes it, allowing the face of a cube to rise from rest and balance on one of its vertexes. The control is based on the principles for controlling the so-called inverted pendulum systems. By means of slight modifications, this control can be extrapolated to the other faces of the cube, obtaining a 3D control in order to reach the end of the project: "The Cubli: A Cube that can Jump Up and Balance.

The inverted pendulum systems are included in the so-called sub actuated systems, which offers great energy savings. This energy efficiency is because, using fewer actuators than degrees of freedom existing in the system, they can achieve a robust control.

PROTOTYPE

The structure of the prototype is based on the one-face prototype made by the ETH University of Zurich. It consists of an aluminum square face, fixed by one of its vertexes to a support that allows it to rotate freely on the vertical plane, and which has the components necessary to carry out the system control: engine, flywheel, sensors, servo and microcontroller.

The microcontroller chosen is a Raspberry Pi Zero W. The choice is due to the compatibility with the Matlab/Simulink environment with which it was intended to work, its light weight, small size and variety of possible communication methods. The engine is a 50W Maxon EC45 FLAT Brushless and the flywheel is a mini motorbike disc. The rest of the components are two IMU sensors to obtain the speeds, accelerations of the prototype and an ADC to convert the analog signal from the engine hall sensors to digital and thus know the speed of the engine and flywheel. The communication with the sensors and ADC is made by means of protocol I2C

MODELING

Using lagrangian mechanics, the Euler-Lagrange equations and the generalized moments method, we obtain the non-linear equations that govern the system. The system is defined by the following state variables:

- θ_b = Angle of the diagonal of the pendulum with respect to the vertical (perpendicular to the base).
- $\dot{\theta}_b$ = Angular velocity of the pendulum, from now on it will be called ωb.
- $\dot{\theta}_b$ = Angular velocity of the flywheel, from now on it will be called ωw

Subsequently, the operating point of the system is defined, resulting in the one where the three state variables have zero value. Finally, the non-linear equations that govern the system are linearized around the operating point.

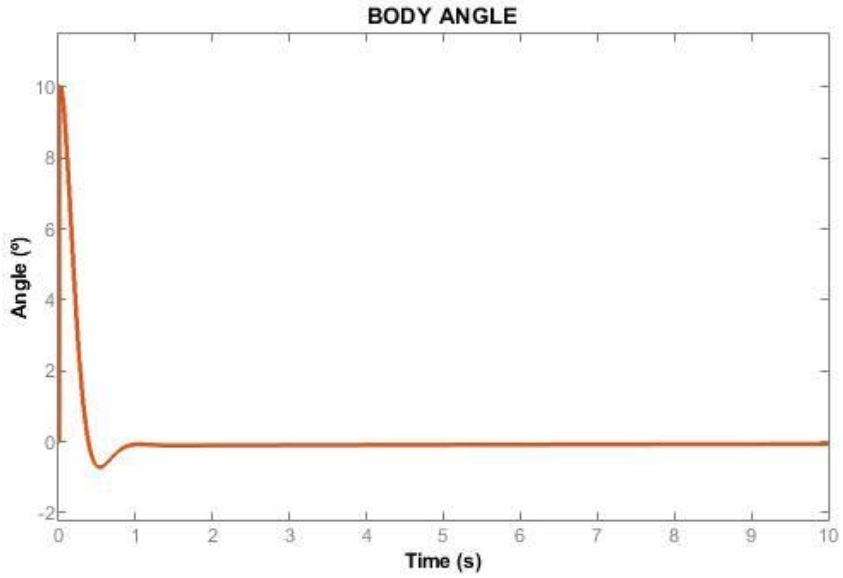
CONTROL DESIGN

It consists of a control by feedback of states where the design of the control is made on the modeling of the system. The third order Butterworth configuration is used, which consists of placing two complex poles and a real one; all with the same module. These poles must meet the needs of the system. They must provide both a small overshoot and enough speed not to allow the system to exceed the recovery limit angle.

It is necessary to be clear that the estimation of the state variables is fundamental since the outputs only depend on the input applied and on the state variables themselves. The estimation of the inclination angle is the most complicated. It is obtained by using two collinear IMUs on the diagonal of the face and from the measurements of their respective accelerometers of the X and Y axis. The estimates of the velocities are practically direct since the speed of the flywheel is provided by the engine hall sensors and the speed of the body by the gyro on the Z axis of the IMUs.

CONTROL IMPLEMENTATION

The control is implemented using the Simulink tool. Before implementation, it is necessary to test the control over the model generated from the simulation system. Failure to do so could result in damage to the hardware. Releasing the body from a 10º angle resulted in the following simulation result in the angle control:



Due to the few inputs and the fact that the states are sequential, it is very useful when implementing a state machine or Moore control to join the jump and roll controls. The states of the state machine are as follows:

1) STOP: This is the initial state where everything is stopped.
2) CALIBRATION: During this state the sensors are being calibrated so that in the later stages they work properly.
3) ACCELERATION: The engine is activated and begins to accelerate until it reaches the speed set to perform the jump.
4) BREAK: When this state is reached, the servo is activated to brake the disc abruptly and generate the torque that raises the face.
5) BALANCE: Begins the roll control that will try to keep the system at the point of operation.

RESULTS AND CONCLUSIONS

The roll control proved to be deficient and unable to maintain the prototype at the point of operation due to vibrations produced by the flywheel. Despite this, all necessary system components were adapted to the prototype and its stability improved. Also, the estimation of all state variables was achieved simultaneously, and the jump control was prepared to be carried out in future developments.

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN 19

1.1 ESTADO DEL ARTE 19

1.1.1 Péndulo invertido sobre carro 19

1.1.2 Péndulo de Furuta 20

1.1.3 Lanzamiento de cohetes 21

1.1.4 Robots bípedos 21

1.1.5 “The Cubli: A Cube that can Jump Up and Balance” 22

1.1.6 “M-Blocks” 22

1.2 MOTIVACIÓN 23

1.3 OBJETIVOS 24

1.4 METODOLOGÍA 24

1.4.1 Tareas 24

1.5 RECURSOS 25

1.5.1 Software 25

1.5.2 Hardware 25

2. PROTOTIPO 26

2.1 Estructura 26

2.2 Microcontrolador 27

2.3 Motor y volante de inercia 28

2.4 Sensores (IMU) 29

2.5 ADC 30

2.6 Alimentación 31

3. COMUNICACIÓN 32

3.1 Protocolo I2C 32

4. MODELADO DEL SISTEMA 34

4.1 Ecuaciones del modelo 34

4.2 Variables de Estado 35
4.3 Punto de operación 36
4.4 Linealización 36
4.5 Identificación de Parámetros 37
4.5.1 Parámetros del motor 37
4.5.2 Distancias y masas 38
4.5.3 Momentos de Inercia 38
5. DISEÑO DEL CONTROL 39
5.1 Estimación de las variables de estado 39
5.1.1 Estimación del ángulo de inclinación 39
5.1.2 Estimación de la velocidad de rotación del péndulo 40
5.1.3 Estimación de la velocidad de rotación del volante de inercia 40
5.2 Control de balanceo 41
5.3 Control de salto 41
6. IMPLANTACIÓN DEL CONTROL 42
6.1 Validación del control 42
6.2 Máquina de estados 43
6.3 Algoritmo de control 45
6.4 Microcontrolador 46
6.5 Hardware 47
6.5.1 Motor 47
6.5.2 IMU 49
6.5.3 ADC 50
7. RESULTADOS 52
7.1 Resultados de la Simulación 52
7.2 Pruebas de los componentes 54
7.2.1 Prueba IMU 55
7.2.2 Prueba MCP3428 56

7.3 Resultado de la implantación 57
7.3.1 Ensayo aceleración 57
7.3.2 Ensayo control de balanceo 58
8. CONCLUSIONES Y RECOMENDACIONES 61
9. ANEXOS 62
9.1 Control Tunning 62
10. Referencias 69

ÍNDICE DE FIGURAS

FIGURA 1: PÉNDULO INVERTIDO SOBRE CARRO 20
FIGURA 2: PÉNDULO DE FURUTA 20
FIGURA 3: CONTROL DIRECCIÓN DE UN COHETE 21
FIGURA 4: ROBOT BÍPEDO 21
FIGURA 5: THE CUBLI 22
FIGURA 6: M-BLOCKS 23
FIGURA 7: PROTOTIPO UNA CARA DEL CUBLI 26
FIGURA 8: CONJUNTO COMPLETO CUBLI 27
FIGURA 9: RASPBERRY PI ZERO W 27
FIGURA 10:MAXON EC45 FLAT BRUSHLESS 50W 28
FIGURA 11: VOLANTE DE INERCIA [11] 29
FIGURA 12: MOTOR Y VOLANTE DE INERCIA [11] 29
FIGURA 13: MPU6050 (IMU) 30
FIGURA 14: ADC3 CLICK 31
FIGURA 15: CONEXIÓN PROTOCOLO I2C 32
FIGURA 16: COMUNICACIÓN I2C 33
FIGURA 17: ILUSTRACIÓN PROTOTIPO UNA CARA 40
FIGURA 18: DIAGRAMA GENERAL SIMULINK 42
FIGURA 19: DIAGRAMA SIMULACIÓN SIMULINK 43
FIGURA 20: TRANSICIÓN DE ESTADOS DE LA MÁQUINA DE ESTADOS [11] 44
FIGURA 21: DIAGRAMA DE SIMULINK DE LA MÁQUINA DE ESTADOS 45
FIGURA 22: DIAGRAMA DE SIMULINK DEL ALGORITMO DE CONTROL 46
FIGURA 23: CONFIGURACIÓN MICROCONTROLADOR 46
FIGURA 24: CONFIGURACIÓN ESCON 36/3 EN ESCON STUDIO 47
FIGURA 25: DIAGRAMA SIMULINK DEL MOTOR 48
FIGURA 26: DIAGRAMA SIMULINK IMU (NIVEL SUPERIOR) 49
FIGURA 27: DIGRAMA SIMULINK IMU (NIVEL INFERIOR) 50
FIGURA 28: DIAGRAMA SIMULINK ADC 51
FIGURA 29: ÁNGULO DEL CUERPO EN SIMULACIÓN 52
FIGURA 30: VELOCIDAD DEL PROTOTIPO EN SIMULACIÓN 53
FIGURA 31: VELOCIDAD DEL MOTOR EN SIMULACIÓN 53
FIGURA 32: CORRIENTE DEL MOTOR EN SIMULACIÓN 54
FIGURA 33: PRUEBA IMU A -45 GRADOS 55
FIGURA 34: PRUEBA IMU A 0 GRADOS 56
FIGURA 35: PRUEBA IMU A 45 GRADOS 56

FIGURA 36: ENSAYO MCP3428 CON EL MOTOR PARADO 57
FIGURA 37: ENSAYO DE ACELERACIÓN 58
FIGURA 38: ÁNGULO DE INCLINACIÓN ENSAYO BALANCEO 59
FIGURA 39: VELOCIDAD DEL PROTOTIPO ENSAYO BALANCEO 59
FIGURA 40: VELOCIDAD DEL MOTOR ENSAYO BALANCEO 59

1. INTRODUCCIÓN

El proyecto se centra en el diseño de un control que permita a la cara de un cubo levantarse desde el reposo y equilibrarse sobre uno de sus vértices. El control se basa en los principios para controlar los denominados sistemas de péndulo invertido [1]. Mediante unas ligeras modificaciones, este control podrá ser extrapolado a las otras caras del cubo, obteniendo un control 3D para así, alcanzar el fin último del proyecto: "*The Cubli: A Cube that can Jump Up and Balance*" [2].

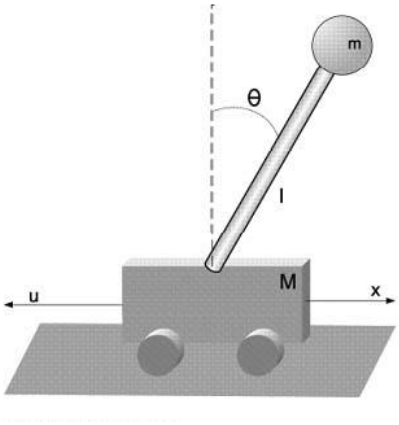
1.1 ESTADO DEL ARTE

En las últimas décadas ha surgido un gran interés por los denominados sistemas subactuados [3], los cuales, ofrecen un gran ahorro de energía. Esta eficiencia energética se debe a que, utilizando menos actuadores que grados de libertad existentes en el sistema, son capaces de conseguir un control robusto.

Dentro de los sistemas subactuados, se encuentran los ya mencionados sistemas de péndulo invertido, caracterizados por tener un punto de estabilidad inestable. La flexibilidad de su modelo matemático convierte a los sistemas modelados como péndulos invertidos, en un exitoso banco de pruebas para diseñar y evaluar diferentes métodos de control lineal y no lineal. Por esta razón, dichos sistemas tienen tantas aplicaciones en los sectores de la ingeniería aeroespacial, robótica o industrial. Debido a que el control realizado en el proyecto está basado en estos sistemas, vamos a presentar a continuación algunos casos similares.

1.1.1 Péndulo invertido sobre carro

Corresponde a la estructura convencional de un péndulo invertido. Se trata de un mecanismo muy sencillo, compuesto por un carro que se desplaza en una trayectoria lineal y sobre el cual, se encuentra un péndulo con libertad para rotar 360º sobre su eje. Es considerado un sistema SISO (Single Input Single Output), que al igual que nuestro proyecto, presenta un punto de estabilidad inestable cuando el péndulo forma 90º con la horizontal. Como no existe ninguna fuerza aplicada que mantenga el péndulo en la posición de equilibrio, cualquier pequeña perturbación lo sacaría de ella. Para conseguir mantener estable el sistema, hay que modificar la aceleración lineal del carro [4].



$$q = \begin{bmatrix} \theta \\ x \end{bmatrix}; \quad \dot{q} = \begin{bmatrix} \dot{\theta} \\ \dot{x} \end{bmatrix} \tag{3}$$

$$\ddot{\theta} = \frac{1}{\Delta}[\gamma w \sin\theta - \beta^2 \dot{\theta}^2 \sin\theta\cos\theta - \beta\cos\theta u] \tag{4}$$

$$\ddot{x} = \frac{1}{\Delta}[-\beta w \sin\theta\cos\theta + J\beta\sin\theta\dot{\theta}^2 + Ju] \tag{5}$$

donde:

$$\Delta = J\gamma - \beta^2\cos^2\theta$$
$$w = mgl$$
$$\gamma = M + m$$
$$\beta = ml$$
$$J = ml^2$$

Figura 1: Péndulo invertido sobre carro

1.1.2 Péndulo de Furuta

Enunciado y resuelto por el profesor K. Furuta, adscrito al Tokio Institute of Technology [5]. El sistema es similar al mencionado anteriormente, con la diferencia de que la aceleración lineal que producía el carro sobre el péndulo para mantenerlo en el punto de equilibrio, viene originada por la aceleración angular que induce un actuador sobre una varilla, en cuyo extremo se encuentra el péndulo.

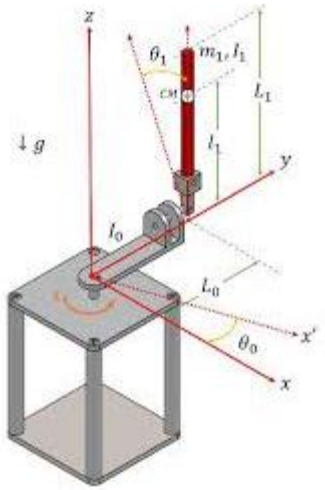


Figura 2: Péndulo de Furuta

1.1.3 Lanzamiento de cohetes

Debido a que las aceleraciones producidas por las toberas del cohete no son del todo simétricas, resulta necesario un control que mantenga a la nave en la dirección correcta. En este caso, el control es algo más complejo, ya que el sistema se modela como un sistema de doble péndulo invertido [6]. El ángulo de inclinación es controlado por medio de la variación del ángulo de la fuerza de empuje.

