UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Diseño de controlador PID para plataforma giroestabilizada

Pablo Tofé Blanco
2015

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Diseño de controlador PID para plataforma giroestabilizada

Autor: Pablo Tofé Blanco

Director: Iván García Daza

TRIBUNAL:

Presidente: David Fernandez Llorca

Vocal 1º: Miguel Ángel Sotelo Vázquez

Vocal 2º: Iván García Daza

CALIFICACIÓN: .....
FECHA: .....

"A mis abuelos, a mis padres y hermana por su apoyo incondicional;

A los que ya no están que nunca se irán;

A Paloma por su apoyo especial y aguantarme todos estos años;

A todos mis amigos y esos buenos momentos que pasamos juntos;

A los amigos que me llevo de esta etapa de mi vida, en especial a Rubén por su gran ayuda incondicional, y a Gonzalo por las largas y nebulosas tardes de estudio mano a mano;

A todas las largas horas que han hecho esto posible;

Por último y no por eso menos especial, agradecer a Iván García el haberme guiado y brindado la posibilidad de realizar este proyecto."

"El mundo un grano de polvo en el espacio; La ciencia de los hombres palabra;

Los pueblos, los animales y las flores de los siete climas son sombras de la nada.

Quiero al amante que gime de felicidad y desprecio al hipócrita que reza una plegaria."

José Monge Cruz

Índice general

I. RESUMEN 5
II. ABSTRACT 7
III. RESUMEN EXTENDIDO 9
IV. MEMORIA 11
1. CAPÍTULO Introducción 13
1.1. Motivaciones 13
1.2. Estado del arte 14
1.2.1. Estabilización inercial mecánica 14
1.2.2. Estabilización inercial con sistema de control 15
1.2.3. Estabilización de imagen 16
1.3. Objetivos perseguidos en el trabajo 17
1.4. Estructura de la memoria 18
2. CÁPITULO Descripción del sistema hardware 19
2.1. Descripción del sistema 19
2.2. Controladora Basecam Electronics BGC 3.1 MOS 20
2.3. Microcontrolador 21
2.4. Inertial Measurement Unit (IMU) 23
2.4.1. Acelerómetros 24
2.4.2. Giroscopios 25
2.4.3. InvenSense MPU-6050 26
2.5. Motores DC Brushless (BLDC) 27
2.5.1. Comparativa con otros motores eléctricos 28
2.5.2. Principio de funcionamiento 29
2.5.3. Secuencias de alimentación 30
2.5.4. Motores BLDC del sistema gimbal 31
2.6. Etapa de potencia 32
2.6.1. Topología del inversor 32
2.6.2. Etapa de potencia de la controladora BGC 3.1 33
3. CÁPITULO Teoría de acondicionamiento de sensores 35
3.1. Teoría del acondicionamiento de acelerómetros 35
3.2. Teoría del acondicionamiento de giroscopios 37
3.3. Problemas derivados de la naturaleza de los sensores 38
3.4. Filtro Complementario 39
3.4.1. Filtro Complementario para control en posición 39
3.4.2. Filtro Complementario para control en velocidad angular 42
4. CÁPITULO Estrategia de control 43
4.1. Planteamiento del sistema de control 43
4.2. Controlador PID 47
4.3. Modelado de la planta 49
5. CAPÍTULO Desarrollo experimental 53

5.1. Adquisición de datos del MPU-6050 53
5.1.1. Acondicionamiento de los acelerómetros 55
5.1.2. Acondicionamiento de los giroscopios 57
5.1.3. Filtrado de señales de los sensores 60
5.2. Manejo de motores brushless BLDC 62
5.3. Comparativa de filtrado de señales 64
5.4. Identificación del modelo de la planta 67
5.5. Simulación del controlador PID 73
5.6. Desarrollo y funcionamiento del controlador PID 76
6. CAPÍTULO Conclusiones y trabajos futuros 81
V. CONTENIDO DEL CD 83
VI. BIBLIOGRAFÍA 85

Índice de figuras

Figura III.1 Representación de los ángulos de orientación Roll, Pitch y Yaw de un vehículo aéreo. 10
Figura IV.1 Sistema de estabilización inercial manual con contrapesos. 14
Figura IV.2 Sistema de estabilización manual (steadycam) con sistema de control. 15
Figura IV.3 Gimbal 3 axis con motores Brushless. 16
Figura IV.4 BaseCam SimpleBGC 32-bit 16
Figura IV.5 Plataforma gimbal usada en el proyecto. 20
Figura IV.6 Tarjeta controladora BGC 3.1 tipo MOS de BaseCam Electronics [12]. 21
Figura IV.7 Diagrama de bloques del procesador ATMEL AVR de 8 bits [7] 22
Figura IV.8 IMU 6 DOF comercial modelo MPU6050 [8]. 23
Figura IV.9 Vista interior de un sensor IMU. 24
Figura IV.10 Imagen tomada con microscopio electrónico de un Acelerómetro MEMS [2]. 24
Figura IV.11 Imagen ampliada de un giroscopio MEMS. 25
Figura IV.12 Imagen ampliada de la sección de un MPU6050 26
Figura IV.13 Bobinado del estator de un motor brushless 29
Figura IV.14 Topología de un motor brushless. 30
Figura IV.15 Secuencias de alimentación de las fases de un motor brushless trifásico [1]. 30
Figura IV.16 Motor brushless comercial. 31
Figura IV.17 Esquema de un inversor trifásico cualquiera conectado a motor de tres fases. 32
Figura IV.18 Explicación gráfica del funcionamiento de un acelerómetro [3]. 36
Figura IV.19 Diagrama vectorial de las aceleraciones en un acelerómetro. 37
Figura IV.20 Diagrama de bloques del filtro complementario [4]. 40
Figura IV.21 Esquema del filtro complementario [4]. 41
Figura IV.22 Ruido leído con el acelerómetro en reposo. 44
Figura IV.23 Ruido leído con el giroscopio en reposo. 44
Figura IV.24 Histograma del ruido en el acelerómetro. 45
Figura IV.25 Histograma del ruido en el giroscopio. 46
Figura IV.26 Diagrama de bloques del LAZO CERRADO de control. 47
Figura IV.27 Diagrama de bloques de la planta en LAZO ABIERTO. 49
Figura IV.28 Abrir el GUI de identificación de sistemas. 50
Figura IV.29 Ventana principal del GUI System Identification Toolbox de Matlab. 51
Figura IV.30 Orientation of Axes of Sensitivity and Polarity Rotation [8]. 54
Figura IV.31 Colocación de la IMU en la plataforma gimbal. 54
Figura IV.32 Fragmento del Datasheet del acelerómetro del MPU6050 [8]. 57
Figura IV.33 Fragmento del Datasheet del giroscopio del MPU6050 [8]. 59
Figura IV.34 Filtro complementario para el control en posición. 61
Figura IV.35 Filtro complementario para el control en velocidad angular. 62
Figura IV.36 Medida del giroscopio en el eje pitch. 65
Figura IV.37 Medida del acelerómetro en el eje pitch. 66
Figura IV.38 Medida del giroscopio en el eje roll. 66
Figura IV.39 Medida del acelerómetro en el eje roll. 67
Figura IV.40 Ensayo en lazo abierto del motor de pitch. 68

Figura IV.41 Respuesta transitoria del modelo de primer orden ante entrada escalón (Pitch).....70
Figura IV.42 Representación de los polos y ceros del sistema de primer orden (Pitch) 70
Figura IV.43 Ensayo en lazo abierto del motor de roll. 71
Figura IV.44 Respuesta transitoria del modelo de primer orden ante una entrada escalón (Roll) 72
Figura IV.45 Representación de los polos y ceros del sistema de primer orden (Roll) 73
Figura IV.46 Diseño Simulink del controlador PID. 75
Figura IV.47 Respuesta ante entrada escalón de los PID diseñados para ambos ejes, Pitch y Roll......75
Figura IV.48 Señales del control ajustado en Lazo Cerrado.....79

Índice de tablas

Tabla IV-1: Características de los motores BLDC. 31
Tabla IV-2 Medias y desviaciones típicas de los sensores obtenidas con Matlab. 45
Tabla IV-3 Ganancias de los reguladores obtenidas mediante PID Tuner. 74
Tabla IV-4 Valores de las ganancias del PID obtenidos en el controlador real......78

I. RESUMEN

En el presente proyecto se desarrolla la programación y la puesta en marcha del software necesario para controlar electrónicamente el hardware de una plataforma estabilizadora de tipo gimbal en dos ejes de movimiento. Se pretende que ésta pueda controlarse de manera eficiente y ajustada.

El diseño de esta plataforma parte de la idea de poder acoplarse a un dron y poder obtener una imagen de forma nítida y estabilizada del campo de visión.

Para llevar a cabo el trabajo se desarrollará el software necesario que utilizará un procesador ATmega328P cuya entrada principal será una unidad de medición inercial (Inertial Measurement Unit – IMU) y mandará las señales pertinentes en un lazo de control a dos motores brushless. Todo ello en una plataforma estabilizadora de dos ejes comercial.

Palabras clave: Gimbal, IMU, Motores Brushless, modulación PWM, PID.

II. ABSTRACT

The following project develops the programming and implementation of the necessary software for electronical controll in a commercial hardware. This hardware is a stabilizing platform called two axis gimbal. It is intended that it can be controlled efficiently and adjusted.

The design of this platform starts with the idea of engaging with a drone and obtain a clear an stabilized field of view image.

To carry out this project, the necesary software will be developed by using an ATmega328P processor whose main entrance will be an inertial measurement unit (IMU). This processor will send appropiate signals in a control loop to two brushless engines. Everithing will be developed in a commercial stabilizing platform on two axis motion.

Keywords: Gimbal, IMU, Brushless engine, PWM modulation, PID.

III. RESUMEN EXTENDIDO

Hoy en día la tecnología más novedosa se incorpora en la construcción de circuitos integrados modernos. Dichos circuitos son capaces de dar prestaciones que hace años no eramos capaces de imaginar. Es el caso de los Micro-Electromechanical Systems: MEMS[1]. Componentes de un tamaño muy reducido que revolucionan el mundo de la electrónica. Gracias a estas nuevas tecnologías, los acelerómetros y los giróscopos on-chip ya sean en unidades separadas o conjuntas, han logrado formar parte de sistemas novedosos como las Unidades de Medición Inercial (IMU[2]) que hoy en día se encuentran en aplicaciones de diversas áreas como el entretenimiento, la investigación biomédica y la robótica, entre otras. El área de la investigación saca provecho cada día de dispositivos como las IMUs con su uso en vehículos autónomos, terrestres o aéreos.

En este trabajo se presenta la implementación del firmware de una plataforma giroestabilizada comercial. Éste tipo de plataformas se usa sobretodo en vehículos aéreos no tripulados y en este contexto; es normal que las imágenes obtenidas en ausencia de una giroestabilización se vean afectadas por los cambios bruscos de posición del vehículo, las vibraciones ocasionadas por sus motores y por las condiciones climatológicas del entorno. De acuerdo a esto, se pretende implementar una plataforma móvil giroestabilizada por medio de una IMU y dos motores brushless, que permita hacer correcciones automáticas de tal forma que se mantenga en todo momento en la posición horizontal y sin vibraciones ante cualquier tipo de perturbaciones del medio. Para lograr este cometido se ha elegido

[1] MEMS del inglés *Micro-Electromechanical Systems*
[2] IMU del inglés *Inertial Measurement Unit*

una IMU de seis grados de libertad que consta de giróscopos y acelerómetros de tipo MEMS en los tres ejes de orientación, para sensar la posición del vehículo en un momento determinado. Teóricamente, para detectar el vector de posición del avión son suficientes las lecturas del acelerómetro; pero referencias sobre este tipo de dispositivos indican que tales dispositivos son sensibles a los desplazamientos y al ruido y por tanto se hace necesario la integración de los datos arrojados por el giróscopo para determinar con mayor precisión el vector "corregido" de posición que corresponde a la inclinación del cuerpo. Además, los controles basados en giróscopos son de vital importancia en lo referente a instrumentos de navegación y mucho más en el caso de los vehículos aéreos, tripulados y no tripulados, ya que se usan para detectar y controlar valocidades angulares en los tres ángulos de orientación típicos de estas aeronaves. Estos tres ángulos de orientación se denominan técnicamente "pitch"(cabeceo), "yaw"(guiño) y "roll" (alabeo). Todos los ángulos y velocidades angulares en los tres ejes de orientación van a ser extraidos de las medidas de una unidad de medición inercial. Después estas señales deben pasar por una etapa de acondicionamiento y por otra etapa de filtrado que va a permitir desarrollar un software preciso y robusto para nuestra aplicación.

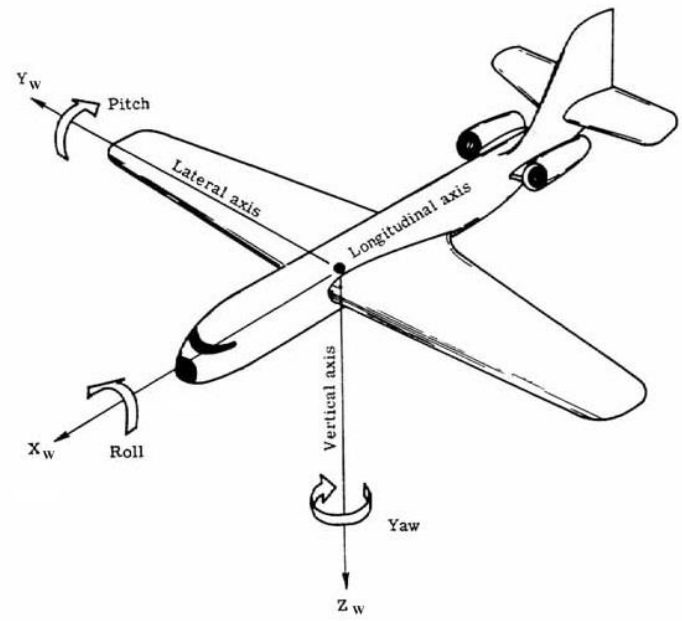

Figura III.1 Representación de los ángulos de orientación Roll, Pitch y Yaw de un vehículo aéreo.

En el presente trabajo se mostrarán los detalles de programación del microcontrolador, adquisición de datos y acondicionamiento de las señales del sensor, control de los actuadores y el desarrollo de la electrónica de control del sistema. El microcontrolador usado será de ocho bits y se aprovechará la potencia de Matlab para comprobar y analizar los resultados de los filtrados de las señales entregadas por los sensores, además de usarlo como herramienta para el diseño del lazo de control que gobernará el sistema.

IV. MEMORIA

1. CAPÍTULO

Introducción

En este capítulo se detallarán las distintas motivaciones que justifican la realización de este proyecto, así como los objetivos que se persiguen. También se desarrollará un apartado dedicado al estado del arte y se hará un esquema en el que se explica a grandes rasgos los distintos apartados documentados.

1.1. Motivaciones

La motivación a la programación de una plataforma gimbal viene dada por la necesidad de estabilizar una cámara acoplada a un vehículo no tripulado. Hoy en día un buen procesado de imagen es crucial para multitud de aplicaciones en diversos campos. Mediante visión artificial se es capaz de detectar cualquier objeto o carácter alfanumérico con poca luz, y a bastante distancia. Los vehículos autotripulados necesitan un sistema de visión para lograr un funcionamiento correcto. Para éstos sistemas y para muchos más, es necesaria una imagen estabilizada y nítida.

Gran parte de la investigación desarrollada en el presente trabajo está destinada a implementar un lazo de control robusto que compense las vibraciones y balanceos de una cámara. Se realizarán comparativas para encontrar la solución más adecuada.

La estabilización de la imágen en vehículos no tripulados es un área en expansión. También lo es para el diseño de maquinaria autónoma para agricultura, de uso industrial o militar y en la industria cinematográfica.

1.2. Estado del arte

La estabilización de imagen es una técnica muy utilizada en muchas industrias. Su objetivo al final es conseguir una imagen nítida y clara. En la industria del cine se pueden encontrar una gran variedad de soportes estabilizadores, desde soportes manuales con contrapesos hasta sistemas grúa muy complejos.

Se puede decir que hay dos grandes grupos de estrategias de estabilización:

- Estabilización de la cámara
- Estabilización de la imagen

1.2.1. Estabilización inercial mecánica

En este caso, se estabiliza la cámara usando principios básicos de inercia. La cámara y unos contrapesos se balancean alrededor del centro de gravedad del sistema. La figura IV.1 muestra dos sistemas diferentes de estabilización inercial mecánico con contrapesos.

Figura IV.1 Sistema de estabilización inercial manual con contrapesos.

También existen sistemas de estabilización mecánica mediante giroscopios. Un giroscopio está formado por un disco que gira alrededor del eje de simetría perpendicular al disco. El principio de funcionamiento se basa en aumentar la inercia entorno al eje de giro del disco para conservar la orientación de dicho eje de rotación ante fuerzas externas que puedan desviarlo.

1.2.2. Estabilización inercial con sistema de control

Este principio básico de esta estabilización inercial se basa en el uso de un sistema de control electrónico. Mediante la utilización de una unidad de medición inercial se obtiene información de la orientación de la cámara en cada instante. Esta información actúa de entrada al sistema de control de modo que éste calcula las actuaciones necesarias para mantener la cámara en posición horizontal y mandar la señal adecuada a los motores.

Estos sistemas guardan una relación con un **sistema cardan** (***sistema gimbal***)**.** Un cardan es un soporte que puede girar alrededor de un eje. Un sistema gimbal enlaza varios cardanes juntos para proporcionar libertad de movimiento al objeto que se encuentra en su centro y responder sin restricciones a las fuerzas externas. En definitiva un cardan proporciona un número de grados de libertad al objeto que se desea estabilizar. Muy usado en equipos de cine y fotografía para permitir un movimiento equilibrado y suave de la cámara.

Alguna de las empresas que comercializan este tipo de soluciones son SiC Visuals[3] o DJI[4]. Sin embargo este tipo de solución está pensado para que sea transportado por una persona.

Figura IV.2 Sistema de estabilización manual (steadycam) con sistema de control.

También hay sistemas en el mercado diseñados para ser transportado en un vehículo aéreo no tripulado o en cualquier vehículo. Estos sistemas son conocidos como *gimbal* y pueden ser de 2 o 3 ejes. Suelen tener motores paso a paso o motores brushless. Se ilustra un ejemplo como el de la Figura IV.3.

[3] Página web del fabricante Sic Visuals: http://sicvisuals.com/
[4] Página web del fabricante DJI: http://www.dji.com/

Figura IV.3 Gimbal 3 axis con motores Brushless.

Existen muchos fabricantes de soportes de éste tipo, pero también existen empresas como BaseCam Electronics [12] que sólo se dedican a diseñar y vender el circuito electrónico y el software que desarrollan el control de la plataforma. La Figura IV.4 muestra el diseño 3D de la PCB controladora de BaseCam Electronics para gimbal de 3 ejes:

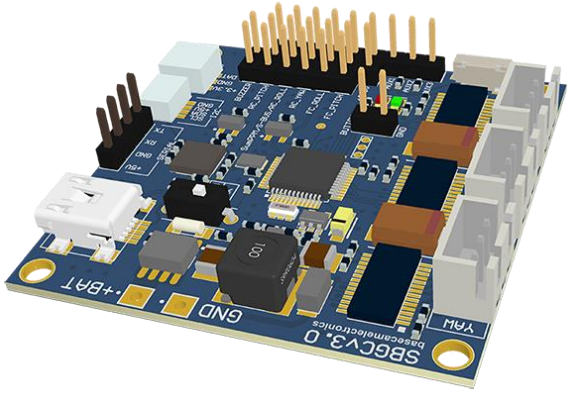

Figura IV.4 BaseCam SimpleBGC 32-bit

1.2.3. Estabilización de imagen

Otra forma de compensar los movimientos de la cámara es mediante el uso de sistemas de estabilización de imagen. La mayor parte de las cámaras de gama media y alta incorporan estos sistemas. Usando la posición de la cámara mediante una IMU se compensan los movimientos no deseados cambiando la posición de la lente. Varias marcas del mercado, entre ellas Canon[5], utilizan esta estrategia.

También existen empresas que se dedican a comercializar software capaz de procesar un video y estabilizarlo.

[5] Canon 2/3 Stabilized Video Lenses

1.3. Objetivos perseguidos en el trabajo

La finalidad del proyecto es poder sostener una cámara y estabilizar la misma ante movimientos bruscos o vibraciones que puedan perjudicar a la grabación obtenida.

De acuerdo con el estado del arte del apartado anterior en cuanto a estabilizaciones de cámaras se refiere, podemos encontrar una amplia variedad de diferentes sistemas en diferentes industrias como la industria del cine o el diseño de vehículos autotripulados.

Se ha descrito desde método más simple, la estabilización inercial mediante elementos mecánicos como contrapesos. Pasando por sistemas basados en giroscopios, hasta sistemas de control electrónico usados para la estabilización e incluso se opta por estabilizar la imagen en vez de la cámara.

El método a desarrollar será la estabilización inercial basada en el uso de un control electrónico en lazo cerrado, más concretamente, un controlador PID.

Éste control electrónico se programará en una placa controladora comercial para uso en sistemas de estabilización, del fabricante Ruso Basecam Electronics. Éste circuito impreso utiliza un microprocesador ATmega328P (Atmel megaAVR) [7]. Este procesador tiene claras ventajas ya que es el procesador usado por Arduino en una plataforma libre [14], basada en una placa con un microcontrolador y un entorno de desarrollo, los cuales proveen facilidad en el uso de la electrónica y su implementación en proyectos multidisciplinares.

