E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Implementación de un dron cuadricóptero con Arduino

Grado en Ingeniería Eléctrica y Electrónica

Trabajo Fin de Grado

Jose Etxeberria Mendez

Javier Goicoechea Fernández

Pamplona, 22 al 30 de Junio de 2015

1 Agradecimientos, dedicatorias o citas

Especiales agradecimientos a:

- Mis padres por financiarme los estudios que me han permitido llegar hasta aquí.

- Dr. Javier Goicoechea Fernández, tutor del proyecto, por ofrecerme su apoyo continuo durante la realización del proyecto.

- La comunidad Arduino por su amplio desarrollo y sus elaborados tutoriales explicativos.

Especiales dedicatorias a:

- La comunidad de jóvenes genios que se sienten intrigados por el mundo electrónico.

- La comunidad Arduino para aumentar su repositorio de proyectos disponibles.

- Tributo a Pantera.

Citas:

- "*Aprender sin pensar es inútil, pensar sin aprender es peligroso*" - Confucio

2 Resumen y lista de palabras clave

2.1 Resumen

El presente proyecto consiste en la implementación de un dron mediante la plataforma de programación Arduino. El dron en cuestión se trata de un cuadricóptero, "*quadcopter*" o simplemente "*drone*", destinado a aplicaciones civiles genéricas, como podría ser filmación de vídeo.

El desarrollo del "*drone*" ha constado de la elaboración software y hardware. Por un lado el código software almacenado en Arduino que permita el estudio de estabilidad del dron. Por otro lado el diseño de una PCB donde se incluyen los dispositivos y conexiones necesarias para el correcto funcionamiento del dron y posibles incorporaciones futuras.

El dron es controlado mediante comunicación Zigbee entre Arduino y un ordenador, utilizando la aplicación XCTU.

2.2 Abstract

In this paper, a hardware drone design that employs Arduino platform is introduced. That drone is a quadcopter. Its purpose is that in a future it could perform generic civil applications such as video recording.

The aim of this project is composed by hardware and software development. On one hand the software code stored on Arduino that enables the drone stabilization study. On the other hand the printed circuit board design that includes all connections and devices required in order to achieve the correct performance and enable future improvements.

This drone is controlled by Zigbee communication between Arduino and a computer, using XCTU application.

2.3 Laburpena

Proiektu hau Arduino programazio plataformaren bitartez "dron" baten sorkuntzan datza. "Dron"-a kuadrikoptero bat da. Erabilera zibil orokorretarako zuzenduta dago, bideo baten filmaketa izan daitekeen bezala.

"Dron"-aren garapena software nahiz hardware-aren lanketan datza. Alde batetik, "Dron"-aren egonkortasun azterketa ahalbidetuko duen Arduinon gordetako software kodea. Bestetik, zirkuitu inprimatuaren plaka baten diseinua, non "Dron"-aren funtzionamendu egokirako beharrezkoak diren gailu eta konexioak eta etorkizunean eman daitezkeen eransketak aurkitzen diren.

"Dron"-a Arduino eta ordenagailu baten arteko Zigbee komunikazio baten bidez kontrolaturik dago, XCTU aplikazioaz baliatuta.

2.4 Key word list

- Drone
- Quadcopter
- Arduino
- Zigbee
- Hardware Design

2.5 Lista de palabras clave

- Dron
- Cuadricóptero
- Arduino
- Zigbee
- Diseño hardware

2.6 Hitz gakoak zerrenda

- Dron
- Koadrikopteroa
- Arduino
- Zigbee
- Hardware Diseinua

3 Cuerpo del documento

3.1 Introducción

El objetivo del presente proyecto consiste en la realización de la programación de un dron, diseño de una placa de circuito impreso e implementación de los distintos periféricos para facilitar el estudio del equilibrio del mismo de forma automática frente a perturbaciones del viento.

En este proyecto se realiza principalmente el trabajo de programación del software de control, desde la obtención de parámetros mediante sensores hasta la modificación del comportamiento de los actuadores en función de ellos. Por otro lado la segunda gran labor en el proyecto consiste en el diseño y montaje de un PCB que incluya los elementos necesarios para el objetivo presentado y futuras expansiones.

Los elementos que forman nuestro cuadricóptero son los siguientes: por el lado mecánico están el chasis y las 4 hélices, y por el lado eléctrico tenemos el microcontrolador Arduino con su correspondiente batería, el sensor IMU (MPU 6050), regulador de tensión, módulo de comunicación Xbee, los reguladores ESC y los 4 motores brushless DC.

En el código de programación se incluye el acondicionamiento de señal en función del sensor, el filtro complementario para promediar los datos del acelerómetro y giróscopo, el sistema de control PID, la generación de la acción de control, la traducción a un movimiento en los actuadores, un pequeño circuito que mide la carga de la batería y el control de parámetros del dron a través de comunicación XBee.

3.2 Justificación y objetivos

La razón de elaboración del proyecto procede de un elevado interés por la electrónica y los elementos interactivos cuyos resultados son muy visuales y entretenidos. Además de un trabajo de fin de grado supone un reto personal el demostrar ser capaz de implementar un dispositivo con el que se pueda interactuar. A su vez, supone una indagación en las últimas modas tecnológicas y en un mercado que está comenzando a ser explotado y continuará con un elevado desarrollo.

Existen dos tipologías diferenciadas para los drones, los hay con estructuras similares a los aviones y los hay con tipologías similares a los helicópteros, esta última será la estructura elegida para este proyecto, un multicóptero, concretamente con 4 hélices, es decir, un cuadricóptero o "*quadcopter*" en inglés.

El proyecto se va a realizar dejando abierta la integración de distintas funcionalidades en un futuro, por lo que el desarrollo no se va a encasillar en una aplicación concreta. En el desarrollo del PCB, en el que se incluye toda la electrónica, se ha previsto la inclusión en el futuro de dos elementos más para ampliar las capacidades del dispositivo, un sensor de presión para controlar la altura y un compás para controlar la orientación horizontal.

El objetivo final es desarrollar un dispositivo interactivo con el cual los alumnos puedan realizar el aprendizaje de una forma más dinámica y potencialmente atractiva. Se propone el uso del dispositivo para el estudio del comportamiento del control PID.

3.3 Contexto tecnológico

Es triste que la ingeniería bélica marque el ritmo del desarrollo tecnológico pero el dron es uno de muchos ejemplos en los que las guerras han promovido el desarrollo de cierta tecnología con fines destructivos y luego ha sido adaptada para usos civiles. [12]

Las aplicaciones civiles a las que son destinados los drones van aumentando con rapidez en los últimos años al permitir labores que por riesgo o dificultad de accesibilidad no son rentables de realizar de otra manera, las principales son: grabación de películas, documentales y eventos, grabación de deportes extremos o zonas interiores, prevención de incendios forestales, vigilancia de fronteras, control de infraestructuras industriales, labores de vigilancia en zonas catastróficas, lucha contra el terrorismo y el narcotráfico, entretenimiento, purificación del aire, investigación de volcanes, búsqueda de supervivientes en zonas azotadas por catástrofes naturales, lucha contra grafiteros, localización de bancos de peces así como de cazadores furtivos, entrega de productos a domicilio, etc. [8] [9]

La principal ventaja que suponen los drones reside en la inexistencia de personas a bordo lo que elimina el riesgo de muerte de los pilotos y permite realizar funciones que no serían posibles con aeronaves tripuladas como investigación en zonas de alta toxicidad química y radiológica.

Por otro lado están otros aspectos como la posibilidad de automatización del aparato, siendo capaz de actuar por sí mismo con una integración de inteligencia artificial e incluso coordinarse con otros drones.

Para aplicaciones como grabación de video, un “drone” de poco tamaño puede ser suficiente, encontrándose modelos sencillos en el mercado rondando los 500€.

A pesar de las grandes posibilidades que nos brindan los drones, presentan también una serie de desventajas técnicas, económicas y éticas.

Las desventajas técnicas tienen en cuenta que este elemento no deja de ser una máquina, controlada a distancia a través de comunicación remota y que requiere una fuente de energía que se va consumiendo, pueden producirse ciertos fallos de comunicación, quedarse sin combustible, reacciones lentas a las indicaciones enviadas, pérdida de comunicación por lejanía o zonas de poca cobertura, así como ocurrió en Irak y Afganistán que los drones fueron hackeados. [11]

Las desventajas económicas influyen principalmente en las aplicaciones militares, en las que estos vehículos aéreos son extremadamente avanzados y extremadamente caros llegando a 50 millones de dólares entre aparato y comunicación del MQ-B Fire Scout, el más grande de su estilo, diseñado para uso experimental en portaaviones.

Las desventajas éticas son posiblemente las más importantes relacionadas con estos aparatos, como suele ocurrir al aparecer nuevas tecnologías siempre surgen muchas preguntas sobre su dirección de desarrollo.

Se desarrollan ya drones de forma experimental con cierta integración de inteligencia artificial que les permite elegir por sí mismos los objetivos que deben ser atacados, tanto este aspecto como el hecho de estar a gran distancia del lugar del conflicto elimina preocupaciones morales y responsabilidad al mando que ordene la operación, difundiendo una insensibilidad inhumana. [12]

Por otro aspecto la comercialización discriminada de estos aparatos sin regulación, permite que cualquier aficionado disponga de estas unidades con las que tomar fotografías o grabaciones a personas constituyendo una grave amenaza para la privacidad personal. En la otra cara de la moneda la regulación por parte de sistemas de gobierno aumenta el sofoco propio del estado policial extendido por medio planeta.

3.4 Cuerpo del trabajo

3.4.1 Estructura

El primer aspecto que se debe decidir es la estructura que vamos a utilizar para nuestro dispositivo, en función de nuestra decisión se desarrollarán por un camino u otro el código de programación y el diseño del PCB.

Habitualmente se utilizan distintas estructuras para el chasis, en "+", en "X", en "Y" o en "H". Las más usadas son en "X" o en "+" ya que de esta forma todos los brazos son iguales y el control se sitúa en el centro de gravedad de la estructura, dentro de las modalidades en "X" las más utilizadas son las de 4, 6 y 8 brazos. [10]

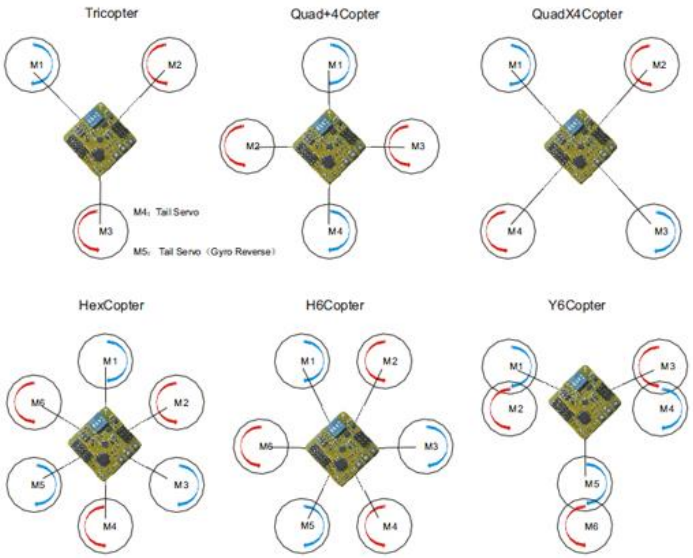


Ilustración 1. Tipologías de drones más habituales y la disposición de sus motores. www.aeromodelismofacil.com/cuadricoptero_primero.htm

La estructura en forma "+" es más sencilla de controlar ya que para el desplazamiento solo es necesario actuar sobre 2 motores dejando los del eje perpendicular constantes, [14] en el modelo "X" estos movimientos son más complicados de controlar, no obstante el tutor Javier Goicoechea ya dispone de una estructura montada en modelo "X" por lo que esa es la estructura utilizada.

3.4.2 Arduino

Para la realización de este proyecto se ha decidido trabajar con la plataforma Arduino ya que se ha trabajado con anterioridad con esta plataforma en asignaturas durante el grado. Se ha elegido este sistema aparte de por ser ya conocido, por la enorme comunidad de internautas que colaboran con sus aportaciones a la hora de proponer soluciones en el foro de Arduino.

Para dar unas pequeñas pinceladas sobre Arduino, se trata de una plataforma de código abierto basado en C y Processing de software gratuito. Arduino se limita a fabricar las placas pero incluso pone a tu disposición los esquemáticos al más mínimo detalle por si algún valiente decide hacerse su propia placa Arduino. Incluye en la mayoría de ejemplares un procesador ATmega en el que se carga la programación deseada.

Arduino como tal no da mucha información sobre cómo utilizar sus productos pero la comunidad que lo rodea esclarece cualquier duda.

Entre qué modelo de Arduino elegir, en la universidad se disponía de varios ejemplares de Arduino UNO y Arduino Nano, se analizaron estos productos y se consideraron una buena elección ya que Arduino UNO tiene muy buenas consideraciones siendo más limitado que su hermano mayor Arduino Leonardo pero es suficiente para la función que se quiere realizar.

Las características básicas requeridas son las siguientes:

- Comunicación serie para intercambiar datos con un módulo Xbee (TX y RX)
- Comunicación I^2C para los distintos sensores, en nuestro caso se utiliza únicamente el MPU 6050 pero el PCB se prepara para la inclusión en el futuro de otros sensores.
- 4 Salidas Digitales para los actuadores
- 1 Entrada Analógica para medir el nivel de batería disponible
- Pin de alimentación para periféricos a 5V y 3.3V.

Tanto el Arduino UNO como el Arduino Nano disponen de un controlador ATMega328 cuyas características y las respectivas placas sobre las que están montadas satisfacen los requisitos.

Ilustración 2 Arduino UNO
http://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg

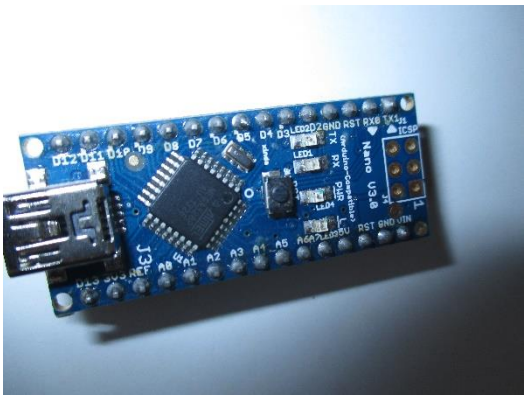


Ilustración 3 Arduino NANO. Fotografía propia.

El Arduino UNO se ha utilizado para el análisis del sistema en laboratorio, la programación de los distintos apartados y su testeo, pero para la puesta en funcionamiento definitiva se utiliza el Arduino Nano ya que dispone de las características que necesitamos con un consumo mínimo de energía, diseñado precisamente para aplicaciones como la que nos ocupa.

3.4.3 Inertial Measurement Unit

Un detalle importante que ayudó a tomar la decisión de utilizar Arduino es la disposición de un bus I2C (Inter-Integrated Circuit) el cual pone en comunicación distintos circuitos integrados, esto permite incluir un sensor IMU con el que mediremos la inclinación en cada eje y en definitiva la orientación de nuestro dron y así cerrar el lazo de control.

Este sensor, "IMU Inertial Measurement Unit", está formado por 3 acelerómetros y 3 giróscopos teniendo la posibilidad de añadirle a parte un magnetómetro de tres ejes que lo complemente para medir el ángulo de giro en el eje Z, a continuación se da una explicación del funcionamiento de cada elemento. El circuito integrado que se utiliza es concretamente un IMU MPU-6050 de Invesense montado sobre una breakout board GY-521 de Fritzing.

Ilustración 4 MPU6050 sobre Breakout Board GY-521 http://playground.arduino.cc/Main/MPU-6050

Se utiliza este sensor por dos razones principalmente. La primera de ellas su precio, el más competitivo en el mercado en los últimos años y que por su extendido uso está verificada su fiabilidad. Este sensor en breakout boards de SparkFun eleva su precio pero del modelo elegido se encuentran ejemplares por incluso menos de 10€.

La segunda razón reside en que se pueda ampliar el sensor con la conexión de un magnetómetro. InveSense ofrece también un sensor que integra el MPU 6050 más un compás de tres ejes, el MPU9150 pero como en este proyecto no se va a implementar, se utiliza este dispositivo para facilitar ampliaciones futuras.

3.4.3.1 Acelerómetro

Los acelerómetros miden el efecto de la fuerza de gravedad sobre ellos mismos, al disponerse 3 acelerómetros formando 3 ejes ortogonales, el aumento de percepción de gravedad en uno de los ejes indica que este se está poniendo en vertical, incidiendo toda la fuerza de gravedad sobre él.

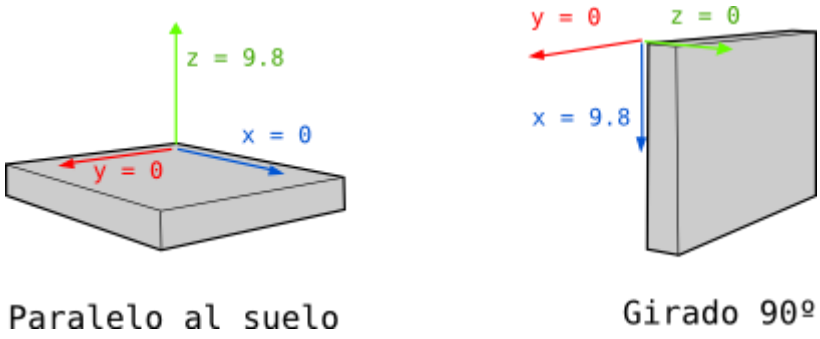


Ilustración 5. Percepción del acelerómetro en función de su posición respecto al campo gravitatorio. http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/

Por hacer un pequeño inciso sobre el acelerómetro, se usa este sensor en su versión MEMS.

La tecnología MEMS, Micro Electro Mecanical Systems, ha logrado reducir a tamaños ridículamente pequeños la fabricación de sistemas que combinan elementos mecánicos y electrónicos.

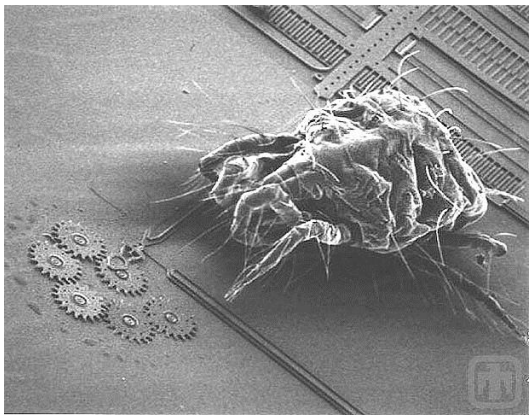


Ilustración 6
http://www.sandia.gov/mstc/mems_info/movie_gallery.html

Uno de los mayores logros de esta tecnología es que se logra construir micro estructuras móviles sobre un sustrato. Esto permite la creación de sistemas eléctricos y mecánicos muy complejos.

Se incluye una imagen de un ácaro sobre un MEMS para visualizar las dimensiones con las que se trata.

Utiliza un material piezo eléctrico de estado casi sólido, comúnmente cuarzo, cargado eléctricamente el cual tiene sobre sí una masa conocida y en consecuencia se comprime en función de la gravedad que afecte sobre él. Esta presión produce una alteración en la alineación de los electrones y protones en el material piezo eléctrico lo que provoca una acumulación de cargas opuestas en ambas superficies del material percibiéndose así una variación en la carga eléctrica que resulta proporcional al esfuerzo experimentado por el material, el cual es proporcional a la aceleración de la gravedad por la segunda ley de Newton (F=m·a). Esta variación correctamente tratada ayuda a tomar la medida de la cantidad de inclinación que tiene un eje, con una serie de sencillos cálculos trigonométricos se determina el giro del eje. [12]

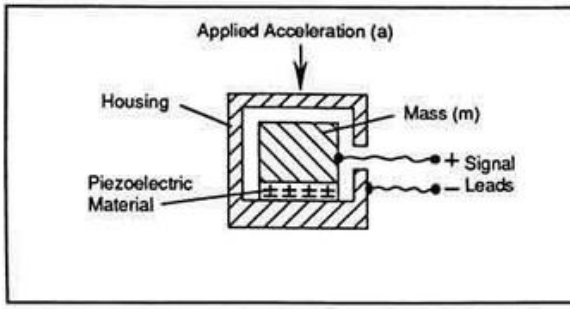


Ilustración 7. Esquema de construcción de un acelerómetro MEMS.
www.pcb.com/techsupport/tech_accel.aspx

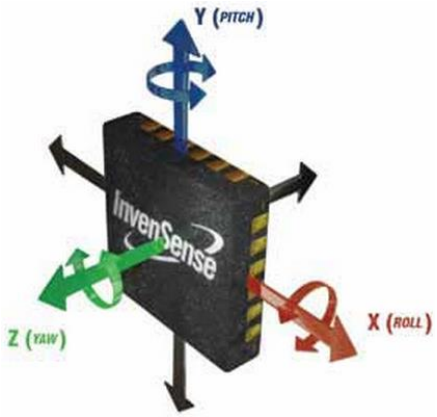


Ilustración 8. Relación de ejes y giros del IMU.
zillion.nl/articles/invesense.html

El valor que el acelerómetro envía a Arduino es de un rango de números enteros de [-16384, 16384] y la sensibilidad se ajusta mediante programación escribiendo en los registros del controlador. En nuestro caso se ha establecido la sensibilidad en 2g de forma que logramos una precisión de 16384 LSB/g o en otras palabras 61 µg/LSB.

El ángulo de giro de un eje, vendrá definido por una combinación del peso percibido en cada uno de los 3 ejes. A continuación se va a explicar el procedimiento de cálculo, situación para la que ya se consideran los offsets eliminados; se analiza el procedimiento más adelante.

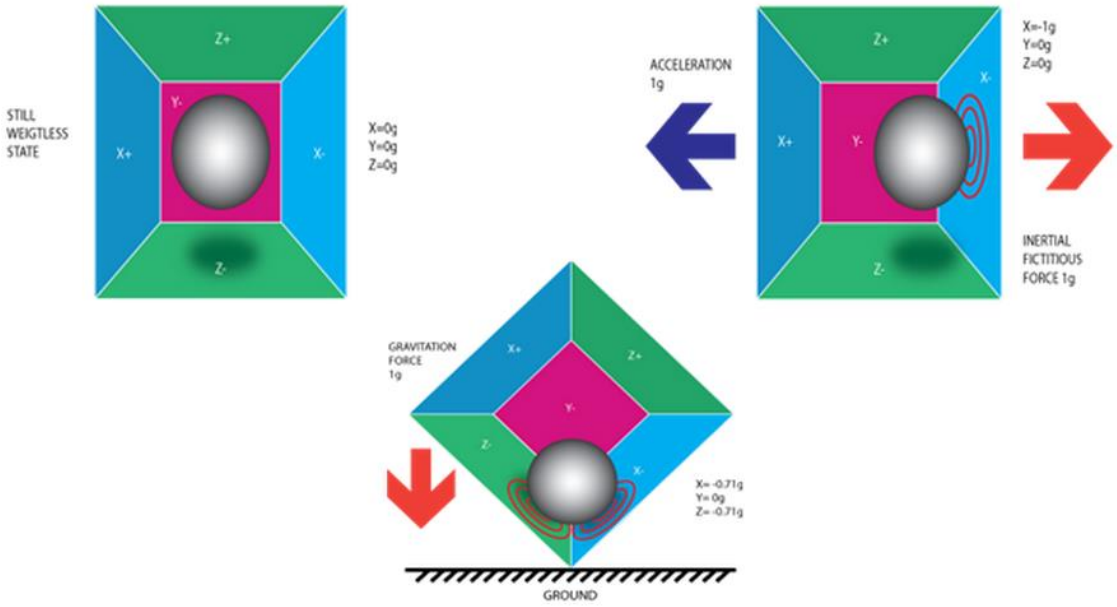
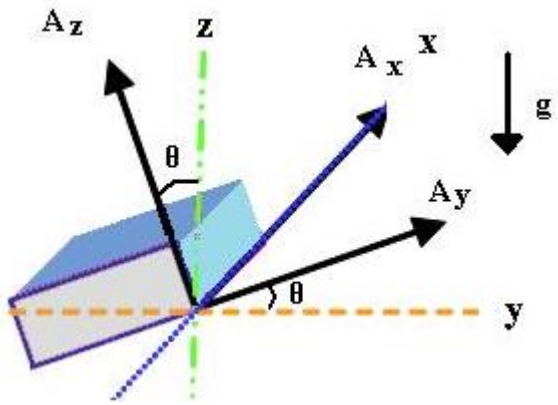


Ilustración 9. Representación del comportamiento del acelerómetro MEMS.
http://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial

Imaginemos un cubo cuya base XY_{CUBO} reposa sobre el plano XY_{REF} e imaginemos un sistema de coordenadas XYZ_{REF} cuyo eje X_{REF} coincide con una arista del cubo XYZ_{CUBO}.

Ahora supongamos que giramos el cubo un ángulo θ_x con el eje X como centro de revolución, al terminar el giro la base del cubo XY_{CUBO} forma un ángulo θ_x con el plano XY_{REF}. La lectura del acelerómetro X, sin offset, sería prácticamente 0 por lo tanto el cálculo del ángulo de giro sería:



$$\theta_x = \tan^{-1}\left(\frac{lectura\ acelerometro\ y}{lectura\ acelerometro\ z}\right).$$

Ilustración 10. Representación del giro de un cuerpo respecto al eje X.
www.instructables.com/files/orig/FS3/I6PI/FXP6OCO7/FS3I6PIFXP6OCO7.jpg

Así como hemos inclinado el plano de la base del cubo (XY_{CUBO}) un ángulo θ_x respecto al plano de referencia (XY_{REF}), el plano XZ_{CUBO} también se ha desplazado un ángulo θ_x respecto al plano de referencia (XZ_{REF}).

En el momento en el que se haga un segundo giro, sobre el eje "Y_{CUBO}" por ejemplo, la lectura del acelerómetro "Y" va a seguir siendo la misma ya que esto viene determinado por la inclinación del eje "Y" respecto al plano XY_{REF} y esta no se modifica.

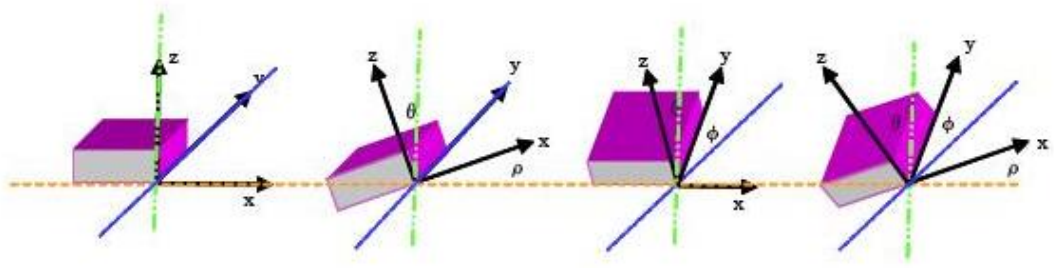


Ilustración 11. Representación del giro de un cuerpo respecto a los tres ejes de coordenadas. http://husstechlabs.com/projects/atb1/using-the-accelerometer/

Los ejes Z_{CUBO} y X_{CUBO} se desplazarán a lo largo del plano XZ_{CUBO}, recordemos que este se encuentra a θ_x grados de XZ_{REF}, la lectura del acelerómetro "Z" va a disminuir al ritmo que aumenta la del acelerómetro "X", sin embargo la suma de los módulos de los acelerómetros "Z" y "X" siempre va a ser constante e igual al valor que transmitía el acelerómetro "Z" al completar el primer giro, ya que el valor en el eje "X" era 0.

En ciertas explicaciones se ha observado cómo se ha hecho una aproximación para estos cálculos y se ha reducido a considerar que el movimiento se hace tan solo en 2 ejes, ya que la aproximación es bastante acertada quedando la expresión como se ha mostrado arriba junto a la ilustración 10. No obstante para una mayor precisión se han tenido en cuenta todos los ejes en el cálculo de cada medida, quedando las expresiones como se muestran debajo.

$$\boldsymbol{\theta_x = \tan^{-1} \frac{A_y}{\sqrt{A_x^2 + A_z^2}}} \qquad \textit{Ecuación 1}$$

$$\boldsymbol{\theta_y = \tan^{-1} \frac{A_x}{\sqrt{A_y^2 + A_z^2}}} \qquad \textit{Ecuación 2}$$

Haciendo uso tan solo del acelerómetro no se puede determinar el giro en el eje Z ya que este giro se producirá con unos valores de X e Y que si el cuadricóptero está perfectamente en horizontal serían 0, y no podríamos extraer ninguna información útil. Para hallar la orientación del eje Z se debe incluir un compás o magnetómetro que tiene el funcionamiento básico de una brújula.

Para el escalado del valor del acelerómetro tan solo hay que tener en cuenta que Arduino saca radianes de la función arcotangente habiendo que multiplicar el resultado por 180/PI para obtener el resultado en ángulos.

A la hora de aplicar un movimiento al dron sin modificar su inclinación, el acelerómetro detecta una modificación en el esfuerzo que percibe por efecto de la gravedad sobre la masa conocida y considera que ha realizado un cambio de inclinación, esto se solucionará con el adecuado uso de un filtro complementario.

3.4.3.2 Giróscopo

El giróscopo consiste en un aparato que mide la orientación de un objeto en el espacio, fue desarrollado por Foucault en 1852 pero ya habían ideas parecidas desde 1813, con el alemán Johann Bohnenberger a quien se le atribuye el descubrimiento del efecto giroscópico. [13]

En aquellas fechas estos aparatos eran puramente mecánicos, grandes y pesados lo que reducía sus aplicaciones al ámbito militar. En los últimos años con el desarrollo de la electrónica y mecánica se han desarrollado giróscopos MEMS, eliminando el problema del tamaño y el peso. [14]

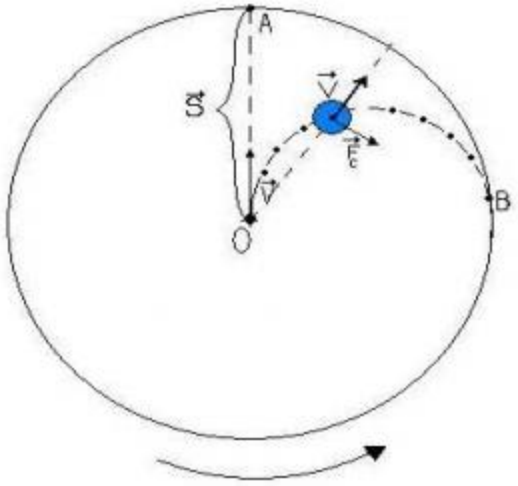
Los giróscopos MEMS se fundamentan en el *efecto coriolis* para tomar sus medidas.

El efecto coriolis está presente en todos los cuerpos de rotación, incluida la propia tierra. Este efecto consiste en la distinta percepción del movimiento al observar desde un sistema de referencia rotatorio (no inercial) a un objeto que se encuentra fuera de este sistema de referencia.

La fuerza de coriolis como tal no existe pero es necesario incluirla si deseamos explicar el funcionamiento del sistema con las leyes de Newton. Estrictamente podría decirse que los sistemas inerciales como tal no existen, ya que la propia Tierra es un sistema no inercial al estar girando sobre sí misma, alrededor del Sol, éste respecto a la Vía Láctea, etc. Sin embargo se aceptan ciertas consideraciones para simplificar los cálculos y en la mayoría de ocasiones la tierra se considera un sistema inercial.

Ejemplo 1 - Un objeto situado en el origen de un disco giratorio horizontal sin rozamiento, observa que debe ir recto hacía el norte para llegar al punto A del disco, no obstante al estar el disco girando en sentido anti horario, al realizar el objeto un movimiento rectilíneo sin rozamiento dirección norte, el destino en el sistema de referencia del disco será el punto B, ya que mientras el objeto se desplaza hacia el norte en línea recta, la plataforma sobre la que se desplaza está girando.

Desde el punto de vista del objeto o desde el de un observador externo estático, el objeto se estará moviendo en línea recta, pero para un observador que se encuentre en el punto A orientación sur girando con el disco el objeto se alejará de él por su izquierda sin que aparentemente haya una fuerza que provoque este movimiento.



-·-·- Recorrido realizado por el objeto para un observador que se encuentra en el sistema de referencia rotatorio.

--- Recorrido realizado por el objeto para un observador externo al disco

Ilustración 12. Representación del efecto Coriolis.
[http://www.gte.us.es/ASIGN/SEA/pracs/Eval_Gir_con_datasheet.pdf]

Como se puede observar la fuerza coriolis aparece en dirección perpendicular al eje de rotación y a la velocidad de desplazamiento del objeto. [15]

Ejemplo 2 - El conocido péndulo de Foucault. Si tenemos un péndulo atado a una estructura giratoria como la tierra pero que se pueda mover libremente, la tierra gira pero el péndulo no, de forma que para un observador estático en el espacio el péndulo realiza una línea recta mientras la tierra rota, mientras que para un observador en la tierra el péndulo gira en función de la rotación de la tierra. [16]

Ilustración 13 The Foucault pendulum, invented by Jean Bernard Léon Foucault in 1851 as an experiment to demonstrate the rotation of the earth [16]

En los ejemplos presentados, la velocidad de giro del sistema no inercial ha provocado una variación en la posición relativa del objeto. Este mismo principio utilizan los giróscopos MEMS.