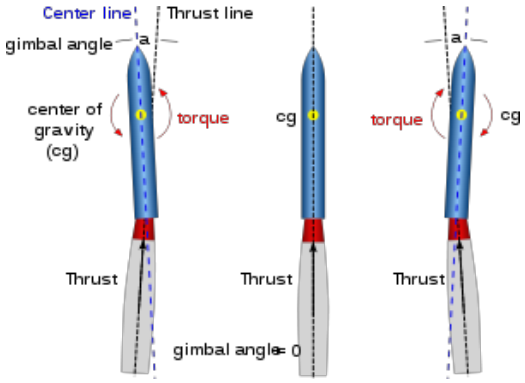


Figura 3: Control dirección de un cohete

1.1.4 Robots bípedos

Uno de los mayores retos que presenta el modelado de un robot bípedo, es conseguir que dicho robot mantenga el equilibrio cuando levanta uno de sus apoyos para avanzar. A menudo, el sistema de equilibrio utilizado es el de un péndulo invertido, en donde la pierna en movimiento se comporta como un péndulo que oscila libremente [7].

Figura 4: Robot Bípedo

1.1.5 “The Cubli: A Cube that can Jump Up and Balance”

Es el sistema en el que está basado este proyecto. Se trata de un cubo en el que tres de sus caras, tienen acopladas un motor que acelera en la dirección deseada un volante de inercia. La principal cualidad del *Cubli,* es su capacidad de saltar desde su posición de reposo para a continuación, equilibrarse sobre una de sus aristas o uno de sus vértices sin la necesidad de un apoyo externo. Para poder realizar el salto inicial, los volantes de inercia son acelerados a altas velocidades y frenados de golpe. Una vez realizada la arrancada y alcanzada la posición de equilibrio, se realiza una medición de estado basada en la unidad de medición inercial (IMU) y el control actúa sobre los motores para permitirle mantener el equilibro [8].

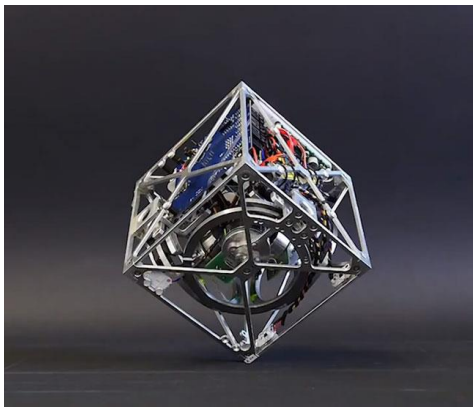


Figura 5: The cubli

1.1.6 “M-Blocks”

Se trata de un robot formado por pequeños módulos cúbicos con una capacidad de autoconfiguración para crear estructuras auto ensamblables [9]. Al igual que el *Cubli*, los pequeños módulos individuales no necesitan apoyos externos para poder realizar desplazamientos, sino que utilizan movimientos pivotantes para alcanzar la estructura deseada.

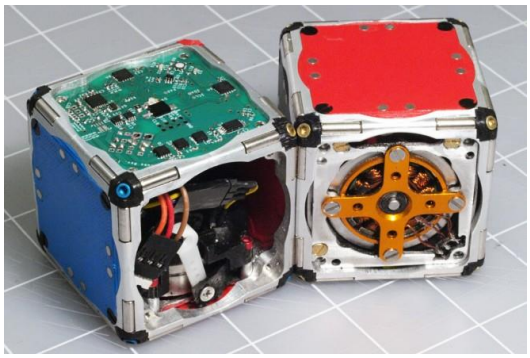


Figura 6: M-Blocks

1.2 MOTIVACIÓN

No cabe duda de que el *Cubli* es un proyecto que llama la atención a primera vista, por lo que la motivación inicial no es otra que la curiosidad. Tras una breve explicación por parte del tutor, se añade el reto de corregir los errores que se presentaron en años anteriores y las ganas de continuar con el trabajo realizado.

Por otra parte, el proyecto se desarrolla en el entorno Matlab/Simulink, el cual es familiar, lo que facilita notablemente el trabajo y la adaptación. A su vez, el proyecto aborda muchos de los conceptos aprendidos en el último año como estudiante de ingeniería electrónica y permite llevar dichos conceptos a la práctica.

Profundizando un poco más, resulta muy llamativo la cantidad de utilidades que tienen los sistemas de péndulo invertido y el excelente banco de pruebas que ofrece el proyecto para evaluar distintos métodos de control. Además, el *Cubli* no solo tiene una utilidad relacionada con la investigación, sino también una práctica. En concreto, la NASA está pensando utilizarlo para la exploración de pequeños cuerpos celestes como asteroides y cometas [10].

1.3 OBJETIVOS

1. Revisión y actualización de toda la electrónica de medida y actuación para el equilibrado de una cara del *Cubli*.
2. Diseño del sistema de control para el equilibrado de la cara partiendo de la posición vertical.
3. Diseño del sistema de aceleración y frenado para equilibrar la cara desde la posición horizontal.

1.4 METODOLOGÍA

1.4.1 Tareas

- Búsqueda de componentes que satisfagan las condiciones y especificaciones mínimas para el correcto funcionamiento.
- Adaptar prototipo de un anterior proyecto a las especificaciones y componentes del nuevo diseño.
- Calibración y pruebas de funcionamiento de todos los sensores y actuadores por separado.
- Diseño del control de balanceo en el ángulo requerido de un eje con Simulink.
- Creación del simulador para probar los diseños elaborados anteriormente que consta de:
    - Modelado analógico y digital de la planta en Matlab.
    - Creación del banco de pruebas de la planta en Simulink.
- Idear e implantar un método de manejo exterior.
- Redacción de la memoria.

1.5 RECURSOS

1.5.1 Software

- Matlab para las funciones y bloques propios y el modelado de la planta
- Simulink para generar todo el código de control en diagrama de bloques
- ESCON: software de control del motor MAXON

1.5.2 Hardware

- Motor DC: MAXON EC45 flat 50W con sensores hall incorporados.
- ESCON 36/3 EC: controlador del motor
- IMU: sensores para calcular aceleración y velocidad de las caras.
- Raspberry pi zero W
- Servo MG90S: para el sistema de frenado.
- MCP3428: ADC para convertir las señales analógicas en digitales

2. PROTOTIPO

2.1 Estructura

La estructura del prototipo está basada en el prototipo de una cara realizado por la universidad ETH de Zurich. Consta de una cara cuadrada de aluminio, fijada por uno de sus vértices a un soporte que le permite girar con libertad sobre el plano vertical, y que tiene acoplada los componentes necesarios para llevar a cabo el control de sistema: motor, volante de inercia, sensores, servo y microcontrolador.

Figura 7: Prototipo una cara del Cubli [11]

Respecto al año anterior [11], se mantuvo la estructura y el diseño del prototipo, pero se realizaron algunas mejoras con el objetivo de solucionar los problemas del anterior proyecto. La base en la que estaba fijada el prototipo se sustituyó por una más grande para mejorar la estabilidad y así tratar de reducir las vibraciones producidas por el motor. La controladora ESCON 36/3 EC se fijó a la nueva base y, el microcontrolador se acopló a la cara externa de la pieza de aluminio.

Finalmente, se añadió una estructura de metacrilato en forma de techo que permite fijar el pulsador, así como las nuevas fuentes de alimentación y un interruptor para cortar la alimentación del motor en caso de que sea necesario. De esta manera, todos los componentes requeridos para el funcionamiento del sistema quedaban acoplados a la estructura del prototipo como se había propuesto al inicio del proyecto.

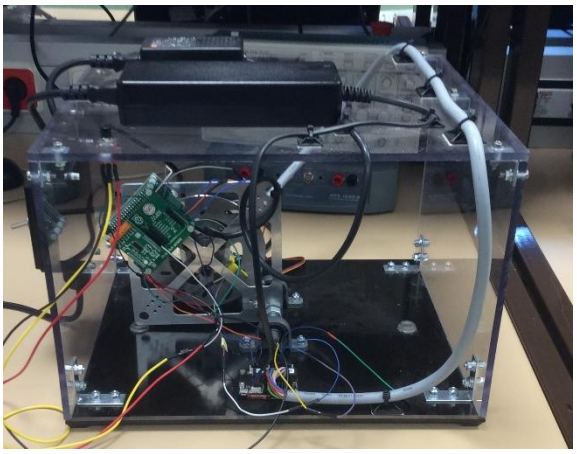


Figura 8: Conjunto completo Cubli

2.2 Microcontrolador

Debido a que el proyecto se iba a desarrollar e implementar en el entorno Matlab/Simulink, el cual ofrece un soporte para los microcontroladores de la familia Raspberry Pi, se eligió como microcontrolador una Raspberry Pi Zero W.

Las principales características que hacen a la Raspberry Pi Zero W el microcontrolador óptimo para el proyecto son su capacidad de procesamiento, los diferentes medios de comunicación que ofrece con el resto de los componentes y, su diferencia de tamaño y peso respecto al resto de modelos Raspberry Pi. Además, hay que destacar la posibilidad que ofrece de añadir componentes gracias a su compatibilidad con las *Click Boards*.

La comunicación con los componentes se ha realizado principalmente mediante los pines GPIO (General Pupose Input Output), ya que permiten de manera sencilla la entrada y salida de señales digitales. También se ha utilizado la señal WIFI para la monitorización en tiempo real del sistema con el ordenador y el puerto I2C para la conexión con el ADC y los sensores IMU.

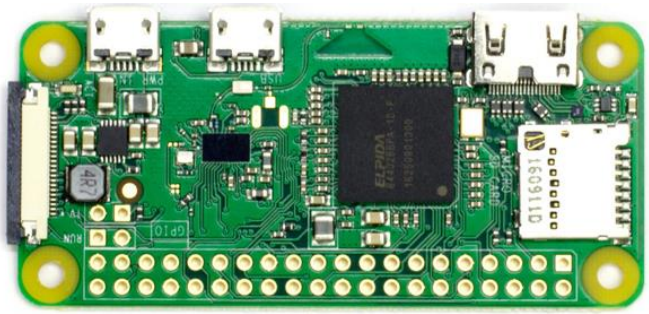


Figura 9: Raspberry Pi Zero W

2.3 Motor y volante de inercia

El motor utilizado es un Maxon EC45 FLAT Brushless de 50W. Este motor tiene una tensión y corriente nominal de 24V y 2.32A, pudiendo alcanzar una velocidad máxima de 10.000rpm. La potencia del motor es considerable debido a que, en el control de balanceo, son necesarias aceleraciones rápidas sobre el volante de inercia que no permitan al sistema superar el ángulo límite de recuperación.

Además de satisfacer las necesidades del sistema, el motor tiene incorporados unos sensores hall que permiten obtener la corriente y la velocidad real del motor. Estas señales, serán procesadas por la controladora ESCON 36/3 EC, que se encarga de la configuración y monitorización del motor a través de la herramienta *ESCON Studio*.

Figura 10:Maxon EC45 FLAT Brushless 50W

Como volante de inercia, se eligió un disco de freno de mini moto debido a la imposibilidad de generar uno en 3D. El disco al estar equilibrado y ser de acero, tiene el peso suficiente para elevar la estructura cuando se active el sistema de frenado y no desequilibrar la estructura.

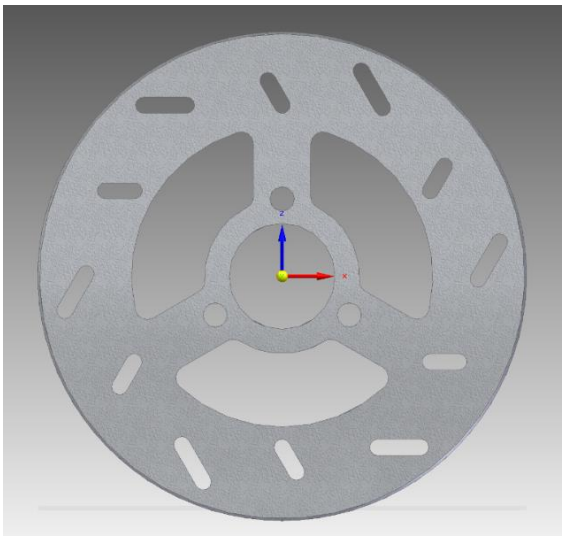


Figura 11: Volante de inercia [11]

La unión del motor y el volante de inercia se realizó con tornillería, tal y como se muestra a continuación.

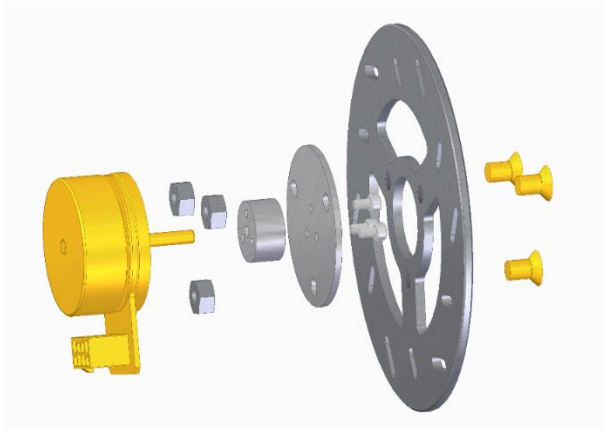


Figura 12: Motor y Volante de inercia [11]

2.4 Sensores (IMU)

Los sensores IMU (Inertial Measurement Unit), son una combinación de giróscopos y acelerómetros que permiten medir la velocidad y aceleración de aquellos cuerpos en los que se encuentran. En este proyecto, se han utilizado dos sensores iguales del modelo MPU6050, formados por un giróscopo y un acelerómetro en cada uno de sus ejes. Su reducido tamaño y peso, permite acoplar ambos sensores al prototipo y de esta manera, mediante el módulo de comunicación I2C que tienen incluidos los sensores, se obtienen las medidas del giróscopo del

eje Z y de los acelerómetros del eje X e Y. Todas estas medidas, serán utilizadas para la estimación de las variables de estado.

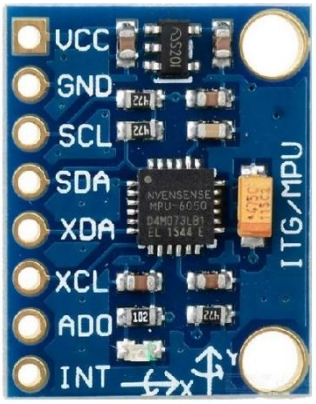


Figura 13: MPU6050 (IMU)

2.5 ADC

Como se ha mencionado anteriormente, gracias a los sensores hall del motor y la microcontroladora ESCON 36/3 EC, es posible obtener la velocidad a la que gira el motor. La única complicación es que la microcontroladora nos proporciona la velocidad en forma de una señal analógica. Como la Raspberry Pi solo es capaz de procesar señales digitales, resulta necesario un conversor de señal analógica a digital.

En el proyecto anterior, la conversión de esta señal analógica se llevaba a cabo con la ayuda de un Arduino Nano externo, el cual, dispone de un conversor de señal analógica a digital propio. Una vez convertida la señal a digital, el Arduino mandaba la medida por protocolo Serial a la Raspberry.

Sin embargo, en este proyecto se decidió sustituir el Arduino Nano por un ADC MCP3428 (*Analog to Digital Convertor)*, cuya conexión con el microcontrolador es directa gracias al *ADC3 click* de la familia *click boards*. Este cambio, simplifica notablemente el sistema ya que la comunicación entre el *ADC3 click* y la Raspberry Pi se realiza, al igual que con las IMU, mediante protocolo I2C. Además, al tener una conexión tan sencilla con el microcontrolador, facilita su acoplamiento a la cara de aluminio del prototipo. Por último, cabe destacar la posibilidad que ofrece de convertir cuatro señales al mismo tiempo, en caso de que sea necesario en futuras ampliaciones del proyecto.