También es necesaria una unidad de medición inercial (Inertial Measurement Unit – IMU) que provee al procesador información sobre la orientación de la cámara en cada instante. Ésta información actúa de entrada al sistema de control PID de modo que éste calcula las actuaciones necesarias basándose en errores presentes, en errores pasados y anticipándose a errores futuros para mandar la señal adecuada a los motores de la plataforma y así, mantener la cámara en posición horizontal.

1.4. Estructura de la memoria

En este apartado se describen brevemente los distintos apartados que componen la memoria:

- **Capítulo 1. Introducción:** Se trata del presente capítulo, introduce el trabajo realizado durante el proyecto y comenta las motivaciones del mismo.
- **Capítulo 2. Descripción del sistema hardware:** La principal tarea de este capítulo es dar una visión de la plataforma giroestabilizadora adquirida describiendo las partes y componentes que la componen. Se estudia el circuito integrado que forma parte de la plataforma. También se detallan los fundamentos de los sensores y los motores, así como sus principales características y aplicaciones.
- **Capítulo 3. Acondicionamiento de sensores:** En el capítulo que aquí se describe, se expone la metodología utilizada a la hora de tratar las señales de los acelerómetros y los giroscopios. También se explican los problemas derivados de la naturaleza de los sensores. Y por último se proponen distintas alternativas de filtrado para solventar dichos problemas.
- **Capítulo 4. Estrategia de control:** En este capítulo se plantea la estrategia de control en lazo cerrado cuya entrada serán las señales de los sensores y tendrá como salida el control de los motores de la plataforma. Se estudia un modelo teórico de la planta. Y por último se realiza un estudio teórico del controlador PID que se va a utilizar.
- **Capítulo 5. Desarrollo experimental:** Se explican todos los pasos seguidos para la implementación del sistema con las herramientas de desarrollo utilizadas. Se comentarán los resultados obtenidos en cada paso con ayuda de Matlab. Se modelará el control electrónico y se realizan las pruebas pertinentes para el correcto funcionamiento del sistema. También se evalúan los errores cometidos.
- **Capítulo 6. Conclusiones y trabajos futuros:** En este capítulo se exponen las conclusiones obtenidas en el desarrollo del proyecto, así como los futuros trabajos que pueden desarrollarse para continuar el desarrollo del producto.

En capítulos posteriores a la memoria se aportan los anexos, el presupuesto del proyecto, el contenido del CD y la bibliografía.

2. CÁPITULO

Descripción del sistema hardware

En este capítulo se detallarán las principales partes que componen el sistema que se va a implementar así como un repaso teórico de cada una y comparativas con productos existentes en el mercado. También se detallarán las características técnicas de los elementos escogidos para el desarrollo del estudio.

2.1. Descripción del sistema

En el presente trabajo se hace uso de una plataforma cardan comercial con chasis y cubiertas protectoras para los motores en aluminio. También tiene bolas de silicona para absorber vibraciones. Y dos motores brushless tipo 2212; uno para el eje roll y otro para el eje pitch. Podemos ver la plataforma en la Figura IV.5.

El circuito impreso del que hace uso el sistema adquirido, es una tarjeta controladora modelo BGC3.1 version MOSFET del fabicante ruso Basecam Electronics. Este fabricante provee las distintas versiones del firmware de todas sus controladoras en su página web junto con la GUI[6] para cada actualización de firmware [12].

[6] GUI del inglés *Grafic User Interface*

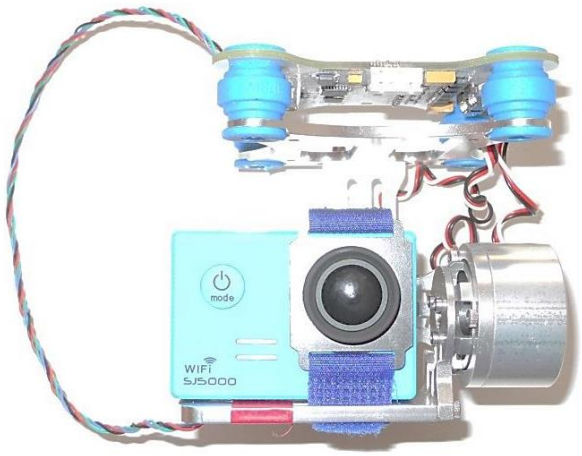

Figura IV.5 Plataforma gimbal usada en el proyecto.

El propósito del trabajo es programar de nuevo este firmware sin hacer uso del que el fabricante nos proporciona. Para ello se debe realizar "ingeniería inversa" y conocer la configuración de la PCB para poder programar de nuevo el firmware del sistema gimbal.

2.2. Controladora Basecam Electronics BGC 3.1 MOS

La tarjeta controladora elegida para el presente trabajo hace uso de un sensor IMU MPU6050 que se comunica por protocolo I2C[7] con el procesador y cuyo funcionamiento se explica más adelante en el trabajo.

En ella se puede observar que en el diseño de la misma han sido separadas la parte de control y la etapa de potencia, por seguridad y para evitar interferencias debidas a las conmutaciones de los MOSFET de potencia. Ambas partes alimentadas por un conector universal a una batería de polímero de lítio LiPo 2–3s de máximo 11,1 V.

La etapa de potencia consta de los siguientes componentes:

- El driver de potencia, formado por dos inversores trifásicos implementados a partir de componentes MOSFET AO4606 [9].
- Red de transistores y resistencias que acondicionan las salidas del microprocesador con los inversores que envían las señales a los motores.
- Conector para la batería LiPo.
- Conectores de salida para los motores.

[7] I2C Inter Integrated Circuit: https://es.wikipedia.org/wiki/I%C2%B2C

La parte de control de la PCB consta también de varios componentes:

- Conector microUSB, para cargar programas en el micro o visualizar datos mediante un protocolo serie. El driver que necesita para comunicarse con cualquier ordenador mediante un protocolo USB es un CP210 Bridge VCP del fabricante Silicon Labs [16].
- El microcontrolador utilizado en éste caso, un ATmega 328P del fabricante ATMEL [7] (el mismo microprocesador que utiliza Arduino [14]).
- Referencia de tensión para adecuar la tensión de la batería LiPo a la tensión necesaria de alimentación del micro y demás componentes que forman parte del hardware que conforma la electrónica de control.
- Entrada y circuito de adquisición de datos para las señales de la IMU conectada al micro a través del bus de comunicaciones serie I2C.
- También hay más elementos en la tarjeta como diodos, resistencias, un oscilador, y extensiones de pines que la tarjeta ofrece por si se desea conectar un sistema de radiofrecuencia para comunicarse con la plataforma gimbal.

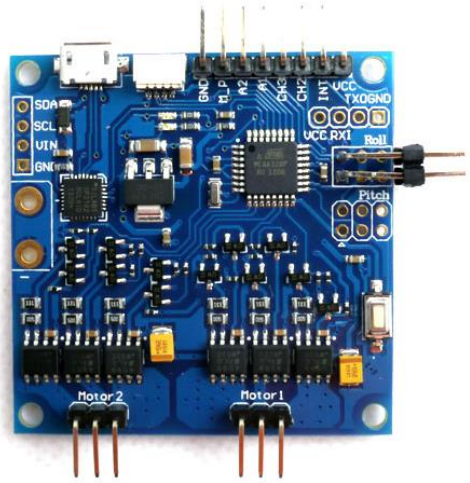

Figura IV.6 Tarjeta controladora BGC 3.1 tipo MOS de BaseCam Electronics [12].

2.3. Microcontrolador

El controlador del que hace uso la placa controladora de Basecam Electronics es un ATmega328P de ATMEL.

El ATmega328P es un microcontrolador CMOS de baja potencia de 8 bits basado en un AVR mejorado con arquitectura RISC. Si se observan las características del procesador en la primera página de su datasheet [7], se puede comprobar que el ATmega328P cuenta con todos los componentes necesarios para poder implementar el presente diseño: protocolo de comunicaciones I2C para leer las señales de los sensores MPU6050 [8], comunicación serie para conectar un USB, tres timers configurables a

diferentes frecuencias, seis salidas PWM para los dos motores [19], y entradas y salidas de propósito general.

El núcleo AVR combina un amplio conjunto de instrucciones con 32 registros de trabajo de propósito general. Estos 32 registros están conectados directamente a la unidad aritmético-lógica (ALU), lo que permite poder acceder a dos registros independientes mediante una sola instrucción ejecutada en el mismo ciclo de reloj. La nueva arquitectura resulta más eficiente en tiempo de cómputo y se consiguen rendimientos hasta diez veces más rápido que otros microcontroladores CISC convencionales.

Con la capacidad de ejecución de varias instrucciones en un solo ciclo de reloj, el ATmega328P logra rendimientos que se acercan a 1 MIPS por MHz, ésto va a permitir un diseño optimizando el consumo de energía reduciendo la velocidad de procesamiento o viceversa.

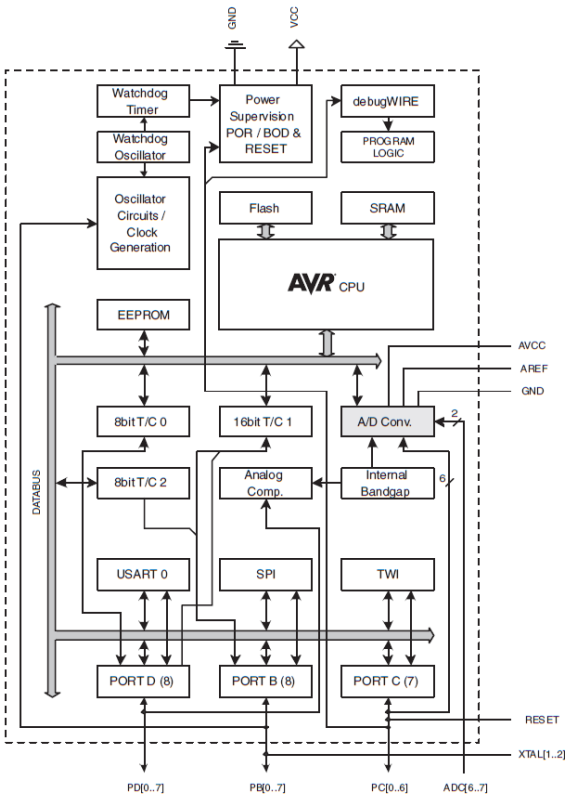

Figura IV.7 Diagrama de bloques del procesador ATMEL AVR de 8 bits [7]

Hoy el ATmega328 se usa comúnmente en múltiples proyectos y sistemas autónomos donde un microcontrolador simple, de bajo consumo, bajo coste es requerido. Tal vez la implementación más común de este chip es en la popular plataforma Arduino, en sus modelos Uno y Nano.

La programación del microcontrolador presente en el dispositivo se hace mediante

un lenguaje propio, *Arduino*, sencillo e intuitivo que favorece un rápido aprendizaje. Este está basado en el lenguaje de alto nivel Processing y en C, por lo que soporta muchas de sus funciones. Además se hace uso del entorno de programación de Arduino [14]. Un software de código abierto con el que poder compilar programas y subir programas al procesador.

2.4. Inertial Measurement Unit (IMU)

Una IMU o unidad de medición inercial es una solución para aplicaciones que requieran el uso conjunto de un acelerómetro, un giroscopio y a veces, un magnetómetro. Todos los sensores suelen ser tipo MEMS, por lo que el tamaño de estos sensores es ínfimo y pueden encontrarse todos implementados en un mismo encapsulado. Su uso está ampliamente difundido en aviones, vehículos aéreos no tripulados (UAV), sistemas de navegación, sistemas de estabilización de imagen y ampliamente integrado en nuevas tecnologías como reconocimiento de gestos, servicios de localización y controles remotos.

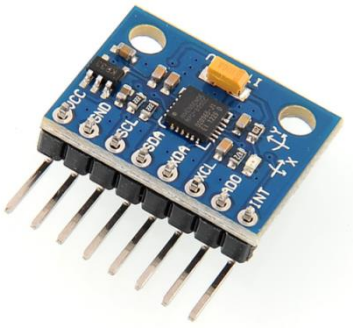

Figura IV.8 IMU 6 DOF comercial modelo MPU6050 [8].

La Unidad de Medición Inercial es un dispositivo electrónico que mide y registra información obtenida acerca de la velocidad, la orientación y los efectos de las fuerzas gravitatorias sobre el dispositivo. Todo gracias a la combinación del acelerómetro y el giroscopio para obtener sistemas de 6 grados de libertad, que pueden ser ampliados hasta 9 incluyendo magnetómetros.

En el presente proyecto se empleará una unidad de medición inercial MPU6050 del fabricante InvenSense [8] de 6 grados de libertad. Dicha IMU incluye acelerómetros y giroscopios de tipo MEMS dispuestos en un sistema de coordenadas cartesiano.

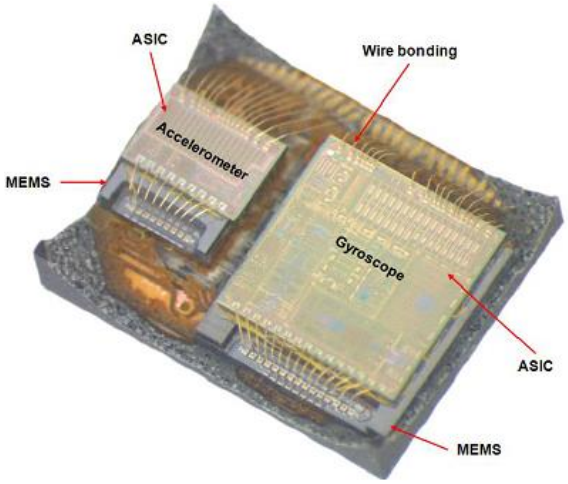

Figura IV.9 Vista interior de un sensor IMU.

2.4.1. Acelerómetros

Son sensores que miden aceleraciones, y pueden utilizarse para la medición de ángulos a partir de las señales que genera en ellos la gravedad. En definitiva, los acelerómetros miden las proyecciónes de la fuerza de la gravedad (conocidas como fuerzas G) respecto a cada eje.

Los acelerómetros micromecánicos (MEMS) son cada vez más empleados en multitud de ámbitos. La principal ventaja de estos dispositivos MEMS es que pueden ser creados mediante técnicas de fabricación microelectrónica en un chip de silicio, a la vez que toda la electrónica necesaria para el sistema de acondicionamiento, adquisición y comunicación en un tamaño muy reducido y a un precio muy competitivo.

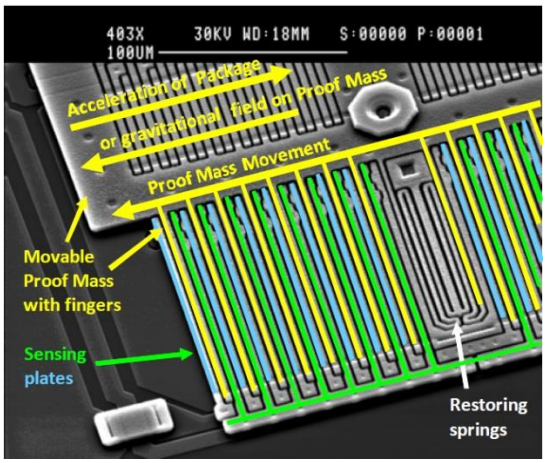

Figura IV.10 Imagen tomada con microscopio electrónico de un Acelerómetro MEMS [2].

La estructura de estos dispositivos se basa por lo general, en placas o filamentos capacitivos internos, algunos móviles y otros fijos al sustrato. Se pueden apreciar la masa fija y los filamentos móviles de sensado en la imagen de la Figura IV.10.

Cada filamento actúa como una placa de un condensador variable. La aceleración o desaceleración en el eje del sensor ejerce una fuerza a la masa central. Al moverse libremente, la masa desplaza las minúsculas placas del microcondensador, provocando un cambio de capacidad, el cual es detectado y procesado para obtener un voltaje de salida. Dicho voltaje resulta ser directamente proporcional a la aceleración.

El inconveniente de los acelerómetros MEMS radica en su gran inestabilidad y sensibilidad al ruido.

2.4.2. Giroscopios

Un giroscopio es un sensor que mide velocidades angulares, y dependiendo de su disposición, puede medir velocidades sobre cualquier eje de rotación.

El modelo clásico de un giroscopio mecánico consiste en un disco con una gran inercia que rota en torno a un eje. Debido a la ley de conservación del momento angular, este móvil rotativo mantiene su orientación respecto su sistema de referencia (*principio de Coriolis*). Estos dispositivos han sido ampliamente utilizados en la tecnología aeronáutica, donde es vital la estimación de la orientación de las aeronaves respecto del suelo.

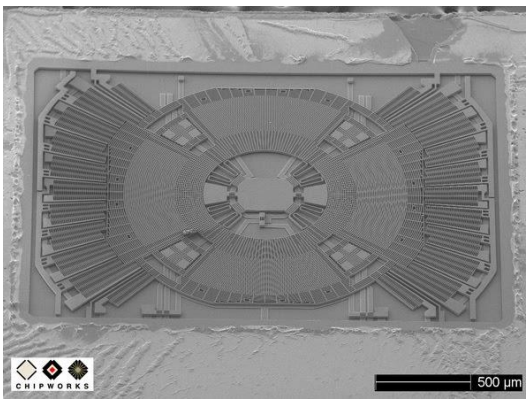

Figura IV.11 Imagen ampliada de un giroscopio MEMS.

Los sensores giroscópicos incluidos en la IMU son también MEMS, su funcionamiento está basado en una pequeña masa de resonancia que se desplaza al variar la velocidad angular, dicha variación es medida mediante una interfaz sensora capacitiva y posteriormente, se conecta a un amplificador diferencial de carga que convierte las variaciones capacitivas en variaciones de tensión. Las fuerzas de Coriolis son proporcionales a la velocidad angular que es la magnitud de interés. Gracias a este diseño, ha sido posible reducir de forma notable el tamaño de estos dispositivos y poder así incluirlos en un circuito integrado.

Es interesante mencionar que, en general, los giróscopos son sensores poco sensibles a giros lentos y de los que se obtiene una señal de velocidad angular limpia.

Es posible obtener mediciones angulares con respecto a un eje de rotación usando un giróscopo. Para ello es necesario realizar una integración discreta de la velocidad angular. Sin embargo, esto provocará una deriva que hará que el valor de salida en reposo se desplace, perdiendo la referencia y obteniendo una lectura errónea del ángulo.

2.4.3. InvenSense MPU-6050

El MPU6050 de InvenSense [8] es un dispositivo MotionTracking, el primer circuito integrado de seis ejes de libertad que combina un giroscopio tipo MEMS de tres ejes, un acelerómetro tipo MEMS de tres ejes, y un Procesador Digital de movimiento™ (DMP) motor acelerador de hardware, todo en un chip ultra pequeño 4x4x0.9 mm.

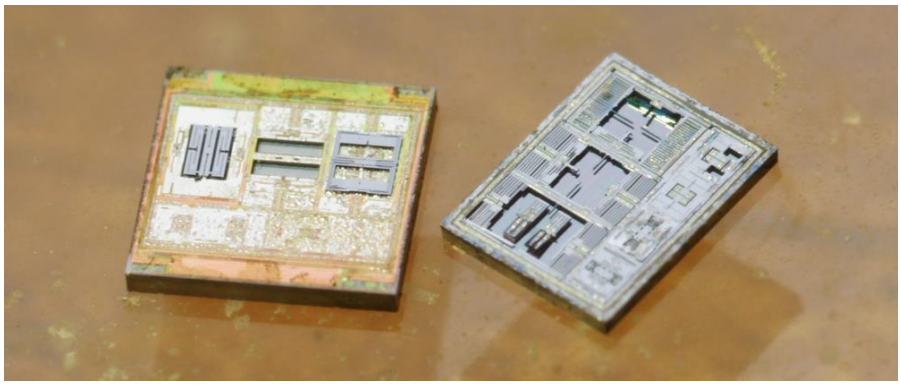

Figura IV.12 Imagen ampliada de la sección de un MPU6050[8]

Mediante el bus I2C del sensor, se pueden introducir otros tres ejes de un magnetómetro externo para proporcionar una salida completa de nueve ejes de libertad.

El MPU6050 cuenta con tres ADCs de 16 bits para la digitalización de las salidas giroscopio y tres ADCs de 16 bits para la digitalización de las salidas del acelerómetro. Para dotar de un mayor sensado de precisión para movimientos rápidos y lentos, se puede programar la sensibilidad del los giroscopios con rangos de ± 250, ± 500, ± 1.000, y ± 2,000 ° / seg (DPS) y la sensibilidad de los acelerómetros con rangos de ± 2 g, ± 4g, ± 8g, y ± 16 g.

Cuenta con un búfer FIFO de 1024 Bytes que ayuda a mejorar el consumo de energía del chip permitiendo que el procesador del sistema pueda leer los datos del sensor

[8] Fuente de la imagen: http://zeptobars.ru/en/read/Invensense-MPU6050-6d-MEMS-IMU-gyroscope-accelerometer

en ráfagas y entrar en un modo de bajo consumo.

La comunicación con todos los registros del dispositivo se realiza utilizando el protocolo I2C a 400kHz.

Como características adicionales, se incluyen un sensor de temperatura integrado y un oscilador en el chip con una tolerancia de variación de ± 1% sobre el rango de temperatura de funcionamiento.

InvenSense ha impulsado el tamaño del c.i. MPU6050 a una huella revolucionaria de 4x4x0.9mm (QFN), mientras que proporciona un mayor rendimiento y menor ruido. El integrado cuenta con una alta tolerancia a los golpes de 10.000 g, y tiene filtros de paso bajo programables para los giroscopios, acelerómetros, y un sensor de temperatura en el chip.