El giróscopo MEMS dispone de 2 masas conocidas con cierta movilidad lograda mediante materiales flexibles. Para que las masas perciban el efecto coriolis deben desplazarse, esto se logra aplicando una vibración a las mismas mediante la interacción de campos electromagnéticos. Al actuar el efecto coriolis el recorrido previsto se ve alterado. [16]

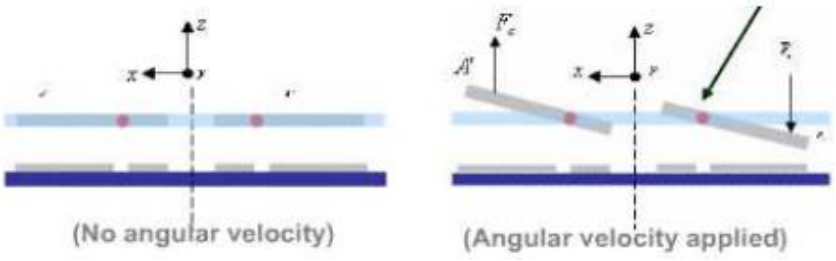


Ilustración 14. Representación del la variación de posición de las masas internas del giróscopo. [http://www.gte.us.es/ASIGN/SEA/pracs/Eval_Gir_con_datasheet.pdf]

Al modificar sus posiciones, estas masas varían su distancia respecto a unos sistemas capacitores que tienen debajo resultando una variación capacitiva que pasa a través de un amplificador operacional para traducirlo a variaciones eléctricas fácilmente cuantificables que finalmente serán traducidas a velocidades de giro.

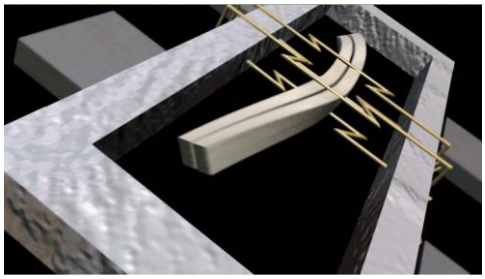


Ilustración 15. Representación del efecto coriolis modificando la posición relativa. http://diyhacking.com/wp-content/uploads/2014/11/gyro.jpg

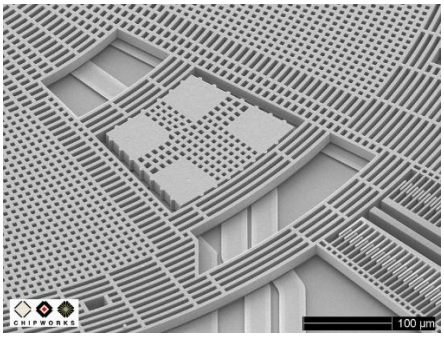


Ilustración. 16. Masa interna de un giróscopo. [https://www.ifixit.com/Teardown/iPhone+4+Gyroscope+Teardown/3156]

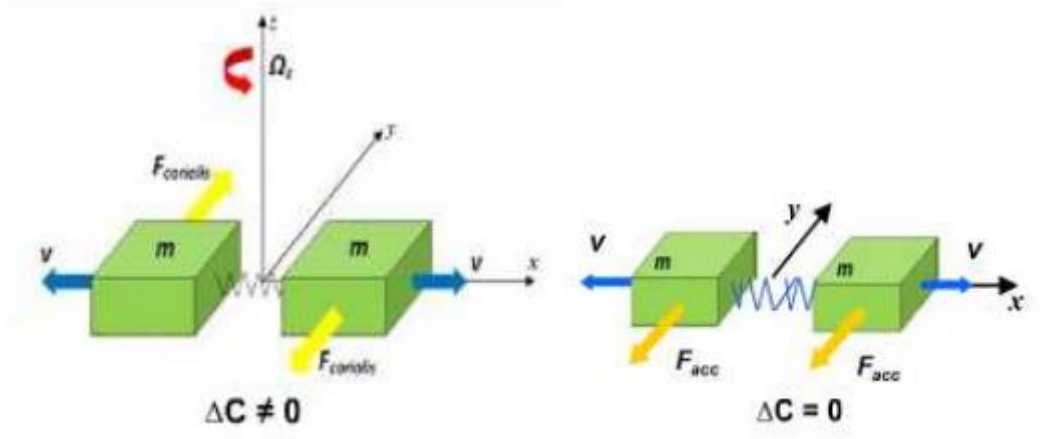


Ilustración 17. Efecto de los movimientos en giróscopo a la izquierda y en acelerómetro a la derecha. http://www.gte.us.es/ASIGN/SEA/pracs/Eval_Gir_con_datasheet.pdf

Mediante la aplicación de la medida diferencial entre los resultados de las dos masas se consigue eliminar una falsa medida de giro que podría ser provocada por una aceleración lineal, la cual sí afecta a los acelerómetros.

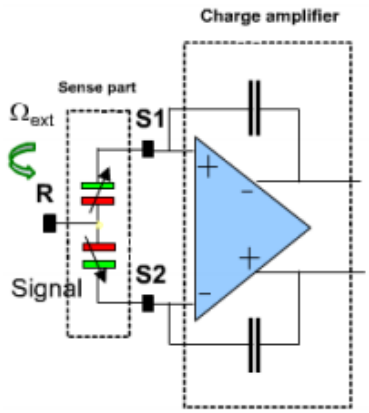


Ilustración 18. Medida diferencial y amplificación de señal. [http://www.gte.us.es/ASIGN/SEA/pracs/Eval_Gir_con_datasheet.pdf]

A la hora de realizar los cálculos se debe tener en cuenta que se ha obtenido una medida de la velocidad de giro, lo cual hay que traducir en ángulos para conocer la inclinación, para lograr esto se integra la velocidad angular.

$$\omega = \frac{d\theta}{dt} \quad \text{---> } \quad \theta = \int \omega \, dt \qquad \textit{Ecuación 3}$$

Traduciendo esta ecuación a programación, la integral se realiza obteniendo el valor de la velocidad y teniendo en cuenta el tiempo que le cuesta al programa completar un ciclo. Este tiempo de ciclo es el dt, cuanto menor sea más rápido obtendremos valores.

Conocido el tiempo de cada ciclo y la velocidad de giro que reporta el giróscopo se calcula la distancia angular recorrida desde la anterior medición. Este valor se añade al ángulo en el que se encontraba el sensor en el instante de muestreo anterior.

Tener que obtener los valores a través de realizar una integral provoca un pequeño error acumulativo en lo que se conoce como el "*drift*" propio del giróscopo, llegando a sacar valores que difieren mucho de la realidad. Este valor acumulativo se produce como consecuencia de sumar repetidamente el error estático del giróscopo a lo largo del tiempo. [20]

Al igual que con los inconvenientes de los acelerómetros este problema se soluciona con la adecuada implementación de un filtro, en nuestro caso el filtro complementario.

Un tiempo de muestreo muy pequeño no tiene por qué ser el adecuado de hecho no lo es y de forma experimental se ha comprobado que lo adecuado es un tiempo de muestreo de 50 ms. Con valores menores las medidas se disparaban.

3.4.4 Filtro complementario

Para juntar los valores de los giróscopos y acelerómetros de forma coherente hay que añadir un filtro que combine las 2 medidas correctamente.

Se planteó la idea de desarrollar un filtro Kalman, este filtro determina en cada instante qué peso se les da al giróscopo y al acelerómetro, pero su elevada complejidad con su tiempo de implementación y al no ser imprescindible se optó por la opción más común que es utilizar un filtro complementario, en el cual se establece la cantidad de aporte de cada medida de forma experimental.

El filtro complementario consiste en un filtro paso bajo y un filtro paso alto que se complementan de tal forma que la salida del filtro complementario siempre será una combinación entre el resultado del filtro paso bajo y paso alto. La expresión que lo define es la siguiente:

$$Valor\ del\ filtro\ complementario\ =\ (alfa) * medida1\ +\ (1 - alfa) * medida2$$

Traducido a las medidas con las que se trabaja:

$$Valor\ del\ filtro\ complementario\ =\ (alfa) *\ giróscopo\ +\ (1 - alfa) * acelerómetro$$

Ecuación 4

Las medidas obtenidas por los sensores son de dos naturalezas distintas. Por un lado tenemos las medidas de los acelerómetros, las cuales son precisas pero turbulentas y de respuesta lenta, son fácilmente perturbables por la presencia de vibraciones, experimentalmente se comprueba cómo los resultados son muy dispersos. La medida del acelerómetro se está viendo influenciada por perturbaciones rápidas pero también se observa cómo sigue la referencia muy de cerca. Para obtener unas medidas más precisas es necesario aplicarles un filtro paso bajo para descartar las variaciones rápidas que presenta.

Por otro lado tenemos las medidas de los giróscopos, estas medidas son muy rápidas y precisas pero se ven afectadas por el efecto del "drift" como se ha comentado con anterioridad. El "drift" es un error lento para el sistema del giróscopo por lo tanto para ignorar esta componente se somete a la señal del giróscopo a un filtro paso alto que discrimina las señales de bajas frecuencias entre ellas el "drift". [16]

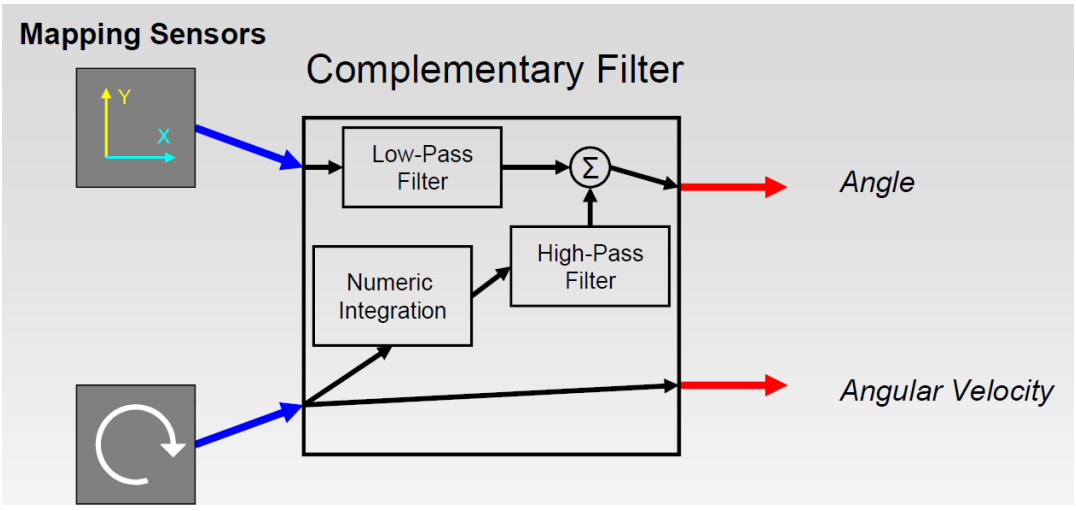


Ilustración 19. Comportamiento del filtro complementario.
Colton, Shane. The Balance Filter

3.4.5 Control PID

Todos los sistemas autónomos trabajan con parámetros que se ven alterados por distintos tipos de perturbaciones. Estas perturbaciones provocan que el resultado de nuestro sistema no sea el esperado, por eso se debe valorar el error que se produce mediante la medición del resultado experimental de la acción llevada a cabo para seguir la referencia. Es para esta medición para la que se utilizan los sensores, en nuestro caso el sensor de inercia MPU-6050.

El control de las señales está presente a nuestro alrededor en todo momento, sin ir más lejos en el teléfono móvil. Actualmente ya hay teléfonos que ajustan el brillo de la pantalla en función de la luminosidad solar. Este proceso se lleva a cabo con un sensor de esta intensidad cuya medida es comparada con una referencia. El resultado del control hace posible saber cuánta luminosidad extra necesita la pantalla para una correcta visualización ante una falta de luz externa o por el contrario permite determinar en qué medida se puede reducir la intensidad lumínica con intenciones de reducir el consumo de batería sin comprometer la correcta visión del dispositivo.

Un ejemplo más tradicional es el control del termostato en el hogar, se establece la referencia en la temperatura deseada y continuamente un sensor evalúa la temperatura de la habitación para establecer si debe activar o mantener los sistemas de calefacción o si debe apagarlos y permitir que el calor se disipe de forma natural. [23]

Los elementos que componen el sistema se pueden clasificar acorde con las fases presentadas en la siguiente ilustración que representa un lazo cerrado de control.

- El sensor de temperatura es el *elemento de medida*.
- Un circuito comparador en un sistema analógico, o un código de programación en un sistema digital, compara la señal del sensor con una referencia y hace el papel de *detector de error*.
- El error es procesado por el controlador en la fase de *control*, se realizan las operaciones oportunas sobre el error para traducirlo en una señal de acción para los actuadores.
- Por último, los actuadores reciben la señal traduciéndola en comportamientos físicos en la fase de *planta* o *proceso*.
- Los efectos resultantes de la actuación son analizados periódicamente por el *elemento de medida* cerrando el lazo y logrando el control de la señal.

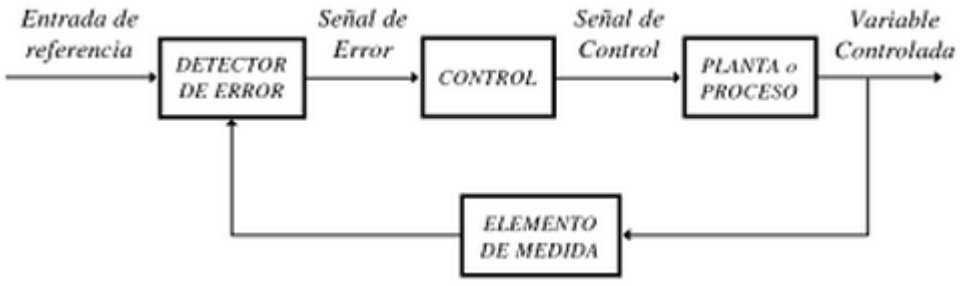


Ilustración 20. Sistema de bloques de lazo cerrado de control.
Spartacus Gomáriz Castro. Universitat Politècnica de Catalunya, 1998.

El sistema que engloba nuestro dron se puede clasificar también según la selección de la figura anterior.

- El sensor de momento de inercia es el *elemento de medida*.
- La fase de *detección de error* se programa en Arduino con una sustracción entre la referencia y la medida del sensor.

$$error = lectura\ de\ sensor - referencia$$

- El *control* al igual que la fase anterior se realiza mediante código de programación en la plataforma Arduino, englobando una serie de operaciones que se presentarán más adelante.
- La *planta* o *proceso* simboliza los cambios que los actuadores provocan para convertir la señal de actuación en una nueva posición.
- La nueva posición del aparato es analizada por el *elemento de medida* de forma periódica cerrando el lazo de control.

En muchas ocasiones el controlador es necesario por el desconocimiento del comportamiento preciso de los actuadores. Este es nuestro caso pues en él influyen todos los aspectos constructivos del dispositivo desde la aerodinámica hasta la distribución de masas pasando por las vibraciones.

Además de la incertidumbre propia de la planta se experimentan unas fuertes perturbaciones que deben tenerse en cuenta. Estas alteraciones pueden ser causadas por fuerzas externas como el viento o la colisión de objetos, pero también pueden producirse por efectos electromagnéticos como ruido eléctrico.

Suponiendo que conociéramos a la perfección los datos de la planta $G_2(s)$. El controlador $G_1(s)$ elabora una señal óptima para alcanzar la posición medida con el sensor $H(s)$. No obstante esta posición no es alcanzada con perfección ya que se ve alterada por las perturbaciones $N(s)$ que se experimentan desde el instante de muestreo del sensor hasta la llegada de la señal a los actuadores y durante la acción de estos.

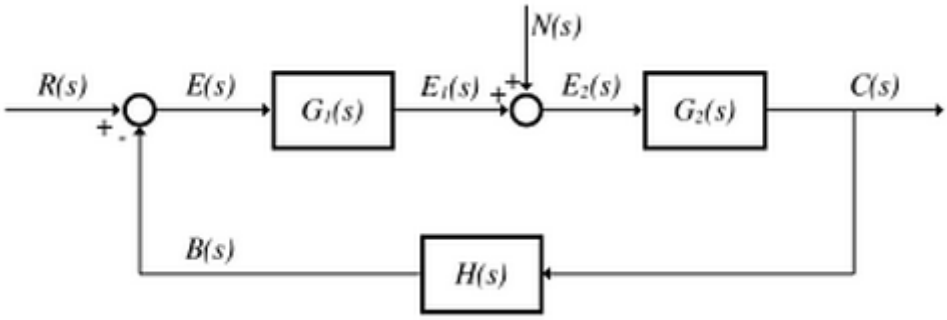


Ilustración 21. Sistema de bloques de lazo cerrado de control con perturbaciones.
Teoría de control: diseño electrónico. Spartacus Gomáriz Castro. Universitat Politècnica de Catalunya, 1998.

Los sistemas de control habituales están formados por tres fases diferenciadas en cuanto a la evaluación del error. La parte proporcional, la parte integral y la parte derivativa. La necesidad de utilización de las fases depende de las características de la aplicación que se realice. [23]

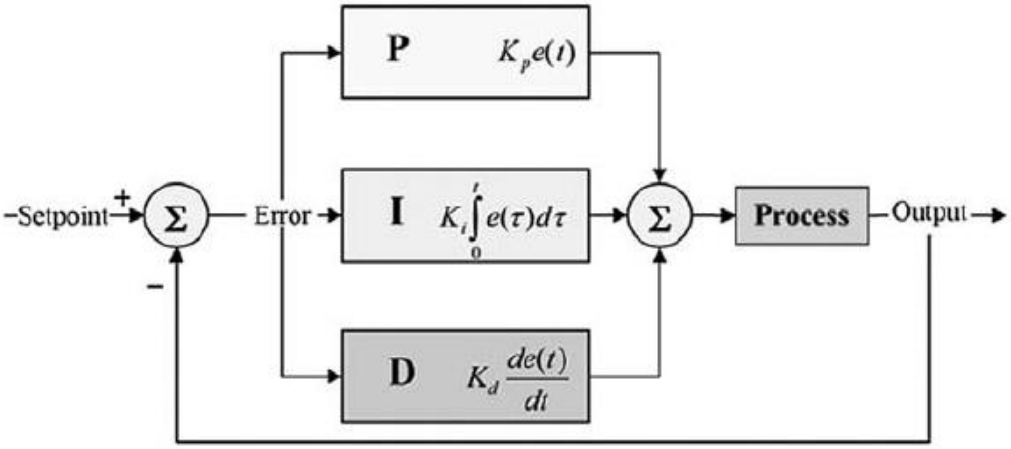


Ilustración 22 Diagrama de bloques clásico de control PID.
Build your own Quadcopter.

La **acción proporcional** $P(t)$, amplia o reduce el error en función de una ganancia proporcional.

$$P(t) = K_p * e(t)$$

Donde:

- $P(t)$ es la acción proporcional de control expresada en función del tiempo.
- K_p es la ganancia proporcional
- $e(t)$ es el error expresado en función del tiempo.

Si por ejemplo entre dos instantes de tiempo el error fuera 3 grados, con $K_p = 10$, la acción proporcional sería 30. Se debe comparar este aumento de la acción de control con el resultado del movimiento y ajustar el valor de la constante para que se adecúe con la cantidad de movimiento deseado.

La **acción integral** $I(t)$, acumula el error a lo largo del tiempo, es ajustada con la ganancia integral K_i.

$$I(t) = K_i \int_0^t e(t)dt$$

Donde:

- $I(t)$ es la acción integral de control expresada en función del tiempo.
- K_i es la ganancia integral.
- $\int_0^t e(t)dt$ es la integral del error.

Al tener un carácter acumulativo esta componente ayuda a ajustar la señal a la referencia. El valor de la integral siempre va a ir en aumento mientras el error no cambie de signo, es decir, cuanto más tiempo tarde la señal en alcanzar la referencia mayor va a ser el valor de la acción integral y por tanto su aporte a corregir el error.

Esta acción tiene una labor muy importante no obstante también acarrea un problema. Cuando se dan un conjunto de perturbaciones inesperadas o cambios rápidos de referencia la parte integral puede sufrir un aumento excesivo del error al inicio del régimen transitorio, esto hace

que se sature. Se produce un aumento de la acción integral que puede incluso superar los valores límites de la máquina en gran medida. En los siguientes ciclos de control el valor del error integral será muy elevado y descenderá lentamente alargando los tiempos de estabilización. Este efecto se denomina "*wind up*". [25]

Para solucionar este problema existen varios sistemas que realizan la integral o no en función de ciertas condiciones. De todas formas es suficiente con una limitación de los valores que puede alcanzar el resultado de la acción integral entre los máximos valores que es capaz de interpretar el actuador.

La **acción derivativa** $D(t)$, evalúa el incremento del error en función del tiempo, ponderado mediante la ganancia derivativa K_d.

$$D(t) = K_d * \frac{d}{dt} e(t)$$

Donde:

- $D(t)$ es la acción derivativa de control expresada en función del tiempo.
- K_d es la ganancia derivativa.
- $\frac{d}{dt} e(t)$ es la derivada del error expresada en función del tiempo.

Esta componente compensa el error de forma proporcional a la velocidad de aumento del mismo y es la encargada de aportar la rapidez al sistema. Se le debe asignar un valor reducido a la ganancia ya que puede verse afectado por el ruido desestabilizando el sistema. [22]

La expresión final de la acción de control es la suma de cada componente.

$$u(t) = C + P(t) + I(t) + D(t) = C + K_p * e(t) + K_i \int_0^t e(t)dt + K_d * \frac{d}{dt} e(t)$$

3.4.6 Acción de control, control de los motores

Los motores a utilizar en nuestro UAV deben ofrecer una relación elevada potencia/peso y velocidad/peso con el propósito de minimizar el peso del conjunto para que suponga un mínimo consumo de corriente y por lo tanto una mayor autonomía.

Varios tipos de motores se han planteado pero descartado por diversas razones, en primer lugar los motores de corriente continua con escobillas tienen un menor par y las escobillas suponen un peso añadido. Por otro lado los motores paso a paso tienen una buena relación potencia/peso pero no tanto la relación velocidad/peso, tienen mucha precisión de giro pero en nuestra aplicación prima la velocidad.

La mayor relación potencia/peso y velocidad/peso para motores de corriente continua se encuentra en los motores sin escobillas, brushless. Dentro de la gran variedad de productos que hay, se han tomado los motores de una marca conocida que presenta bastante información sobre los productos. BPHobbies.

Entre los productos de este fabricante se ha elegido el *Cheetah A2208-14 Brushless Outrunner Motor*. En la tabla que se muestra a continuación, proporcionada por el fabricante, se puede calcular la cantidad de peso a levantar por el motor, teniendo en cuenta que se utilizan 4 motores se debe multiplicar la capacidad de un motor por 4 y se obtiene la posibilidad de levantar incluso 200 kg. Este peso es muy elevado y no va a ser necesario un funcionamiento tan fuerte, pero siempre es recomendable sobredimensionar los motores ya que su eficiencia ronda el 80%. Finalmente el empuje que se genere dependerá de las dimensiones de las hélices. De entre varios motores de la misma serie, los motores elegidos disponen de la eficiencia de aceleración intermedia.

Aunque se va a seguir sin superar el peso que pueden superar los motores, con mucho margen, se debe tener en cuenta que es posible que se añada una carcasa, o cámara de video u otros elementos de los que se desconoce su peso previamente.

Propeller:	Volts (V):	Amps (A):	Thrust (g/oz):	Power (W):	Thrust Efficiency
APC 7 x 5 E	11.8	12.9	19	152	.13 oz/W
APC 7 x 4 E	11.9	11.2	18.8	132	.14 oz/W
APC 7 x 4 SF	11.7	12.2	18.90	143	.13 oz/W
APC 6 x 5.5 E	11.7	9.5	11.3	111	.10 oz/W
APC 8 x 4 E	8	8.5	14.7	68	.22 oz/W
APC 9 x 4.7 SF	7.8	12.6	18.3	98	.19 oz/W

Ilustración 23. Tabla ilustrativa de las potencias en función de las dimensiones de las hélices, para el motor Cheetah A2208-14 Brushless Outrunner Motor.
http://www.bphobbies.com/view.asp?id=V450327&pid=B1898550

A la hora de elaborar la acción de control hay que tener en cuenta cuál va a ser la dinámica de movimiento del "*drone*", y cuál va a ser la posición del sensor respecto al sistema de referencia de la estructura, ya que la variación de inclinación de la estructura será interpretada por el sensor y debemos establecer unos comportamientos correctos.

Si el sensor nos manda un aviso de que el giro en el eje X realizado ha sido de 30^{O}, la acción de control debe enviarse a los actuadores para que corrijan 30^{O} en el eje X, no en el Y o en otro distinto. Por lo tanto se debe tener en cuenta la orientación del sensor en la placa y de la placa en la estructura.

A la hora de estabilizar el dron, se debe prestar cuidado en no modificar la altura del mismo, ya que continuas variaciones acabarían haciendo que perdiéramos el control de la altura. Para lograr este objetivo al realizar los giros respecto a cualquier eje se reduce la velocidad de 2 motores en la misma magnitud en la que se aumenta la de los otros 2.

Esta magnitud que va cambiando es el resultado del lazo de control PID, que no es más que la diferencia entre la inclinación buscada y la inclinación en la que se encuentre en ese momento.

Para explicar los movimientos que realiza el dron asociado con la variación de velocidad de sus actuadores nos ayudamos de la siguiente ilustración, donde el eje longitudinal es el X con sentido positivo según se indica y el lateral el eje Y con sentido positivo hacia la derecha, y por último el eje vertical se completa acorde con el sistema levógiro, sentido positivo del eje Z hacia arriba.

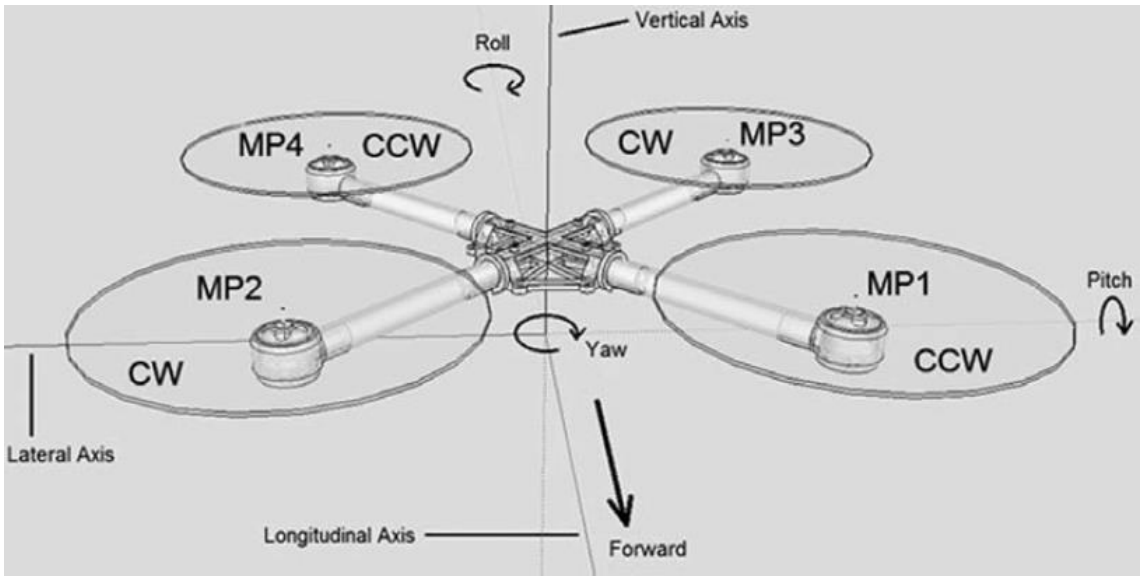


Ilustración 24. Concordancia de ejes en quadcopter de tipología X.
Build your own Quadcopter

Acorde con las etiquetas de la ilustración la asignación del sentido de giro de los motores se establece cruzados 2 a 2 en sentido horario (CW – Clockwise) y antihorario (CCW – Counter Clockwise).

Los efectos giroscópicos y aerodinámicos generan un momento de inercia en el centro de masas que hace girar al dron respecto al eje vertical. Con esta tipología se anulan mutuamente los momentos de inercia y el dispositivo no se moverá inesperadamente. [14]

Si se observa en la ilustración 1 el primer ejemplo con tres brazos dispone del izquierdo en un sentido, el derecho en el contrario y en la cola se ve una anotación que indica que dispone de 2 motores. En el último ejemplo de la misma ilustración, se exponen dos motores en cada brazo cada uno girando en un sentido, compensando así el par producido por el giro.

Para hacer la analogía de giros se utilizan las etiquetas MP1, MP2, MP3, MP4 como indicadores de la velocidad de cada motor. Una misma velocidad en todos los motores se traduce en una determinada altura y posición horizontal, que equivale a MP1=MP2=MP3=MP4=constante (altura).

Controlando los cambios de velocidad de los motores se pueden lograr los giros en los 3 ejes alcanzando las inclinaciones y orientaciones deseadas. En la ilustración 24 los sentidos de giro positivos no se corresponden con la habitual asignación del sistema levógiro, están invertidos, por valernos de la ilustración se utilizan los sistemas de referencias indicados en la ilustración, tanto para las referencias longitudinales como radiales.

Para desplazar el dron hacia la derecha en la imagen, se debe realizar el giro en el sentido positivo del *eje X*, *roll*, las expresiones de las señales de actuación son las siguientes

- MP1 = altura - PID_roll
- MP2 = altura + PID_roll
- MP3 = altura - PID_roll
- MP4 = altura + PID_roll

De esta forma, si tuviéramos una medida positiva del ángulo de giro row, el resultado del PID realizado sobre el ángulo X nos informará de que debemos reducir el giro, dándonos un valor resultante de PID_roll negativo. Este valor va a provocar que la parte izquierda se eleve, la parte de la derecha descienda, desplazando el dron a la derecha.

Para desplazar el dron hacia el frente de la imagen, se debe realizar el giro en el sentido negativo del *eje Y*, *pitch*, las expresiones de las señales de actuación son las siguientes.

- MP1 = altura - PID_pitch
- MP2 = altura - PID_pitch
- MP3 = altura + PID_pitch
- MP4 = altura + PID_pitch

De esta forma la pareja del fondo se elevará, la pareja del frente descenderá y el dron se desplazará hacia delante.

Para realizar un giro del dron en vertical en el sentido positivo (CW), *yaw*, se debe aumentar la intensidad en los motores que giran en sentido horario y disminuir la misma cantidad en los que giran en sentido anti horario para mantener la altitud. Las expresiones de la velocidad se muestran a continuación. [17]

- MP1 = altura - PID_yaw
- MP2 = altura + PID_yaw
- MP3 = altura + PID_yaw
- MP4 = altura - PID_yaw

Para obtener el sentido de giro contrario en todos los ejemplos presentados sería necesario cambiar todos los signos de las operaciones, si la salida de cada PID fuera de signo constante. No obstante tal y como se ha presentado antes, se debe puntualizar que esta magnitud es el resultado del control PID que trabaja con la medida de los sensores, los cuales dan el resultado positivo o negativo según el ángulo de giro en el que se encuentren.

El signo de la salida del PID viene dado en función de la diferencia entre la medida de los sensores y la referencia asignada. El sensor en la placa se orienta en el mismo sistema de referencia que la estructura. De esta forma se ajusta automáticamente la variación de velocidad necesaria en cada momento para compensar el ángulo y alcanzar la referencia.

Para la programación de la acción de control se estableció una programación inicial a vistas de ajustarse con la posición final del sensor en la estructura.

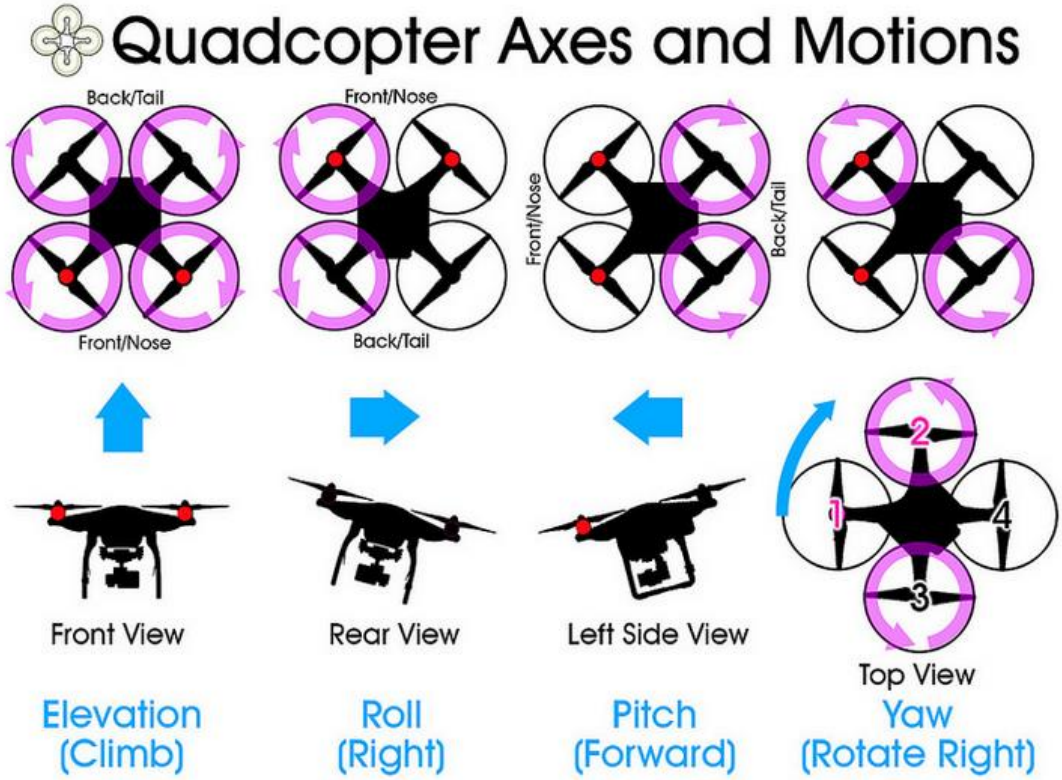


Ilustración 25. Comportamiento del dron en función de la acción de control de sus motores. Drones: Multidisciplinary Uses. Instructor Ric Stephens Portland State University. https://www.flickr.com/photos/ricstephens/16989545847/in/photostream/

En la ilustración 25 se muestran los movimientos que realiza el dron en función de la actuación de los motores. Los motores sobre los que se indica la flecha con la dirección de movimiento representan los que aumentan su intensidad, sobre los que no hay indicación representan los motores que disminuyen su intensidad.