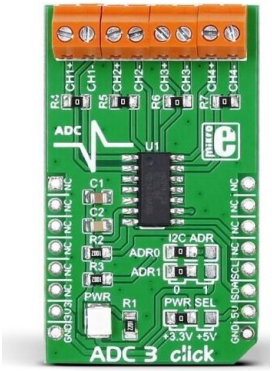


Figura 14: ADC3 click

2.6 Alimentación

Se requieren dos tensiones diferentes para alimentar los componentes del sistema. La Raspberry Pi Zero W necesita una tensión de 5V y el motor MAXON requiere una de 24V. Para ello se utilizaron dos fuentes de alimentación distintas.

En el caso de la primera fuente de alimentación, la potencia no tiene gran relevancia ya que Raspberry Pi Zero W no necesita corrientes elevadas para su correcto funcionamiento. Por esta razón, se eligió una fuente de 5V con una potencia de 20W.

Sin embargo, durante el control de balanceo, surgen picos de corrientes en el motor cuando este produce las aceleraciones angulares necesarias. Es por ello, que la potencia de la fuente en este caso cobra una gran importancia en la elección de la alimentación del motor. Finalmente, tras la revisión de los resultados obtenidos en el curso pasado, se eligió una fuente de alimentación de 24V con una potencia de 160W.

3. COMUNICACIÓN

3.1 Protocolo I2C

Como ya se ha mencionado anteriormente, el protocolo I2C (*Inter-Integrated Circuit*) se utiliza para la comunicación entre el microcontrolador, el ADC y los sensores IMU. La comunicación I2C es común en aquellos sistemas donde existe un dispositivo central, conocido también como *Maestro*, que necesita comunicarse con varios dispositivos subordinados, conocidos también como *Esclavos*. En este proyecto, el microcontrolador actúa como maestro mientras que las IMU y ADC como esclavos.

El protocolo se caracteriza por utilizar tan solo dos líneas de datos a los que todos los dispositivos se encuentran conectados: SDA (*Serial Data*) y SCL (*Serial Clock*). Por la primera línea circula la información en paquetes de 8 bits, mientras que la segunda línea es controlada por el maestro y se utiliza para sincronizar todos los intercambios de información. Ambas líneas se encuentran en pull-up.

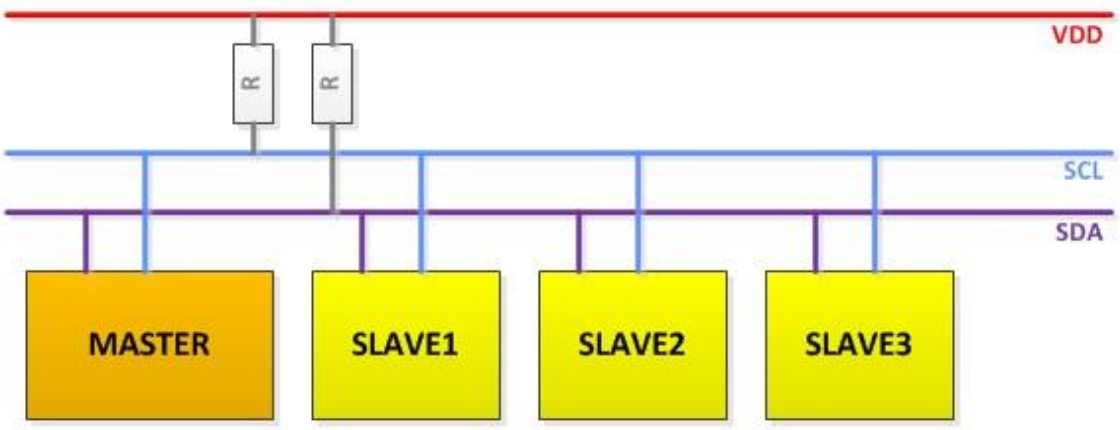


Figura 15: Conexión Protocolo I2C

La línea SDA solo puede cambiar cuando la señal de reloj se encuentra en nivel bajo; excepto para marcar el inicio y final de la comunicación. Por esta razón, para iniciar la comunicación, el maestro envía un flanco descendiente mientras el reloj se encuentra en nivel alto.

A continuación, el maestro manda un paquete de 8 bits, donde los primeros 7 bits son la dirección del esclavo con el que desea comunicarse y el último bit, el tipo de comunicación

que desea: escritura (1) o lectura (0). Después de enviar un paquete de 8 bits, el maestro espera a recibir un bit de vuelta por parte del esclavo denominado: ACK (*Acknowledge*). El ACK recibido es 0 si el esclavo ha recibido la información y 1 si no la ha recibido.

Por último, una vez compartida toda la información, el maestro envía un flanco ascendente cuando el reloj se encuentra en nivel alto, marcando así el final de la comunicación.

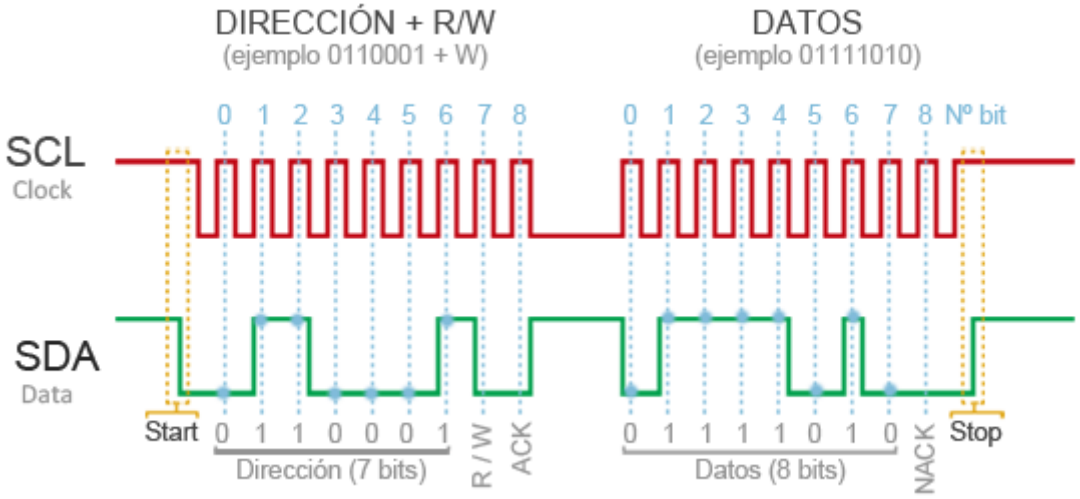


Figura 16: Comunicación I2C

4. MODELADO DEL SISTEMA

4.1 Ecuaciones del modelo

Las ecuaciones dinámicas del sistema se apoyan de la mecánica lagrangiana [11] y hacen uso de las ecuaciones de Euler-Lagrange. El planteamiento que se siguió está basado en el proyecto del Cubli realizado por departamento de señales y sistemas de la "Chalmers University of Technology" [12].

En primer lugar, se define el Lagrangiano del sistema, siendo este la energía cinética (T) del sistema menos su energía potencial (V):

$$L = T - V \tag{1}$$

La formulación lagrangiana simplifica muchos problemas físicos ya que no depende del sistema de referencia elegido por lo que ambos términos del Lagrangiano se expresan en términos de las coordenadas y velocidades generalizadas. Para resolver el Lagrangiano, se definen las ecuaciones de Euler-Lagrange:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathrm{L}}{\partial \dot{\mathrm{q}}_{\mathrm{k}}}\right) - \frac{\partial \mathrm{L}}{\partial \mathrm{q}_{\mathrm{k}}} + \frac{\partial \mathrm{R}}{\partial \dot{\mathrm{q}}_{\mathrm{k}}} = \tau_{\mathrm{k}} \tag{2}$$

Donde $\frac{\partial \mathrm{R}}{\partial \dot{\mathrm{q}}_{\mathrm{k}}}$ modela las fuerzas disipativas del sistema y τ_{k} los pares de inercia.

Tras introducir los parámetros de la energía potencial y cinética del sistema, el Lagrangiano se formula de la siguiente manera:

$$L = \frac{1}{2} I_b \dot{\theta}_b^2 + \frac{1}{2} I_w \left(\dot{\theta}_b + \dot{\theta}_w\right)^2 + \frac{1}{2} m_b (\dot{\theta}_b l)^2 - (l_b m_b + m_w l) g \cos\theta_b \tag{3}$$

Donde:

- Θ_b = ángulo de la diagonal de la cara del péndulo respecto el eje vertical
- Θ_w = ángulo del volante de inercia respecto la diagonal del péndulo
- m_w = masa del volante de inercia y rotor del motor
- m_b = masa del péndulo
- l_b = momento de inercia del péndulo sobre su eje de rotación

- I_w = momento de inercia del volante de inercia y rotor del motor sobre el eje del motor
- l = distancia del eje de rotación del motor al eje de rotación del péndulo
- l_b = distancia del eje de rotación del péndulo a su centro de gravedad
- g = aceleración de la fuerza de gravedad (9.81 ms^{-2})

A continuación, tras combinar el método de los momentos generalizados [13] (derivada del langragiano respecto a las velocidades angulares) y las ecuaciones de Euler-Lagrange, se despejan las aceleraciones angulares del volante de inercia y del péndulo. Las ecuaciones obtenidas son las ecuaciones principales no lineales que modelan el sistema:

$$\ddot{\theta}_b = \frac{(m_b l_b + m_w l) g \sin\theta_b - T_m - C_b \dot{\theta}_b + C_w \dot{\theta}_w}{I_b + m_w l^2} \quad (4)$$

$$\ddot{\theta}_w = \frac{(I_b + I_w + m_w l^2)(T_m - C_w \dot{\theta}_w)}{I_w (I_b + m_w l^2)} - \frac{(m_b l_b + m_w l) g \sin\theta_b - C_b \dot{\theta}_b}{I_b + m_w l^2} \quad (5)$$

Donde:

- C_b = Coeficiente de rozamiento del péndulo con el eje de rotación
- C_w = Coeficiente de rozamiento del motor
- T_m = Par del motor ($T_m = K_m * i$) ($K_m = 5.24 * 10^{-3}$ NmA^{-1})

4.2 Variables de Estado

Las variables de estado de un sistema dinámico son el conjunto más pequeño de variables que definen el estado del sistema. En el sistema anterior las variables de estado son:

- θ_b = Ángulo de la diagonal del péndulo respecto a la vertical (perpendicular a la base).
- $\dot{\theta}_b$ = Velocidad angular del péndulo, a partir de ahora se denominará ω_b.
- $\dot{\theta}_w$ = Velocidad angular del volante de inercia, a partir de ahora se denominará ω_w.

4.3 Punto de operación

Definidas las variables de estado y las ecuaciones que definen nuestro modelo, solo falta determinar el punto para el cual el sistema se encuentra en equilibrio inestable y en el que se desea mantener el sistema. El punto de operación del sistema se encuentra cuando la diagonal del péndulo se sitúa perpendicular a la base, es decir, $\theta_b = 0$. Una vez hallado el valor de θ_b es posible calcular el valor del resto de variables de estado en el punto de equilibrio: $\dot{\theta}_b = 0$; $\dot{\theta}_w = 0$. Por lo tanto, se puede concluir que el punto de operación es cuando todas las variables de estado son nulas.

4.4 Linealización

Para llevar a cabo una representación de espacio de estados, es necesario que todas las ecuaciones del sistema sean lineales. De esta manera, las derivadas de las variables de estado y las salidas solo dependen de la entrada y de las propias variables de estado. La representación standard de un espacio de estados es la siguiente:

$$\frac{dX}{dt} = AX + BU \tag{6}$$

$$Y = CX + DU \tag{7}$$

Donde las matrices del sistema una vez linealizadas son:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \frac{(m_b l_b + m_w l) g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & \frac{C_w}{I_b + m_w l^2} \\ -\frac{(m_b l_b + m_w l) g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & -\frac{C_w (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} 0 \\ -\frac{K_m}{I_b + m_w l^2} \\ \frac{K_m (I_b + I_w + m_w l^2)}{I_w (I_b + m_w l^2)} \end{bmatrix} \tag{9}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

4.5 Identificación de Parámetros

La identificación de los parámetros utilizados del sistema es el último paso necesario para poder realizar la implantación del modelo en el entorno Simulink. Para el cálculo de las distancias se utilizó la herramienta Solid Edge, y para los parámetros de los componentes la información procedente de los fabricantes.

La mayor complejidad se encontraba en el cálculo de los momentos de inercia del volante y la estructura. El momento de inercia del volante se calculó mediante la geometría de la pieza (única y monomórfica) y el de la estructura, al haber sido diseñada con Solid Edge, se pudo obtener de manera inmediata gracias a la herramienta *Esamblaje*.

4.5.1 Parámetros del motor

Los parámetros del motor que se necesitaron fueron:

- Máxima velocidad permitida: 10.000 rpm
- Máxima corriente permitida: 2,3 A nominal, 9 A de pico
- Constante térmica del devanado: 4.95 K/W
- Constante de par motor (Km): 33,5 mNm/A

4.5.2 Distancias y masas

Las distancias están medidas cuando el sistema se encuentra en la posición de equilibrio inestable y desde el punto de giro.

- Distancia hasta el eje de rotación del volante de inercia: 91,925mm
- Distancia hasta el centro de gravedad del sistema: 85 mm

4.5.3 Momentos de Inercia

Momentos de inercia relacionados con el volante de inercia:

- Momento de inercia del disco = $2 \cdot 10^{-4}$ kg/m^2
- Momento de inercia del rotor = $1{,}35 \cdot 10^{-5}$ kg/m^2
- Momento de inercia del conjunto (I_w) = $2{,}135 \cdot 10^{-4}$ kg/m^2

Momento de inercia de la estructura completa:

- Momento de inercia de la estructura (I_B) = $7{,}843 \cdot 10^{-3}$ kg/m^2

5. DISEÑO DEL CONTROL

5.1 Estimación de las variables de estado

Una vez definidas las variables de estado y linealizadas las ecuaciones del sistema, se procede al diseño del control del péndulo invertido. Hay que tener claro que la estimación de las variables de estado es fundamental ya que, como se ha menciona anteriormente, las salidas solo dependen de la entrada aplicada y de las propias variables de estado.

En los siguientes apartados se procede a explicar como se realizan las estimaciones de cada una de las variables de estado.

5.1.1 Estimación del ángulo de inclinación

A partir de las medidas de los acelerómetros del eje X e Y procedentes de la IMU y gracias a que la gravedad es constante tanto en dirección como en módulo, es posible calcular el ángulo de inclinación respecto a la normal. El ángulo es calculado a partir de las proyecciones de la gravedad sobre los ejes X e Y de las coordenadas locales de la IMU.