Para una mayor flexibilidad a la hora de alimentar el chip, el MPU6050 opera en un rango de voltaje de 2.375V-3.46V (VDD).

2.5. Motores DC Brushless (BLDC)

Michael Faraday (1791−1867), fue el descubridor del principio del motor eléctrico, descubrió la inducción, o lo que es lo mismo, la generación de una corriente eléctrica en un conductor en movimiento en el interior de un campo magnético físico. Para calcular la inducción magnética se tiene que aplicar la Ley de Faraday-Lenz:

$$\varepsilon = -N \cdot \frac{d\emptyset}{dt}$$

Donde ε es la tensión inducida en voltios, N es el número de espiras, $\emptyset$ es el flujo del campo magnético en Webbers, y la expresión $\frac{d\emptyset}{dt}$, es la variación del flujo magnético con el tiempo. En definitiva, esta ecuación relaciona la fuerza electromotriz (FEM) inducida sobre un conjunto de espiras con la variación del flujo magnético.

Los motores eléctricos son máquinas eléctricas rotatorias que transforman la energía eléctrica en energía mecánica basándose en el principio de la Ley de Faraday-Lenz. Debido a sus múltiples ventajas, entre las que cabe citar su economía, limpieza, comodidad y seguridad de funcionamiento, el motor eléctrico ha reemplazado en gran parte a otras fuentes de energía, tanto en la industria como en el transporte, las minas, el comercio, o el hogar.

En cuanto a los tipos de motores eléctricos genéricamente se distinguen motores monofásicos, que contienen un juego simple de bobinas en el estator, y polifásicos, que mantienen dos, tres o más conjuntos de bobinas dispuestas en círculo.

Según la naturaleza de la corriente eléctrica transformada, los motores eléctricos se clasifican en motores de corriente continua y motores de corriente alterna, que a su vez, se agrupan según su sistema de funcionamiento, en motores de inducción y motores sincrónicos.

El motor que se utiliza en el presente trabajo es un tipo de motor de corriente continua sin escobillas o motor brushless.

Los motores de corriente continua sin escobillas (BLDC) o motores DC brushless representan el último desarrollo de la historia en cuanto a motores eléctricos DC se refiere. Actualmente, los motores BLDC se emplean en sectores industriales tales como: Automóvil, Aeroespacial, Consumo, Médico, equipos de automatización e instrumentación y robótica.

2.5.1. Comparativa con otros motores eléctricos

Los motores brushless tienen como característica no emplear escobillas para la conmutación en la transferencia de energía; la conmutación se realiza electrónicamente. Esta propiedad elimina el gran problema que poseen los motores eléctricos convencionales con escobillas (*brushed*), los cuales tienen rozamientos, disminuyen el rendimiento, desprenden calor, son ruidosos y requieren una sustitución periódica de las escobillas.

Pese al mayor coste y a la complejidad del control de los motores BLDC frente a los motores DC con escobillas y frente a los motores de inducción, tienen algunas ventajas que hacen que sean idóneos para un amplio rango de aplicaciones:

- Mayor rendimiento debido a no existir pérdidas en el rotor (característica que hace realmente interesante este tipo de motores para aplicaciones alimentadas con baterías en las que reducir las pérdidas al máximo es algo crítico para aumentar la autonomía).
- Mayor eficiencia y por tanto mayor vida útil, ya que al carecer de escobillas para realizar la conmutación, estos motores no requieren apenas mantenimiento.
- Excelente relación par/peso, permitiendo realizar diseños más ligeros y pequeños con muy buenas características de funcionamiento.
- Mayor respuesta dinámica.
- No hay deslizamiento.
- Menor ruido.
- Mejor disipación de calor.

2.5.2. Principio de funcionamiento

Se puede decir que los motores BLDC son un tipo de motor síncrono. Pues el campo magnético generado por el estator y el campo magnético generado por el rotor giran a la misma frecuencia. Esto va a provocar que un motor brushless tenga la característica de no presentar el común "deslizamiento" que se produce en otros tipos de motores. Los motores BLDC se dividen en motores de una sola fase, motores de dos fases y motores de tres fases. Los motores de tres fases son los más populares y utilizados.

El ***estator*** de un motor BLDC consiste en un conjunto de láminas de acero apiladas con bobinados colocados en sus ranuras de forma axial denominados fases. Cada una de estas fases está construida por numerosas espiras interconectadas. Una o más bobinas colocadas en las ranuras se interconectan para formar una fase. Cada una de estas bobinas se distribuye en la periferia del estator para formar un número par de polos.

La mayoría de los motores BLDC tienen tres devanados o fases en el estator pudiendo estar estos conectados en triángulo o en estrella, aunque la configuración más común es la conexión en estrella. Las tensiones inducidas por el estator son de forma trapezoidal y cada fase inducirá una señal de tensión desfasada 120° respecto la siguiente.

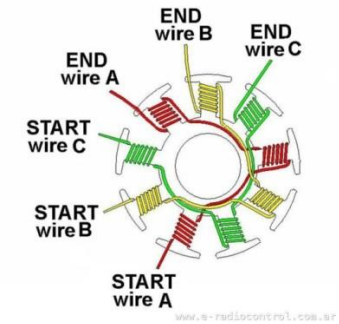

Figura IV.13 Bobinado del estator de un motor brushless[9]

El ***rotor*** es de imán permanente y puede variar desde dos hasta ocho pares de polos alternativos de Norte (N) y Sur (S). En función de la densidad de campo magnético requerido en el rotor, se escoge el material magnético adecuado para su fabricación. Los imanes permanentes normalmente están compuestos a base de ferrita.

[9] Fuente de la imagen: http://www.kawaner6s.es/forum/index.php/topic/35858-reparar-stator/

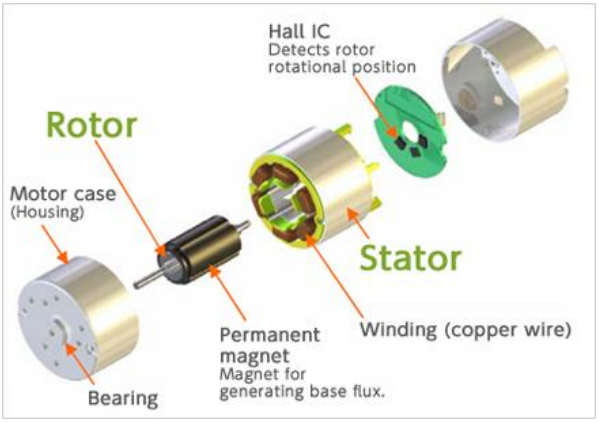

Figura IV.14 Topología de un motor brushless[10].

2.5.3. Secuencias de alimentación

Para conseguir el giro de un motor brushles trifásico, se necesita seguir unas secuencias de alimentación que se encargan de mandar los estímulos a una o dos fases del motor en cada instante (véase Figura IV.15). Para que el motor funcione, y por lo tanto genere par se tiene que dar la condición de que exista un ángulo entre el campo magnético generado por los devanados alimentados y el campo magnético propio de los imanes del rotor.

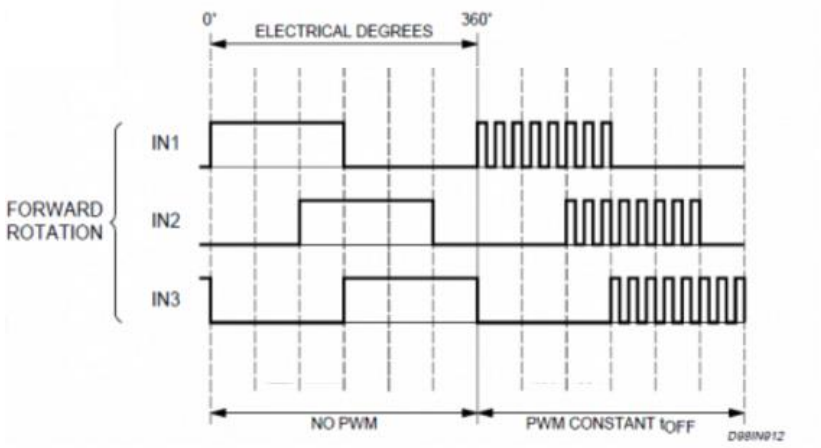

Figura IV.15 Secuencias de alimentación de las fases de un motor brushless trifásico [1].

Para obtener el máximo par y un funcionamiento perfecto el objetivo es mantener siempre este ángulo lo más cercano a 90° dando lugar al par máximo para unas condiciones de corriente dadas. De esta manera la secuencia en cada momento ha de ir adecuándose al giro del rotor de forma que se mantenga lo máximo posible la perpendicularidad entre ambos campos magnéticos.

[10] Fuente de la imagen: http://www.nidec.com/en-NA/technology/capability/brushless/

Cuando el rotor gira, se va a generar una fuerza contraelectromotriz (Back FEM) que se induce en el estator, opuesta a la tensión suministrada.

2.5.4. Motores BLDC del sistema gimbal

El sistema gimbal adquirido para el presente proyecto incluye dos motores tipo 2212 de tres fases conectadas en estrella, los cuales poseen las siguientes características:

BLDC A 2212 / 13 T Technical features	
Nº of Cells	2 -3 LiPo / 6 -10 NiCd/NiMH
Kv	1000 rpm/V
Max Efficiency	80%
Max Efficiency current	4-10A (>75%)
Nº Load Current	0.5A , 10V
Internal Resistance	90mΩ
Max Current	12A/60s
Max Watts	150W
Weight	52.7g /1.86oz
Shaft Diameter	3.17mm
Poles	14
Motor dimensions	27.5x30 (mm)
Motor bolt pattern (back of motor)	16mm x 19mm (hole to hole)
Recommended Model Weight	300-800g / 10.5 – 28.2oz
Minimum Recommended ESC	18A

Tabla IV-1: Características de los motores BLDC.

Como podemos observar en la Tabla IV-1, los motores que componen nuestro sistema tienen siete pares de polos (o catorce polos) y un factor Kv de mil revoluciones por cada voltio. O lo que es lo mismo, estos motores son capaces de girar a mil revoluciones por minuto por cada voltio que se incrementa su tensión de entrada. Cuanto mayor sea el factor Kv, mayor es la velocidad, pero el par máximo del motor se ve reducido. Es un parámetro importante a la hora de elegir cualquier motor brushless que se pueda encontrar en el mercado ya que condiciona totalmente el diseño.

Figura IV.16 Motor brushless comercial.

2.6. Etapa de potencia

Un motor DC brushless necesita de una etapa de potencia para suministrar tensión y corriente a las bobinas del motor. Para ello, al motor BLDC se le conecta un convertidor DC-AC trifásico, el cual consiste en tres ramas inversores, con dos interruptores de potencia por cada rama. Cada uno de estos interruptores se activa en función de una señal PWM[11] distinta para cada uno.

Las señales PWM que llegan a los interruptores de potencia, están alimentando y apagando las fases según las secuencias de la Figura IV-15. Como regla general, la frecuencia de PWM es por lo menos 10 veces la frecuencia máxima del motor.

2.6.1. Topología del inversor

Los convertidores DC-AC también son conocidos como inversores. Pueden tener salida variable en voltaje y frecuencia. Es típico el uso de estos inversores como drivers de potencia para motores.

La topología del inversor queda definida por el modo de conexión de los dispositivos de conmutación que conforman el puente. Cada rama del puente consta de dos dispositivos de conmutación. Los inversores de una rama (medio puente, dos dispositivos de conmutación) o dos ramas (puente completo o puente en H, cuatro dispositivos de conmutación) son empleados para controlar motores monofásicos y bifásicos; mientras que los de tres ramas son usadas para controlar motores trifásicos (seis dispositivos de conmutación) siendo éste el inversor de mayor uso.

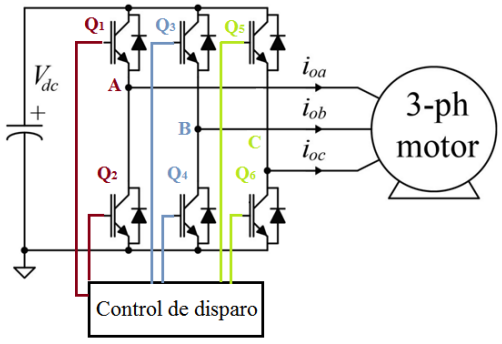

Figura IV.17 Esquema de un inversor trifásico cualquiera conectado a motor de tres fases.

Los componentes electrónicos utilizados en inversores como conmutadores de

[11] PWM del inglés *Pulse Width Modulation.*

potencia pueden ser transistores (BJT o MOSFET), IGBT, GTO o tiristores de conmutación forzada. Todos ellos necesitan de un sistema de control que se encargará de que éstos elementos conmuten siguiendo la secuencia correcta.

Las tres señales que activan los conmutadores de cada rama de potencia están desfasadas 120° entre sí, y dichas señales son las que alimentan el estator y provocarán el ciclo de giro del motor brushless.

Es importante considerar que los interruptores precisan de un tiempo mínimo, tanto en la apertura para anular la corriente, como en el cierre para su establecimiento. Este tiempo de espera se denomina generalmente *tiempo muerto* y debe de ser respetado y tenido en cuenta durante el diseño por motivos de seguridad. Protege de cortocircuitos a la fuente de alimentación, y evita que dos señales de control estén activas al mismo tiempo. Para mayor seguridad, la corriente de descarga producida en las conmutaciones circulará por unos diodos dispuestos en paralelo con cada interruptor.

2.6.2. Etapa de potencia de la controladora BGC 3.1

Como ya se ha comentado en el apartado 2.2 de este capítulo, la tarjeta controladora BGC 3.1 de Basecam Electronics incluye una etapa de potencia para el manejo de los motores.

Esta etapa de potencia está basada en dos convertidores DC/AC o dos inversores trifásicos formados cada uno por seis transistores MOSFET, dos por cada rama del inversor. Cada pareja de conmutadores se encuentran dentro del circuito integrado AO4606 del fabricante Alpha&Omega Semiconductor[12]. Podemos ver el datasheet del componente en la referencia [9] de la bibliografía.

Estos MOSFETS cambian su estado de corte a saturación y viceversa debido a la secuencia de control basada en señales PWM programadas en el procesador que dejan pasar la tensión de entrada del inversor (señal de la batería[13]) a los motores brushless. Todos los sistemas de seguridad y protección de la tarjeta controladora están implementados mediante una red de transistores y demás componentes que se encargan del control de los tiempos muertos y las protecciones ante sobretensiones o cortocircuitos.

[12] Web del fabricante Alpha&Omega Semiconductor: http://www.aosmd.com/
[13] Batería utilizada LiPo 3s (Polímero de lítio de 3 celdas) entrega 11,1V DC

3. CÁPITULO

Teoría de acondicionamiento de sensores

En la realidad no podemos hacer uso de señales provenientes de sensores sin acondicionarlos previamente, ya sea para transformar estas medidas a la magnitud deseada, o para filtrar el posible ruido y solventar posibles errores provocados por la naturaleza del sensor o factores externos al sistema.

Para este caso, es necesario acondicionar los valores que nos proporciona cada sensor en bruto antes de realizar un filtrado para optimizar al máximo la medida obtenida.

3.1. Teoría del acondicionamiento de acelerómetros

El acelerómetro MEMS de la IMU mide aceleraciones, realmente fuerzas por unidad de masa:

$$\vec{F} = m\,\vec{a} \quad \rightarrow \quad \vec{a} = \frac{\vec{F}}{m}$$

Dichas aceleraciones realmente representan las proyecciónes de la fuerza de la gravedad (conocidas como fuerzas G) respecto a cada eje. La unidad de medida utilizada para medir aceleraciones tipicamente es m/s^2, pero por la relacción que tiene el acelerómetro con la gravedad, la unidad de medida usada es $g = 9{,}81\ m/s^2$.

La aceleración puede expresarse en 3 ejes basados en un sistema cartesiano: X, Y y

Z, las tres dimensiones del espacio.

Normalmente se comparan éstas variaciones de aceleración con la aceleración de la gravedad terrestre. Gracias a esto, podemos usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto cada eje.

Si la IMU está perfectamente alineada con el suelo. Entonces, como se puede ver en la Figura IV.18, el eje Z marcará $1g$ ó 9,81 m/s^2, y los otros dos ejes marcarán 0. Ahora supongamos que giramos la IMU 90 grados. Ahora es el eje X el que está perpendicular al suelo, por lo tanto marcará la aceleración de la gravedad.

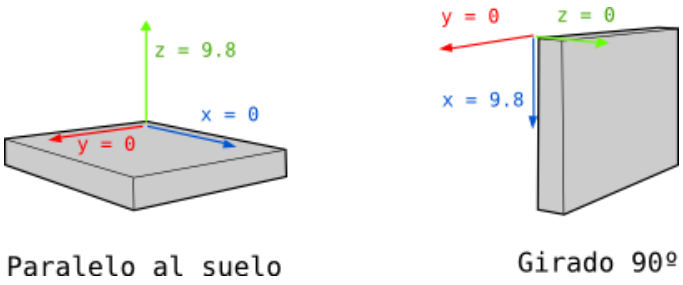

Figura IV.18 Explicación gráfica del funcionamiento de un acelerómetro [3].

Para obtener la posición angular con el acelerómetro, sabemos que el vector de la gravedad es siempre visible en el acelerómetro. Si conocemos el módulo de la gravedad, y sabemos que medida tienen las aceleraciones de los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la IMU (Figura IV.19)

Esto se puede hacer fácilmente mediante el uso de una función trigonométrica arco-tangente:

$$\theta_X = arctg\left(\frac{A_x \cdot g}{\sqrt{(A_y \cdot g)^2 + (A_z \cdot g)^2}}\right)$$

$$\theta_Y = arctg\left(\frac{A_y \cdot g}{\sqrt{(A_x \cdot g)^2 + (A_z \cdot g)^2}}\right)$$

Donde A_x, A_y, A_z son las componentes vectoriales de la aceleración, que multiplicadas por la constante de la aceleración de la gravedad (g), representan las proyecciones de la fuerza de gravedad o fuerzas G.

En el apartado de desarrollo experimental se describe el algoritmo que permitirá calcular los ángulos a partir de las aceleraciones medidas con los acelerómetros.

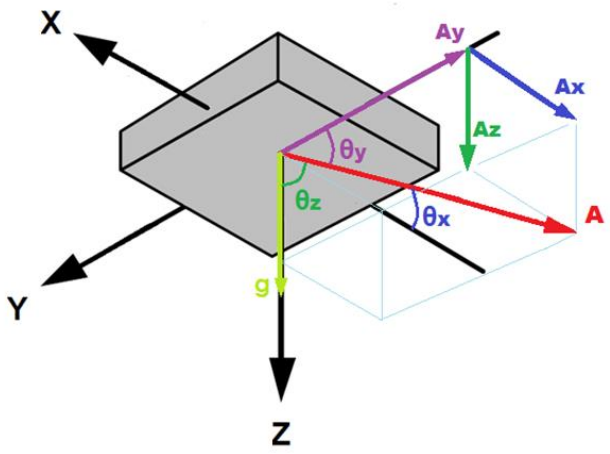

Figura IV.19 Diagrama vectorial de las aceleraciones en un acelerómetro.

3.2. Teoría del acondicionamiento de giroscopios

Un giroscopio mide la velocidad angular respecto a cada uno de los tres ejes cartesianos. Por lo tanto cuando el sistema se encuentre en estado estacionario, la velocidad será nula en cualquiera de los tres ejes. Las unidades de medida para la velocidad angular que utilizan de los giroscopios del MPU6050 son *grados / segundo* (°/s).

Si queremos saber el ángulo de orientación de la IMU, también podemos obtenerlo a partir de las medidas del giroscopio. Para calcular el ángulo, hay que integrar la velocidad angular respecto al tiempo.

$$\int \omega_x(t)\,dt = \Delta\theta_x(t)$$

$$\int \omega_y(t)\,dt = \Delta\theta_y(t)$$

$$\int \omega_z(t)\,dt = \Delta\theta_z(t)$$

Este cálculo se realizará a partir de los valores obtenidos con los sensores discretizados mediante los ADCs con los que cuenta el MPU6050. En tiempo discreto, dicha integración queda así:

$$\theta_{x_k} = \theta_{x_{k-1}} + \omega_x \cdot \Delta t$$

$$\theta_{y_k} = \theta_{y_{k-1}} + \omega_y \cdot \Delta t$$

$$\theta_{z_k} = \theta_{z_{k-1}} + \omega_z \cdot \Delta t$$

Las medidas en bruto o en RAW de los giróscopos tienen un offset o linea base que es importante calcular y restar a la medida para obtener la velocidad angular sin dicho offset.

La implementación de los algoritmos necesarios para la adquisición de las medidas RAW, su integración discreta y el cálculo del offset se explicarán en el desarrollo experimental del capítulo 5.

3.3. Problemas derivados de la naturaleza de los sensores

Ambos transductores están condicionados por las variables del entorno, ruido y errores, que van a contribuir en la dificultad de la lectura correcta de los mismos.

El primer problema que nos vamos a encontrar, van a ser las interferencias EMIs, RFIs que pueden afectar a cualquier dispositivo electrónico que se encuentre en el entorno apropiado para que se den interferencias de este tipo. Esto se puede solventar con un buen diseño de la plataforma estabilizadora. En la plataforma escogida en el presente trabajo, se suministran los cables de conexión a los sensores con un rizado que favorecerá la disminución del ruido e interferencias en las medidas tomadas.

