ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

ITINERARIO ELECTRÓNICO

CONTROL Y FABRICACIÓN DE UN PENDULO INVERTIDO MEDIANTE UN VOLANTE DE INERCIA

Autor: Víctor Arias Blanco

Director: Juan Luis Zamora Macho

Director: José Porras Galán

Madrid

Agosto 2018

(Esta página se ha dejado en blanco a propósito)

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. VÍCTOR ARIAS BLANCO

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:
DISEÑO E IMPLANTACIÓN 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 …30……. de ……Agosto……... de …2018…….

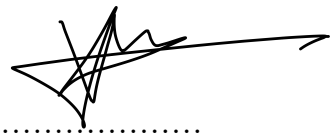
ACEPTA

Fdo……

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

Proyecto realizado por el alumno/a:

Víctor Arias Blanco



Fdo: …… Fecha: 30 / Agosto / 2018

Autorizada la entrega del proyecto cuya información no es de carácter confidencial

Directores del proyecto:

Juan Luis Zamora Macho

Fdo: …… Fecha: 30 / Agosto / 2018

José Porras Galán

Fdo: …… Fecha: ……/……/……

VºBº del Coordinador de Proyectos

Aurelio García Cerrada

Fdo: …… Fecha: 30 / Agosto / 2018

(Esta página se ha dejado en blanco a propósito)

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

ITINERARIO ELECTRÓNICO

CONTROL Y FABRICACIÓN DE UN PENDULO INVERTIDO MEDIANTE UN VOLANTE DE INERCIA

Autor: Víctor Arias Blanco

Director: Juan Luis Zamora Macho

Director: José Porras Galán

Madrid

Agosto 2018

(Esta página se ha dejado en blanco a propósito)

Resumen

Introducción y objetivos

El control que se está llevando a cabo en este proyecto se basa en los principios para controlar sistemas llamados de péndulo invertido. Los péndulos invertidos son sistemas inestables en su punto de equilibrio. La planta que se aborda en el proyecto está englobada dentro de los denominados sistemas subactuados, que están muy presentes en la innovación actual, debido a que, con un control robusto, se puede gobernar su funcionamiento con menos actuadores que los grados de libertad existentes. Ese ahorro en el número de actuadores tiene una correlación directa en la eficiencia energética de los sistemas, ahora mismo uno de los pilares de todo proyecto electrónico.

Centrándonos en la parte más específica, el fin último del proyecto sería la construcción de una cara de un cubo que pueda levantarse desde reposo y equilibrarse mediante el control de un motor unido a un volante de inercia. Este proyecto comprendería tanto la construcción del prototipo como el control al completo en todas sus fases: aceleración, elevación y balanceo.

Metodología

Para llevar a cabo el proyecto hay seguir los siguientes pasos:

- Diseño y montaje del prototipo
- Modelado del sistema
- Diseño e implantación del control

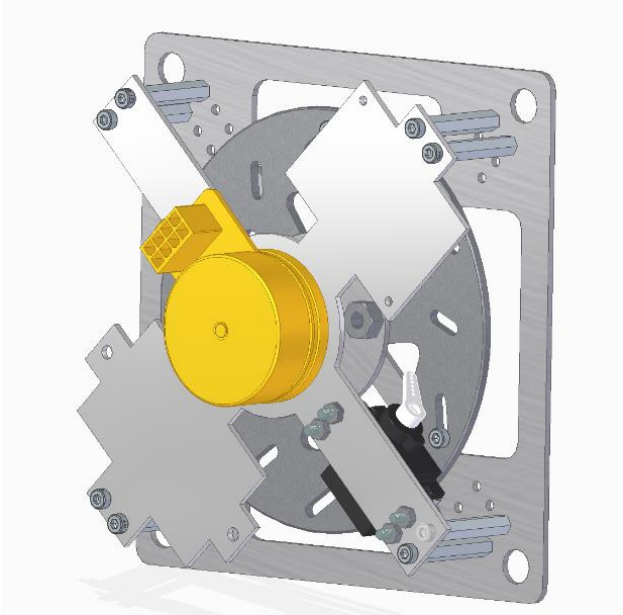
Diseño del prototipo

El diseño y posterior montaje del prototipo tiene un gran impacto en en los resultados finales. Esto es así porque grandes límites y problemas de este proyecto vienen del valor de las masas y la cantidad de vibraciones.

Se ha realizado un diseño para que la estructura sea ligera y así, las fuerzas generadas por el motor y el volante de inercia sean capaces de moverla sin gran dificultad. El tema de las vibraciones es también muy importante porque los sensores de aceleración necesarios son sensibles a éstas. Las mayores vibraciones son generadas por una mal alineación del eje del motor con el volante de inercia, además de por una deficiente fijación. Por estos motivos se ha realizado la estructura en aluminio y con vaciados en las piezas y se ha intentado ejecutar un ensamblaje robusto. El material escogido para la mayoría de las piezas ha sido el aluminio por la ligereza y resistencia, aunque el volante de inercia se necesitó poner de acero

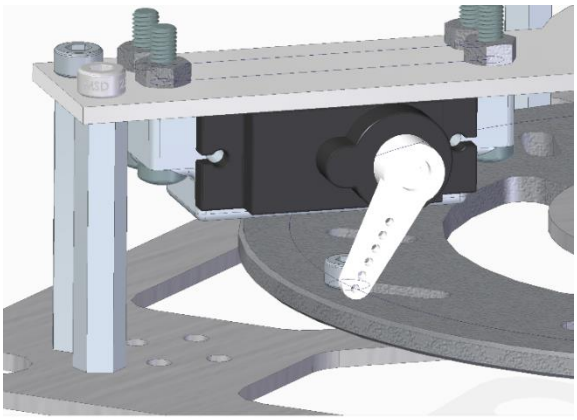
para que compensase todo el peso de los componentes.

La estructura se compone de una cara exterior cuadrangular, y unos soportes para componentes, unidos mediante tornillería. El volante de inercia es un disco de minimoto, debido a la complejidad de generar uno propio suficientemente bien equilibrado.



Diseño estructura y el volante de inercia

El sistema de frenado consta de un servomotor que interrumpe el giro del volante. Para una mayor precisión en el frenado se colocaron unos tornillos en las gavetas del volante. Este sistema sólo se activa en la frenada para levantar la cara hasta una posición cercana a los 90º respecto de la superficie, ya que en el balanceo actuaría únicamente el motor.



Sistema de frenado

El diseño se plasmó mediante el software 3D de SolidEdge debido a que permite tanto el modelado por piezas como un ensamblaje posterior de éstas. Asimismo, facilita la posterior presentación de las piezas con una herramienta para la creación de planos.

En cuanto a los componentes utilizados, han sido necesarios dos IMUs (*Inertial Measurement Unit)* para la captación de las aceleraciones y velocidades angulares de la estructura y los sensores hall presentes en el motor para la velocidad angular del motor. La localización de las IMUs es en los extremos de diagonal del soporte para tener unas medidas redundantes que son necesarias para el cálculo del ángulo. Como actuadores se necesitará un servomotor para el frenado, además del motor unido al volante de inercia, que se sitúa en la parte inferior de la diagonal del soporte. Aparte de estos componentes, será preciso un microcontrolador para el cual se optó por una Raspberry Pi Zero W.

Modelado

Para hacer cualquier prueba, simulación o cálculo que necesite de precisión respecto del sistema real es necesario tener un modelo de dicho sistema. Para ello, y teniendo en cuenta cómo se va a proceder para diseñar el control, se genera un modelado no lineal mediante sus ecuaciones. Las ecuaciones del conjunto se sacan mediante relaciones dinámicas y cinemáticas que describan el sistema mecánico.

Una vez la función con las ecuaciones está preparada, se linealiza en cuatro matrices respecto a un punto de operación. Este tipo de linealización es necesario para el control mediante realimentación de estado, ya que para éste se necesitan en el modelo las variables de estado y sus derivadas. Las variables de estado de este sistema son:

- El ángulo de inclinación, Θ_B
- La velocidad angular del cuerpo, w_B
- La velocidad angular del motor, w_w

Y su punto de operación para el equilibrio sería el valor nulo para las tres.

Diseño del control

Con el modelado previo se puede diseñar un control que cumpla con las especificaciones requeridas para el proyecto. Estas serían que fuese rápido pero amortiguado. La rapidez y amortiguación se deben a que el control debe actuar antes de que se sobrepase el ángulo límite para la recuperación, pero sin un sobrepaso que lo supere por el lado contrario.

Como se ha comentado previamente, se utiliza un control por realimentación de estado, en el que las variables de estado son las citadas en el anterior apartado. Para el diseño se colocaron tres polos en configuración Butterworth de tercer orden, con uno real y dos complejos. Para llegar a la configuración óptima mediante simulación se iteraron estos valores. El resultado de la simulación para los valores fijados y una desviación de 10º respecto del punto de operación es la siguiente.

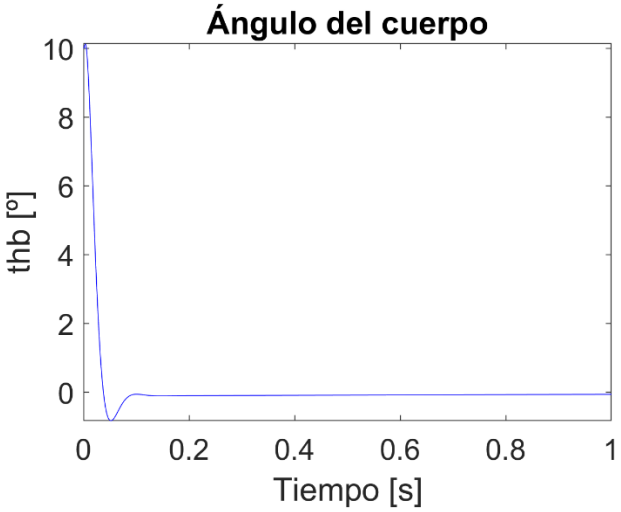


Gráfico del control del ángulo

Otra parte importante dentro del diseño del control es la estimación de variables. Es necesario estimar estas variables ya que las medidas directas de los sensores no se pueden utilizar como entradas, hay que tratarlas u operarlas para que nos den las medidas escaladas de las variables de estado.

Las dos velocidades angulares son prácticamente directas, sólo teniendo que filtrarlas o escalarlas para su correcta utilización. En cambio, el ángulo de inclinación hay que calcularlo a partir de las aceleraciones lineales aportadas por los dos sensores redundantes. La redundancia se debe a que se desea eliminar las aceleraciones generadas por el movimiento de la estructura, y así filtrar las proyecciones de la gravedad que son las necesarias para el cálculo del ángulo.

En estas medidas se genera uno de los grandes problemas de todo proyecto electrónico, el ruido de medida. Al necesitar unas medidas precisas y no tener varios sensores que midan las mismas magnitudes con diferentes métodos, el sistema es muy sensible a estos ruidos. Dichos ruidos afectan tanto al control que puede modificarlo de un sistema totalmente estable y amortiguado a uno prácticamente oscilante si la magnitud del ruido es alta.

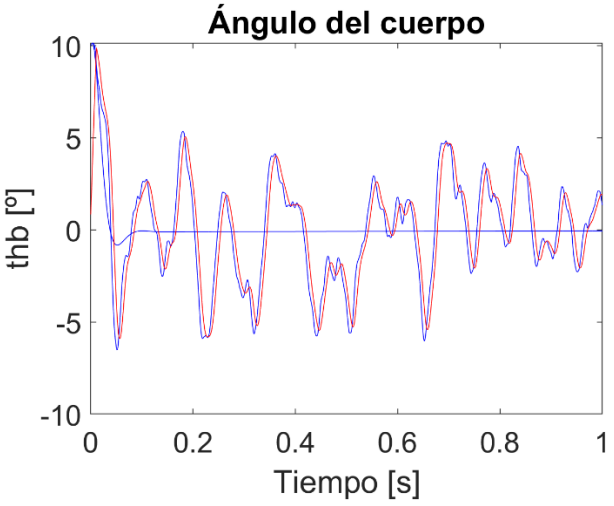


Gráfico control del ángulo con mucho ruido de medida

Por último, también se necesita un control del flujo de estados, ya que el sistema pasa por una secuencia de varios muy diferenciados. Para llevar a cabo esa secuencia dependiente de diferentes estados se ha elaborado una máquina de estados que pasa por los siguientes:

- *STOP:* Este sería el estado de comienzo, en el que el sistema se encuentra en reposo.
- *CALIBRATION:* Al entrar en este estado los sensores comienzan a tomar medidas sabiendo en que posición se encuentran para calibrarse.
- *ACCELERATION:* El motor comienza a acelerar hasta que se alcanza una velocidad definida como óptima para el salto.
- *BREAK:* Se frena en seco el disco generando una elevación de la estructura, que al entrar en rango de balanceo pasa al siguiente estado.
- *BALANCE:* Comienza la etapa de balanceo hasta que se pare el sistema o haya alguna error o perturbación excesiva que haga caer la estructura.

Implementación del control

La implementación del sistema se realiza mediante el software Simulink de la compañía MathWorks. Este programa tiene soporte propio para la familia Raspberry Pi mediante bloques de configuración y actuación. Se ha creado un programa que aúne todos los procesos que son necesarios llevar a

cabo para que el prototipo funcione correctamente, cargándolo en la raspberry directamente desde Simulink. Entre esos procesos se encuentra la lectura de medidas y la actuación sobre componentes, que todo está volcado en dicho programa. Hay una excepción a la congregación de procesos, la controladora del motor tiene un software propio y necesita ser configurada desde ahí, aunque las señales de entrada necesarias se mandan desde la Raspberry.

Las principales señales utilizadas son:

- Consigna del motor mediante PWM
- Señal de habilitación del motor
- Señales de CLK y SDA del protocolo I2C para las IMUs
- Señal PWM para el servomotor
- Puerto serie de la Raspberry proveniente de un Arduino con la medida de velocidad de motor ya digitalizada.

Resultados y conclusiones

En este proyecto se ha logrado diseñar un prototipo de péndulo invertido con la forma de una cara de un hexaedro cuyo fin el levantarse desde reposo y equilibrarse sobre su esquina. En cuanto al sistema mecánico, no se ha podido implantar todo lo diseñado para optimizar las capacidades del sistema, teniendo un método de frenado con falta de robustez y una aceleración que produce algunas vibraciones, pero ya solucionado en los diseños actualizados. En cambio, a se corrigieron desajusten en el sistema de sujeción a la base, en el ensamblado y elección de componentes.

En cuanto al control, se ha dispuesto un software muy esquematizado y potente a la hora de actualizar contenido, así como un modelo adaptativo a la complejidad y precisión de la simulación. El control cumple los requisitos de velocidad para poder estabilizarse dentro del rango de ángulos, superior a otros proyectos similares, resistiendo ruido de medidas sin hacerse inestable.

En definitiva, para una mejora del proyecto existente habría que proceder a aplicar los cambios de diseño ya creados, modificar la adaptación del eje para reducir las vibraciones y realizar pruebas de identificación de parámetros de modelo para precisar los valores obtenidos por otros métodos.

Abstract

Introduction and objectives

The control that is being carried out in this project is based on the principles to control so-called inverted pendulum systems. Inverted pendulums are unstable systems at their equilibrium point. The plant that is addressed in the project is encompassed within the so-called underwater systems, which are very present in the current innovation, due to the fact that, with robust control, its operation can be governed with less ac- tors that the existing degrees of freedom. This saving in the number of actuators has a direct correlation in the energy efficiency of the systems, right now one of the pillars of every electronic project.

Focusing on the most specific part, the ultimate goal of the project would be the construction of a face of a cube that can be raised from rest and balanced by the control of an engine attached to a flywheel. This project would comprise both the construction of the prototype and the complete control in all its phases: acceleration, elevation and balancing.

Methodology

To carry out the project, follow the following steps:

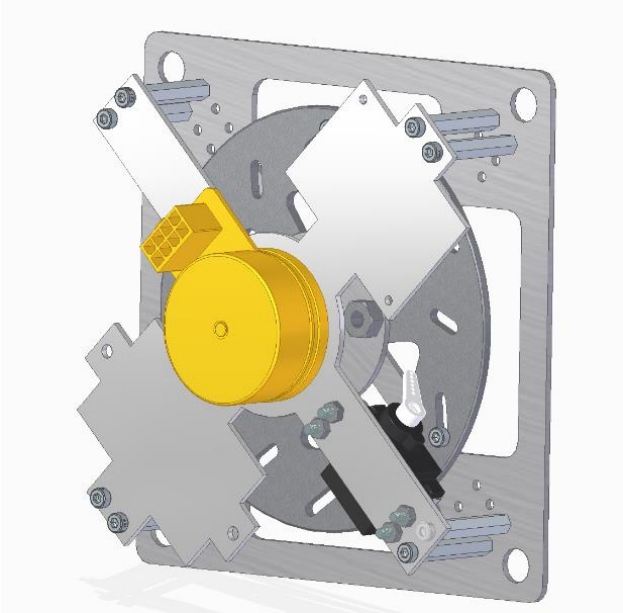
- Design and assembly of the prototype
- System modeling
- Design and implementation of control

Prototype design

The design and subsequent assembly of the prototype has a great impact on the results. This is so because great limits and problems of this project come from the value of the masses and the amount of vibrations.

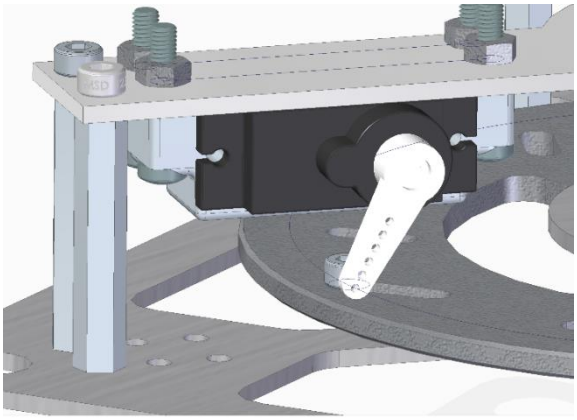
A design has been made so that the structure is light and so, the forces generated by the motor and the flywheel are able to move it without great difficulty. The subject of vibrations is also very important because the necessary acceleration sensors are sensitive to them. The greatest vibrations are generated by misalignment of the motor shaft with the flywheel, in addition to poor fixing. For these reasons the structure has been made in aluminum and with castings in the pieces and an attempt has been made to execute a robust assembly. The material chosen for most of the pieces was aluminum due to the lightness and resistance, although the flywheel needed to be put in steel to compensate for the weight of the components.

The structure is composed of a quadrangular exterior face, and supports for components, joined by screws. The flywheel is a minimotorbike disk, due to the complexity of generating one's own sufficiently well balanced.



Structure and inertial Wheel design

The braking system consists of a servomotor that interrupts the rotation of the wheel. For greater precision in braking, screws were placed in the steering wheel drawer. This system is only activated when braking to raise the face to a position close to 90º with respect to the surface, since only the engine would act on the roll.



Break system

The design was captured using SolidEdge's 3D software because it allows both piece modeling and subsequent assembly. Likewise, it facilitates the subsequent presentation of the pieces with a tool for the creation of plans.

Regarding the components used, two IMUs (Inertial Measurement Unit) were needed to capture the acceleration and angular velocities of the structure and the hall sensors present in the motor for the angular velocity of the motor. The location of the IMUs is in the diagonal extents of the support to have redundant measurements that are necessary for the calculation of the angle. As actuators, a servo motor for braking will be needed, in addition to the motor attached to the flywheel, which is located at the bottom of the diagonal of the support. Apart from these components, a microcontroller will be required for which a Raspberry Pi Zero W was chosen.

Model

To make any test, simulation or calculation that needs precision with respect to the real system it is necessary to have a model of said system. To do this and taking into account how to proceed to design the control, a non-linear model is generated through its equations. The equations of the set are drawn through dynamic and kinematic relationships that describe the mechanical system.

Once the function with the equations is ready, it is linearized into four matrices with respect to one point of operation. This type of linearization is necessary for the control by means of state feedback, since for it the state variables and their derivatives are needed in the model. The state variables of this system are:

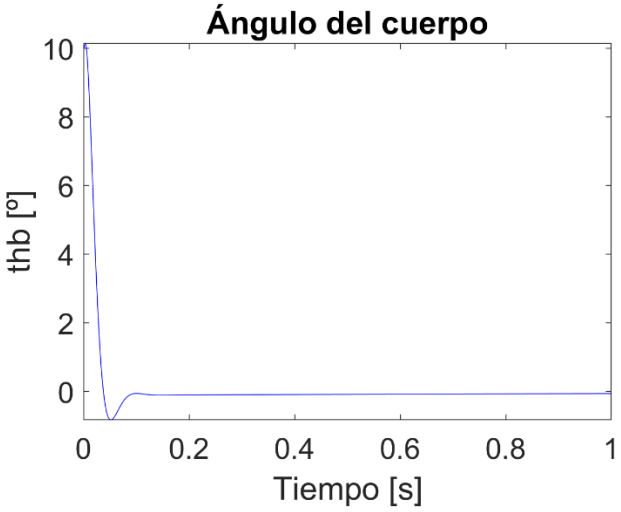
- The angle of inclination, Θ_B
- The angular velocity of the body, w_B
- The angular speed of the motor, w_w

And its point of operation for the equilibrium would be the null value for the three.

Control design

With pre-modeling, a control can be designed that complies with the specifications required for the project. These would be quick but cushioned. The speed and damping are due to the fact that the control must act before the limit angle for recovery is exceeded, but without exceeding it by the opposite side.

As previously mentioned, a state feedback control is used, in which the state variables are those cited in the previous section. For the design, three poles were placed in third-order Butterworth configuration, with a real one and two complexes. To reach the optimal configuration through simulation, these values were iterated. The result of the simulation for the fixed values and a deviation of 10º with respect to the operation point is the following.

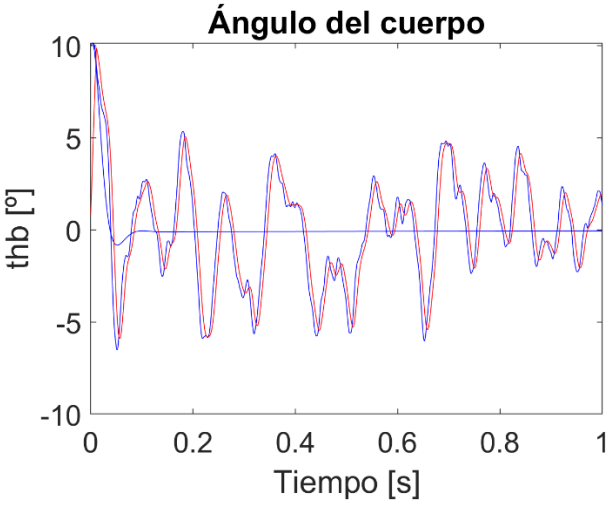


Angle control graphic

Another important part of the control design is the estimation of variables. It is necessary to estimate these variables since the direct measurements of the sensors cannot be used as inputs, they must be treated or operated to give us the scaled measures of the state variables. The two angular speeds are practically direct, only having to filter them or scale them for their correct use. Instead, the angle of inclination must be calculated from the linear accelerations provided by the two re-flooding sensors. The redundancy is due to the fact that it is desired to eliminate the accelerations generated by the movement of the structure, and thus filter the projections of gravity that are necessary for the calculation of the angle.

In these measures, one of the biggest problems of all electronic projects, the meas-

urement noise, is generated. By needing precise measurements and not having several sensors that measure the same magnitudes with different methods, the system is very sensitive to these noises. These noises affect both the control that can modify it from a completely stable and damped system to a practically oscillating one if the magnitude of the noise is high.



Noisy angle control design

Finally, a control of the flow of states is also needed, since the system goes through a sequence of several very differentiated ones. In order to carry out this sequence dependent on different states, a state machine has been developed that goes through the following:

- - *STOP*: This would be the starting state, in which the system is at rest.
- - *CALIBRATION:* Upon entering this stage, the sensors begin to take measurements knowing what position they are in to calibrate themselves.
- - *ACCELERATION:* The engine begins to accelerate until a speed defined as optimum for the jump is reached.
- - *BREAK:* The disc is braked dry generating a structure elevation, which when entering the balance range passes to the next state.
- - *BALANCE:* Start the stage of balance until the system is stopped or there is some error or excessive disturbance that makes the structure fall.

Control implementation

The implementation of the system is done through the Simulink software of the company MathWorks. This program has its own support for the Raspberry Pi family through configuration and action blocks. A program has been created that unites all the processes that are necessary to carry out so that the prototype works correctly, loading it in the Raspberry directly from Simulink. Among these processes is the reading of measures and acting on components, that everything is focused on this program. There is an exception to the process congregation, the motor controller has its own software and needs to be configured from there, although the necessary input signals are sent from the Raspberry.

The main signals used are:

- - Motor setpoint using PWM
- - Engine enabling signal
- - Signals of CLK and SDA of the I2C protocol for IMUs

- - PWM signal for servomotor
- - Serial port of the Raspberry provided with an Arduino with the motor speed measurement already digitized.

Results and conclusions

In this project it has been possible to design a prototype of inverted pendulum with the shape of a face of a hexahedron whose purpose is to rise from rest and balance on its corner. As for the mechanical system, it has not been possible to implement everything designed to optimize the capabilities of the system, having a braking method with lack of robustness and an acceleration that produces some vibrations, but already solved in the updated designs. On the other hand, a mismatch was corrected in the fastening system to the base, in the assembling and choice of components.

As for the control, a very schematic and powerful software has been set up when updating content, as well as an adaptive model to the complexity and precision of the simulation. The control meets the speed requirements to be able to stabilize within the range of angles, superior to other similar projects, resisting measurement noise without becoming unstable.

In short, for an improvement of the existing project it would be necessary to proceed to apply the design changes already created, to modify the adaptation of the axis to reduce vibrations and to carry out tests of identification of model parameters to specify the values obtained by other methods.

Índice de la memoria

Índice de ilustraciones

Índice de ecuaciones

Parte 1: Memoria

(Esta página se ha dejado en blanco a propósito)

Capítulo 1: Introducción

El control que se está llevando a cabo en este proyecto se basa en los principios para controlar sistemas llamados de péndulo invertido. Los péndulos invertidos son sistemas inestables en su punto de equilibrio. La planta que se aborda en el proyecto está englobada dentro de los denominados sistemas subactuados, que están muy presentes en la innovación actual, debido a que, con un control robusto, se puede gobernar su funcionamiento con menos actuadores que los grados de libertad existentes. Ese ahorro en el número de actuadores tiene una correlación directa en la eficiencia energética de los sistemas, ahora mismo uno de los pilares de todo proyecto electrónico.

Centrándonos en la parte más específica, el fin último del proyecto sería la construcción de una cara de un cubo que pueda levantarse desde reposo y equilibrarse mediante el control de un motor unido a un volante de inercia. Este proyecto comprendería tanto la construcción del prototipo como el control al completo en todas sus fases: aceleración, elevación y balanceo.

1.1. Motivación

A pesar de que en un primer momento el control de los péndulos invertidos no tenga una aplicación muy directa en la vida cotidiana, a pesar de que cada vez se abre un mayor espacio para ello, sirve de gran ayuda para proyectos académicos de aprendizaje sobre controles y sensorización. Esto se debe a que, típicamente, necesitan un mayor número de sensores y menos actuadores, aunque cuidadosamente controlados, que otros sistemas mecánicos. Que se utilicen varios sensores y que los actuadores necesiten de un gran control permite que el aprendizaje, tanto sobre componentes y su montaje, como sobre control sea exhaustivo que es el fin del proyecto.

El hecho de que no tengan una implicación directa sobre los desarrollos de mayor actualidad no implica que no la tengan indirectamente, ya que la investigación y perfeccionamiento de controles en este ámbito sirve para aplicaciones de uso común como puede ser el movimiento de robots bípedos o estabilización de cohetes en despegue. A pesar de no estar en la primera plana de los grandes inventos, son vitales para la consecución de muchos otros. Eso sí, como se ha dicho previamente, hay un germen de desarrollo de estos sistemas para utilización directa como puede verse con el segway y sus distintas versiones.

Dentro de los posibles proyectos dentro de los péndulos invertidos se eligió *The Cubli*, que es el prototipo terminado de un cubo que se estabiliza mediante la aceleración de volantes de

inercia. El hecho de que a pesar de su complejidad sea realizable es la atracción del proyecto, que en una menor escala por el tiempo disponible se realizará uno de los tres ejes de éste, siendo aun así desafiante.