En la práctica, las aceleraciones angulares producidas por el volante de inercia interfieren en la medida de los acelerómetros de la IMU por lo que el método anterior no tiene sentido aplicarlo. Por esta razón se emplea el método llevado a cabo por la universidad ETH de Zurich en su proyecto del Cubli; basado en la compensación de las aceleraciones producidas por el volante mediante el uso de dos IMU colineales en la diagonal de la cara. El procedimiento se muestra a continuación:

$$ {}_{ai}m \coloneqq (r_i\ddot{\theta}_b + g\sin\theta_b, -r_i\dot{\theta}_b - g\cos\theta_b, 0), \qquad i = 1,2 \tag{12}$$

Los términos dinámicos se eliminan resolviendo la siguiente ecuación donde $\mu = \frac{r_1}{r_2}$:

$$ {}_{a1}m - \mu\ {}_{a2}m = ((1-\mu)g\sin\theta_b, -(1-\mu)g\cos\theta_b, 0) =: (m_x, m_y, 0) \tag{13}$$

De las cuales se puede despejar la siguiente estimación del ángulo:

$$\hat{\theta}_b = atan(-\frac{m_x}{m_y}) \tag{14}$$

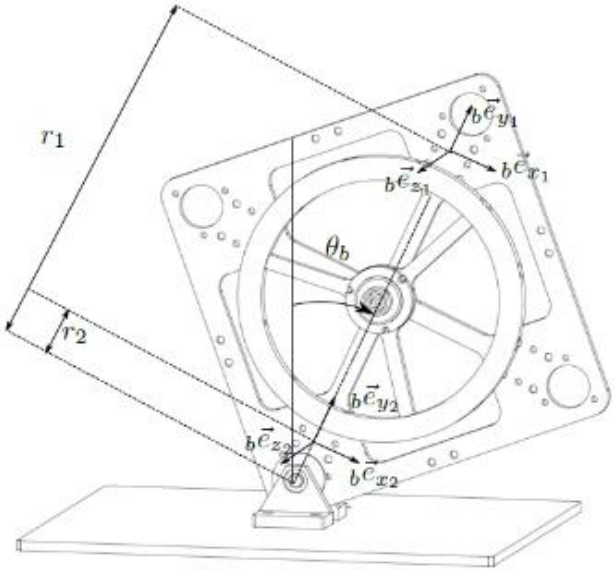


Figura 17: Ilustración prototipo una cara

5.1.2 Estimación de la velocidad de rotación del péndulo

La estimación de la velocidad de rotación es prácticamente directa. Se obtiene de la medida del giróscopo en el eje Z de la IMU. Una vez se tienen las medidas de ambas IMUs, se hace la media para reducir el error y se cambia a rad/s.

$$\widehat{\omega}_b = \frac{\omega_{z1} + \omega_{z2}}{2} \tag{15}$$

5.1.3 Estimación de la velocidad de rotación del volante de inercia

Al ser la velocidad de rotación del volante de inercia la misma que la velocidad del rotor, resulta muy sencilla su lectura. Los sensores hall que tiene el motor permiten la lectura de la velocidad del rotor a través de la controladora ESCON 36/3 EC. La salida de la controladora se lee mediante el ADC, cuya lectura se convierte de voltios a rad/s

5.2 Control de balanceo

Consiste en un control por realimentación de estados. Es la parte más compleja del proyecto y una vez se consigue balancear la cara de un cubo sobre su eje, resulta muy sencillo extrapolarlo al resto de las caras para la obtención de un control 3D. El diseño del control se realiza sobre el modelado del sistema explicado anteriormente.

En primer lugar, antes de empezar a diseñar el control de balanceo, es necesario que estén fijados los polos del sistema en las posiciones convenientes. Para ello, se ha elegido la configuración Butterworth de tercer orden, que consiste en colocar dos polos complejos y uno real; todos con el mismo módulo. Dichos polos, tienen que cumplir con las necesidades del sistema. Deben proporcionar tanto un sobrepaso pequeño, como la rapidez suficiente para que no permitan al sistema superar el ángulo límite de recuperación.

Una vez fijados los polos, se aplica la fórmula de Ackerman, que nos permite calcular la correcta realimentación de las variables de estado para alcanzar el punto de equilibrio. Como se puede observar a continuación, la fórmula de Ackerman trata de igualar los autovalores del lazo cerrado al producto de los polos que ya se han fijado.

$$|\lambda I - (A - KB)| = p1 \cdot p2 \cdot p3 \tag{16}$$

5.3 Control de salto

No se trata de un control como tal, ya que simplemente consiste en activar el sistema de frenado cuando la velocidad del disco alcanza una velocidad suficientemente alta para elevar el sistema. La activación del freno se obtiene mediante el comparador de velocidad de la controladora ESCON 36/3 EC y está fijada cuando el motor alcanza las 3000 rpm.

6. IMPLANTACIÓN DEL CONTROL

La implantación del control se lleva a cabo en la herramienta Simulink y sigue una organización en forma de bloques que facilita su uso y entendimiento.

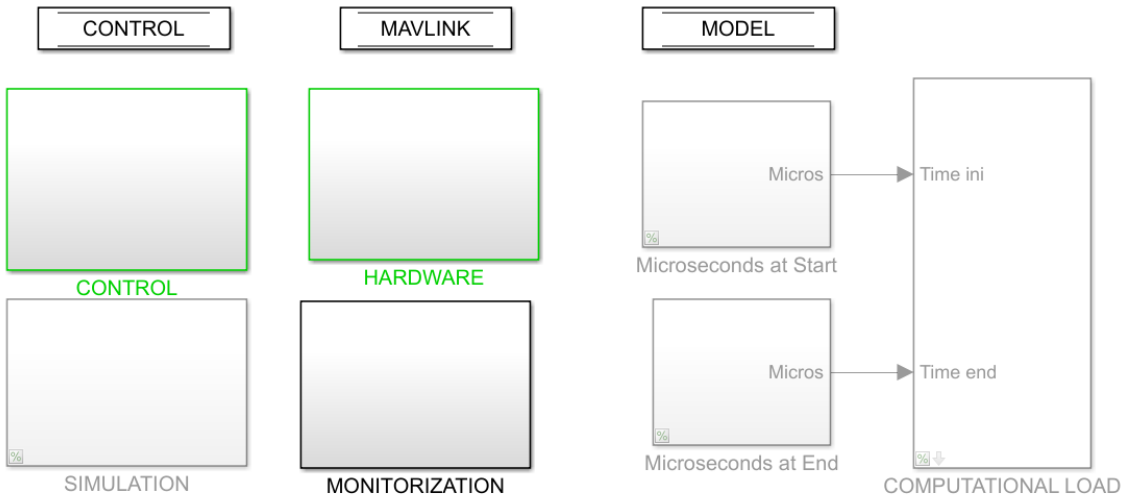


Figura 18: Diagrama general Simulink

Como se puede observar existen cuatro grandes bloques en los que se divide el proyecto: control, hardware, simulación y monitorización. En función de cómo se quiera correr el código (real-time simulation, fast-simulation o Implementación), se comentarán unos bloques u otros. Debido a conflictos de variables, hardware y simulación no pueden estar sin comentar al mismo tiempo.

6.1 Validación del control

Antes de poder realizar la implementación del control es necesario probarlo en simulaciones. En caso de no hacerlo, el hardware podría dañarse debido a excesos de corriente. Para la validación del control, se diseñó el siguiente diagrama de simulación que incluye el modelo del Cubli y una representación del sistema con los mismos buses para las variables que los que se van a usar cuando se realice la implementación en hardware. De esta manera, la simulación se acerca significativamente a la realidad y permite obtener un control más preciso.

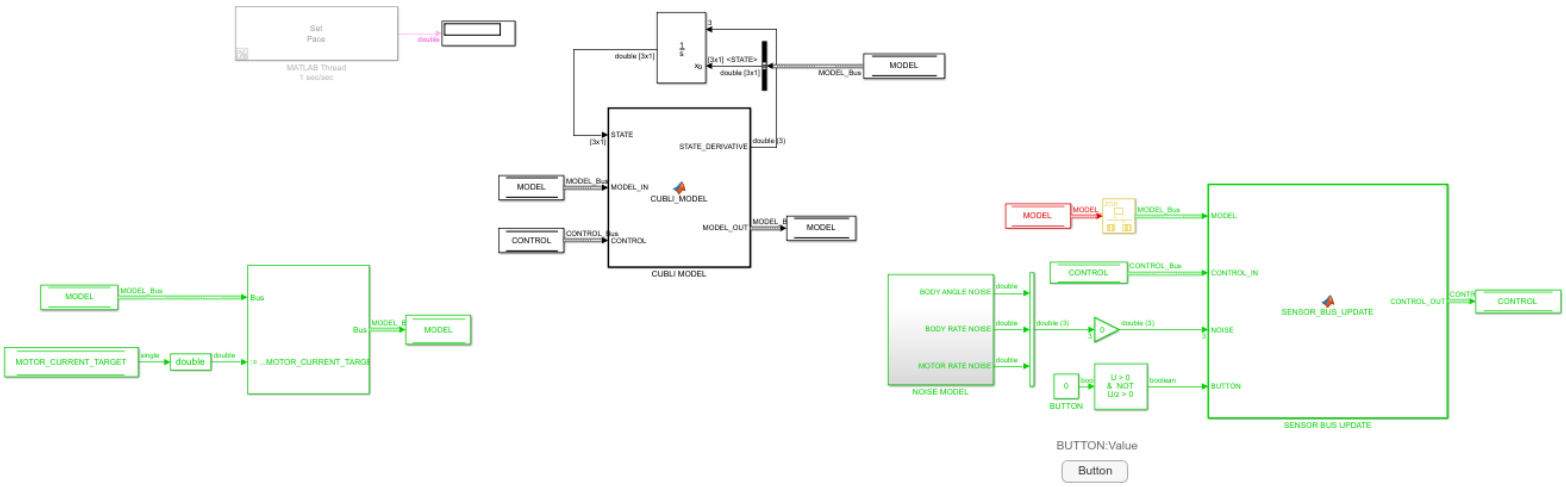


Figura 19: Diagrama Simulación Simulink

6.2 Máquina de estados

A la hora de implantar el control, debido a que hay pocas entradas y los estados son secuenciales, resulta muy útil una máquina de estados o máquina de Moore para unir los controles de salto y balanceo. En la máquina de Moore, se pasa de un estado a otro en función de las entradas y las salidas solo dependen del estado en el que se encuentre el sistema. A continuación, se explica brevemente cada uno de los estados y la transición de uno a otro:

6) STOP: Se trata del estado inicial en donde todo está parado. Para comenzar y pasar al siguiente estado hay pulsar el botón.

7) CALIBRATION: Durante este estado se están calibrando los sensores para que en las etapas posteriores funcionen adecuadamente. Como en el anterior estado, para pasar al siguiente estado hay que pulsar el botón.

8) ACCELERATION: Se activa el motor y este empieza a acelerarse hasta alcanzar la velocidad fijada para realizar el salto. Cuando se recibe la señal de activación del freno por parte de la controladora, pasa al siguiente estado.

9) BREAK: Al llegar a este estado se activa el servo para frenar el disco de forma brusca y generar el par que eleve la cara. Una vez se llega al ángulo límite del control, que se ha estimado en 10º desde la normal, entramos en el último estado.

10) BALANCE: Comienza el control de balanceo que intentará mantener el sistema en el punto de operación. Este estado termina al pulsar el botón, que hace volver al sistema al estado inicial.

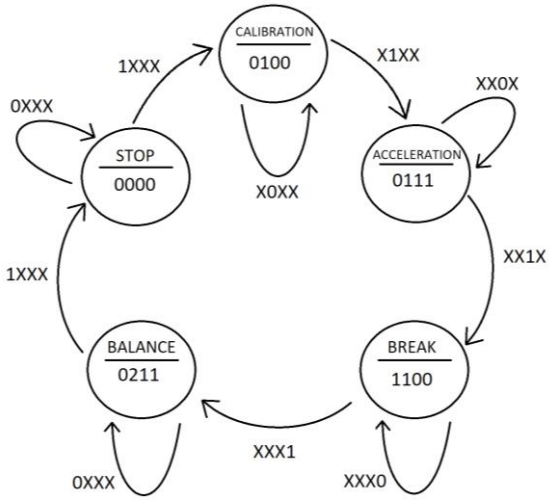


Figura 20: Transición de estados de la máquina de estados [11]

Teniendo en cuenta los diferentes estados y sus transiciones, podemos enumerar todas las entradas y salidas de nuestra máquina de estados:

- Entradas:
  - Botón: Se trata de la única entrada manual del sistema, se utiliza para iniciar y finalizar la secuencia y si el usuario lo desea, para la transición de estados manualmente.
  - Habilitación por velocidad: Variable booleana que se activa cuando el volante alcanza la velocidad predefinida en la configuración de la controladora del motor y sirve para comenzar la etapa de frenado.
  - Habilitación por ángulo: Ésta se activa cuando el ángulo está en el rango de ±10º desde la normal para pasar al estado de balanceo.

- Salidas:
  - SERVO_MODE: Actuador para el freno; se activa sólo en la etapa de freno.

- CONTROL_MODE: Cambia el modo del control, siendo posible estar parado, calibrando o en balanceo.
- MOTOR_MODE: Modifica el estado del motor en función del estado en el que se encuentre el sistema. Permitiendo o no el paso de corriente, acelerando el motor hacia una dirección u otra de manera fija o permitiendo que gire libremente en función del control de balanceo.

La máquina de estados se ha implantado en Simulink mediante un bloque con una función de Matlab que va realizando los cambios de estados y la actualización de las salidas en función de las entradas.

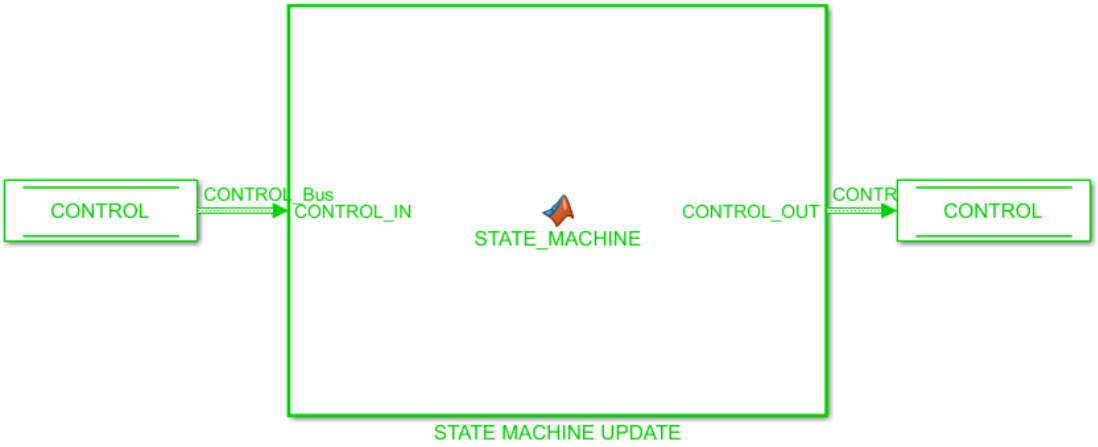


Figura 21: Diagrama de Simulink de la máquina de estados

6.3 Algoritmo de control

Al igual que la máquina de estados, la actualización del mando se realiza a través de un bloque de Simulink con una función de Matlab. Esta función es la encargada de calcular según las entradas, la corriente que se debe enviar al motor para alcanzar el punto de operación. La variable *CONTROL_MODE* determina el control que se utiliza en cada instante.

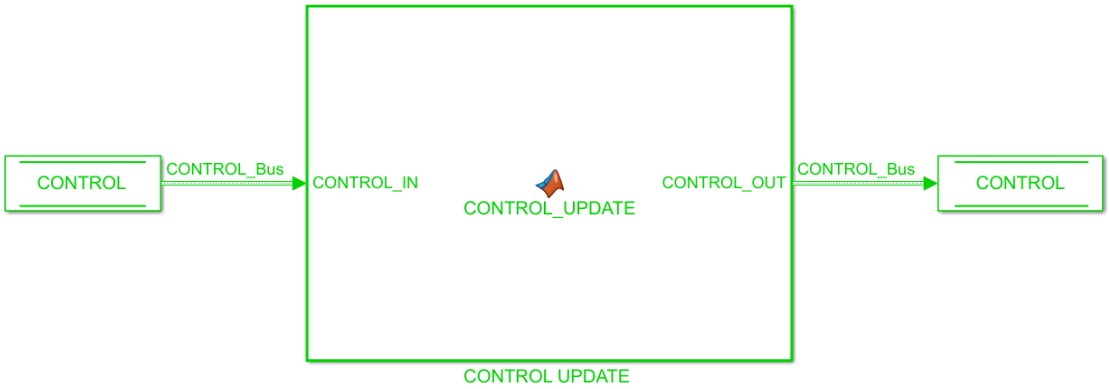


Figura 22: Diagrama de Simulink del algoritmo de control

6.4 Microcontrolador

La comunicación entre Simulink y la Raspberry Pi Zero W se lleva a cabo mediante conexión inalámbrica TCP/IP, por lo que resulta necesario conocer la IP de nuestro microcontrolador antes de poder realizar la implantación del control. Además de la IP, es necesario introducir el usuario y contraseña de la Raspberry y encontrarse conectados a la misma red WIFI.