Las lecturas de los giroscopios son precisas y poco susceptibles a fuerzas externas en sus medidas. Pero la integración discreta de la velocidad es muy lenta y se produce un error que se acumula en el tiempo. Esto quiere decir que la medida tiene tendencia a la deriva, o en inglés, *drift*. Por lo que nuestra medida no volverá a la referencia cero aunque el sistema esté en la posición inicial de reposo.

Filtrar estas señales será fundamental para reducir el impacto del ruido y los errores en nuestras mediciones y cálculos.

Hay distintos filtros aplicados mediante algoritmos basándose su mayoría en modelos de ecuaciones de estado. Uno de estos sistemas de filtrado más conocidos es el llamado Filtro de Kalman. Se utiliza como herramienta de procesado de datos de orientación, navegación y control de vehículos, en particular aeronaves y naves espaciales. Además, el filtro de Kalman es un concepto ampliamente aplicado en el análisis de series matemáticas utilizado en los campos de procesamiento de señales y el de la econometría. Es también un recurso principal en el campo de planificación de movimientos de robots y control, que también se puede usar en la optimización de la trayectoria. Éste filtro es

considerado uno de los mayores hallazgos del siglo XX.

Otra técnica de filtrado para acelerómetros y giroscopios es el Filtro Complementario, una versión simplificada del filtro de Kalman que utiliza menos recursos del procesador y que nos proporciona medidas con bastante exactitud.

Para desarrollar el control de nuestro sistema se tienen en cuenta dos estrategias claramente diferenciadas. Por un lado se puede optar por el *control de posición* basado en las medidas angulares que obtenemos del acelerómetro tratando de atenuar el ruido en la lectura del mismo con las velocidades del giroscopio integradas en el tiempo. Y por otro lado podemos optar por un *control de velocidad*, basándose en las velocidades medidas por giroscopio, que sin necesidad de integrarlas serán más precisas. Esta distinción será crucial a la hora de definir las diferentes formas de filtrar las señales que se van a explicar en los apartados sucesivos.

En el presente trabajo se van a comparar las medidas del acelerómetro y del giroscopio y se van a implementar técnicas de filtrado que serán posteriormente comparadas para así obtener la mejor medida para nuestra aplicación final.

3.4. Filtro Complementario

Dependiendo de la estrategia de control elegida, se puede implementar un filtro complementario de diferentes formas para una IMU con las características anteriormente mencionadas:

- *Control de posición.* El filtro complementario se encarga de fusionar las medidas del acelerómetro y del giroscopio teniendo un ángulo final corregido con la integración discreta de la velocidad angular.
- *Control de velocidad*. En este caso, el filtrado sirve para mejorar la medida de velocidad angular del giroscopio conforme a los valores filtrados en los estados anteriores.

3.4.1. Filtro Complementario para control en posición

El filtro en general se puede reducir a utilizar el acelerómetro para medir el ángulo y el giroscopio para corregir la medida del acelerómetro mediante la integración discreta de la velocidad angular.

El acelerómetro es muy rápido y sensible, por lo que sus datos son fiables sólo a largo plazo, y se ha de utilizar un ***filtro paso bajo*** que dejará pasar sólo a valores que cambien a largo plazo que son estables en el tiempo, filtrando las fluctuaciones a corto plazo.

El filtro de paso bajo elimina los ruidos y las aceleraciones lineales. Pero produce un retardo en la percepción del ángulo, por lo que puede provocar problemas en la estabilización.

Sin embargo, podemos combinar los datos recogidos por el giroscopio para eliminar el retardo. Los datos del giroscopio son fiables sólo a corto plazo, ya que su medida tiene tendencia a la deriva cuando se le aplica una integración discreta. Al contrario que el filtro de paso bajo usado en el acelerómetro, para el giroscopio se utiliza un ***filtro de paso alto*** que sólo permite el paso a las señales de corta duración (o a corto plazo). El diagrama de bloques del filtro podría corresponderse con la Figura IV.20.

No podemos utilizar los datos del giroscopio directamente (integrados en el tiempo para pasar de velocidad angular a ángulo) dado que este tiene una deriva temporal que los inutiliza; sin embargo, a corto plazo sirven para eliminar el retardo producido por el filtro de paso bajo aplicado al acelerómetro.

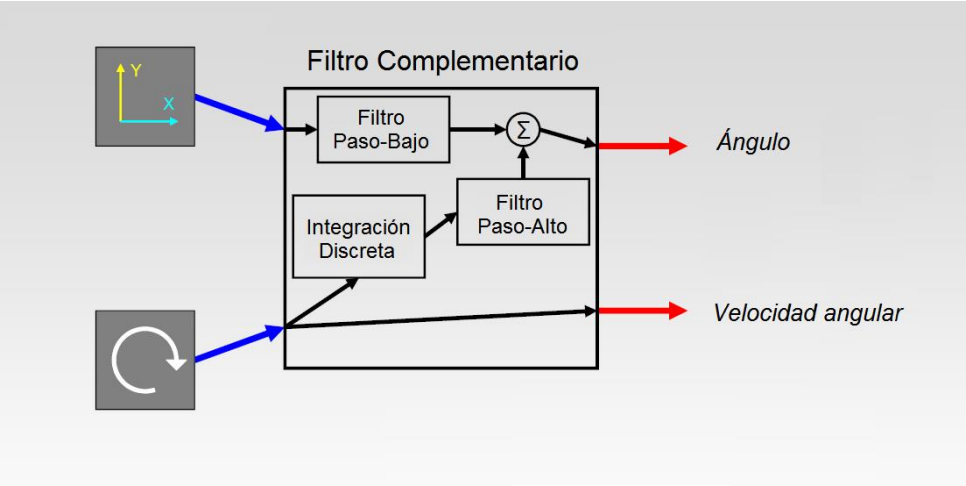

Figura IV.20 Diagrama de bloques del filtro complementario [4].

La fórmula resultante de combinar/complementar los dos filtros es:

$$Ángulo_K = \alpha \cdot \left(Ángulo_{K-1} + VelGyro \cdot \Delta t\right) + (1 - \alpha) \cdot ÁnguloAcc$$

Dónde α es la ganancia del filtro de paso alto y $(1 - \alpha)$ es la ganancia del filtro de paso bajo; $Ángulo_K$ y $Ángulo_{K-1}$ es el resultado del ángulo final en la muestra actual y la anterior; $VelGyro \cdot \Delta t$ es el desplazamiento relativo en base a la velocidad angular; y

$\acute{A}nguloAcc$ es el ángulo calculado con la salida del acelerómetro mediante la arco-tangente. Esta fórmula es la misma para los ejes X e Y.

El filtro se llama complementario porque combina sus dos partes de manera que siempre suman uno, consiguiendo una estimación precisa y lineal en la salida.

A partir de aquí podemos deducir la constante de tiempo para los filtros paso-bajo y paso-alto:

$$\tau = \frac{\alpha \cdot dt}{(1 - \alpha)}$$

Por lo que si sabemos la frecuencia de muestreo (dt^{-1}) y la constante de tiempo del filtro (τ), podemos escoger la ganancia del filtro (α).

Para periodos de tiempo más cortos de la constante de tiempo (τ), la integración de la velocidad del giroscopio tiene prioridad y las aceleraciones ruidosas se filtran. Para períodos más largos de la constante de tiempo (τ), la medida del acelerómetro es prioritaria pues el giroscopio está marcado por una deriva. Resumiendo, el filtro define el límite de confianza entre el giroscopio y el acelerómetro.

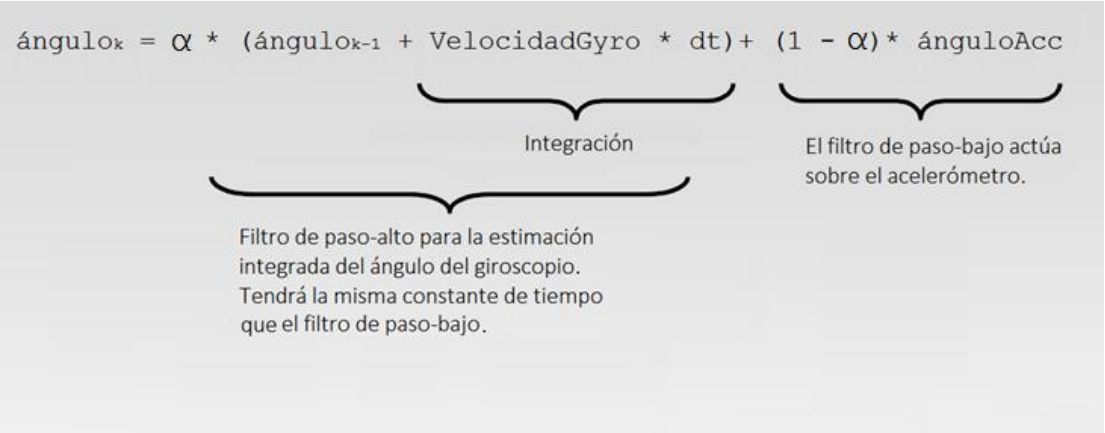

Figura IV.21 Esquema del filtro complementario [4].

Las ventajas de este filtro complementario son:

- Reduce el ruido y la deriva producido por los sensores
- Reduce el retardo en la estimación del ángulo
- No implica un coste excesivo en tiempo de proceso

3.4.2. Filtro Complementario para control en velocidad angular

En el desarrollo experimental de la memoria, se va a demostrar que para desarrollar un control estable y robusto para nuestro sistema gimbal, las medidas del acelerómetro son muy ruidosas y el resultado de filtrar dichas señales sigue siendo demasiado ruidoso para implementar dicho control. Por tanto se optará por el control basado en la velocidad calculada por los giroscopios. Como se tomarán directamente las medidas del sensor sin integrar, éstas serán mucho más precisas y fiables.

El argumento teórico del filtro complementario para el giroscopio es el mismo al usado en el anterior filtro complementario. Se aplica el filtro de paso alto a las medidas filtradas en estados anteriores y el filtro de paso bajo a la medida de velocidad del giroscopio. El filtro sigue combinando sus dos partes de manera que siempre suman uno, consiguiendo una estimación precisa y lineal en la salida.

$$VelocidadFiltrada_K = (1 - \alpha) \cdot VelocidadFiltrada_{K-1} + \alpha \cdot VelGyro$$

Al igual que en el aprtado anterior, podemos deducir la constante de tiempo para los filtros paso-bajo y paso-alto como:

$$\tau = \frac{\alpha \cdot dt}{(1 - \alpha)}$$

Por lo que si sabemos la frecuencia de muestreo (dt^{-1}) y la constante de tiempo del filtro (τ), podemos escoger la ganancia del filtro (α). Y también, para periodos de tiempo más cortos de la constante de tiempo (τ), el filtrado de estados anteriores tiene prioridad y los velocidades recibidas en el instante actual debidas a pequeños movimientos se filtran.

Para períodos más largos de la constante de tiempo (τ), la medida de velocidad del giroscopio es prioritaria frente a las velocidades filtradas en estados anteriores. Resumiendo, este filtro define el límite de confianza entre el las medidas del giroscopio en el instante actual y los valores de sus medidas filtradas en instantes anteriores.

Las ventajas de aplicar el filtro complementarioa la medida del giroscopio son:

- No implica un coste excesivo en tiempo de proceso
- No hace falta el uso de las medidas ruidosas del acelerómetro
- No es necesaria la integración discreta y por tanto no hay deriva en las medidas

4. CÁPITULO

Estrategia de control

En este capítulo, se detallan los fundamentos teóricos de la electrónica de control del sistema y el desarrollo de la implementación de la misma.

4.1. Planteamiento del sistema de control

Para conseguir la estabilización total del sistema, es necesario gobernar los actuadores a partir de las señales obtenidas por los sensores mediante un control en bucle cerrado.

Éste lazo de control puede ser implementado de diferentes maneras atendiendo a las características intrínsecas de los sensores o los motores. Desde un principio se plantean dos estrategias diferentes para dicho propósito: control de posición angular o control de velocidad angular.

Para estudiar la viabilidad de dichas estrategias de control, se realiza un estudio estadístico con ayuda de Matlab de las lecturas del acelerómetro y del giroscopio. Para ello, se toman medidas de los sensores mientras se encuentran en reposo, se guardan en un archivo de texto y se procesan con Matlab, guardando las medidas como variables en el WorkSpace.

Se obtienen las gráficas de las medidas de los sensores en reposo en función del tiempo donde se pueden apreciar los valores entre los que oscila el ruido de cada sensor.

Dichas gráficas se corresponden a las Figuras IV.22 y IV.23 , donde se puede apreciar que para las medidas en reposo del acelerómetro, el ruido oscila entre -1.2º y 0.2º. Y las medidas en reposo del giroscopio, contienen un ruido que oscila entre ±0.3º/s. Si se comparan estos valores entre sí, podemos darnos cuenta de que el ruido del acelerómetro tiene un rango de oscilación bastante mayor al del giroscopio.

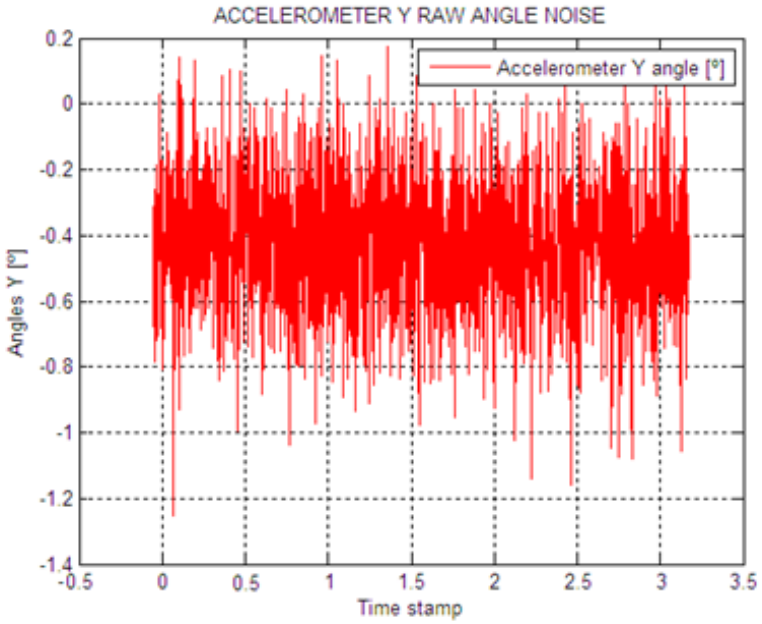

Figura IV.22 Ruido leído con el acelerómetro en reposo.

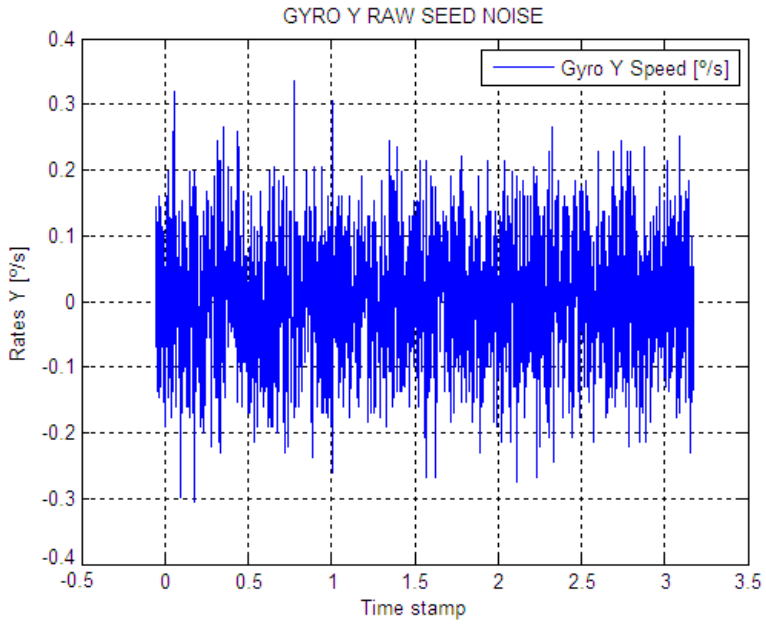

Figura IV.23 Ruido leído con el giroscopio en reposo.

Se calculan la media y la desviación típica de dichas muestras tomadas en ambos sensores. Estos valores se pueden ver en la Tabla IV-2.

	Media	***Desviación típica***
Acelerómetro	$\bar{X} = -0.4338°$	$\sigma = \pm 0.1965°$
Giroscopio	$\bar{X} = 0.0071°/s$	$\sigma = \pm 0.0933°/s$

Tabla IV-2 Medias y desviaciones típicas de los sensores obtenidas con Matlab.

Además se obtienen los histogramas del ruido que acompaña a las medidas leidas en los sensores (Figuras IV.24 y IV.25). Ambos histogramas representan una distribución gaussiana del ruido de las medidas. El giroscopio tiene media nula, como era de esperar debido a la situación de reposo del sensor durante la toma de muestras. Y el acelerómetro presenta una asimetría en su histograma, además de su media no ser nula debido a la leve inclinación del sensor en el momento del ensayo.

Comparando los resultados obtenidos, se puede observar como la desviación típica obtenida con el acelerómetro es mayor a la que se obtiene en el osciloscopio, por lo tanto tendrá una medida más dispersa. También el número de muestras cercanas a la media según los histogramas es menor para el acelerómetro.

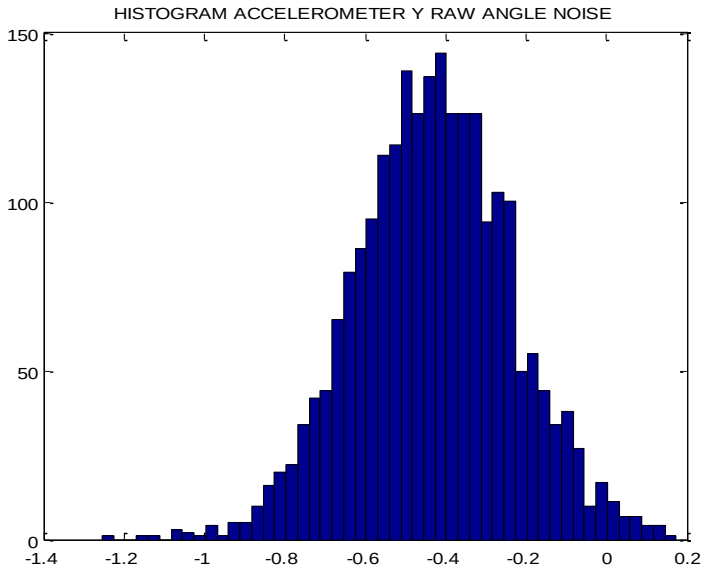

Figura IV.24 Histograma del ruido en el acelerómetro.

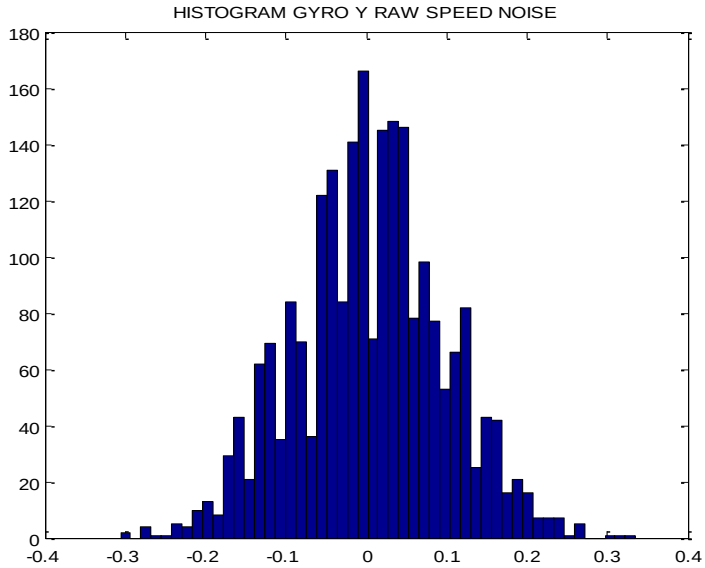

Figura IV.25 Histograma del ruido en el giroscopio.

En vista de la comparativa del párrafo anterior, se puede concluir diciendo que el acelerómetro es un sensor más ruidoso y sensible a perturbaciones externas que el giroscopio.

Para implementar el control de poscición angular se precisa del ángulo obtenido de la señal de los acelerómetros mediante el cálculo de la arcotangente, pero como se ha analizado en los párrafos anteriores, el acelerómetro es muy sensible a fuerzas externas y bastante ruidoso. Por lo que se necesita la corrección de la medida con un filtro complementario usando el giroscopio que también necesita ser acondicionado, lo que implica una integración discreta que provocará una deriva en la medida del ángulo del giroscopio. Aún así el ángulo obtenido para el control de posición angular seguirá siendo demasiado ruidoso, por lo que se descarta esta estrategia de control.

Al final, la opción elegida es la implementación de un control de velocidad angular basado en la medida del osciloscopio ya que es más preciso y no hace falta integrar su señal, evitanto la deriva de su medida.

La velocidad de los motores puede ser controlada en un lazo cerrado de control a través de la medición de la velocidad angular dada por el sensor.

Para ello se calcula un error de velocidad restando la velocidad de referencia o consigna y la velocidad medida y se usa un controlador PID que amplifique el error de velocidad y ajuste dinámicamente el ciclo de trabajo de los motores PWM. El diagrama de bloques del control se corresponde con el de la Figura IV.26. Donde la referencia de velocidad es *R(s)* y la medida de velocidad es *Y(s),* ambas en grados por segundo (º/s), el

error de velocidad es *e(s)* y la salida del controlador PID es *u(t),* que a su vez es la entrada a los motores.