1.2. Estado del arte

El diseño y control de sistemas modelados como péndulos invertidos está generalizado en casi todos los ámbitos de la ingeniería industrial y fuera de ésta, como en la robótica o la ingeniería aeroespacial. Pero a pesar de ya formar parte de la médula de una gran cantidad de sistemas complejos, su enunciación y desarrollo comenzó con prototipos muy simples. Por ello vamos a realizar un viaje a través de estos sistemas.

1.2.1. Péndulo invertido sobre carro

Seguramente es el péndulo invertido más simple que se puede crear, dado que sólo consta de un carro con desplazamiento lineal que sostiene la base de un péndulo. Hay un punto de estabilidad con el carro parado y el péndulo a 90º de la horizontal pero cualquier mínima perturbación lo sacaría de ese estado. Para mantenerlo en ese punto, o en cualquier otro dentro de los valores posibles, hay que modificar la aceleración lineal del carro. Para estabilizarlo se compensa la aceleración angular de la masa pendular con la aceleración lineal del carro, teniendo en cuenta que, al acelerar el carro en sentido positivo, la masa se desplazara negativamente y viceversa [1].

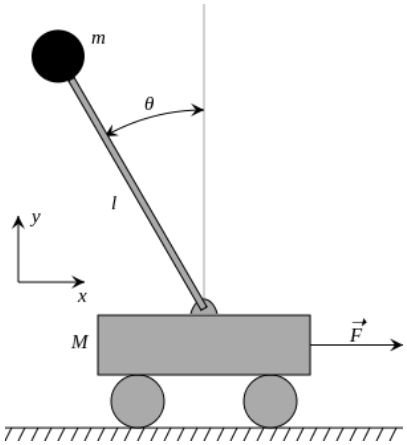


Ilustración 1: Péndulo invertido sobre carro

1.2.2. Péndulo de Furuta

Es un conocido experimento, enunciado y resuelto por el profesor K. Furuta que se asemeja al sistema anterior, pero en este caso la aceleración lineal que se ejerce sobre la base del péndulo para mantenerlo estable viene originada por la aceleración angular de una varilla unida a ésta [2]. Es un experimento más complejo debido a que, por lo general, las aceleraciones angulares cuestan más controlarlas para su estabilización al ser menos intuitivas.

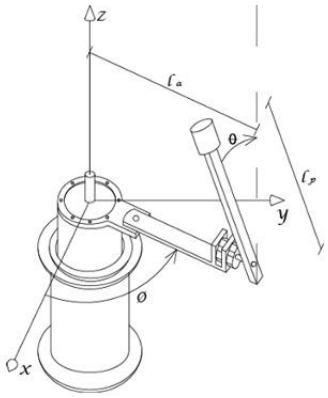


Ilustración 2: Péndulo de Furuta

1.2.3. Humanoides bípedos

Aunque a los humanos el hecho de andar de forma bípeda no aporta gran complejidad, en la robótica sí causa problemas que necesitan solución. Uno de ellos es la inestabilidad del robot cuando levanta uno de sus dos apoyos, por lo que la pierna apoyada se modela como un péndulo invertido permitiendo su estabilidad. Esta estabilización es en ambos ejes paralelos al suelo, ya que se vuelve inestable, necesitando equilibrarse y ajustar la velocidad para que la proyección del centro de gravedad no se salga de la base de apoyo de ese pie si no se está en posición de apoyar el otro [3].

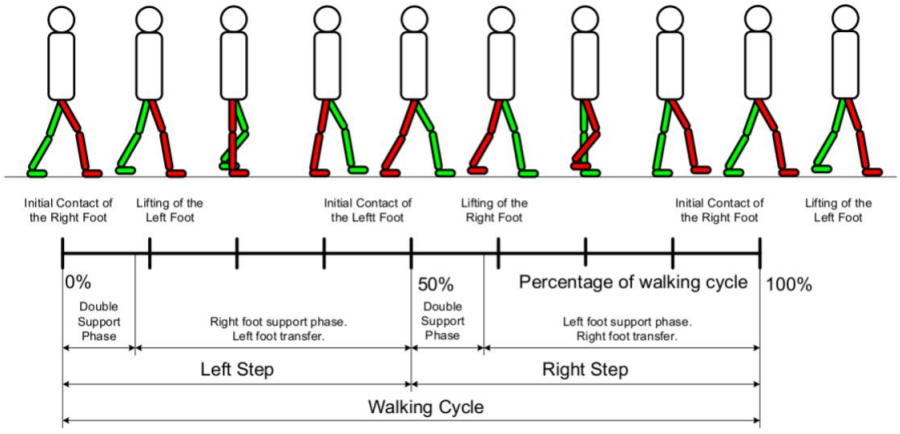


Ilustración 3: Ciclo de movimiento de un robot bípedo

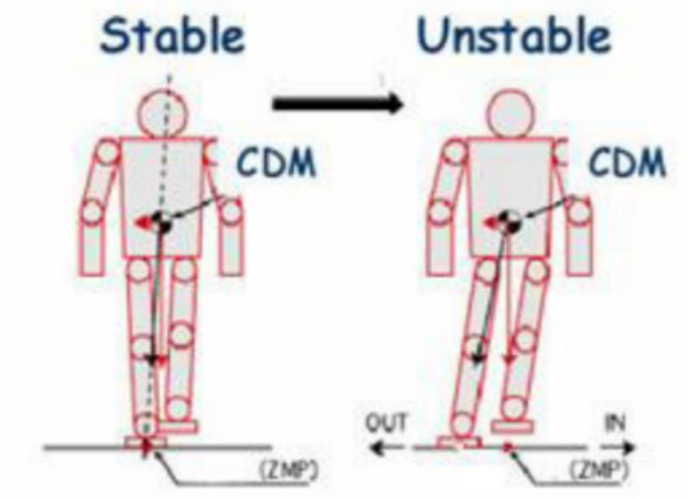


Ilustración 4: Estabilización lateral de un robot bípedo

1.2.4. "The Cubli: A Cube that can Jump Up and Balance"

Este es el sistema que se intenta reproducir en este proyecto con algunas variaciones. Es un proyecto realizado por la ETH Zurich en el que año tras año fueron mejorando el prototipo que comenzó como una propuesta de una sola cara [4]. Bosquejando un poco el funcionamiento, cada cara del cubo consta de un motor que acelera en un sentido u otro un volante de inercia, esta aceleración sirve para generar fuerzas y mantener la cara en la posición deseada, pese a la existencia de perturbaciones. Para llegar a una posición cercana al ángulo que se desea, es necesario acelerar hasta alcanzar una velocidad elevada y frenar bruscamente para que dé un salto antes de comenzar a balancearse. Este sería el fin último de posteriores proyectos.

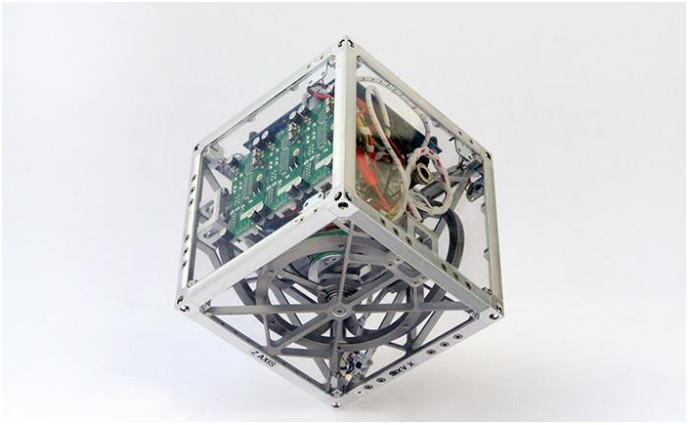


Ilustración 5: The Cubli

1.2.5. Despegue de cohetes

Se va a introducir una aplicación que puede sonar algo sorprendente al decir que en aeronáutica se utilizaban modelos de péndulo invertido. En el despegue de las naves, se requiere que el morro siempre apunte en la dirección calculada para que acabe en la posición correcta, pero las grandes aceleraciones de las toberas no son exactamente simétricas y menos lo son aún las inclemencias del tiempo. Por ello, es necesario realizar un control modelándolo como un péndulo invertido, aunque un tanto más complejo, para mantener la nave estable y controlada [5]. En dicho control, los actuadores son las inyecciones de carburante a los motores o rotando la tobera principal, dependiendo del tipo de cohete.

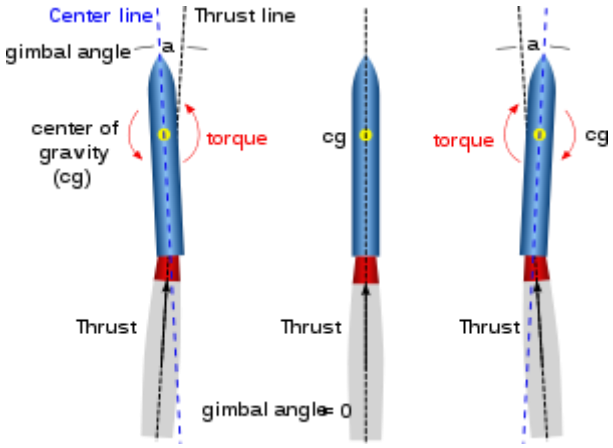


Ilustración 6: Control del despegue de un cohete

1.2.6. "M-Blocks"

Los "M-Blocks" son una propuesta de MIT para crear estructuras auto-ensamblables mediante una inteligencia artificial que controle varios de estos pequeños cubos independientes con un fin concreto: crear la estructura requerida [6]. Aunque por ahora es un proyecto a pequeña escala y de investigación académica, puede ser el comienzo de robots o máquinas industriales de mayores dimensiones, formadas por otros menores y que puedan desempeñar diferentes acciones en función del software instalado.

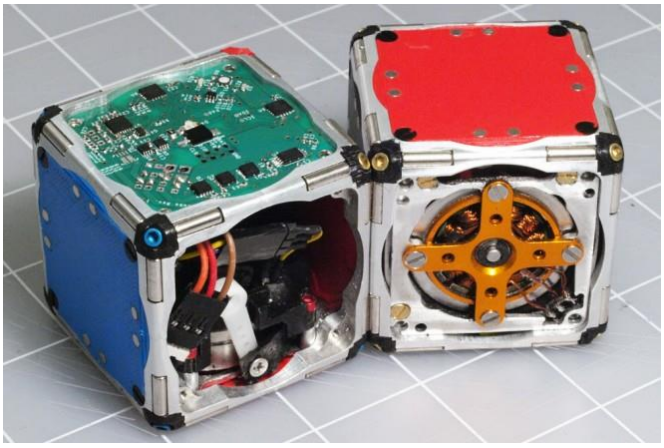


Ilustración 7: M-Blocks

1.2.7. Exploración espacial

También en el ámbito aeroespacial, la NASA está explorando la posibilidad de usar robots cúbicos para la exploración de asteroides debido a la microgravedad de estos astros [7]. El hecho de que todo el par necesario para el desplazamiento de estas estructuras cúbicas es totalmente interno y no dependiente de la fuerza de rozamiento, y por ende de la gravedad, que necesitarían sistemas basados en rodadura. Esta idea viene inspirada por otros mecanismos muy similares al deseado, pero testados en la Tierra, que son "The Cubli" y los "M-Blocks" del MIT, ya mencionados ambos proyectos anteriormente.

1.3. Objetivos

Los objetivos principales del proyecto son los siguientes.

1. Diseño y fabricación del prototipo
2. Modelado de la dinámico de una cara

3. Diseño y simulación del sistema de control
4. Implantación del control

1.4. Metodología

1.4.1. Tareas

Para la plena consecución de los objetivos del proyecto se han realizado las tareas enunciadas a continuación:

- Diseño 3D de las piezas necesarias para la estructura de un eje.
- Impresión o mecanizado de las piezas.
- Montaje del sistema.
- Identificación de parámetros para el modelado.
- Diseño de un control que funcione en simulación con los parámetros obtenidos.
- Puesta a punto de todos los sensores para la correcta adquisición de datos.
- Ensayo iterativo de dichos controles hasta su correcto funcionamiento en el sistema real.
- Ensayo de la elevación mediante frenado súbito.
- Redacción de la memoria.

1.5. Recursos

Las herramientas que se emplearán para la consecución del proyecto incluirán el siguiente equipamiento de software y hardware:

1.5.1. Software

- Solid Edge ST9: Programa de diseño en 3D donde se diseñará el sistema.
- Matlab: Herramienta matemática y de procesamiento de datos con los que se diseñaron el control y las comunicaciones, así como la configuración del programa en general
- Simulink: Herramienta de simulación de Matlab mediante la cual se conectaron todos los campos previamente configurados.
- ESCON Studio: Asistente de configuración de la controladora del motor Maxon.

1.5.2. Hardware

- Motor de corriente continua Maxon EC45 flat 50W que sirve de actuador principal del sistema al ser el que otorga el movimiento del volante
- Controladora de motor ec, denominada ESCON Module 36/3
- Servomotor MG 90S que se le asignará la función de frenado.
- Microcontrolador Raspberry Pi Zero W.
- IMU (Unidad de Medición Inercial) para medir la aceleraciones y velocidades angulares del sistema.
- Convertidor DC-DC que se utiliza para dotar de la tensión necesaria a la controladora, 24V, a partir de la batería existente.
- Batería LIPO de tres celdas que aporta una tensión de 11.3, aunque totalmente cargada supera los 12V.

1.6. Estructura del documento

A continuación, se describirá el contenido de las principales divisiones del documento para su correcta interpretación.

En el diseño del prototipo se explican las partes y procesos necesarios para la construcción del prototipo sobre el que se van a hacer las pruebas del proyecto. Se incluyen los componentes utilizados, su localización y las imágenes del diseño 3D realizado.

El apartado que constituye el modelado es mediante el que se puede simular cualquier tipo de prueba para obtener datos fiables. Es muy importante debido a que no se puede probar el prototipo con un sistema que no sea equiparable a lo simulado, ya que todo lo tenido en cuenta sería falso y peligraría la integridad de los componentes. En este capítulo se incluye desde las ecuaciones dinámicas hasta la identificación de parámetros.

Una vez obtenido el modelo se puede empezar a diseñar el control, que es el capítulo que sucede, para el cual son necesarias las estimaciones de las medidas de las variables de estado. Estas medidas no son directas a partir de los sensores, si no que se tienen que calcular o estimar para que el control pueda funcionar. Una vez son correctas se establece el tipo de control a implantar y las especificaciones de éste, iterando hasta que se determina el óptimo.

En la siguiente sección se trata la implantación de lo simulado en el diseño en componentes reales. Se explica como se utiliza el programa de implantación, que en este caso es Simulink,

con el microcontrolador, los sensores y actuadores. Además de la interacción con los componentes físicos también se exponen los algoritmos usados para el control de balanceo así como de los diferentes estados en los que se puede encontrar el sistema.

La fase de aclaración de los resultados obtenidos de las pruebas hechas sobre el prototipo se localiza como penúltimo capítulo de la memoria. Se exponen los datos y se analiza la validez de éstos según los criterios marcados por los objetivos para llegar a una conclusión en el capítulo siguiente.

Por último, se hace una reflexión sobre el avance del proyecto y los resultados obtenidos del trabajo, así como la aportación personal hacia el proyecto total en el apartado de conclusiones, aportaciones y futuros desarrollos. Además de dichas conclusiones, se presentan los pasos a seguir y errores a resolver en el caso de que se continuase el proyecto.

Capítulo 2: Diseño del prototipo

2.1. Descripción general

El prototipo consta de una estructura metálica que soporta a todos los componentes necesarios para su funcionamiento. La estructura tiene la forma de una cara de cubo, junto con una cara auxiliar, el cual estaría formado por tres conjuntos como este proyecto y que sería el fin de los desarrollos futuros.

Los componentes que se necesitan en este prototipo serán principalmente: un motor, un volante de inercia, los sensores, un sistema de frenado y un microcontrolador. El motor y el volante de inercia serán los causantes de generar las fuerzas para actuar sobre el sistema. Los sensores se necesitan para saber cómo se comporta el conjunto para actuar en consecuencia. El sistema de freno habilita los frenazos bruscos sobre el volante, necesarios para elevar la estructura desde el reposo. Y por último el microcontrolador, que es el cerebro que conexiona y organiza todos los sucesos que ocurren, así como calcula las siguientes acciones a realizar.

2.2. Estructura

Lo primero a realizar es el diseño de la cara donde se localizará el eje con el volante de inercia, siendo ésta compatible para el posterior ensamblaje de otras caras, así como de un soporte para componentes esenciales como el motor, su controladora y el microcontrolador. No todas las caras serían iguales, dado que es necesario que se ajusten a los diferentes componentes, aunque las caras en las que se localizan los volantes de inercia sí compartirían diseño, que es el que se muestra más adelante. Para su diseño se deben tener en cuenta unas premisas, debe ser una estructura ligera y ampliable. Debe ser ligera porque cuanto mayor sea la diferencia entre el volante de inercia, más el motor, respecto al sistema fijo, menor será la potencia necesaria para desplazarlo. También necesita ser ampliable ya que el destino del proyecto es que vaya completándose paulatinamente el control de un cubo completo.

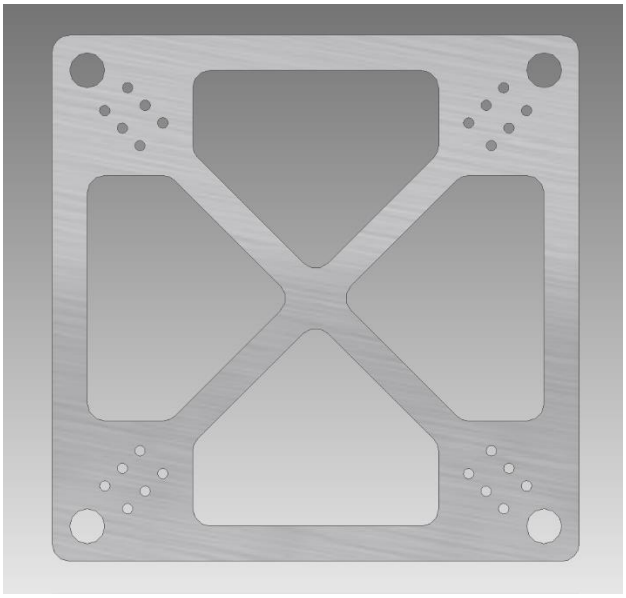


Ilustración 8: Cara de eje del Cubli

Para unirlo con otras caras son los taladros de métrica 10, junto con otra pieza en forma de escuadra que sirve de nexo de unión. Además de para la unión de más caras, uno de esos taladros servirá como eje exterior de un rodamiento, para simular el equilibrio sobre una arista del cubo.

No obstante, esos taladros no son las únicas acciones sobre la pieza, también existen otros de métrica 3 para la sujeción del soporte del motor, y otros componentes, y unos vaciados de material para la reducción de peso.

En el diseño en 3D del proyecto está incluida una cara auxiliar para soportar los componentes que no cupiesen en la cara principal. Esta cara, pese a romper la simetría del conjunto, no tendría tanto impacto en las dinámicas al sostener los componentes poco pesados del prototipo.

2.3. Motor y volante de inercia

Se tratan de las piezas con mayor importancia del proyecto, ya que de ellas parte toda la potencia transmitida al resto del cuerpo. Para optimizar esta potencia se tratará de maximizar el número de revoluciones que pueda alcanzar, alejar el peso del centro de rotación y maximizar la diferencia entre el volante motor y el peso de la estructura y componentes.

Para cumplir estas necesidades, se cree conveniente que esta pieza sea de acero, pero por la imposibilidad de mecanizar una pieza así se dispondrá de un disco de minimoto. El hecho de utilizar este disco nos aporta diversas ventajas: en primer lugar, estará equilibrado ya que era de competición, en segundo lugar, tiene unos surcos exteriores que, a pesar de empeorar el peso y de

no alejarlo del centro de rotación, nos permite generar un sistema de frenado, como veremos posteriormente.

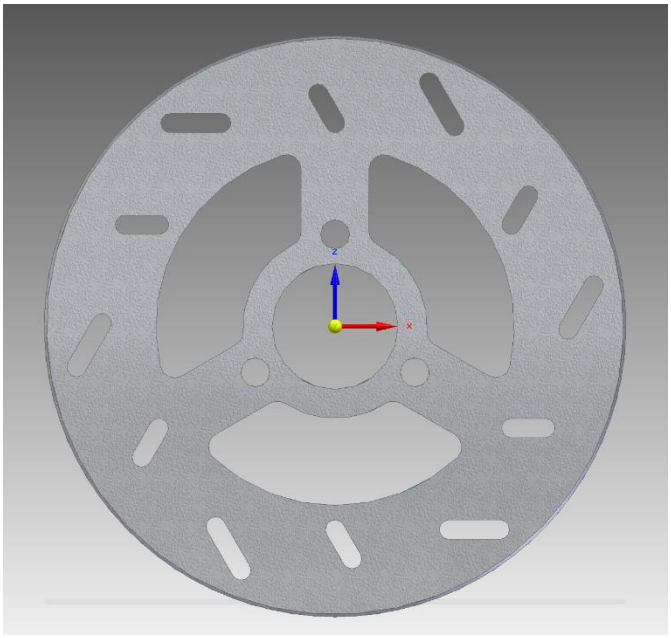


Ilustración 9: Volante inercia eje Cubli

Respecto al motor, el un Maxon EC45 FLAT Brushless de 50W que puede llegar hasta una velocidad de 10.000rpm. Se utiliza un motor de una potencia considerable debido a que, a pesar de no necesitar de ésta para alcanzar una velocidad suficiente para elevarlo, la aceleración sí generará un papel vital para el control del balanceo al ser la aceleración rápida la acción que general la fuerza para este fin.

Ilustración 10: Motor Maxon EC45 Flat 50W

Para acoplar el motor, de eje muy fino, al volante de inercia se necesitarán unas piezas intermedias. La primera es un disco de aluminio que se une al volante y que servirá de apoyo para el acoplamiento “Oldham” que unirá el disco de aluminio al eje. Todas las uniones se realizan

mediante tornillería de distinto metraje, perpendicular al plano del disco, excepto el acoplamiento al eje que se realizara vástagos internos que se aprietan al eje.

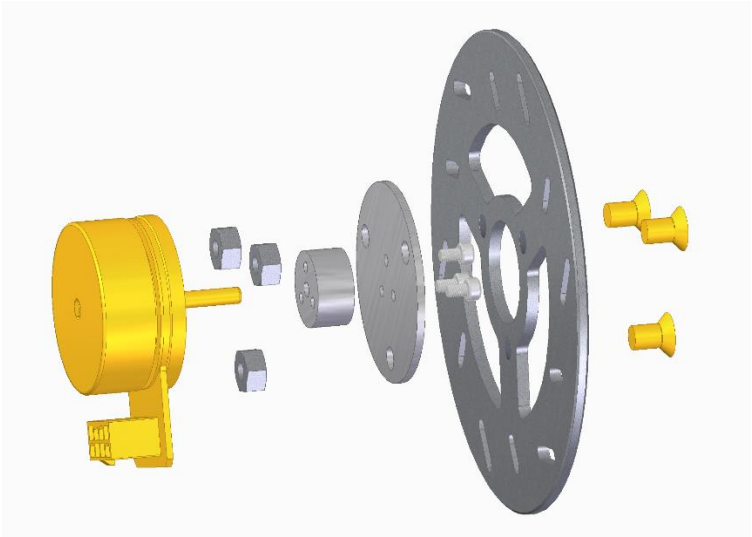


Ilustración 11: Conjunto Motor + Volante

Una vez se tiene la unión de rotor-volante, se necesita un soporte para el motor que canalice la potencia a la estructura y que esta se mueva y venza su peso, esto se diseñara como una pieza de aluminio con unos taladros para el motor y otro para acoplarlo a la cara antes diseñada mediante unos separadores de placas electrónicas.

2.4. Sistema de frenado

Para el sistema de frenado se debatieron varias ideas, desde un sistema de zapatas hasta un bloqueo directo del disco. En cuanto al sistema de zapatas, se concluyó que no era factible debido a la rapidez de frenado que se necesitaba, ya que el frenado por fricción no es brusco. Por ello se decidió que el sistema de bloque era el adecuado.

Una vez se tenía claro el método de frenado, se comenzó a discurrir sobre el diseño exacto a implantar. En un primer momento, se optó por la sencillez dado que la complejidad de los controles y sensores estaba acortando el tiempo facilitado para el proyecto. El diseño más simple fue sujetar un servo a la parte inferior del soporte del motor, desde ahí, al girar la pieza del servo ésta chocaría con los tornillos que se han dispuesto en los agujeros del volante de inercia, frenando en seco el disco. La idea era correcta, pero se subestimó la fuerza ejercida por el disco, que doblaba la patilla del servo. Para solucionarlo se cubrió esta con una pletina de metal, pero esto sólo desplazó el problema hasta el tornillo que sujeta la patilla al servo, que rompía en el choque.

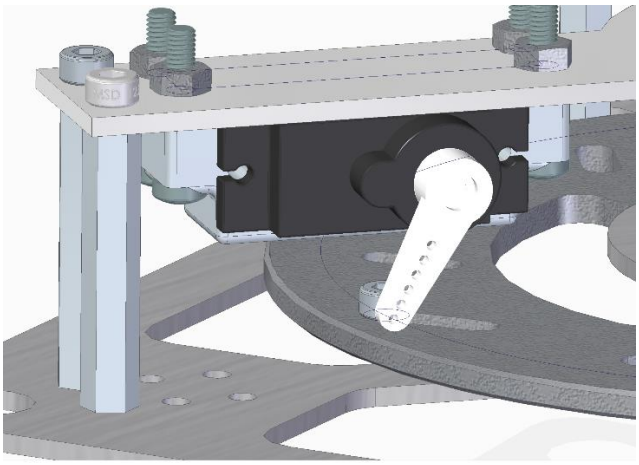


Ilustración 12: Sistema de frenado existente Cubli

Dado que esta solución no era adecuada, y viendo que el principal problema era el contacto directo del tornillo con el servo, se llevó a cabo un diseño más complejo en el que el servo, en la misma posición, sólo desplazaba una pletina de metal y era ésta la que recibiría el impacto. Teóricamente debería funcionar, aunque no se ha podido probar por problemas de tiempo para su fabricación.

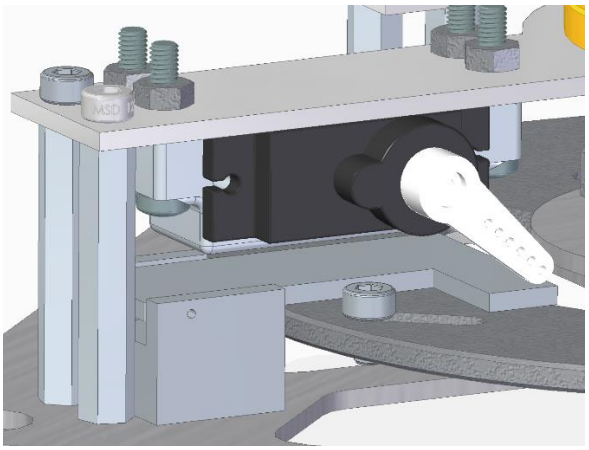


Ilustración 13: Sistema de frenado mejorado Cubli

2.5. Alimentación

Para todo el conjunto se necesitan diversas tensiones de alimentación, pero todas ellas parten de una batería de tres celdas de 11,1V y 1500mAh, capaz de aportar la corriente necesaria al sistema. Para poder aportar las diferentes tensiones a partir de la anterior, se incluye en el conexionado dos convertidores DC-DC, un reductor y un elevador, que nos aseguraran el cambio de los 12,2V, en carga completa de la batería, a dos líneas de 5V y 24V respectivamente.

La línea de 5V alimentará a casi todos los componentes, incluyendo a la Raspberry, al Arduino nano, a los sensores y al servomotor. En cambio, la de 24V se utiliza como alimentación

de la controladora del motor, y por tanto de éste también, que es una tensión alta para temas referidos a la electrónica. En caso de necesitar una línea extra de 3,3V para algún componente, ésta se podría extraer de la Raspberry al incluir un conversor interno y una salida para esta tensión.

2.6. Microcontrolador

La Raspberry Pi Zero W ha sido el microcontrolador que finalmente se ha establecido como definitivo, aunque se pasó por otros previos, mayormente de la familia de los STM32F4. Los motivos por los que se escoge típicamente son principalmente la capacidad de procesamiento, muy superior a los STM, el tamaño y peso, y la multitud de comunicaciones disponibles incorporadas. En cuanto a la capacidad de procesamiento, en este proyecto no ha sido tan vital el cambio ya que, a pesar de su superioridad, el sistema no necesitaba de una ejecución en paralelo de tantas tareas, pero siempre agrada una mayor rapidez para los cálculos de control. El peso y tamaño son una mejora, exceptuando el caso del STM32F45G, más aún si incluimos el rango de posibilidades que nos permite. Las comunicaciones de módulos WiFi y Bluetooth incorporados ayudan a la hora de crear un sistema inalámbrico funcional y de rápida actuación.