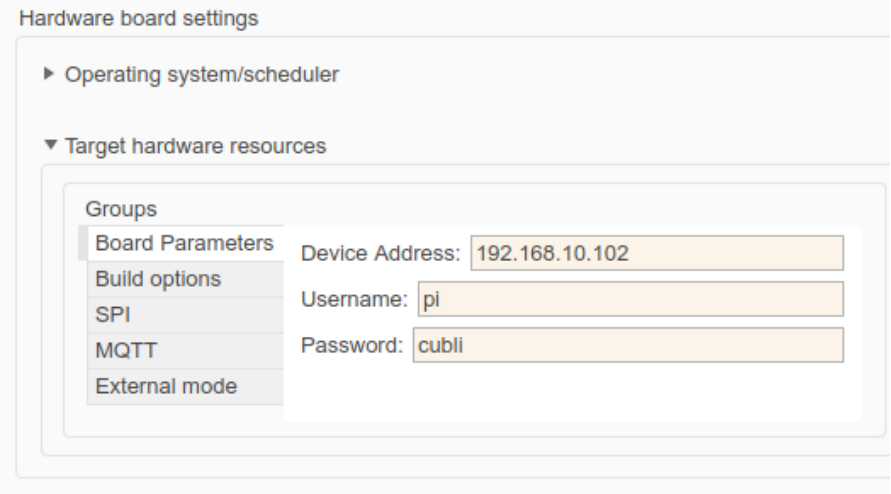


Figura 23: Configuración Microcontrolador

Este tipo de conexión no permite solamente implantar el control en el microcontrolador, sino también llevar a cabo un seguimiento a tiempo real de lo que está sucediendo en el sistema. De esta manera, es posible probar uno a uno los componentes del sistema, facilitando la identificación de errores durante la implantación.

6.5 Hardware

6.5.1 Motor

La configuración del motor se lleva a cabo mediante la controladora ESCON 36/3 EC con la ayuda de la herramienta de software ESCON Studio. El software permite configurar el tipo de entradas y salidas y definir sus rangos de operación. Además, ofrece la posibilidad utilizar la controladora como regulador de corriente o tensión. La configuración de entradas y salidas es la siguiente:

- Entrada digital 1: Señal PMW que controla la corriente de consigna del motor. La controladora esta configurada para leer señales PMW entre el 10% y el 90%, representando respectivamente una corriente de -8A y 8A.
- Entrada digital 2: Señal de habilitación del motor. Activo alto.
- Salida digital 1: Salida booleana que toma valor alto cuando el motor supera las 3000rpm en cualquiera de los dos sentidos.
- Salida analógica 1: Proporciona un promedio de la velocidad a la que está girando el motor en ese momento. La salida analógica se mueve en un rango de 0 a 2V para velocidades de ± 3.500rpm.

El rango de tensión seleccionado de la salida analógica 1 se debe a que el ADC no es capaz de leer tensiones superiores a 2.048V. A pesar de que el motor puede llegar a velocidades de ± 10.000rpm, el rango establecido es de ±3.500rpm ya que, la máxima velocidad que alcanzará el motor es de 3.000 rpm que será cuando salte el sistema de frenado.

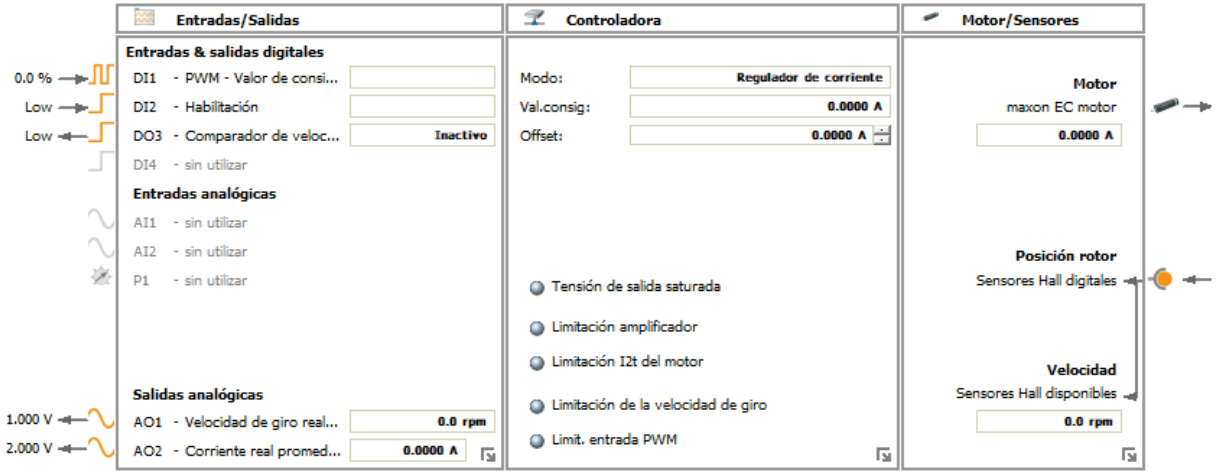


Figura 24: Configuración ESCON 36/3 en ESCON studio

A continuación, se muestra el diagrama de bloques del control del motor. El bloque superior consiste en una función de Matlab que se encarga de seleccionar las salidas del motor en función del estado en el que se encuentre el sistema. La variable *MOTOR_ENABLE* se encarga de la habilitación del motor mientras que la variable *CURRENT_unit*, proporciona el valor deseado de la señal PMW. Los valores de ambas variables dependen del valor de la variable *MOTOR_MODE*.

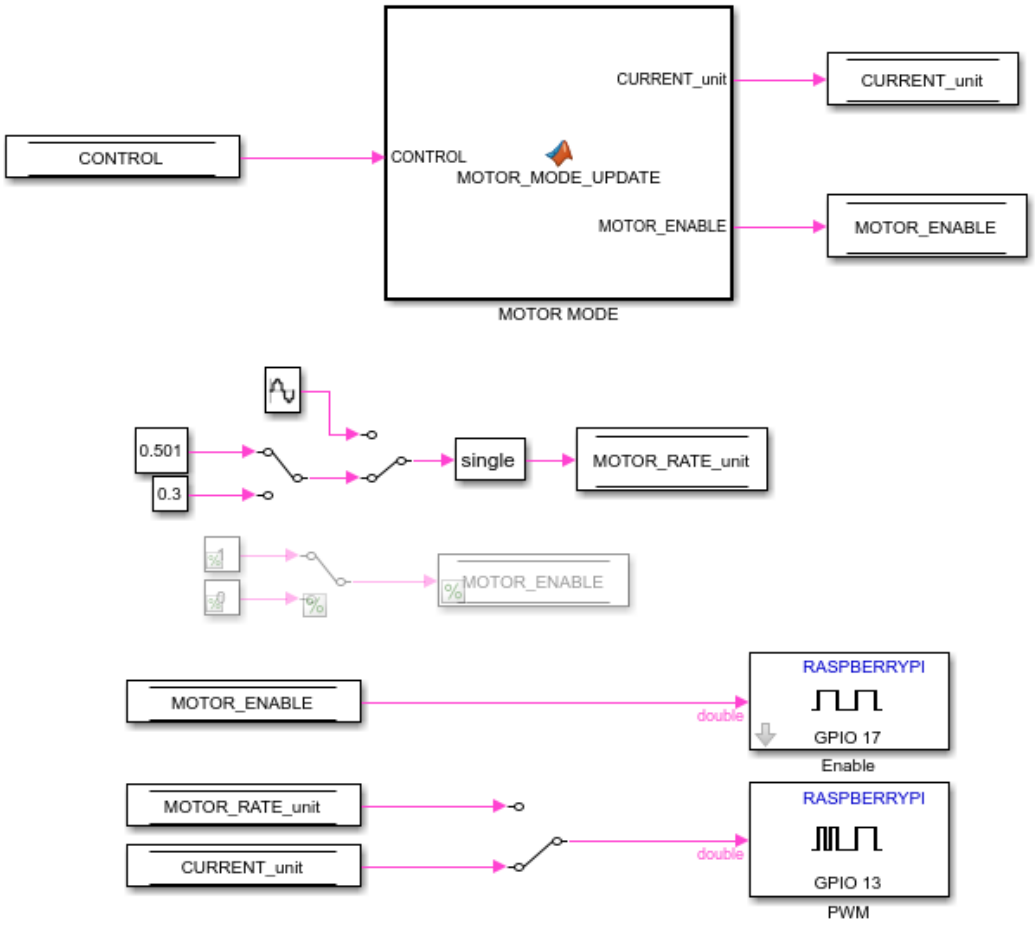


Figura 25: Diagrama Simulink del motor

6.5.2 IMU

El primer paso para utilizar los sensores MPU6050, es su correcto conexionado para el uso del protocolo I2C. Para ello, hay que alimentar los sensores correctamente a 3.3V y conectar los pines SDA y SCL de cada uno a los de la Raspberry Pi. Como los sensores solo tienen un pin de dirección que se pueda modificar, se debe colocar el pin ADD de uno de los sensores a tierra y el del otro a VCC de tal forma que uno de los pines tendrá la dirección 0x69 y el otro la dirección 0x68.

Las medidas proporcionadas por los sensores no se pueden utilizar directamente, es necesario escalarlas para obtenerlas en unidades del SI. En el caso de la obtención del ángulo de inclinación, será necesario escalar las medidas y operarlas como se ha explicado en el apartado *Estimación del ángulo de inclinación.*

En los siguientes diagramas de Simulink es posible diferenciar dos niveles claros. Un nivel inferior donde se realiza la lectura y escalado de las medidas de los acelerómetros y giróscopos, y un nivel superior en el que se lleva a cabo el cálculo del ángulo de inclinación mediante una función Matlab.

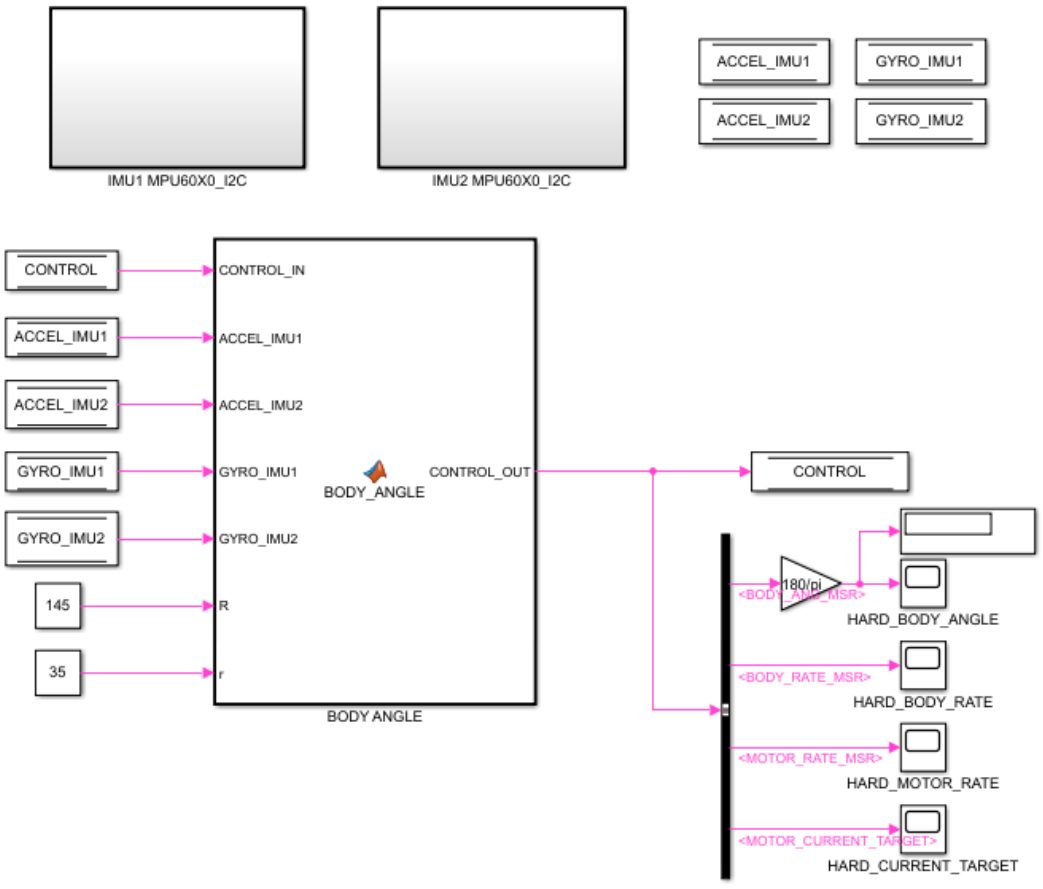


Figura 26: Diagrama Simulink IMU (nivel superior)

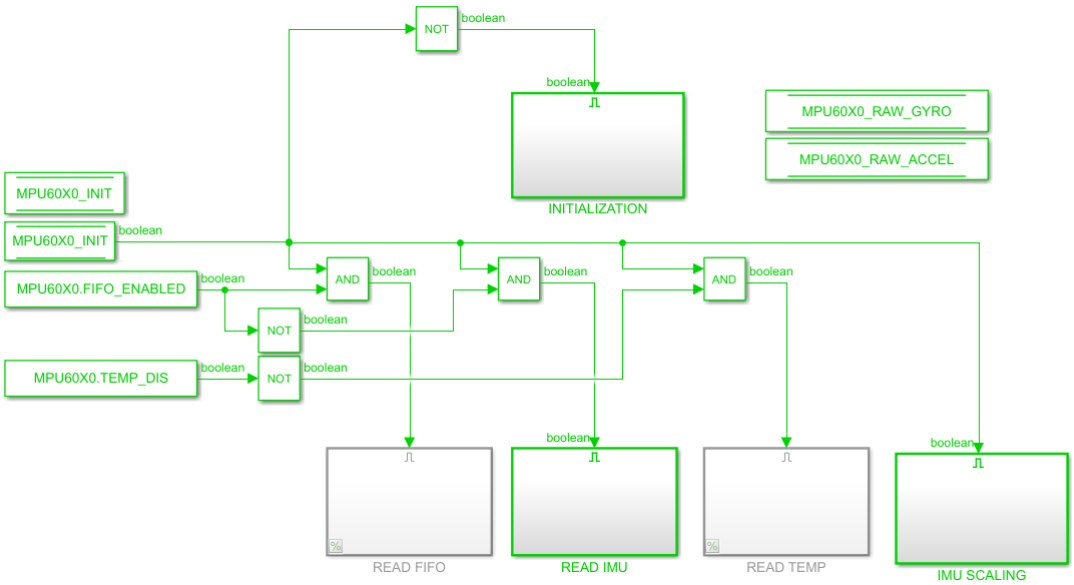


Figura 27: Digrama Simulink IMU (nivel inferior)

6.5.3 ADC

Como ya se ha mencionado anteriormente, el MCP3428 es el encargado de transformar la señal analógica procedente de la controladora ESCON 36/3 en señal digital. La lectura del ADC se realiza en una función Matlab y se transforma de voltios a rad/s antes de que de valor a la variable *MOTOR_RATE_MSR*.