Se expone como otra opción la situación de la placa de una tipología en +, simplificándose el control de los motores. Para esta tipología el control de los giros en los distintos ejes solo requerirá de la actuación sobre dos de los cuatro motores, manteniendo los otros dos con la intensidad constante para no perder altura.

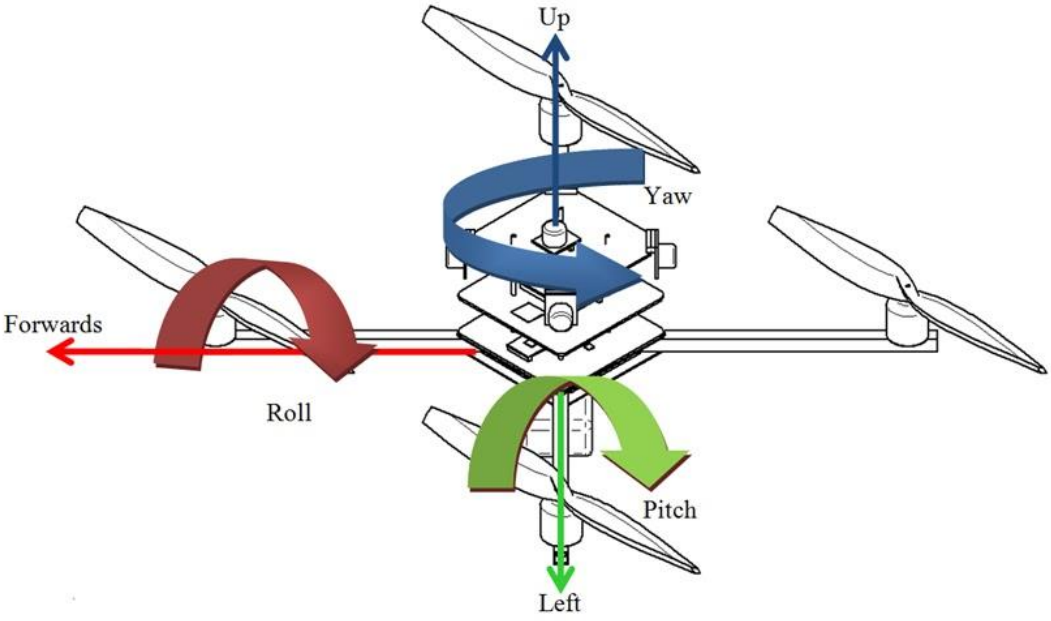


Ilustración 26. Drone Plues Configuration. Institute of System Dynamics.
http://hpcas.blogspot.com.es/2013_06_01_archive.html

La configuración de las órdenes de control para realizar los movimientos basándonos en la ilustración 26 se presentan a continuación. Se considera un sistema levógiro con sentidos de giro positivo anti horario. El ángulo de giro roll y pitch tienen el sentido positivo contrario a la imagen:

Control de roll

- MP1 = altura
- MP2 = altura + PID_roll
- MP3 = altura
- MP4 = altura - PID_roll

Control de pitch

- MP1 = altura + PID_pitch
- MP2 = altura
- MP3 = altura - PID_pitch
- MP4 = altura

Control de yaw

- MP1 = altura - PID_yaw
- MP2 = altura + PID_yaw
- MP3 = altura - PID_yaw
- MP4 = altura + PID_yaw

3.4.7 DesignSpark

Para la conexión de todos los elementos y su adhesión a la estructura se ha construido una placa de circuito impreso para lo que se ha utilizado el software gratuito Design-Spark proporcionado por RS.

En el diseño de la placa impresa ha sido necesario a su vez el diseño de cada uno de los elementos ya que ninguno de los productos a utilizar son de RS; los componentes de RS se encuentran fácilmente con la función ModelSource del programa Design Spark.

Algunos dispositivos incluían archivos para el programa Eagle, programa mediante el cual también se diseñan placas de circuito impreso al igual que con Design Spark, y mediante una función adicional del programa Eagle se han podido transcribir a código ASCII para luego ser leídos utilizando el Design Spark. El problema de estos archivos de Eagle es que viene información del conexionado interior del dispositivo o elemento del que se trate, cuando en nuestro caso lo que importan son las distancias de los pines. Finalmente estos archivos han servido de ayuda eliminando el resto de información que sobraba y analizando la importante.

Para todos los elementos se ha creado un símbolo personalizado tanto en el esquemático como en la PCB. A lo largo del desarrollo del PCB se han ido implementando pequeñas mejoras como la inclusión de un interruptor, un circuito medidor de batería y alguna que otra huella para futuras ampliaciones de las funciones del "*drone*".

Los elementos incluidos en la placa de circuito impreso son los siguientes:

- Un interruptor de 2 fases NA para las fases GND e INPUT que estarán directamente conectadas a una batería. El interruptor en cuestión se compró en una tienda y no disponían de datos técnicos por lo que para obtener las medidas se realizó un procedimiento bastante rudimentario pero funcional, se pintaron las patas del interruptor y se marcó la huella en un papel para luego medirlo con precisión. El interruptor dispone de otros 2 pares de contactos NC que no se utilizan.

Ilustración 27 Interruptor.

- Se ha decidido la utilización de una batería Li-Po (Litio – Polímero) para la alimentación de todo el sistema por su excelente relación capacidad/peso.

  Se ha comparado con otros ejemplares como las baterías Ni-Cd (Níquel – Cadmio) pero la baja relación capacidad/peso destina estas baterías a elementos como coches o lanchas eléctricas en las que el peso no es un factor tan importante.

  Por otro lado las baterías de Ni-MH (Níquel – Metal hidruro) mejoran en gran medida su relación capacidad/peso respecto a las baterías de Ni-Cd, con un precio que no es desorbitado, como en las baterías LiPo. El inconveniente de este tipo de baterías es la sensibilidad que tiene a la hora de cargarla, un exceso de horas de carga deteriora enormemente la batería, se deben utilizar cargadores inteligentes.

  En último y definitivo lugar se han evaluado las baterías LiPo, estas baterías han aumentado sus ventas en el mercado al incluirse en dispositivos móviles modernos.

Presentan la mayor relación capacidad/peso a costa de elevar su precio a cifras desorbitadas, no obstante para la aplicación que se desea la autonomía y el peso del dispositivo están directamente relacionados. Un mayor peso implica una mayor acción de control y un mayor consumo de los motores para compensar la fuerza de la gravedad.

Hay que prestar especial atención a la hora de realizar la carga de estas baterías, han sido causantes de gran número de incendios. Al entrar el litio en contacto con el oxígeno, sin necesidad de otro comburente, se inicia una combustión que alcanza miles de grados y tiene un comportamiento casi explosivo. [18]

Estas baterías están formadas por distintos segmentos de 3.7V y en función del número de segmentos que tenga en serie, en sus bornes se establece una tensión u otra. En nuestro caso se utiliza una batería LiPo 3s lo que significa que tiene una tensión de 11.1V

El gran inconveniente de estas baterías es, al contrario que en las baterías de Ni-MH, la excesiva descarga de la misma. Un descenso de la tensión de las celdas por debajo del 80% puede provocar la inutilización permanente de la batería. Para protegernos de que se produzca este problema, es necesario conocer la tensión en la batería en todo momento.

- A la salida del interruptor se encuentra el circuito medidor de batería que no es más que un divisor de tensión que nos permite medir la tensión de la batería. Según las indicaciones de su carcasa su tensión habitual de funcionamiento es de 11.1V y según las curvas de carga habituales de las baterías LiPo de esas características, 9V es un voltaje representativo de batería descargada. Se añade un divisor de tensión 1:4 para que se obtenga un valor máximo de 3V y un valor mínimo de 2.25 estando así el valor centrado entre los límites de la entrada analógica del Arduino Nano mediante la que se monitoriza la medida a través del ADC concretamente mediante la entrada A3. Además del divisor de tensión se incluye una resistencia serie y dos diodos que aseguran el rango de voltaje entre 0V y 5V y limitan la entrada de corriente.

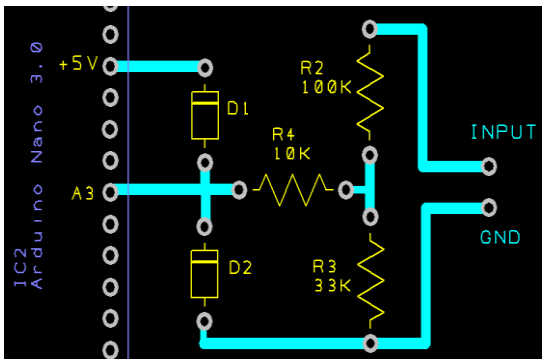


Ilustración 28 Circuito Batería

- Los valores elegidos para el divisor de tensión, además de establecer una proporción 1:4 están pensados para que no suponga un consumo extra de potencia, para ello el nivel de impedancia es relativamente elevado.

  Teniendo en cuenta que el máximo valor en el punto intermedio va a ser 3V a causa del divisor de tensión, se estima la corriente máxima que puede llegar a entrar por el terminal analógico, aunque en la práctica será bastante menor.

$$\frac{3V}{10000\Omega} = 3 * 10^{-4}A = 0.3mA$$

Se ha añadido una resistencia de 10kΩ siguiendo las indicaciones del fabricante del procesador del que dispone Arduino Nano, el ATmega 328P, en el datasheet se indica que las entradas analógicas están optimizadas para una resistencia de salida de máximo 10kΩ. [22]

- A continuación del interruptor, en paralelo con el circuito de la batería, se encuentra el regulador de tensión L7805C, en encapsulado TO-220, que limita la tensión de la batería, 11.1V, para proporcionar una salida constante de 5V.

Ilustración 29 TO-220 L7805C

- Una vez nivelada la tensión a 5V se alimenta con ella al Arduino y a los conectores en los que se enchufan los ESC. Para implementar esta conexión se han soldado 4 zócalos macho-macho de tres pines en los que enganchar cada cable de conexiones de los motores.

- Los ESC además de conectarse a los 5V, conexión rotulada como VCC, se conectan a tierra (GND) y cada uno a una salida digital del Arduino (D3-D4-D5-D6).

- La alimentación del resto de elementos de la placa se realiza a través de las salidas de +5V y 3V3 del Arduino Nano. +5V para la alimentación del compás HMC6352 y 3.3V para la alimentación de los sensores MPU6050 y BMP180 y del módulo de conexión Xbee.

- Se ha establecido una capa de cobre en la parte inferior de la placa que está conectada al cable de entrada GND. A esta capa de cobre se conectan tan solo los pines que deben estar a tierra.

  El objetivo de esta capa de cobre es disminuir las pequeñas variaciones de tensión que pueden encontrarse en los distintos puntos de la placa por caída de tensión en la propia placa. También actúa como aislante entre distintas pistas evitando así interferencias.

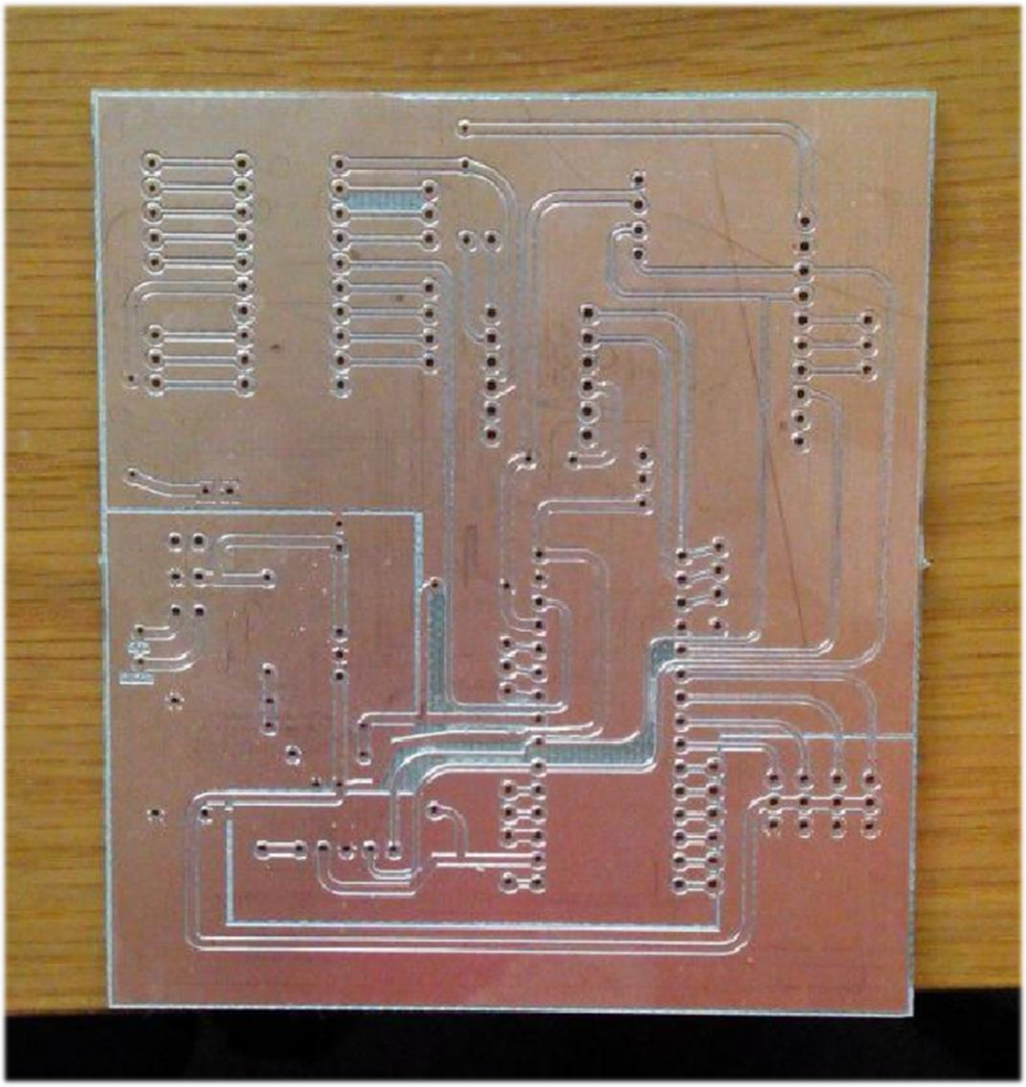


Ilustración 30 Placa impresa del "drone" fabricada

El cobre no es un conductor ideal, pues no existen, por lo que aunque sea ínfima, presenta una resistencia que provoca cierta caída de tensión entre dos puntos de una pista, al trabajar con componentes eléctricas de escalas muy reducidas, pequeñas variaciones pueden resultar en comportamientos anómalos. Añadiendo la capa de cobre a una misma tensión se reduce la caída de tensión y se unifica en toda la placa.

La capa de tierra se ha dividido en tres zonas. Dos secciones alimentan los dispositivos del circuito de señal, los sensores, el microprocesador, el módulo de comunicación y su enlace. Se han dividido en dos por la existencia de una gran "isla" producida.

Una "isla" en PCB se refiere a una sección de la capa de cobre que no está comunicada con el resto de la capa debido a las restricciones necesarias para que no se produzcan cortocircuitos no deseados con las pistas.

Una tercera sección alimenta los dispositivos del circuito de potencia, el circuito de entrada de la alimentación, el circuito medidor de batería y la conexión de los servomotores.

Estas tres secciones de cobre se comunican entre sí a través de una sola pista, de forma que las tres secciones sean equipotenciales.

- Para la comunicación del Arduino con los sensores se utiliza la comunicación I²C, Inter-Integrated Circuit, la cual actúa mediante las conexiones SDA para la transmisión de datos y SCL para la marca de reloj, conecta todos los sensores periféricos de Arduino. Se comentará el protocolo I²C con más detalle en próximos apartados.

Ilustración 31 MPU6050

  El sensor que se utiliza es el MPU6050 alimentado a 3.3V, no obstante se realiza el diseño para incluir en un futuro el compás HMC6352 alimentado a 5V y el sensor digital de presión BMP 180 a 3.3V.

  La estructura completa del sensor de inercia se denomina "MPU-6050 GY-521 Breakout Board". Esta construcción facilita la comunicación I²C, incorpora a su vez un regulador de tensión que permite que se conecte la alimentación a 5V.

- Para la comunicación del Arduino con el módulo Xbee se utiliza la comunicación serie, mediante la conexión de los pines "RX0" y "TX1" del Arduino Nano a sus correspondientes DOUT y DIN del módulo Xbee. No obstante las salidas del Arduino tienen 5V de tensión mientras que el módulo Xbee admite como máximo 3.3V de entrada por sus terminales siendo necesario reducir la tensión de la salida de datos del Arduino para que el módulo Xbee los pueda recibir sin sufrir daños y aumentar la tensión de la salida de datos del módulo Xbee para que el Arduino los pueda recibir con precisión.

Ilustración 32 XBee Module

  En ciertas ocasiones el circuito elevador no se utiliza y es suficiente con que entren 3.3V en RX0 de Arduino, no obstante en nuestro caso se ha utilizado un conversor de niveles que implementa las dos funciones, el divisor de tensión reductor y el circuito elevador, para el que se utiliza un transistor MOSFET y dos resistencias.

  La conexión entre el Arduino y el módulo Xbee se hace a través de una Breakout Board la cual se caracteriza por tener una distancia entre sus pines de 0.1", la medida habitual, y sobre la que se incorpora el módulo Xbee que tiene una distancia entre pines propia.

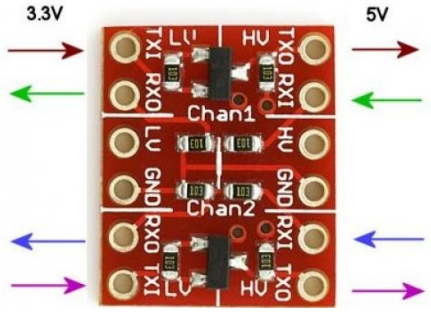


Ilustración 34 Conversor de niveles

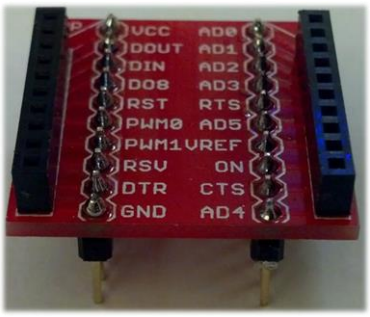


Ilustración 33 Xbee ZB Module Breakout Board

Se ha optado por añadir en el pin 15 (D5) del módulo Xbee el denominado Associate LED. Este LED parpadeará de diferentes modos en función del estado del módulo con la red. Este pin debe activarse estableciendo su valor en 1, HIGH; está así configurado por defecto. Cuando el módulo no esté conectado a una red el LED permanecerá encendido y cuando se conecte con alguna red comenzará a parpadear con una frecuencia que puede ser ajustada mediante el comando LT del módulo. [23]

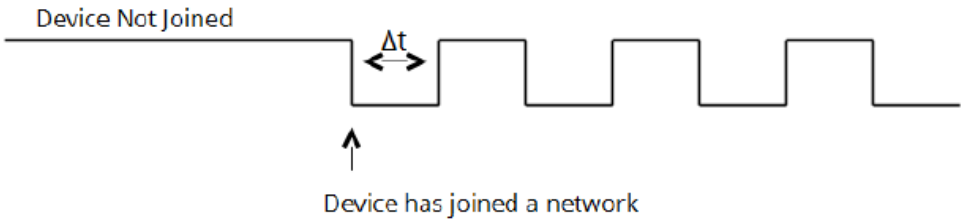


Ilustración 35 Indicación del Associate LED en función del estado de la conexión. Xbee/XBee-PRO ZigBee RF Modules User Guide

- A la hora de diseñar las conexiones de la placa se ha tenido especial interés en situarlas en la parte de debajo de la placa. Este procedimiento tiene como objetivo facilitar las labores de soldadura, teniendo en cuenta que se van a realizar de forma manual, se deben disponer los elementos de forma que la soldadura se pueda hacer cómodamente.

  Todos los elementos que se utilizan en el sistema son "Through Hole", es decir, para ensamblarlos a la placa sus patillas atraviesan de lado a lado la placa, y se sueldan por el lado que tenga las conexiones. Al tener un diseño sencillo ha sido posible colocar todos los elementos en una cara de la placa y todas las conexiones en la otra cara. A la hora de soldar por un lado solo accede a las conexiones y las patillas que sobresalen de cada elemento.

  En tres ocasiones ha sido necesario recurrir a la parte superior, no obstante en esas situaciones se ha tratado de reducir al mínimo la longitud de la conexión en la capa superior de forma que a la hora de soldar se ha podido realizar fácilmente con un cable que hace de puente, de esta forma no hace falta cubrir de cobre y luego mecanizar la parte de arriba de la placa, con su consiguiente ahorro en materia, energía y tiempo.

  En general para un diseño óptimo de cualquier PCB el número de vías se debe reducir al mínimo posible por razones económicas y de fiabilidad, cuantas menos juntas de soldadura haya que hacer, menos riesgo de error se produce, y cada vía implica dos juntas de conexión, la de la capa superior y la de la capa inferior.

  Las vías se utilizan para pasar de una capa de cierta tensión a la otra, en este caso tan solo hay una capa pero se utiliza el término vía por su similitud de funcionalidad, ya que se han utilizado para evitar el cruce de líneas de corriente.

- Se debe prestar atención en el diseño de las pistas, un grosor demasiado pequeño puede limitar el funcionamiento de la placa al no permitir circular toda la corriente disponible.

En este caso la corriente que circula por los elementos es muy reducida y no se han presentado problemas, en dos ocasiones se ha reducido incluso el grosor de la pista de alimentación para cuadrarlo en la estructura de las conexiones.

La corriente máxima que puede consumir el Arduino UNO es de 500mA en estado de funcionamiento normal, sin encontrarse en modo SLEEP. Este es el consumo límite máximo de Arduino utilizando todas sus funcionalidades. Estos cálculos son una estimación ya que la información al respecto es un tanto dispersa e influyen gran cantidad de factores en el consumo de corriente, desde la resistencia interna de los elementos utilizados hasta el aumento de temperatura de los dispositivos.

- El circuito que se utiliza no tiene un consumo elevado de corriente, el módulo inalámbrico XBee es el segundo elemento que más corriente consume con hasta 205mA en transmisión y un consumo de potencia máximo de 63mW. [22]

  Teniendo en cuenta estas características se ha realizado una estimación del grosor mínimo que deben tener las pistas para procurar una correcta transmisión de la energía.

Conductor width (in)	Current (Amps)			
	½ oz	1 oz	2 oz	3oz
0.005	0.13	0.50	0.70	1.00
0.010	0.50	0.80	1.40	1.90
0.020	0.70	1.40	2.20	3.00
0.030	1.00	1.90	3.00	4.00
0.050	1.50	2.50	4.00	5.50
0.070	2.00	3.50	5.00	7.00
0.100	2.50	4.00	7.00	9.00
0.150	3.50	5.50	9.00	13.00
0.200	4.00	6.00	11.00	14.00

Ilustración 36. Relación entre el ancho de los conductores y la capacidad de transmisión de corriente. Jesús Corres. Diseño de tarjetas electrónicas. Tema 3.

- Un consumo de 0.5A de corriente nos sitúa como mínimo en una anchura de conductor de:

$$0.01inch * 25.399 \frac{mm}{inch} = 0.25399mm$$

Este valor de anchura se encuentra muy por debajo de las anchuras de pista definidas en este proyecto, la pista que menor anchura tiene es precisamente de 0.254mm pero esta pista está destinada para la transmisión de datos, y por ella circula una corriente muy inferior a la que circula por las pistas de alimentación.

	Name	Width
X	Power Min	0.635
X	Power Nom	0.914
X	Signal Min	0.254
X	Signal Nom	0.381

Ilustración 37. Asignación de anchuras a las distintas pistas en Design Spark.

Se suele hacer una recomendación para obtener un buen comportamiento.

Anchura de pista GND = 2 * Anchura de pista alimentación = 2 * Anchura de pista de señal.

En la ilustración anterior la anchura *Power Nom* se asocia con las pistas de alimentación, y la *señal Signal Nom* se asigna a las de señal.

En un principio se utilizó la asignación *Power Nom* para las pistas de tierra pero al cubrir la placa con la capa de tierras las pistas de tierra ya no hicieron falta.

- Un aspecto fundamental a la hora del diseño del camino de las pistas es el control de los giros y ángulos que realicen las mismas, un ángulo recto de 90^{O} puede provocar grandes pérdidas de corriente, para las escalas en las que se trabaja. Existen dos métodos fundamentales para realizar correctamente estos giros, añadiendo ángulos 45^{O} o añadiendo curvas en las esquinas, con un radio suficiente para que el giro no sea brusco.

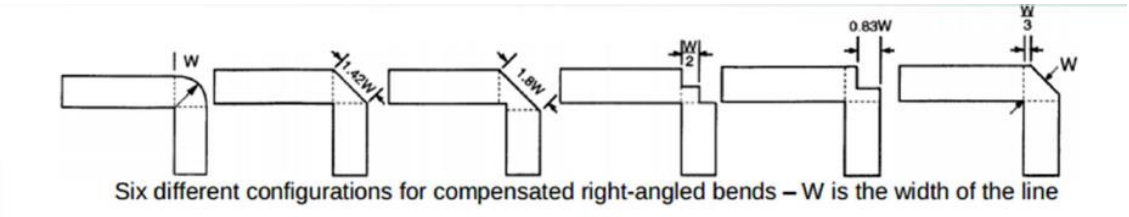


Ilustración 38. W = sección de transmisión de corriente en función del tipo de giro de la pista. Jesús Corres. Diseño de tarjetas electrónicas. Tema 3.

Se ha optado por aplicar el método de los radios para minimizar la reducción de sección en las pistas.

En ciertas aplicaciones que exigen una extrema velocidad y precisión, cuando hay pistas paralelas se añaden unos serpenteos que retrasan la señal para que la corriente que circule por la pista con radio más pequeño tarde lo mismo que la de radio más grande, la cual recorrerá mayor distancia, no obstante en esta aplicación no son necesarias transmisiones tan exactas.

- Además del grosor de las pistas, la longitud de las mismas también es un aspecto importante. Cuanto menor sea la longitud de una pista menor será la caída de tensión presente entre el inicio y el final de la misma. Se debe procurar realizar las pistas lo más cortas posibles, sobre todo las de señal, las cuales tienen una anchura más pequeña y por lo tanto su resistencia por cada unidad de longitud es mayor, al no poder la corriente circular con tanta facilidad.

Este aspecto toma especial importancia al hablar del regulador de tensión. Este debe situarse lo más cerca posible de la entrada analógica de Arduino. Para minimizar las variaciones de tensión.

- Otro aspecto de diseño que hay que evaluar es la distancia entre pistas. Un cortocircuito de las mismas puede provocar la inutilización del sistema, especialmente si se trata de un cortocircuito entre las pistas de alimentación y GND, estas pistas deben estar especialmente distantes entre sí, dentro de las escalas con las que trabajamos, estableciéndose un límite mínimo de 0.1 mm.

  El espaciado debe ser bastante más acusado (1-3 mm) en los bordes de la placa con el propósito de evitar daños en la misma al cortarla con la maquinaria.

  A su vez los pads tienen ciertas restricciones, el cobre que rodea el pad debe tener el suficiente radio para facilitar la soldadura, con más razón al realizarse de forma manual.

  El programa Design Spark realiza automáticamente la comprobación de errores de espaciado o grosor de pistas y pads en función de los parámetros que se preestablezcan. No obstante se debe evitar utilizar los márgenes mínimos establecidos para asegurar una mayor fiabilidad.

- Además de seguir las indicaciones presentadas, en todo proceso se debe optimizar energía, tiempo y material, para optimizar la fabricación de la placa el tamaño debe ser ajustado al mínimo. En menor medida esto también colabora en la reducción de peso, un aspecto fundamental para cualquier sistema de movimiento con alimentación independiente de la red eléctrica.

- La disposición de los distintos elementos en la placa también es un aspecto a evaluar. Se debe buscar un acceso fácil a aquellos elementos que lo requieran, en este caso los elementos que más atención precisan en cuanto a su disposición son el interruptor, el Arduino Nano y el módulo XBee.

  En primer lugar el interruptor debe tener un acceso rápido para facilitar una rápida acción en caso de emergencia.
  En segundo lugar el Arduino Nano se debe disponer de tal forma que permita la conexión del cable programador mini USB-B.

  Ambos elementos van a ser manipulados con frecuencia y esto no debe afectar a otros elementos de la placa por lo que hay que dejar un área disponible para la correcta manipulación.

  En tercer y último lugar, el módulo XBee tiene una serie de características especiales de colocación, consecuencia de su función. La transmisión de señales por vía radio frecuencia es sensible a perturbaciones e interferencias que pueden reducir la efectividad o impedir su funcionamiento. No se deben situar elementos metálicos en el área frente al módulo de comunicación, ni siquiera una pista pasando por delante de ella.

Estos 3 elementos se han dispuesto junto a distintos lados de la placa para un acceso rápido de los primeros y para evitar interferencias en el último.

Por otro lado, en la medida de lo posible, se debe distribuir el peso de forma adecuada para acercar el centro de masas lo más posible al centro de la placa, así como para evitar la curvatura de la placa.

- En la aplicación a la que se destina el dron, la placa va a sufrir un gran estrés térmico y mecánico que puede afectar a la integridad de las soldaduras, por eso se deben seguir una serie de pasos para realizarla correctamente.

  En primer lugar los elementos deben estar bien verticales a la hora de soldarlos ya que la posterior manipulación es imposible, o acaba en rotura, o requiere de desoldar el elemento.

  Se debe aplicar el soldador en primer lugar sobre la patilla del pin a soldar, para elevar su temperatura y a la hora de aplicar el estaño no sea como un simple pegamento si no que se logre una aleación entre la patilla y el estaño, esto nos aporta una gran rigidez.

  Hay que tener cuidado al aplicar el soldador en la patilla de los componentes ya que al ser el metal tan buen transmisor del calor, la temperatura del componente en sí se puede elevar hasta inutilizarlo. La soldadura debe ser una cuestión de 2 segundos más o menos.

  Tanto la escasez de estaño como el exceso del mismo se traducen en una mala soldadura, la huella de un aplique de estaño bien realizado no debe tener forma esférica por exceso, ni escasear. Incluso en el color de la soldadura se puede percibir un buen trabajo ya que al realizarse correctamente el resultado es un color metálico brillante.

  Una vez realizadas todas las soldaduras se le ha sometido a la placa a una serie de vibraciones producidas manualmente, haciendo de ensayo de vibraciones, no obstante para su producción en masa sería necesario un proceso automático de verificación.

  Para lograr establecer la placa a 45^{O} con las aspas se realizó un juego con ángulos rectos que aseguraba una buena posición. Se realizaron las marcas en la placa buscando los huecos que verificaran un agarre correcto y la posición de 45^{O}. A continuación con el taladro del laboratorio se realizaron los agujeros y se ensambló todo el montaje en la estructura.

- A continuación se incluyen los diseños finales del esquemático y del archivo PCB.