Ilustración 14: Imagen componente Raspberry Pi Zero W

Los motivos previos son muy útiles y críticos en otros proyectos, aunque para éste la condición primordial para el cambio fue el soporte directo de Mathworks para el uso de Raspberry con Matlab. Esto se debe a que, terminando de configurar el hardware para un microcontrolador STM, los bloques no oficiales que se utilizan para su manejo no funcionaban correctamente lo que impedía el desarrollo del proyecto. Tras varios intentos fallidos de hacerlo funcionar, se llegó a la conclusión de cambiar a la Raspberry que se usa actualmente, que exceptuando un problema de compatibilidad a la hora de introducir el sistema operativo que hubo que solucionar, al ser el último modelo sacado a mercado, las dificultades que han existido han sido siempre de configuración, que siempre son abordables.

De la Raspberry utilizamos para la comunicación con los componentes su puerto I2C, puerto UART y varios de los pines GPIO (*General Purpose Input Output*) cuyo esquema se puede ver especificado en el anexo. Además de dichos enlaces, se han utilizado también la señal WiFi, el puerto HDMI y el mini UART para su configuración y monitorización. Por internet se hace la monitorización con el ordenador en tiempo real, así como la descarga del programa desde el ordenador al dispositivo. La mini UART y el HDMI se han usado en los estados primigenios del proyecto para abordar la configuración de todas las interfaces de la Raspberry, siendo el HDMI la salida a pantalla y la miniUART la entrada de teclado.

La conexión de los componentes a la placa es la siguiente, los sensores IMU mediante I2C en las direcciones 0x68 y 0x69, el Arduino mediante el puerto serie y el servo y controladora con los GPIO tanto de salida como entrada. A este conexionado habría que añadir la alimentación de todos los componentes que podría ser tanto de 5V como de 3,3V.

En temas de localización, el microcontrolador se situará en la cara auxiliar debido a q su poco peso no alterará en demasía la simetría y el centro de gravedad de la pieza. Se ancalra mediante tornillos a la superficie cuidando de no cortocircuitar los pines de éste con el metal de la estructura.

2.7. Sensores (IMU)

Sensores como tal, sólo se utilizan dos y del mismo tipo, los denominados IMU (*Inertial Measurement Unit*) que se utilizan ara medidas inerciales de los cuerpos a los que van adheridos. Para realizar estas medidas inerciales, dispone de 3 pares de acelerómetro y giróscopo, uno para cada eje. Pero en nuestro sistema no utilizaremos las medidas de todos ellos, si no que sólo tendremos en cuenta las medidas de los acelerómetros en ejes X e Y, y del giróscopo del eje Z, todo en coordenadas locales del sensor. Para este proyecto sea utilizado el modelo MPU6050 que ha sido fiable para otros proyectos similares.

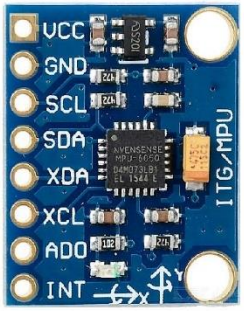


Ilustración 15: Imagen de componente IMU MPU6050

Estos sensores tan útiles, no son de los denominados "*Plug & Play*", que se traduciría como conectar y jugar, y se refiere a que sólo hay que conectarlo y medir. Para que las medidas tomadas sean útiles es necesario configurar previamente el sensor mediante los registros propios a los que accedes por la comunicación I2C que poseen.

En el prototipo se han localizado ambos sensores en la diagonal principal de la cara que ostenta el volante de inercia, en la misma pieza en la que apoya el motor. A pesar de encontrarse en la diagonal, uno de ellos está situado cerca del eje de rotación de la cara, mientras que el otro se encuentra en el externo exterior de la estructura.

2.8. Comunicaciones

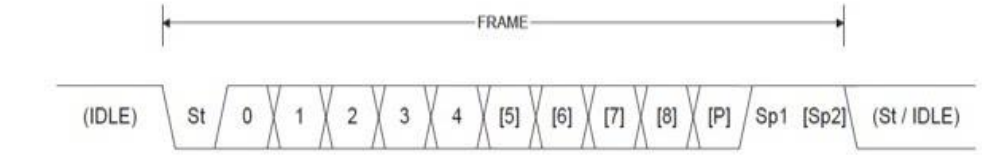
2.8.1. Puerto serie (UART)

UART (*Universal Asynchronous Receiver/Transformer*) controla la interfaz del puerto serie, cuya comunicación destaca por su sencillez y por estar muy generalizado entre los dispositivos cotidianos, a pesar del gran avance de la tecnología inalámbrica. Se argumenta que es un protocolo simple dado que sólo consta de unos bits de configuración al principio (*Start)* y final (*Parity* y *Stop*), siendo el resto bits de información secuencialmente emitidos bit a bit. Los dispositivos tienen unos buffers de recepción y transmisión donde guardan toda esta información antes de introducirla al dispositivo o ser enviada, siendo ambos procesos inversos [8].

El protocolo físico serie que puede síncrono o asíncrono.

Comunicación síncrona: Tanto emisor como receptor utilizan una misma referencia de reloj, por lo que el emisor le envía una señal de comienzo de emisión y a partir de ahí el receptor asume que eil resto son datos hasta que se le envíe una señal de finalización de comunicación.

Comunicación asíncrona: No se utiliza una señal de reloj para sincronizar, en vez de eso ambos dispositivos están previamente configurados para un *baudrate*, paridad de bits y numero de bits de *Stop* específicos. Tiene un protocolo por byte de información definido, con un bit de *Start* para avisar de un envío de información, posteriormente el byte de información y terminando con el bit de paridad y los bits de *Stop*.



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

Ilustración 16: Protocolo Puerto Serie

Las principales ventajas de este protocolo es la necesidad de un único cable en caso de transmisión unidireccional y de un sistema de comunicación sencillo, siendo un inconveniente la velocidad no excesivamente alta.

2.8.2. Protocolo I2C

La comunicación I2C (Inter-Integrated Circuit) es típica entre sistemas donde un dispositivo central (maestro) tiene varios dispositivos subordinados (esclavos) que le aportan información. Es bastante utilizado en estos sistemas por su simplicidad, reducido volumen de cables y gran número de esclavos permitidos [9].

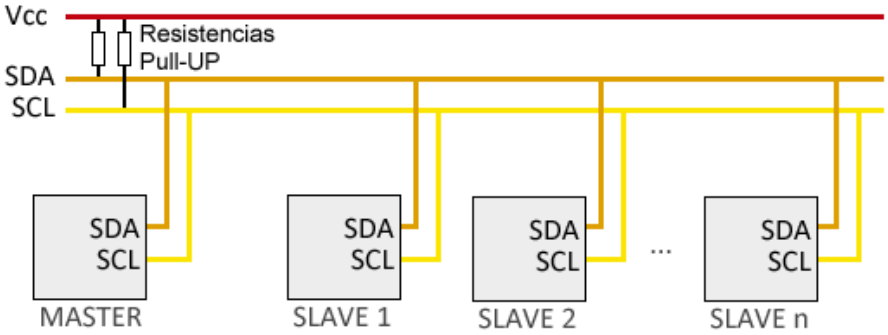


Ilustración 17: Esquema Bus I2C

Este protocolo de capa física se caracteriza por envía datos en serie en paquetes de 8 bits a través del cable de datos (SDA) mientras que por el otro cable necesario envía una señal de reloj (SCL) para sincronizar la comunicación. Ambos canales están conectados en pull-up.

Para la comunicación con los esclavos, éstos poseen una dirección única de 7 bits, existiendo un octavo para definir si la comunicación es de lectura (1) o de escritura (0). El protocolo de las señales se describe a continuación:

1. EL maestro envía un flanco descendiente mientras la señal de reloj está en 1, lo que implica el comienzo de la comunicación, ya que los flancos siempre suelen ser con el CLK a 0.
2. Se envía la dirección del esclavo deseado junto con el bit de escritura/lectura, esperando en *acknowledge* (ACK) del dispositivo seleccionado.
3. Se comienzan a enviar los datos que se precisan en paquetes de un byte, con su ACK correspondiente después de cada paquete para que se confirme la recepción.
4. Se envía un flaco ascendente con el CLK en alto para terminar la comunicación.

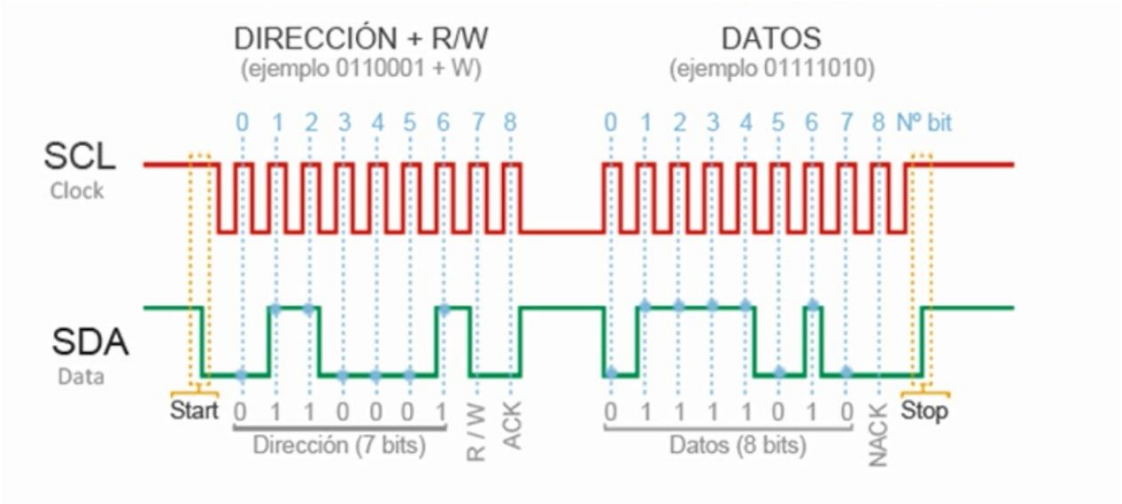


Ilustración 18: Protocolo I2C

2.8.3. Protocolo UDP

UDP (*User Data Protocol*) es un protocolo de capa de transporte del modelo OSI, que no está orientado a conexión, por lo que no proporciona un método de protección de errores, control de flujo ni confirmación de recepción. Es un protocolo muy simple que sólo divide los datos a enviar en segmentos más fáciles de administrar, aportando a cada uno un encabezado con el puerto de origen, el puerto de destino, el número de bytes a enviar y un número de comprobación del encabezado [10].

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

Ilustración 19: Protocolo UDP

Al ser una capa de transporte, necesita un medio por el que propagarse, la capa física, que en este caso es la red a través del modelo WiFi de la Raspberry Pi Zero W.

2.8.4. Protocolo MAVLINK

MavLink es un protocolo de comunicaciones que no está referido a la capa física del modelo OSI, sería una capa de aplicación, que serviría para la organización de los mensajes que se desean enviar. Para ello necesita de otro protocolo de comunicaciones como podría ser Serial, TCP/IP o UDP, siendo este último el utilizado en este proyecto junto con la capa física de WiFi [11].

Para dicha "organización" se establecen unas funciones de codificación y decodificación, típicamente programadas en C, que organizan el mensaje con unos datos a enviar, insertados entre unos bits de encabezado y terminación.

Entre los bits de encabezado se encuentran datos como:

- Inicio
- Longitud: número de bytes del *Payload* (información a enviar)
- Número de secuencia: para reordenar paquetes en caso de pérdida o que sean de gran tamaño
- Identificación del sistema: para casos de varios sistemas
- Identificación de componente
- Identificación de mensaje

Por último, se envía tras el *Payload* un código de comprobación (Checksum), que resulta de un cálculo entre los bits del mensaje, para verificar si el mensaje es correcto cuando se lea en destino.

2.9. Sistema total

Los sistemas previamente explicados, junto con alguna otra pieza auxiliar para el montaje, se unen para formar la totalidad del péndulo invertido, a excepción de algunos componentes que no se han modelado por su complejidad, aunque se han tenido en cuenta sus dimensiones para el diseño. A continuación, se muestra el montaje final de un eje el cubli que será el sistema a controlar en el proyecto.

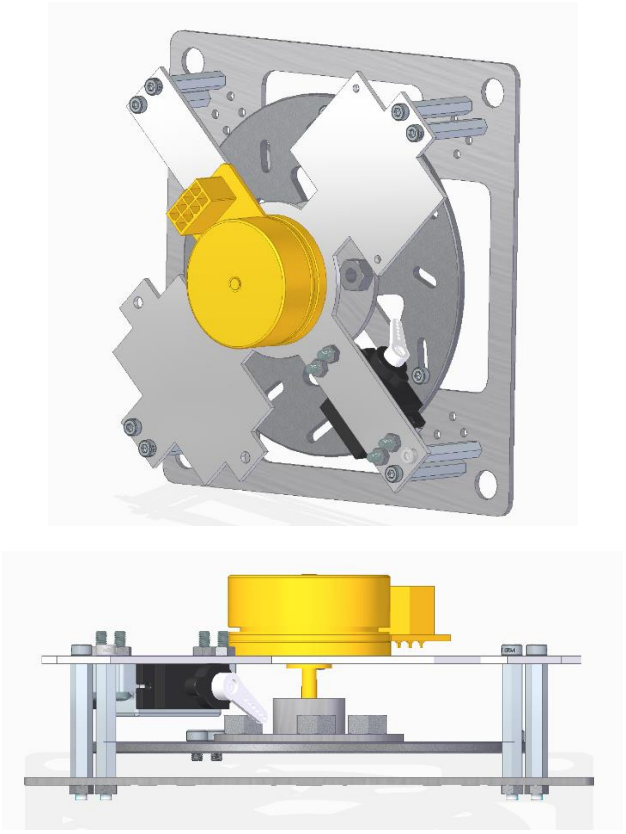


Ilustración 20: Cara completa Cubli

Capítulo 3: Modelado y dinámica del sistema

3.1. Ecuaciones del modelo

Para el planteamiento de las ecuaciones dinámicas del sistema se hace uso de la mecánica lagrangiana. La mecánica lagrangiana es ideal para sistemas con fuerzas conservativas en cualquier sistema de coordenadas, y se plantea en forma de energías potencial (V) y cinética (T) dando la siguiente ecuación:

$$L = T - V \qquad (1)$$

Las fuerzas disipativas e impulsos se pueden tener en cuenta separando las fuerzas externas en la suma de fuerzas potenciales y no potenciales, lo cual lleva a las ecuaciones de Euler-Lagrange [12]. El planteamiento de estas ecuaciones se basa en el usado por el departamento de señales y sistemas de la "Chalmers University of Technology" para su proyecto de Cubli [13].

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} + \frac{\partial R}{\partial \dot{q}_k} = \tau_k \qquad (2)$$

Donde τ_k es los pares de inercia y $\frac{\partial R}{\partial \dot{q}_k}$ modela las fuerzas disipativas.

Sustituyendo los parámetros de las energías potencial y cinética, el lagrangiano se modela como:

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

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
- I_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})

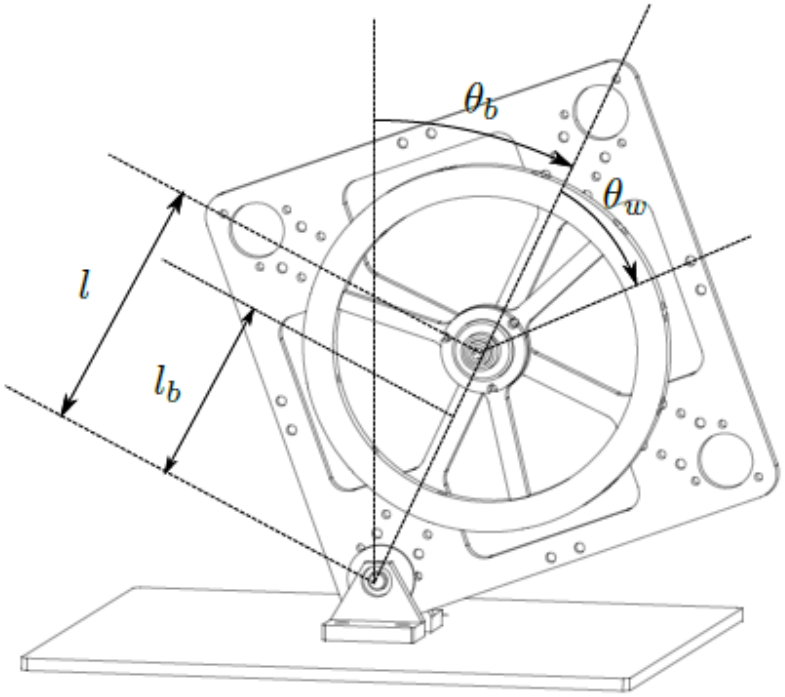


Ilustración 21: Prototipo del péndulo invertido indicando ángulos y distancias [3]

De las derivadas parciales del lagrangiano respecto las velocidades de giro se obtienen los siguientes resultados, los cuales son el primer término de la ecuación de Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_b}\right) = I_b\ddot{\theta}_b + m_b l^2\ddot{\theta}_b + I_w(\ddot{\theta}_b + \ddot{\theta}_w) \qquad (4)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_w}\right) = I_w\ddot{\theta}_b + I_w\ddot{\theta}_w \qquad (5)$$

Por otro lado, derivando el lagrangiano respecto de los ángulos de giro resulta en:

$$\frac{\partial L}{\partial \theta_b} = (l_b m_b + m_w l) g \sin \theta_b \qquad (6)$$

$$\frac{\partial L}{\partial \theta_w} = 0 \qquad (7)$$

Por último, se incluyen los rozamientos y el par motor:

$$\tau_w = Tm \qquad (8)$$

$$\frac{\partial R}{\partial \dot{\theta}_b} = C_b \dot{\theta}_b \qquad (9)$$

$$\frac{\partial R}{\partial \dot{\theta}_w} = -C_w \dot{\theta}_w \qquad (10)$$

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})

3.2. Formulación en espacio de estado

Combinando todas las ecuaciones anteriores y resolviendo para hallar las ecuaciones diferenciales que definan el sistema de manera no lineal, se despeja las aceleraciones angulares del ángulo del péndulo y del volante de inercia obteniendo las ecuaciones principales que modelan el sistema del péndulo invertido controlado por volante de inercia:

$$\ddot{\theta}_b = \frac{(m_b l_b + m_w l) g \sin \theta_b - T_m - C_b \dot{\theta}_b + C_w \dot{\theta}_w}{I_b + m_w l^2} \qquad (11)$$

$$\ddot{\theta}_w = \frac{(I_b + I_w + m_w l^2)(T_m - C_w \dot{\theta}_w)}{I_w(I_b + m_w l^2)} - \frac{(m_b l_b + m_w l) g \sin\theta_b - C_b \dot{\theta}_b}{I_b + m_w l^2} \qquad (12)$$

La ecuación de la derivada de la primera variable de estado no se ha escrito explícitamente al ser trivial, ya que sería la segunda variable de estado.

En este sistema las variables de estado son:

$$X = (\theta_b \ \dot{\theta}_b \ \dot{\theta}_w)$$

Siendo:

- θ_b = Ángulo de la diagonal del péndulo.
- $\dot{\theta}_b$ = 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.

3.3. Punto de operación

Una vez tenemos las ecuaciones necesarias para definir nuestro modelo y en el formato requerido para el control en espacio de estado que disponemos a implantar, se ha de encontrar el punto de control para el cual el sistema es estable y en el que se desea mantener al sistema.

Para calcular este punto de operación se ha de tener primero en cuenta cuales son las referencias de posición, en este caso el ángulo. Estas referencias vienen dadas por las propias ecuaciones de la dinámica del modelo, y por ello se toma que el ángulo de inclinación 0 es cuando la diagonal coincide con la normal, es decir, perpendicular a la base.

Cuando ya se tienen claras las referencias para no malinterpretar los resultados, se anulan las derivadas de las variables de estado y se evalúan en las ecuaciones. La derivada del ángulo de inclinación es otra variable de estado así que ya tenemos el punto de operación para esta variable. Con las otras dos ecuaciones de las derivadas y el resultado previo calculamos los otros dos valores. Finalmente, el punto de operación queda con todas las variables nulas, *X0= (0,0,0).*

3.4. Linealización

Una vez tenemos la planta no lineal, hay que linealizarla en unas matrices constantes para que las derivadas de las variables de estado y las salidas sólo dependan de la entrada y de las

propias variables de estado para el punto de operación. Una vez esto sea así procederemos a diseñar el control en sí. Recordando lo escrito anteriormente, las variables de estado son X=*(θ_b $\dot{\theta}_b$ $\dot{\theta}_w$)* y el punto de operación *X0=(0,0,0).*

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

$$Y = CX + DU \qquad (13)$$

Para el cálculo de las matrices de forma teórica habría que realizar operar las ecuaciones hasta sacar las relaciones entre las diferentes variables antes de evaluarlas en el punto de operación. A pesar de que se calculado mediante funciones aportadas por nuestro entorno de desarrollo, se muestran cuáles serían sus expresiones teóricas sin evaluar.

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

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

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

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

3.5. Implantación en Simulink del modelo

Todas las dinámicas del modelo se implantan en Simulink mediante una función definida por el usuario de Matlab. El bloque define como un código de Matlab que realice las funciones que se escriban. En el código se configura como entradas las variables de estado y la variable de mando o entrada del modelo, las salidas serán las derivadas de las variables de estado y éstas mismas. Las derivadas pasarán por un integrador para realimentarse como entradas siendo variables de estado. Con las variables de estado de salida se calcula el mando mediante el futuro control.

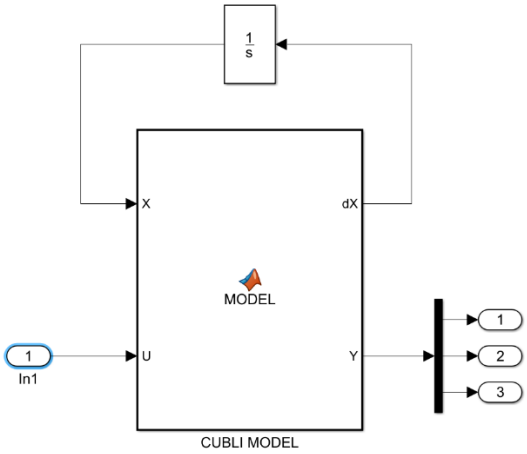


Ilustración 22: Diagrama de Simulink del modelo

3.6. Identificación de parámetros

La identificación de parámetros se ha dividido en cinco partes para su mejor distribución visual y la mayor rapidez en su búsqueda, pero se podría haber separado en dos únicas secciones: una de parámetros medidos o extraídos de las especificaciones y otra de parámetros estimados mediante ensayos, que serán las dos últimas partes.

3.6.1. Parámetros del motor

Están tomados de la hoja de especificaciones del fabricante, y los parámetros que se necesitaron tanto para el control como para la configuración de la controladora fueron:

- Máxima velocidad permitida: 10.000rpm
- 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

3.6.2. Masas

Se necesitan dos masas por separado, una la del cuerpo y otra la del volante de inercia más el motor:

- Masa Cuerpo (m_b) = 207,3 g
- Masa volante (m_w) = 207,75 g

3.6.3. Distancias

Las distancias a las que nos referimos son desde el punto de giro, en este caso la intersección de la cara con el eje con rodamientos. Como en las pruebas llevadas a cabo el sistema estaba equilibrado respecto al eje de simetría de la cara, las distancias sólo hacen referencia al módulo en este eje. Las dos distancias que se necesitan son:

- Distancia hasta el eje de rotación del volante de inercia: 91,925mm
  Esta medida se adquirió directamente del archivo en SolidEdge realizado.

- Distancia hasta el centro de gravedad del sistema: 85 mm
  Dado que estas distancias son necesarias para la simulación, los datos del centro de gravedad del sistema se calcularon teniendo en cuenta los componentes que desde un comienzo se sabía que eran necesarios, desplazando ligeramente el centro de gravedad hacia la base.

3.6.4. Momento de inercia rozamiento del volante

El momento de inercia que se ha tomado para el volante inercia se calculó por la geometría de la pieza al ser una única y monomórfica. A este momento hay sumarle el del rotor del motor que viene en la hoja de especificaciones del motor. Ambos se pueden sumar al estar referenciados al mismo punto, al ser concéntricos.

- 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

3.6.5. Momento de inercia de la estructura

A diferencia del momento de inercia anterior, la estructura está compuesta por muchos componentes diversos con formas no estándar. Este hecho supone una complejidad para su cálculo, pero al haber diseñado la estructura con un software en 3D se puede determinar estos datos a partir del ensamblaje de SolidEdge. Gracias a este programa sacamos el valor del momento de inercia.

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

Capítulo 4: Diseño del control

4.1. Estimación de las variables de estado

La correcta estimación de las variables de estado es vital para conocer la evolución futura del sistema, ya que para cualquier instante sólo son necesarias éstas y la entrada aplicada para que quede determinada. Es por eso, que la medición precisa del ángulo y velocidad del péndulo y la velocidad del volante de inercia resulta esencial para que el sistema realice su función correctamente.

4.1.1. Estimación del ángulo de inclinación

La mayor complejidad, a la hora de adecuar las medidas en crudo para su utilización posterior, ha sido la utilización de los acelerómetros de las IMU para el cálculo del ángulo entre la normal y la diagonal del ángulo. Esto se debe a que el ángulo es calculado a partir de las proyecciones de la gravedad sobre los ejes X e Y respecto al sistema de coordenadas local de la IMU $\{A_i\}$. Se sabe el ángulo porque al ser la gravedad una aceleración constante, tanto en módulo como en dirección, la tangente de ambas proyecciones daría el ángulo con la diagonal.

Siguiendo este método surge un problema derivado de que el propio fin del volante de inercia es generar aceleraciones angulares, lo que perturba toda medida de las proyecciones de la gravedad, siendo imposible discernir el ángulo a menos que el sistema esté en reposo. Para solucionar este inconveniente se optó por revisar literatura específica de proyectos similares, encontrando un método de matrices muy utilizado y que utiliza varias medidas de acelerómetros. En concreto se utilizaba una IMU en cada cara lo que aportaba muchas medidas para utilizar [8]. A pesar de que el método matricial se decantaba por ser el adecuado, éste no fue satisfactorio al sólo ser posible colocar los acelerómetros en una cara. Esto es debido a que la excepción para el cumplimiento del teorema en el que se basaba es que las medidas no pueden ser coplanarias.

Por todo lo anterior, se ha concluido que la mejor opción para medir el ángulo del péndulo en el sistema es la que lleva a cabo la universidad ETH de Zurich en su artículo “The Cubli: A Cube that can Jump up and Balance”. A continuación, se presentan las ecuaciones para llegar a la medida deseada, que se basa en la compensación de esa aceleración generada mediante la medida IMUs colineales en la diagonal. Para la correcta interpretación de las ecuaciones posteriores, los subíndices de las aceleraciones lineales ($_{ai}$m) reflejan el acelerómetro al que se está haciendo referencia.