Para poder realizar la implantación del ADC se cambió en primer lugar el POWER SELECT del MCP3428, de la configuración predeterminada (3.3V) a 5V. A continuación, como la dirección I2C predeterminada resultaba ser 0x68, la misma que una de las direcciones I2C de los sensores IMU, se cambió a 0x6E gracias a que el ADC3 *click* permite el cambio de los dos bits menos significativos de la dirección I2C.

A pesar de que solo es necesario transformar una señal, el siguiente diagrama de Simulink permite la lectura de todos los canales del ADC en caso de que en futuros proyectos sea necesario.

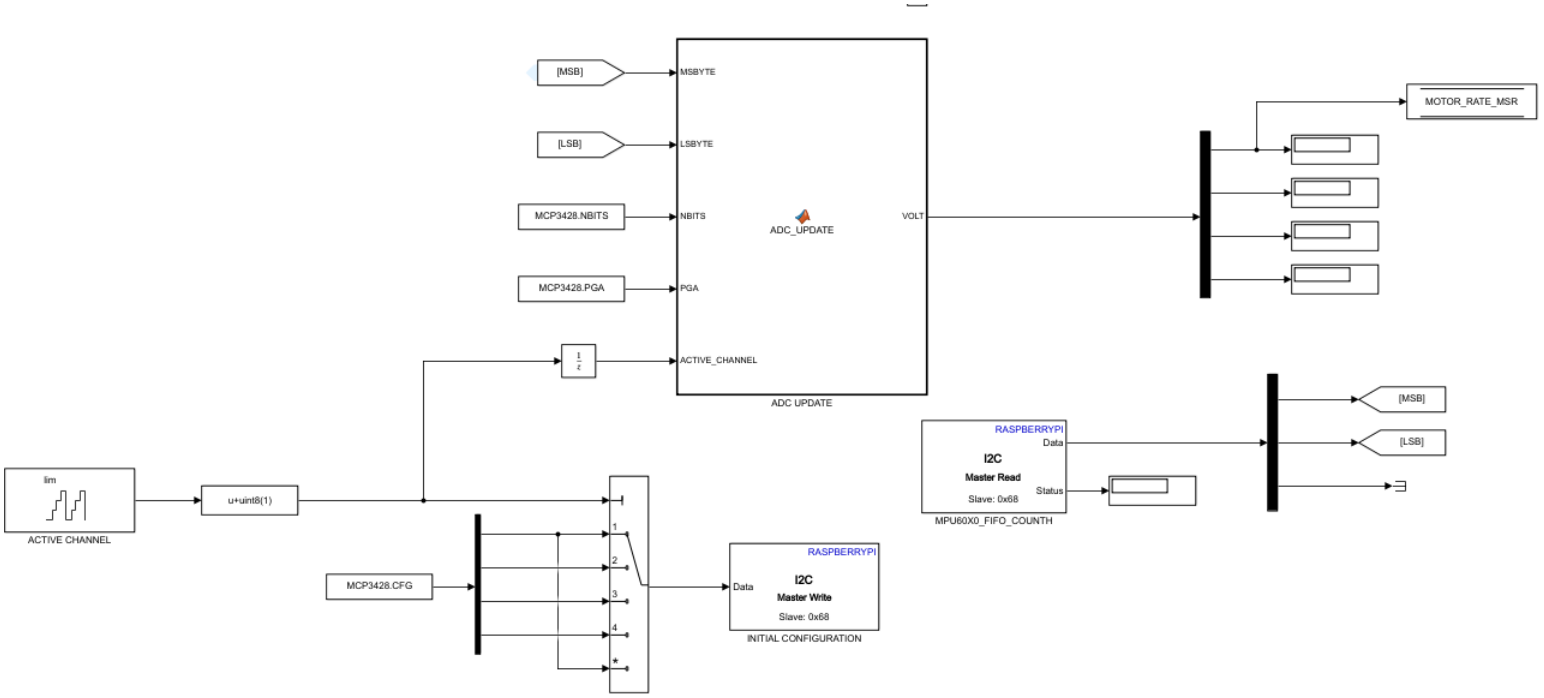


Figura 28: Diagrama Simulink ADC

7. RESULTADOS

7.1 Resultados de la Simulación

Las simulaciones se llevaron a cabo con el cuerpo soltado con un ángulo de 10º respecto al punto de operación (ángulo a partir el cual entra el control de balanceo). Finalmente, se obtuvieron los resultados deseados de simulación con una seta de 0,7 y una pulsación natural de 1,2 veces la del control proporcional.

La simulación se centró en obtener unos resultados apropiados de la velocidad del motor, corriente del motor y velocidad angular del cuerpo, así como un valor final nulo del ángulo del cuerpo. Los resultados se muestran a continuación:

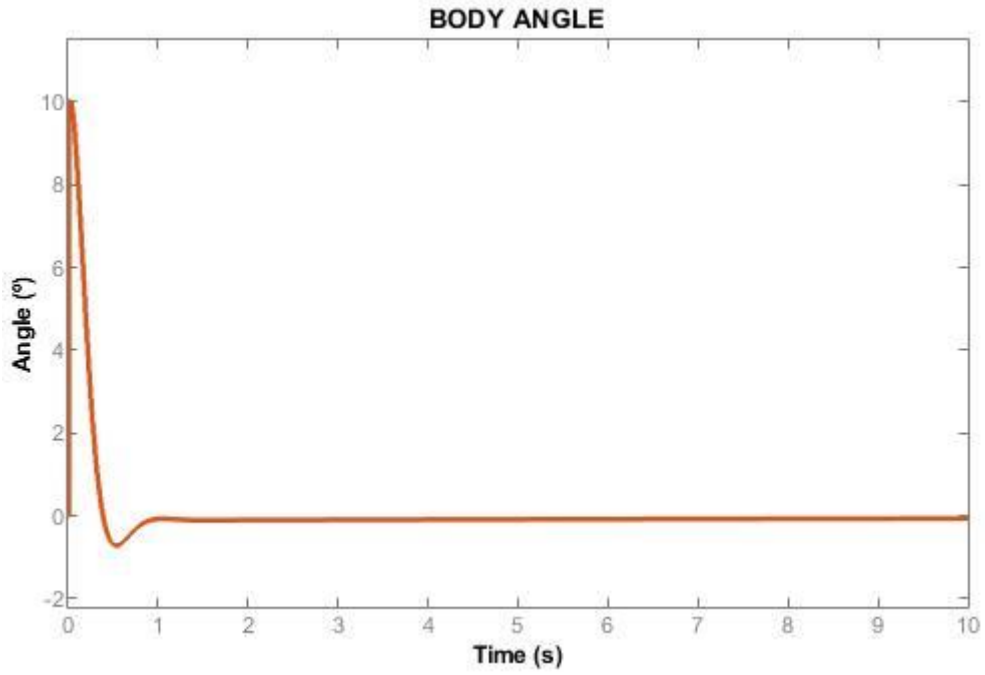


Figura 29: Ángulo del cuerpo en Simulación

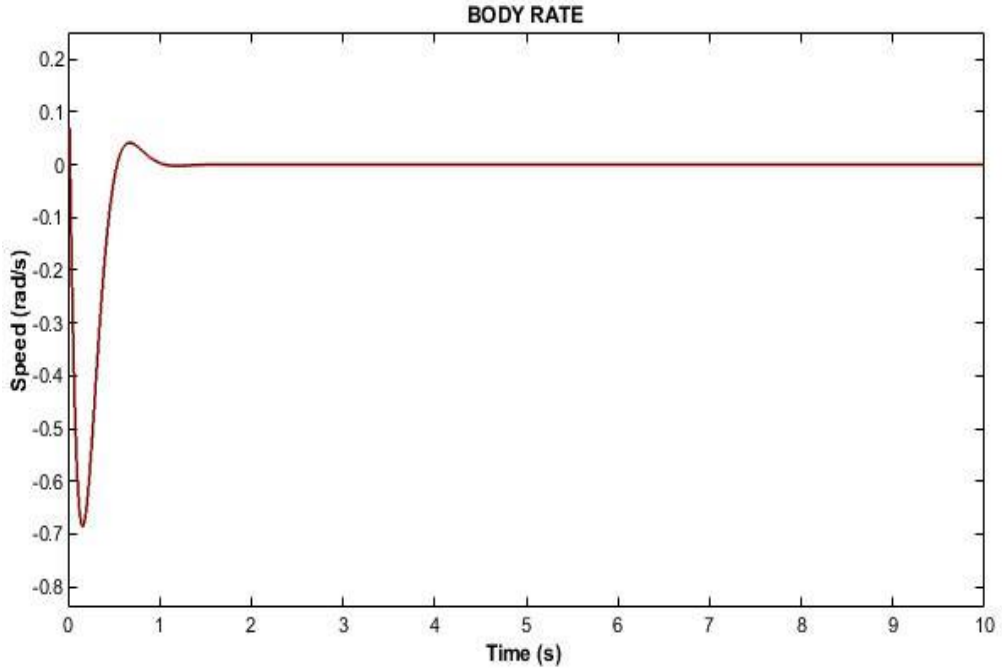


Figura 30: Velocidad del prototipo en Simulación

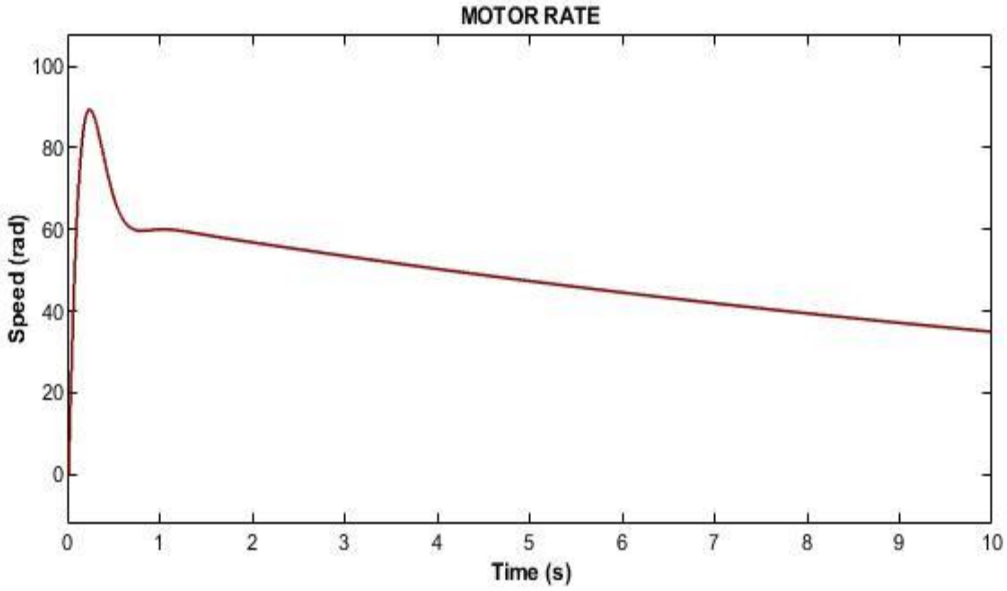


Figura 31: Velocidad del motor en Simulación

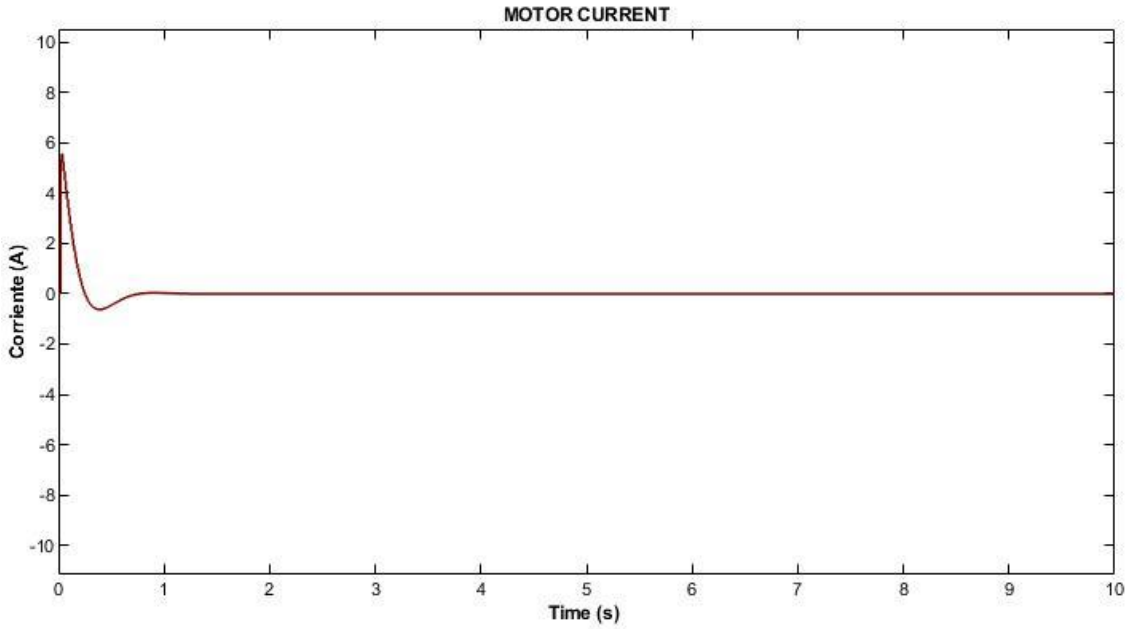


Figura 32: Corriente del motor en Simulación

Como se ha mencionado anteriormente los resultados fueron los deseados ya que se consiguió un valor nulo del ángulo de manera rápida y con un sobrepaso pequeño, que era el objetivo principal.

El sobrepaso inicial del resto de variables de estado es comprensible ya que el ángulo inicial es muy elevado. A pesar de ello, todas las variables reaccionan correctamente. La intensidad del motor se encuentra dentro de las permitida en las especificaciones de este por lo que el control se considera apropiado para su implantación.

7.2 Pruebas de los componentes

En primer lugar, se realizó una prueba de cada uno de los componentes por separado para comprobar su correcto funcionamiento ya que resulta fundamental para la estimación de las variables de estado.