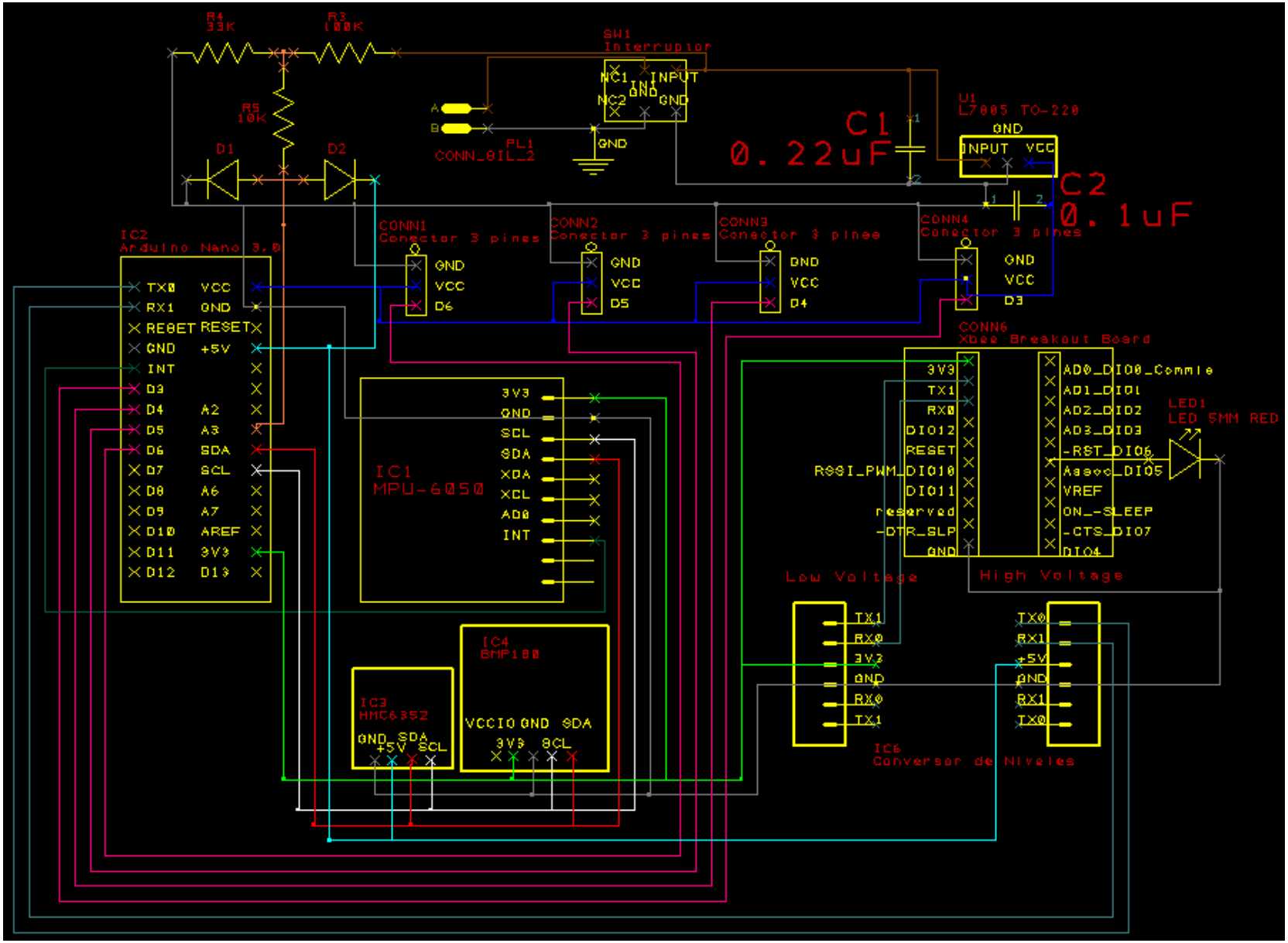


Ilustración 39 Diseño esquemático 1.0 realizado mediante Design Spark

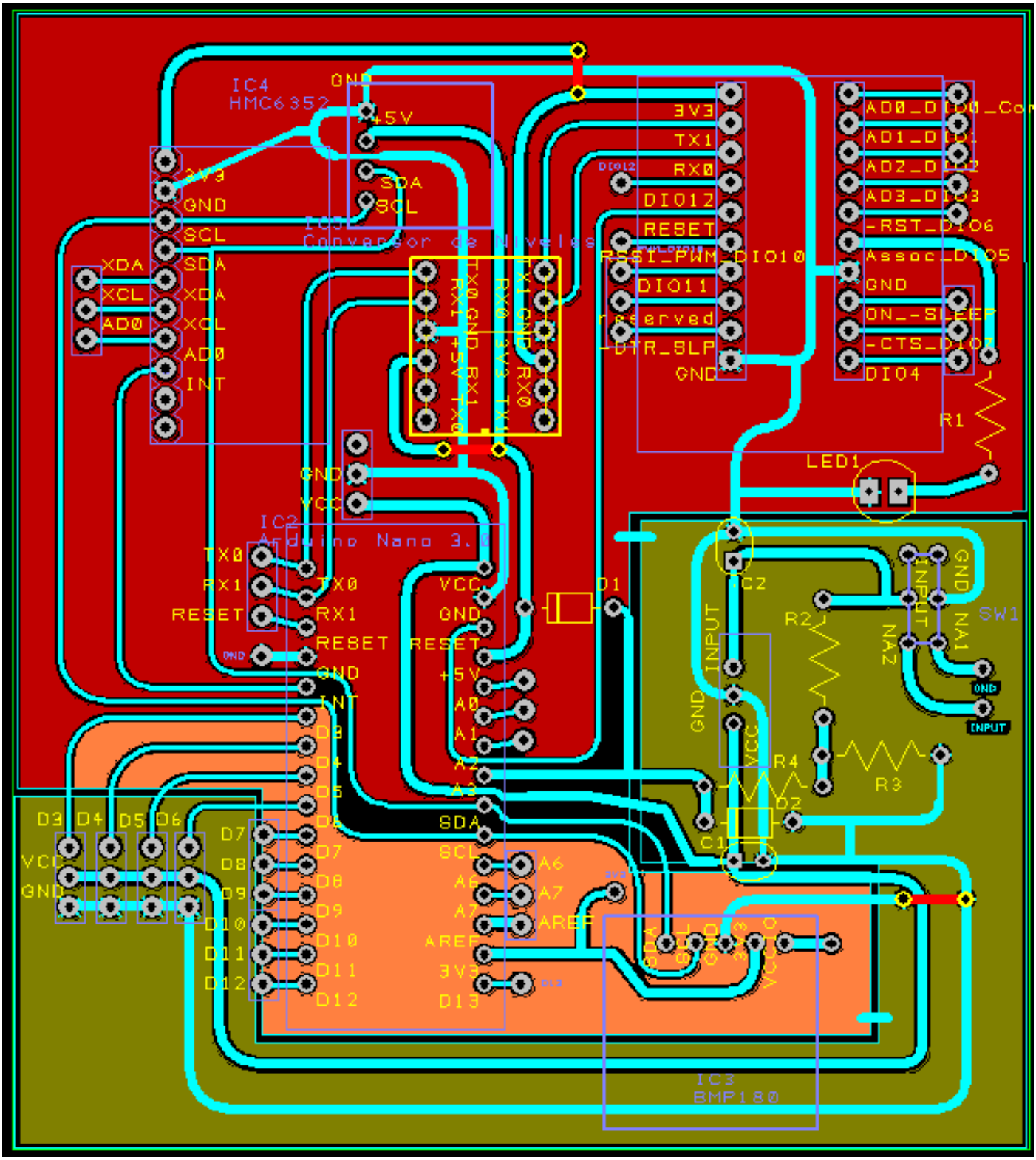
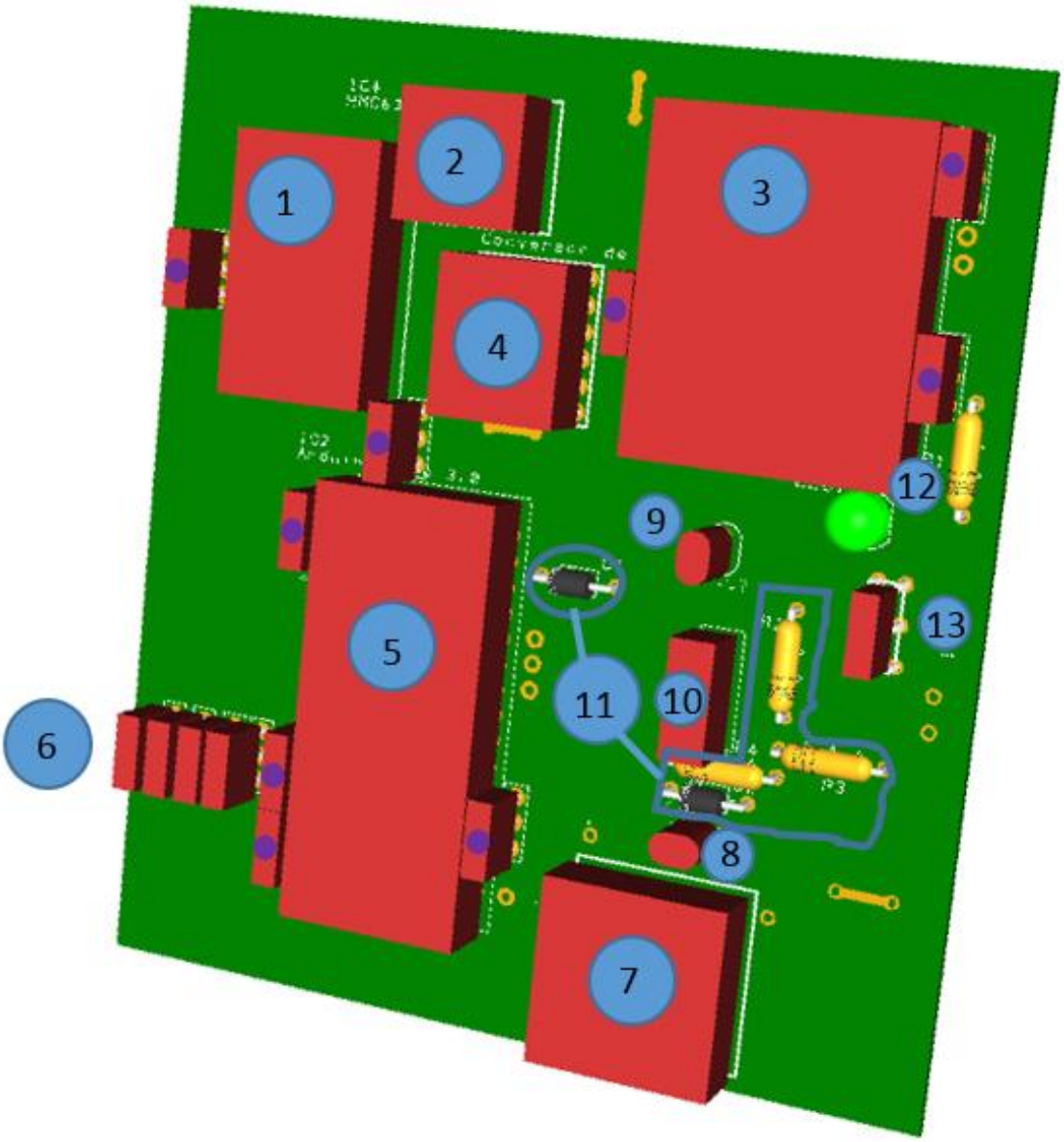


Ilustración 40 Diseño Placa de Circuito Impreso 1.0 mediante Design Spark

Se incluye a continuación una simulación el estado final del PCB realizado con el propio programa Design Spark.



1 – MPU 6050 Inercial Measurement Unit

2 – HMC6352 Compass Module

3 – Xbee ZB Breakout Board

4 – Logic Level Converter

5 – Arduino Nano v3.0

6 – Servomotor plugs.

7 – BMP180

8 – Condensador de salida del regulador de tensión

9 – Condensador de entrada del regulador de tensión

10 – Regulador de tensión. TO 220 – L7805C.

11 – Circuito medidor de batería

12 – Associate Led

13 – Interruptor

- Pines libres

3.4.7.1 Observaciones de la placa

En el momento en el que se fabricó la placa se procedió a la soldadura de los componentes. Como el proyecto se va a destinar a la universidad, la inclusión de los elementos más caros, los otros PCB's, como el microprocesador o los sensores, se realiza sobre zócalos para poder extraerlos si fuera necesario; para sustituirlos en caso de fallo o si fuera requiriera su uso en otra aplicación.

Al proceder con las labores de soldadura se observó cómo se había producido ciertos errores en el diseño.

- La información utilizada para diseñar la distancia entre las filas de pines de la placa para adaptar el módulo Xbee no coincidía con el módulo utilizado finalmente. Para solucionar el diseño hubo que taladrar de nuevo los agujeros y conectar las pistas.

- Se produjo un fallo en el diseño de la disposición de los pines del elemento interruptor y hubo que ajustar el agujero de nuevo con taladro.

- En el diseño de la huella de Arduino Nano en un inicio se realizó el correspondiente a Arduino Nano v2.3 hasta esclarecer la versión que se utilizaría finalmente, que resultó ser Arduino Nano v3.0. Se comprobó la correspondencia de los pines y a primera vista se fijó que eran iguales, pero tras las pruebas de funcionamiento ha resultado no ser así.
  En Arduino Nano v2.3 los pines 19…26 se corresponden con los pines analógicos A7…A0 respectivamente.
  En cambio en Arduino Nano v3.0 los pines 19…26 se corresponden con los pines analógicos A0…A7 respectivamente.

- Las pistas del conversor de niveles están mal diseñadas, ya que las conexiones entre este y Arduino están intercambiadas. El pin TX0 del conversor de niveles debe ser conectado con el pin de lectura del Arduino, RX0. De la misma manera el pin RX1 del conversor de niveles debe ser conectado con el pin de escritura del Arduino TX1.

- Se presenta un problema de comunicación entre el Arduino Nano y el ordenador pero no es por culpa del diseño de la placa. Se comprueba su funcionamiento con el director del proyecto y verifica que no funciona.

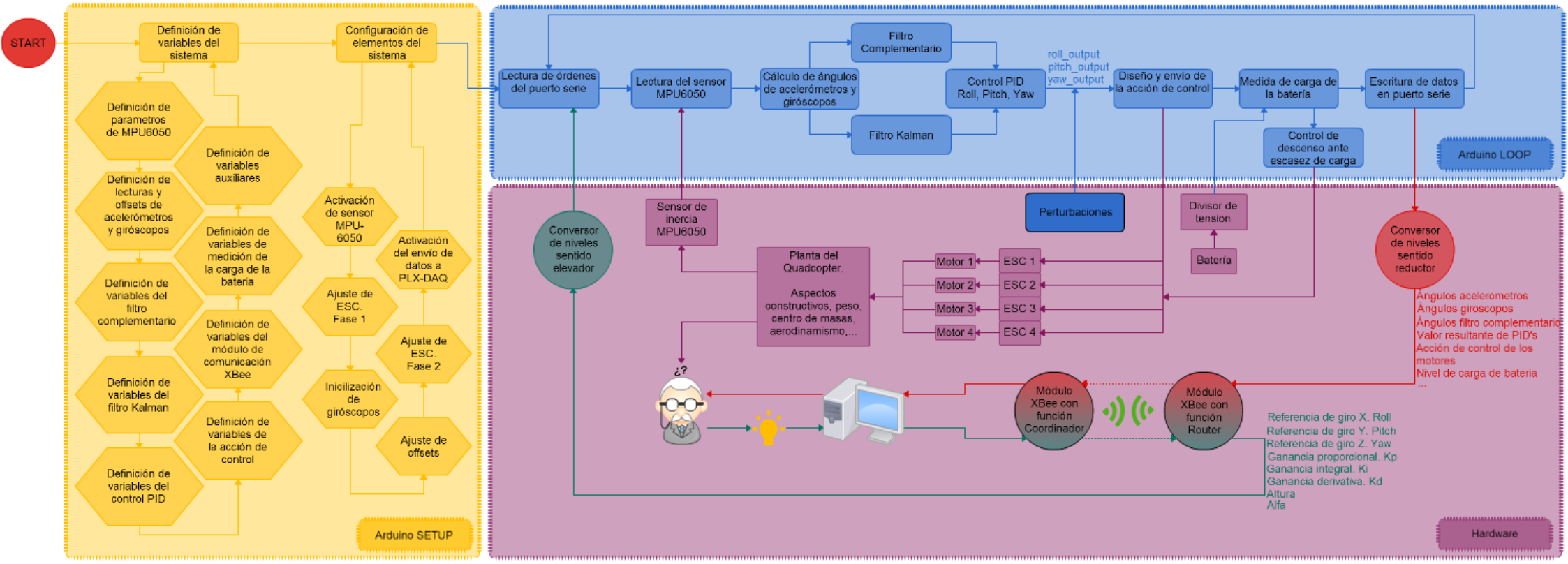
Mientras se espera a recibir un nuevo ejemplar de Arduino Nano, se dispone la protoboard en la estructura del dron enganchado a una de sus extremidades. En la protoboard se añade el sensor MPU 6050 y el Arduino UNO, utilizado para las pruebas. Al incorporar el sensor en la protoboard el sistema de referencia del sistema coincide con la dirección de las "alas" y se produce un cambio en la tipología del sistema que pasa a ser en forma de cruz +.

Analizados todos los errores de la placa se propone la realización de una versión 2.0 que por falta de tiempo no se presenta.

3.4.8 Desarrollo de la programación

El código de programación incluye los siguientes aspectos.

3.4.8.1 Organigrama de programación



3.4.8.2 Lectura de los sensores

El desarrollo del código de programación se realiza con el Arduino UNO como placa de pruebas y comienza con su comunicación con el sensor MPU 6050, todo ello está afianzado sobre una protoboard para facilitar la movilidad.

Para la comunicación entre estos dos elementos y demás sensores que se quisieran y pudieran añadir se utiliza el bus I^2C (Inter – Integrated Circuit). I^2C, también denominado TWI (Two Wire Interface) es un bus creado por Phillips en 1970 para reducir la utilización de cables en el conexionado de sus circuitos.

Tan solo dispone de dos conexiones, SDA por donde se transmiten los datos, y SCL por el que se transmite una señal de reloj. Se pueden conectar un total de 112 dispositivos con una distancia total máxima de 1 metro. [22]

Arduino UNO tiene sus conexiones I^2C en sus pines A4 a A5 [23]

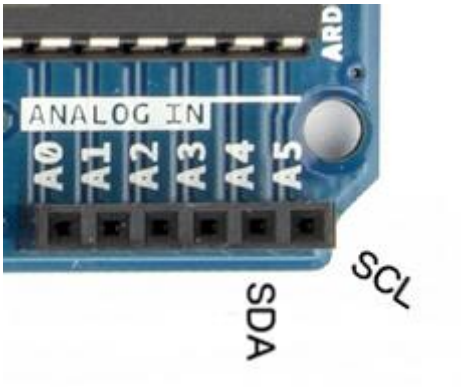


Ilustración 41. Localización de los pines de comunicación I^2C.
http://tronixstuff.com/2010/10/20/tutorial-arduino-and-the-i2c-bus/

Si se observa la ilustración 41 se puede ver como los pines 3 y 4 del sensor MPU 6050 se rotulan como SCL y SDA respectivamente, es entre estos terminales entre los que se establece la comunicación.

A la hora de realizar el diseño del Design Spark se ha tenido en cuenta que el dispositivo final a utilizar en el proyecto es el Arduino Nano v3.0, el cuál según los datos proporcionados por Arduino dispone de la comunicación I^2C a través de las mismas salidas que el Arduino UNO. [24]

Arduino UNO/Nano		Inter-Integrated Circuit
A4	← · →	SDA
A5	← · →	SCL

Se han verificado las funciones de los puertos A4 y A5 con el mapa de pines del procesador ATmega328. [25]

Realizadas las conexiones se ha proseguido con la comunicación software de ambos elementos.

Para establecer la configuración necesaria del sensor MPU 6050 se ha incluido un código desarrollado por Krodal en junio de 2012. En este código se definen una serie de registros, los

pines del sensor, la dirección del bus I2C y la reestructuración de registros. Se incluye el código de comunicación con el sensor en el *Anexo 1*.

Ya en el bucle de configuración del programa se inicia la comunicación TWI con la orden Wire.begin(). Al dejar el campo entre paréntesis en blanco se configura el dispositivo como maestro, para establecer cualquier dispositivo como esclavo se debe incluir entre paréntesis la dirección de 7 bits que le corresponda. [26]

Una vez ejecutada la orden anterior, el sensor envía los valores medidos de forma continua, en este momento es necesaria la reestructuración de registros. El MPU 6050 guarda la información del byte alto en la dirección baja mientras que Arduino trabaja al revés, con el fin de interpretar la información correctamente se invierte el orden de los registros alto y bajo.

Ordenados los registros, los valores observados en el monitor serie se pueden observar en la siguiente ilustración.

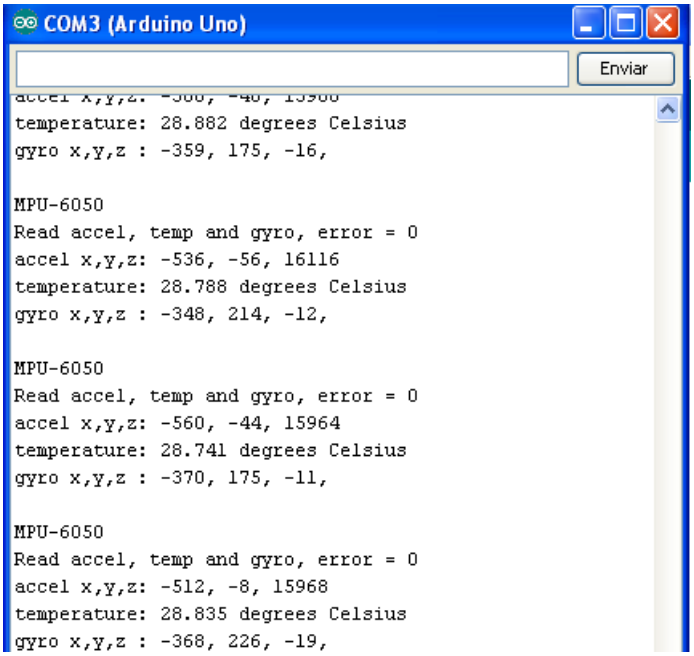


Ilustración 42 Monitor serie Arduino. Valores sin escalar.

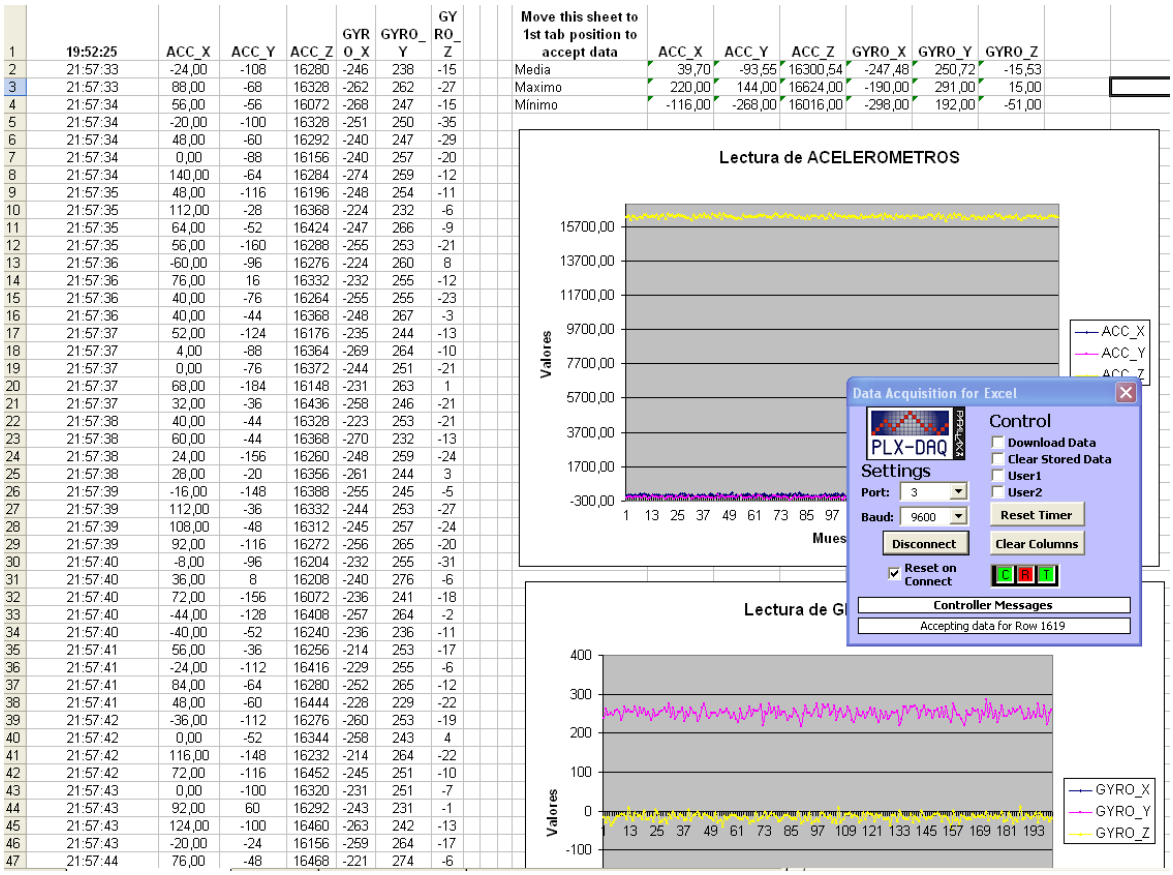
Como se puede observar los valores recibidos por el puerto serie de Arduino tienen un rango de hasta 16384, cuando el eje en cuestión está en vertical, en este caso el eje Z; experimentando toda la fuerza de gravedad. Este valor es la cantidad de LSB que representan una magnitud de esfuerzo gravitacional experimentado por el eje.

3.4.8.3 Exportación de los datos mediante Parallax-DAQ

Se ha añadido en la programación una serie de líneas de código que permiten la exportación de los valores que llegan por el monitor serie de Arduino a un documento Excel.

Esta herramienta se trata concretamente del software Parallax Data Acquisition tool (PLX-DAQ), un macro para Microsoft Excel que guarda los datos de cualquier microcontrolador Parallax a tiempo real. Dispone de funcionabilidad para hasta 26 canales diferentes. [31]

El código de programación de la escritura en Excel se incluye en el *Anexo 2*.



	19:52:25	ACC_X	ACC_Y	ACC_Z	GYRO_X	GYRO_Y	GYRO_Z
2	21:57:33	-24,00	-108	16280	-246	238	-15
3	21:57:33	88,00	-68	16328	-262	262	-27
4	21:57:34	56,00	-56	16072	-268	247	-15
5	21:57:34	-20,00	-100	16328	-251	250	-35
6	21:57:34	48,00	-60	16292	-240	247	-29
7	21:57:34	0,00	-88	16156	-240	257	-20
8	21:57:34	140,00	-64	16284	-274	259	-12
9	21:57:35	48,00	-116	16196	-248	254	-11
10	21:57:35	112,00	-28	16368	-224	232	-6
11	21:57:35	64,00	-52	16424	-247	266	-9
12	21:57:35	56,00	-160	16288	-255	253	-21
13	21:57:36	-60,00	-96	16276	-224	260	8
14	21:57:36	76,00	16	16332	-232	255	-12
15	21:57:36	40,00	-76	16264	-255	255	-23
16	21:57:36	40,00	-44	16368	-248	267	-3
17	21:57:37	52,00	-124	16176	-235	244	-13
18	21:57:37	4,00	-88	16364	-269	264	-10
19	21:57:37	0,00	-76	16372	-244	251	-21
20	21:57:37	68,00	-184	16148	-231	263	1
21	21:57:37	32,00	-36	16436	-258	246	-21
22	21:57:38	40,00	-44	16328	-223	253	-21
23	21:57:38	60,00	-44	16368	-270	232	-13
24	21:57:38	24,00	-156	16260	-248	259	-24
25	21:57:38	28,00	-20	16356	-261	244	3
26	21:57:39	-16,00	-148	16388	-255	245	-5
27	21:57:39	112,00	-36	16332	-244	253	-27
28	21:57:39	108,00	-48	16312	-245	257	-24
29	21:57:39	92,00	-116	16272	-256	265	-20
30	21:57:40	-8,00	-96	16204	-232	255	-31
31	21:57:40	36,00	8	16208	-240	276	-6
32	21:57:40	72,00	-156	16072	-236	241	-18
33	21:57:40	-44,00	-128	16408	-257	264	-2
34	21:57:40	-40,00	-52	16240	-236	236	-11
35	21:57:41	56,00	-36	16256	-214	253	-17
36	21:57:41	-24,00	-112	16416	-229	255	-6
37	21:57:41	84,00	-64	16280	-252	265	-12
38	21:57:41	48,00	-60	16444	-228	229	-22
39	21:57:42	-36,00	-112	16276	-260	253	-19
40	21:57:42	0,00	-52	16344	-258	243	4
41	21:57:42	116,00	-148	16232	-214	264	-22
42	21:57:42	72,00	-116	16452	-245	251	-10
43	21:57:43	0,00	-100	16320	-231	251	-7
44	21:57:43	92,00	60	16292	-243	231	-1
45	21:57:43	124,00	-100	16460	-263	242	-13
46	21:57:43	-20,00	-24	16156	-259	264	-17
47	21:57:44	76,00	-48	16468	-221	274	-6

Move this sheet to 1st tab position to accept data

	ACC_X	ACC_Y	ACC_Z	GYRO_X	GYRO_Y	GYRO_Z
Media	39,70	-93,55	16300,54	-247,48	250,72	-15,53
Maximo	220,00	144,00	16624,00	-190,00	291,00	15,00
Mínimo	-116,00	-268,00	16016,00	-298,00	192,00	-51,00

Ilustración 43 Lecturas del MPU-6050 en reposo

Se habrán percatado de los elevados valores que se muestran en la imagen, estos valores son los recibidos del sensor MPU 6050 cuyas características deben ser distinguidas entre las del acelerómetro y el giróscopo.

3.4.8.4 Ajuste de la señal de cada sensor

En un principio se reprodujeron 2000 valores de cada componente en el documento Excel estando el sensor reposando en horizontal, con los datos se hizo una media aritmética para obtener el offset. Estos cálculos aritméticos pueden observarse en la parte superior derecha de la ilustración 43. El valor de la media se asignó a una constante del programa, se sustrajo a las medidas procesadas y se comenzó a realizar los ensayos de precisión.

Para aprovechar las funciones de Excel se realizan gráficas con los valores y así observar con mayor facilidad las variaciones en cada momento.

La protoboard sobre la que reposan el Arduino UNO y el sensor MPU 6050 se coloca sobre una escuadra para asegurar linealidad y rectitud. Se hace uso de un transportador de ángulos asentado junto al borde de la mesa para asegurar verticalidad y permitir la maniobra del escalímetro girando sobre el eje del transportador. Se han realizado una serie de ensayos para verificar la precisión de la medida.

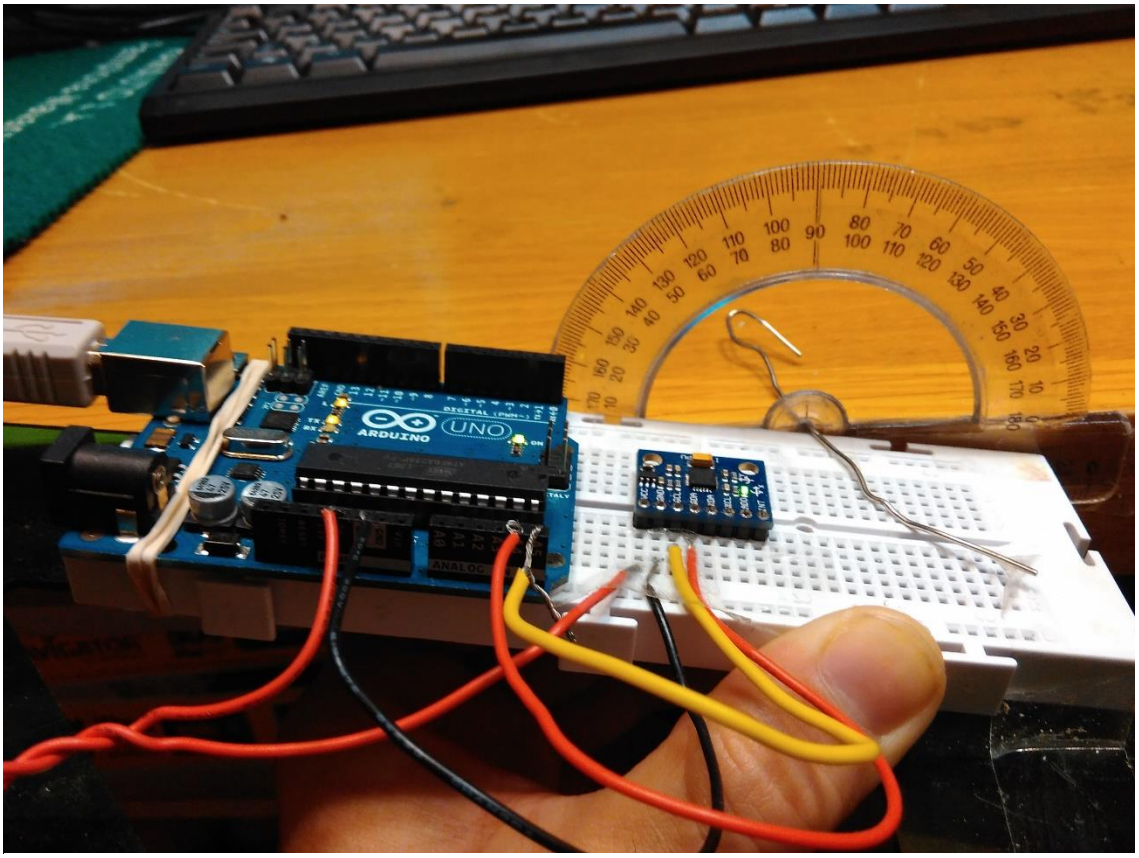


Ilustración 44 Montaje de protoboard en escuadra y medida con transportador de ángulos.

Para verificar el cumplimiento de las predicciones se comparan periódicamente los resultados obtenidos en Excel con las indicaciones de los utensilios utilizados.

3.4.8.4.1 Acelerómetro

Este aspecto se ha comentado con anterioridad pero se hace hincapié con la ayuda de la ilustración 43 en la que se observa el rango de medida del acelerómetro [-16384,16384]. Su configuración se ha dejado como la configuración de serie, con el registro AFS_SEL=0 establece que se alcanzará un rango de ±2g con una sensibilidad de 16384 LSB/g. En otras palabras, cada variación de una unidad digital será equivalente a una variación de 61 µg/LSB.

Se recuerda que para la obtención del ángulo de los acelerómetros se utiliza la tangente del cociente de sus componentes, al tratarse de un cociente la escala a la que se encuentren no influye y simplemente con la correcta expresión de las ecuaciones 1 y 2 con los offsets ya eliminados se obtiene el resultado. La sensibilidad del acelerómetro no supone un problema.

En un inicio se añadió una constante para ajustar los resultados a los objetivos ya que como se muestran a continuación difieren un poco de la realidad dando los valores por lo bajo. Se percibe mejor en ángulos altos. No obstante la aplicación de esta medida hizo que se perturbara en mayor medida el rango de valores cercanos a 0. Para lograr un equilibrio bueno en todo el rango de valores útiles, y asumiendo que el dron no va a sobrepasar una inclinación de 40º salvo perturbaciones fuertes, finalmente se optó por eliminar esta constante de escalado.

fx `=ATAN(1.101503552*(C47-$Z$S)/(D47-$AA$4))*180/PI()`

Conforme los datos son recibidos se les aplica la ecuación antes presentada para obtener los valores del ángulo en Excel. Con posterioridad este cálculo se traspasaría ya directamente al código de Arduino.