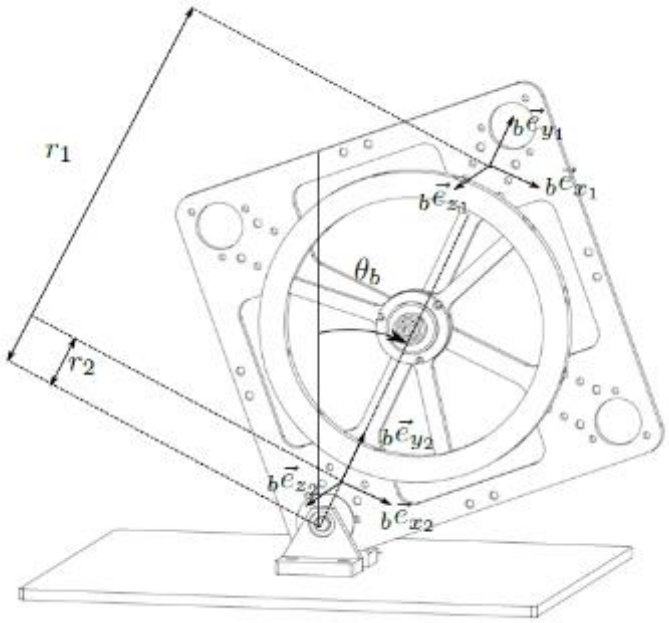


Ilustración 23: Estimación del ángulo [3]

$$ {}_{ai}m \coloneqq (r_i\ddot{\theta}_b + g\sin\theta_b, -r_i\dot{\theta}_b - g\cos\theta_b, 0), \qquad i = 1{,}2 \qquad (14)$$

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

$$ {}_{a1}m - \mu\ {}_{a2}m = ((1-\mu)g\sin\theta_b, -(1-\mu)g\cos\theta_b, 0) =: (m_x, m_y, 0) \qquad (15)$$

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

$$\hat{\theta}_b = atan(-\frac{m_x}{m_y}) \qquad (16)$$

4.1.2. Estimación de la velocidad de rotación de la cara del péndulo

La estimación de la velocidad de rotación es más directa, ya que es la proporcionada por los giróscopos de las IMU en el eje Z del sistema de coordenadas local $\{A_i\}$. La medida obtenida sólo se pasa a rad/s para que coincida con las magnitudes del SI, además de hacer una media entre ambas medidas para decrementar el posible error.

$$\widehat{\omega}_b = \frac{\omega_{z1} + \omega_{z2}}{2} \qquad (17)$$

4.1.3. Estimación de la velocidad de rotación del volante de inercia

Para la obtención de la velocidad del volante, que es igual a la del motor, se utilizan los sensores hall del motor, configurados y leídos por la controladora ECSON EC 36/3. Ésta nos aporta una señal analógica promediada de la velocidad de rotación real con un rango de tensión entre 0 y 4V. El promediado se realiza para una adquisición con las altas frecuencias atenuadas gracias a un filtro paso-bajo de 5Hz de frecuencia de corte. Hay que tener en cuenta que como los máximos rangos de velocidad son ±10.000rpm, por lo que en la recepción de la señal habrá que mapearla entre estos límites y convertirla a rad/s.

4.2. Control de balanceo

El control para balancear un eje sobre la posición de ángulo del cuerpo nulo (Θ_B=0), que es el punto de operación para dicho control, es el control más complejo y crítico del proyecto. Se necesita un sistema robusto y preciso, ya que cuando se implanten el resto de los ejes, los sistemas que controlan cada uno de ellos serán extremadamente similares al ya realizado. Esto se debe a que, al ser ejes ortonormales, las perturbaciones entre ejes se deberán principalmente al incorrecto montaje más que a interacciones teóricas.

El balanceo se domina con un diseño e implantación mediante un control por realimentación de estados. Lo primero que se necesita para el diseño es un modelo del sistema a controlar. El modelado no ha de ser lineal necesariamente, de hecho, en este caso no lo es y se debe a que se linealiza para el punto de operación alrededor del que se maniobra. Las ecuaciones del modelo son las ya explicadas previamente en el capítulo de "*Modelado y Dinámica*".

El control trata de asignar unas constantes a la realimentación de las variables de estado, para que se haga estable y reaccione como se ha deseado sobre el punto de operación. Este diseño se basa en la colocación de los polos del control en la posición óptima, tomando como referencia los polos de la planta, para que el sistema reaccione acorde a lo planteado. Para ello, se usan las funciones que aporta Matlab a la hora de sacar dichas constantes mediante el comando *place*, siempre y cuando antes se hayan calculado las matrices A y B, así como haber fijado los polos

del sistema en las posiciones convenientes. Se presenta la fórmula matricial teórica que permite el cálculo de dichas constantes, que se suele denominar fórmula de Ackerman, que se trata de igualar los autovalores del lazo cerrado al producto de los polos que ya se han fijado.

$$|\lambda I - (A - KB)| = p1 \cdot p2 \cdot p3 \qquad (18)$$

A la hora de fijar los polos, se ha utilizado una configuración Butterworth de tercer orden, que se trata de colocar dos polos complejos y otro real, siendo todos de mismo módulo. Hay que tener en cuenta que a todos los cálculos anteriores hay que introducir el hecho de que se ha de calcular de forma discreta, ya que el tiempo de muestreo tendrá efectos sobre el control al tener una implantación digital.

Se eligieron unos polos que cumplieran las especificaciones necesarias para el funcionamiento del control, que serían alta rapidez pero sin mucho sobrepaso. La velocidad del control se necesita porque se debe actuar ágilmente para que el péndulo no recorra mucho ángulo antes de corregirse, tanto porque el fin del proyecto es que esté prácticamente fijo en la posición, como porque un incremento de ángulo podría ocasionar que no se pueda recuperar. Este último fin es el mismo por el que no se puede dar un gran sobrepaso, ya que si supera el límite que puede corregir el motor, la cara caerá.

4.3. Control de salto

Formalmente no se debería denominar control, ya que no se diseña ninguno de forma teórica y es totalmente empírico al probar las velocidades a las que frenar el disco hasta que éste se levante hasta el ángulo precisado. Una vez se determina la velocidad para que dicha elevación sea óptima, se automatiza el frenado para la velocidad definida y así siempre conseguir el resultado óptimo. Para conocer esta velocidad se utiliza una señal digital de la controladora que nos avisa de haber alcanzado el valor predefinido, en vez de utilizar la medida que se obtiene de la señal analógica al considerarse más rápida y precisa al no necesitar de etapas intermedias de procesamiento.

Capítulo 5: Implantación del control

5.1. General

La implementación de todos los sistemas se hará mediante la herramienta Simulink, generando un programa que englobe a todos ellos. Como el hardware y la simulación no podrían ejecutarse a la vez, dado que sobrescribirían los datos, uno de éstos está siempre comentado para que sea el otro el que actúe. Se presenta el esquema general del programa.

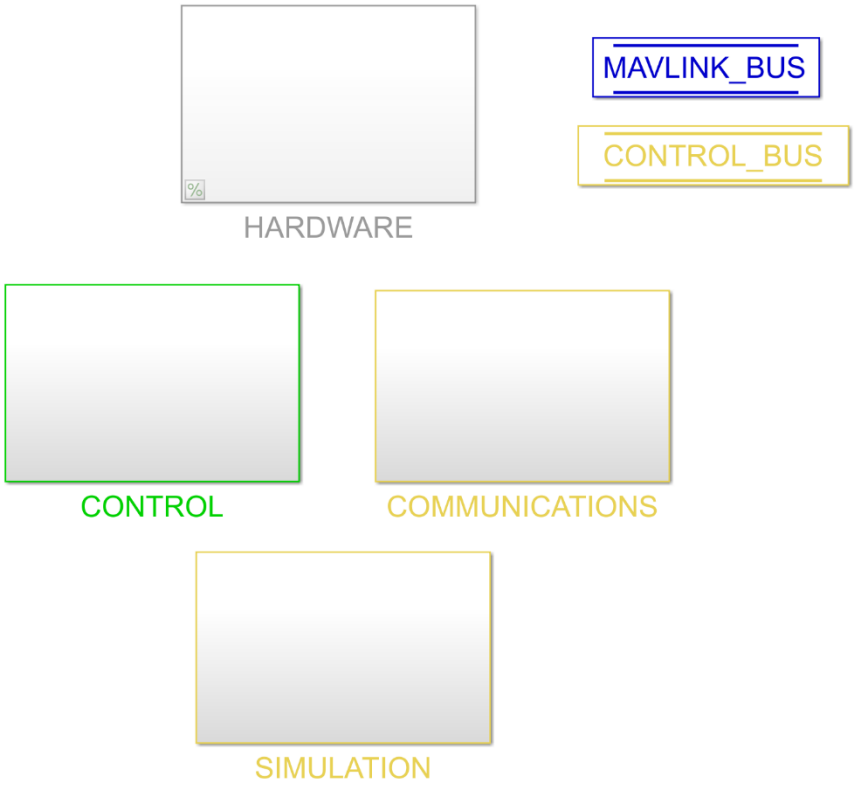


Ilustración 24: Diagrama general de Simulink del programa

En este se pueden determinar las cuatro partes en las que se divide: hardware, control, comunicaciones y simulación.

5.2. Simulaciones

Para probar el control y comprobar su correcto funcionamiento antes de implementarlo junto al hardware, se realizan una serie de simulaciones. El hecho de que se pruebe previamente

es debido a que, en el caso de no operar adecuadamente, los actuadores podrían sufrir sobre-corrientes o ser instados a acciones imposibles.

Para el diseño del control, se usó una simulación escindida del programa principal para tratar de reducir toda posible interacción con el resto del sistema y, además, simplificar el diagrama. Cuando se hubo observado que el control se comportaba correctamente con los parámetros de diseño escogidos, se optó por traspasar estos diseños a una simulación ya incluida en el mismo programa que el hardware, control y comunicaciones.

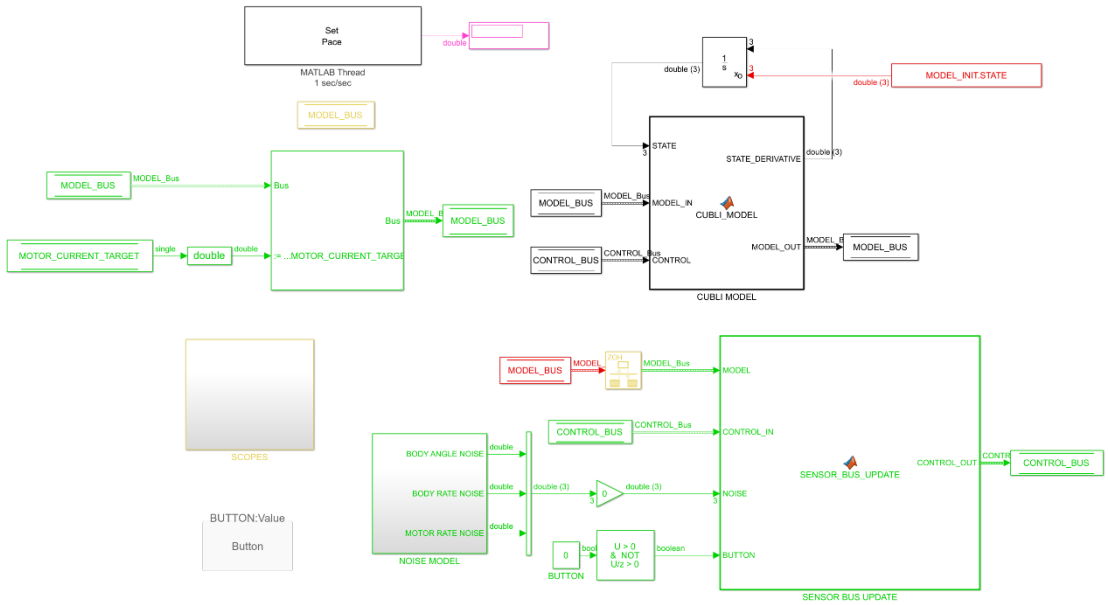


Ilustración 25: Diagrama de Simulink de la simulación

El diagrama mostrado representa la implementación de la simulación en el programa final, para llevar a cabo las pruebas simuladas mas realistas posibles y con los mismos campos que se usarán en el programa real. Para ayudar a probar el propio código, en vez de realimentar las propias señales de la planta como entradas al sistema, como se haría típicamente, éstas se mueven a los buses que usará el sistema real para tomar las medidas, pudiendo introducir incluso ruido en éstas.

5.3. Microcontrolador

La implantación de la Raspberry Pi Zero en Simulink se hace mediante un conjunto de bloques aportado por Mathworks, una vez instalado el software pertinente. Una vez descargado y listo para usar, se utilizan los bloques aportados para la función necesaria, siempre configurándolos para el modelo de Raspberry utilizado. Existen bloques para los protocolos de comunicación,

para la activación de salidas o la lectura de pines, sólo hay que elegir el bloque y usarlo según las indicaciones. En las ilustraciones de próximos apartados se pueden dilucidar estos bloques, ya que tienen el rótulo "RASPBERRYPI" en azul.

En cuanto a la carga del programa en el dispositivo, ésta se hace mediante una conexión inalámbrica TCP/IP, por lo que es necesario que se aporte el programa la dirección I.P. de la Raspberry, así como el usuario y contraseña. Una vez introducidos estos datos en el programa, y cohabitando el ordenador y el microcontrolador en la misma red, Simulink se encarga de cargar lo programado en el dispositivo.

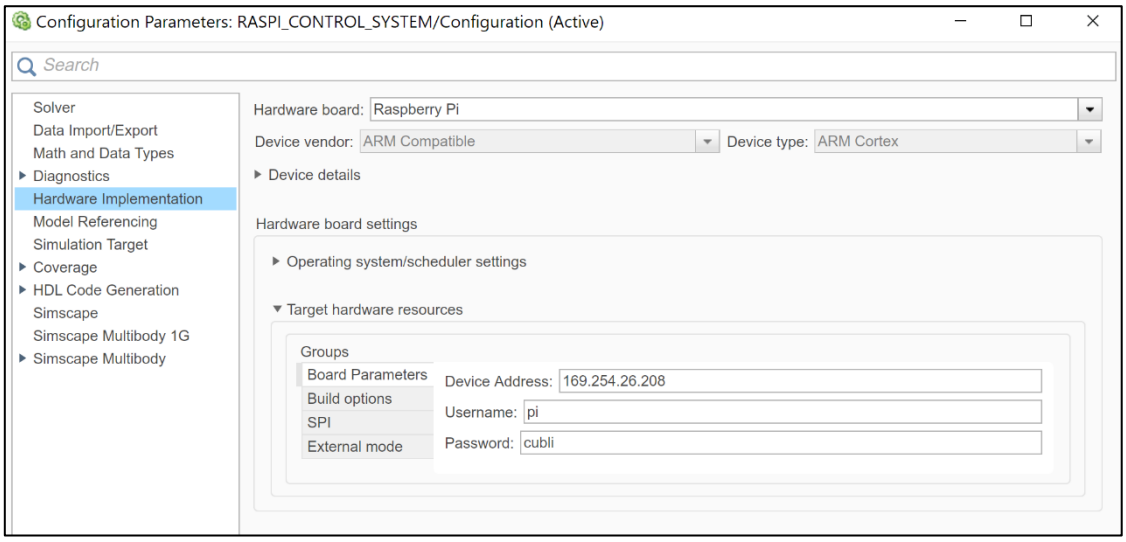


Ilustración 26: Configuración de Simulink para Raspberry

5.4. Actuación

5.4.1. Servomotor

Los servomotores son unos dispositivos que tienen un rango de movimiento generalmente de 180º, dado que son motores para desplazamientos finitos y no para un giro continuado. El utilizado en este proyecto es un MG90S, que es de engranajes metálicos para resistir una mayor fuerza en el eje, utiliza un PWM de 50 Hz de frecuencia (20ms) y un ciclo de trabajo entre 1 y 2 ms, siendo 1 ms la referencia para 90 grados y 2 ms para -90º.

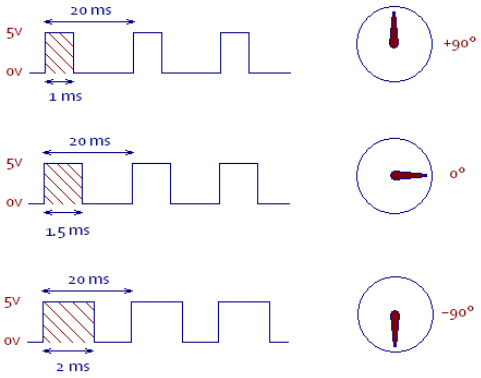


Ilustración 27: Señal servomotor

Para su funcionamiento sólo se necesita un bloque de salida de PWM para una Raspberry Pi, exactamente uno específico para servomotores, en el cual la entrada es en grados entre 0 y 180. Por ello, se dispone de una variable que se modifica desde el control para actuar sobre esta salida.

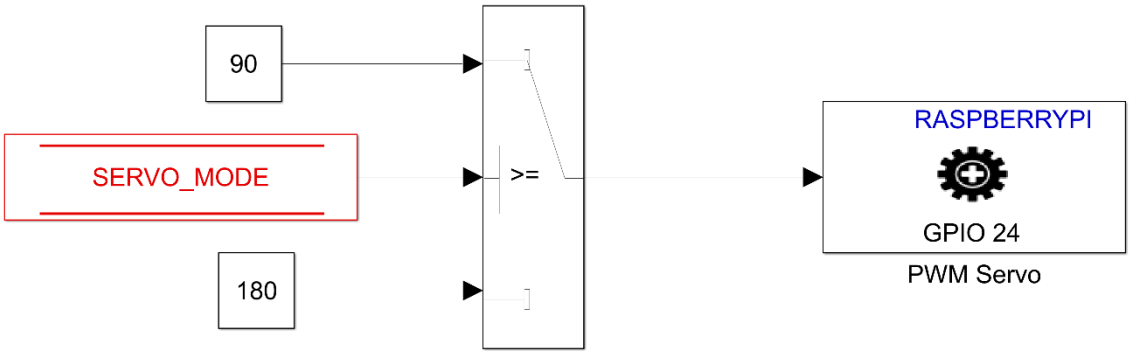


Ilustración 28: Diagrama de Simulink del servomotor

5.4.2. Motor

Se ha utilizado un Motor EC MAXON 45 Flat de 50W mediante una controladora ESCON 36/3. Se utiliza una controladora ya que permite una fácil configuración y monitorización del motor mediante una herramienta de software denominada ESCON Studio. Con dicho software se pueden configurar el tipo de consigna las salidas deseadas, los modos de funcionamiento y otras características respectivas al control y configuración.

El motor tiene las señales para su alimentación, los devanados y los sensores hall, pero todo ello va a la controladora, siendo las entradas y salidas de ésta configurables. La distribución definitiva es:

- Entrada digital 1: consigna de corriente mediante PWM, rango de 0 a 100% de ciclo de trabajo suponiendo esto una corriente de ± 9A.
- Entrada digital 2: Habilitación del motor, activo alto.
- Salida digital 1: speed limit 3000rpm, se refiere a una salida booleana que toma valor alto cuando el motor supera las 3000rpm en cualquiera de los dos sentidos.
- Salida analógica 1: corriente real promediada, es un promedio de la corriente que está consumiendo el motor.
- Salida analógica 2: velocidad real promediada, es un promedio de la velocidad a la que está girando el motor en ese momento. Ambas salidas analógicas se mueven en un rango de 0 a 4V para velocidades de ± 10.000rpm.

El funcionamiento del motor es sencillo una vez dominado el software que lo configura, a pesar de ello hay que lidiar con algunas dificultades con las señales que entran o salen de la controladora.

La primera es la alimentación, el motor necesita 24V de tensión para su correcto ejercicio, que en un sistema de electrónica de 5V o menos, resulta complicado de abastecer. Para solucionarlo se instaló una batería de alta descarga de 3 celdas que aportaba 12V aproximadamente y toda la corriente necesaria. Con esta batería y un regulador de 5 V teníamos la alimentación de todos los componentes exceptuando el motor, para el que se necesitará un conversor DC-DC de 12 a 24V que duplicase la tensión a costa de la corriente. A pesar de que el conversor introducía ruido en la señal, la controladora admitía como válida la alimentación.

El otro inconveniente principal fue con la salida analógica de velocidad. Distinguimos ésta de la otra salida analógica debido a que la de corriente no se utiliza más que para consulta directa de un instrumento de medida. Por otro lado, la velocidad se debe realimentar al sistema constantemente para que el control conozca dicha velocidad en todo momento. Como previamente ya hemos repetido, la salida es analógica pero la Raspberry Pi no admite entradas analógicas al no disponer de un conversor AC-DC propio. Por esta causa, se adquirió un Arduino Nano externo que si dispone de conversor y que mandaría la medida por protocolo Serial. La comunicación serie se elige debido a que se necesita exclusivamente un cable, al ser comunicación unidireccional, y posee una velocidad aceptable como sensor configurado a un baudrate de 115.200, un bit de stop y sin paridad.

El resto de posibles dificultades fueron por la especificidad de los conectores para la controladora, que al ser de diseño industrial requerían conectores propios para todas las señales.

A continuación, se muestran los bloques de control del motor, que están divididos en la parte de escritura, que serían las salidas hacia el motor, y lectura, entradas al sistema de control.

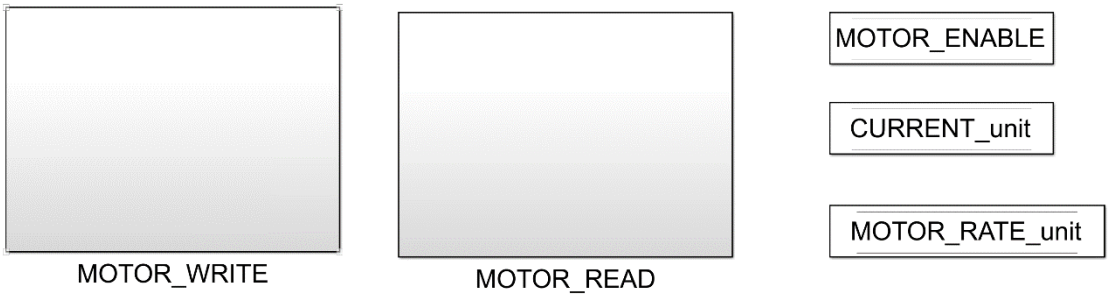


Ilustración 29: Diagrama general de simulink del motor

Se procede a enseñar los diagramas internos del motor, de escritura en un primer lugar y posteriormente el de lectura. En las señales de actuación para el motor se puede observar que exite la de habilitación y la de consigna, siendo posible en esta última si gobernar mediante velocidad o mediante corriente con el interruptor. El bloque superior se encarga de seleccionar las combinaciones de salidas para el motor en función del estado en el que se encuentre el sistema.

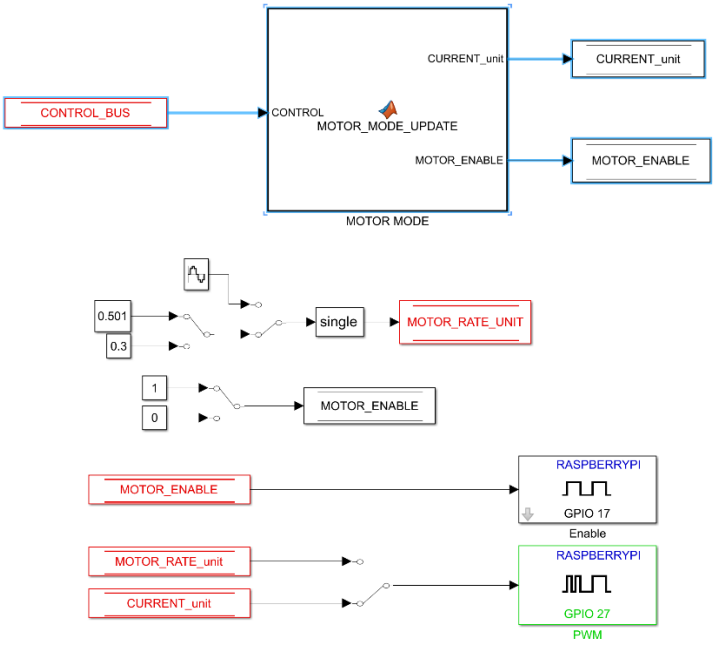


Ilustración 30: Diagrama de Simulink de las señales de actuación del motor

En las señales de entrada al control se utilizan dos opciones, mediante un pin digital de propósito general como sería el caso del detector de límite velocidad, que es una señal booleana, o utilizando el bus serie para recibir medidas analógicas. Cada medida viene en dos paquetes de 8 bits que ha de ser ensamblado para su correcta interpretación.

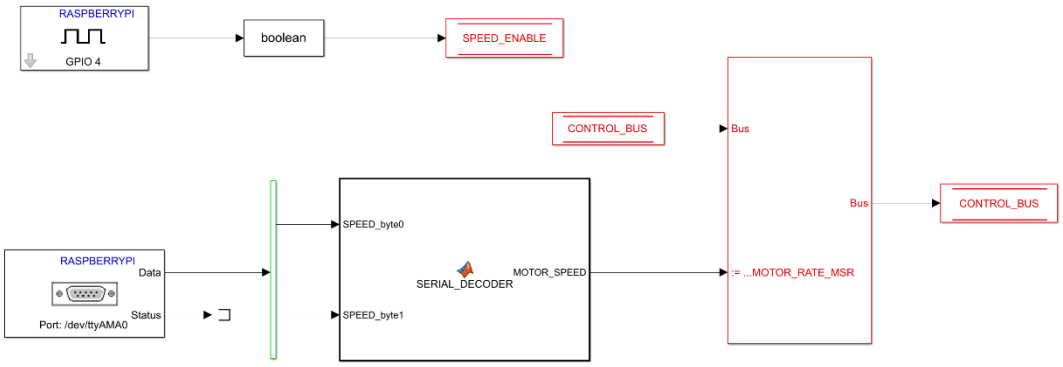


Ilustración 31: Diagrama de Simulink de las entradas enviadas por el motor

Además del control desde la Raspberry, es necesaria una configuración de la controladora del motor mediante el software propio ESCON Studio. En este hay que fijar las entradas, salidas y el rango de valores de consigna que se van a introducir para la correcta interpretación de ésta.

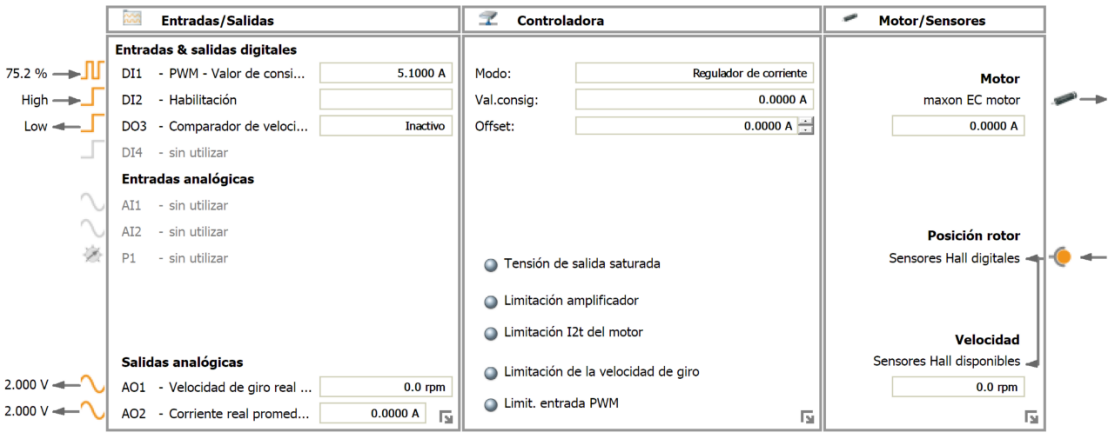


Ilustración 32: Configuración controladora del motor ESCON Studio

5.5. *Sensores (IMU)*

Previo a configurar todos los registros se debe asegurar que la dirección del dispositivo es la correcta y que ambos tienen una diferente, así como de que esté correctamente conectado. Para ello se ha de alimentar con los pines de VCC y GND, conectar los pines del protocolo I2C de SCL y SDA de cada uno a los de la Raspberry y seleccionar la dirección con el pin ADD. Como sólo tenemos un pin de dirección, sólo se puede configurar el último bit de ésta conectando ese pin a tensión o a tierra (0x69 o 0x68 en HEX), por lo cual únicamente se pueden conectar dos sensores como éste al mismo I2C.

Las medidas que aporta el sensor son lo que se llaman crudas, una vez tomadas se han de escalar y operar dentro del programa. Lo primero es escalarlas para reducir el ruido de medida y

que estén referidas al Sistema Internacional, m/s^2 para aceleraciones y rad/s para velocidad angular. Una vez esto sea correcto, se dispondrá de la función para calcular el ángulo de inclinación de la cara a partir de las aceleraciones como se indica en el apartado **¡Error! No se encuentra el origen de la referencia.**, que es para lo que se requieren dichas medidas.