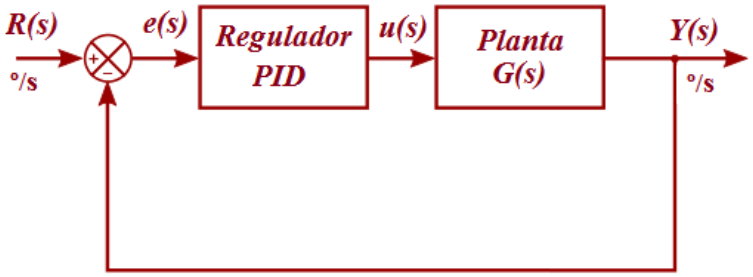

Figura IV.26 Diagrama de bloques del LAZO CERRADO de control.

4.2. Controlador PID

Un controlador o regulador PID [6] es un mecanismo de control basado en una realimentación o feedback. Éste calcula el error o desviación entre un valor medido y un valor deseado producido por las perturbaciones que se introducen en la planta.

Un control PID consiste en un algoritmo compuesto por tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional depende del error actual. El Integral depende de los errores pasados y el Derivativo es una predicción de los errores futuros. La suma de estas tres acciones es capaz de minimizar el error del sistema y ajustar el comportamiento dinámico del proceso:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \cdot \frac{de}{dt}$$

Donde K_p , K_i y K_d, todos positivos, son los coeficientes para los términos proporcional, integral y derivativo, respectivamente.

La ecuación equivalente en el Dominio de Laplace del controlador PID es la siguiente:

$$U(s) = K_p + \frac{K_i}{s} + K_d \cdot s$$

La *acción de control proporcional* consiste en el producto entre la señal de error y la constante proporcional (K_p) para lograr que el error en estado estacionario se aproxime a cero. Existen valores límites para la constante proporcional a partir de los cuales, el

sistema puede alcanzar valores superiores a los deseados provocando el fenómeno de sobreoscilación, que por razones de seguridad no debe sobrepasar el 30%. Para conseguir estabilidad en el sistema, lo ideal es que la parte proporcional no produzca ninguna sobreoscilación. La parte proporcional no considera el tiempo, asume un error permanente, pero realmente el error si que depende del tiempo, serán las acciones integral y derivativa quienes tengan en cuenta la variación del error con el tiempo.

La *acción de control Integral* tiene como propósito disminuir y eliminar el error residual en estado estacionario que provoca la acción proporcional. Este término es el error acumulado en instantes anteriores que va compensando la acción actual del controlador. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo, multiplicándola por una constante (K_i) y sumándola a la acción proporcional. Sin embargo, dado que el término integral responde a los valores acumulados en el pasado, puede provocar que se rebase el valor deseado. El control integral se utiliza para eliminiar el offset (desviación permanente de la variable con respecto al punto de consigna) que provoca la acción proporcional.

La *acción derivativa* se manifiesta cuando hay un cambio en el valor absoluto del error, esto quiere decir que si el error es constante, solamente actúan las acciones proporcional e integral. Su función es calcular la velocidad con la que cambia el error evitando que el error se incremente. Para ello, se deriva el error respecto al tiempo y se multiplica por una constante (K_d). Después se suma a las anteriores acciones de control.

La acción derivativa predice el comportamiento del sistema y por lo tanto mejora el tiempo y la estabilidad del sistema. Si el tiempo de acción derivativa es grande, la velocidad de cambio del error será alta y habrá inestabilidad en el proceso. Cuando por el contrario, el tiempo de acción derivativa es pequeño, la señal de salida oscilará demasiado sobre el valor de consigna. El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones. Esta acción se utiliza en pocas ocasiones debido a su sensibilidad al ruido.

Algunas de las aplicaciones más comunes son:

- Lazos de Temperatura (Aire acondicionado, Calentadores, Refrigeradores, Extrusores, etc.)
- Lazos de Nivel (control del nivel en tanques, depósitos, etc. de cualquier líquido.)
- Lazos de Presión (control de presión predeterminada en tanques, tubos, recipientes, etc.)

- Lazos de Flujo (mantienen la cantidad de flujo dentro de una tubería)
- Lazos de Control de Motores (Control de la excitación de motores según una entrada de control)

4.3. Modelado de la planta

El diseño de un controlador continuo o discreto, ya sea mediante técnicas clásicas o en variables de estado, requiere de un modelo de la planta a controlar que caracterice su comportamiento dinámico. Este modelo permite al diseñador realizar y validar mediante simulación el ajuste de los parámetros del controlador PID que permiten obtener una respuesta que satisfaga las especificaciones de diseño. En este apartado se plantean diferentes alternativas para obtener el modelo de un sistema como paso previo al diseño de un controlador.

Básicamente, un modelo es una herramienta que permite predecir el comportamiento de un sistema sin necesidad de experimentar sobre él.

Existen dos métodos principales para obtener el modelo de un sistema:

- *Modelado teórico.* Se trata de un método analítico, en el que se recurre a leyes básicas de la física para describir el comportamiento dinámico de un fenómeno o proceso.
- *Identificación del sistema.* Se trata de un método experimental que permite obtener el modelo de un sistema a partir de datos reales recogidos de la planta bajo estudio.

Para obtener el modelo de la plataforma gimbal usada en este proyecto se va a hacer uso del método de identificación del sistema con ayuda de Matlab.

El primer paso a seguir es la identificación de la planta en lazo abierto. El diagrama de bloques en lazo abierto se puede ver en la Figura IV.27, donde la entrada a la planta esta vez será la señal de excitación de los motores, y la salida será la velocidad que obtenemos tras filtrar la señal del sensor.

Figura IV.27 Diagrama de bloques de la planta en LAZO ABIERTO.

La herramienta Matlab dispone de una amplia colección de funciones aplicables al campo de la identificación de sistemas, agrupadas en el *System Identification Toolbox*, que constituyen una herramienta de gran utilidad para la identificación y modelado de sistemas dinámicos. Estas funciones incluyen diferentes algoritmos para el ajuste de parámetros en todo tipo de modelos lineales. Así mismo, permiten examinar las propiedades de los modelos obtenidos, e incluso realizar un preprocesamiento de los datos utilizados para la identificación, en caso de que sea necesario. Las versiones del Toolbox de Identificación a partir de la versión 4.0 permiten trabajar en dos modos distintos:

- *Modo comando*. En este modo, se trabaja directamente desde la ventana de comandos de Matlab, realizando llamadas a las diferentes funciones que componen el Toolbox de Identificación.
- *Mediante Interfaz de Usuario (GUI).* A diferencia de otros toolboxes que proporciona Matlab, el Toolbox de Identificación proporciona un interfaz gráfico que facilita el trabajo con las funciones anteriores, de forma transparente al usuario, sin necesidad de llamarlas desde la ventana de comandos.

Para arrancar el interfaz de usuario, basta con teclear en la línea de comandos de Matlab el comando `>>ident` tal y como se muestra en la Figura IV.28. Aparecerá una ventana como la mostrada en la Figura IV.29.

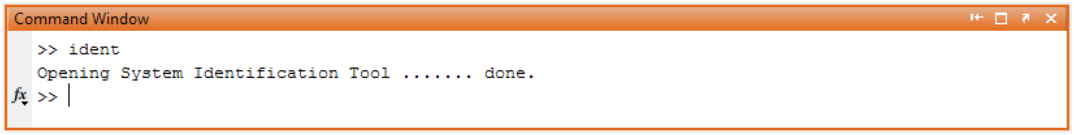

Figura IV.28 Abrir el GUI de identificación de sistemas.

La entrada y salida del lazo abierto que introduciremos en la tabla de datos serán las variables obtenidas con el software implementado que se guardan en el WorkSpace de Matlab. Se podrán examinar estos datos en el menú “Data Views”. Y se puede realizar un preprocesado de los mismos en el menú desplegable “Preprocess”.

Para la estimación de los modelos se usa el menú desplegable “Estimate”, donde podremos elegir el tipo de modelo que más interese según las exigencias del control. Los modelos obtenidos se incluirán automáticamente en el tablero de modelos. Podremos analizar las propiedades de los modelos obtenidos mediante diversas representaciones de los mismos, escogidas en el menú “Model Views”.

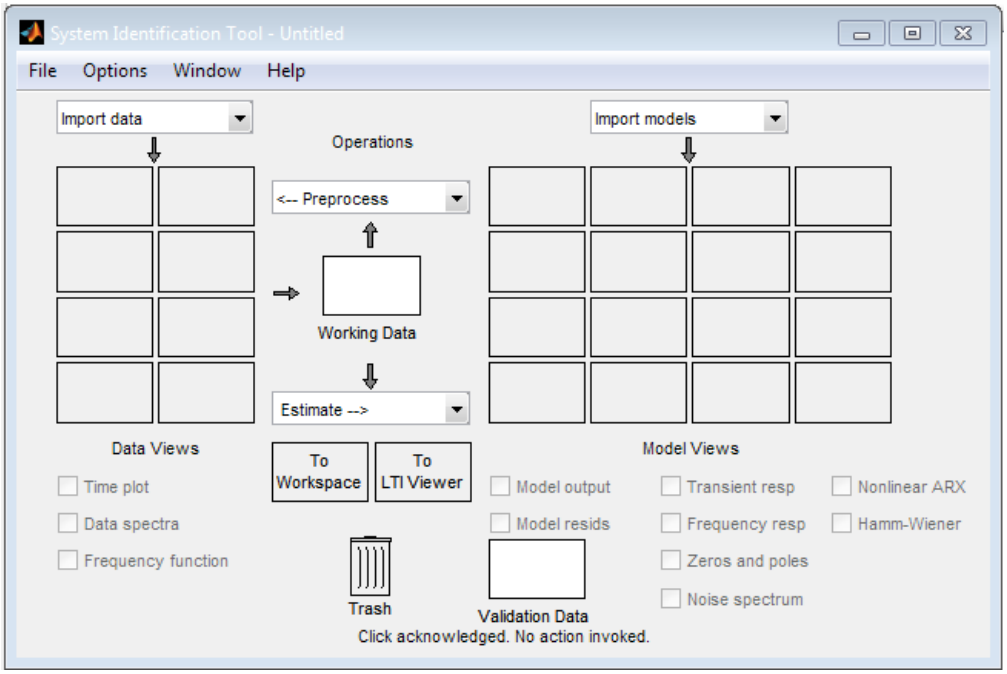

Figura IV.29 Ventana principal del GUI System Identification Toolbox de Matlab.

La planta utilizada en el presente trabajo se modelará mediante su representación en función de transferencia como un sistema de orden uno con la ayuda del software presentado en este apartado.

5. CAPÍTULO

Desarrollo experimental

El presente capítulo se centra en la presentación de los resultados de funcionamiento y puesta en marcha del sistema.

Se describe el método para adquirir las señales de los sensores de la IMU y una comparativa de los diferentes métodos implementados para el filtrado de señal. También se dan las pautas necesarias para escribir el código que maneja los motores brushless. Así como el funcionamiento de todos los transductores en un lazo cerrado de control con un regulador PID.

Por último, se realiza un balance de los resultados y de los errores cometidos.

5.1. Adquisición de datos del MPU-6050

Para la adquisición de los datos arrojados por los sensores del MPU6050 se puede optar por utilizar los registros internos del sensor o hacer uso de la librería MPU6050 creada por Jeff Rowberg [17]. Quien debido a la falta de buena documentación disponible sobre el funcionamiento interno de este dispositivo, ha publicado dicha librería que permite un uso totalmente transparente del dispositivo para todas las funciones que soporta el procesador digital de movimiento (DMP) y para su completa configuración.

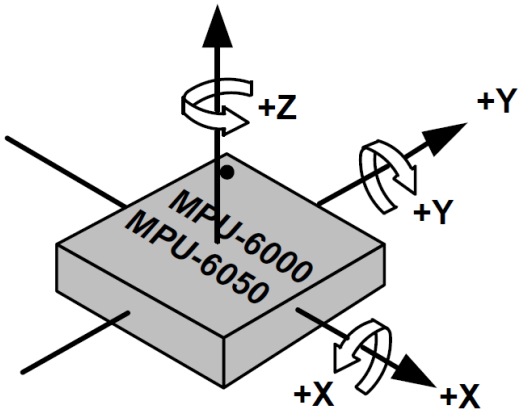

Figura IV.30 Orientation of Axes of Sensitivity and Polarity Rotation [8].

La colocación del sensor en el sistema y conocer los ejes que toma como referencia el sensor son dos aspectos de crucial importancia a la hora de adquirir las señales y su posterior procesamiento. La Figura IV.30 corresponde a un diagrama del datasheet del dispositivo y muestra la disposición que toman los ejes en el chip tomando como referencia la patilla número uno del circuito integrado marcada con un punto negro.

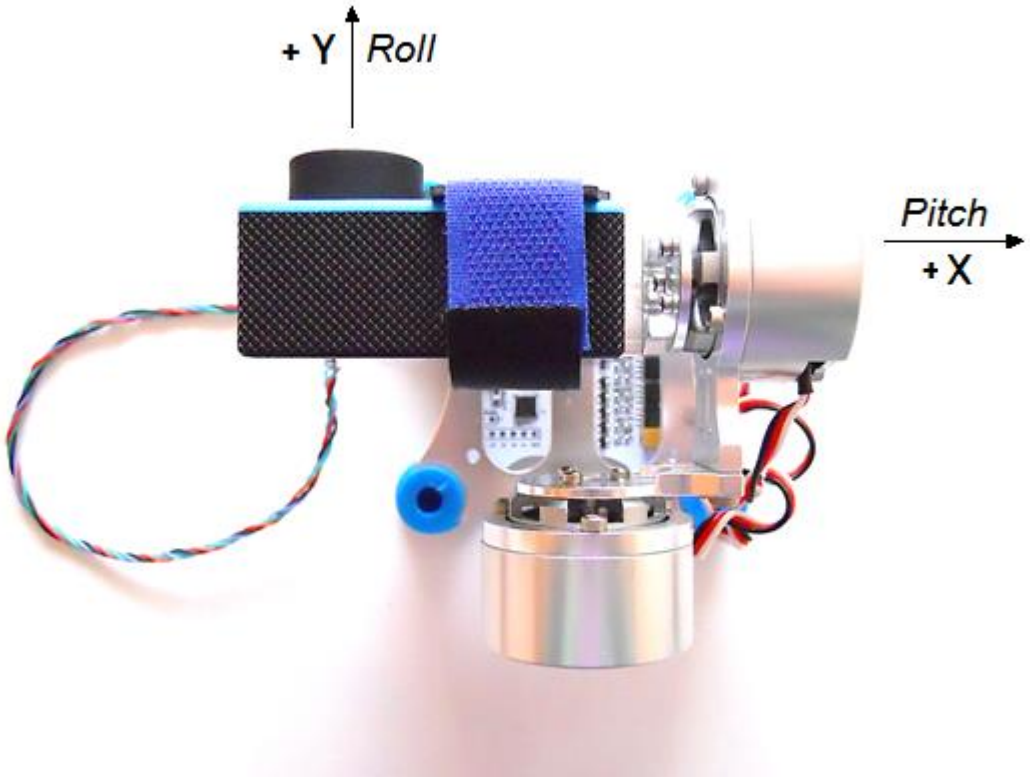

Figura IV.31 Colocación de la IMU en la plataforma gimbal.

En la plataforma gimbal usada en este proyecto se coloca el sensor MPU6050 en la pletina de aluminio que soporta la cámara. De tal forma que según los ejes de referencia ilustrados en la Figura IV.30, el plano formado por los ejes XY será el plano donde se

encuentre la cámara. El eje roll se corresponderá con el eje Y, cuyo sentido positivo se corresponderá con el sentido de disparo del objetivo de la cámara. Y el eje pitch será el eje X con sentido positivo mirando hacia el motor brusless que hace girar el eje pitch.

Esta configuración se muestra en la Figura IV.31, una imagen real de la plataforma gimbal usada en el trabajo en la que se puede ver lo explicado en el anterior párrafo siguiendo la referencia que el fabricante del MPU6050 proporciona en su datasheet (Figura IV.30).

5.1.1. Acondicionamiento de los acelerómetros

Como se ha introducido en el capítulo 3, los acelerómetros son transductores capaces de medir aceleraciones como fuerzas por unidad de masa. Pero para el presente trabajo no es de utilidad medir aceleraciones, pero se pueden obtener los ángulos de inclinación de cada eje de orientación comparando dichas aceleraciones con la aceleración de la gravedad terrestre.

Si sabemos que la gravedad es aproximadamente 9.81 m/s², y sabemos la aceleración en los tres ejes del acelerómetro, mediante trigonometría es posible calcular el ángulo de inclinación de cada eje de la IMU. Esto se puede hacer fácilmente mediante el uso de la arco-tangente trigonométrica, una función fácil de implementar en el código con el nombre `atan2`.

El primer paso para comunicarse con el acelerómetro es leer las medidas en el registro correspondiente del MPU6050 por medio de comunicación con el bus I2C y obtener los valores RAW del sensor guardándolos en variables de tipo entero de 16 bits. El siguiente código muestra el procedimiento a seguir para la comunicación con el dispositivo, las librerias necesarias, la inicialización del mismo y la toma de datos RAW que debe hacerse una vez en cada periodo de muestreo.

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

#define TIME_STEP              10          //Periodo de muestreo
#define FACTOR_GYRO            131         //Sensibilidad giroscopio
#define GRAVITY                9.81        //Constante de gravedad
#define FACTOR_ACEL            2048        //Sensibilidad acelerómetro
#define R2G                    180 / PI    //Conversor radianes a grados

MPU6050 myMPU6050;                         //creamos objeto myMPU6050
```

```
int16_t ax, ay, az, gx, gy, gz;              //Variables RAW del MPU6050
long int timer;

void setup() {

  Wire.begin();           //Abrimos conexión I2C
  Serial.begin(115200);   //Abrimos conexión con el monitor serial

  /*Inizializamos y testeamos el MPU6050*/
  Serial.println("Inicializando MPU6050...");
  myMPU6050.initialize();
  Serial.println(myMPU6050.testConnection() ? "MPU6050 conexión con
éxito" : "MPU6050 error");

  /*mostramos la temperatura del MPU6050 por el monitor*/
  int16_t tempRaw = myMPU6050.getTemperature();
  double temperature = (double)tempRaw / 340.0 + 36.53;
  Serial.print("Temperature MPU6050: ");
  Serial.println(temperature);
}

void loop() {
  timer = millis();//Guarda el tiempo de inicio del bucle

  myMPU6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); //Lee RAW MPU6050

  timer = (millis() - timer); //Calcula tiempo invertido en los cálculos
  if(timer > TIME_STEP)
    Serial.println("Error: Overflow TIME_STEP");
  else
    delay(TIME_STEP - timer); //Pausa tiempo necesario para completar TS
}
```

Para obtener el valor de la aceleración en la escala adecuada, es necesario dividir la señal adquirida entre el factor de escala correspondiente o sensibilidad del sensor. Dicha sensibilidad está condicionada por el fondo de escala que queramos elegir para las medidas tomadas en el dispositivo. De tal forma que, si queremos un fondo de escala del rango de ±16g, en una salida en complemento a dos con un ADC de 16 bits, basta con dividir cada valor en bruto del sensor entre una sensibilidad de 2048 LSB/g. Esta información se puede conseguir en las especificaciones técnicas del acelerómetro que se pueden encontrar en el datasheet del MPU6050 proporcionado por el fabricante. (Ver Figura IV.32).

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

6.2 Accelerometer Specifications

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

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	

Figura IV.32 Fragmento del Datasheet del acelerómetro del MPU6050 [8].

El siguiente fragmento de código muestra el cálculo de los ángulos de inclinación a partir de las medidas del acelerómetro. Se aplica la fórmula trigonométrica arco-tangente a las aceleraciónes de cada eje divididas entre la sensibilidad y por último, se pasa de radianes a grados.

```
/*Cálculo de los ángulos del acelerómetro*/
float angulo_ax = (atan2(((ax) * GRAVITY / FACTOR_ACEL) , sqrt
(square(((ay) * GRAVITY / FACTOR_ACEL)) + square(((az) * GRAVITY /
FACTOR_ACEL))))) * R2G;
float angulo_ay = (atan2(((ay) * GRAVITY / FACTOR_ACEL) , sqrt
(square(((ax) * GRAVITY / FACTOR_ACEL)) + square(((az) * GRAVITY /
FACTOR_ACEL))))) * R2G;
```

Donde `FACTOR_ACEL` es la sensibilidad del acelerómetro, `GRAVITY` es la constante de la aceleración de la gravedad y `R2G` es una constante para pasar de radiantes a grados.

Los ángulos obtenidos deben ser variables de tipo coma flotante dependiendo de la arquitectura del procesador utilizado.

5.1.2. Acondicionamiento de los giroscopios

Los giroscopios MEMS del MPU6050 miden velocidades angulares en º/s. Se hará uso de la librería MPU6050 para obtener sus valores. Como se ha explicado en el capítulo 3, podemos optar por utilizar directamente las velocidades angulares para realizar el control, o la integración discreta de estas velocidades angulares para obtener los ángulos de orientación.

El problema que tiene la integración discreta es que provoca una deriva en la medida de los giroscopios. Para evitar esta deriva, necesitamos crear una función que calcule la media de los valores RAW en reposo u offset y restarlo a las señales. Esto va a ayudarnos a quitar gran parte del error y no acumularlo en las siguientes medidas. Las siguientes líneas de código corresponden a la función que calcula la linea base de la medida de la velocidad angular proporcionada por los giroscopios.