1	ACC_Y	ACC_Z	GYRO_X	GYRO_Y	GYRO_Z	ANG_X	ANG_Y
2	-748	16984	1560	598	84	-1,89560175	0,29
3	-584	15616	-404	414	-107	-1,54167724	-0,54
4	-904	15916	-281	303	-15	-2,50067163	0,445
5	-888	16352	-287	-84	2	-2,38764906	1,107
6	-816	16020	-306	-736	43	-2,21518755	1,755
7	-868	16064	-276	759	-37	-2,36821866	2,007
8	-912	16352	-287	1170	-48	-2,45973253	1,937
9	-780	16212	-301	969	-57	-2,08054543	2,22
10	-872	16328	-314	992	-80	-2,34294057	1,458
11	-824	16112	-308	21	17	-2,22724891	1,794
12	-888	16296	-331	698	-50	-2,39564347	2,269
13	-836	16472	-319	182	-1	-2,21559715	1,864
14	-836	16112	-310	341	-27	-2,26382351	1,843
15	-768	16328	-322	472	-25	-2,03002739	1,855
16	-808	16172	-300	87	-30	-2,17060494	2,189
17	-928	16244	-311	260	-16	-2,52402568	1,623
18	-880	16128	-328	-199	25	-2,39559698	1,927
19	-840	16180	-323	720	-37	-2,26669526	1,702
20	-868	16320	-305	318	-16	-2,332021	1,953
21	-728	16204	-334	387	-25	-1,92388515	1,081
22	-964	16204	-345	-231	21	-2,6391561	1,881
23	-860	16184	-300	614	-56	-2,32683802	1,556
24	-796	16120	-342	-48	-5	-2,1408658	1,732
25	-840	16192	-317	385	-10	-2,26505852	1,652
26	-868	16288	-317	-253	24	-2,33648515	1,872
27	-840	15952	-281	538	-50	-2,29824734	1,442
28	-924	16172	-303	293	-23	-2,5228281	1,447
29	-824	16224	-289	-95	-7	-2,21226716	1,455
30	-944	16148	-297	327	-10	-2,58727789	1,803

Ilustración 45 Primer ensayo de la medida de los acelerómetros del eje X.

Las ecuaciones comentadas en código se traducen debajo:

//Cálculo de ángulos acelerómetro

*ang_acc_x= atan2(const_experimental_escalado_x*Fy,(sqrt(pow(Fz,2)+pow(Fx,2))))*const_radian_ang_escalado;*

*ang_acc_y= atan2(const_experimental_escalado_y*Fx,(sqrt(pow(Fz,2)+pow(Fy,2))))*const_radian_ang_escalado;*

A continuación se muestran unas imágenes de los ensayos realizados. Cada escalón representa un aumento de 10 grados en la inclinación de la protoboard excepto el último escalón que es un aumento de 5 grados.

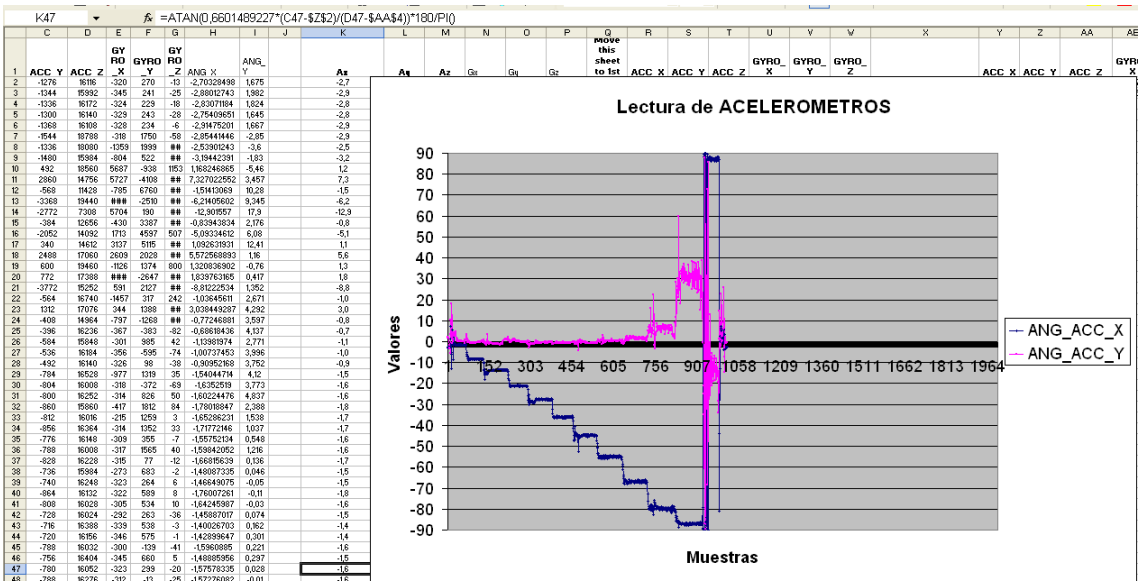


Ilustración 46 Ensayo realizado sobre el eje X sin constante de ajuste

En la ilustración anterior cuando la plataforma del acelerómetro se encuentra a 40^{O}, de las operaciones está resultando 29^{O}. Esta prueba fue la primera y no se le aplicó ningún ajuste.

En la siguiente imagen se comprueba el efecto de añadir la constante. Se puede observar como a pesar de la constante los ángulos de valores elevados difieren mucho de la realidad, marcando por ejemplo 71^{O} cuando la inclinación es de 80^{O}. A su vez los valores de ángulos cercanos a 0^{O} sobrepasan su referencia.

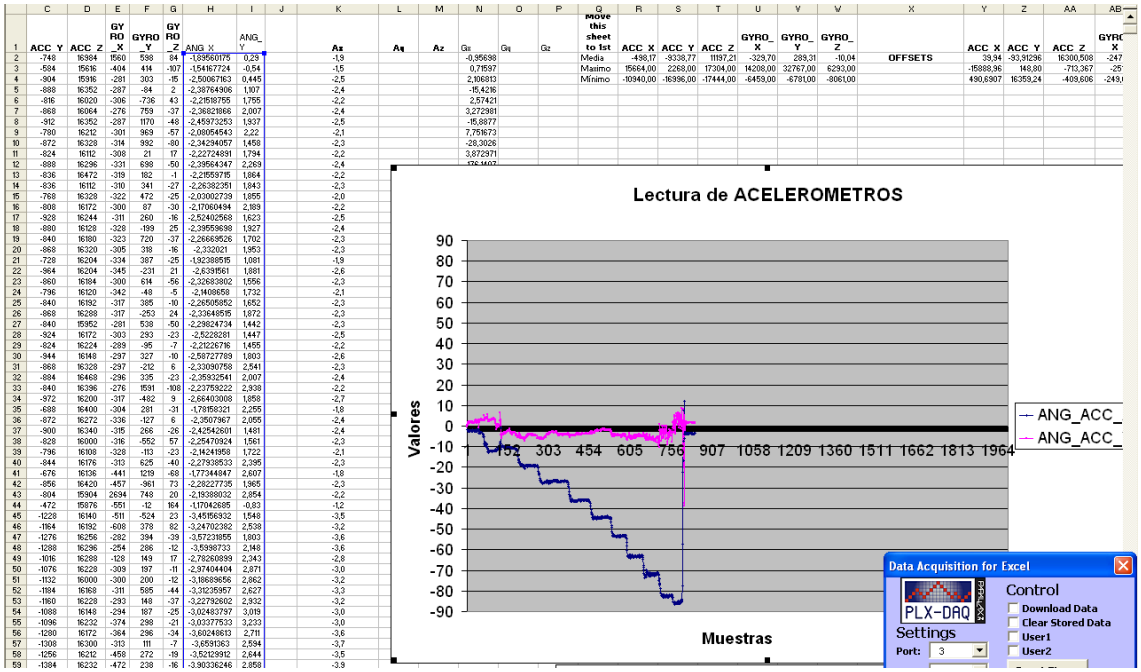


Ilustración 47 Ensayo del acelerómetro realizado sobre eje X con constante de ajuste.

Este efecto de un distinto error en función del ángulo en el que nos encontremos viene relacionado con la no linealidad de la función arcotangente con la que trabajamos, en la que 45^{O} del eje de ordenadas corresponde a 1 unidad en el eje de abscisas y para valores de X mayores de 5 los grados se disparan, con arctg(5)=78.69^{O}.

Hasta los 30^{O} la función arcotangente tiene un comportamiento muy lineal. De hecho en muchas ocasiones se utiliza el simple cociente entre sus componentes para calcular el ángulo, sin efectuar la función arcotangente. No obstante a partir de los 30^{O} la función presenta una curvatura que finaliza en una asíntota a 90^{O}.

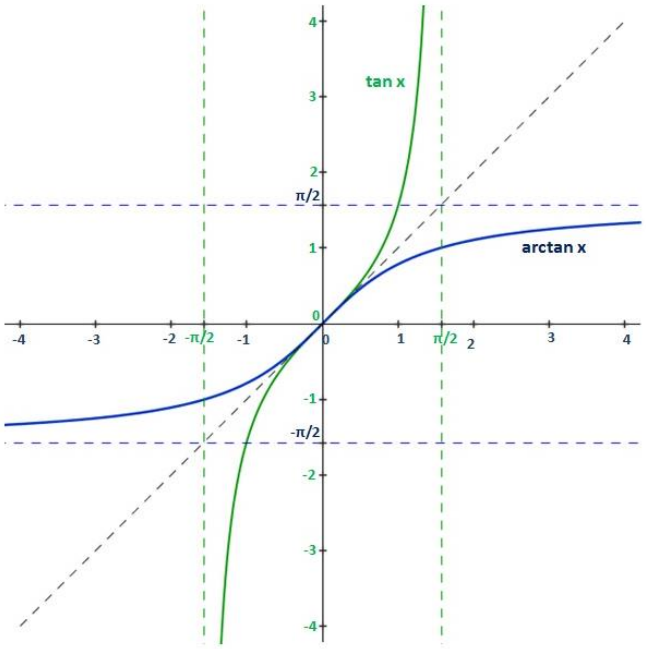


Ilustración 48 Función Arcotangente.
http://www.universoformulas.com/matematicas/trigonometria/arcotangente/

En el ensayo realizado sobre eje Y se han conseguido buenos resultados con la aplicación de la constante de ajuste, pero los resultados de este acelerómetro ya eran de serie más precisos que los del acelerómetro del eje X.

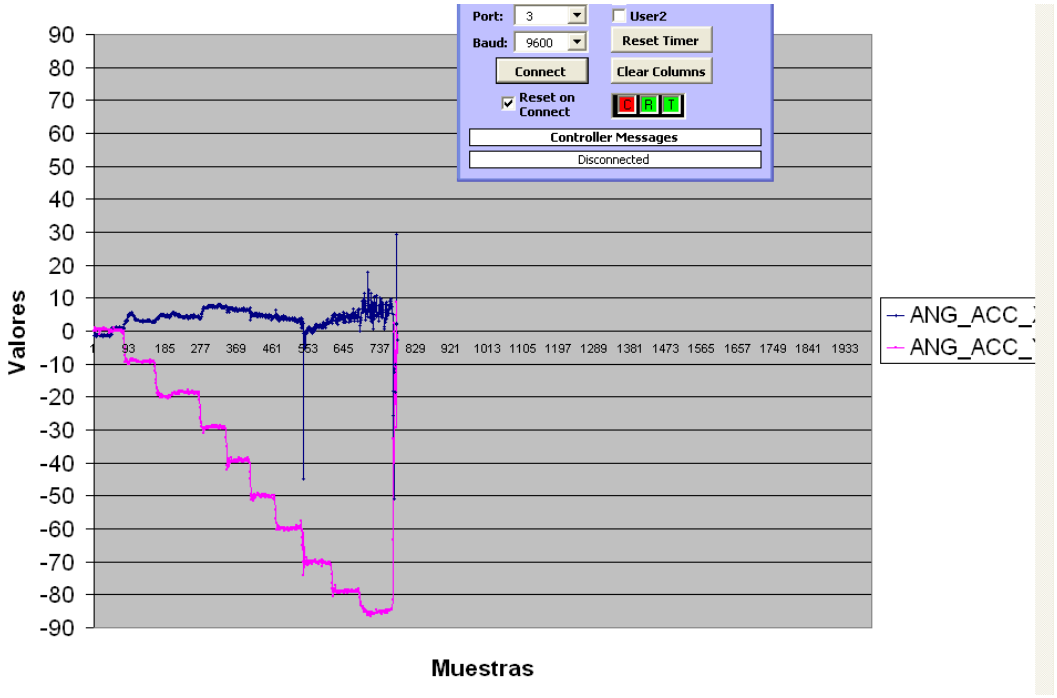


Ilustración 49 Ensayo realizado sobre el eje Y con constante de ajuste.

En la ilustración que se expone a continuación se observan las medidas fluctuantes de los acelerómetros antes de realizar el cálculo de los offsets. Una vez realizada la media de los 100 primeros valores este offset se le sustrae a los valores que se reciben a partir de ese instante eliminando el error que trae de serie el sensor.

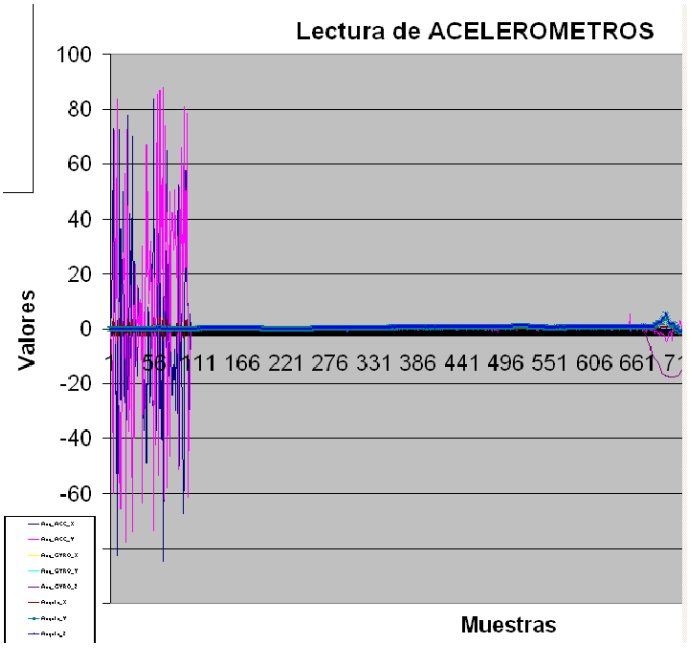


Ilustración 50 Cálculo de offsets y sustracción de los mismos

El código de programación se incluye en el *Anexo 3*.

3.4.8.4.2 Giróscopo

Para las pruebas con el acelerómetro esta medida de ajuste del offset fue suficiente, no obstante en el futuro nos encontramos con que este ajuste del offset era demasiado impreciso para el cálculo de los giróscopos.

Al hacer las pruebas de funcionamiento de los giróscopos, se obtenía un "drift" muy acusado. En las diversas pruebas el offset no se actualizaba. Quedaba una diferencia notable con el offset real, estando en horizontal los valores recibidos del sensor se alejaban con rapidez de la referencia de 0º.

Para disminuir el efecto del drift se ha añadido un ajuste automático del offset de los sensores al inicio de funcionamiento del dispositivo. En el momento de inicio del mismo, espera 1 segundo para equilibrar todas las constantes y una subrutina comienza a realizar una media aritmética con los 100 primeros valores generados por los sensores. El resultado final es almacenado en una variable denominada offset que es sustraída al valor del sensor en cada momento para conocer cualquier otra inclinación en la que se encuentre el dispositivo que no sea con la que ha sido calibrado.

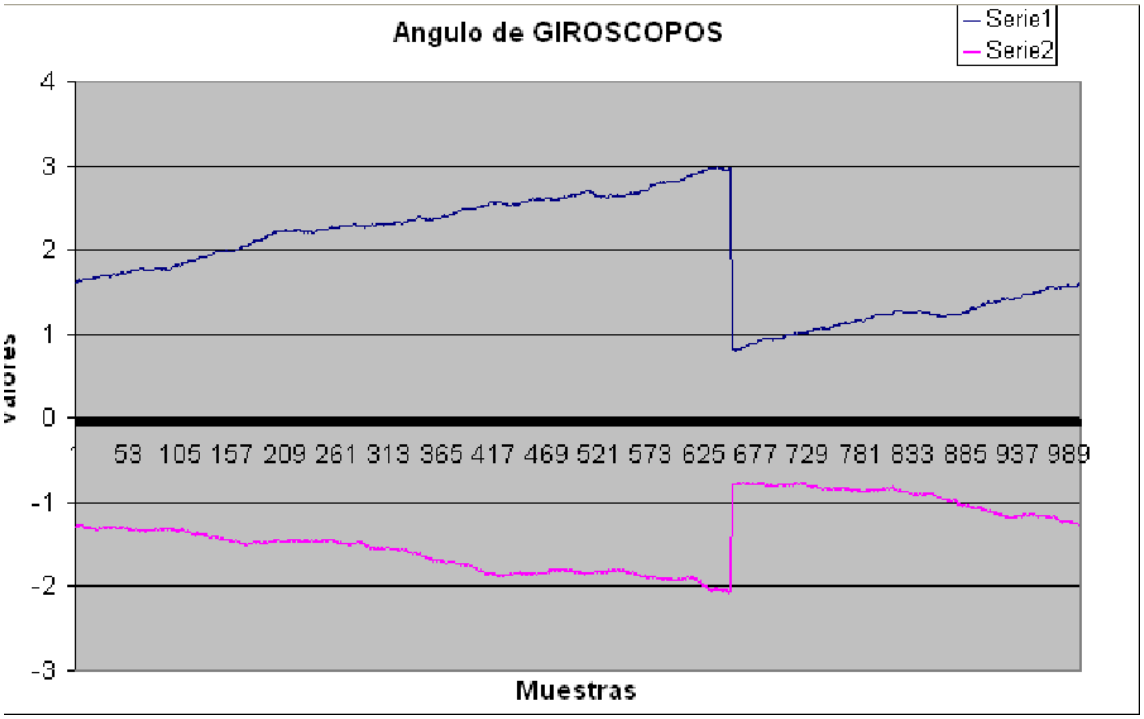


Ilustración 51 Drift con giróscopo en horizontal. Giróscopo X – Amarillo. Giróscopo Y – Cian. Modelo con subrutina.

Se incluye el código de programación en el *Anexo 3* junto con el del acelerómetro

Finalmente se optó por programar un cálculo automático directo en el programa, sin subrutina con un código bien optimizado reduciendo el tiempo de procesamiento. Se realiza la media de 50 valores al inicio de cada programa. Antes de comenzar a realizar la media de valores se mantiene la anterior dinámica de esperar unos milisegundos para descartar posibles medidas iniciales desequilibradas.

El código de programación del ajuste automático de offsets se incluye en el *Anexo 4*.

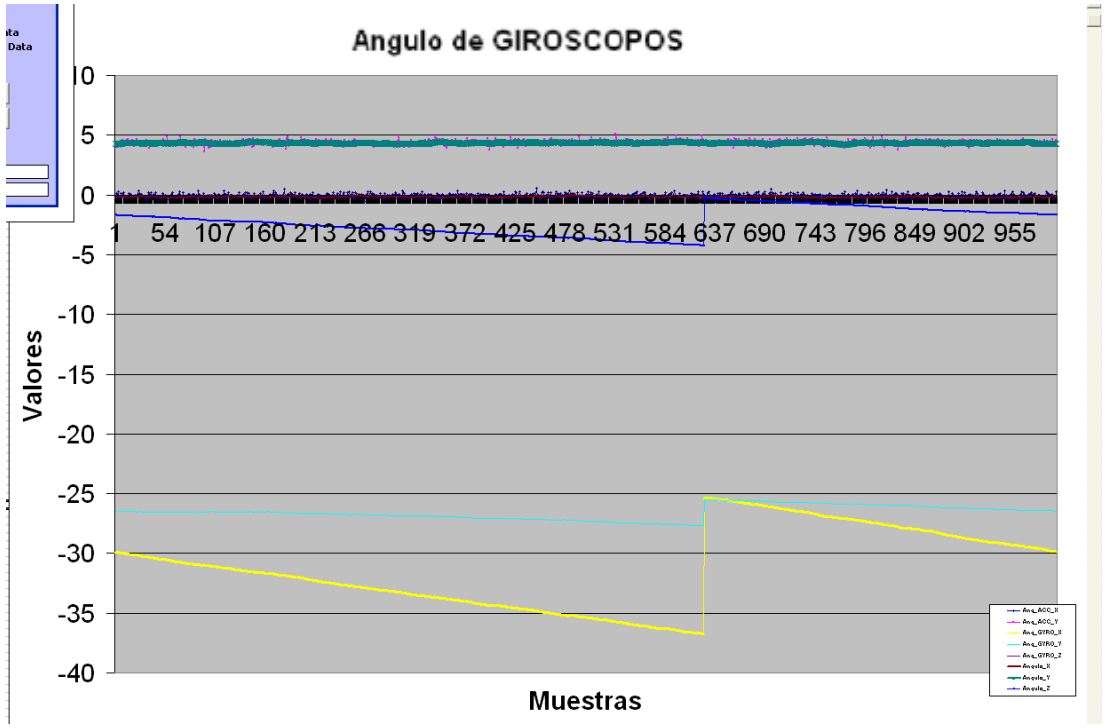


Ilustración 52. Drift con giróscopo en horizontal. Giróscopo X – Amarillo. Giróscopo Y – Cian. Modelo sin subrutina.

En la ilustración 52 el drift que se muestra es bastante acusado, la ilustración ha sido tomada con el dron en horizontal y tras esperar 2500 ciclos de programa. Esta desviación puede parecernos demasiado drástica pero el efecto de esta desviación será ignorado gracias al filtro complementario como se ha explicado con anterioridad.

El cálculo de los ángulos con la medida de los sensores se consigue mediante la ecuación 3. Como se ha explicado con antelación en Arduino debemos manejar el tiempo de ciclo del programa.

Es fundamental obtener el tiempo exacto que ha transcurrido desde que se ha producido la medición previa, ya que este tiempo será multiplicado por la velocidad angular para obtener el ángulo girado y un mal ajuste puede concluir en nefastos resultados.

Para realizar esta operación con precisión se hace uso de la función de Arduino *millis()* la cual nos reporta el número de segundos transcurridos desde el inicio del programa. Guardando el valor en una variable y restándosela a la medida *millis()* del siguiente ciclo se obtiene el tiempo transcurrido. Este tiempo transcurrido se asigna a la variable *dt* ya que hace referencia al diferencial de tiempo asignado en la integral de la velocidad angular.

Al multiplicar el diferencial de tiempo por la velocidad se debe tener en cuenta que el valor de *dt* está dado en mili segundos y nosotros necesitamos segundos, por lo que hay que ajustar la variable dividiendo entre mil.

Tras estas operaciones ya se tiene el resultado del ángulo desplazado, pero se debe tener en cuenta que este giro no es un resultado absoluto si no relativo a la posición previa. Por lo tanto hay que añadir la anterior posición angular para conocer la posición final.

$$\varphi = \varphi_0 + \omega \cdot t$$

Ilustración 53 Movimiento Circular Uniforme.
https://www.fisicalab.com/apartado/ecuaciones-mcu#contenidos

Finalmente el resultado se obtiene añadiendo el valor del ángulo calculado en el anterior ciclo de programa.

Se tiene en cuenta que al procesar los datos ofrecidos por los giróscopos, éstos están ajustados a una cierta escala. En nuestro caso la configuración que se ha utilizado es la que venía de serie con una capacidad de detectar velocidades de hasta 250 ^{o}/s.

El sensor de inercia dispone de ADCs (Analog-Digital Converter) de 16 bits. Lo que equivale a un rango de [-32768,32768]. Es decir, conforme se varíe la velocidad de giro de nuestros giróscopos van a reportar valores en ese rango de medidas. No obstante a nosotros nos interesa medir los ángulos, para ello realizamos la integral de la velocidad, pero estos valores de velocidad tienen un rango de [-250,250] por lo tanto los valores deben ser escalados.

Teniendo en cuenta el límite de velocidad establecido en $250\,{}^{o}/_{s}$, se consigue una precisión de:

$$\frac{250\,{}^{o}/_{s}}{32768\,LSB} = 7.6294 * 10^{-3}\ \frac{{}^{o}/_{s}}{LSB}$$

Expresado de forma inversa:

$$\frac{32768\,LSB}{250\,{}^{o}/_{s}} = 131.072\ \frac{LSB}{{}^{o}/_{s}}$$

Aplicando el ajuste del tiempo a segundos e incluyendo estos valores en la expresión principal, se genera una constante que denominamos sensibilidad.

$$\varphi = \varphi_0 + \omega * \frac{250\,{}^{o}/_{s}}{32768\,LSB} * \frac{1s}{1000ms} * dt(ms)$$

$$\varphi = \varphi_0 + \omega * 7.6294 * 10^{-6}\,\frac{{}^{o}/_{s}}{LSB} * \frac{1s}{ms} * dt(ms)$$

Con el trabajo de las unidades se puede apreciar el efecto de la integral, convirtiendo la velocidad angular en posición angular, al evaluar su efecto a lo largo del tiempo.

$$\varphi = \varphi_0 + \omega * 7.6294 * 10^{-6}\frac{o}{LSB} * dt$$

$$Sensibilidad\_gyroscopo\ =\ 7.6294 * 10^{-6}\frac{o}{LSB} *$$

A continuación se muestra un ejemplo del movimiento en un eje y su seguimiento de la referencia al volver a su posición de reposo. Se ve un aumento gradual a baja frecuencia que es el efecto de la deriva, de la cual no debemos preocuparnos mientras se mantenga entre unos márgenes razonables.

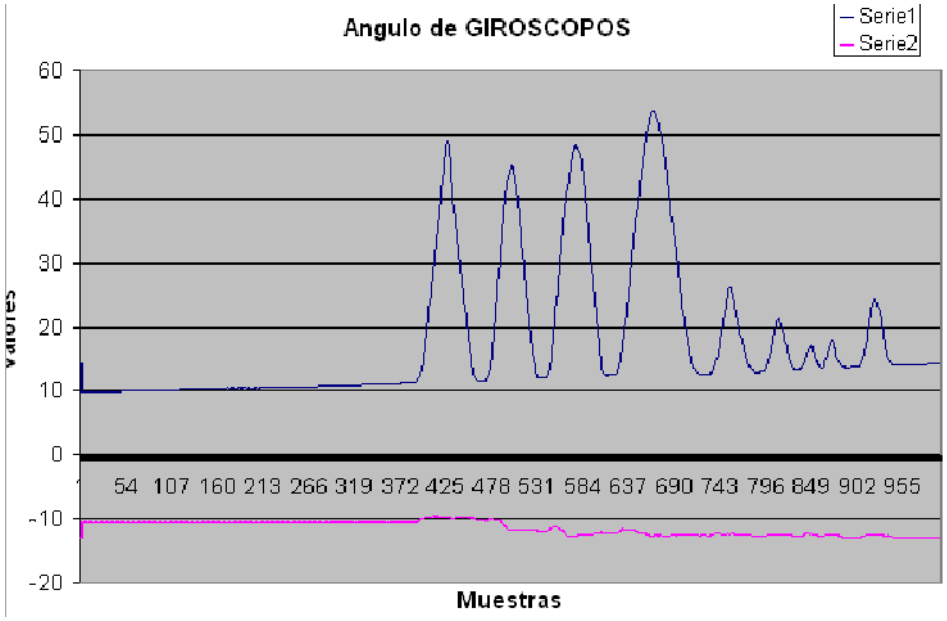


Ilustración 54 Ensayo del giróscopo realizado sobre eje X.

No obstante, si no se tiene en cuenta la deriva, el seguimiento del movimiento es muy acertado y preciso siguiéndolo con velocidad.

Sin embargo en ciertas ocasiones con movimientos muy rápidos como golpes los resultados se disparan, en este ejemplo la plataforma de pruebas se golpeó y se dejó en reposo. El comportamiento se observa entorno a la muestra 850.

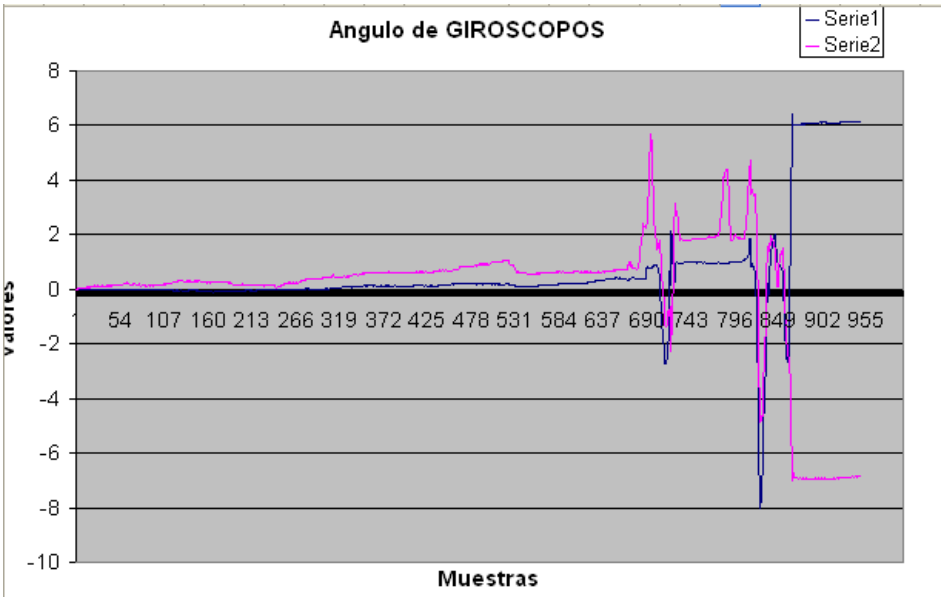


Ilustración 55 Ensayo de giróscopos ante vibración rápida.

3.4.8.5 Filtro complementario

La escritura del filtro complementario en Arduino es sencilla transcribiendo la ecuación 4. Se comenzaron a realizar pruebas con distintos valores de alfa, la constante que dicta el peso de cada componente.

El siguiente ensayo ha sido realizado en el mismo instante que la ilustración 54 e incluye el resultado del filtro complementario.

Tanto en la ilustración como en el resto del documento a partir de ahora, cuando se haga referencia a *angulo_x* se hará alusión al resultado del filtro complementario para el eje X. Y su análogo con el eje Y.

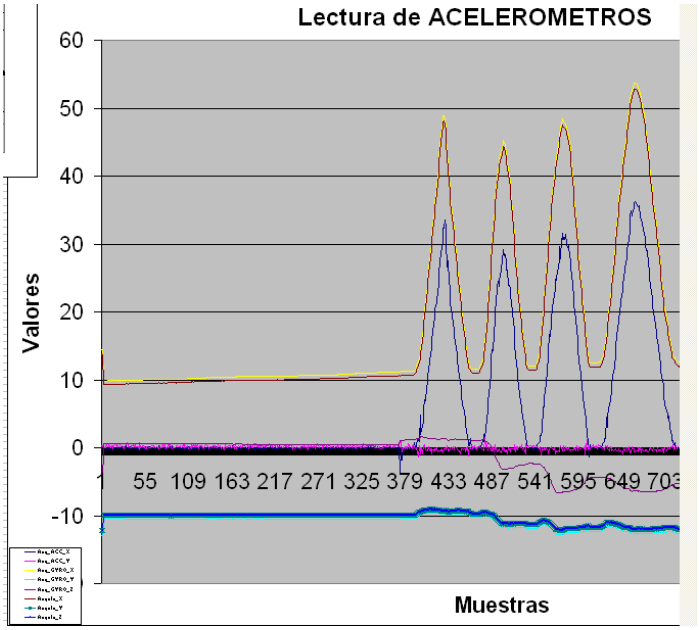


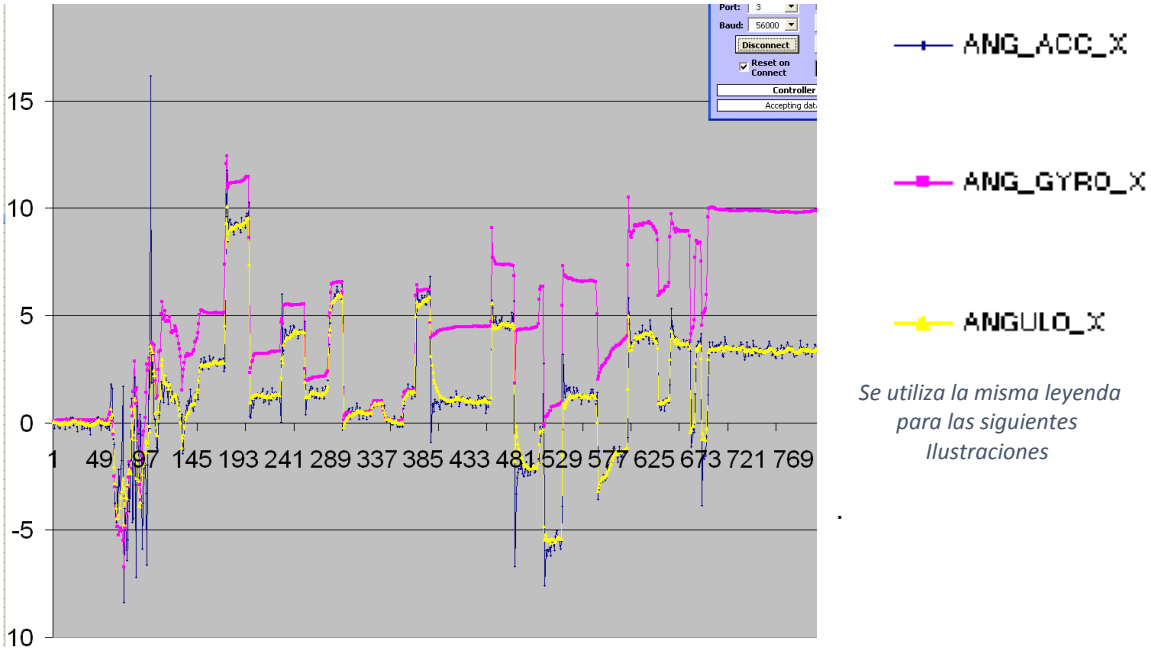
Ilustración 56. Ensayo de filtro complementario. Amarillo - Giróscopo X. Azul - Acelerómetro X. Marrón – angulo_x

Tal y como se muestra en la figura el resultado del filtro complementario sigue con precisión la referencia del giróscopo sin percibir las fluctuaciones de alta frecuencia del acelerómetro, no obstante la componente de baja frecuencia del giróscopo sigue estando presente en el resultado del filtro.