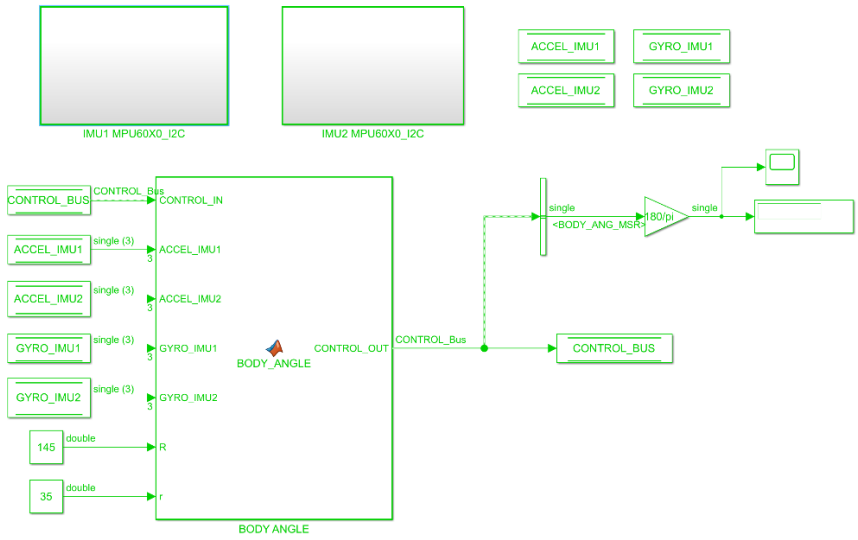


Ilustración 33: Diagrama de Simulink del cálculo del ángulo del cuerpo

En Simulink se ha separado la recepción de medidas y su escalado, que se hace independiente para cada IMU, del cálculo del ángulo, para el que se necesitan todas las medidas. En el primer diagrama, de un nivel superior, cohabitan los bloques de recepción de datos con el de cálculo del ángulo, siendo el segundo diagrama la ampliación de uno de los bloques de las IMU, casi idénticos exceptuando unos pocos registros.

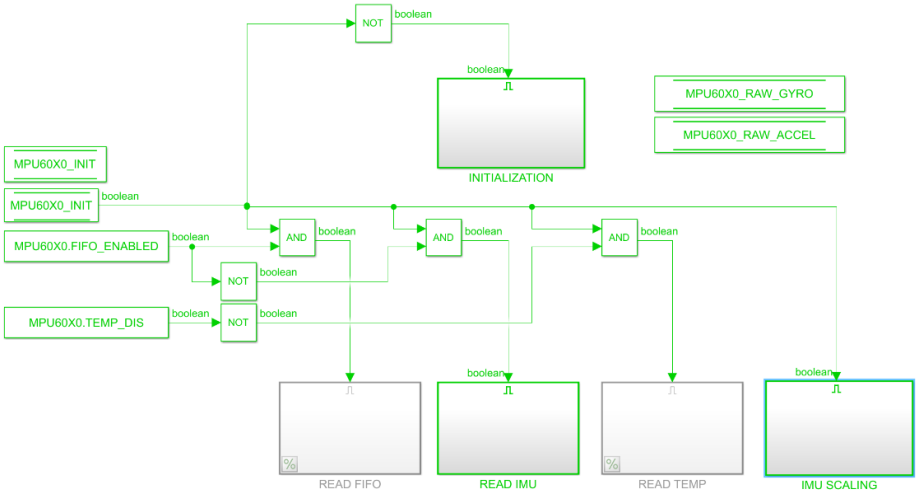


Ilustración 34: Diagrama Simulink de la lectura y escalado de una IMU

5.6. Algoritmo de control

Para actualizar el mando que se debe aplicar se ha programado una función de Matlab que realice el proceso. Típicamente se haría realimentando las variables de estado multiplicadas por las constantes, o asignando unas medidas a las variables. Pero como esos datos necesitan estar accesibles por varias secciones del programa, y además se hacen comprobaciones de validez, resulta más sencillo configurarlo como código en vez de bloques.

La función tiene un filtrado de que control utilizar según las opciones escogidas en la configuración principal, siendo en este caso el más completo. Con las medidas recibidas y todos los parámetros disponibles, calcula el mando que se debe enviar al actuador para corregir las posibles variaciones respecto del punto de operación, que en este caso será una corriente. En caso de requerir una mayor comprensión de este código, éste se encuentra en el Anexo.

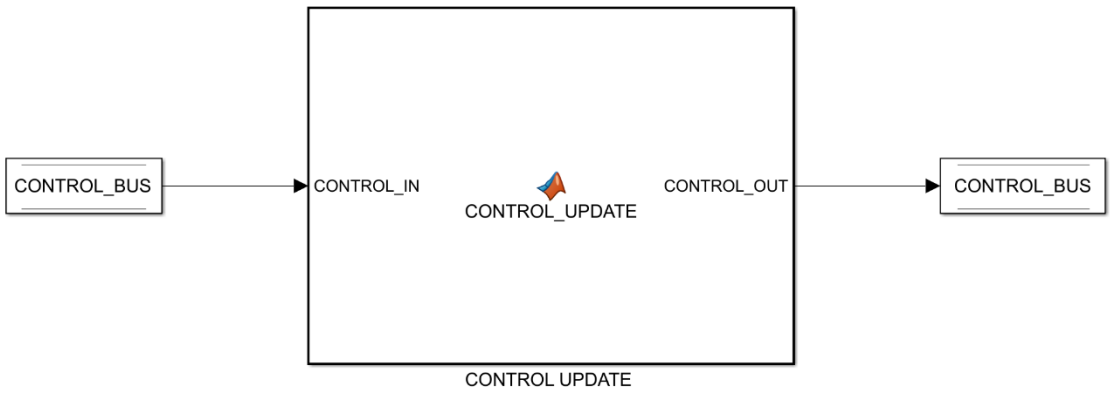


Ilustración 35: Diagrama de Simulink de la actualización del control

5.7. Máquina de estados

Para aunar los controles previamente mencionados se utiliza una máquina de Moore, es decir una máquina de estados cuas salidas sólo dependen del estado actual. La máquina de estados actual es sencilla debido al reducido número de entradas, salidas y estados, sobre todo de lo último ya que además de pocos tienen una sucesión lineal. Así se procede a enunciar los estados existentes:

- *STOP*: es el estado de comienzo y fin, todo está parado y no hay ningún contador corriendo. Para comenzar y pasar al siguiente estado se ha de pulsar el botón.
- *CALIBRATION*: en este estado, aunque no hay movimiento, se están calibrando los sensores para que en las etapas posteriores funcionen adecuadamente. Tiene un temporizador que tras su cumplimiento pasa a la entapa posterior.
- *ACCELERATION*: comienza el control de salto, que consta de dos estados, en este se acelera el disco de inercia hasta que se llega a los 3000rpm, en ese momento salta una señal que hace avanzar la sucesión.
- *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, se continúa el orden.
- *BALANCE*: aquí entra el control de balanceo que intentará mantener el el sistema estable entorno al ángulo 0. Este estado termina al pulsar el botón, que hace volver al estado de reposo.

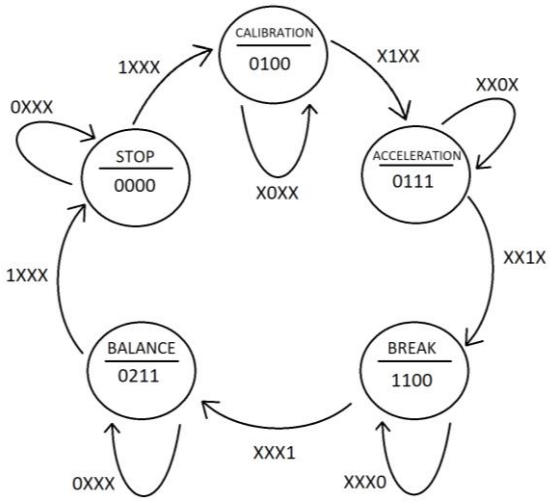


Ilustración 36: Maquina de estados

Aunque ya se pueden deducir las entradas existentes a partir de las transiciones entre estados, se procede a especificar dichas entradas.

- Botón: ésta es la única entrada manual que aparece en el sistema, utilizándose para el comienzo y parada de la secuencia, ya que depende del usuario que lo active.
- Temporizador: se toma como entrada a pesar de ser una variable incremental que corre durante todo el proceso y sirve para implementar transiciones mediante incrementos de tiempo, como en el caso de la calibración.
- Habilitación por velocidad: es una variable booleana que se activa cuando volante alcanza la velocidad predefinida y sirve para comenzar la etapa de frenado.

- Habilitación por ángulo: ésta se activa cuando el ángulo está en el rango de ±10º dese la normal para pasar al estado de balanceo.

Por último, quedan las salidas, que se tratan tanto de los actuadores como de variables que activan zonas del programa según el estado en el que se encuentre el sistema.

- *SERVO_MODE*: es el actuador para el freno y se activa sólo en la etapa de freno.
- *CONTROL_MODE*: cambia el modo del control, siendo posible estar parado, calibrando o en balanceo.
- *MOTOR_ENABLE*: permite el paso de corriente, activando así el motor.
- *MOTOR_MODE*: modifica el modo en el que se mueve el motor, acelerando hacia un lado, otro o libre para el balanceo.

En Simulink, esta máquina de estados quedó implantada como un bloque con una función de Matlab que realizada todos los cambios de estado y la actualización de entradas y salidas mediante código.

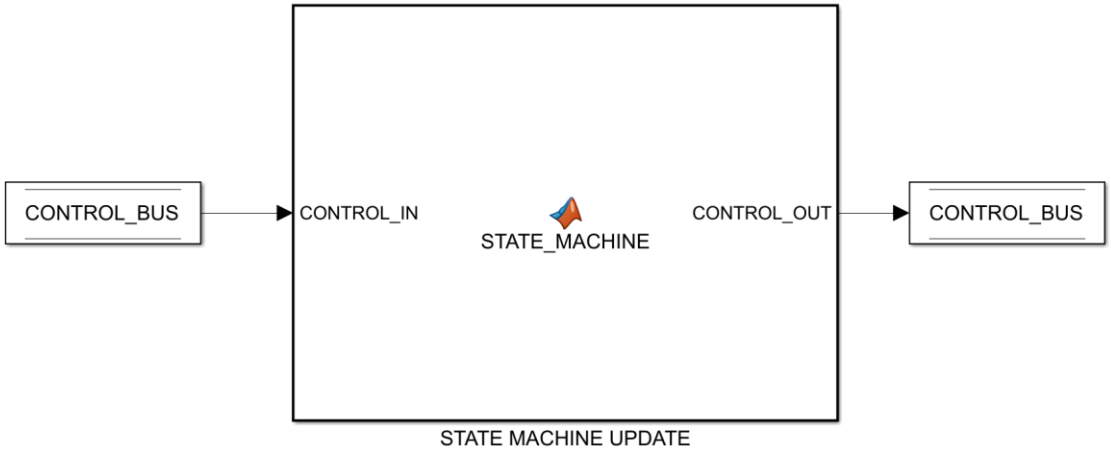


Ilustración 37: Diagrama de Simulink de la máquina de estados

5.8. Programa de monitorización

Para no necesitar cargar el programa con una comunicación continua, e invasiva, entre el programa de configuración y el dispositivo, se utiliza un programa escindido del general. Este programa de monitorización permite una carga normal en el dispositivo, y que sea éste mediante una comunicación programada previamente, el que envíe los datos requeridos por el protocolo fijado. El programa está diseñado para recibir y enviar mediante protocolos Serial, TCP/IP y UDP,

siendo este último el utilizado en este proyecto. Esta comunicación utiliza el protocolo de enlace MAVLINK que se ha explicado previamente en el apartado 2.8.4.

En el programa existen bloques dedicados a cada posible acción. Además del de comunicación, los principales serán *"SCOPES"* y *"HARDWARE"*. El primero es para la lectura de las señales recibidas, cambiando las unidades o visualizándolo como una onda dependiendo de la preferencia. El segundo es para enviar consignas que se cambiarán en el dispositivo al enviar el mensaje y estará dividido según los actuadores existentes.

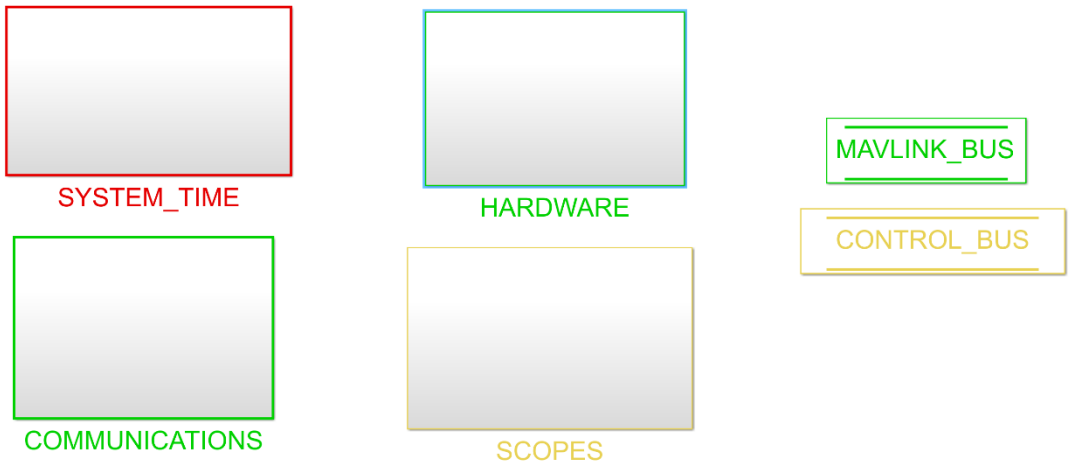


Ilustración 38: Diagrama de Simulink de monitorización

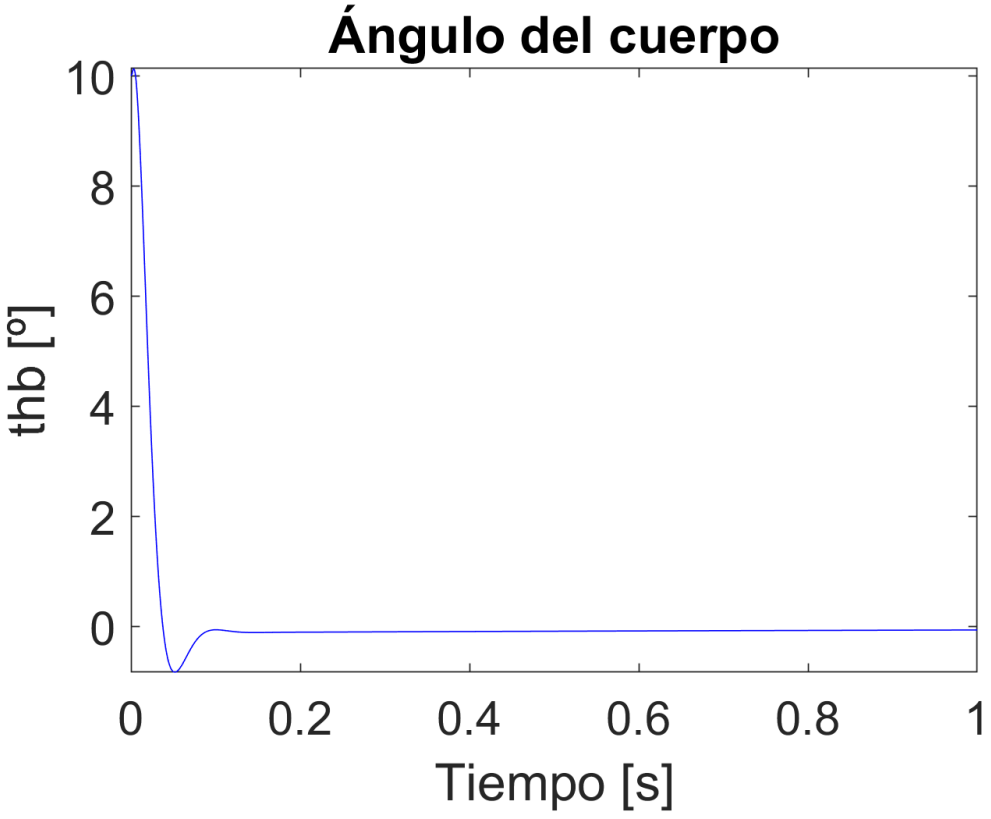
Capítulo 6: Resultados

6.1. Resultados de las simulaciones

Para comprender mejor el funcionamiento del sistema se han realizado varias simulaciones. En éstas se han tomado distintos parámetros para su mejor representación, y se mostrarán únicamente los parámetros más significativos.

Previamente a implantar el diseño del control en el programa final se realizó una iteración de diversos controles en el programa previo, teniendo como aceptable la morfología de uno que ostentaba una seta de 0,7 y una pulsación natural de 1,2 veces la del control proporcional.

Los resultados gráficos de estas pruebas tras luna simulación con el cuerpo soltado a 10º de la posición de equilibrio son los siguientes:



C

Ilustración 39: Ángulo del cuerpo sin ruido

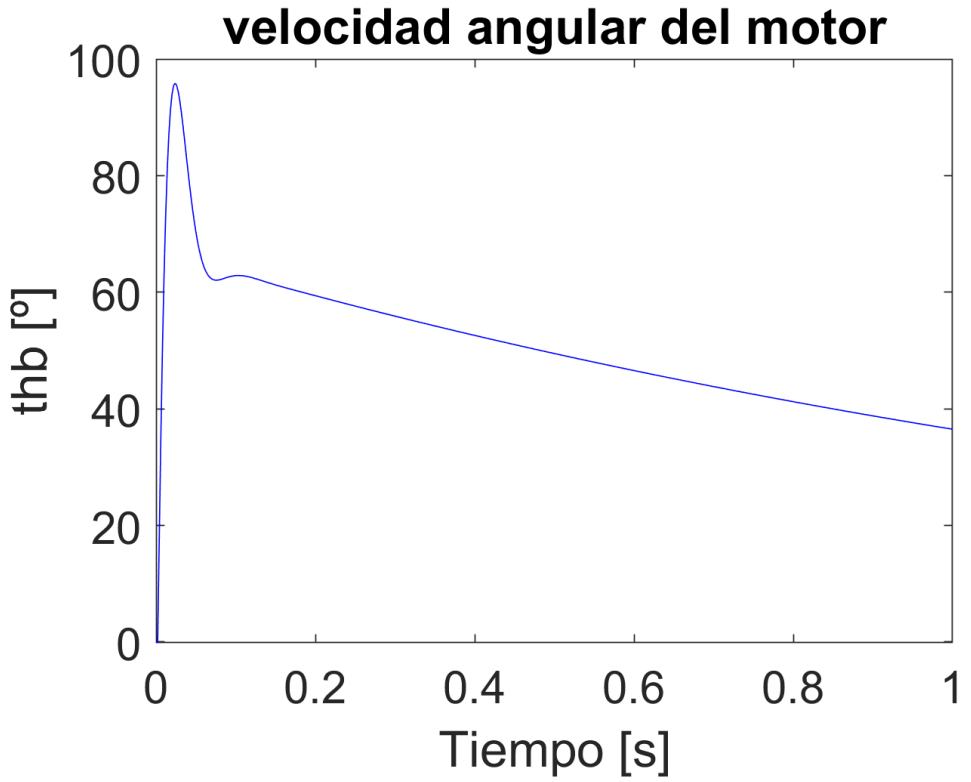


Ilustración 40: Velocidad angular del motor sin ruido

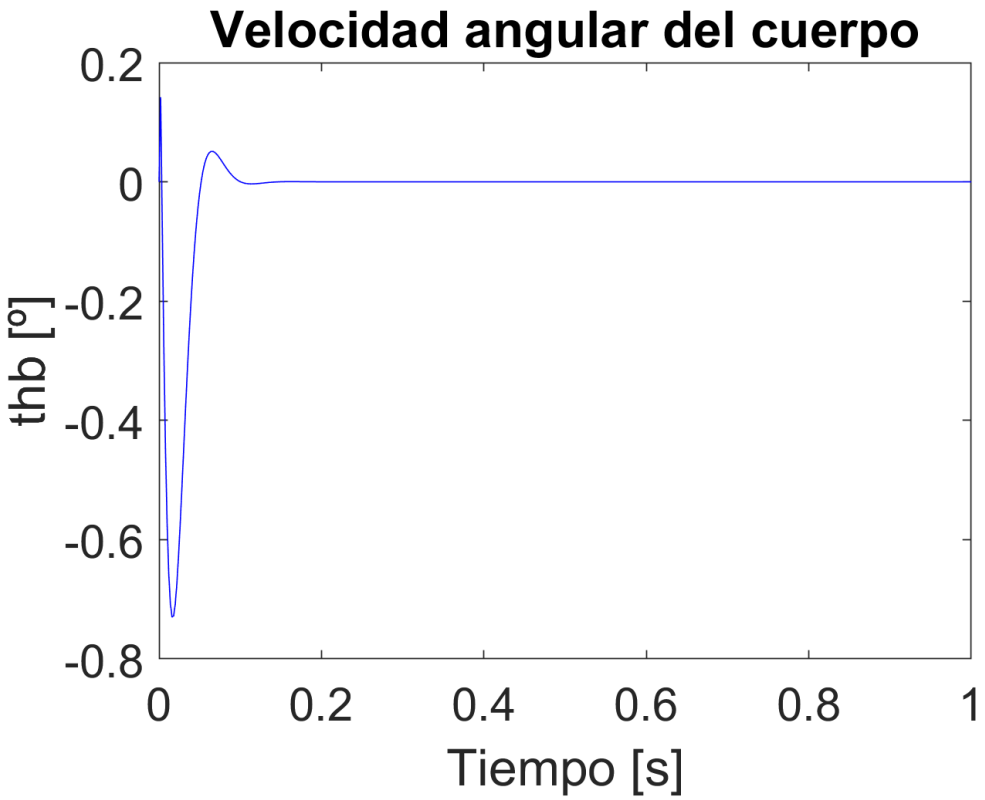


Ilustración 41: velocidad angular del motor sin ruido

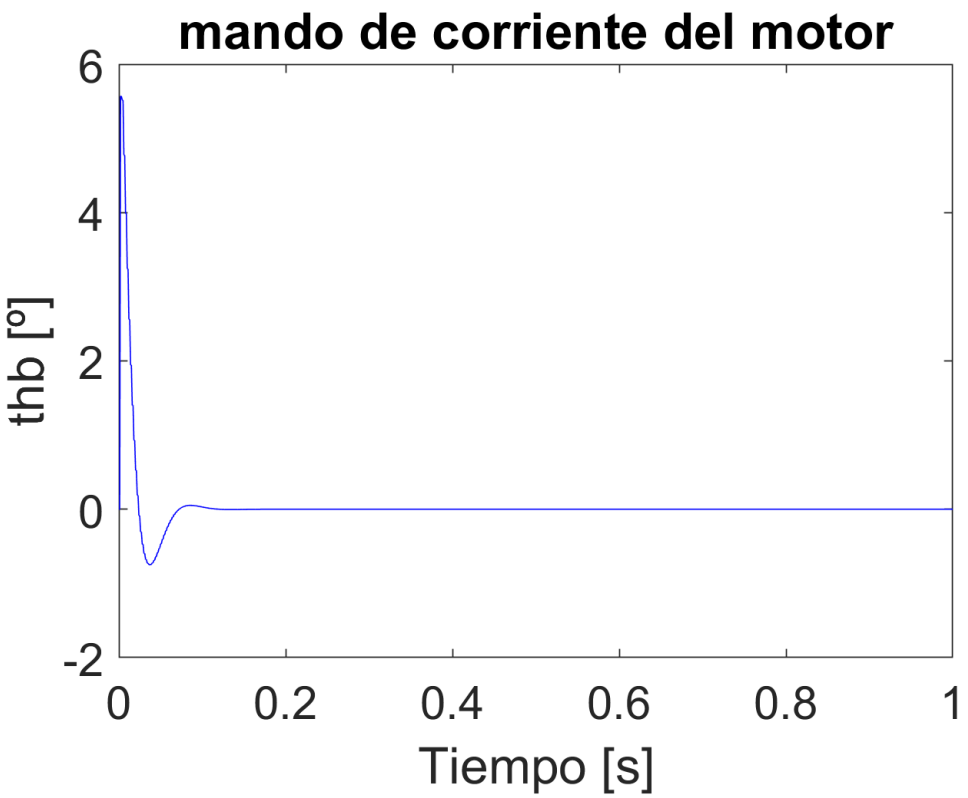


Ilustración 42: corriente del motor sin ruido

A pesar de qu algunos picos y sobrepasos, principalmente en las velocidades y corrientes, son amplios, la formal del ángulo es correcta. Esto se debe a que al necesitar una rapidez considerable debido al estado del sistema es obligado acotar un sobrepaso que acaba haciéndose estable dentro del los límites del motor tanto en corriente como en velocidad angular.

Ahorra pasamos a introducir un ruido considerable, hasta un 20%, en todas las medidas necesarias para el control, simulando un caso extremo. Las formas de onda son las siguientes:

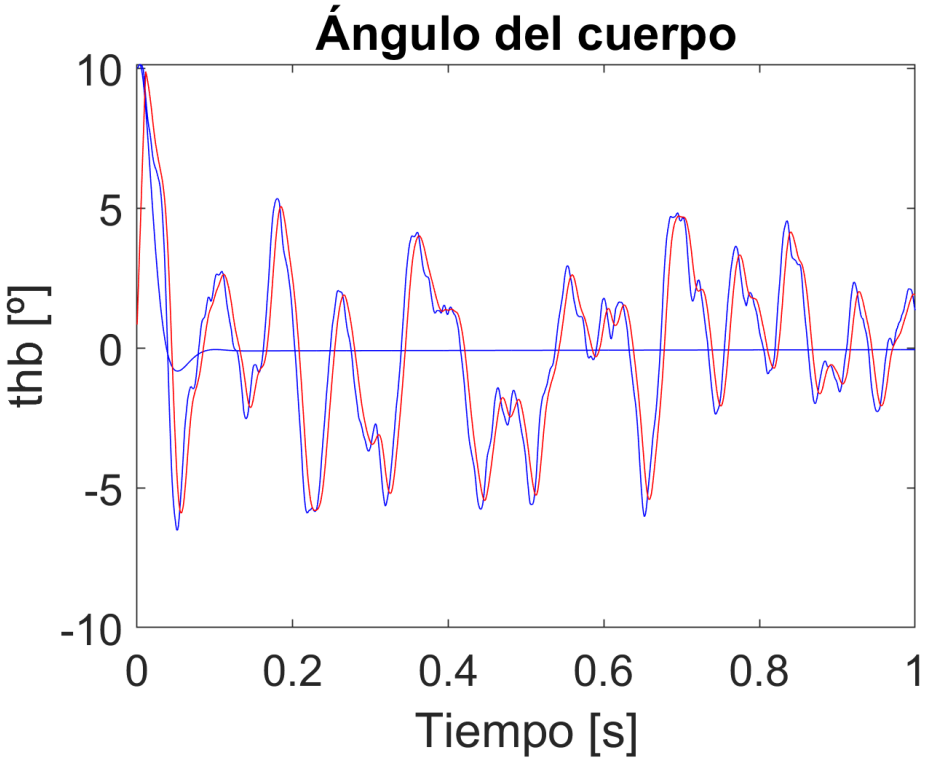


Ilustración 43: ángulo del motor con ruido de medida

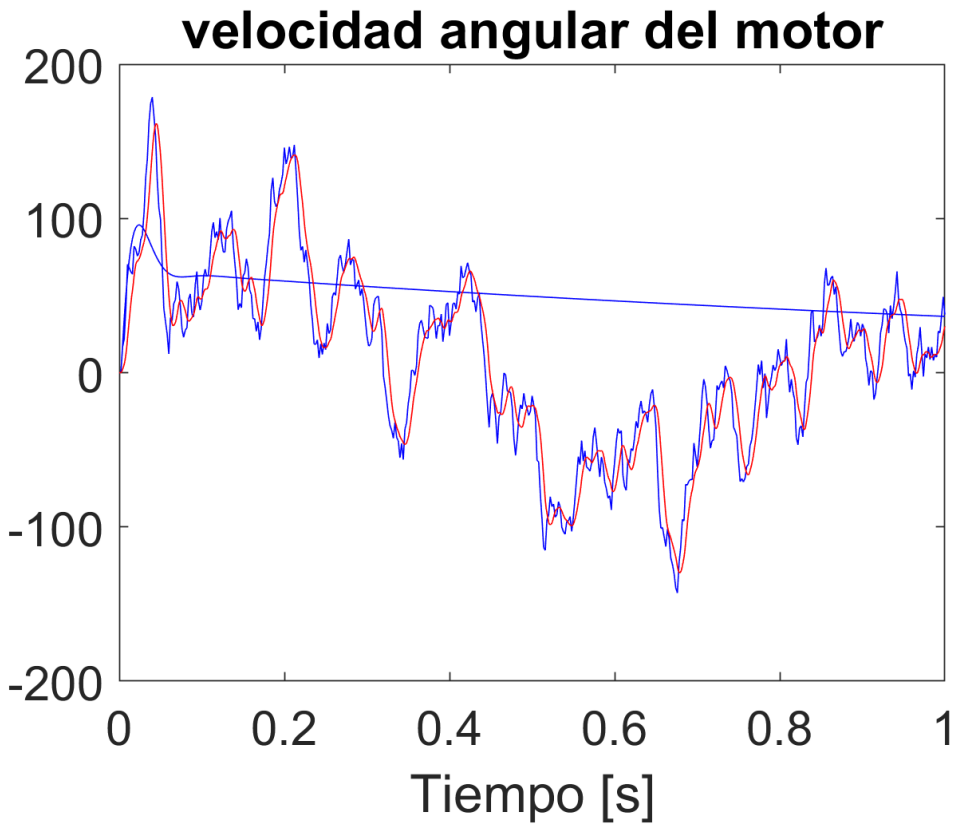


Ilustración 44: Velocidad del cuerpo con ruido de medida

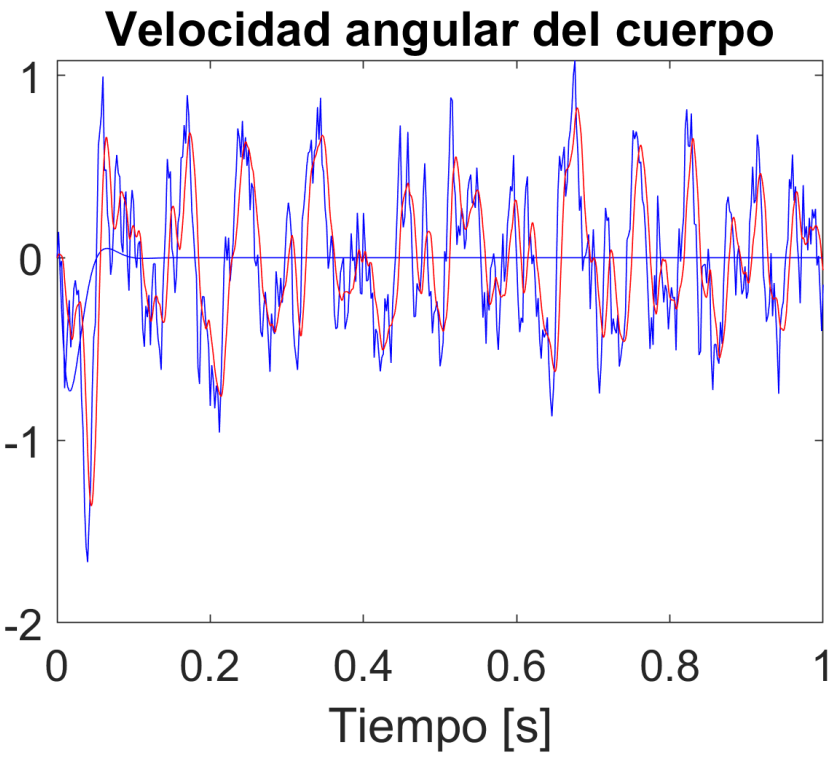


Ilustración 45: velocidad del motor con ruido de medida

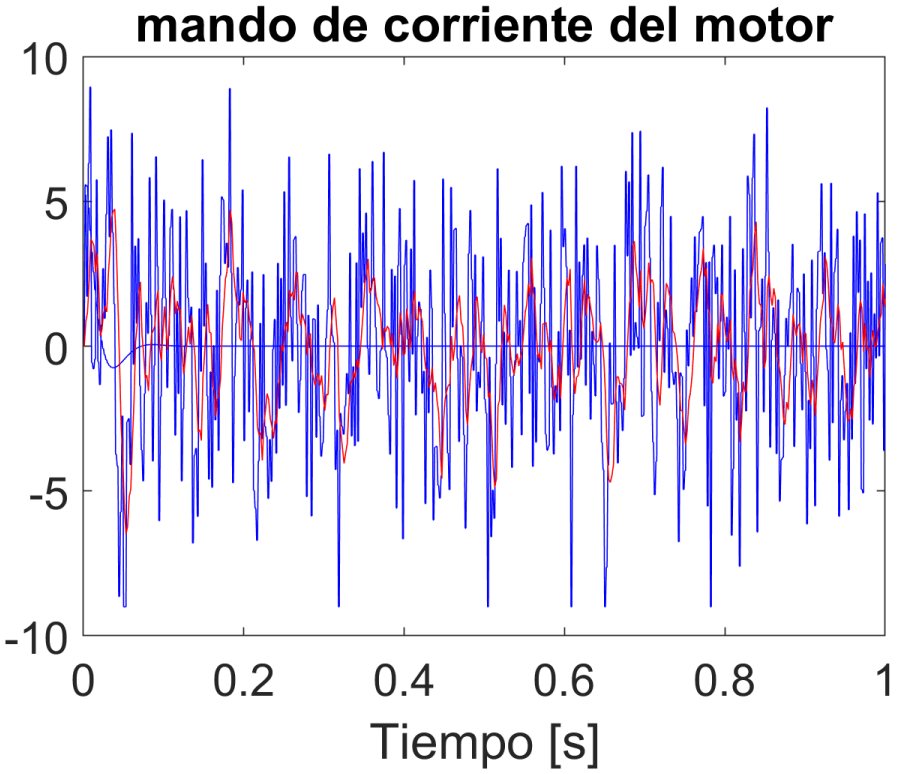


Ilustración 46: corriente del motor con ruido de medida

En las imágenes anteriores se pueden observar dos señales similares, la azul sería la medida tomada tal cual mientras que la roja es la misma señal pasada por un filtro de medias móviles para intentar reducir algo el ruido. Al introducir una gran cantidad de ruido de medida el control empeora llegando a ser oscilante, pero nunca se hace inestable, lo que sería aceptable si el ángulo de variación fuese pequeño.

6.2. Resultados de las pruebas

6.2.1. Servomotor

El sistema de frenado que se llegó a implementar resulta deficiente debido a que no es capaz de parar el disco sin sufrir deformaciones permanentes. Al intentar frenar el volante de inercia la patilla del servo se dobla errando en su fin. En el caso de reforzar dicha patilla con una pequeña pletina de metal, el tornillo que la sujeta al servomotor se parte. Estos episodios se generan cuando se aumenta la velocidad de giro del motor por encima de 1500 rpm, siendo necesario para que se genere una fuerza suficiente para levantar la estructura.

Ilustración 47: Fallos sistema de frenado

Debido a dichas causas, se diseñó un nuevo sistema de frenado con el fin de evitar roturas y mejorar su eficiencia. El nuevo sistema de frenado, así como el antiguo, puede observarse en el apartado 2.4. Para evitar impactos en el servomotor y que así este no se rompa se utiliza el servo para mover una pletina unida a la estructura q será la que reciba el golpe. Las simulaciones de esfuerzo avalan el diseño, aunque no se pudo implantar por falta de tiempo.

6.2.2. Sensores (IMU)

6.2.3.1. Lectura de los acelerómetros

Para comprobar la correcta adquisición de datos por parte de los acelerómetros se lleva a cabo las pruebas del ángulo de inclinación, ya que es el fin último de estos sensores. Como se puede identificar en las ilustraciones x y x, el ángulo es negativo cuando ella cara se localiza a la derecha y positivo si está en la izquierda. El signo y la magnitud son correctos, ahora es necesario comprobar la precisión de la medida.

Para hacer una prueba de precisión, se comparará la medida aportada por el procesado de los acelerómetros con la medida física del ángulo mediante un transportador. Las imágenes los valores del transportador hay que restarles 90º ya que la referencia de este proyecto son los 90º con la horizontal. Se considerará un margen de ± 0,5º debido al ruido de medida que existe en la comunicación, que puede alterar la precisión de la medida dentro de un intervalo aceptable.

Ilustración 48: Comprobación del ángulo central

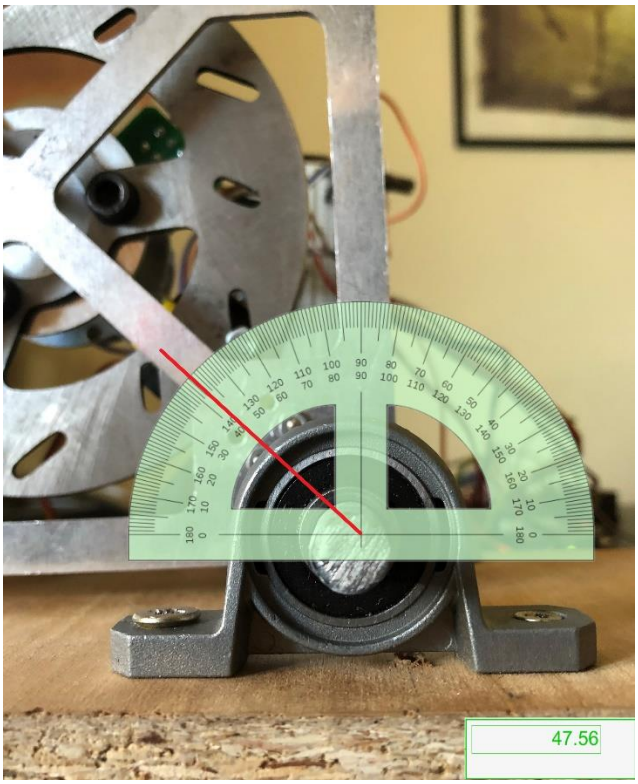


Ilustración 49: Comprobación ángulo izquierdo

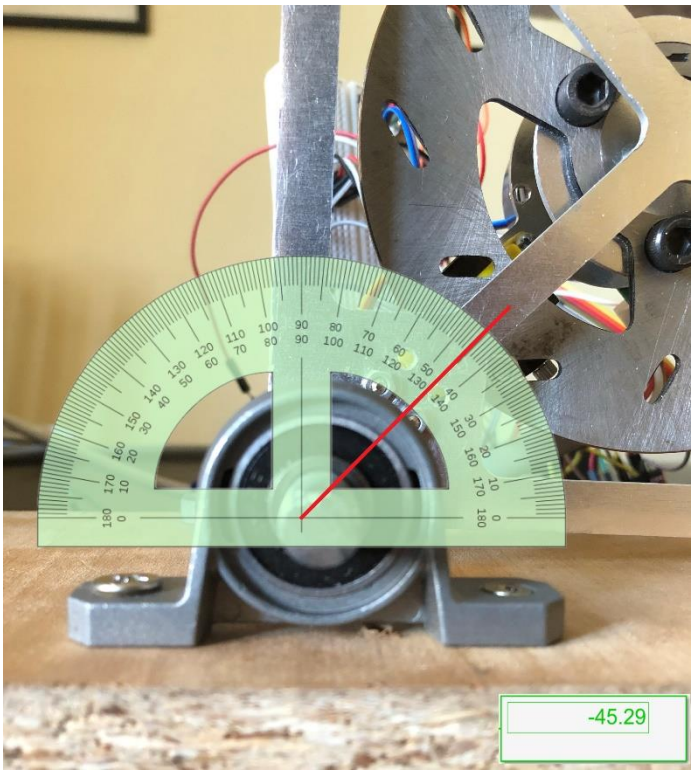


Ilustración 50: Comprobación ángulo derecho

Como se puede determinar a partir de las imágenes, las medidas del ángulo en reposo funcionana dentro del margen por lo que se da como satisfecho el objetivo.

6.2.3.2. Lectura de los giróscopos

Para una lectura más limpia de las velocidades de rotación, se ha hecho un filtrado de alta frecuencia, ya que los movimientos a reconocer son mecánicos y por tanto de una frecuencia mucho menor que el ruido existente. Para comprobar el funcionamiento correcto de ambos sensores se dibujan los dos en el mismo gráfico, con el fin de demostrar que miden lo mismo, exceptuando un offset de una respecto a la otra por su localización.

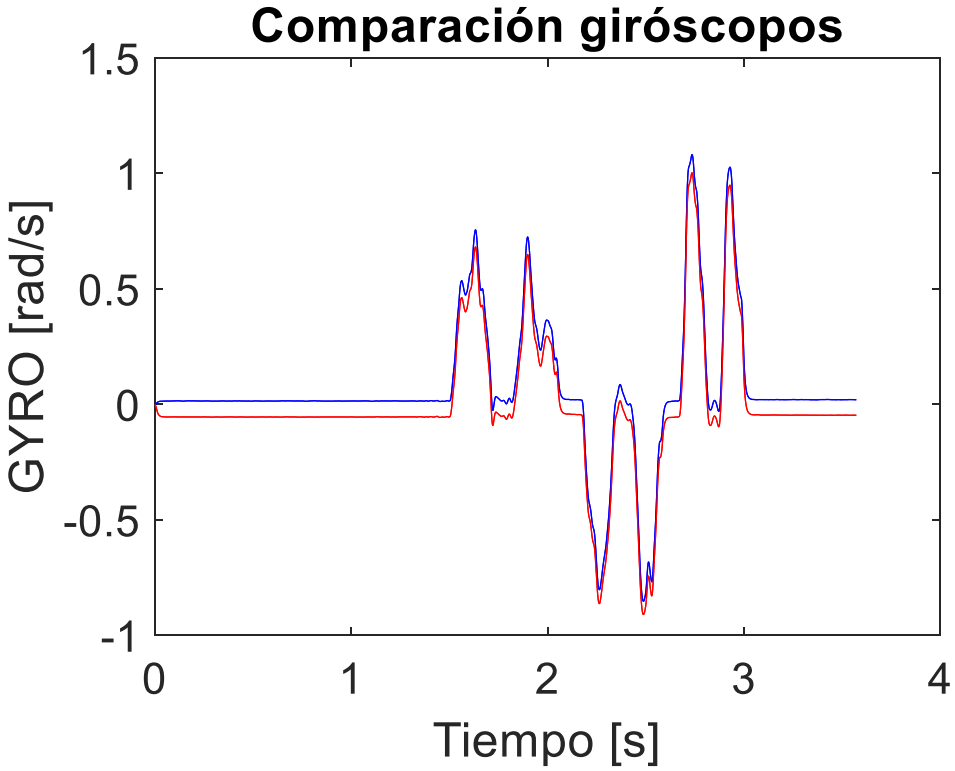


Ilustración 51: Comparación de la lectura de los giróscopos

Después de comprobar que las dos medidas son prácticamente idénticas, se ha de analizar su relación con el ángulo de inclinación, ya que éste variaría con posterioridad a la velocidad. Cuando la velocidad pasa por su valor nulo, el ángulo debería ser constante o encontrarse en un punto de inflexión.

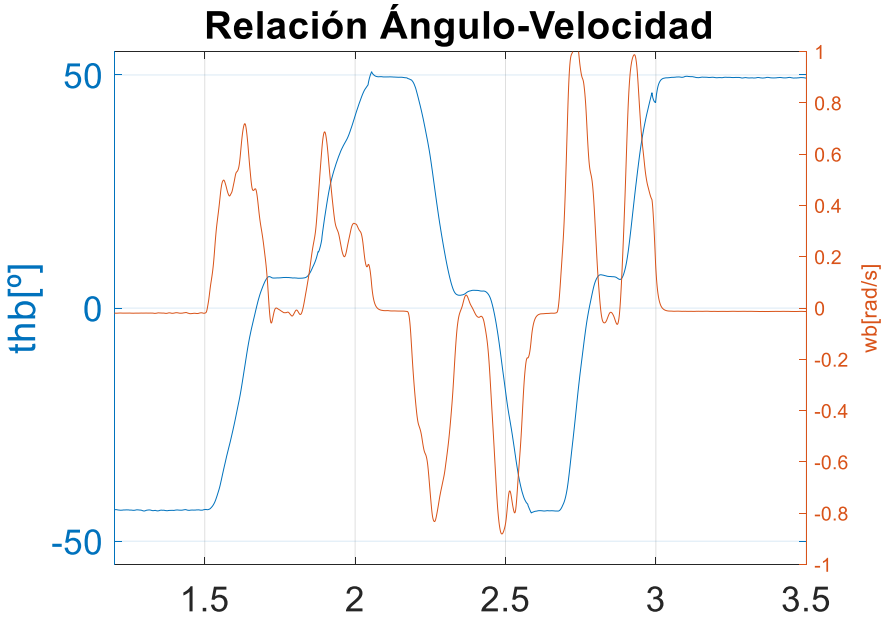


Ilustración 52: Relación entre el ángulo de inclinación y su velocidad angular

6.2.3.Máquina de estados

Es vital que en un proyecto electrónico la máquina de estados funcione correctamente, ya que es el proceso que coordina todos los demás. Si existe un fallo en la definición de los estados, el sistema no funcionará a pesar de que sus componentes lo hagan por separado. Para comprobar el correcto paso entre estados se ha instalado un LED RGB que cambie de color en función del estado en el que se encuentre. La prueba constaba de la ejecución de toda la secuencia añadiendo el LED para constatar que los procesos se ejecutaban en el estado correcto. La prueba fue satisfactoria por lo que se continua con las pruebas del control.

6.2.4.Control en sistema real

La implementación del control se generó un problema por el acople de los actuadores y sensores, así como unos requisitos de intensidad que la alimentación del motor no fue capaz de copar. Al acelerar el motor bruscamente, junto al volante de inercia, éste generaba unas vibraciones no despreciables, que a pesar de los filtros y acondicionamientos de las medidas seguían sin ser despreciables. Además del problema con las vibraciones, al exigir un control de la estructuraq lo mantenga en pie, el requerimiento de corriente es demasiado elevado para la alimentación del motor, que deja de regular y aumenta la tensión. Este aumento de tensión es detectado por el motor que corta el suministro por seguridad, siendo incapaces de controlar el sistema en estas condiciones. Por falta de tiempo para su resolución, debido a la fecha de entrega del proyecto, se apuntan estos errores para ser el punto de comienzo en futuras revisiones.

Capítulo 7: Conclusiones, aportaciones y futuros desarrollos

El objetivo de este proyecto ha sido la creación de una cara de un cubo con un motor unido a un volante de inercia pudiendo equilibrarse sobre un eje situado en una esquina. Además de ese balanceo, el conjunto debía elevarse hasta la posición de equilibrio desde el reposo mediante la aceleración y frenado brusco del volante. Además del fin último del prototipo funcional, ha sido de gran importancia el diseño tanto de la estructura en 3D como de un modelo fiable y el sistema de simulación para ser fácilmente perfeccionado en futuras aportaciones.

El primer sistema de soporte no era nada fiable y robusto al sólo estar compuesto por un rodamiento sin unión de apriete. Por ello, se modificó a un sistema de un eje introducido en dos soportes de rodamientos que le dan una mayor estabilidad y reducen el balanceo. Otro sistema que no presenta gran fiabilidad es el de frenado, para el que se diseñó una alternativa pero que no se pudo llevar a cabo físicamente. La mejora de este sistema se basa en desplazar la acción directa del servo a una indirecta, siendo éste el que mueve a la pieza que soporta la colisión y no él mismo.

La simulación del sistema ha sido correcta, llegando a implementar una simulación del ruido de medida. La simulación ha respondido correctamente a ángulos con gran diferencia respecto al punto de operación sin hacerse inestables, aunque se introduzca gran cantidad de ruido, aunque sí cuasi oscilantes. Por esto anterior se concluyen las siguientes aportaciones:

- Diseños de modelos 3D en Solid Edge válido para la fabricación y creación de planos de las piezas. Incluyendo diseños aplicables en futuros desarrollos.
- Modelado matemático del sistema incluyendo la generación de ruido de medida para un mayor realismo.
- Diseño de un control por realimentación de estados robusto ante ruido de medida.
- Creación de un estimador de estados que permita la obtención de las variables del control a partir de las medidas
    - En especial, la capacidad de determinación del ángulo de inclinación a partir de los acelerómetros, eliminando la perturbación por la aceleración angular de la estructura. También un estimador para el futuro proyecto de tres ejes por otro método.

- Creación de un software que permita la configuración de múltiples sensores IMU en un mismo proyecto, incluyendo la comunicación por el protocolo MAVLINK, al necesitar de varios códigos.
- Diseño de un entorno de simulación particularizado para el proyecto.
- Montaje de un prototipo y puesta en marcha de todos los componentes

A pesar de la resistencia probada a ruido de medida, el volante de inercia al acelerarse genera muchas vibraciones que afectan principalmente a los acelerómetros, que se encargan de proveer el ángulo de inclinación, medida vital para el sistema. Para reducir esta afección habría dos soluciones, mejorar el anclado, montaje y equilibrio del disco para que genere menos vibraciones, o poner una amortiguación a los sensores.

Habría varias posibles mejoras para el proyecto, desde perfeccionamiento del motor hasta implementación de diseños ya creados. En cuanto al control y modelado, si se pasase un software de optimización del control para las especificaciones se mejoraría la respuesta. Además, se mejorarían los parámetros del modelo con una versión más reciente del programa y mediante pruebas de identificación de parámetros. En cuanto al montaje, hay dos principales mejoras si se fabrican los diseños creados durante este proyecto. El primero sería del sistema de frenado y el otro es la cara auxiliar y el nuevo soporte para el motor, que permitiría montar todos los componentes en el prototipo.

Referencias

[1] F. Bote Ortega, «Modelado y control del péndulo invertido sobre carro mediante sistemas híbridos,» *Universidad de Sevilla,* 2012.

[2] M. Valera, M. Valles y M. Cardo, «Desarrollo y control de un péndulo de Furuta,» *Universidad Politécnica de Valencia,* 2002.

[3] M. González-Fierro Palacios, «Modelado dinámico de Robots Humanoides mediante álgebra espacial,» *Dep. Ing. de Sistemas y Automática, Univ. Carlos III Madrid,* 2009.

[4] ETH Zurich, «Cubli:Research D'Andrea,» Institute For Dynamic Systems and Control: ETH Zurich, Febrero 2011. [En línea]. Available: http://www.idsc.ethz.ch/research-dandrea/research-projects/cubli.html. [Último acceso: 25 Enero 2018].

[5] L. Cora Ibarra y E. Zapico, «Análisis de factibilidad de guiado de un cohete sonda mediante actuadores pirotécnicos de tipo Vernier,» *Tercer Congreso Argentino de Ingeniería Aeronáutica,* 2014.

[6] CSAIL MIT, «M-Blocks Modular Robotics: Research CSAIL,» MIT Computer Science & Artificial Intelligence Lab, 8 Diciembre 2018. [En línea]. Available: https://www.csail.mit.edu/research/m-blocks-modular-robotics. [Último acceso: 25 Enero 2018].

[7] E. Ackerman, «NASA Funds Robotic Tumbling Cubes for Space Exploration: IEEE Spectrum,» IEEE Spectrum, 13 Agosto 2014. [En línea]. Available: https://spectrum.ieee.org/automaton/robotics/military-robots/nasa-funds-robotic-tumbling-cubes-for-space-exploration. [Último acceso: 31 Enero 2018].

[8] R. Linares Ruiz, J. A. Quijano Vásquez y G. A. Holguín Lodoño, «Implementación del protocolo Bluetooth para la conexión inalámbrica de dispositivos electrónicos programables,» *Universidad technológica de Pereira,* 1995.

[9] A. Serna Ruiz, F. A. Ros García y J. C. Rico Noguera, «Guía práctica de sensores,» *Creaciones Copyright*, 2010.

[10] A. Cama Pinto, E. De la Hoz Franco y D. Cama Pinto, «Las redes de sensores inalámbricos y el internet de las cosas,» *INGE CUC*, vol. 8, nº 1, pp. 163-172, 2012.

[11] C. Pose, J. C. J. Giribet y A. Ghersin, «Sistema de navegación, guiado y control para un VANT multirotor.,» 2014.

[12] E. W. Weisstein, «Euler-Lagrange Diferential Equation,» MathWorld, [En línea]. Available: http://mathworld.wolfram.com/Euler-LagrangeDifferentialEquation.html.

[13] M. Gajamohan, M. Merz, I. Thommen y R. D'Andrea, *The Cubli: A Cube that can Jump Up and Balance*, 2012.

[14] J. A. Acosta, «Furutas's Pendulum: A Conservative Nonlinear Model for Theory and Practise,» Hindawi Publishing Corporation, Sevilla, 2009.

[15] R. C. Salguero, Sistemas Mecánicos SubActuados, Universidad Nacional de Ingeniería, Facultad de Ingeniería Eléctrica y Electrónica, Marzo 2001.

[16] M. Antonio Cruz, C. Márquez Sánchez, R. Silva Ortigoza y C. A. Merlo Zapata, «Sistemas mecánicos subactuados: péndulos invertidos,» 10 Enero 2014. [En línea]. Available: http://www.boletin.upiita.ipn.mx/index.php/ciencia/553-cyt-numero-41/840-sistemas-mecanicos-subactuados-pendulos-invertidos1.

[17] E. Bjerke y B. Pehrsson, «Developement of a Nonlinear Mechatronic Cube,» *Department of Signal and Systems, Division of Automatic control, Automation and Mechatronics, Chalmers University of Technology*, 2016.