7.2.1 Prueba IMU

La prueba de los sensores IMU consistió en situar el prototipo en las dos posiciones de reposo y en el punto de operación (en los que la diagonal se encuentra a -45º, 0º y 45º respecto a la vertical) y observar las diferencias entre la situación real y la medición. En las siguientes imágenes se muestran los resultados:

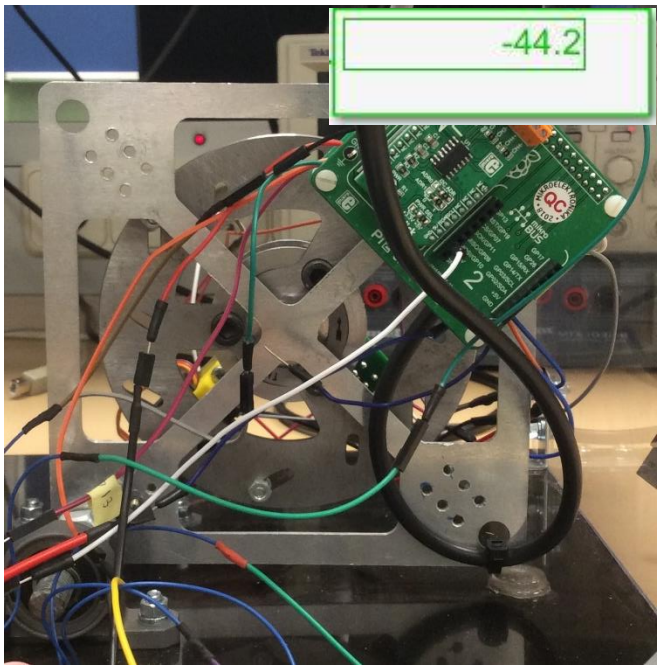


Figura 33: Prueba IMU a -45 grados

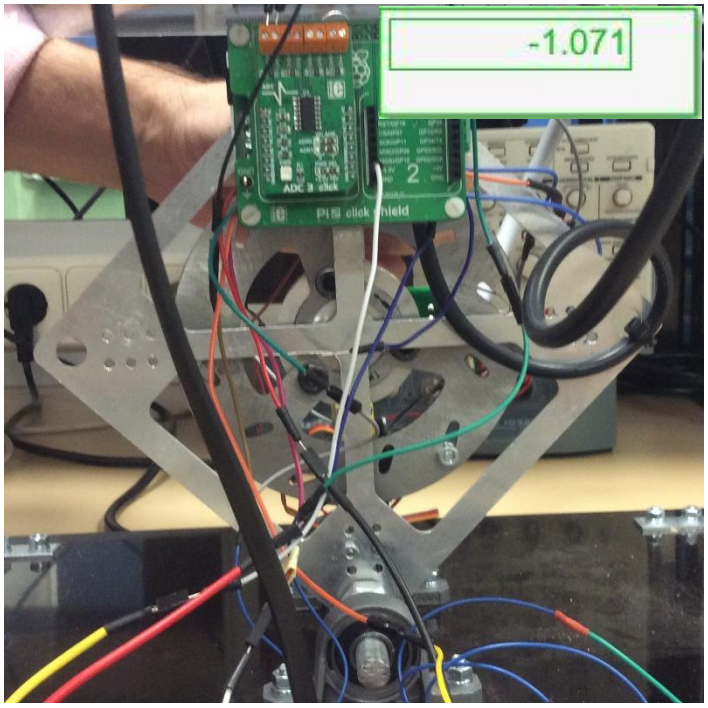


Figura 34: Prueba IMU a 0 grados

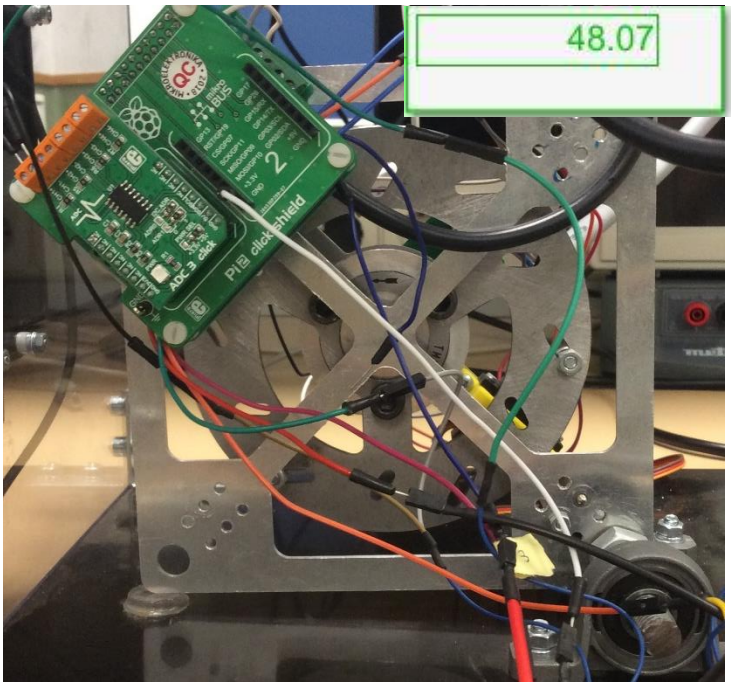


Figura 35: Prueba IMU a 45 grados

La prueba de la IMU permite observar que existe un pequeño error en la estimación del ángulo de inclinación: 0.8 º para -45º, 1.071 para 0º y 3.07 º en el caso de encontrarse a 45º. Las mediciones de las dos primeras posiciones son correctas y el error se considera tolerable al ser tan pequeño. En el último caso el error es considerable, pero se acepta debido a que la goma de apoyo se encuentra desgastada y la diagonal se encuentra en un ángulo algo superior a 45º. Tras la realización de la prueba, se procedió a introducir un offset de 1.071º en la medición del ángulo para que el punto de equilibrio se encuentre a 0º.

7.2.2 Prueba MCP3428

La prueba del ADC, consistió en comprobar que cuando el disco se encuentra frenado, la medición por parte del MCP3428 era cero. También se comprobó que la medición realizada por el programa ESCON STUDIO de la velocidad del motor cuando este se encontraba en movimiento en ambas direcciones era similar a la leída por el ADC.

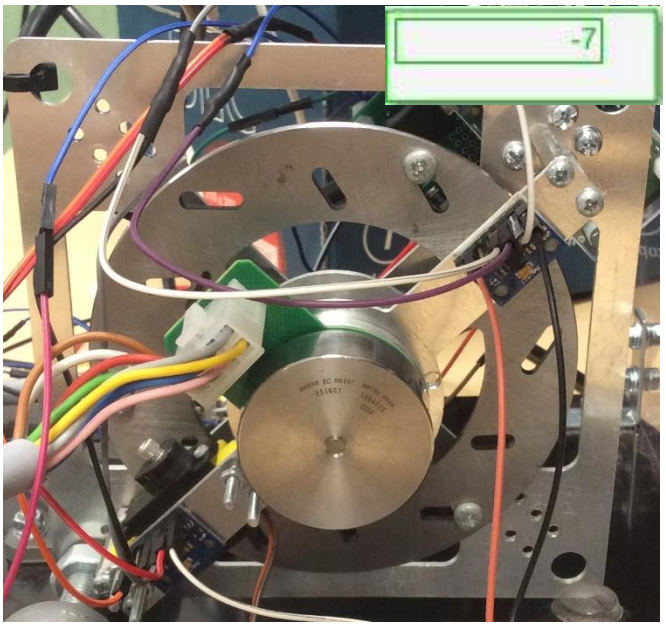


Figura 36: ensayo MCP3428 con el motor parado

Se puede apreciar como al igual que con la prueba de la IMU, existe un pequeño error en la medición de las velocidades. En este caso es de -7 rpm y de la misma manera que en la medición del ángulo, se le añade un offset de 7 rpm.

7.3 Resultado de la implantación

Una vez se han comprobado los diferentes componentes por separado, se realizó la implantación del control sobre el sistema. Debido a que no se dispone de un sistema de frenado para poder realizar el salto, se tuvieron que realizar dos ensayos diferentes, uno para cada control.

7.3.1 Ensayo aceleración

El objetivo de este primer ensayo es comprobar que la máquina de estados funciona correctamente en los primeros estados y que el prototipo soporta la etapa de aceleración hasta alcanzar 3000 rpm. El resultado del ensayo es el siguiente:

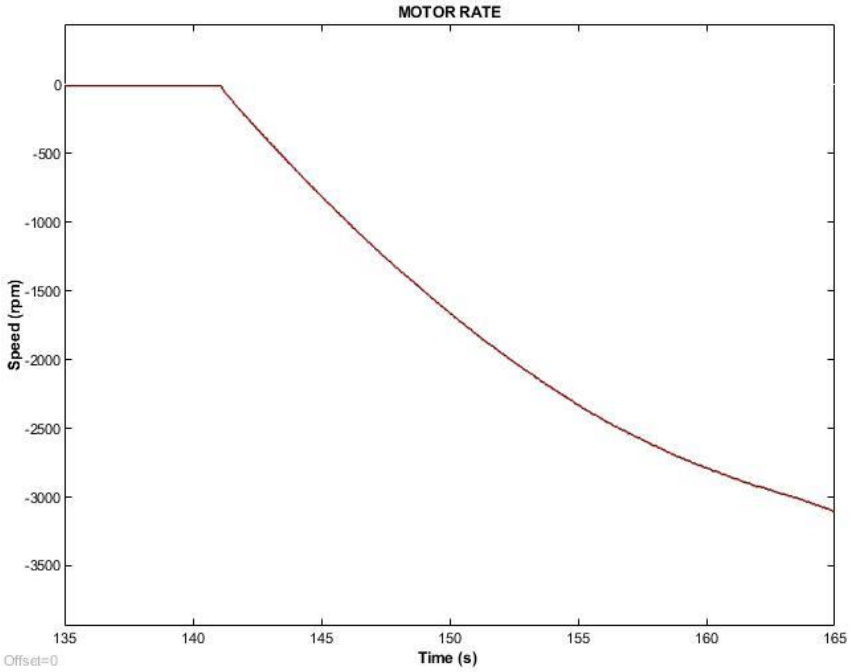


Figura 37: Ensayo de Aceleración

Se puede observar en los resultados como el prototipo superó las 3000 rpm alcanzando así la velocidad marcada para la activación del freno. Hay que destacar, que se amplió el tiempo de aceleración debido a las vibraciones que el prototipo generaba en un principio.

7.3.2 Ensayo control de balanceo

El segundo ensayo tiene como objetivo probar el funcionamiento del control de balanceo sobre el punto de operación. En este caso, una vez nos encontramos en estado de calibración, al pulsar el botón, el motor no comenzará a acelerarse, sino que entrará en un estado de transición para colocar el prototipo en la posición de equilibrio. Una vez alcanzada dicha posición, se pulsará el botón para que se active el control de balanceo. Los resultados del ensayo son los siguientes:

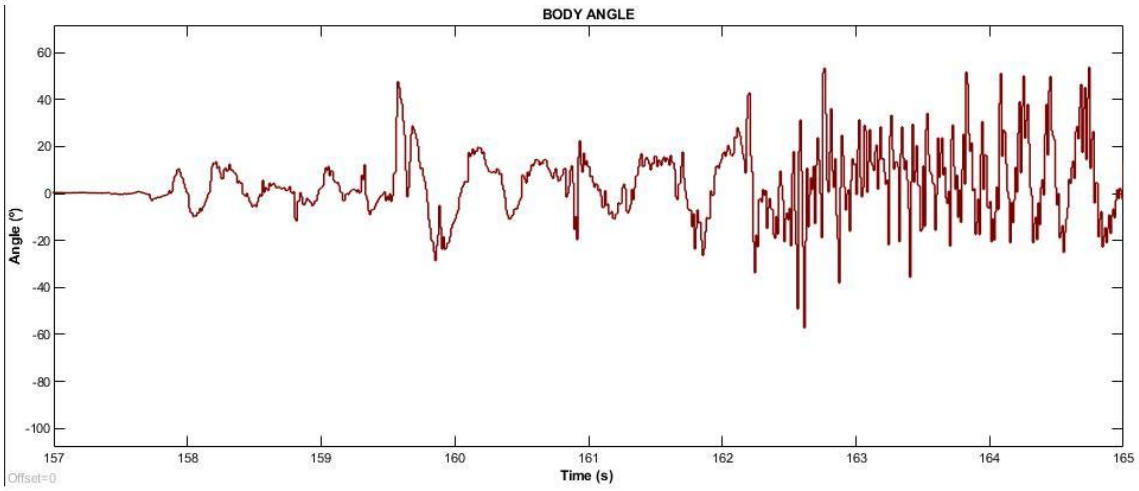


Figura 38: Ángulo de inclinación ensayo balanceo

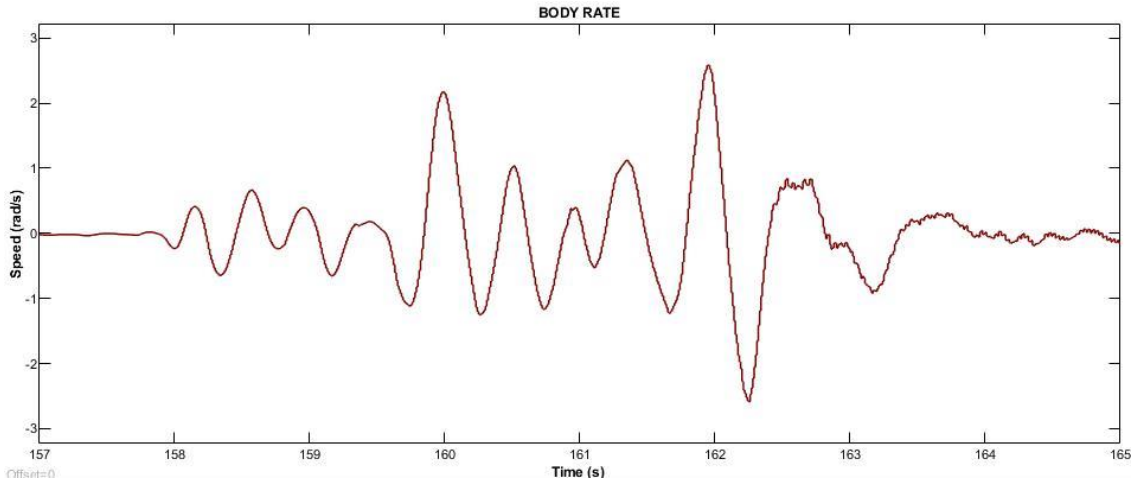


Figura 39: Velocidad del prototipo ensayo balanceo

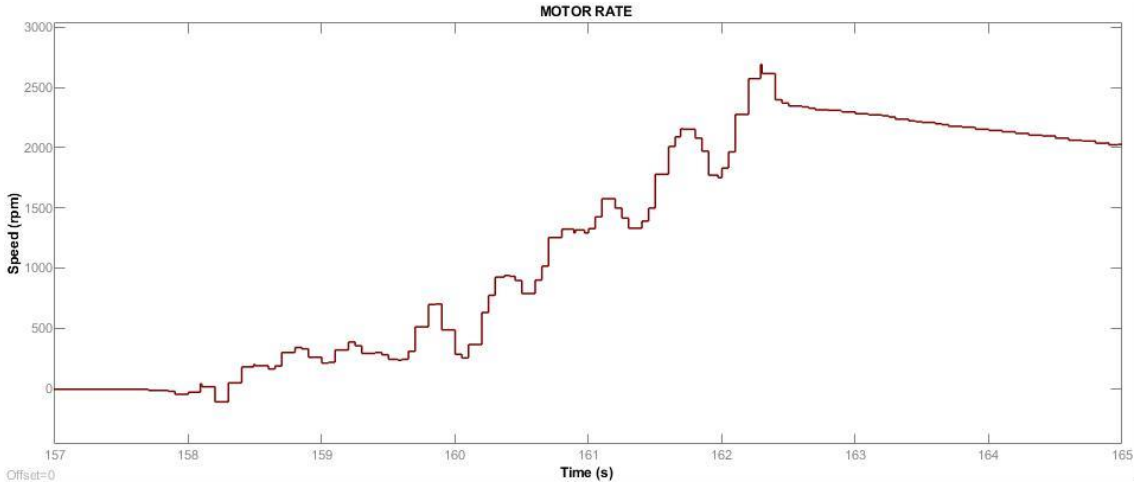


Figura 40: Velocidad del motor ensayo balanceo

Los resultados del ensayo no fueron los deseados. Al producirse las aceleraciones angulares necesarias sobre el volante de inercia, este producía vibraciones que afectaban a las medidas de las IMUs. Estas modificaciones en las medidas de las IMUs, afectaba directamente a la estimación del ángulo de inclinación, provocando que el control de balanceo no fuese capaz de estabilizar el sistema.

8. CONCLUSIONES Y RECOMENDACIONES

El objetivo final del proyecto era elaborar un sistema de control que permitiese a la cara de un cubo balancearse sobre uno de sus vértices. El control de balanceo resultó ser deficiente e incapaz de mantener al prototipo en el punto de operación debido a las vibraciones producidas por el volante de inercia.

Sin embargo, la revisión y actualización del anterior proyecto se realizó con éxito. Se actualizó el código de Matlab y Simulink al nuevo formato. Se mejoró la estabilidad al cambiar la estructura del prototipo y, se consiguió acoplar a este todos los componentes del sistema con sus respectivos conexionados. Esto resulto posible gracias a que se simplificó el sistema al sustituir el Arduino nano por el ADC3 *click* y la fuente de alimentación con un elevador de tensión por la nueva alimentación. Además, con la nueva alimentación se solucionaron los problemas de potencia que se habían dado en el proyecto anterior y que impedían el correcto funcionamiento del motor.