Tras muchos análisis descartando posibles fallos y ajustando un tiempo de muestreo muy preciso no se consiguió encontrar la solución y se continuó con la utilización de un código que tenía el filtro complementario incluido al que nos referiremos como código 2.

El código de programación se incluye en el *Anexo 5.*

Los estudios del valor de la constante del filtro se realizaron con este nuevo código y a continuación se analizan los resultados desde los valores más bajos a los más elevados.



Se utiliza la misma leyenda para las siguientes Ilustraciones

Ilustración 57 Ensayo de filtro complementario. Alfa = 0.80

En la ilustración 57 se percibe cómo los datos del giróscopo distan bastante de los del acelerómetro y a pesar de eso el resultado del filtro sigue con exactitud la referencia del acelerómetro, ignorando la deriva del giróscopo. En este ejemplo el resultado varía más de lo deseado en una situación de reposo como se observa en los últimos valores. Se debe disminuir la aportación del acelerómetro.

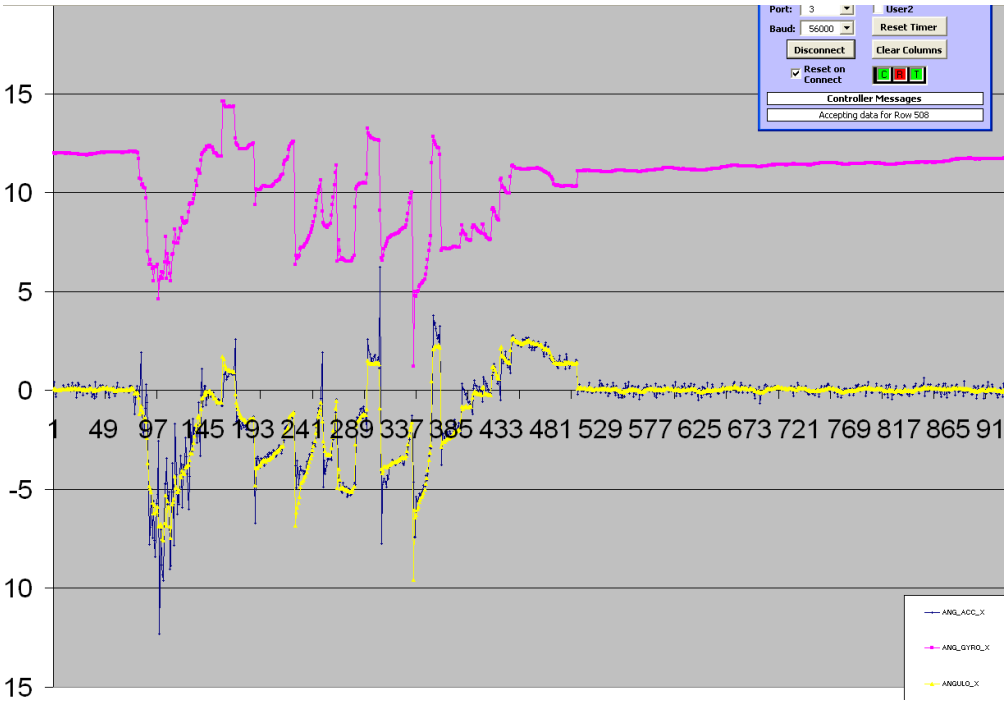


Ilustración 58. Ensayo del filtro complementario. Alfa = 0.85.

En la ilustración 58 las fluctuaciones del acelerómetro han disminuido sin poner en compromiso la velocidad ni referencia. Se continúa aumentando alfa buscando el valor más óptimo. Haciendo esto se consigue que el filtro paso alto admita menos frecuencias bajas y que el filtro paso bajo admita menos frecuencias altas.

Ilustración 59. Ensayo de filtro complementario. Alfa = 0.88

Se percibe un seguimiento más fino sin tantas variaciones sin comprometer la velocidad de respuesta.

La siguiente imagen corresponde con el resultado más óptimo encontrado, con un seguimiento preciso de la referencia del acelerómetro con el nivel de fluctuaciones del giróscopo.

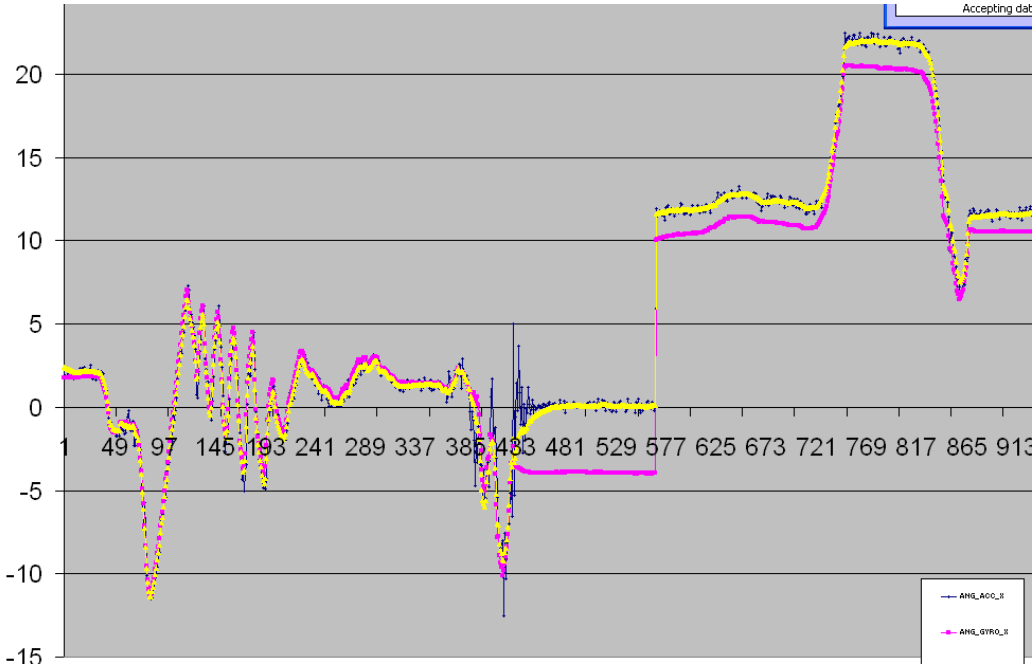


Ilustración 60. Ensayo de filtro complementario. Alfa = 0.90

Se continúa aumentando el valor de alfa para comprobar si pudiera existir una configuración más apropiada.

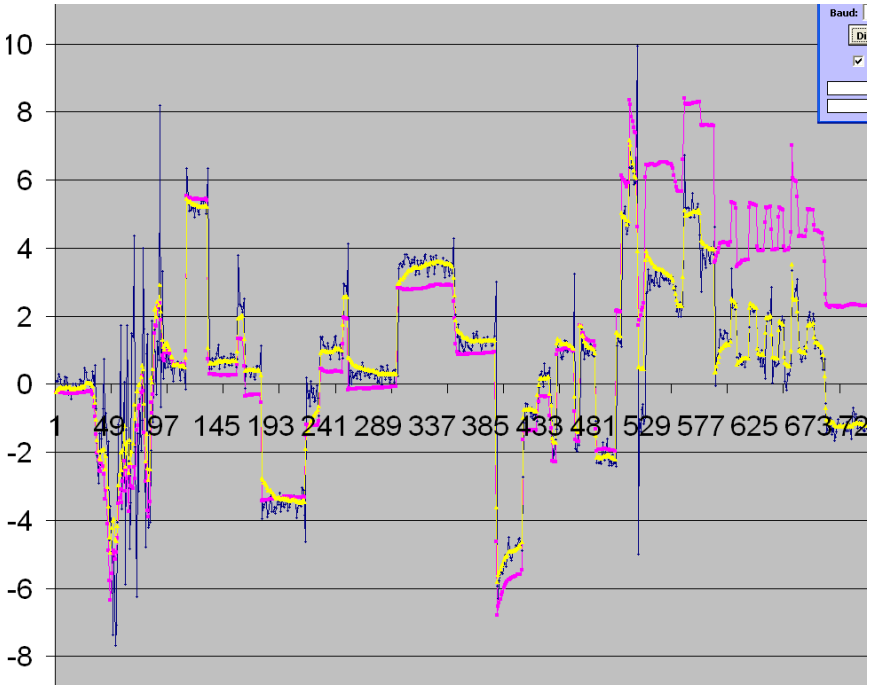


Ilustración 61. Ensayo de filtro complementario. Alfa = 0.91

En estas dos Ilustraciones se repara en un gradual aumento en el tiempo que tarda *angulo_x* en alcanzar la referencia del acelerómetro

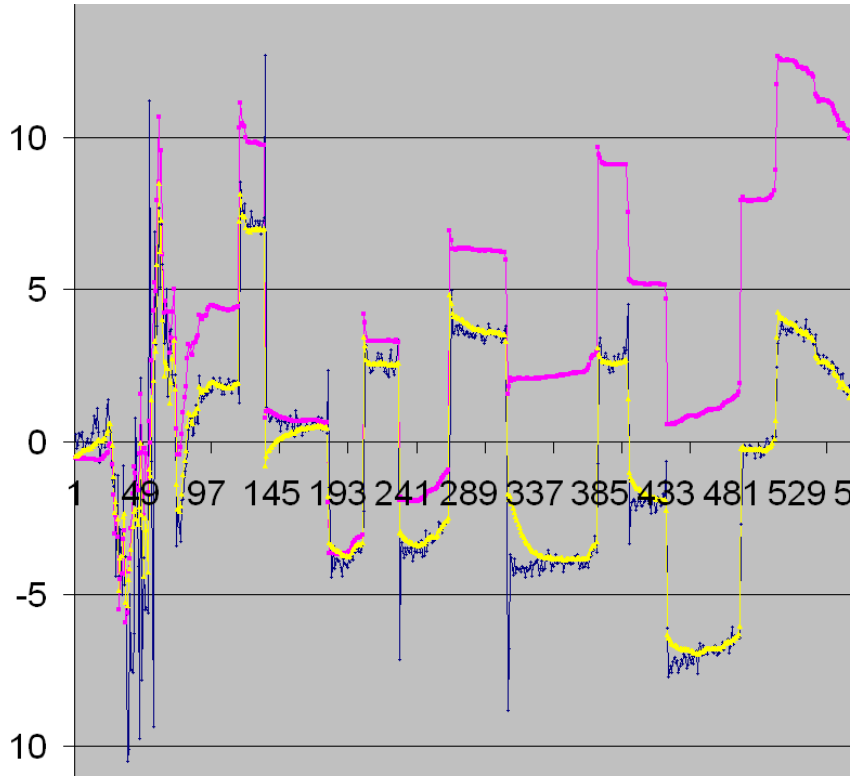


Ilustración 62. Ensayo de filtro complementario. Alfa = 0.92

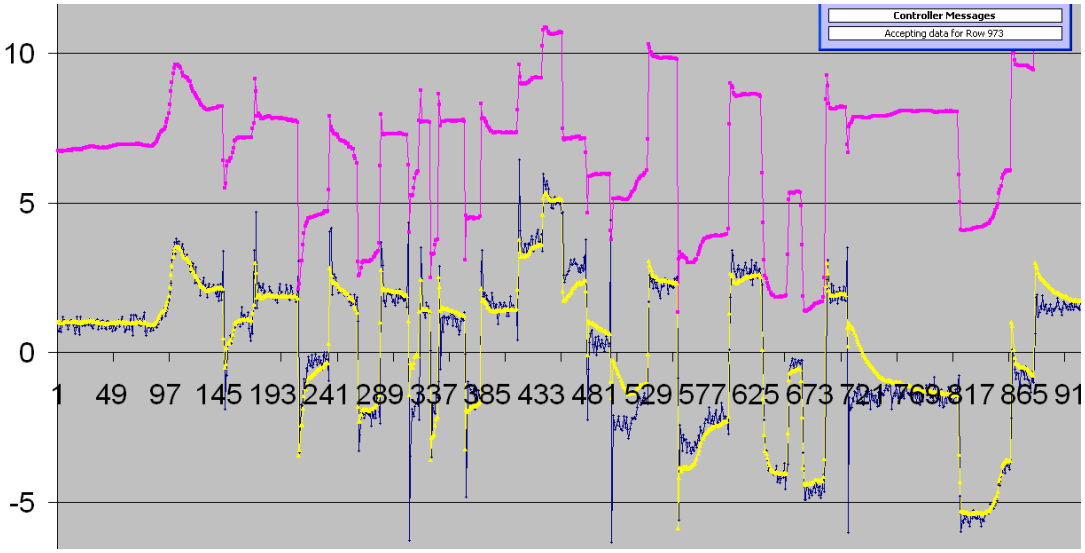


Ilustración 63. Ensayo de filtro complementario. Alfa = 0.95

En estas Ilustraciones el efecto de las fluctuaciones del acelerómetro es casi imperceptible, sin embargo el retraso de seguimiento de la señal es muy grande. En la segunda ilustración el resultado está completamente tergiversado al no valorar casi la influencia del acelerómetro.

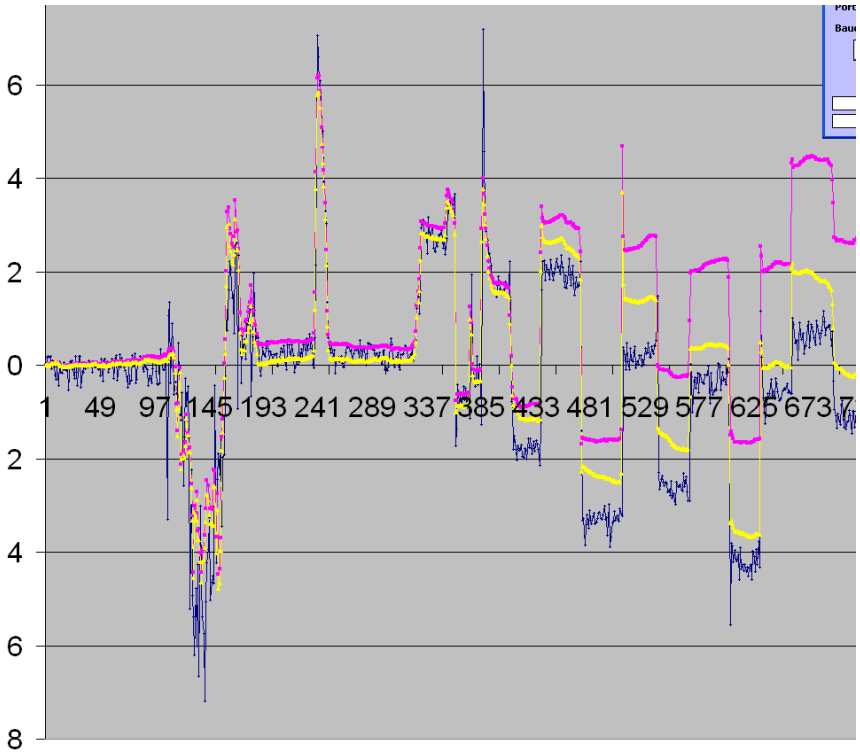


Ilustración 64. Ensayo de filtro complementario. Alfa = 0.99

Tras analizar los resultados obtenidos se ha optado por fijar el valor de alfa en 0.90 unidades por tres razones principales.

- No presenta las fluctuaciones propias del acelerómetro.
- No presenta la deriva propia del giróscopo.
- No se ve comprometida su velocidad de respuesta

En posteriores pruebas realizadas revisando el código realizado por mí, se consiguió encontrar la razón de los resultados con tan elevada deriva.

Gracias a la implementación del filtro complementario se tiene un resultado del ángulo más preciso. Este valor del ángulo se ha añadido en el cálculo del giróscopo para cada eje representando el estado anterior, sustituyendo así el valor del último ángulo del giróscopo en la ecuación por el último ángulo del filtro complementario.

En las próximas Ilustraciones se percibe el diferente comportamiento según se utiliza el ángulo del giróscopo o el ángulo del filtro complementario para calcular el nuevo ángulo del giróscopo.

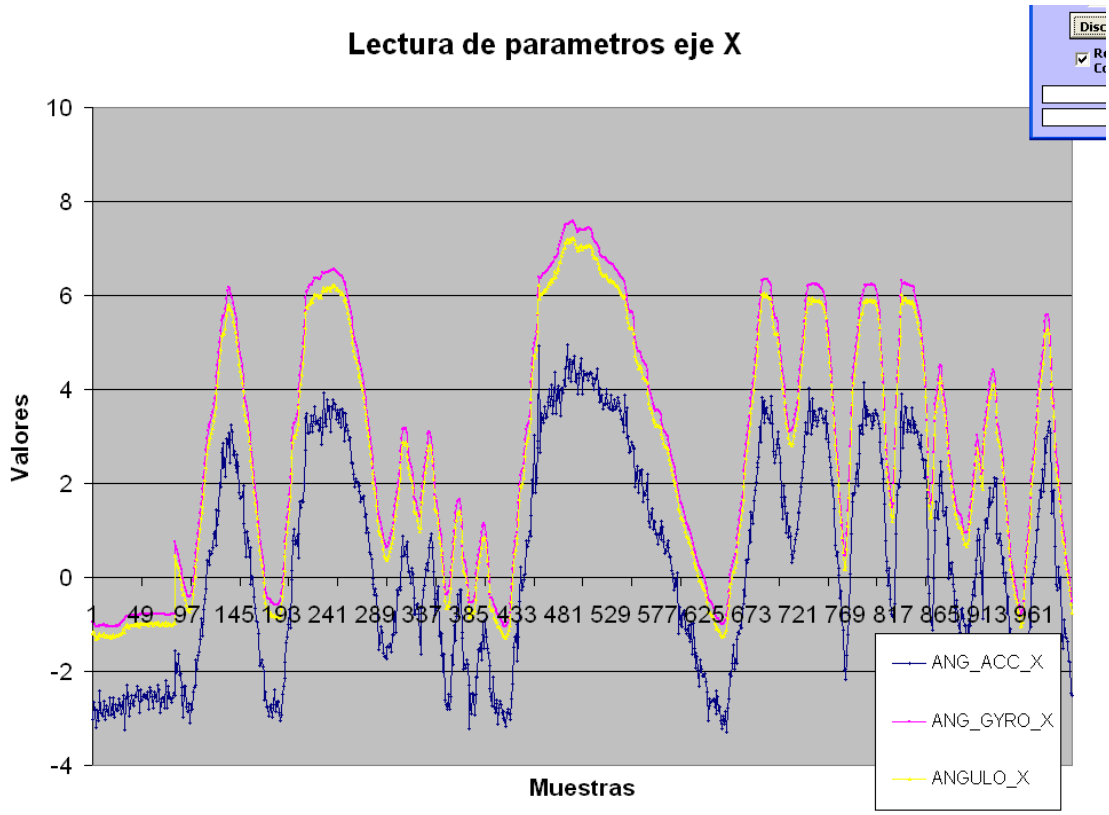


Ilustración 65 Ensayo realizado sobre eje X. Alfa=0.88

(Giróscopo) *>> Ang_gyro_x= Ang_guro_x + velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo_x = Ang_gyro_x*alfa + Ang_acc_x*(1-alfa)*

En la ilustración 65 se reconoce cómo la deriva del giróscopo ha sido eliminada y aporta los mismos valores a baja frecuencia que el acelerómetro. De esta forma la salida del filtro complementario sigue con exactitud la referencia sin experimentar las perturbaciones del acelerómetro

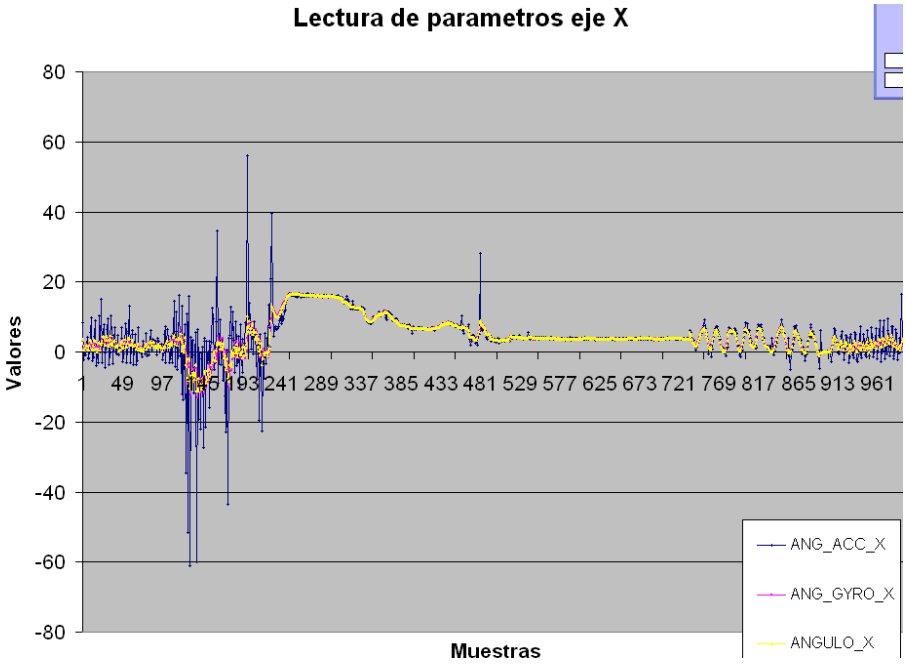


Ilustración 66. Ensayo realizado sobre eje X. Alfa = 0.88

(Giróscopo) *>> Ang_gyro_x= Angulo_x + velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo_x = Ang_gyro_x*alfa + Ang_acc_x*(1-alfa)*

El mismo comportamiento se contempla en los ensayos realizados sobre el eje Y representados en las siguientes Ilustraciones, con alguna discrepancia.

En la ilustración que viene a continuación, al igual que en la primera prueba de valores del eje X presentada en la ilustración 65, se repara en que el resultado del filtro sigue incondicionalmente la referencia del giróscopo.

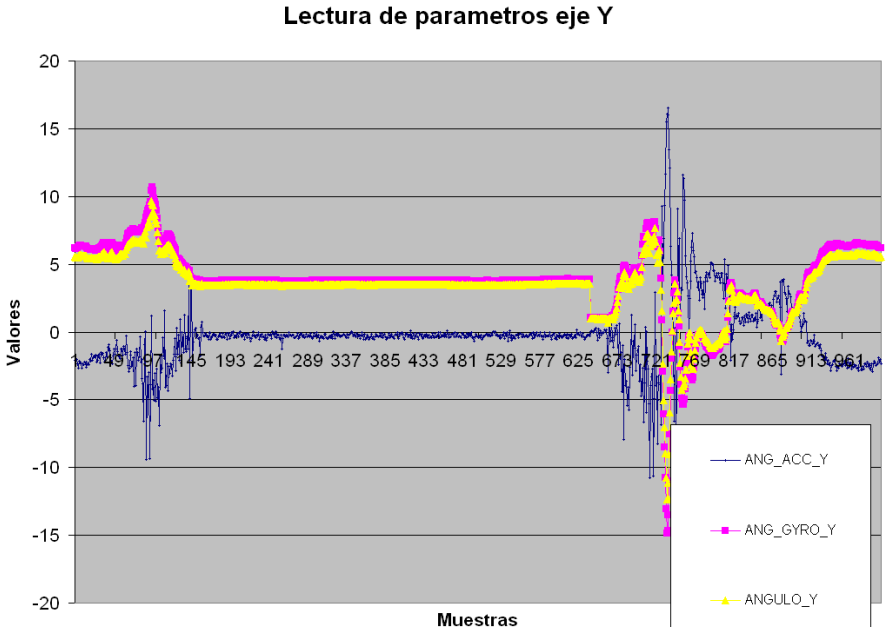


Ilustración 67. Ensayo realizado sobre eje Y. Alfa = 0.90

(Giróscopo) *>> Ang_gyro_y= Ang_gyro_y + velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo_y = Ang_gyro_y*alfa + Ang_acc_y*(1-alfa)*

En la próxima imagen se observa un comportamiento típico de un sistema de fase no mínima.

Al aumentar la inclinación el acelerómetro se comporta con normalidad alcanzando rápidamente la medida adecuada, sin embargo el giróscopo reporta unos valores de signo contrario durante los primeros instantes de cada giro.

Por si fuera poco este desvío también provoca que la señal entera se retrase como figura en la ilustración 68.

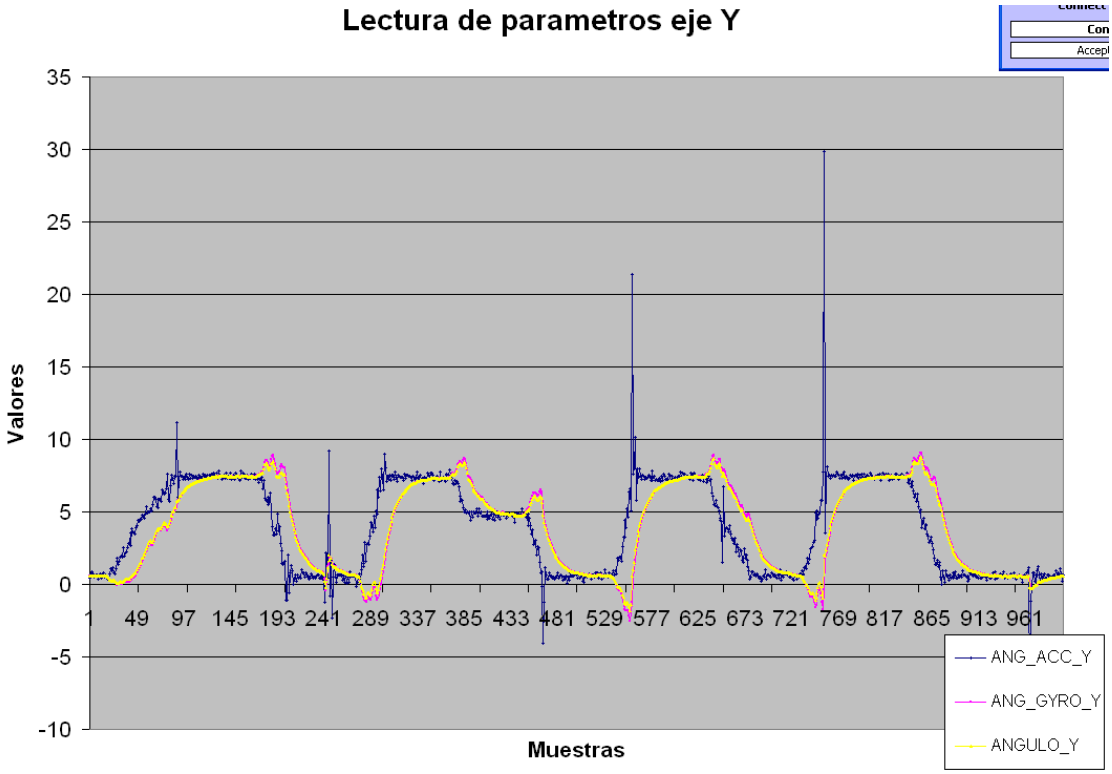


Ilustración 68 Ensayo realizado sobre eje Y. Alfa = 0.90

(Giróscopo) >> Ang_gyro_y= Angulo_y + velocidad*dt*sensibilidad_gyroscopo

*(Filtro) >> Angulo_y = Ang_gyro_y*alfa + Ang_acc_y*(1-alfa)*

Con esta medida se mejora tanto el resultado del giróscopo como el resultado del filtro complementario. El drift del giróscopo ha sido prácticamente eliminado al mantener esta modificación, el resultado del filtro complementario continúa realizando un seguimiento demasiado ajustado al comportamiento del giróscopo, pero como éste aporta un resultado muy apropiado se toma la solución a la deriva por válida.

No obstante nos ha surgido un nuevo problema con los resultados anómalos del giróscopo Y, a pesar de utilizar el mismo código que el del eje X.

Revisando el código con posterioridad se ha logrado identificar por qué el giróscopo no presenta deriva. Para el cálculo del ángulo del giróscopo se ha utilizado el ángulo del filtro complementario, y posteriormente para el cálculo del ángulo del filtro complementario se ha utilizado el dato del ángulo del giróscopo recientemente calculado.

Se presentan a continuación los distintos códigos utilizados para una comparación más fácil.

En el primer caso se utiliza esta configuración: *Ilustracion*es 65 y 67.

*>> ang_gyro_x= ang_gyro_x+ velocidad*dt*sensibilidad_gyroscopo*

*>> angulo_x=ang_gyro_x*0.92 + ang_acc_x*0.08*

En esta situación el angulo_x sigue la referencia del giróscopo y dista enormemente del acelerómetro.

En el segundo caso se utiliza esta configuración: *Ilustracion*es 66 y 68.

*>> ang_gyro_x= angulo_x+ velocidad*dt*sensibilidad_gyroscopo*

*>> angulo_x=ang_gyro_x*0.92 + ang_acc_x*0.08*

En esta situación tanto el *angulo_x* como el *ang_gyro_x* siguen con exactitud al acelerómetro, pero modificamos los resultados reales del giróscopo al sumarle el ángulo previo del filtro complementario, en lugar del ángulo previo del giróscopo.

En el tercer caso se utiliza esta configuración: *Ilustracion*es 69 y 70.

*>> ang_gyro_x= ang_gyro_x+ velocidad*dt*sensibilildad_gyroscopo*

*>> angulo_x=(angulo_x+velocidad*dt*sensibilidad_gyroscopo)*0.92 + ang_acc_x*0.08*

En esta situación el valor del giróscopo no se ve alterado, presenta su característico drift pero el resultado del filtro complementario no lo presenta.

A la hora de asignar el valor del filtro complementario en lugar de tomar el valor del giróscopo calculado con anterioridad, se recalcula con el ángulo del filtro complementario, quedando el código 1 y código 2 diferenciados tan solo sintácticamente. Los resultados se muestran a continuación.

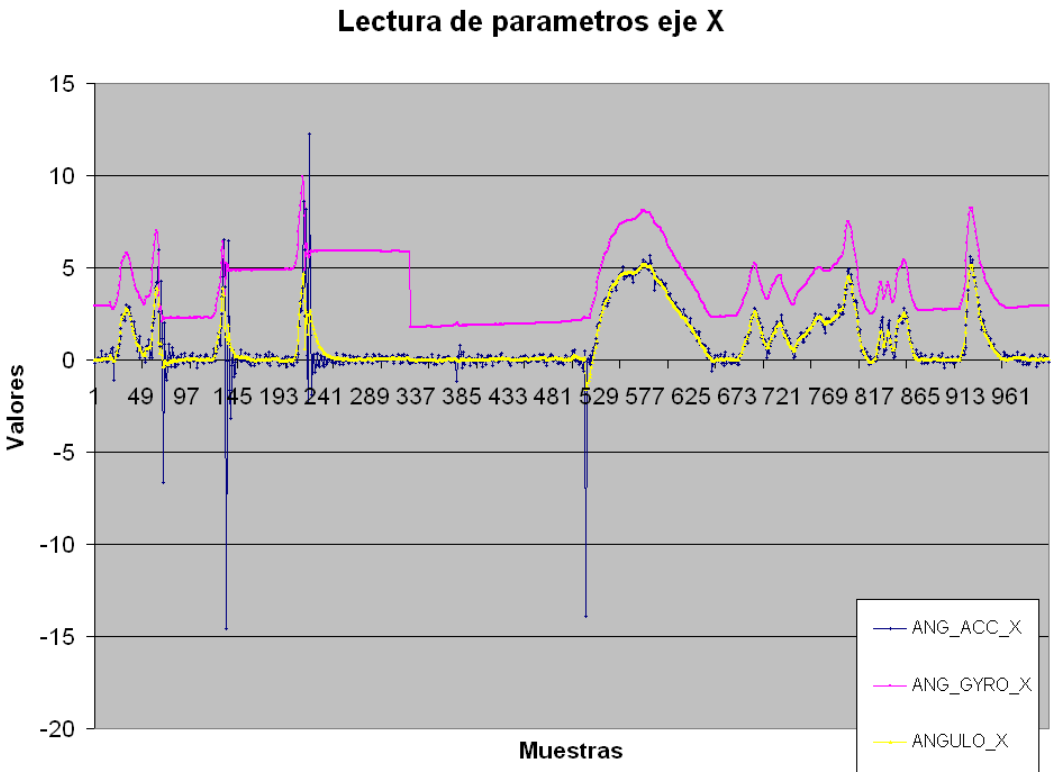


Ilustración 69 Ensayo realizado sobre eje X. Alfa = 0.88

(Giróscopo) *>> Ang_gyro_x= Ang_gyro_x+velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo _x = (Angulo_x + velocidad*dt*sensibilidad_gyroscopo)*alfa + ang_acc_x*(1-alfa)*

En este último ejemplo el resultado es muy adecuado, se consigue ignorar la deriva del giróscopo y un buen seguimiento de la referencia del acelerómetro. Incluso se consiguen corregir los fallos de medida del giróscopo. En la anterior ilustración se ha dejado reposar el dron en horizontal y se han dado unos pequeños golpes bruscos a la estructura. El giróscopo indica un cambio de inclinación, cuando en realidad no es así, pero el filtro se encarga de corregirlo para seguir la referencia del acelerómetro. Muestras 145-200 y 240-330.