Anexo A: Código fuente

Código del diseño del control

```
%% GENERAL PARAMETERS
% Control sampling time (s)
CONTROL.PARAM.SAMPLING_TIME = single(MODEL.PARAM.SAMPLING_TIME);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %% MODEL PARAMETERS
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s = tf('s');
% Gravedad [m/s^2]
g = MODEL.PARAM.GRAVITY;
CONTROL.PARAM.GRAVITY = g;
% Distancia punto anclaje a eje motor (OO') [m]
l = MODEL.PARAM.WHEEL_DISTANCE;
% Distancia punto anclaje a centro de gravedad cuerpo cubli (0G) [m]
lb = MODEL.PARAM.BODY_DISTANCE;
% Masa cuerpo del cubli [kg]
mb = MODEL.PARAM.BODY_MASS;
% Momento inercia cuerpo cubli [kg*m^2]
Ib = MODEL.PARAM.BODY_INERTIA;
% Coeficiente dinamico friccion cuerpo cubli
Cb = MODEL.PARAM.BODY_VISCOUS_FRICTION_COEFF;
% Masa volante de inercia + rotor motor [kg]
mw = MODEL.PARAM.WHEEL_MASS;
% Momento inercia volante + rotor [kg*m^2]
Iw = MODEL.PARAM.WHEEL_INERTIA;
% Coeficiente dinamico friccion volante + rotor
Cw = MODEL.PARAM.WHEEL_VISCOUS_FRICTION_COEFF;
% Constante de par de motor DC [N*m/A]
Kt = MODEL.PARAM.MOMENTUM_MOTOR_CONST;
% MAX and MIN CURRENT
MAX_CURRENT = MODEL.PARAM.MAX_CURRENT;
MIN_CURRENT = MODEL.PARAM.MIN_CURRENT;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONTROL_POSITION_MODE: 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 RATE CONTROL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONTROL_CURRENT_MODE:  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_CCD_MODE (CASCADE):  0. NOT APPLIED   /  1. APPLIED
% ANOTHER VALUE: APPLIED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CONTROL_MODE_WHEEL = 1;
CONTROL_MODE_ANG = 6;
if CONTROL_MODE_WHEEL<1 || CONTROL_MODE_WHEEL>6 || CONTROL_MODE_ANG<1 || CONTROL_MODE_ANG>6
    CONTROL_CCD_MODE = 0;
else
    CONTROL_CCD_MODE = 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Control design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ts=MODEL.PARAM.SAMPLING_TIME;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plant model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Operating point %%%%%
X0=MODEL.PARAM.X0;
U0=MODEL.PARAM.U0;
[matA,matB,matC,matD]=linmod('MODEL',X0,U0);
% State variables are organized as: X=[thb(t) wb(t) ww(t)]
% Levitator transfer function in continuous time
P=ss(matA,matB,matC,matD);
matP=tf(P);
% Natural frequency and damping factor
[wnP,setaP]=damp(eig(matA));
```

```
%Transfer function between Input current and angle
Plant=matP(1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Discrete plant
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z=tf('z',ts);
% 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;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Natural frequency and damping factor of dominant plant poles
[wnPlant,setaPlant]=damp(Plant);
wn=1.2*wnPlant(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)) -wnPlant(1)];
% Closed loop poles in discrete time
polosd_lc=exp(ts*polos_lc);
% Gain matrix design
K=place(matAd,matBd,polosd_lc);
Ka = K;
%% Integral control for reference specification
% % Matrices ampliadas
% matAad=[matAd zeros(3,1) ; -ts*matCd(2,:) eye(1)];
% matBad=[matBd; -ts*matDd(2,:)];
% % Pulsacion natural y amortiguamiento de los polos complejos dominantes
% wn=0.75*wnP(1);
% seta=0.5;
% % Polos de lazo cerrado en tiempo continuo
% polos_lc=[wn*(-seta+1j*sqrt(1-seta^2)) wn*(-seta-1j*sqrt(1-seta^2)) -wn -15*wn];
% % Polos de lazo cerrado en tiempo discreto
% polosd_lc=exp(ts*polos_lc);
% % Diseño del control
% Ka=place(matAad,matBad,polosd_lc);
% % colocar los polos
% K=Ka(:,1:3);
% Ki=Ka(:,4);
%% PID PARAM
CONTROL.PARAM.CONTROL_MODE_ANG=uint8(1);
% Control pid parameters
% CONTROL.PARAM.K_1=K_1;
CONTROL.PARAM.K_ANG=single(Ka(2)/Ka(1));
CONTROL.PARAM.Ti_ANG=single(1000);
CONTROL.PARAM.Td_ANG=single((Ka(3))/(Ka(2)));
CONTROL.PARAM.b_ANG=single(0);
CONTROL.PARAM.N_ANG=single(10000);
CONTROL.PARAM.MAX_CONTROL_ANG=single(9000*2*pi/60);
CONTROL.PARAM.MIN_CONTROL_ANG=single(-9000*2*pi/60);
CONTROL.PARAM.DER_INPUT_ANG=0;
CONTROL.PARAM.ANTIWINDUP_ANG=0;
% Discrete integral and derivative calculations
CONTROL.PARAM.DER_DISC_TYPE_ANG=uint8(4);
CONTROL.PARAM.INT_DISC_TYPE_ANG=uint8(3);
CONTROL.PARAM.CONTROL_MODE_WHEEL=uint8(1);
% Control parameters
% CONTROL.PARAM.K_2=K_2;
CONTROL.PARAM.K_WHEEL=single(Ka(1));
CONTROL.PARAM.Ti_WHEEL=single(1000);
CONTROL.PARAM.Td_WHEEL=single(0);
CONTROL.PARAM.b_WHEEL=single(1);
CONTROL.PARAM.N_WHEEL=single(100);
CONTROL.PARAM.MAX_CONTROL_WHEEL=single(7);
CONTROL.PARAM.MIN_CONTROL_WHEEL=single(-7);
CONTROL.PARAM.ANTIWINDUP_WHEEL=0;
CONTROL.PARAM.DER_INPUT_WHEEL=0;
% Discrete integral and derivative calculations
CONTROL.PARAM.DER_DISC_TYPE_WHEEL=uint8(0);
CONTROL.PARAM.INT_DISC_TYPE_WHEEL=uint8(0);
% Feedback state control parameters
%CONTROL.PARAM.K_WHEELNT=single(Ki);
CONTROL.PARAM.K_P=single(K);
% Control mode for cascade control
CONTROL.PARAM.CONTROL_CCD_MODE=uint8(CONTROL_CCD_MODE);
%% INPUT
% TARGET
CONTROL.INPUT.WHEEL_TARGET=single(0);
```

```
CONTROL.INPUT.CURRENT_TARGET=MODEL.PARAM.U0;
% MEASURED AND ESTIMATED VARIABLES
CONTROL.INPUT.SPEED_ENABLE = single(0);
CONTROL.INPUT.IMU.BODY_ANGLE = single(0);
CONTROL.INPUT.IMU.GYRO_MEAN(3) = single(0);
CONTROL.INPUT.MOTOR_CURRENT = single(0);
CONTROL.INPUT.MOTOR_SPEED = single(0);
% BUTTON
CONTROL.INPUT.BUTTON=logical(0);
%% OUTPUT
% CONTROL VARIABLES
CONTROL.OUTPUT.EC_MOTOR.CURRENT = single(0);
%% STATE
% SYSTEM STATUS
% / 0. STOP                 / 1. ACCELERATION
% / 2. BREAK                / 3. CALIBRATION
% / 4. BALANCE
CONTROL.STATE.CURRENT_STATUS = uint8(0);
CONTROL.STATE.PREVIOUS_STATUS = uint8(0);
% CONTROL MODE
% / 0. INITIALIZATION / 1. ESTIMATION    / 2. CASCADE CONTROL
CONTROL.STATE.CONTROL_MODE = uint8(0);
CONTROL.STATE.SAMPLING_COUNT=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);
```