```
/*Función para calcular el offset del giroscopio*/
void GetOffset_Gyro(float* offset_gx, float* offset_gy, float*
offset_gz) {
  int i;
  double acuX = 0;
  double acuY = 0;
  double acuZ = 0;

  for (i = 0; i < NUM_MUESTRAS; i++) {
    myMPU6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    acuX += gx;
    acuY += gy;
    acuZ += gz;
    delay(1);
  }
  acuX /= NUM_MUESTRAS;
  acuY /= NUM_MUESTRAS;
  acuZ /= NUM_MUESTRAS;

  *offset_gx = acuX;
  *offset_gy = acuY;
  *offset_gz = acuZ;
}
```

Esta función simplemente calcula la media de las medidas del giroscopio durante las primeras dos mil muestras, mientras el sistema se encuentra en reposo. Esta media o bias, será el offset de la señal del giroscopio.

El primer paso para comunicarnos con nuestro giroscopio es de nuevo, leer los registros internos del dispositivo por medio del bus I2C y obtener los valores RAW del MPU6050 guardándolos en variables de tipo entero de 16 bits.

Ahora es necesario obtener el valor de la velocidad angular en la escala adecuada, para ello, se ha de dividir la señal adquirida entre la ganancia o sensibilidad del sensor. Dicha sensibilidad está condicionada por el fondo de escala que queramos elegir para las medidas tomadas en el dispositivo. De tal forma que, si queremos un fondo de escala del rango de ±250 º/s, suficiente para el sistema gimbal, con un ADC de 16 bits, basta con dividir cada valor en bruto del sensor entre un factor de escala o sensibilidad de 131 LSB/(º/s). La Figura IV.33 corresponde al fragmento del datasheet donde se encuentran las

especificaciones de los giroscopios del MPU6050.

6.1 Gyroscope Specifications

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

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	

Figura IV.33 Fragmento del Datasheet del giroscopio del MPU6050 [8].

El acondicionamiento de la señal de los giroscopios para la obtención de la velocidad angular se puede ver en el siguiente fragmento de código, donde se resta el offset calculado y después se divide entre el factor de escala `(FACTOR_GYRO)`:

```
/* Cálculo de velocidades angulares para 3 ejes */
float speedX = (gx - offset_gx) / FACTOR_GYRO;
float speedY = (gy - offset_gy) / FACTOR_GYRO;
float speedZ = (gz - offset_gz) / FACTOR_GYRO;
```

Una vez obtenida la velocidad, como se ha explicado en el capítulo 3, se puede obtener la posición angular a partir de la integración de la medida de velocidad angular:

```
/* Cálculo de ángulos del giroscopio mediante integración discreta */
float angulo_gx = angulo_gx + (speedX * TIME_STEP);
float angulo_gy = angulo_gy + (speedY * TIME_STEP);
float angulo_gz = angulo_gz + (speedZ * TIME_STEP);
```

Donde `TIME_STEP` es el periodo de muestreo impuesto en el sistema.

Tanto los ángulos, como las velocidades obtenidas deben ser variables tipo coma flotante dependiendo de la arquitectura del procesador utilizado. En este caso, se utilizará tipo `float` que en el caso del ATmega 328P corresponde a una variable de 32 bits (4 byte) de memoria, suficiente para la aplicación.

5.1.3. Filtrado de señales de los sensores

Como ya se ha estudiado en el capítulo 3, el filtro complementario usa los conceptos básicos de un filtro de paso bajo y un filtro de paso alto para fusionar medidas en diferentes instantes de tiempo y atenuar errores dotando de mayor peso a la medida que más convenga según la función. Se han aplicado dos modos de filtrado diferentes según la estrategia de control a seguir, con afán de realizar una comparativa final y elegir una estrategia u otra.

a) Filtro Complementario para control en posición

El código implementado para el filtro complementario para control en posición se calcula en cada iteración dentro del bucle principal del programa. El algoritmo resultante queda de la forma:

```
/* FILTRO COMPLEMENTARIO para ángulos finales */
float angulo_fCx = ALPHA * (angulo_fCx + (speedX * TIME_STEP)) + (1 -
ALPHA) * angulo_ax;
float angulo_fCy = ALPHA * (angulo_fCy + (speedY * TIME_STEP)) + (1 -
ALPHA) * angulo_ay;
```

Donde `ALPHA` es la constante del filtro de paso bajo que va a ser un valor comprendido entre 0 y 1. El valor de esta constante para el filtro de paso alto debe ser complementario, por lo que tendrá una ganancia de (`1 - ALPHA`). Ambas constantes siempre suman uno para que el filtro de paso bajo y el filtro de paso alto se combinen y sean complementarios.

En la figura IV.34 se muestra el funcionamiento de este filtro. Para la muestra de resultados se ha usado una constante del filtro de 0.1 que significa que en el cálculo del ángulo tienen más importancia las medidas del acelerómetro que la velocidad angular integrada. Se puede corroborar que el ruido del acelerómetro es demasiado elevado para utilizarlo como señal en un control electrónico ya que la señal filtrada se ve afectada también por este ruido.

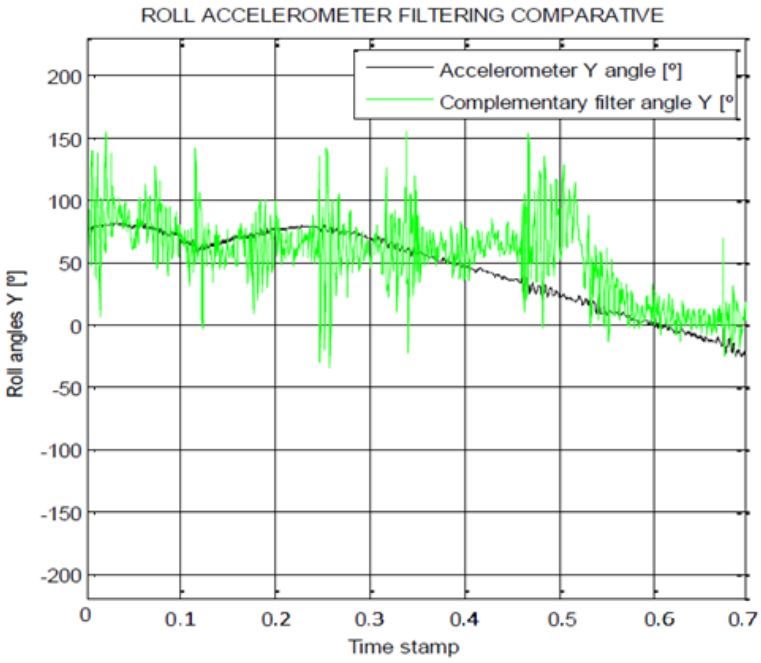

Figura IV.34 Filtro complementario para el control en posición.

b) Filtro Complementario para control en velocidad

El algoritmo implementado para el filtro complementario para control en velocidad también debe calcularse en cada iteración dentro del bucle principal del programa. El algoritmo resultante queda de la forma:

```
/* FILTRO COMPLEMENTARIO para velocidad angular de giroscopios */
float speedfCx = (1 - ALPHA) * speedfCx + ALPHA * speedX;
float speedfCy = (1 - ALPHA) * speedfCy + ALPHA * speedY;
```

Donde nuevamente, `ALPHA` es la constante del filtro que va a ser un valor comprendido entre 0 y 1. El valor de esta constante para el filtro de paso alto será (`1 - ALPHA`) y de nuevo, ambas constantes siempre suman uno para que las medidas filtradas mediante el filtro de paso bajo y el filtro de paso alto se combinen y sean complementarios.

La Figura IV.35 muestra el funcionamiento del filtro complementario aplicado a la velocidad angular del giroscopio. Para el ensayo que se ilustra se ha utilizado una constante del filtro de valor 0.02, es decir, se establece una mayor confianza en las medidas filtradas en instantes anteriores. Se puede ver como el filtro juega un papel importante en la medida de la velocidad, eliminando el ruido y provocando un pequeño retardo en la medida.

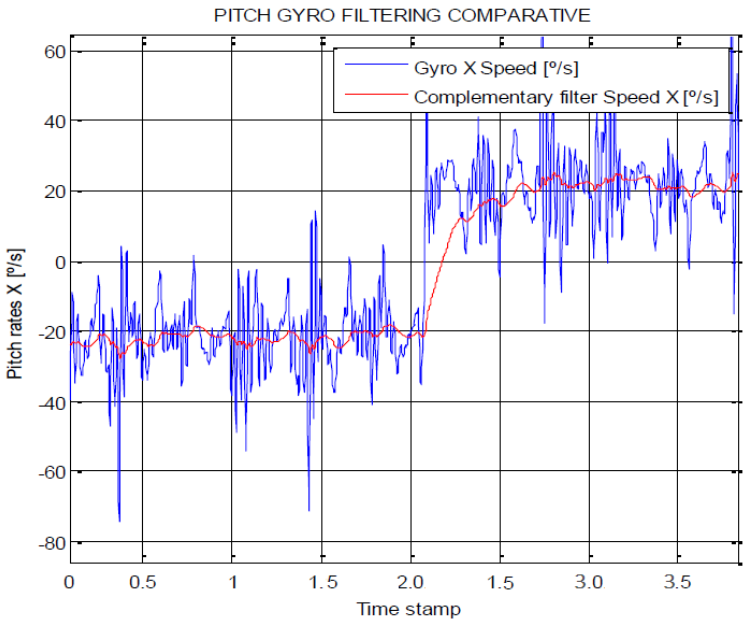

Figura IV.35 Filtro complementario para el control en velocidad angular.

5.2. Manejo de motores brushless BLDC

Para implementar el programa necesario para poder controlar los motores brushless hay que hacer uso de los tres timers que dispone el ATmega 328P de ATMEL. Todos los timers tienen dos salidas, que se corresponden a dos pines del microprocesador. El Timer 0, el único de 8 bits controla los pines 3 y 11 y aplicando el preescalado unidad, conseguimos una frecuencia de salida en dichos pines de 62,5 kHz. Los dos restantes timers (Timer1 y Timer2) son de 16 bits y controlan los pines 5, 6, 9 y 10 respectivamente. Aplicando el divisor unidad, obtenemos una frecuencia de salida en sus pines de 31.372,55 Hz, aproximadamente 32 kHz. Estos seis pines mencionados (3, 11, 5, 6, 9 y 10), son las seis salidas PWM que dispone el ATmega 328P [15].

La siguiente función, describe como programar los timers del ATmega 328P para configurar las salidas PWM a la frecuencia adecuada:

```
void setPwmFrequency(int pin) {
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
    if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | 0x01; //Timer0_0x01->set f=62500Hz
    } else {
      TCCR1B = TCCR1B & 0b11111000 | 0x01; //Timer1_0x01->set f=32kHz
    }
  }
  else if(pin == 3 || pin == 11) {
    TCCR2B = TCCR2B & 0b11111000 | 0x01; //Timer2_0x01->set f=32kHz
  }
}
```

Una vez configurados los timers y las frecuencias de las salidas PWM, hay que enviar pulsos de ancho modulado a dichas frecuencias para alimentar las tres fases del motor siguiendo las secuencias de conmutación oportunas. Para ello creamos tres arrays con una cantidad generosa de elementos con fin de mejorar la resolución. Estos tres arrays describen tres funciones senoidales desfasadas 120º con una amplitud que va desde 0 hasta 255, valor máximo de 8 bits que representa el ancho de pulso total de un periodo.

Para enviar los valores de 0 a 255 a las salidas PWM es necesario utilizar la función `analogWrite(pin, value)` a la que se debe pasar como argumentos el pin o la salida PWM a utilizar y el valor de la anchura de pulso deseada. Para describir una vuelta completa en un motor BLDC es necesario recorrer el array antes mencionado por todas sus posiciones variando el ancho de pulso de tal forma que se siga una función senoidal con la máxima resolución posible.

Se desarrolla el siguiente código para el manejo de motores brushles:

```
#define TIME_STEP  10

#define IN1  3// Definición de pines del motor pitch
#define IN2  5
#define IN3  6

unsigned char sinPWM_A[360], sinPWM_B[360], sinPWM_C[360];
int arrayIndex = 0;

float u = 0.0; // Señal de control del motor

void setup() {

  setPwmFrequency(IN1); // PWM frequency = 32 kHz   (make unaudible)
  setPwmFrequency(IN2); // PWM frequency = 62,5 kHz   (make unaudible)
  setPwmFrequency(IN3);

  pinMode(IN1, OUTPUT); // Se configuran los pines como salidas
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);

  for(int i = 0; i < 360; i++){ // Inicialización de los arrays seno
    sinPWM_A[i] = (unsigned char) 127.5 + 127.5 * sin(2.0 * PI / 360.0 *
float(i) + 0.0 * PI / 3.0);
    sinPWM_B[i] = (unsigned char) 127.5 + 127.5 * sin(2.0 * PI / 360.0 *
float(i) + 2.0 * PI / 3.0);
    sinPWM_C[i] = (unsigned char) 127.5 + 127.5 * sin(2.0 * PI / 360.0 *
float(i) + 4.0 * PI / 3.0);
  }

  /*Motores a posición de reposo*/
  analogWrite(IN1, sinPWM_A[0]);
  analogWrite(IN2, sinPWM_B[0]);
  analogWrite(IN3, sinPWM_C[0]);
```

```
  analogWrite(IN4, sinPWM_A[0]);
  analogWrite(IN5, sinPWM_B[0]);
  analogWrite(IN6, sinPWM_C[0]);

  delay(1000);// Se espera 1s para que se queden totalmente en reposo
}

void loop() {
  timer1 = millis();
  moveMotorPitch(u, sinPWM_A, sinPWM_B, sinPWM_C);
  timer2 = millis();

  delay(TIME_STEP - (Timer2-Timer1));
}

void moveMotorPitch(float u, unsigned char* sinPWM_A, unsigned char*
sinPWM_B, unsigned char* sinPWM_C){

  arrayIndexPitch += u; // Se incrementa el índice del vector según u

  int posArray = (int)arrayIndexPitch;

  if(posArray >= 360){
    posArray -= 360;
    arrayIndexPitch -= 360;
  }
  if(posArray < 0){
    posArray += 360;
    arrayIndexPitch += 360;
  }