Entrando con el ensayo en el eje Y se debe analizar un aspecto de la *Ilustración* 67 que no había sido comentado por el momento. Los valores que resultan del acelerómetro y del giróscopo tienen signos opuestos, se aprecia claramente en las primeras muestras de la gráfica.

En la *Ilustración* que le sigue a la anterior, aplicando el segundo método de cálculo del filtro, este fallo aparentemente se ha sustituido por un comportamiento de orden mínimo. No obstante esto no es más que un transitorio provocado por el signo opuesto que tienen las dos componentes. El ángulo del giróscopo es recalculado con el resultado del ángulo complementario compensando el valor negativo a costa de un pequeño retraso.

Al aplicar el tercer método de cálculo del filtro al eje Y, el giróscopo sigue su propio camino con signo opuesto al acelerómetro y el ángulo complementario presenta un retraso y el mismo comportamiento con los primeros valores de movimientos grandes.

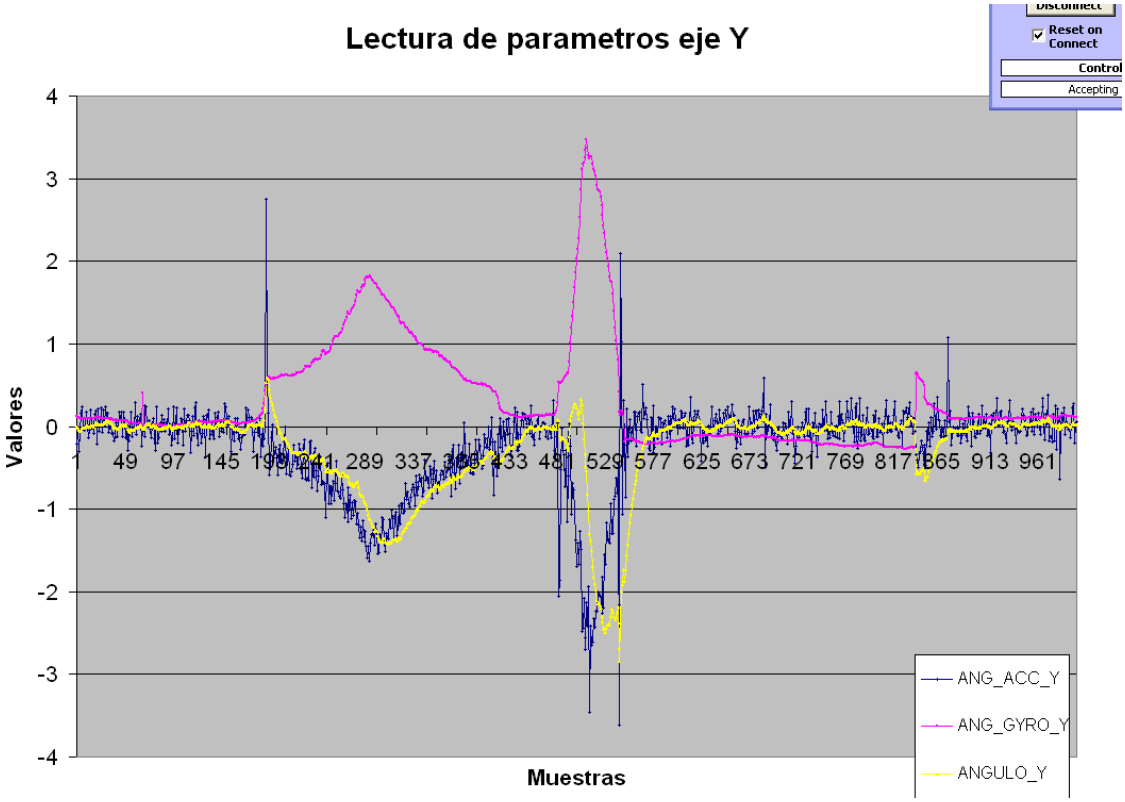


Ilustración 70. Ensayo realizado sobre el eje Y. Alfa = 0.90

(Giróscopo) *>> Ang_gyro_y= Ang_gyro_y+velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo _y = (Angulo_y + velocidad*dt*sensibilidad_gyroscopo)*alfa + ang_acc_y*(1-alfa)*

Al inclinar el dron en el sentido positivo del eje Y según la serigrafía del sensor obtenemos la gráfica de encima, es el acelerómetro el que envía unos valores de signo opuesto a las indicaciones gráficas. Para corregir esta situación tan solo se debe añadir un signo negativo en el argumento del arcotangente al calcular el ángulo del acelerómetro.

Aplicando todas las soluciones el resultado del ángulo complementario para el eje Y queda así:

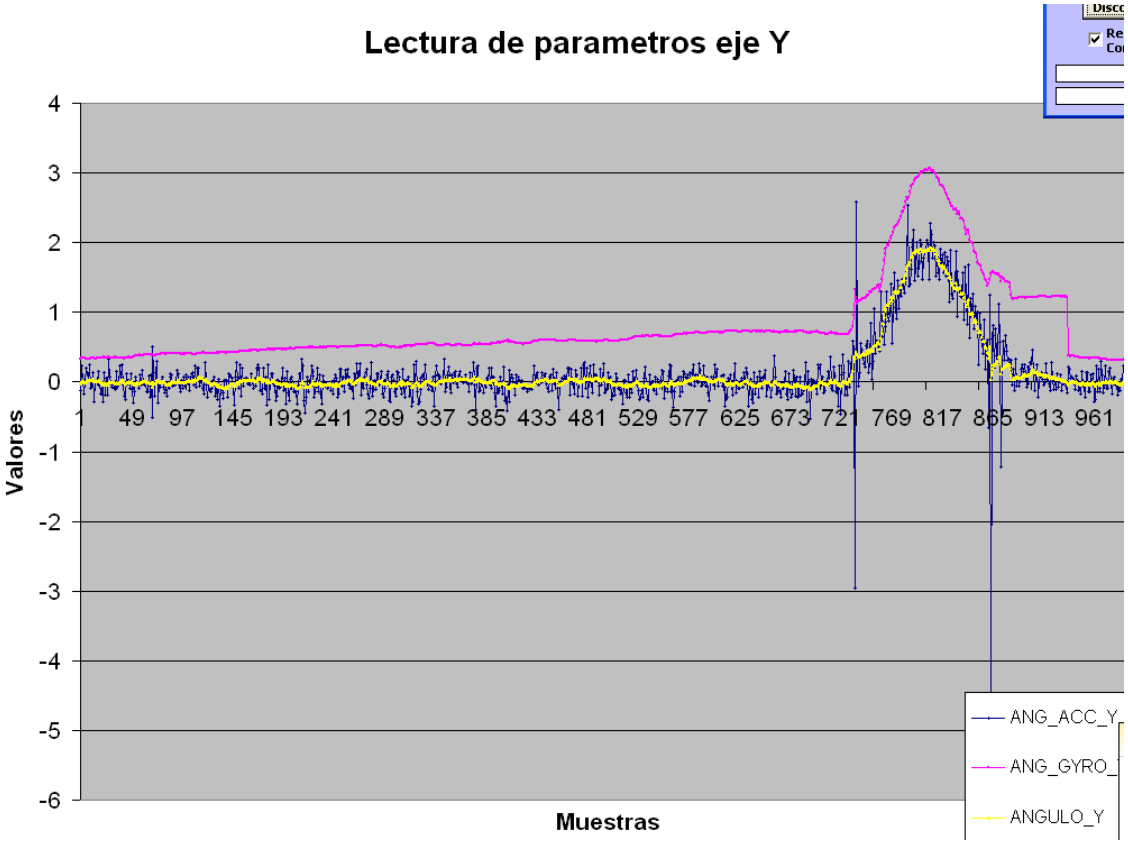


Ilustración 71 Ensayo realizado sobre el eje Y. Alfa = 0.90

(Acelerómetro)

*>> ang_acc_y=atan2(-const_experimental_escalado_y*Fx,(sqrt(pow(Fz,2)+pow(Fy,2))))*const_radian_ang_escalado*

(Giróscopo) *>> Ang_gyro_y= Ang_gyro_y+velocidad*dt*sensibilidad_gyroscopo*

(Filtro) *>> Angulo _y = (Angulo_y + velocidad*dt*sensibilidad_gyroscopo)*alfa + ang_acc_y*(1-alfa)*

En la ilustración de arriba se muestra que se han eliminado todos los fallos y comportamiento anómalos.

Antes de lograr paliar los errores de este código se había avanzado con el código 2 el cual está más depurado y tiene menor tiempo de ciclo. Se continúa utilizando el código 2 con la configuración del filtro como se comentaba con anterioridad, con $alfa = 0.90$

3.4.8.6 Control PID

La implementación del control PID en Arduino no es más que la traducción de las expresiones matemáticas a código de programación.

La referencia del sistema se asigna a una variable y la lectura de los sensores es la salida del filtro complementario.

La estimación de las ganancias se realiza de forma experimental con el método de prueba y error. Existen otros métodos como el de Ziegler-Nichols, en este método mediante la ganancia proporcional a la que comienza la oscilación y su periodo de repetición haciendo uso de unas tablas se determina la configuración de las ganancias.

El método de prueba y error consta de los siguientes pasos:

- En primer lugar se deben anular las ganancias integral y derivativa igualando sus respectivas variables K_i y K_d a 0.
- En segundo lugar se aumenta el valor de la ganancia proporcional para que alcance la referencia hasta que se produzcan las primeras oscilaciones en el sistema.
- A continuación se fija el valor alcanzado en la constante proporcional y se comienza a aumentar el valor de la constante integral hasta lograr eliminar la oscilación presente por el paso anterior.
- Por último se procede aumentando gradualmente la ganancia derivativa para aumentar la velocidad de respuesta cuidando de que no se desestabilice el sistema.[25]

En el cálculo de la componente integral del error se recurre de nuevo a la función *millis()* de Arduino para determinar el tiempo exacto que ha transcurrido desde la anterior medición. La incorrecta asignación del tiempo de ciclo conlleva un erróneo cálculo de la acción proporcional desestabilizando el sistema.

Para limitar el efecto "*wind up*" se incluye en el cálculo de la acción integral una limitación máxima y mínima que corresponde con 180 y 0 respectivamente, los valores entre los que se encuentra la señal mandada a los actuadores desde Arduino.

En nuestro diseño existen dos sistemas de control, el sistema que controla el giro respecto al eje X, *roll* y el sistema que controla el giro respecto al eje Y, *pitch*.

El control PID de cada sistema se realiza de forma independiente, la relación entre estos sistemas se establece en la formulación de la acción de control, ya que para la tipología en X se deben controlar todos los motores para cada giro siendo necesaria una coordinación entre las aportaciones de las acciones *pitch* y *roll*.

El código de programación del control PID se incluye en el *Anexo 6*.

3.4.8.7 Cálculo de la acción de control

Para lograr enviar a los motores la señal adecuada se hace uso de la librería "*Servo.h*", esta librería permite la utilización de modulación PPM (Pulse Position Modulation). Esta señal se envía a través de cualquier salida digital del Arduino aumentando bastante el rango de dispositivos que se pueden controlar a la vez.

La modulación PPM que transmite el Arduino se basa en el mismo principio que el PWM (Pulse Width Modulation – Modulación por anchura de pulso).

Ambos sistemas tienen una amplitud de pulso fija, logran la modulación controlando el tiempo de encendido y apagado.

En la modulación PWM, en un periodo de muestreo, el valor de salida viene definido por la relación entre la anchura de la franja en la que la salida está HIGH respecto a la anchura de la franja en la que la salida está LOW. Una señal que tuviera ¾ partes HIGH y ¼ parte LOW daría como resultado el 75% de la tensión nominal de salida.

Sin embargo en la modulación PPM, en un periodo de muestreo, solo se muestra un pulso en la salida en el punto que coincide con el paso de HIGH a LOW de una señal PWM. En todo momento el sistema sabe cuándo comienza el periodo de muestreo y tan solo necesita saber la posición respecto al inicio que determine el porcentaje de tiempo de muestreo abarcado. Los pulsos que se generan en PPM son siempre estrechos con la misma anchura.

Una señal que tuviera un pulso en la salida de la modulación, a ¾ del tiempo de muestreo, podría decirse que ha tardado un 75% del tiempo de muestreo en producirse. Por lo tanto la señal que se envíe será un 75% de la tensión nominal de salida.

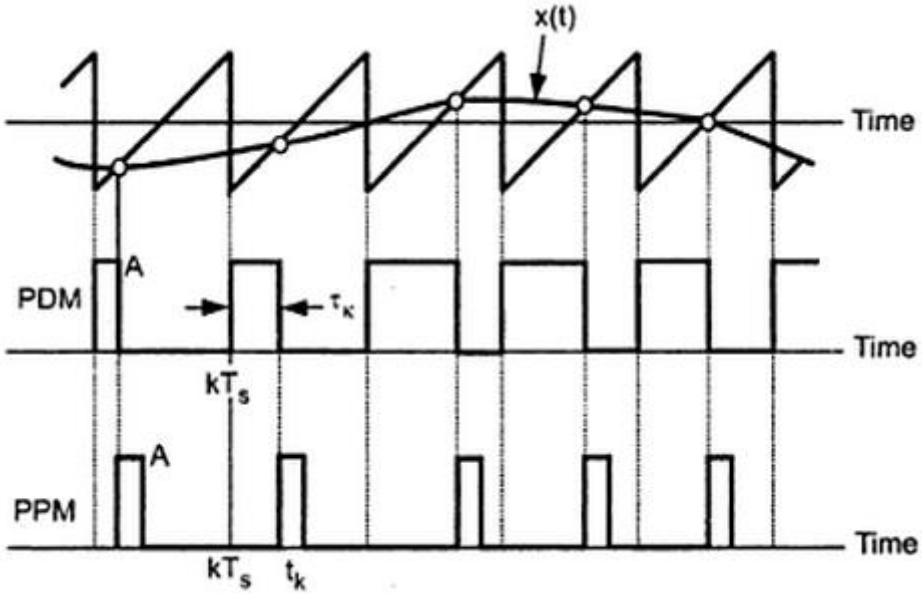


Ilustración 72. Correspondencia entre modulación PDM (o PWM) y PPM.
Analog And Digital Communication Engineering. 3-41

Controlar estas variaciones de tensión es trabajo de los ESC, Electronic Speed Controller, son el único elemento que falta entre el Arduino y los motores para completar la conexión del sistema.

A grandes rasgos estos variadores se encargan de proporcionar un voltaje de salida determinado a los motores en función de la señal de entrada recibida del Arduino.

Para que los controladores electrónicos de velocidad realicen una correcta provisión de voltaje se deben calibrar al inicio indicando los valores máximos y mínimos que pueden tomar los motores. Se deben calibrar todos por igual ya que si no fuera así la formulación de la señal de control tendría que estar personalizada para cada motor, asumiendo que siempre se calibraran igual.

Para calibrar los reguladores se debe indicar en ellos el valor máximo durante los 2 primeros segundos de activación y a continuación enviar el valor mínimo. La emisión de un pitido largo de cada motor indica que han sido calibrados según las referencias recibidas. Se incluye el código de programación en el *Anexo 7*.

Para hacer uso de la librería *Servo.h* se debe crear un "objeto Servo" por cada variable a controlar y a continuación se realiza la asignación de estos objetos a sus respectivos pines digitales de Arduino. Cada objeto representa a un motor.

Motor	PIN (nombre)	PIN (numero)
M1	D3	6
M2	D4	7
M3	D5	8
M4	D6	9

Se debe puntualizar que la función que utiliza la librería *Servo.h* para asignar los objetos servo a una dirección solo cuenta con los pines digitales, por lo tanto la numeración será 3, 4, 5 y 6 para los motores M1, M2, M3 y M4 respectivamente; en lugar de sus números de pines 6, 7, 8 y 9.

En la anterior sección relacionada con la actuación de los motores se han presentado las asignaciones en función de si la tipología utilizada es "x" o "+".

De la misma forma en la programación se incluyen los códigos para el funcionamiento de ambas configuraciones.

En primer lugar, el diseño original, con tipología “x”:

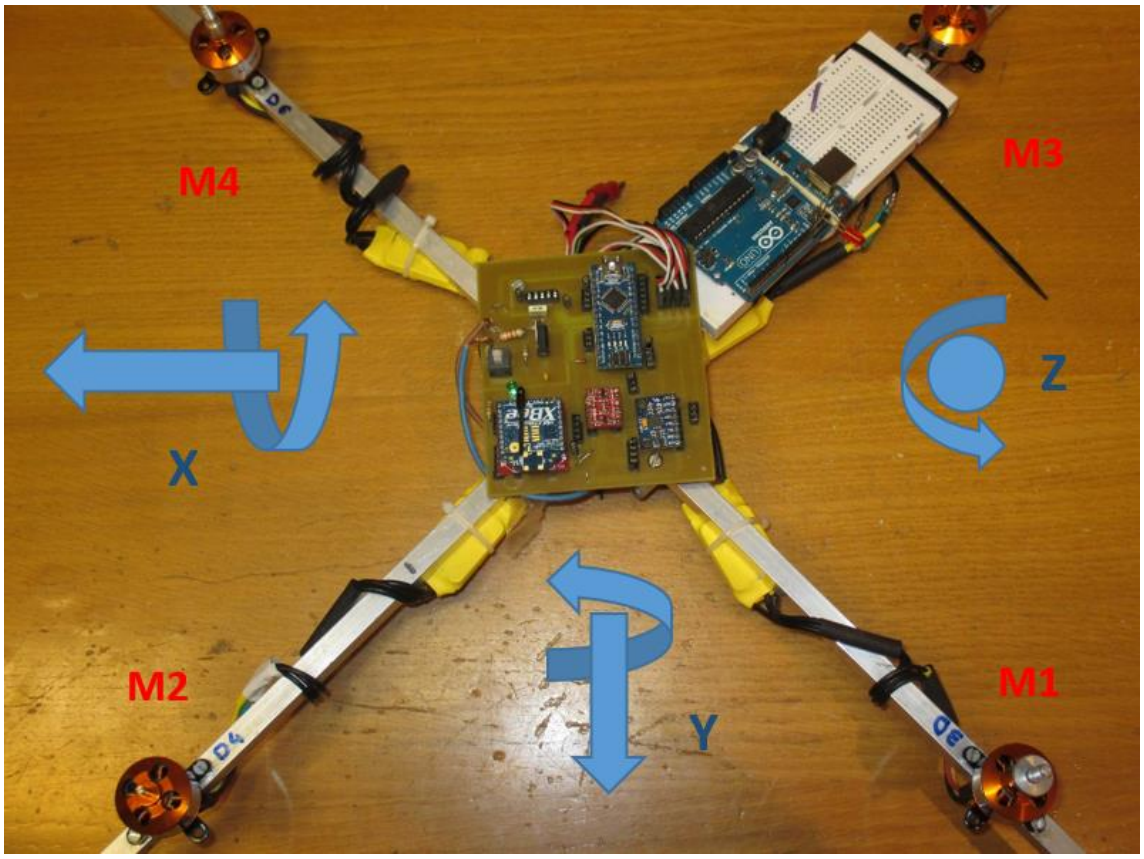


Ilustración 73. Fotografía del dron configurado para el control con Arduino Nano. Tipología X.

//Acción de control para tipología x

Motor 1 -> *accion_M[1]=altura-pitch_output-roll_output;*

Motor 2 -> *accion_M[2]=altura+pitch_output-roll_output;*

Motor 3 -> *accion_M[3]=altura-pitch_output+roll_output;*

Motor 4 -> *accion_M[4]=altura+pitch_output+roll_output;*

Para entender la asignación de signos se deben seguir las indicaciones de las flechas. En cada imagen están dispuestas acordes con los ejes de coordenadas del sensor en función de su posición respecto a las aspas.

En segundo lugar la tipología “+”:

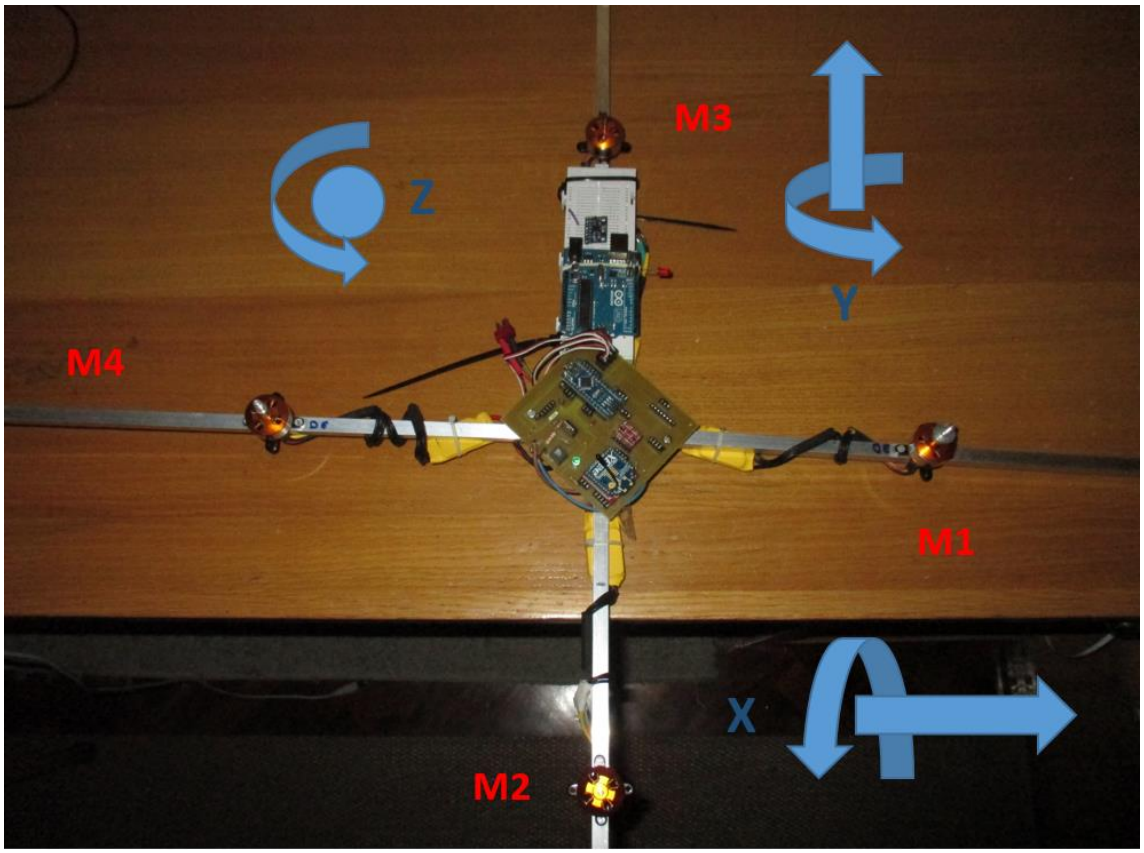


Ilustración 74. Fotografía del dron configurado para el control con Arduino UNO. Tipología +.

//Acción de control para tipología +

Motor 1 -> *accion_M[0]=altura+pitch_output;*

Motor 2 -> *accion_M[1]=altura+roll_output;*

Motor 3 -> *accion_M[2]=altura-roll_output;*

Motor 4 -> *accion_M[3]=altura-pitch_output;*

Antes de escribirse finalmente los valores en el objeto servo, se someten a una limitación para que no supere sus límites de funcionamiento.

Se utiliza la misma técnica que el “Anti-WindUp” del giróscopo. Se analizan los valores y si alguno supera un valor máximo, se iguala la variable al valor máximo, de esta forma ningún valor superará la limitación. Del mismo modo se procede con los valores mínimos. En este caso y como suele ser habitual al utilizar la librería *servo.h*, los límites se asignan en 0 y 180.

Se incluye el código de programación de las órdenes de control en el *Anexo 8*.

3.4.8.8 Comunicación inalámbrica vía Xbee

Uno de los principales objetivos alcanzados en el desarrollo del dron ha sido implementar un módulo de comunicación XBee para comunicar el dispositivo con un ordenador.

Se planteó la posibilidad de otros métodos de comunicación pero todo ellos suponían mayores consumos de batería, como por ejemplo un par de métodos comúnmente utilizado que es la comunicación Bluetooth, o la comunicación Wifi. XBee ofrece una longitud suficiente de incluso 1000m con algunos módulos.

Para lograr la comunicación se dispone de dos módulos XBee. Estos dos módulos deben ser de la misma serie para que se puedan comunicar entre sí. Los módulos de los que se dispone se tratan de dos Zigbee RF Modules XBee PRO (Serie2 – S2) fabricados por Digi International.

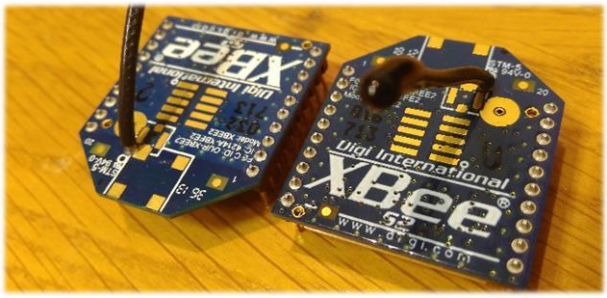


Ilustración 75 Zigbee RF Modules XBee PRO S2. Top view

Uno de los módulos se encuentra conectado a la placa y se comunica con el hardware de Arduino mediante la comunicación serie a través del conversor de niveles como se ha presentado con anterioridad.

El segundo módulo se conecta a una placa que permite su conexión serial con otros dispositivos a través de un puerto mini USB-B, la Xbee Explorer USB Breakout Board.

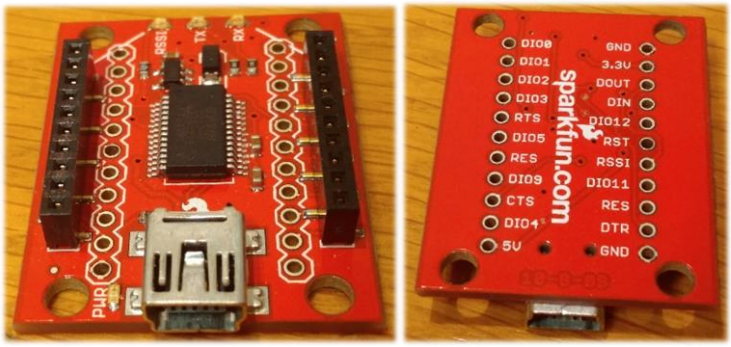


Ilustración 76. Placa de conexión de módulo Xbee con entrada mini USB-B. Drcha: Top view. Izqda.: Bottom view.

Este módulo se conecta al ordenador y mediante el software XCTU proporcionado por el fabricante se configura la conexión. No obstante, no es necesaria la utilización de este software para configurar los módulos XBee.

Configuration & Test Utility Software
A Digi International Inc. product.

Se ponen a disposición del usuario una serie de comandos, que precedidos por los caracteres AT, el módulo XBee reconoce como órdenes de configuración. Se podría hacer uso de Arduino para configurar los módulos pero el programa XCTU es una herramienta gratuita que facilita la configuración de los

módulos al establecer un contacto directo con el módulo de forma automática y presentar sus distintas características.

Para que ambos módulos puedan mantener una conversación las direcciones de destino de los mensajes de cada uno deben ser la dirección del receptor. Esta dirección es un código de 2 bytes, *adress high* y *adress low*, en primer lugar para verificar que los dos módulos disponibles se pueden comunicar entre sí deben tener la misma *adress high*.

Addressing
Change addressing settings

SH Serial Number High	13A200
SL Serial Number Low	40AC4C49
MY 16-bit Network Address	0
DH Destination Address High	13A200
DL Destination Address Low	40AFCEB2

Addressing
Change addressing settings

SH Serial Number High	13A200
SL Serial Number Low	40AFCEB2
MY 16-bit Network Address	7CC8
DH Destination Address High	13A200
DL Destination Address Low	40AC4C49

Se puede encontrar la información de la dirección en la parte inferior de los módulos, o también utilizando el programa XCTU el cual realiza una lectura de los parámetros fijos del módulo. Se presenta una captura de pantalla en la imagen anterior.

Ilustración 77. Zigbee RF Modules XBee PRO S2. Bottom view

Se debe tener cuidado en que ambos canales tengan asignado el mismo PAN ID y la misma tasa de transmisión de datos (baudios). Estos parámetros pueden ser asignados a través del programa.

A las redes Zigbee también se las llama de área personal, *personal area network* (PAN). Cada red XBee se define en una única PAN ID, en todos los casos el PAN ID dispone de 16 bits lo que permite 65535 redes distintas. Aunque se trata de un gran número de conexiones para ciertas aplicaciones era escaso por lo que se desarrollaron los módulos de 64 bits de PAN ID. [27]

Para las primeras pruebas se han configurado ambos módulos con PAN ID 7 y a 9600 baudios.

El módulo XBee que se encuentra conectado al ordenador debe ser configurado como *Coordinador* de la red Zigbee. Mientras que el módulo que se conecta a la placa y al Arduino debe ser configurado como *End Device* (dispositivo final) o *Router*.

En teoría la configuración de *End Device* es la más correcta pero experimentalmente se ha comprobado que daba errores al inducirse el modo SLEEP continuamente. La configuración estilo *Router* no realiza la función SLEEP y también se comunica correctamente con el elemento *coordinador* por lo que se toma finalmente esta configuración a riesgo de aumentar el consumo energético.

Una vez configurados ambos módulos y conectados cada uno en sus respectivas posiciones se establece la conexión entre ellos automáticamente. Esto se puede verificar en la placa conectada al ordenador al observar el LED RSSI encendido de forma permanente.

Ilustración 78. Comunicación establecida entre los módulos XBee. Prueba hardware. Los LED de arriba abajo: RSSI, TX, RX.

Mediante el software de Digi International se visualiza esta comunicación entre los módulos.

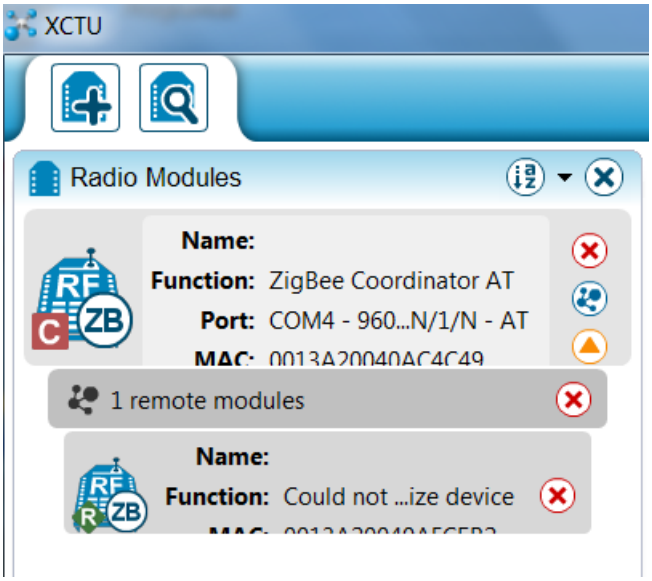


Ilustración 79. Comunicación establecida entre los módulos XBee. Prueba Software.

Por algún fallo del software no se puede ver la dirección MAC completa del *router*, el módulo remoto que tiene una R sobre fondo verde en su icono. Se observa la del *coordinador* y se puntualiza que el MAC es una combinación de las direcciones alta y baja del dispositivo.

No se permite la lectura de los parámetros del módulo XBee con función *router* por haber seleccionado el método de comunicación AT, el cual es el idóneo para comunicaciones punto a punto, mientras que la otra variedad que se propone en el programa, la comunicación ATI, está orientada a redes con varios elementos, y en ese caso sí se puede leer los datos de los módulos remotos.

A la hora de realizar la comunicación entre los dos módulos se tuvo una larga lista de fallos. En ciertos momentos se encontraban los módulos y más adelante se perdían. Hubo que reiniciar el firmware de los módulos gran número de ocasiones e intercambiar *coordinador* por *router* para tratar de solucionar estos extraños fallos.

Más adelante se solucionó un fallo con el programa Arduino en sí al desconectar el puerto serie del módulo XBee y se probó esta medida también a la hora de establecer la conexión entre los módulos XBee con resultados acertados por el momento.

Una vez establecida la conexión entre los módulos se puede volver a conectar la comunicación serie entre Arduino y el módulo XBee.

Las primeras pruebas se realizaron con un código simple en Arduino que mandaba continuamente el mensaje "Hola red Zigbee". En un principio no se lograba cargar la programación en el Arduino pero esto era causa una vez más de la conexión serie entre Arduino y el módulo XBee. Al menos la entrada RX fue necesario desconectar para cargar el programa.

Con el programa cargado y la comunicación XBee establecida se puede conectar definitivamente la comunicación serie de la placa. En ese momento se comienzan a visualizar ya en el monitor serie de XCTU los datos que están llegando y conforme van llegando el LED RX de la placa miniUSB se ilumina.

Se prepara un código sencillo en Arduino para que escriba por el puerto serie los datos que recibe por el mismo puerto serie.

```
>> if (Serial.available())
>> {
>>      ch=Serial.read();
>>      Serial.println(ch);
>> }
```

Pero este sencillo programa en un primer momento no funcionó. Se hicieron diversas pruebas para tratar de localizar el error. Escribiendo directamente en el monitor serie de Arduino si se conseguía que el programa se ejecutara correctamente. Pero a través de los módulos de radio frecuencia no.

Al menos el pin de entrada del módulo XBee *router* "DIN" funcionaba correctamente ya que estaba recibiendo la frase "Hola red Zigbee" y enviándosela al *coordinador*. Teniendo en cuenta que en el ordenador se recibía la información del puerto serie sin problemas, pocas posibilidades había de que hubiese un fallo en la comunicación *wireless*.

Así como en la recepción de datos se ilumina el LED RX, en la escritura lo hace el TX y se pudo comprobar como esa función si se estaba realizando.