Por otro lado, también se consiguió la estimación de todas las variables de estado, tanto de manera independiente como de manera conjunta para el control de balanceo. El control de salto se preparó para que, en futuros desarrollos, con un sistema de frenado eficaz, se pueda realizar.

En cuanto a posibles mejoras del proyecto, destacaría la importancia de conseguir reducir las vibraciones producidas por el volante de inercia. La modificación de la estructura del prototipo realizada mejoró, pero no resultó suficiente. Algunas de las posibles soluciones pasarían por incorporar algunos amortiguamientos a los sensores o mejorando el control para que el motor no arranque de manera tan brusca.

Otra de las posibles mejoras sería modificar el modelo del sistema ya que el momento de inercia y peso de los componentes acoplados a la cara de aluminio no se tienen en cuenta en él.

9. ANEXOS

9.1 Control Tunning

```
function CONTROL = CONFIG_CONTROL(MODEL)

%% GENERAL PARAMETERS
% Control sampling time (s)
CONTROL.PARAM.SAMPLING_TIME = single(MODEL.PARAM.SAMPLING_TIME);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% MODEL PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% GRAVITY (m/s^2)
CONTROL.PARAM.GRAVITY = MODEL.PARAM.GRAVITY;
% MAX and MIN SPEED WHEEL
MOTOR_RATE_MAX = MODEL.PARAM.MOTOR_RATE_MAX;
MOTOR_RATE_MIN = -MOTOR_RATE_MAX;
% MAX and MIN CURRENT
MOTOR_CURRENT_MAX = MODEL.PARAM.MOTOR_CURRENT_MAX;
MOTOR_CURRENT_MIN = -MOTOR_CURRENT_MAX;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% CONTROL_MODE_ANG: 1. CONTROL P         /  2. CONTROL PI /
%                     3. CONTROL PD_ERR   /  4. CONTROL PD_OUT
%                     5. CONTROL PID_ERR  /  6. CONTROL PID_OUT
% ANOTHER VALUE: WITHOUT ANGLE CONTROL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% CONTROL_MODE_MOTOR:  1. CONTROL P         /  2. CONTROL PI /
%                     3. CONTROL PD_ERR   /  4. CONTROL PD_OUT
%                     5. CONTROL PID_ERR  /  6. CONTROL PID_OUT
% ANOTHER VALUE: WITHOUT SPEED CONTROL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
CONTROL_MODE_ANG = 4;
CONTROL_MODE_MOTOR = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% STATE-SPACE CONTROL DESIGN
% STATE: X = [thb(t) wb(t) wm(t)]
% INPUT: U = iref(t)
% SAMPLING TIME
ts = CONTROL.PARAM.SAMPLING_TIME;
% OPERATING POINT
X0 = zeros(3,1);
U0 = 0;
% LINEARIZATION
cd('../../SIMULINK')
[matA,matB,matC,matD]=linmod('CUBLI_MODEL',X0,U0);
cd('../SOFTWARE_COMPONENTS/MODEL')
P=ss(matA,matB,matC,matD);
```

```
% DISCRETE-TIME MODEL
% State space model in discrete time
Pd=c2d(P,ts,'zoh');
% State space matrices
matAd=Pd.a;
matBd=Pd.b;
matCd=Pd.c;
matDd=Pd.d;

% matAd=expm(matA*ts);
% matBd=(matAd-eye(3))/matA*matB;
% matCd=matC;
% matDd=matD;
% Extended matrices for integral control
% matAad = [matAd zeros(3,1) ; -ts*matCd(2,:) eye(1)];
% matBad = [matBd; -ts*matDd(2,:)];

% Natural frequency and damping factor of the dominant plant poles
[wnP,~] = damp(eig(matA))

% Closed loop poles in discrete time
wn=1.2*wnP(2);
seta=0.7;
% Closed loop poles in continuous time
polos_lc=[wn*(-seta+1j*sqrt(1-seta^2)) wn*(-seta-1j*sqrt(1-seta^2)) -
wnP(3)]
polosd_lc = exp(ts*polos_lc);
% Gain matrix design
K = place(matAd,matBd,polosd_lc)

% PID transformation
K_ANG = single(K(1)/K(3));
Td_ANG = single(K(2)/K(1));
K_MOTOR = single(K(3));
N_ANG = single(100);

% Least-squares optimization
% K_ANG = single(-5.3028e+02);
% Ti_ANG = single(1.6062e+01);
% Td_ANG = single(3.4146e-02);
% N_ANG = single(5.6446e+01);
% K_MOTOR = single(1.4676e+00);

%% PID ANGLE CONTROL
% Control mode
CONTROL.PARAM.CONTROL_MODE_ANG = CONTROL_MODE_ANG;
% PID parameters
CONTROL.PARAM.K_ANG = K_ANG;
CONTROL.PARAM.Ti_ANG = single(100);
CONTROL.PARAM.Td_ANG = Td_ANG;
CONTROL.PARAM.b_ANG = single(0);
CONTROL.PARAM.N_ANG = N_ANG;
CONTROL.PARAM.MAX_CONTROL_ANG = single(MOTOR_RATE_MAX);
CONTROL.PARAM.MIN_CONTROL_ANG = single(MOTOR_RATE_MIN);
CONTROL.PARAM.DER_INPUT_ANG = uint8(1);
CONTROL.PARAM.ANTIWINDUP_ANG = true;
% Discrete integral and derivative calculations
CONTROL.PARAM.DER_DISC_TYPE_ANG = uint8(4);
CONTROL.PARAM.INT_DISC_TYPE_ANG = uint8(0);

%% PID MOTOR SPEED CONTROL
```

```
% Control mode
CONTROL.PARAM.CONTROL_MODE_MOTOR = CONTROL_MODE_MOTOR;
% PID parameters
CONTROL.PARAM.K_MOTOR = K_MOTOR;
CONTROL.PARAM.Ti_MOTOR = single(100);
CONTROL.PARAM.Td_MOTOR = single(0);
CONTROL.PARAM.b_MOTOR = single(1);
CONTROL.PARAM.N_MOTOR = single(100);
CONTROL.PARAM.MAX_CONTROL_MOTOR = single(MOTOR_CURRENT_MAX);
CONTROL.PARAM.MIN_CONTROL_MOTOR = single(MOTOR_CURRENT_MIN);
CONTROL.PARAM.ANTIWINDUP_MOTOR = boolean(0);                %Esto era
con uint
CONTROL.PARAM.DER_INPUT_MOTOR = uint8(0);
% Discrete integral and derivative calculations
CONTROL.PARAM.DER_DISC_TYPE_MOTOR = uint8(0);       %Esto era sin uint
CONTROL.PARAM.INT_DISC_TYPE_MOTOR = uint8(0);       %Esto era sin uint

%% INPUT
% MEASURED AND ESTIMATED VARIABLES
CONTROL.INPUT.BODY_ANG = single(0);
CONTROL.INPUT.BODY_RATE = single(0);
CONTROL.INPUT.MOTOR_RATE = single(0);
CONTROL.INPUT.BODY_ANG_MSR = single(0);
CONTROL.INPUT.BODY_RATE_MSR = single(0);
CONTROL.INPUT.MOTOR_RATE_MSR = single(0);
CONTROL.INPUT.MOTOR_RATE_TARGET = single(0);
% SPEED ENABLE
CONTROL.INPUT.SPEED_ENABLE = boolean(0);
% BUTTON
CONTROL.INPUT.BUTTON = boolean(0);

%% OUTPUT
% CONTROL VARIABLES
CONTROL.OUTPUT.MOTOR_CURRENT_TARGET = single(0);

%% STATE MACHINE
% SYSTEM STATUS
% / 0. BOOTING            / 1. CALIBRATION
% / 2. BODY POSTIONING    / 3. ACCELERATION
%/  4. BREAK              / 5. BALANCE CONTROL
CONTROL.STATE.CURRENT_STATUS = uint8(0);
CONTROL.STATE.PREVIOUS_STATUS = uint8(0);
% CONTROL MODE
% / 0. INITIALIZATION / 1. MOTOR MODE
% / 2. BALANCE MODE
CONTROL.STATE.CONTROL_MODE = uint8(0);
% SYSTEM TIME INITIALIZATION
CONTROL.STATE.TIME_UNIX_US_LSB = uint32(0);
CONTROL.STATE.TIME_UNIX_US_MSB = uint32(0);
CONTROL.STATE.TIME_UNIX_US = 0;
CONTROL.STATE.TIME_BOOT_MS = uint32(0);
CONTROL.STATE.TIMER = single(0);
CONTROL.STATE.SAMPLING_COUNT = uint8(0);
% LED MODE
CONTROL.STATE.LED_MODE = uint8(0);

%
CONTROL.STATE.MOTOR_MODE = uint8(0);
CONTROL.STATE.SERVO_MODE = uint8(0);
return
```

PRESUPUESTO

1. Mediciones

1.1 Hardware

COMPONENTES	UNIDADES
Maxon EC45 FLAT Brushless de 50W	1
Raspberry Pi Zero W	1
ESCON Module 36/3	1
ADC3 clcik	1
Fuente de alimentación 24V 160W	1
Fuente de alimentación 5V 20W	1
Disco Mini moto	1
MPU6050	2

1.2 Software

PROGRAMA	UNIDADES
Matlab/Simulink student	1
ESCON Studio	1

1.3 Mano de obra

TAREAS	HORAS
Montaje Hardware	20
Pruebas del Sistema	200
Estudio	100
Memoria	100

2. Precios unitarios

2.1 Hardware

COMPONENTES	€/Unidad
Maxon EC45 FLAT Brushless de 50W	143.36
Raspberry Pi Zero W	10.53
ESCON Module 36/3	145.09
ADC3 clcik	28.83
Fuente de alimentación 24V 160W	104.45
Fuente de alimentación 5V 20W	27.10
Disco Mini moto	7.56
MPU6050	10.54

2.2 Softwareº

PROGRAMA	€/UNIDAD
Matlab/Simulink Student	125.00
ESCON Studio	0

2.3 Mano de obra

COMPONENTES	€/HORA
Montaje Hardware	30
Pruebas del Sistema	50
Estudio	30
Memoria	25

3. Sumas Parciales

3.1 Hardware

COMPONENTES	UNIDADES	€/UNIDAD	COSTE (€)
Maxon EC45 FLAT Brushless de 50W	1	143.36	143.36
Raspberry Pi Zero W	1	10.53	10.53
ESCON Module 36/3	1	145.09	145.09
ADC3 clcik	1	28.83	28.83
Fuente de alimentación 24V 160W	1	104.45	104.45
Fuente de alimentación 5V 20W	1	27.10	27.10
Disco Mini moto	1	7.56	7.56
MPU6050	2	10.54	21.08

3.2 Software

PROGRAMA	UNIDADES	€/UNIDAD	COSTE (€)
Matlab/Simulink Student	1	125.00	125.00
ESCON Studio	1	0	0

3.3 Mano de obra

COMPONENTES	UNIDAD	€/UNIDAD	COSTE (€)
Montaje Hardware	20	30	600
Pruebas del Sistema	150	50	7500
Estudio	50	30	1500
Memoria	80	25	2000

4. Presupuesto general

CONCEPTO	COSTE (€)
Hardware	488.00
Software	125.00
Mano de obra	11.600
Total	**12.213**

10. Referencias

[1] J.Aracil, «El péndulo invertido: un desafío para el control no lineal,» *Revista Iberoamericana de Automática e Informática Industrial,* vol. 2, nº 2, pp. 8-19, 2005. Available: https://recyt.fecyt.es/index.php/RIAII/article/download/57074/34847. [Último acceso: Agosto 2019].

[2] M. Gajamohan, «The Cubli: A Cube that can Jump Up and Balance,» 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6385896. [Último acceso: Agosto 2019].

[3] M. Antonio-Cruz, «Sistema mecánicos subactuados: Péndulos invertidos,» Instituto Politécnico Nacional, CIDETEC, Boletín nº 41, 2014. [En línea]. Available: http://www.boletin.upiita.ipn.mx/index.php/ciencia/553 -cyt-numero-41/840-sistemas-mecanicos-subactuados-pendulos-invertidos1. [Último acceso: Agosto 2019].

[4] J. García, «Las leyes de Newton en el modelado y control del péndulo invertido sobre un carro,» Revista de Tecnología e Innovación, Vol.3, No. 9, pp. 11-19, 2016. [En línea]. Available: http://www.ecorfan.org/bolivia/ researchjournals/Tecnologia_e_innovacion/vol3num9/Revista_Tecnologia_e_Innovacion _V3_N9_16_2.pdf.. [Último acceso: Agosto 2019].

[5] K. Furuta, «Swing Up Control of Inverted Pendulum,» Tokyo institute of technology, 1991. [En línea]. Available: http://web.mit.edu/jlramos/www/Arquivos/ paper%20furuta.pdf [Último acceso: Agosto 2019].

[6] N.Singh and K.Bhangal, «Robust Control of Double Inverted Pendulum System,» Journal of Automation and Control Engineering, Vol.5, No 1, 2017. [En línea]. Available: http://www.joace.org/uploadfile/2017/0906/20170906024219355.pdf [Último acceso: Agosto 2019].

[7] S. González Mejía, J. M. Ramírez Scarpetta y E. J. Avella Rodríguez,, «Técnicas de control para el balance de un robot bípedo: un estado del arte,» Universidad del Valle, Vol.19, No.43, pp. 139-162, 2015. [En línea]. Available: http://www.scielo.org.co/pdf/tecn/v19n43/v19n43a11.pdf [Último acceso: Agosto 2019]

[8] B. R. Andrievsky, «Global Stabilization of the Unstable Reaction-wheel Pendulum,» Russian Academy of Sciences. 2011 [En línea]. Available: https://link.springer.com/ content/pdf/10.1134%2FS0005117911090189.pdf.. [Último acceso: Agosto 2019].

[9] J. W. Romanishin, K. Gilpin y D. Rus, «M-Blocks: Momentum-driven, Magnetic Modular Robots,» 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6696971. [Último acceso: Agosto 2019].

[10] M. Pavone, «Spacecraft/Rover Hybrids for the Exploration of Small Solar System Bodies,» NASA, 2014. [En línea]. Available:

https://www.nasa.gov/content/spacecraftrover-hybrids-for-the-exploration-of-small-solar-system-bodies/. [Último acceso: Agosto 2019].

[11] V. A. Blanco, «Control y fabricación de un péndulo invertido mediante volante de inercia,» TFG ICAI, 2018 [En línea]. Available: https://repositorio.comillas.edu/jspui/bitstream/11531/22631/2/TFG%20-%20Arias%20Blanco%2C%20Victor.pdf. [Último acceso: Agosto 2019].

[12] M. Cosenza, «Mecánica Clásica,» Libro, Universidad de los Andes, 2016. [En línea]. Available: https://studylib.es/doc/5113896/mec%C3%A1nica-cl%C3%A1sica. [Último acceso: Agosto 2019].

[13] E. Bjerke y B. Pehrsson, «Development of a Nonlinear Mechatronic Cube,» Master's thesis, Chalmers University of Technology, . [En línea]. Available: http://publications.lib.chalmers.se/records/fulltext/233543/233543.pdf. [Último acceso:Agosto 2019].

[14] I. Mauleón y A. Denial, «El Método Generalizado de los Momentos,» Libro, Universidad de Alicante, Instituto Valenciano de Investigaciones Económicas S.A. [En línea]. Available: https://web2011.ivie.es/downloads/docs/wpasec/wpasec-1995-06.pdf. [Último acceso: Agosto 2019].