  analogWrite(IN1, sinPWM_A[posArray]);
  analogWrite(IN2, sinPWM_B[posArray]);
  analogWrite(IN3, sinPWM_C[posArray]);
}
```

Para el motor del eje roll, se implementa el mísmo código, cuidando el posible solape entre variables si se mueven ambos motores a la vez y cambiando las salidas PWM de los pines 3, 5 y 6, del motor pitch, por los pines 9, 10 y 11.

5.3. Comparativa de filtrado de señales

Para comparar la calidad de las señales enviadas por los acelerómetros y giroscopios de la IMU y sus valores filtrados, se ha creado un código que envía las señales de los acelerómetros y giroscopios de los ejes pitch y roll filtradas y sin filtrar (como se ha explicado en al apartado 5.1) a un archivo de texto que posteriormente se tratará con Matlab. Este envío de datos se hace a través del puerto serie una vez por cada iteración del bucle principal. Mientras se toman las muestras de las medidas, se mueven el motores a velocidad constante en ambas direcciones para conseguir gráficas más suaves y así conocer sus efectos.

Para estos ensayos se han configurado ambos filtros con una ganancia de 0.02, lo que implica que practicamente no se tiene en cuenta la integración de la velocidad angular para el filtro complementario del ángulo, y que el filtro complementario de la velocidad angular calcula su valor a partir de valores pasados.

La primera gráfica (Figura IV.36), representa la velocidad sensada en el eje de orientación pitch mientras da vueltas el motor que hace girar el mismo eje. Se puede observar que la velocidad oscila entre ±20 º/s cambiando de dirección siempre a velocidad constante. La señal del giroscopio es ruidosa, debido al ruido que ocasiona el movimiento del motor a cada paso, pero el filtro complementario para control de velocidad filtra perféctamente la velocidad manteniendo una señal más limpia y constante que la de la velocidad angular medida directamente con el giróscopo.

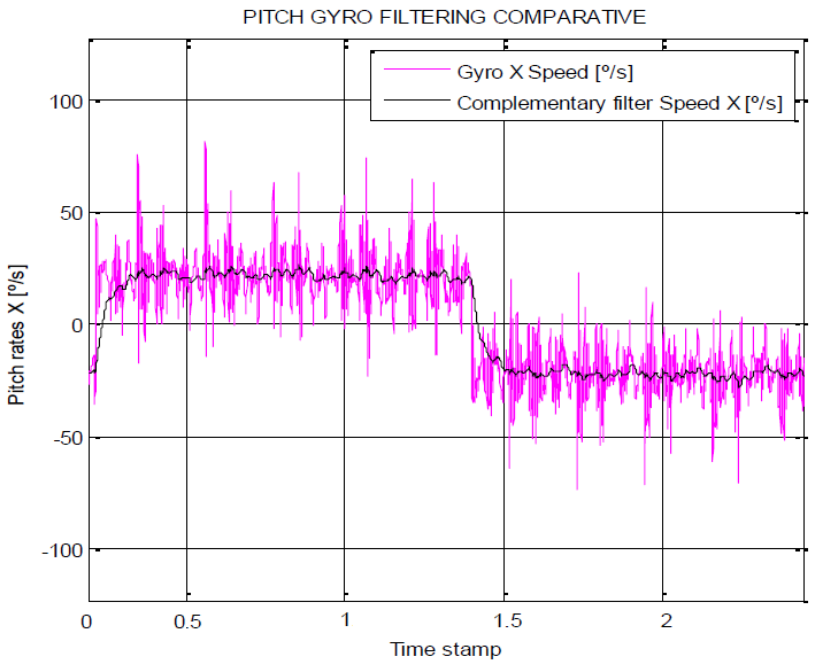

Figura IV.36 Medida del giroscopio en el eje pitch.

La Figura IV.37 muestra el filtrado del acelerómetro en el eje pitch mientras se mueve a velocidad constante el motor del eje roll. En ella se puede apreciar el ruido intrínseco al acelerómetro, pero también el ruido es muy notable en la medida filtrada, lo que provocacaría la inestabilidad de cualquier sistema.

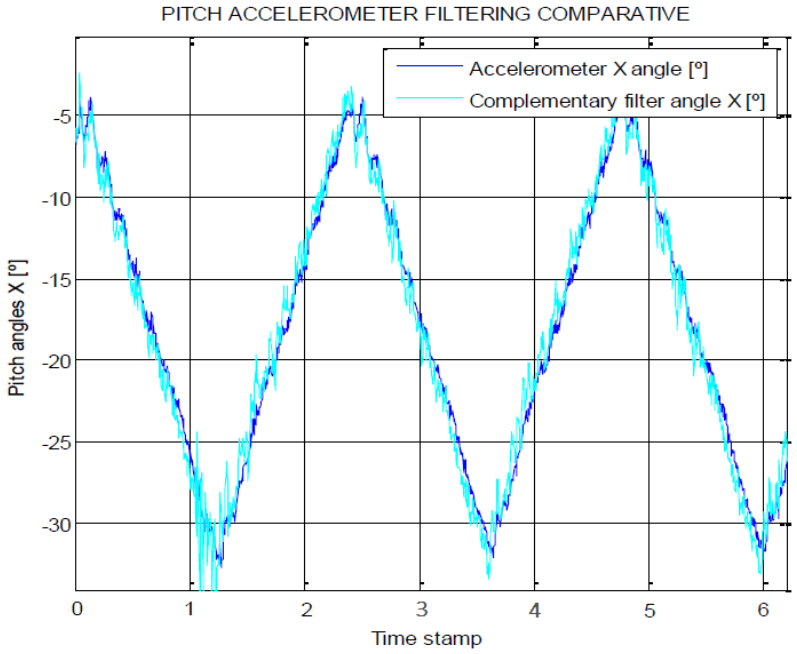

Figura IV.37 Medida del acelerómetro en el eje pitch.

En la gráfica de la Figura IV.38 se vuelve a ver lo ocurrido en la Figura IV.36, la señal del giroscopio está afectada por el movimiento del motor a cada paso creando ruido en la medida, pero el filtro complementario para control de velocidad filtra este ruido manteniendo una velocidad que oscila nuevamente entre ±20 º/s.

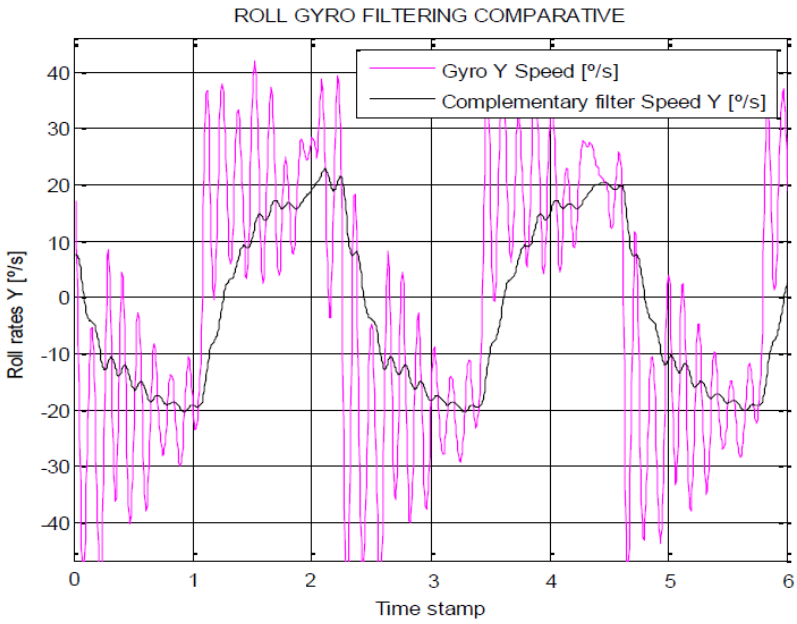

Figura IV.38 Medida del giroscopio en el eje roll.

Por último, en la gráfica de la Figura IV.39 se ven los ángulos descritos por el acelerómetro del eje Y durante el giro a velocidad constante del motor de pitch. Se pueden ver los efectos del ruido en la media del ángulo y como afectan éstos al valor filtrado.

Como conclusión, podemos asumir que pese al ruido provocado por los pasos de conmutación de las fases de los motores, la medida de los giroscopios es más precisa pues mediante un filtrado obtenemos una señal limpia y constante. Mientras que las señales obtenidas mediante los acelerómetros, son también ruidosas pero su valor de filtrado se ve bastante afectado por el ruido del acelerómetro y por los efectos de la integración discreta de la velocidad angular del mismo eje cuando dicha velocidad sufre cambios bruscos.

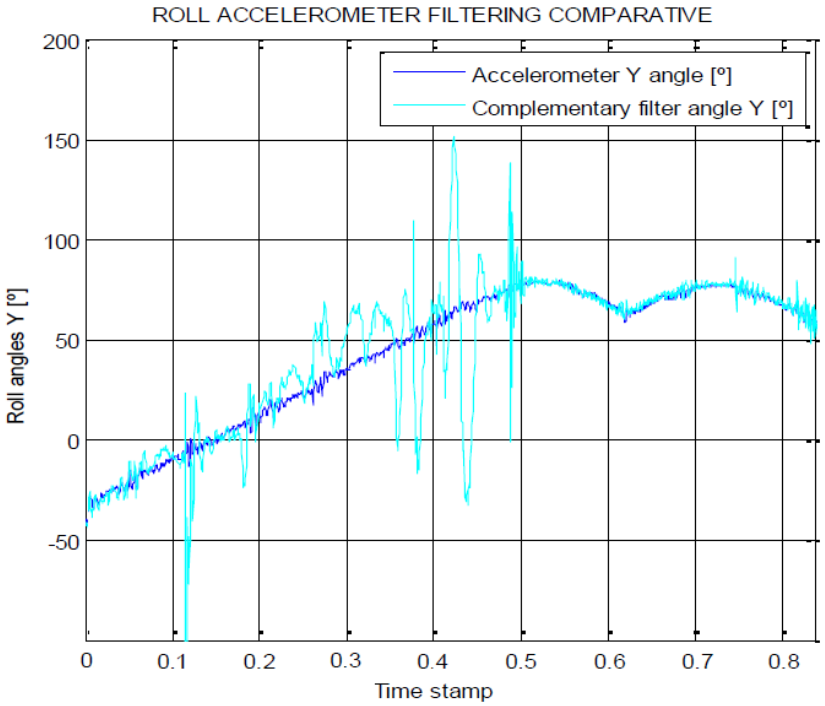

Figura IV.39 Medida del acelerómetro en el eje roll.

Por tanto, la implementación del control electrónico que gobierna el sistema se realizará a partir de la velocidad angular filtrada del giroscopio. El modelo de control será en velocidad, más concretamente, en º/s. El diagrama de bloques que ilustra el control se corresponde al de la Figura IV.26.

5.4. Identificación del modelo de la planta

El modelado de la planta requiere de un programa que simule el comportamiento de la plataforma gimbal en lazo abierto. Para ello, se coloca el sensor IMU en la posición adecuada y se mueve cada motor por separado para ver el estímulo que ocasionan en los sensores. Los motores se mueven en secuencias que cambian de dirección y de velocidad cada ciertos bucles transcurridos, para mejorar así la identificación del modelo de la planta.

Es importante destacar que las ecuaciones que modelan el comportamiento del sistema no son iguales para el sistema sin carga que con carga. Todos los pasos que se describirán en adelante se realizarán con el sistema cargado, esto es, con la cámara dispuesta en el lugar correcto para ser giroestabilizada.

La entrada al lazo abierto del sistema será la señal de control de los motores y la salida serán las velocidades angulares filtradas de las medidas del giroscopio, tal como se ilustra en la Figura IV.27.

Con las variables en el WorkSpace de Matlab, se pueden obtener gráficas para comprobar que la señal de los sensores sigue perfectamente el giro de los motores. Estas variables WorkSpace serán utilizadas como entrada y salida para obtener el modelo de la planta mediante el *System Identification Toolbox* de Matlab explicado en el apartado anterior que desarrolla la estrategia de control en el presente capítulo.

a) Modelado de la planta correspondiente al eje de orientación Pitch

La gráfica obtenida para modelar el comportamiento dinámico del motor que hace girar al eje pitch se puede ver en la Figura IV.40.

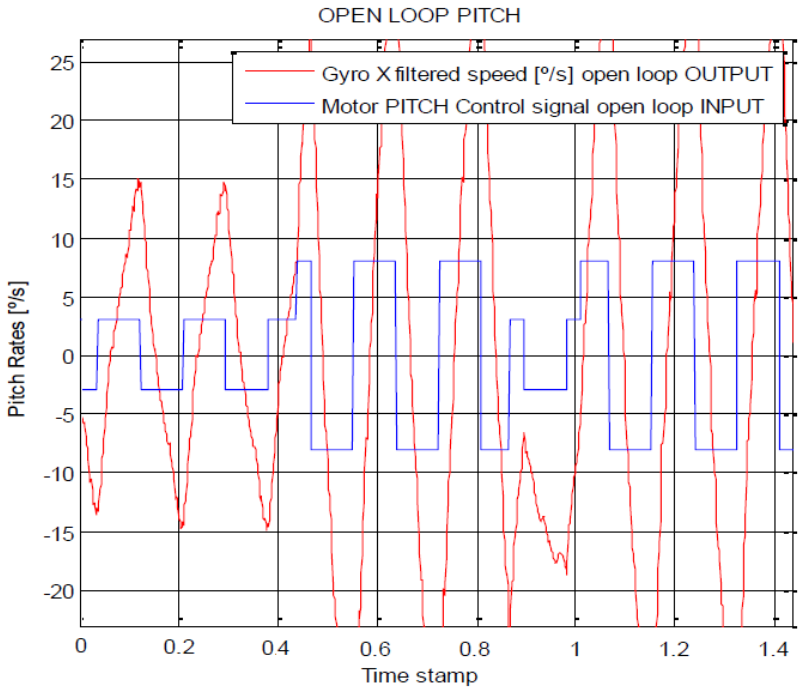

Figura IV.40 Ensayo en lazo abierto del motor de pitch.

Nada más obtener la gráfica correspondiente, se abre el GUI del *System Identification Toolbox* en el que se procede a introducir las variables de entrada y salida correspondientes. Son variables en el dominio del tiempo y por tanto en esta fase es

deseable fijar el tiempo de muestreo utilizado en el código (10 ms). Utilizando el menú desplegable *Estimate*, podemos elegir el tipo de modelo que queremos obtener. Para el caso concreto en que nos encontramos, se calculan los modelos expresados como función de transferencia en Laplace, configuramos el modelo para que sea de primer orden (un polo y ningún cero) y elegimos dominio del tiempo discreto con Ts = 0.010 s.

Dicho toolbox es capaz de enviar los resultados del modelo obtenido a la ventana de comandos de Matlab. Para el motor correspondiente al eje de orientación pitch, la función de transferencia en Laplace que podemos obtener es la siguiente:

```
tf1 =

From input "u1" to output "y1":
         (0.169 +/- 0.001044)
---------------------------------------
(1 +/- 1) - (0.9806 +/- 0.0002457) z^-1

Name: tf1
Sample time: 0.01 seconds

Parameterization:
   Number of poles: 1   Number of zeros: 0
   Number of free coefficients: 2
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Termination condition: Near (local) minimum, (norm(g) < tol).
Number of iterations: 2, Number of function evaluations: 5

Estimated using TFEST on time domain data "LAPitch".
Fit to estimation data: 93.58% (simulation focus)
FPE: 1.586, MSE: 1.58
       More information in model's "Report" property.
```

Podemos obtener un gráfico de la respuesta transitoria del modelo ante una entrada escalón, mediante el *System identification Tollbox* de Matlab. Dicho gráfico se ilustra en la Figura IV.41. La respuesta representada para el modelo de la planta se corresponde con una respuesta sobreamortiguada, con un tiempo de retardo de aproximadamente 0,4 segundos.

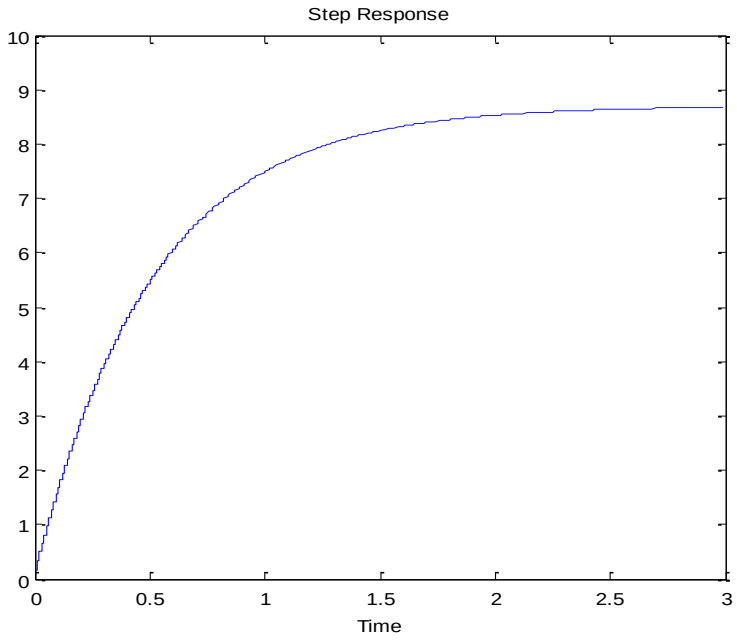

Figura IV.41 Respuesta transitoria del modelo de primer orden ante entrada escalón (Pitch)

La Figura IV.42 es una representación de los polos y ceros del sistema identificado en lazo abierto.

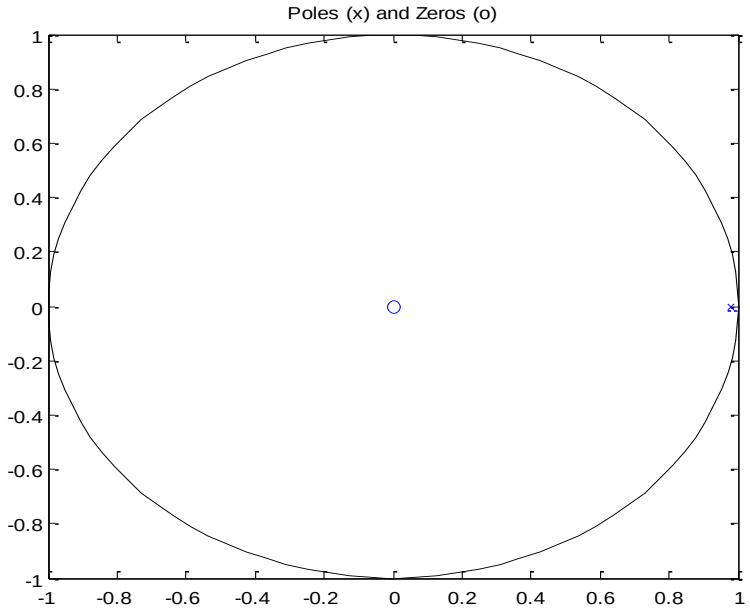

Figura IV.42 Representación de los polos y ceros del sistema de primer orden (Pitch)

b) Modelado de la planta correspondiente al eje de orientación Roll

Ahora vamos a obtener lo mismo pero para el eje de orientación roll, cuya gráfica obtenida a partir de las variables del WorkSpace de Matlab se puede ver en la Figura IV.43, donde se aprecia claramente el cambio de dirección y velocidad que toma el motor y que el filtrado de la señal del giroscopio que mide las variaciones en el eje roll sigue perfectamente el movimiento del motor.

De nuevo, en el GUI del *System Identification Toolbox* se van a introducir las variables de entrada y salida correspondientes al eje roll. Vuelven a ser variables en el dominio del tiempo y por tanto en esta fase es necesario indicar el tiempo de muestreo utilizado en el código (10 ms). En el menú desplegable *Estimate* se va a elegir el tipo de modelo que se quiere obtener. Para el caso concreto en que nos encontramos, se calcula el modelo de primer orden como función de transferencia en Laplace, se debe escoger la opción de dominio del tiempo discreto con Ts = 0.010 s.

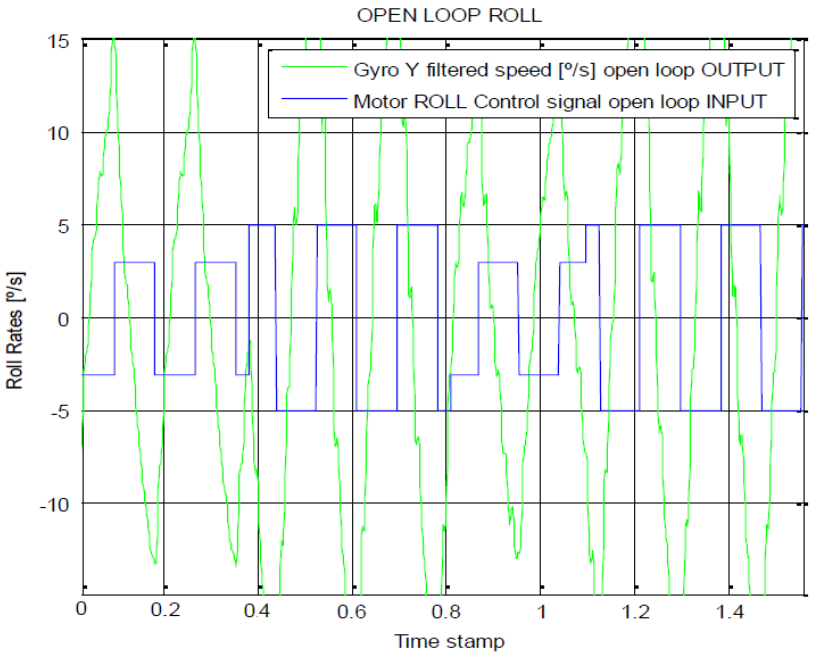

Figura IV.43 Ensayo en lazo abierto del motor de roll.

Para el modelo obtenido para el motor correspondiente al eje de orientación roll, lafunción de transferencia en Laplace obtenida con el *System Identification Toolbox* es:

```
tf2 =

From input "u1" to output "y1":
```

```
         (0.169 +/- 0.001044)
---------------------------------------
(1 +/- 1) - (0.9806 +/- 0.0002457) z^-1

Name: tf2
Sample time: 0.01 seconds

Parameterization:
   Number of poles: 1   Number of zeros: 0
   Number of free coefficients: 2
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Termination condition: Near (local) minimum, (norm(g) < tol).
Number of iterations: 2, Number of function evaluations: 5

Estimated using TFEST on time domain data "LAPitch".
Fit to estimation data: 93.58% (simulation focus)
FPE: 1.586, MSE: 1.58
       More information in model's "Report" property.
```

El gráfico que representa la respuesta transitoria del modelo identificado para el eje roll ante una entrada escalón se ilustra en la Figura IV.44. Esta gráfica corresponde a una respuesta sobreamortiguada. Y su tiempo de retardo es de aproximadamente 0.4 segundos.

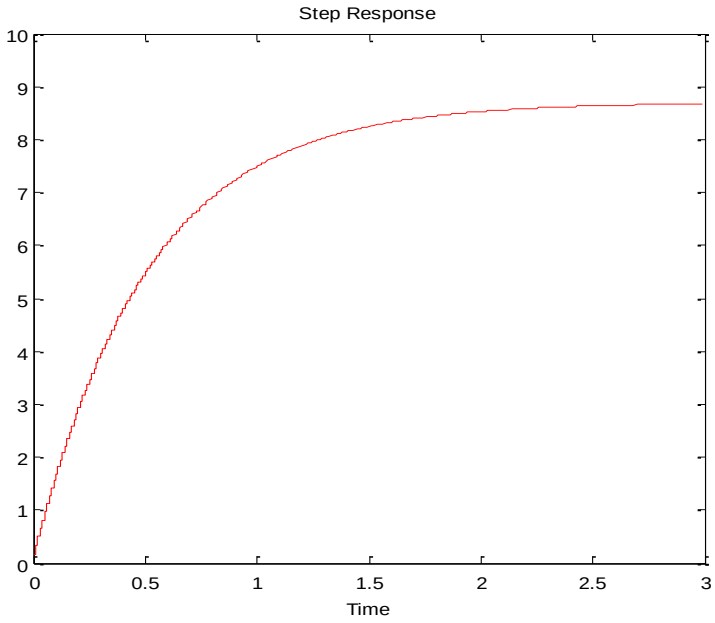

Figura IV.44 Respuesta transitoria del modelo de primer orden ante una entrada escalón (Roll)

En la Figura IV.45 se puede observar la representación de los polos y ceros del sistema de primer orden identificado en lazo abierto.

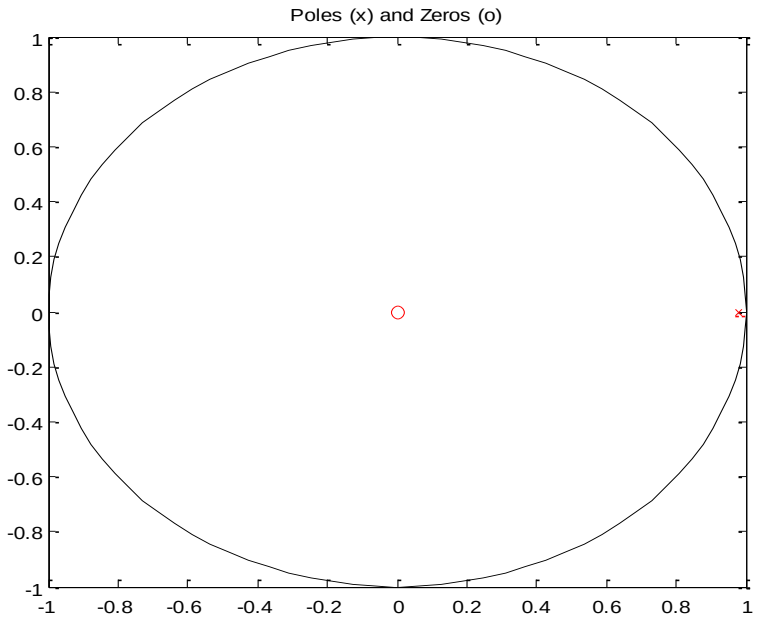

Figura IV.45 Representación de los polos y ceros del sistema de primer orden (Roll)

5.5. Simulación del controlador PID

A partir del modelo identificado de la planta en lazo abierto en el anterior apartado de este mismo capítulo, se puede crear un modelo mediante el toolbox *Simulink* de Matlab y comprobar teoricamente como se comporta la planta con un regulador PID en lazo cerrado.

Se pueden usar diferentes toolbox de Matlab para encontrar las ganancias del regulador PID que mejor se ajustan al sistema gimbal, el toolbox elegido que se usará en el presente trabajo será *PID Tuner* de *Control System Toolbox*. Con dicho fin, se escribe un archivo modelo en Matlab que construya las funciones de transferencia discretas obtenidas mediante el toolbox de identificación y posteriormente abra *PID tuner* con dichas funciones de transferencia:

```
Ts = 0.01; %10 ms

%PITCH
sysPitch = tf([0.169],[1 0.9806],Ts,'variable','z^-1')
pidtool (sysPitch)

%ROLL
sysRoll = tf([0.169],[1 0.9806],Ts,'variable','z^-1')
pidtool (sysRoll)
```

El toolbox *PID Tuner* es capaz de volcar las ganancias del regulador en el WorkSpace de Matlab, en la Tabla IV-3 se pueden observar las ganancias obtenidas.

	Pitch	***Roll***
Kp	0.0586	0.0586
Ki	11.7195	11.7195
Kd	0	0

Tabla IV-3 Ganancias de los reguladores obtenidas mediante PID Tuner.

Se puede observar que para ambos modelos, el *PID Tuner* devuelve los mismos valores para las constantes del regulador PID.

Ahora, con dichas ganancias se va a construir un modelo *Simulink* para simular la respuesta del PID diseñado ante una entrada de tipo escalón. El diagrama de bloques construido se puede ver en la Figura IV.46. En dicho diseño, se hace uso de bloques en tiempo discreto y se manda la señal de salida a una variable WorkSpace de tipo array que se graficará en Matlab mediante el siguiente código:

```
Ts = 0.01; %10 ms
time = 0:Ts:10;

sim 'simulacionPID';

subplot (1,2,1);
plot (time, stepRespPitch, 'g', time, Step, 'b')
axis([0 10 -0.1 1.1])
title('PITCH PID STEP RESPONSE');
xlabel('Time');
grid on

subplot (1,2,2);
plot (time, stepRespRoll, 'c', time, Step, 'm')
axis([0 10 -0.1 1.1])
title('ROLL PID STEP RESPONSE');
xlabel('Time');
grid on
```

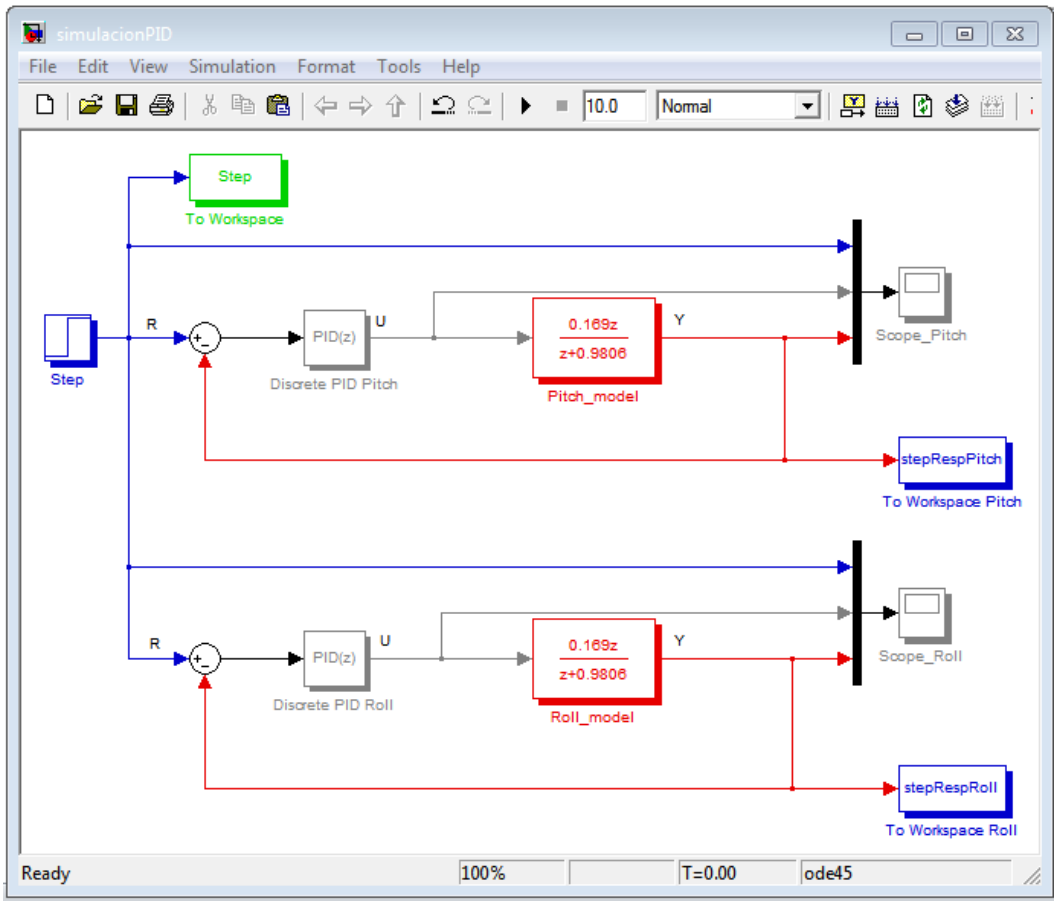

Figura IV.46 Diseño Simulink del controlador PID.

La respuesta transitoria ante entrada escalón del regulador PID se corresponde con la Figura IV.47, en la que se puede observar una respuesta sobreamortiguada para ambos ejes de orientación. El tiempo de retardo de ambas respuestas es de aproximadamente 1,68 segundos cuando llega al 50% de su valor final.

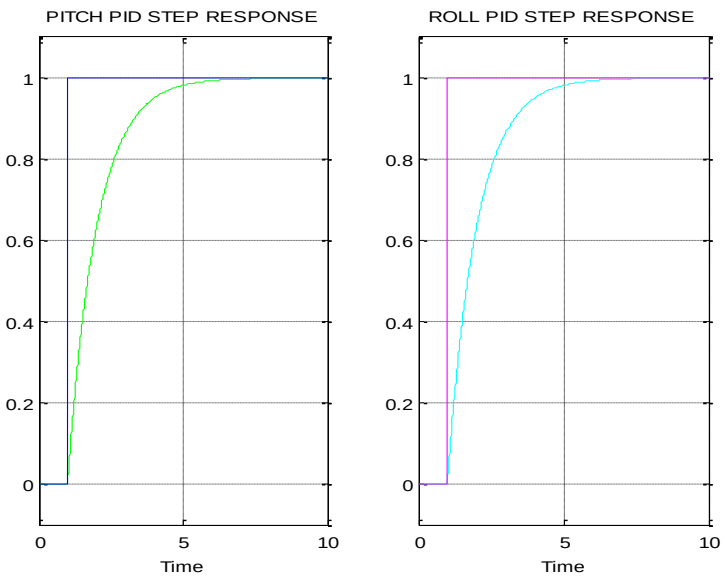

Figura IV.47 Respuesta ante entrada escalón de los PID diseñados para ambos ejes, Pitch y Roll.

5.6. Desarrollo y funcionamiento del controlador PID

El funcionamiento del regulador PID requiere del desarrollo de un código para el control del bucle en lazo cerrado. Donde dicho código utiliza como entrada al regulador la señal filtrada de la velocidad angular, como salida del regulador y entrada a la planta, la señal de control a los motores brushless, y como salida del lazo de control, la velocidad angular que los motores describen, tal y como se ilustró en la Figura IV.26.

Para implementar el regulador PID en el código fuente desarrollado para el presente trabajo, se opta por la creación de una librería en un lenguaje soportado por arduino llamada PID. Ésta librería se compone de una función que desarrolla el algoritmo del PID y de una estructura de datos que guarda las variables necesarias para su uso.

Dichas funciónes pueden ser llamadas desde cualquier parte del código fuente. La inicialización de la estructura de datos necesita los siguientes parámetros:

1. La estructura de datos `t_PID`
2. Las ganancias `Kp`, `Ki` y `Kd`
3. El tiempo de muestreo `dt`
4. Los valores máximo y mínimo de la salida del regulador `Umin` y `Umax`.

Y la función PID que calcula la acción de control de los motores requiere una llamada con los siguientes parámetros:

1. La estructura de datos `t_PID`
2. La referencia o consigna de velocidad `target`
3. La velocidad actual con la que se calculará el error, `current`

El código resultante del archivo PID.cpp es el siguiente:

```
#include "PID.h"

//Inicialización de la estructura
void init_tPID(t_PID* PID, float Kp, float Ki, float Kd, float dt, float Umin,
float Umax)
{
    PID->e_t = 0;
    PID->Laste_t = 0;
    PID->e_i = 0;
    PID->e_d = 0;
    PID->u_t = 0;
    PID->Ts = dt;
    PID->Kp = Kp; //tune
    PID->Ki = Ki;
    PID->Kd = Kd;
    PID->Umin = Umin;
    PID->Umax = Umax;
}
```

```
/*------------------Función Controlador PID---------------------*/
void PID(t_PID* PID, float target, float current)
{
    PID->e_t = target - current;
    PID->e_i += PID->e_t * PID->Ts;
    PID->e_d = (PID->e_t - PID->Laste_t) / PID->Ts;
    PID->u_t = PID->Kp * PID->e_t + PID->Ki * PID->e_i + PID->Kd * PID->e_d;
    PID->Laste_t = PID->e_t;
    if (PID->u_t < PID->Umin){
        PID->e_i -= PID->e_t * PID->Ts; // anti windup
        PID->u_t = PID->Umin;
    }else if(PID->u_t > PID->Umax){
        PID->e_i -= PID->e_t * PID->Ts;
        PID->u_t = PID->Umax;
    }
}
```

Puede suceder que la variable de control alcance y sobrepase los límites prefijados del actuador. Cuando esto pasa, el bucle realimentado permanece en su límite independientemente de la salida del proceso (valores `Umin` y `Umax`). Si se usa un controlador con acción integral, el error continuará integrándose, produciendo una deriva y por consiguiente incrementando su valor sin parar. Esto significa que el término integral puede volverse muy grande y producirse el efecto llamado "windup". Para evitar que ocurra este fenómeno se utiliza el método "anti windup" que consiste en recalcular la integral de tal forma que su nuevo valor proporciona una salida en el límite de la saturación.

La acción de control que se impondrá a los motores resulta ser la variable `u_t` que calcula la función del controlador PID. Dicha acción de control será la entrada a la planta, cuya salida será la salida realimentada del lazo de control, en º/s.

Cuando el sistema se encuentra en régimen estacionario, la medida filtrada de los giroscopios oscila en el rango de ±0.05 º/s. Esto provoca que la constante Kp del PID amplifique el error en cada instante provocando que el sistema vibre cuando se tendría que encontrar en reposo. La solución a este inconveniente es implementar una banda o *zona muerta* para que el PID no tenga en cuenta el error intrínseco a la medida del giroscopio en la banda de error o ruido. En sistemas de control, se entiende por *zona muerta* a la banda que delimita la mínima señal de entrada necesaria para que el sistema de control responda.

Para el correcto control del sistema giroestabilizador, se debe tener especial cuidado de no generar ninguna velodidad angular en el sistema en el momento inicial y poder así calcular correctamente el offset de la medida. La implementación del diseño de control teniendo en cuenta el peso de la cámara ayuda a un cálculo correcto del offset, pues encontrar el centro de gravedad del sistema para que la cámara se mantenga en posición horizontal en reposo es bastante fácil debido al diseño de la plataforma gimbal adquirida

para la cámara usada. Esto será bastante útil para mantener la posición adecuada en el instante inicial.

A partir del modelo de la planta teórico calculado en el apartado 5.4 de este mismo capítulo, se pueden usar diferentes toolbox de Matlab para encontrar las ganancias del regulador PID que mejor se ajustan al sistema gimbal (apartado 5.5). Pero en la realidad, el ajuste óptimo del controlador se realiza manualmente.

Con el fin de poder ajustar las ganancias del PID en tiempo de ejecución se desarrolla un script en Python para manejar el puerto serie que sea capaz de aunmentar o disminuir los valores de las tres constantes según se demande. Además se requiere que dicho script devuelva el valor actual de cada ganancia del PID.

Para ajustar manualmente el PID con ayuda del script desarrollado, primero se han de poner las tres constantes a cero. Seguidamente se aumenta Kp hasta que se perciban oscilaciones y se disminuye hasta un nivel ligeramente inferior al que aparecieron las oscilaciones. El siguiente paso es incrementar la ganancia integral (Ki) hasta que el proceso mejore el tiempo de establecimiento. Finalmente, si fuera necesario, se incrementa Kd hasta alcanzar un sistema críticamente amortiguado o hasta notar que el sistema vuelve rápido a la referencia tras sufrir un cambio brusco de entrada.

A la hora de ajustar el PID influye también la ganancia del filtro complementario, ya que si es próxima a cero, gran parte del valor filtrado se debe a los valores filtrados en instantes previos, y esto produce un retardo en la medida de la velocidad angular. Retardo que se percibe en la estabilización del sistema.

Con el método descrito en los anteriores párrafos, se ajusta el regulador hasta obtener unas constantes que estabilicen el sistema. El ajuste se realiza en tiempo de ejecución con el script creado. También se ajustan los valores de la zona muerta y la ganancia del filtro complementario, los valores obtenidos se muestran en la Tabla IV-4.

	DEAD_ZONE = ±0.4	**ALPHA = 0.3**
PID eje Pitch	**Kp = 0.07**	**Ki = 0.15**
PID eje Roll	**Kp = 0.02**	**Ki = 0.02**

Tabla IV-4 Valores de las ganancias del PID obtenidos en el controlador real.

Si se recuerdan los valores obtenidos para las ganancias del regulador en el apartado 5.5 mediante el *PID Tuner* de Matlab, se puede comprobar que distan bastante de

los valores que ajustan el sistema real. No obstante, los valores obtenidos teóricamente ayudan a intuir el comportamiento del sistema antes de implementarlo.

Si que es común, tanto en el cálculo teórico de las ganancias como en el ajuste manual, que la constante derivativa sea nula. Por lo que se entiende que el controlador implementado para el lazo de control es en realidad un controlador PI o controlador Proporcional Integral.

La gráfica de la Figura IV.48 corresponde al resultado de la estabilización bajo las condiciones del regulador ajustado con las ganancias obtenidas para el eje de orientación pitch. En los primeros instantes de la gráfica se puede observar como el efecto de la zona muerta es capaz de lograr que la salida de control de los motores los mantenga en reposo. También se ve con claridad como la salida del PID contrarresta los picos de velocidad que mide el giroscopio, enviando una señal con cierta simetría y signo opuesto a los motores para que corrijan el error de velocidad.

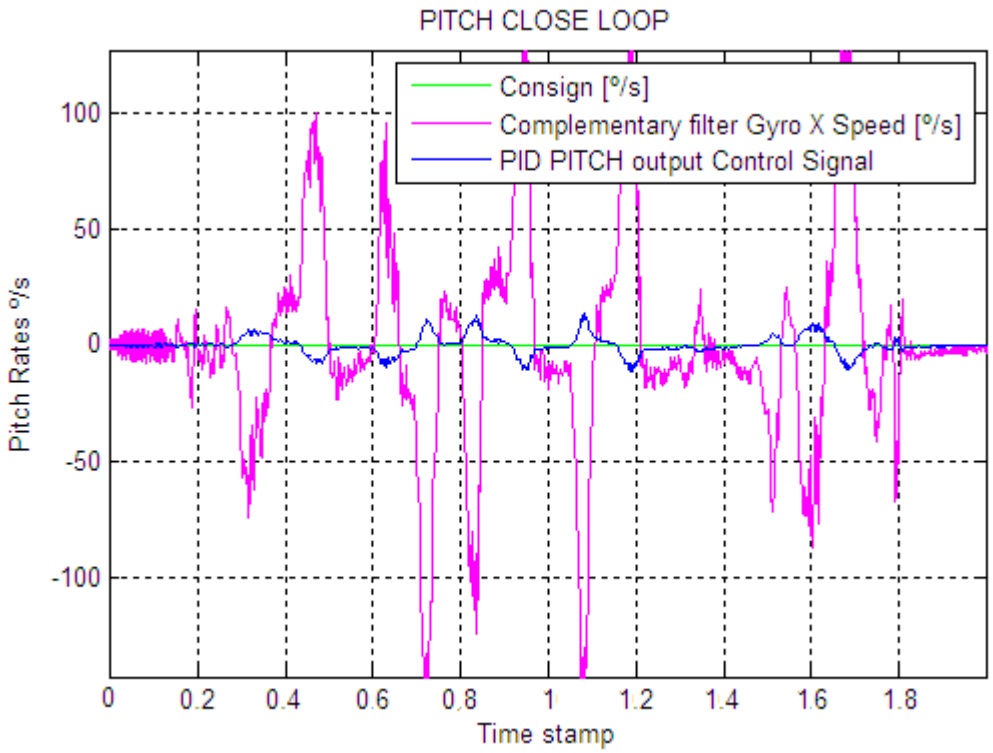

Figura IV.48 Señales del control ajustado en Lazo Cerrado

6. CAPÍTULO

Conclusiones y trabajos futuros

En conclusión, el resultado de este proyecto es un regulador PID que gobierna las acciones de una plataforma giroestabilizada.

En un primer momento se optó por la implementación de un control de posición partiendo de las señales de los acelerómetros del MPU6050, ya que este componente ofrece la posibilidad de filtrar su ruido mediante un filtro complementario que fusiona los ángulos obtenidos a partir de las aceleraciones con la integración discreta de las velocidades porporcionadas por sus giroscopios. Se ha realizado una comparativa mediante la herramienta Matlab de la medida que proporcionan los acelerómetros y los giroscopios y debido al elevado ruido que acompaña a la medida de los acelerómetros y al mal filtrado que produce la integración discreta de la velocidad, se ha comprobado que para nuestra aplicación en concreto, obtendremos mayor precisión usando unicamente las medidas filtradas de los giroscopios, y por tanto, realizar un control electrónico de velocidad es la opción más indicada.

La arquitectura de control desarrollada en el trabajo ha requerido el uso de casi toda la capacidad de memoria del procesador, es por eso que han aparecido problemas derivados de la falta de almacenamiento de programa para desarrollar nuevas soluciones. También para esta solución implementada se necesitan más requisitos hardware como dos timers extra para manejar las interrupciones temporizadas que darían paso a los motores y sensores de efecto hall para mover con mayor precisión los motores.

Debido a la falta de los timers mencionados en el anterior párrafo, los tres timers

usados que configuran las salidas PWM a la frecuencia determinada, provocan que ciertas funciones de Arduino dejen de funcionar en los periodos de tiempo correctos. Esto es un problema cuando se trata de realizar un control electrónico donde el tiempo de muestreo es un parámetro realmente importante en la aplicación. Para solucionar ésto, y que el bucle se ejecute en un periodo de muestreo real, se ha implementado una función que produce un retardo de un tiempo conocido mediante funciones del procesador que no se ven afectadas por el desajuste que ocasionan los timers. Así se puede ejecutar cada bucle del programa en un tiempo de muestreo exacto.

Se puede concluir diciendo que una fase de diseño acertada desde el principio de cualquier proyecto marcará el curso de las fases sucesivas y su correcto desarrollo.

Como futuros trabajos se pueden destacar los siguientes:

- Desarrollo del código fuente del controlador en lazo cerrado con interrupciones temporizadas para la activación de motores y así mejorar los tiempos del sistema.
- Desarrollo de un filtro de Kalman para el uso de giroscopios.
- Desarrollo de una interfaz gráfica para usuarios con la que se puedan cambiar las constantes del regulador PID, las ganancias de filtrado y el tiempo de muestreo.
- Comunicación remota con plataforma Android (NFC + Wi-Fi, GSM, Ethernet….)
- Implementación del sistema en una plataforma gimbal de tres ejes.

Por supuesto, todos estos trabajos futuros requieren de una controladora con un procesador más potente que el ATmega 328P de 8 bits usado en el presente trabajo.

V. CONTENIDO DEL CD

Se incluye un CD que contiene toda la documentación necesaria para la realización de este proyecto. La siguiente enumeración muestra los distintos contenidos:

- Datasheet del sensor MPU6050 utilizado en el diseño, así como de los componentes utilizados en el inversor trifásico AO4606.
- Datasheet del ATmega328P.
- Drivers para Windows CP210 VCP de Silicon Labs.
- Códigos fuente de la programación del microcontrolador implementados en el desarrollador de Arduino.
- Scripts generados para el intercambio de datos por el puerto serie del procesador escritos en Python.
- Archivos generados para las pruebas realizadas en Matlab.

Además contiene la **memoria en formato PDF**.

VI. BIBLIOGRAFÍA

[1] Guía "Spining BLDC (Gimbal) motors at super slow speeds with Arduino and L6234". Fecha de última consulta: Diciembre de 2015. http://www.berryjam.eu/2015/04/driving-bldc-gimbals-at-super-slow-speeds-with-arduino/

[2] Guía de desarrollo de aplicaciones con acelerómetros de tres ejes "Tilt Sensing Using a Three-Axis Accelerometer" de Mark Pedley. Fecha de última consulta: Diciembre de 2015. https://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf

[3] Manual "Tutorial de Arduino y MPU-6050". Fecha de última consulta: Diciembre de 2015. http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

[4] Guía "The Balance Filter" por Shane Colton. Fecha de última consulta: Diciembre de 2015. https://b94be14129454da9cf7f056f5f8b89a9b17da0be.googledrive.com/host/0B0ZbiLZrqVa6Y2d3UjFVWDhNZms/filter.pdf

[5] Teoría del Filtro Complementario "Reading a IMU Without Kalman: The Complementary Filter" por Pieter Jan. Fecha de última consulta: Diciembre de 2015. http://www.pieter-jan.com/node/11

[6] Teoría del controlador PID "PID Controller". Fecha de última consulta: Diciembre de 2015. https://en.wikipedia.org/wiki/PID_controller

[7] Datasheet ATmega 328P de ATMEL. Fecha de última consulta: Diciembre de 2015. http://www.atmel.com/images/doc8161.pdf

[8] Datasheet MPU-6050 InvenSense. Fecha de última consulta: Diciembre de 2015. http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

[9] Datasheet AO4606 Alpha & Omega Semiconductor. Fecha de última consulta: diciembre de 2015. http://pdf1.alldatasheet.es/datasheet-pdf/view/88643/AOSMD/AO4606.html

[10] Proyecto relacionado (robot de balanceo) “Balancing robot for dummies”. Fecha de última consulta: Diciembre de 2015. http://www.x-firm.com/?page_id=148

[11] Proyecto relacionado “Brushless Gimbal with Arduino” de Andreas Biekert and Jonah Grubb. Fecha de última consulta: Diciembre de 2015. http://www.instructables.com/id/Brushless-Gimbal-with-Arduino/

[12] Repositorio de firmware para tarjeta controladora BaseCam. Fecha de última consulta: Diciembre de 2015. https://www.basecamelectronics.com/downloads/8bit/

[13] Fabricante de controladoras de código abierto Martinez Brushless Gimbal. Fecha de última consulta: Diciembre de 2015. http://www.itsqv.com/QVW/index.php?title=How_To_-_Martinez_Brushless_Gimbal_Controller_Set-Up

[14] Arduino y su repositorio de código abierto. Fecha de última consulta: Diciembre de 2015. www.arduino.cc

[15] Arduino PWM repositorios. Fecha de última consulta: Diciembre de 2015. https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM

[16] Repositorio de los drivers CP210 de Silicon Labs: https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

[17] Librería MPU6050, por Jeff Rowberg. Fecha de última consulta: Diciembre de 2015. http://www.i2cdevlib.com/devices/mpu6050#source

[18] Universidad de Alcalá de Henares: Asignaturas de Electrónica de Potencia, Sistemas Robotizados y Técnicas de Control.