Ejemplo de tratamiento de medidas

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FILTRADO Y ESCALADO DE LAS MEDIDAS DE UNA IMU
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ACCEL_LAST,GYRO_LAST,GYRO_MEAN,ACCEL_MEAN] = ...
          IMU_SCALING(RAW_GYRO,RAW_ACCEL,ORIENTATION,...
                      ACCEL_SSF,GYRO_SSF,GYRO_FILT_FREQ,ACCEL_FILT_FREQ,...
                      GRAVITY,IMU_SAMPLING_TIME)
%% PARAMETERS
GYRO_ALFA_FILT = exp(-IMU_SAMPLING_TIME*2*pi*GYRO_FILT_FREQ);
ACCEL_ALFA_FILT = exp(-IMU_SAMPLING_TIME*2*pi*ACCEL_FILT_FREQ);
%% INITIALIZATION
persistent initialize ACCEL_FILT ACCEL_BUFFER ...
                      GYRO_BUFFER GYRO_FILT
if isempty(initialize)
    ACCEL_FILT = single([0 0 GRAVITY]');
    ACCEL_BUFFER = single(ones(9,1)*[0 0 GRAVITY]);
    GYRO_FILT = single(zeros(3,1));
    GYRO_BUFFER = single(zeros(9,3));
    initialize = 1;
end
%% IMU ORIENTATION
switch ORIENTATION
    case 0 % None
        % Tests
    case 8 % Roll180l
        RAW_ACCEL = int16([1 -1 1]').*RAW_ACCEL;
        RAW_GYRO = int16([1 -1 -1]').*RAW_GYRO;
%         RAW_ACCEL_MEAN = int16([1 -1 1]').*RAW_ACCEL_MEAN;
%         RAW_GYRO_MEAN = int16([1 -1 -1]').*RAW_GYRO_MEAN;
    case 14 % Roll180Yaw270
        RAW_ACCEL = int16([-1 -1 1]').*RAW_ACCEL([2 1 3]);
        RAW_GYRO = -RAW_GYRO([2 1 3]);
%         RAW_ACCEL_MEAN = int16([-1 -1 1]').*RAW_ACCEL_MEAN([2 1 3]);
%         RAW_GYRO_MEAN = -RAW_GYRO_MEAN([2 1 3]);
    case 39 %Roll270Pitch215
        S=[-cos(pi/4) sin(pi/4) 0;0 0 1;sin(pi/4) cos(pi/4) 0];%Matriz de ro-
tación
        RAW_ACCEL = int16(S*single(RAW_ACCEL));
        RAW_GYRO = int16(S*single(RAW_GYRO));
    otherwise
end
%% IMU SCALING
ACCEL = single(RAW_ACCEL)/ACCEL_SSF*GRAVITY; % + ACCEL_OFFS; % m/s^2
GYRO  = single(RAW_GYRO)/GYRO_SSF*pi/180; % + GYRO_OFFS; % rad/s
% ACCEL_MEAN = single(RAW_ACCEL_MEAN)/ACCEL_SSF*GRAVITY; % + ACCEL_OFFS; %
m/s^2
% GYRO_MEAN  = single(RAW_GYRO_MEAN)/GYRO_SSF*pi/180; % + GYRO_OFFS; % rad/s
%% FIRST-ORDER LINEAR ADDITIONAL FILTER
ACCEL_FILT = ACCEL_ALFA_FILT*ACCEL_FILT + (1-ACCEL_ALFA_FILT)*ACCEL;
GYRO_FILT = GYRO_ALFA_FILT*GYRO_FILT + (1-GYRO_ALFA_FILT)*GYRO;
%% MEAN
ACCEL_MEAN = mean([ACCEL_BUFFER; ACCEL_FILT'])';
GYRO_MEAN = mean([GYRO_BUFFER; GYRO_FILT'])';
%% BUFFER UPDATE
ACCEL_BUFFER = [ACCEL_BUFFER(2:end,:);ACCEL_FILT'];
GYRO_BUFFER = [GYRO_BUFFER(2:end,:);GYRO_FILT'];
%% OUTPUT
ACCEL_LAST = ACCEL_FILT;
GYRO_LAST = GYRO_FILT;
```

Anexo B: Datasheets

En esta sección se adjuntan los enlaces a las hojas de datos de los componentes electrónicos utilizados para la realización de este proyecto:

1. Raspberry Pi Zero W
   https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-zero.pdf
2. Motor EC 45 flat 50 W
   https://www.maxonmotor.es/maxon/view/product/motor/ecmotor/ecflat/ecflat45/387250?etcc_cu=onsite&etcc_med=Header%20Suche&etcc_cmp=mit%20Ergebnis&etcc_ctv=Layer&query=EC%2045%20flat%2050%20W
3. ESCON Module 36/3
   https://www.maxonmotor.es/maxon/view/product/control/4-Q-Servokontroller/414533?etcc_cu=onsite&etcc_med=Header%20Suche&etcc_cmp=mit%20Ergebnis&etcc_ctv=Layer&query=ESCON%2036
4. Servomotor MG90S
   http://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf
5. IMU MPU6050 Datasheet
   https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
6. IMU MPU6050 Register Map
   https://www.olimex.com/Products/Modules/Sensors/MOD-MPU6050/resources/RM-MPU-60xxA_rev_4.pdf

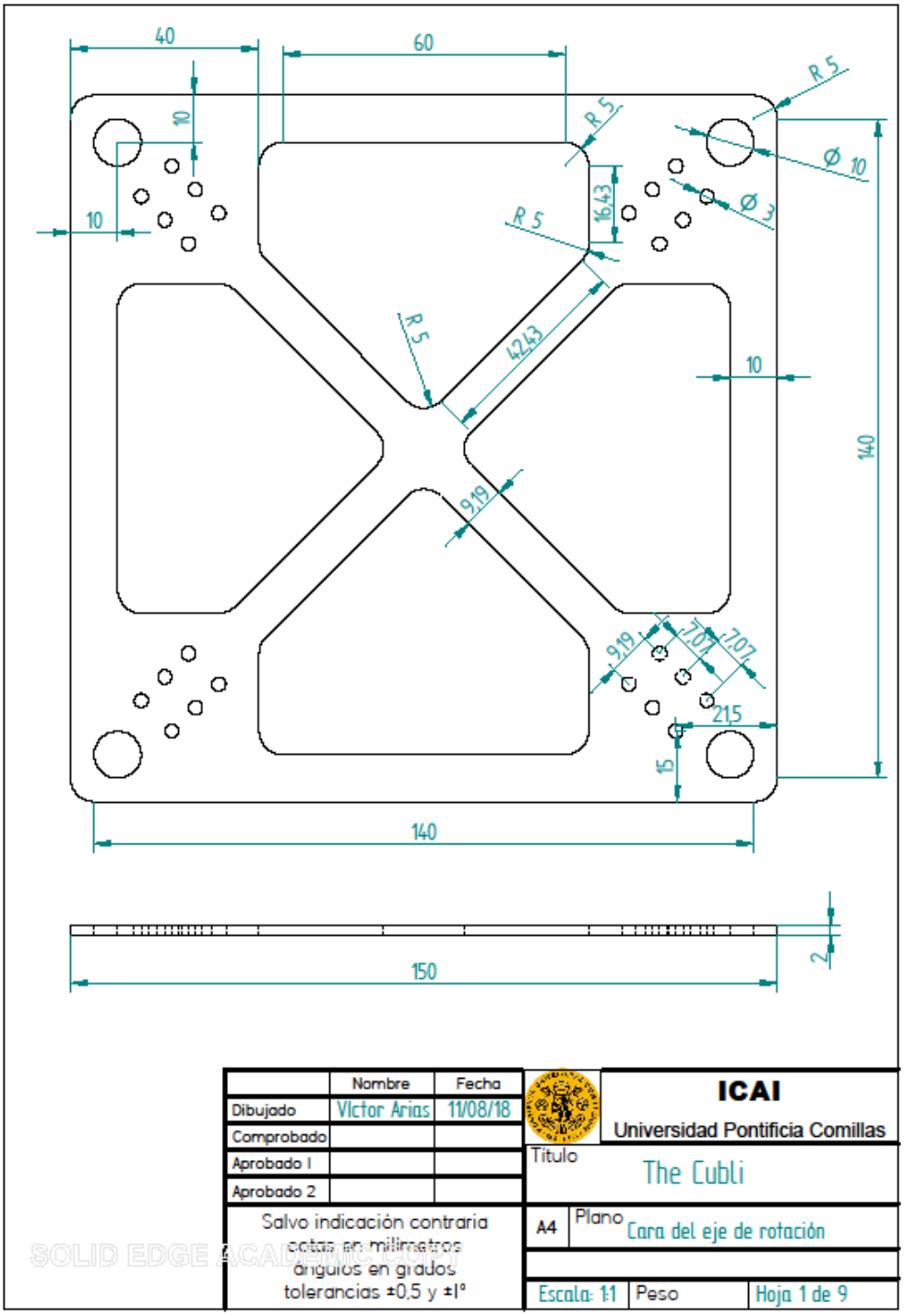
(Esta página se ha dejado en blanco a propósito)

Parte 2: Planos

(Esta página se ha dejado en blanco a propósito)

Capítulo 1: Planos de piezas

1.1. Cara del eje de rotación



	Nombre	Fecha
Dibujado	Víctor Arias	11/08/18
Comprobado		
Aprobado 1		
Aprobado 2		

Salvo indicación contraria
cotas en milímetros
ángulos en grados
tolerancias ±0,5 y ±1°

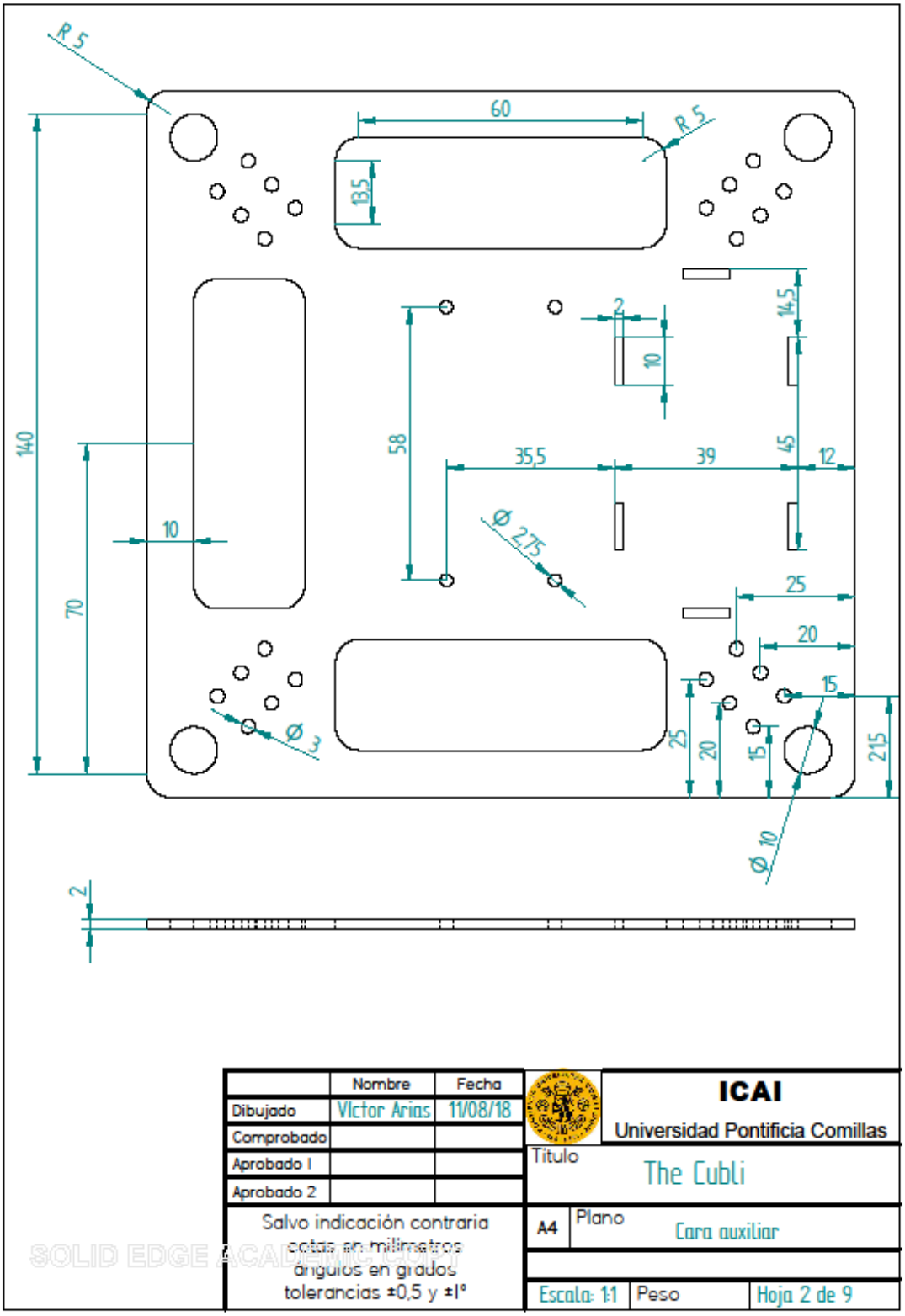
ICAI
Universidad Pontificia Comillas

Título: The Cubli

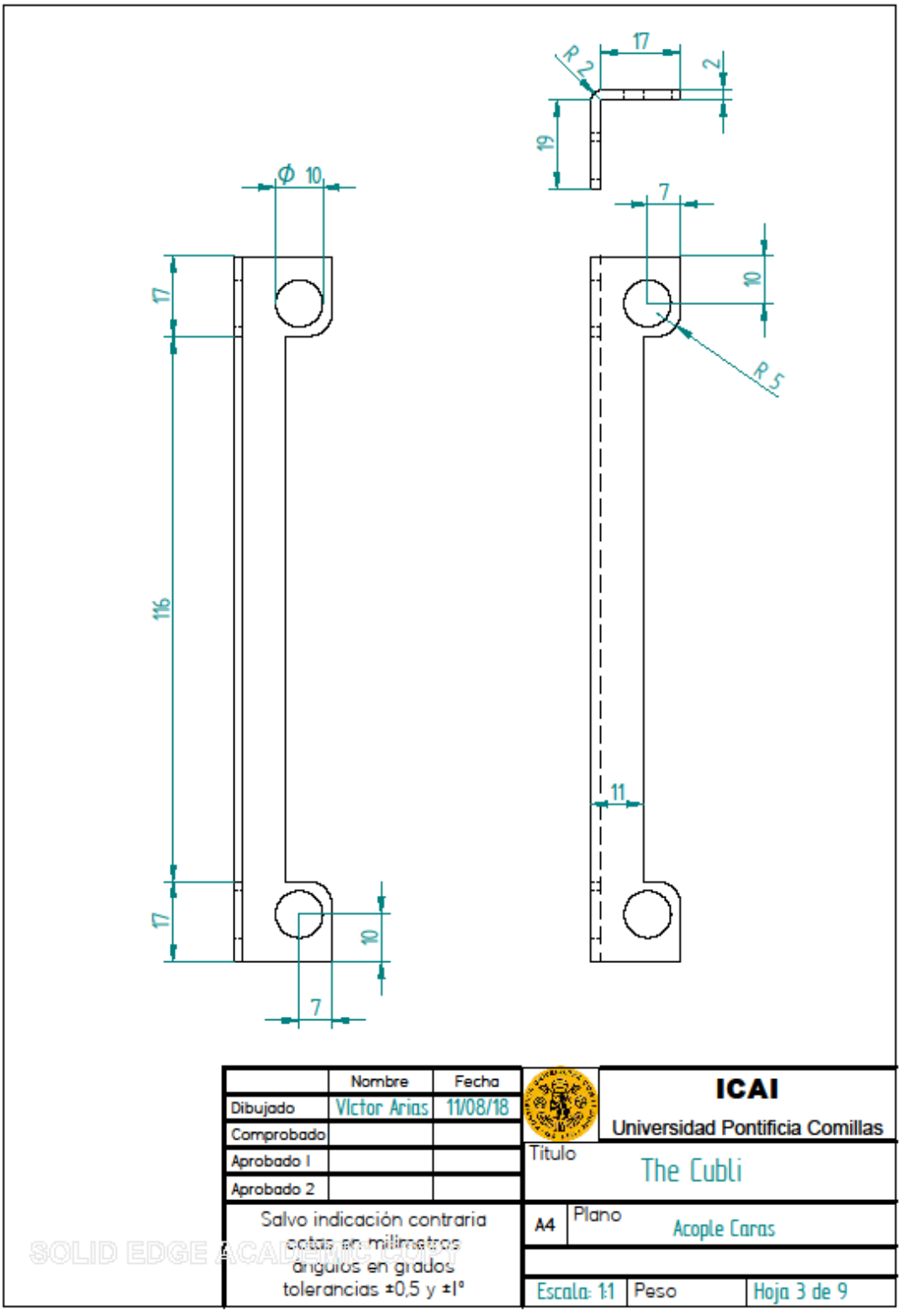
A4 | Plano: Cara del eje de rotación

Escala: 1:1 | Peso | Hoja 1 de 9

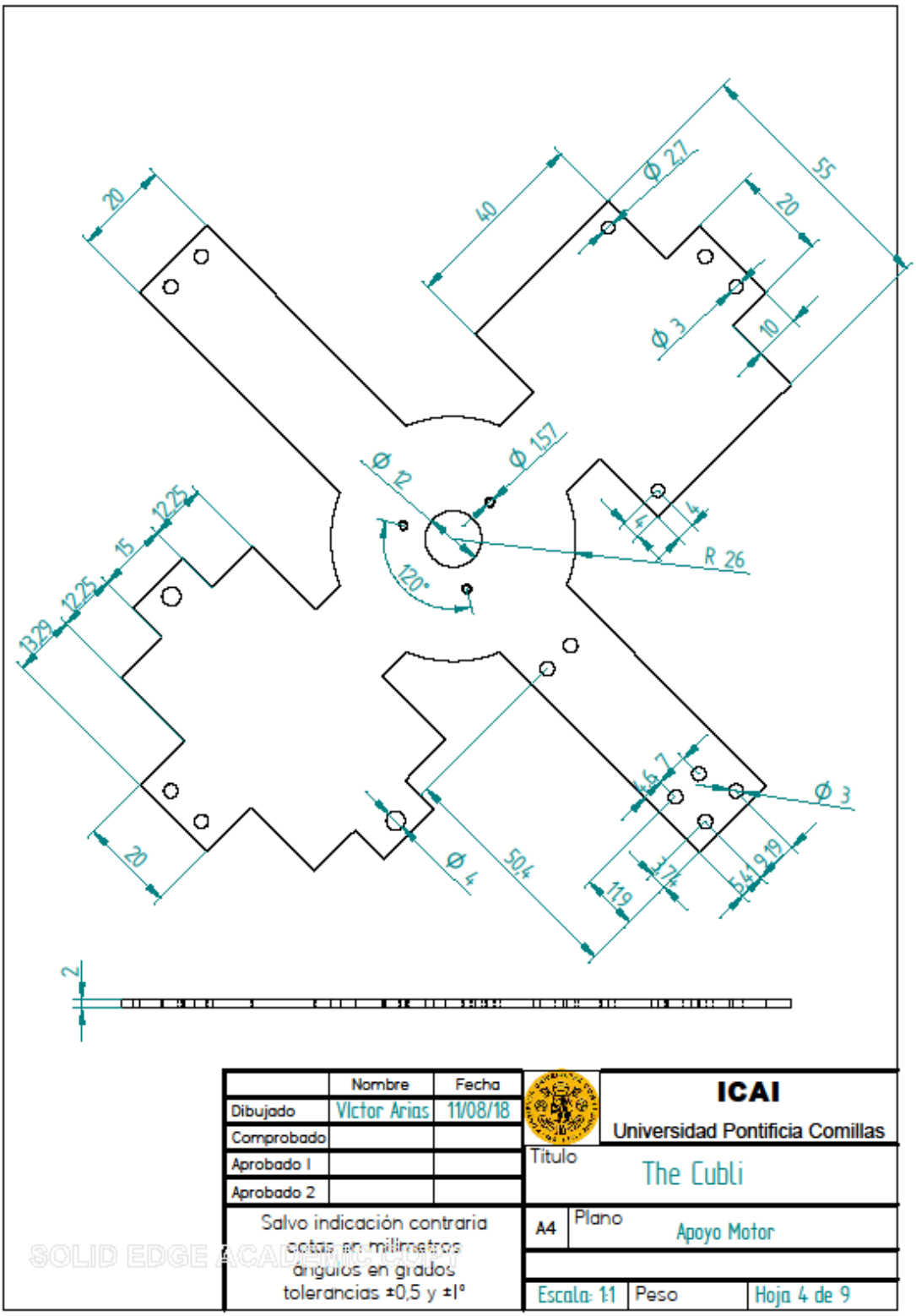
1.2. Cara auxiliar



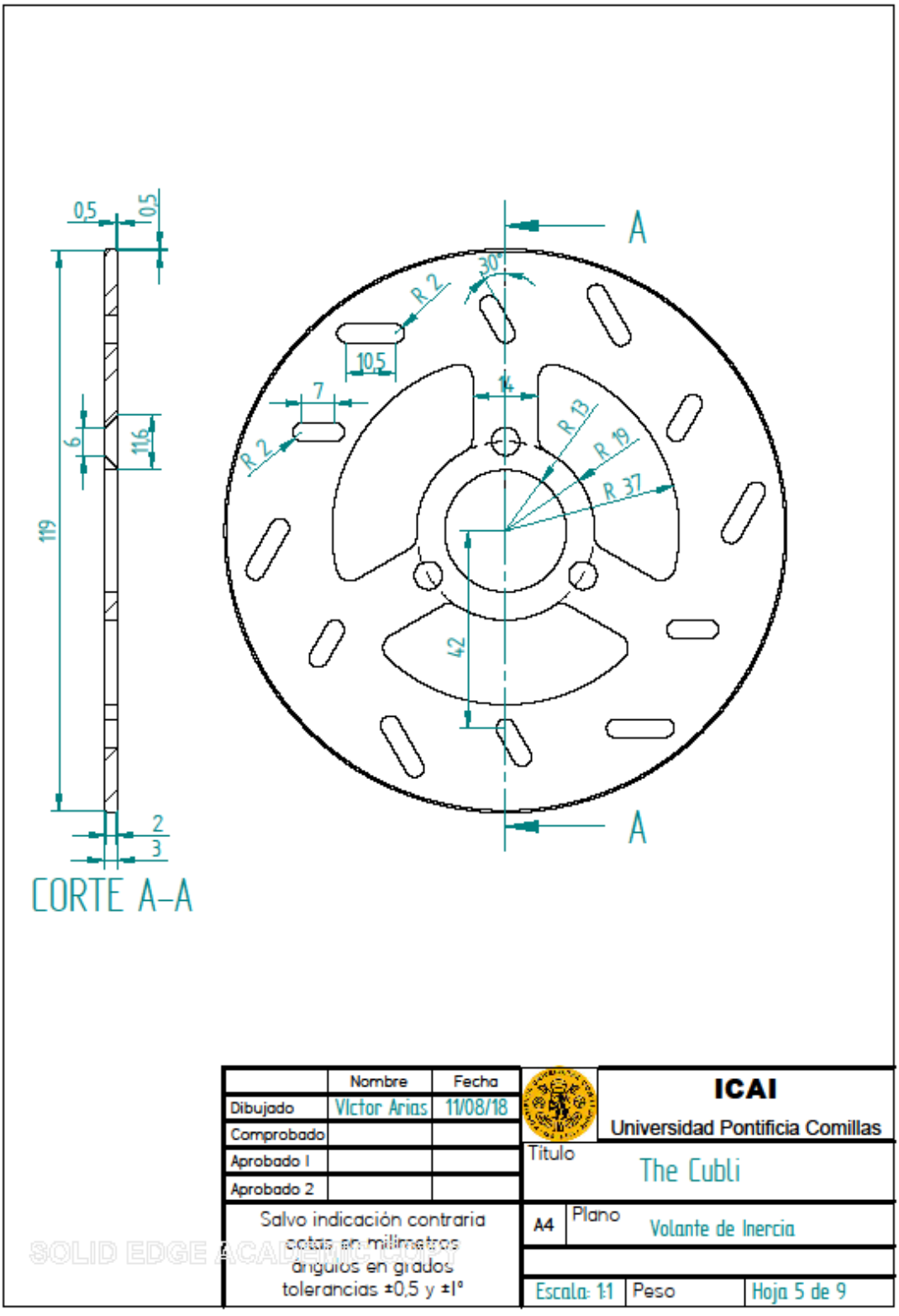
1.3. Acoplamiento de caras



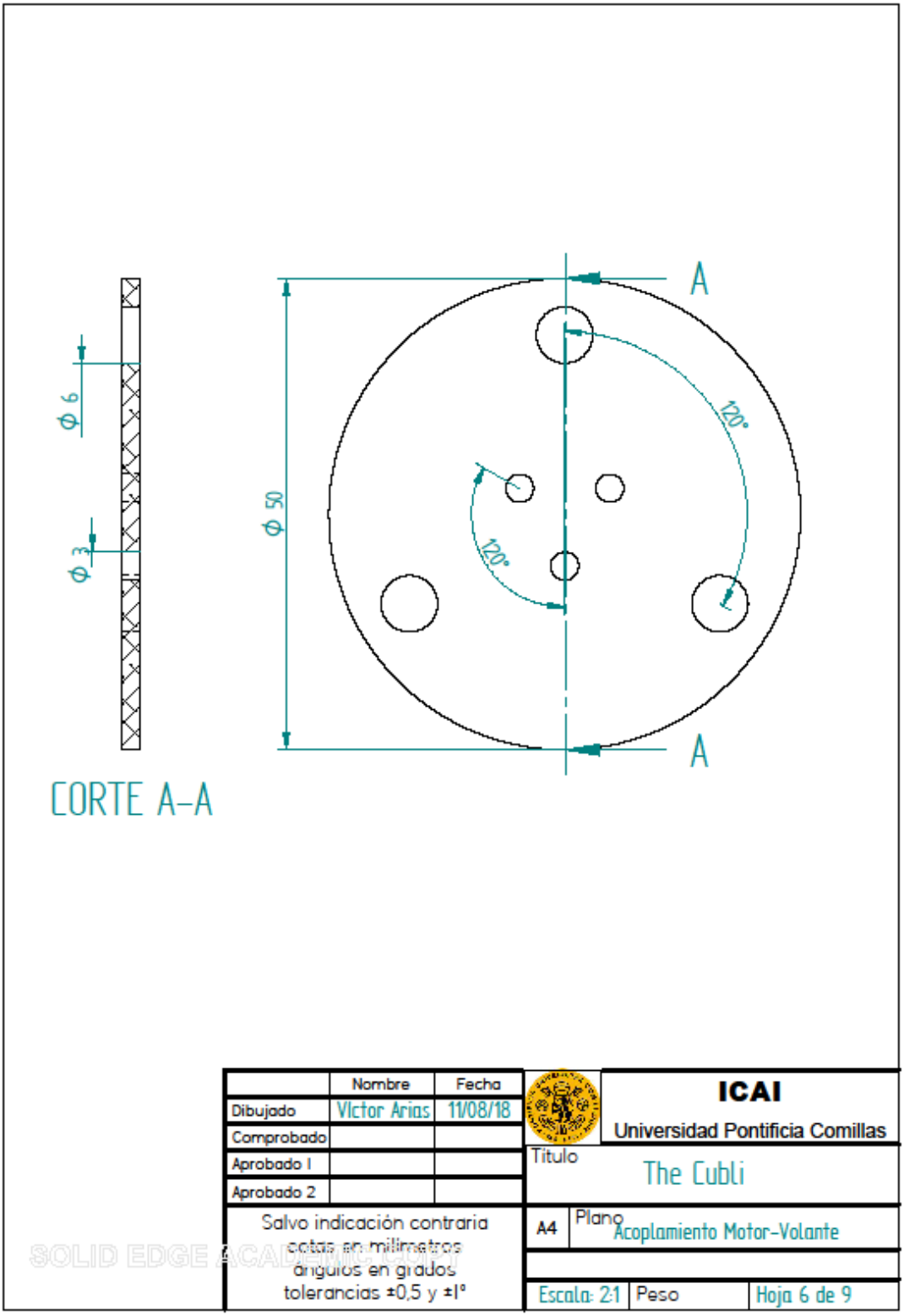
1.4. Apoyo del motor



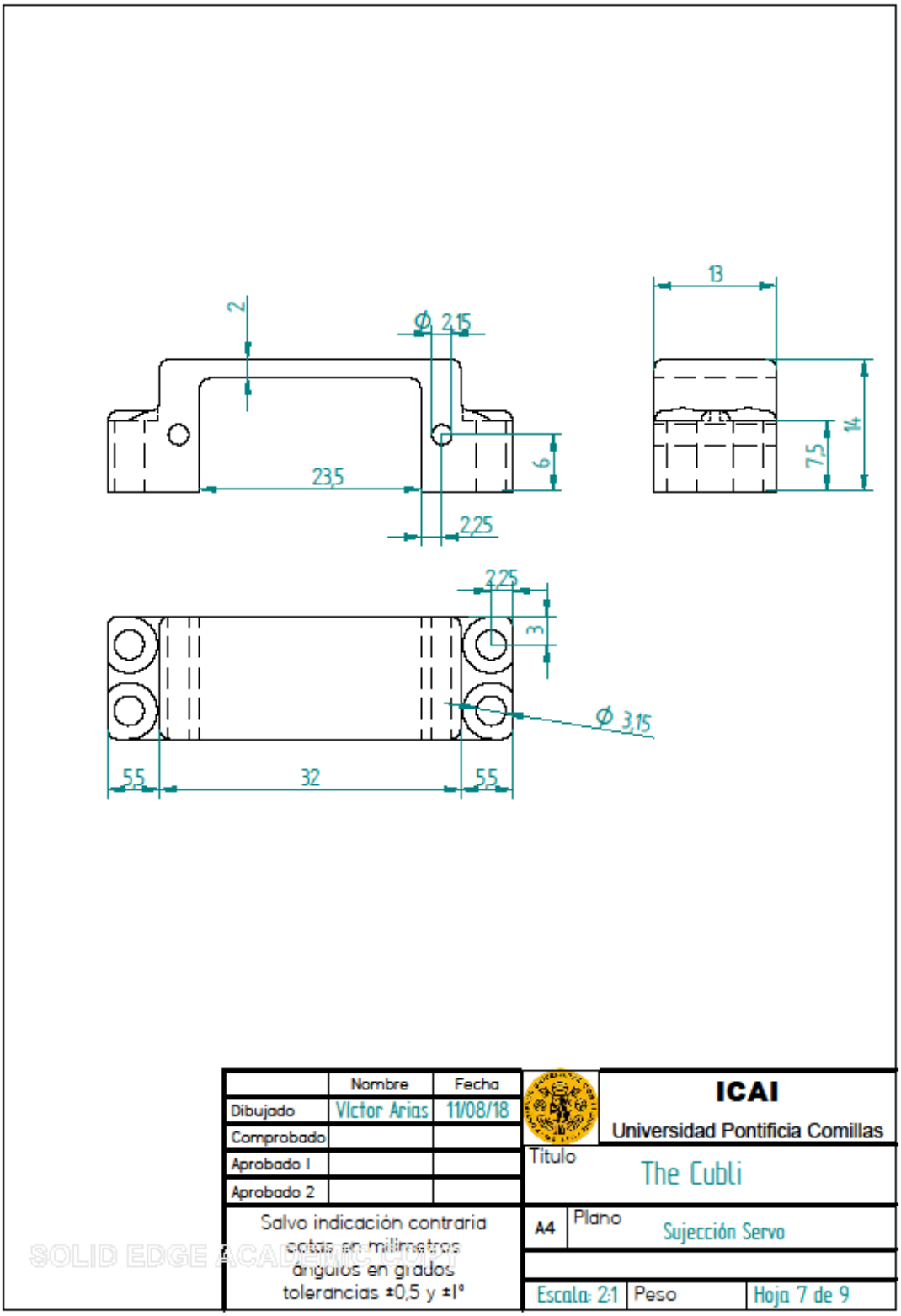
1.5. Volante de inercia



1.6. Acoplamiento del motor al volante de inercia



1.7. Sujeción del servomotor



	Nombre	Fecha
Dibujado	Víctor Arias	11/08/18
Comprobado		
Aprobado 1		
Aprobado 2		

Salvo indicación contraria
cotas en milímetros
ángulos en grados
tolerancias ±0,5 y ±1°

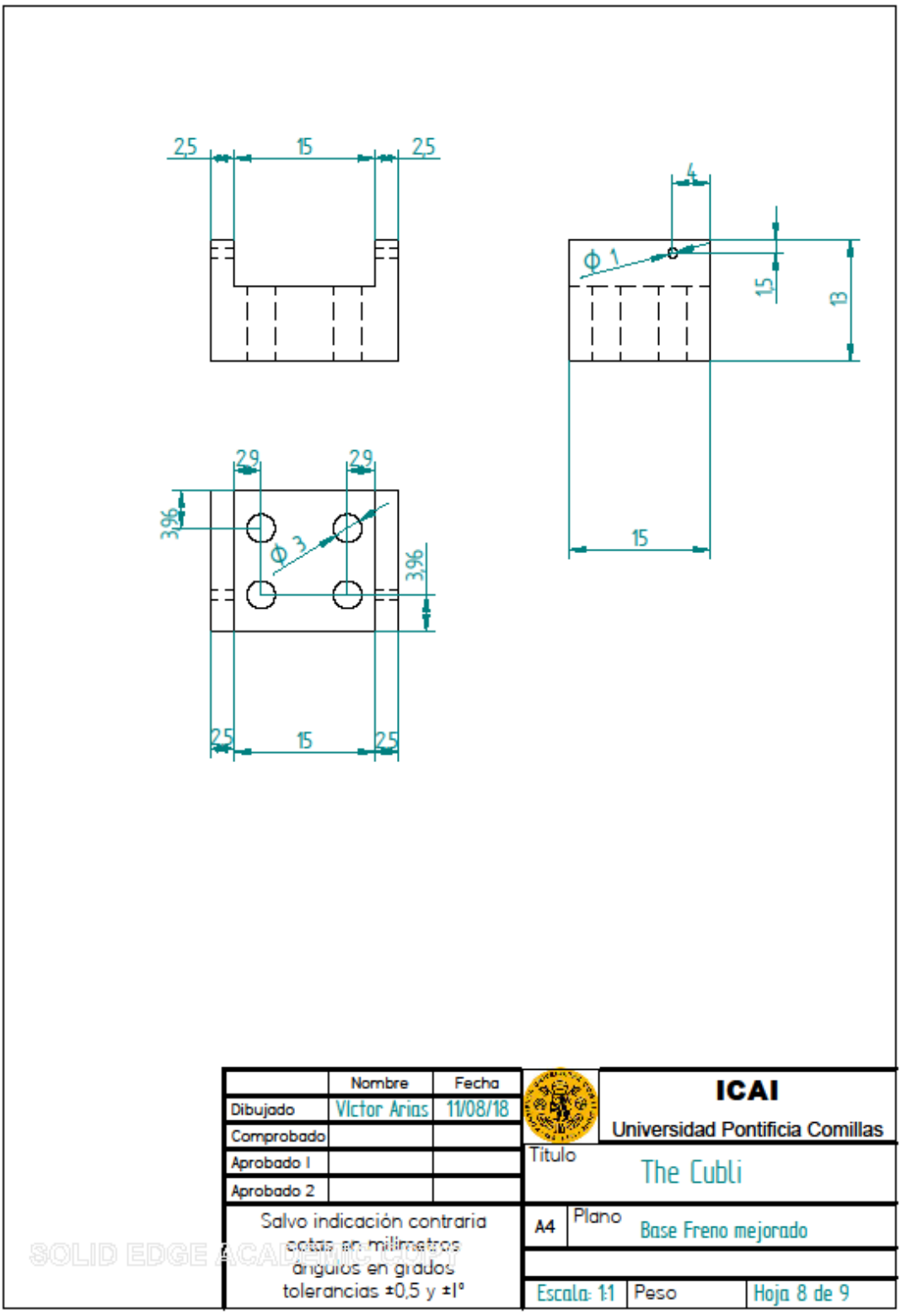
ICAI
Universidad Pontificia Comillas

Título: The Cubli

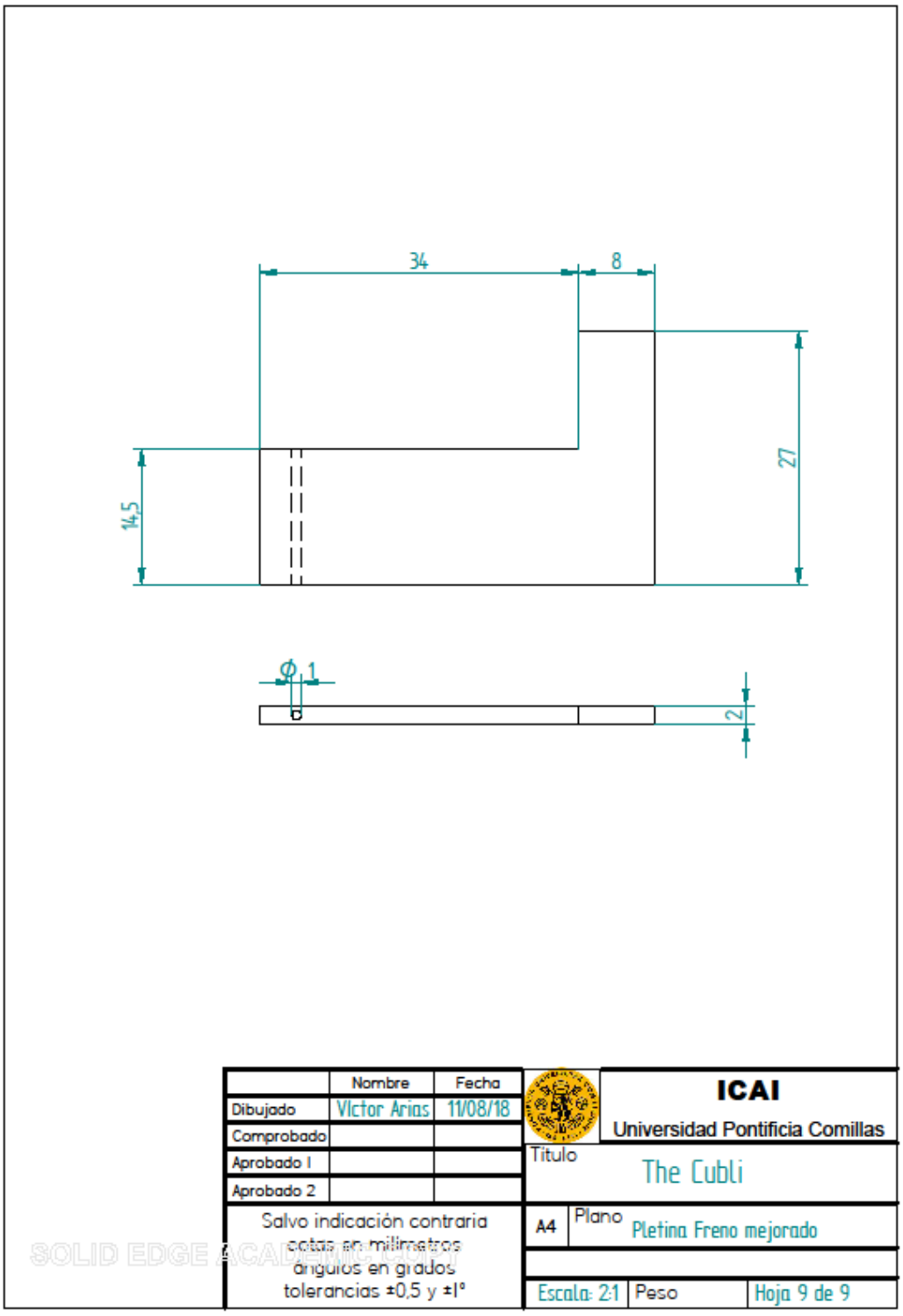
A4 | Plano: Sujección Servo

Escala: 2:1 | Peso | Hoja 7 de 9

1.8. Base del sistema de freno mejorado



1.9. Pletina del sistema de freno mejorado



Capítulo 2: Esquemas electrónicos

2.1. Conexionado IMU y servomotor

Las conexiones de la Raspberry con las IMUs son bastante sencillas al sólo constar de los dos cables para el bus I2C, que ambas comparten, y la alimentación. Con el servomotor es incluso más simple, ya que además de la alimentación sólo necesita la conexión del PWM.

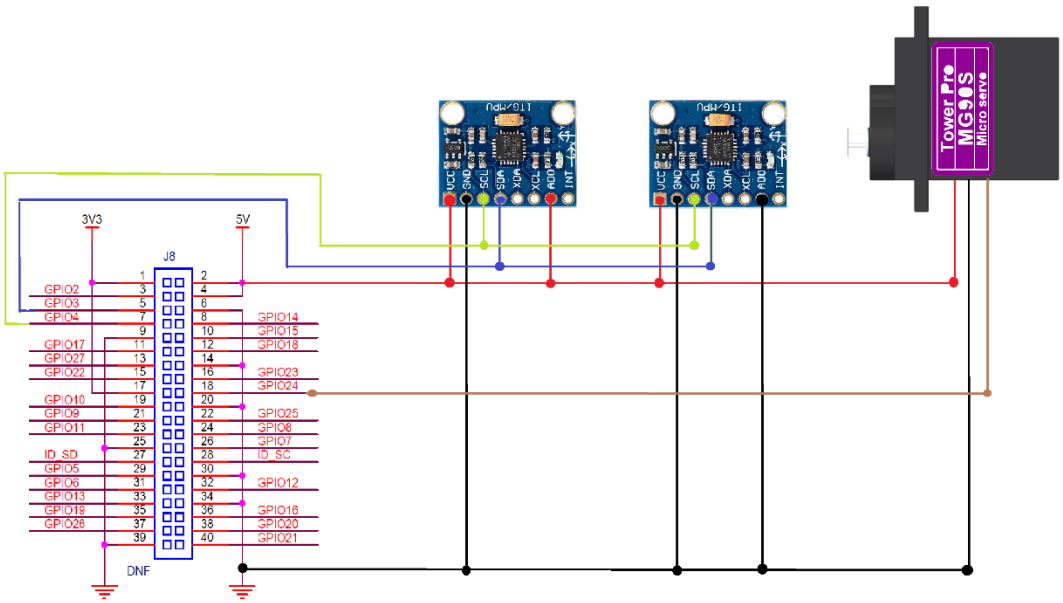
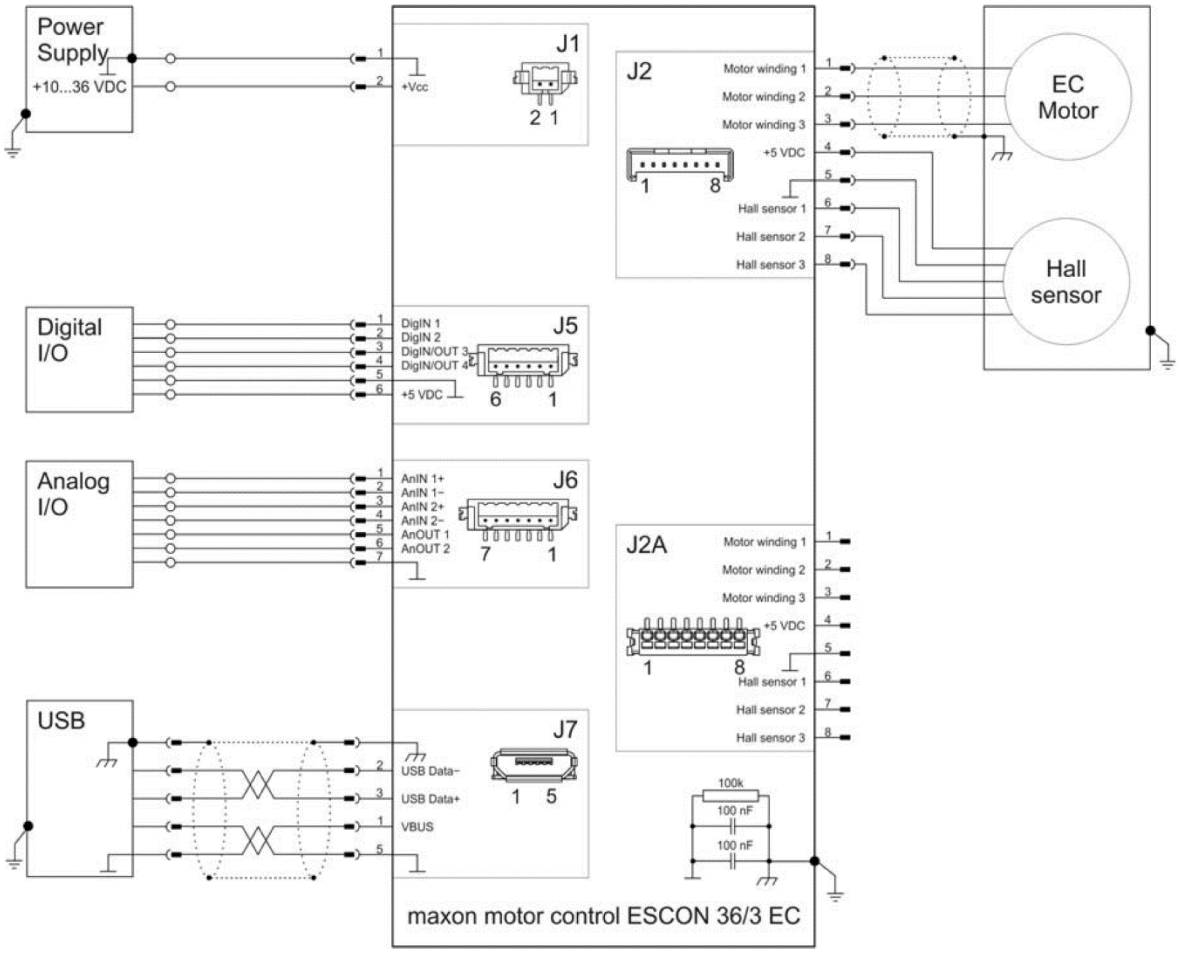


Ilustración 53: Esquema eléctrico IMUs y servomotor

2.2. Conexionado ESCON Module 36/3

2.3. Conexionado Motor EC a ESCON 36/3

Los conectores para el motor, denominados J2 y J2A, son redundantes con conexiones diferentes para utilizar uno u otro según el conector que se prefiera. En nuestro caso se usó el J2A por la facilidad de sujeción sin componentes a añadir.



(Esta página se ha dejado en blanco a propósito)

Parte 3: Presupuesto

(Esta página se ha dejado en blanco a propósito)

Capítulo 1: Mediciones

1.1. Componentes principales

CÓDIGO	COMPONENTES	CANTIDAD
0101	Lámina de aluminio 300x300x2 mm	1
0102	Servo MG90s	1
0103	Motor Maxon EC 45 flat 50W, con sensores hall	1
0104	Raspberry Pi Zero W	1
0105	ESCON Module 36/3	1
0106	Batería Li-Po 1500mAh	1
0107	Disco freno minimoto 120 mm	1
0108	Giróscopo/Acelerómetro MPU6050	2

1.2. Componentes del montaje

CÓDIGO	COMPONENTES	CANTIDAD
0201	Juego 10 cables macho-hembra 200 mm	2
0202	Tornillos M3	10
0203	Tuercas M3	4
0204	Tornillos M2	2
0205	Tuercas M2	2
0206	Tornillos M6	3
0207	Tuercas M6	3
0208	Separador 25mm H/M M3	4

1.3. Equipo y herramientas

CÓDIGO	ELEMENTO	CANTIDAD
0301	Ordenador	1
0302	Osciloscopio 2 canales	1
0303	Polímetro	1
0304	Soldador	1
0305	Otras herramientas	1

1.4. Software

CÓDIGO	PROGRAMA	CANTIDAD
0401	Matlab / Simulink Student	1
0402	Solid Edge ST9	1
0403	Escon Studio	1

1.5. Mano de obra

CÓDIGO	PROGRAMA	HORAS
0501	Diseño hardware	100
0502	Montaje hardware	5
0503	Diseño del control	150
0504	Pruebas y solución de problemas	300
0505	Documentación del proyecto	100

Capítulo 2: Precios unitarios

2.1. Componentes principales

CÓDIGO	COMPONENTES	€/UNIDAD
0101	Lámina de aluminio 300x300x2 mm	14.66
0102	Servo MG90s	4.90
0103	Motor Maxon EC 45 flat 50W, con sensores hall	143.36
0104	Raspberry Pi Zero W	10.53
0105	ESCON Module 36/3	145.09
0106	Batería Li-Po 1500mAh	12.99
0107	Disco freno minimoto 120 mm	7.56
0108	Giróscopo/Acelerómetro MPU6050	10.54

2.2. Componentes del montaje

CÓDIGO	COMPONENTES	€/UNIDAD
0201	Juego 10 cables macho-hembra 200 mm	2.90
0202	Tornillos M3	0.0291
0203	Tuercas M3	0.0256
0204	Tornillos M2	0.0291
0205	Tuercas M2	0.0256
0206	Tornillos M6	0.0291
0207	Tuercas M6	0.0256
0208	Separador 25mm H/M M3	0.1324

2.3. Equipo y herramientas

CÓDIGO	ELEMENTO	€/UNIDAD
0301	Ordenador	499.00
0302	Osciloscopio 2 canales	456.00
0303	Polímetro	19.70
0304	Soldador	7.48
0305	Otras herramientas	97.00

2.4. Software

CÓDIGO	PROGRAMA	€/UNIDAD
0401	Matlab / Simulink student	125.00
0402	Solid Edge ST9	100.00
0403	Escon Studio	0.00

2.5. Mano de obra

CÓDIGO	PROGRAMA	€/HORA
0501	Diseño hardware	50.00
0502	Montaje hardware	20.00
0503	Diseño del control	40.00
0504	Pruebas y solución de problemas	60.00
0505	Documentación del proyecto	40.00

Capítulo 3: Sumas parciales

3.1. Componentes principales

CÓDIGO	COMPONENTES	CANTIDAD	€/UNIDAD	COSTE TOTAL (€)
0101	Lámina de aluminio 300x300x2 mm	1	14.66	14.66
0102	Servo MG90s	1	4.90	4.90
0103	Motor Maxon EC 45 flat 50W, con sensores hall	1	143.36	143.36
0104	Raspberry Pi Zero W	1	10.53	10.53
0105	ESCON Module 36/3	1	145.09	145.09
0106	Batería Li-Po 1500mAh	1	12,99	12.99
0107	Disco freno minimoto 120 mm	1	7.56	7.56
0108	Giróscopo/Acelerómetro MPU6050	2	10.54	21.08
			Total	**360.37**

3.2. Componentes del montaje

CÓDIGO	COMPONENTES	CANTIDAD	€/UNIDAD	COSTE TOTAL (€)
0201	Juego 10 cables macho-hembra 200 mm	2	2.90	4.80
0202	Tornillos M3	10	0.0291	0.291
0203	Tuercas M3	4	0.0256	0.124
0204	Tornillos M2	2	0.0291	0.0582
0205	Tuercas M2	2	0.0256	0.0512
0206	Tornillos M6	3	0.0291	0.0873
0207	Tuercas M6	3	0.0256	0.0768
0208	Separador 25mm H/M M3	4	0.1324	0.5296
			Total	**6.02**

3.3. Equipo y herramientas

CÓDIGO	ELEMENTO	CANTIDAD	€/UNIDAD	COSTE TOTAL (€)
0301	Ordenador	1	499.00	499.00
0302	Osciloscopio 2 canales	1	456.00	456.00
0303	Polímetro	1	19.70	19.70
0304	Soldador	1	7.48	7.45
0305	Otras herramientas	1	97.00	97.00
			Total	**1,079.15**

3.4. Software

CÓDIGO	PROGRAMA	CANTIDAD	€/UNIDAD	COSTE TOTAL (€)
0401	Matlab / Simulink Student	1	125.00	125.00
0402	Solid Edge ST9	1	100.00	100.00
0403	Escon Studio	1	0.00	0.00
			Total	**225.00**

3.5. Mano de obra

	PROGRAMA	HORAS	€/UNIDAD	COSTE TOTAL (€)
0501	Diseño hardware	100	50.00	5,000
0502	Montaje hardware	5	20.00	100
0503	Diseño del control	150	40.00	6,000
0504	Pruebas y solución de problemas	300	50.00	15,000
0505	Documentación del proyecto	100	40.00	4,000
			Total	**30,100.00**

Capítulo 4: Presupuesto general

CONCEPTO	**COSTE (€)**
COMPONENTES PRINCIPALES	360.37
COMPONENTES DEL MONTAJE	6.02
EQUIPO Y HERRAMIENTAS	1,079.15
SOFTWARE	225.00
MANO DE OBRA	30,100.00
TOTAL	**31,770.54**