Se investigó con el osciloscopio la señal enviada desde el *coordinador*. Al *router* llegaba la señal sin problemas, al conversor de nivel también, pero la salida del conversor de nivel a 5V permanecía constantemente a 5V, sin transmitir la señal.

La clave resultó estar en las conexiones del conversor de niveles. Se recuerda la ilustración 34 en la que se indica que la señal del lado de baja tensión debe entrar por el pin TX1 y sale por su correspondiente en el lado de alta tensión. Por lo tanto la salida de datos por el pin TX0 se interpreta en Arduino como una entrada, es decir debe ir conectado a RX0.

Este aspecto se ve reflejado en el apartado de observaciones de la placa.

Utilizando la protoboard con Arduino UNO y cableando directamente las conexiones tal que TX0 del conversor de niveles coincidiera con RX0 de Arduino UNO y RX1 del conversor con TX1 de Arduino.

Haciendo de nuevo uso del osciloscopio en la entrada RX0 de Arduino se consiguió observar la trama de datos recibida por el módulo XBee.

Ilustración 80. Trama de datos correspondiente a "Hola" recibido por el pin RX0 de Arduino UNO.

Una vez corregida la conexión se comprobó que se realizaba correctamente la comunicación bidireccional entre el módulo *coordinador* y el *router*. Para ello se utilizó el código presentado anteriormente.

El objetivo de establecer esta conexión inalámbrica a tiempo real entre el ordenador y el dron es poder controlar los parámetros que lo configuran para observar los cambios que se perciben en su comportamiento.

De forma muy sencilla se puede agregar unas líneas de código para enviar el valor de las variables al puerto serie. Se deben utilizar las mismas órdenes que se usan para las pruebas durante el desarrollo del dron.

>> Serial.print(variable)

>> Serial.println(variable)

Esta opción se puede implementar si se desea monitorizar las variables para simular un movimiento con algún programa como Processing.

Para hacer un uso correcto del programa se deben respetar una serie de normas a la hora de mandar el valor por el monitor serie. Estas normas son necesarias para que el programa entienda correctamente a qué variable nos referimos y la magnitud que deseamos enviar.

El formato de envío de datos al dron debe ser el siguiente:

- Los dos primeros caracteres a enviar corresponden a la magnitud que se desea modificar, escribiendo siempre en minúsculas. Las variables que se pueden modificar y su referencia se muestran a continuación.

Variable a modificar	Escritura en el *monitor* serie
Alfa	aa
Altura	al
Referencia de giro X, roll	rx
Referencia de giro Y, pitch	py
Referencia de giro Z, yaw	yz
Ganancia proporcional, Kp	kp
Ganancia integral, Ki	ki
Ganancia derivativa, Kd	kd

- De seguido sin espacio se debe incluir el valor a introducir con un máximo de 6 cifras decimales.
- Como separador decimal debe usarse el punto.
- Se debe terminar con el carácter f, que señaliza el fin de la trama.

Se muestra a continuación un ejemplo, en él se puede observar como finalmente es asignado el valor a la variable correspondiente, para una mejor comprensión y comprobar los efectos del programa se ha escrito en la barra de envío del monitor serie el comando que recientemente se había enviado.

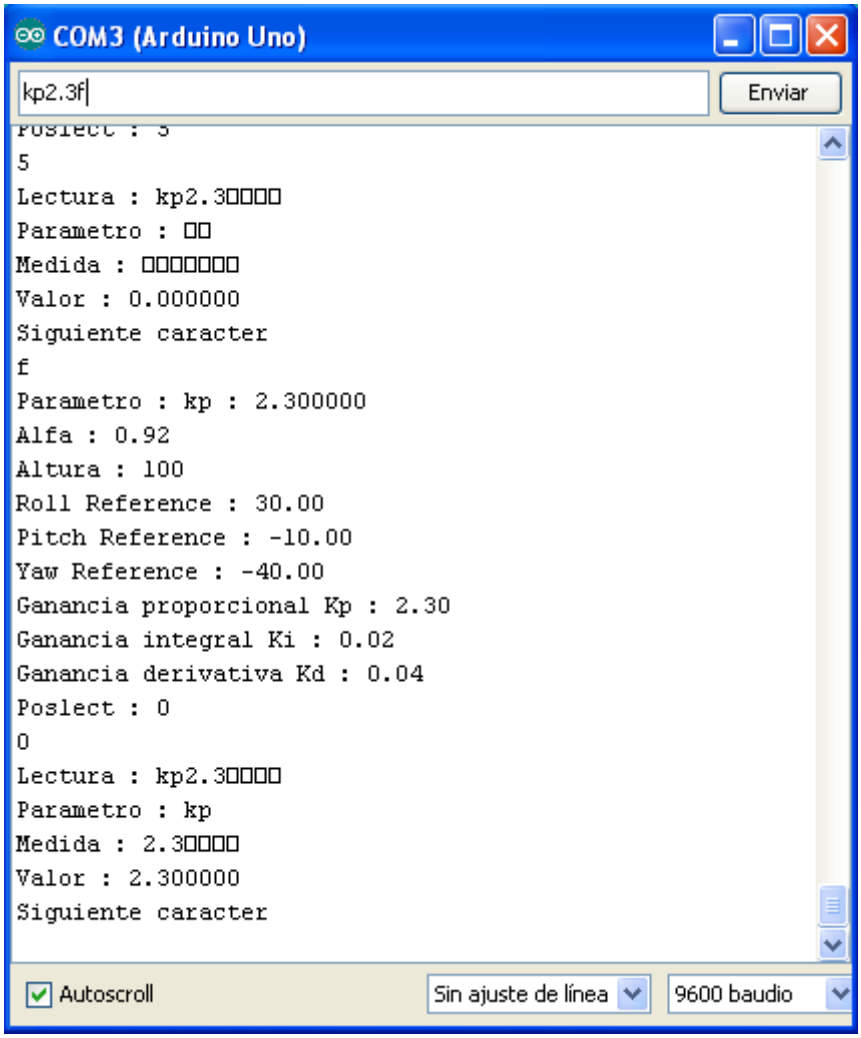


Ilustración 81. Resultado del código ante una entrada de "kp2.3f" por el puerto serie.

Previamente se ha añadido cierto valor al resto de variables del sistema utilizando también el envío a través del puerto serie. Se puede comprobar como el valor asignado a cada variable se mantiene correctamente al continuar el programa y asignar nuevos valores.

Se incluyen unas capturas de pantalla del funcionamiento del programa haciendo uso de la comunicación inalámbrica.

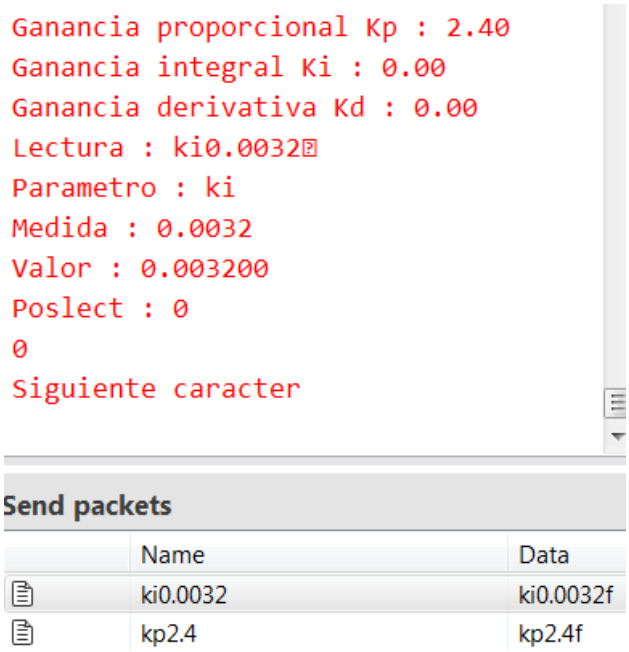


Ilustración 82. Resultado del código XBee a través de los módulos. Valor pequeño no mostrado.

Como los valores de las ganancias integral y derivativa son muy pequeños, es necesario utilizar muchos decimales para sus valores. La variable float reporta valores con hasta 7 cifras decimales, pero la función *>> Serial.print(variable)* de forma predeterminada muestra tan solo 2 cifras decimales, para mostrar las unidades deseadas tan solo se debe añadir la cantidad en la función. *>>Serial.print(variable,6)*.

El código completo de comunicación se incluye en el *Anexo 9*.

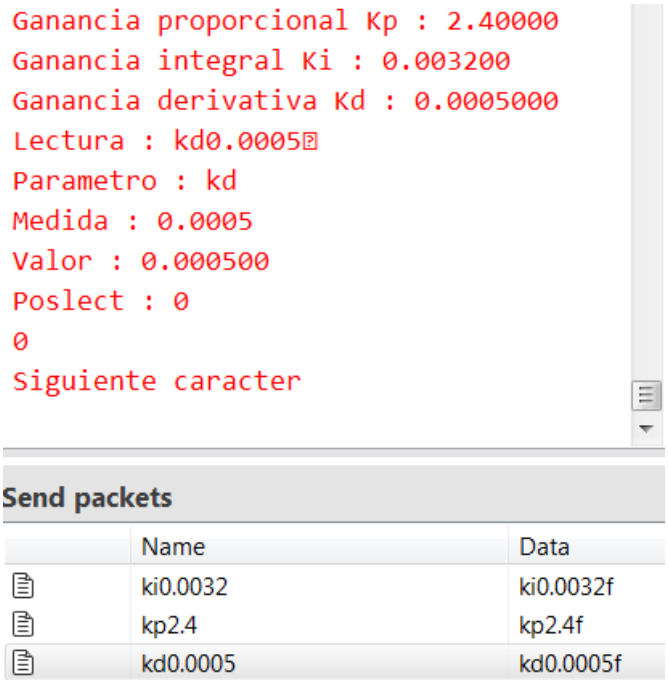


Ilustración 83. Resultado del código XBee a través de los módulos. Cifras decimales reajustadas..

El código ejecutado se realizó primero por un lado y luego se hicieron unas pequeñas modificaciones para añadirlo al código principal del programa como una subrutina denominada "*xbee()*".

A esta subrutina se accede al inicio del void loop() si en el puerto serie se encuentra algún valor (>> *if (Serial.available()*).

Las primeras pruebas dieron resultados incorrectos que se solucionaron añadiendo un pequeño delay de 50 ms entre cada lectura del puerto serie.

Con este último añadido el código de Arduino ya está listo para realizar un estudio de la estabilidad modificando los distintos parámetros que lo definen de forma inalámbrica.

3.4.8.9 Monitorización de la carga de la batería

En la sección de Design Spark se ha introducido el montaje del circuito de la batería.

Tal y como se comenta en dicho apartado, se implementa un divisor de tensión 1:4 para escalar los 12V de la batería a 3V y poder medirlos con la entrada analógica de Arduino.

El divisor de tensión está formado concretamente por una resistencia de $33k\Omega$ y otra de $100k\Omega$ quedando la siguiente relación.

$$V_{out} = V_{in} * \frac{33k\Omega}{100k\Omega + 33k\Omega} = V_{in} * 0.248$$

Aplicándolo a la batería de 12V:

$$V_{out} = 12V * 0.248 = 2.977V$$

Por el pin analógico de Arduino deben entrar valores con tensión entre 0V y 5V. Estos valores serán traducidos por el ADC (Analogic-Digital Converter) integrado en Arduino. El conversor tiene una resolución de 10 bits, lo que significa que traduce los valores a números enteros entre 0 y 1024. Esto supone que para obtener el resultado en voltaje se debe escalar la medida.

Dividiendo el valor de entrada máximo, entre 1024 se obtiene el número de mV por unidad digital.

$$^{5V}/_{1024LSB} = 4.883\,^{mV}/_{LSB}$$

De forma inversa predecimos el valor digital.

$$^{1024LSB}/_{5V} = 204.8\ ^{LSB}/_{V}$$

Sabiendo que la tensión máxima que va a alcanzar el pin analógico es 3V, el ADC traduce esta tensión a 614.

Con la función *>> analog.Read()* se procede con la lectura de la entrada analógica que nos muestra el resultado del conversor ADC. Para conocer el valor en tensión de esta medida se multiplica por el factor $4.883\,^{mV}/_{LSB}$.

El objetivo es conocer el valor de la tensión que hay en bornes de la batería, antes de aplicar el divisor de tensión que modifica su valor, para corregir esta modificación tan solo se debe dividir la señal entre el mismo factor de conversión que se consigue con el divisor de tensión, 0.248.

De esta forma el resultado nos dará en todo momento el valor real de tensión de la batería.

$$V_{divisor} = lectura\ ADC * 4.883\,^{mV}/_{LSB} \rightarrow V_{divisor} = 614 * 4.883\,^{mV}/_{LSB} = 2998.2mV$$

$$V_{bateria} = {}^{V_{divisor}}/_{0.248} \rightarrow V_{bateria} = {}^{2998.2mV}/_{0.248} = 12089.4mV \rightarrow 12.1V$$

En la anterior sección se comenta que se disponen dos diodos para estabilizar la tensión, a 5V y a tierra. Es importante que tanto este diodo como el Arduino con el que se realice la medida estén conectados a la misma tierra, ya que en caso contrario se pueden producir fluctuaciones en la medida.

Una vez obtenido el valor de carga de la batería, se ha desarrollado un sencillo código para aterrizar el dron en caso de falta de carga.

Se compara el valor de tensión de la batería con el valor de tensión de descarga obtenido de las hojas de especificaciones de la batería, en caso de falta de ella, para baterías LiPo de 12V este valor suele estar en torno a los 10V.

Si el resultado de la comparación indica que la carga es menor que este valor, disminuye en una unidad la variable altura. De esta forma el dron desciende progresiva y lentamente. Utilizando este método se consigue mantener el control PID y el resto de funciones de control operativas durante el descenso.

El valor de la variable altura continúa disminuyendo por debajo de 0 hasta llegar a -180 con la intención de contrarrestar las asignaciones de acción de roll, pitch y yaw que pudieran darse en los últimos instantes de aterrizaje.

El código de programación se incluye en el *Anexo 10*.

3.5 Relación de costes

En este apartado se realiza una enumeración de los elementos utilizados y su correspondiente precio unitario.

En primer lugar se evalúan los costes de producción de un solo ejemplar, el prototipo inicial.

Artículo	Clase	Unidades	Precio Unidad	Precio total
MATERIAL				1 UNIDAD
Arduino Nano v3.0 + Cable Macho USB - A <-> Macho MiniUSB -B	PCB	1	21,520 €	21,520 €
Arduino UNO rev. 3 + Cable Macho USB-A <-> Macho USB-B	PCB	1	17,330 €	17,330 €
XBee PRO ZB (S2) RF Module	PCB	2	22,950 €	45,900 €
XBee ZB Module Breakout Board	PCB	1	9,950 €	9,950 €
SparkFun Xbee Explorer USB (Retired Product) SparkFun Xbee Explorer USB (New version)	PCB	1	24,950 €	24,950 €
SparkFun Converter (Retired Product) SparkFun Logic Level Converter Bi-Directional (New version)	PCB	1	2,950 €	2,950 €
MPU - 6050 Inercial Measurement Unit	PCB	1	6,500 €	6,500 €
TO-220 L7805C. Voltage regulator	PCB	1	0,430 €	0,430 €
0.22uF Capacitor	PCB	1	0,378 €	0,378 €
0.1uF Capacitor	PCB	2	0,171 €	0,342 €
0.01uF Capacitor	PCB	1	0,351 €	0,351 €
Green LED	PCB	1	0,118 €	0,118 €
Diode 1N4148	PCB	1	0,041 €	0,041 €
10K Resistor	PCB	1	0,100 €	0,100 €
33K Resistor	PCB	1	0,100 €	0,100 €
100K Resistor	PCB	1	0,100 €	0,100 €
Interruptor	PCB	1	1,220 €	1,220 €
Zócalo 3 pines macho	PCB	4	1,050 €	4,200 €
Zócalo 20 pines hembra	PCB	6	0,718 €	4,308 €
Electronic Speed Controller ESC	Potencia	4	7,000 €	28,000 €
Brushless Motors. BL-2208/14	Potencia	4	16,950 €	67,800 €
Batería 9V	Potencia	1	24,990 €	24,990 €
Cargador batería	Accesorios	1	10,990 €	10,990 €
Cableado	Accesorios	1	2,000 €	2,000 €
Tornillería	Estructura	1	3,000 €	3,000 €
Hélice 9x7	Estructura	4	4,100 €	16,400 €
Conos	Estructura	4	3,200 €	12,800 €
Sujeción del motor	Estructura	4	1,750 €	7,000 €
Barra aluminio 1x1x80cm	Estructura	2		0,000 €
TOTAL MATERIAL				313,768 €

MANO DE OBRA				1 UNIDAD
Diseño (Autónomo contrato externo por horas)	Personal	100	25	2.500 €
Construcción (Autónomo contrato externo por horas)	Personal	50	25	1.250 €
TOTAL MANO DE OBRA				3.750 €
SUBCONTRATACIÓN				1 UNIDAD
Fabricación de PCB 90x100 1 cara	Maquinaria	1	222,18 €	222,18 €
TOTAL SUBCONTRATACIÓN				222 €
COSTES INDIRECTOS				1 UNIDAD
Vida útil soldador. *	Maquinaria	0,000153	300	0,0459 €
Vida útil equipos informáticos. *	Maquinaria	0,0115	1000	11,5000 €
TOTAL COSTES INDIRECTOS				11,546 €
***Vida útil estimada a 3 años de funcionamiento diario durante 12 horas.** **Uso realizado:** **2 horas soldador.** **150 horas equipos informáticos.**				
SUMA FINAL DE PRESUPUESTO				4.297,494 €

En segundo lugar se presenta una relación de costes estimados para la fabricación de una serie de 100 unidades de este producto.

En esta situación se supone que el trabajo de diseño ya está realizado y no se evalúa.

Artículo	Clase	Unidades	Precio unidad	Coste de un ejemplar	Coste de 100 ejemplares
MATERIAL					100 UNIDADES
Arduino Nano v3.0 + Cable Macho USB - A <-> Macho MiniUSB -B	PCB	1	16,400 €	16,400 €	1.640,000 €
Arduino UNO rev. 3 + Cable Macho USB-A <-> Macho USB-B	PCB	1	15,000 €	15,000 €	1.500,000 €
XBee PRO ZB (S2) RF Module	PCB	2	20,660 €	41,320 €	4.132,000 €
XBee ZB Module Breakout Board	PCB	1	8,460 €	8,460 €	846,000 €
SparkFun Xbee Explorer USB (Retired Product) SparkFun Xbee Explorer USB (New version)	PCB	1	21,210 €	21,210 €	2.121,000 €
SparkFun Converter (Retired Product) SparkFun Logic Level Converter Bi-Directional (New version)	PCB	1	2,510 €	2,510 €	251,000 €
MPU - 6050 Inercial Measurement Unit	PCB	1	2,765 €	2,765 €	276,500 €
TO-220 L7805C. Voltage regulator	PCB	1	0,250 €	0,250 €	25,000 €
0.22uF Capacitor	PCB	1	0,301 €	0,301 €	30,100 €
0.1uF Capacitor	PCB	2	0,128 €	0,256 €	25,600 €
0.01uF Capacitor	PCB	1	0,280 €	0,280 €	28,000 €
Green LED	PCB	1	0,100 €	0,100 €	10,000 €
Diode 1N4148	PCB	1	0,017 €	0,017 €	1,700 €
10K Resistor	PCB	1	0,022 €	0,022 €	2,190 €
33K Resistor	PCB	1	0,022 €	0,022 €	2,190 €
100K Resistor	PCB	1	0,022 €	0,022 €	2,190 €
Interruptor	PCB	1	0,973 €	0,973 €	97,280 €
Zócalo 3 pines macho	PCB	4	0,980 €	3,920 €	392,000 €
Zócalo 20 pines hembra	PCB	6	0,554 €	3,324 €	332,400 €
Electronic Speed Controller ESC	Potencia	4	6,960 €	27,840 €	2.784,000 €
Brushless Motors. BL-2208/14	Potencia	4	12,650 €	50,600 €	5.060,000 €
Batería 9V	Potencia	1	24,390 €	24,390 €	2.439,000 €
Cargador batería	Accesorios	1	10,390 €	10,390 €	1.039,000 €
Cableado	Accesorios	1	1,500 €	1,500 €	150,000 €
Tornillería	Estructura	1	2,000 €	2,000 €	200,000 €
Hélice 9x7	Estructura	4	1,100 €	4,400 €	440,000 €
Conos	Estructura	4	2,150 €	8,600 €	860,000 €
Sujeción del motor	Estructura	4	1,750 €	7,000 €	700,000 €
Barra aluminio 1x1x80cm	Estructura	2		0,000 €	0,000 €
TOTAL MATERIAL				253,872 €	25.387,150 €

MANO DE OBRA					
Construcción - Ingeniero (1 persona en plantilla 5 meses)	Personal	5	2.800 €	140 €	14.000 €
Construcción - Personal auxiliar (2 personas en plantilla 5 meses)	Personal	10	1.600 €	160 €	16.000 €
TOTAL MANO DE OBRA				300 €	30.000 €
LABORES CONSTRUCCIÓN					
Fabricación de PCB 90x100 1 cara	Maquinaria	1	5,39 €	5,39 €	539,00 €
TOTAL LABORES DE CONSTRUCCIÓN				5,39 €	539 €
COSTES INDIRECTOS					
Vida útil soldador. *	Maquinaria	0,0152	300	0,0457 €	4,566 €
Vida útil equipos informáticos. *	Maquinaria	0,1522	1000	1,52 €	152,207 €
Logística y transporte	Logística	1	5 €	5 €	500 €
TOTAL COSTES INDIRECTOS				6,57 €	657 €
***Vida útil estimada a 3 años de funcionamiento diario durante 12 horas.** **Uso realizado:** **2 horas soldador por ejemplar.** **20 horas equipos informáticos por ejemplar.**					
SUMA FINAL DE PRESUPUESTO				565,829 €	56.582,923 €
			21% IVA	118,824 €	11882,423 €
			TOTAL	**684,653 €**	**68.465,323 €**

Hay ciertos aspectos que no se han tenido en cuenta como el local de trabajo, maquinaria a comprar, ajuste a normativas comerciales, coste de seguridad social de los empleados, seguro empresarial, etc.

Realizando un análisis superficial de los precios actuales del mercado, se ofertan cuadricópteros de distintas envergaduras y potencias, con una base de 400€ a nada que se busque un poco de fiabilidad.

Hay otros ejemplares con mayor calidad de materiales y precisión de control que alcanzan cifras de 10.000 o 15.000€.

Al estar realizado con un código abierto como Arduino, la venta de este producto podría proporcionar al cliente la oportunidad de controlar ciertos parámetros y realizar añadidos lo cual vuelve al producto un elemento muy versátil.

A su vez la envergadura del dron alcanza los 100 cm, un tamaño intermedio entre las ofertas actuales.

Por estas razones se ha estimado un precio recomendado de venta de 750€.

Una vez terminado el proyecto, vendidas todas las unidades, se obtendría un beneficio estimado de:

75.000,000€ - 68.465,323€ = 6534,677€

3.6 Conclusión

- Se ha diseñado el sistema completo de un "*quadcopter*" con el análisis de sus componentes y funciones y la cohesión entre ellos.

- El esquema eléctrico de conexiones del sistema ha sido diseñado y desarrollado mediante el software Design Spark.

- Se ha implementado el diseño en una tarjeta de circuito impreso y se han hecho las labores de soldadura y verificación.

- Una vez desarrollada la placa se ha montado en la estructura del dron a 45 grados respecto a las aspas y se ha cableado con el sistema de potencia.

- La programación del código en Arduino, permite la actuación de los distintos componentes y funciones de forma coordinada. Entre los objetivos alcanzados se encuentran la obtención del ángulo de los sensores, su conversión a un valor estable mediante el filtrado, la asignación de la acción de control de los actuadores, la medida de carga de la batería y la preparación del control PID para su aplicación.

- La configuración XBee ha sido implementada con éxito permitiendo la comunicación inalámbrica con el dron. A su vez en Arduino se ha conseguido realizar un código que permite el control de los parámetros del dron.

- El dron está operativo para su uso en el aprendizaje de los alumnos, ofreciéndoles un estudio del PID interactivo e innovador.

3.7 Líneas de futuro

- Desarrollo de la segunda versión de la placa con las observaciones corregidas.
- Construcción de un marco para el PCB para minimizar el estrés mecánico y el efecto de las vibraciones.
- Construcción mediante impresión 3D de una estructura para recubrir el dron y que disponga de apoyos.
- Construcción mediante impresión 3D de unas protecciones para las hélices
- Análisis de las variaciones producidas por los efectos de la temperatura en la resistividad y el estrés térmico.
- Aplicación de recubrimientos protectores para el PCB.
- Inclusión del compás HMC6352 y el sensor de presión BMP 180.
- Programación del filtro Kalman.
- Programación de un control PID de altura.
- Programación de una rutina de despegue.
- Programación de una rutina de aterrizaje.
-

3.8 Anexos

3.8.1 Anexo 0. Código completo de programación.

En este primer anexo se presenta el código de programación completo para analizar el sistema como una unidad global.

En los anexos consecutivos se muestran los extractos de las distintas secciones de programación, precedidas de un diagrama de bloques explicativo de cada funcionamiento.

```
/*
QUADTERA. Quadcopter Pantera. Arduino.cc
************************************************************************************
Usuario: Jose92. Fecha: Junio 2015.

*** English
This code allows the stabilization control of a drone. The drone has a sensor to feedback the control loop and a XBee module
to recieve information and act on the system variables.
For MPU-6050 sensor configuration a "Krodal" code part, called "MPU-6050 Accelerometer + Gyro", has been used.
************************************************************************************

*** Basque
Kode honek "dron" baten egonkortasunaren kontrola ahalbidetzen du.
Kontrol-lazoa ixteko sentsore bat dauka eta XBee modulu bat
zeinaren bidez sistemako parametroetan ekiteko informazioa jasotzen den.
HPU-6050 sentsorearen konfiguraziorako, "Krodal" erabiltzailearen
"MPU-6050 Accelerometer + Gyro" kodearen estraktu bat erabili da.
************************************************************************************

*** Castellano
Este código permite el control de estabilización de un dron.
Dispone de un sensor para cerrar el lazo de control y un módulo XBee
a través del cuál se recive información para actuar sobre los parámetros del sistema.
Para la configuración del sensor MPU-6050 se ha tomado un extracto del código del usuario "Krodal",
"MPU-6050 Accelerometer + Gyro".
************************************************************************************
*/



#include <Wire.h>
#include <math.h>
#include <Servo.h>

//****************************************************************************\\
// **** **** Configuracion MPU-6050. Codigo realizado por "Krodal" **** **** \\
#define MPU6050_ACCEL_XOUT_H       0x3B   // R
#define MPU6050_PWR_MGMT_1         0x6B   // R/W
#define MPU6050_PWR_MGMT_2         0x6C   // R/W
#define MPU6050_WHO_AM_I           0x75   // R


// Defines for the bits, to be able to change // between bit number and binary definition.
// By using the bit number, programming the sensor // is like programming the AVR microcontroller.
// But instead of using "(1<<X)", or "_BV(X)", // the Arduino "bit(X)" is used.#define MPU6050_D0 0
#define MPU6050_D1 1
#define MPU6050_D2 2
#define MPU6050_D3 3
#define MPU6050_D4 4
#define MPU6050_D5 5
#define MPU6050_D6 6
#define MPU6050_D7 7

// AUX_VDDIO Register
#define MPU6050_AUX_VDDIO MPU6050_D7  // I2C high: 1=VDD, 0=VLOGIC

// Default I2C address for the MPU-6050 is 0x68.// But only if the AD0 pin is low.
// Some sensor boards have AD0 high, and the// I2C address thus becomes 0x69.
#define MPU6050_I2C_ADDRESS 0x68


// Declaring an union for the registers and the axis values.
// The byte order does not match the byte order of // the compiler and AVR chip.
// The AVR chip (on the Arduino board) has the Low Byte // at the lower address.
// But the MPU-6050 has a different order: High Byte at // lower address, so that has to be corrected.
// The register part "reg" is only used internally, // and are swapped in code.
typedef union accel_t_gyro_union
```

```
{
  struct
  {
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  } reg;
  struct
  {
    int x_accel;
    int y_accel;
    int z_accel;
    int temperature;
    int x_gyro;
    int y_gyro;
    int z_gyro;
  } value;
};
// ==== ==== FIN. Configuracion MPU-6050. Codigo realizado por "Krodal"==== ==== \\
//================================================================================\\


//############################################################\\
//************************************************************\\
// **** **** DEFINICION DE VARIABLES DEL PROGRAMA **** **** \\
//************************************************************\\


//********************************************************************\\
// **** **** Definicion de lecturas de acelerometro y giroscopo **** **** \\
int readings[6];                              // Lecturas

float angulo_acc[3];                          // Angulo de cada acelerometro
float offsetAccelX, offsetAccelY, offsetAccelZ;   // Offsets acelerometro

float S_gyro[3];                              // Sensibilidad del giroscopo
float angulo_gyr[3];                          // Angulo de cada giroscopo
float offsetGyroX, offsetGyroY, offsetGyroZ;      // Offsets giroscopo
int dT=0;                                     // Tiempo de muestreo del giroscopo
double last_t=0;                              // Tiempo acumulado desde el inicio, para uso en el calculo del angulo del giroscopo
// ==== ==== FIN. Definicion de lecturas de acelerometro y giroscopo ==== ==== \\
//================================================================================\\


// **** **** Definicion de variables del Filtro Complementario **** **** \\
float angulo_cmp[3];                          // Angulo resultante del filtro complementario de cada eje
float alfa = 0.90;                            // Parametro de asignacion de valores al filtro paso alto y paso bajo
// ==== ==== FIN. Definicion de variables del Filtro Complementario ==== ==== \\
//================================================================================\\


////************************************************************\\
//// **** **** Definicion de variables del Filtro Kalman **** **** \\
//int gyroResult[3], accelResult[3];
//float timeStep = 0.02;
```

```
//unsigned long timer;
//float pitchGyro = 0;
//float pitchAccel = 0;
//float pitchPrediction = 0;                    //Salida filtro Kalman eje X
//float rollGyro = 0;
//float rollAccel = 0;
//float rollPrediction = 0;                     //Salida filtro Kalman eje Y
//float giroVar = 0.1;
//float deltaGiroVar = 0.1;                     //Parametro filtro Kalman
//float accelVar = 5;
//float Pxx = 0.1;                              // angle variance
//float Pvv = 0.1;                              // angle change rate variance
//float Pxv = 0.1;                              // angle and angle change rate covariance
//float kx, kv;
//float angulo_klm[3];
//// ==== ==== FIN. Definicion de variables Filtro Kalman ==== ==== \\
//////=============================================================\\


//*************************************************************\\
// **** **** Definicion de variables del control PID **** **** \\
int dT2=0;                                      // Tiempo de ciclo del control PID
double last_t2=0;                               // Tiempo acumulado desde el inicio, para su uso en el control PID
float Kp=1,Ki=0,Kd=0;                           // Constantes proporcional, integral y derivativa respectivamente

//ROLL. Control de giro en el eje X
float roll_referencia=0;                          //
float roll_error_prop;
float roll_error_int;
float roll_error_der;
float roll_error_prop_last;
float roll_output;

//PITCH. Control de giro en el eje Y
float pitch_referencia=0;
float pitch_error_prop=0;
float pitch_error_int=0;
float pitch_error_der=0;
float pitch_error_prop_last=0;
float pitch_output=0;

//YAW. Control de giro en el eje Z
float yaw_referencia=0;
float yaw_error_prop=0;
float yaw_error_int=0;
float yaw_error_der=0;
float yaw_error_prop_last=0;
float yaw_output=0;
// ==== ==== FIN. Definicion de variables del control PID ==== ==== \\
//=============================================================\\


//*********************************************************************\\
// **** **** Definicion de variables de la Acción de control **** **** \\
//Altura
float altura=110;
int pulso_min=800;
int pulso_max=2100;
int espera=0;
int accion_M[3];                      // Accion de control del motor M1,M2,M3,M4


Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;
```

```
// ==== ==== FIN. Definicion de variables de la Acción de control ==== ==== \\
//=========================================================================\\


//*************************************************************\\
// **** **** Definicion de variables comunicación XBee **** **** \\
char ch='0';
int posicion=0;
int terminado=0;
char lectura[8]={0,0,0,0,0,0,0,0};
char parametro[2]={0,0};
int poslect=0;
char medida[6]={0,0,0,0,0,0};
float valor;
// ==== ==== FIN. Definicion de variables comunicación XBee ==== ==== \\
//=============================================================\\


//*************************************************************\\
// **** **** Definición de variables de la medición de la batería **** **** \\
float voltaje_divisor=0;
float voltaje_bateria=0;
float lectura_analog_divisor;
// ==== ==== FIN. Definición de variables de la medición de la batería ==== ==== \\
//=============================================================\\


//*************************************************\\
// **** **** Definicion de otras variables **** **** \\
const float pi=3.141569;          // Numero PI
accel_t_gyro_union accel_t_gyro;  // Reestructuracion de datos
int row=0;                        // Para la función de escritura del Parallax DAQ
// ==== ==== FIN. Definicion de otras variables ==== ==== \\
//=====================================================\\


//=============================================================\\
// ==== ==== FIN. DEFINICION DE VARIABLES DEL PROGRAMA ==== ==== \\
//=============================================================\\
//#############################################################\\


//############################ SETUP ############################\\

void setup()
{


// **** **** Configuracion MPU-6050. Codigo realizado por "Krodal" **** **** \\
  int error;
  uint8_t c;
  // Initialize the 'Serial' class for the XBee reading.
  Serial.begin(57600);
  // Initialize the 'Wire' class for the I2C-bus.
  Wire.begin();
// default at power-up:  //    Gyro at 250 degrees second  //    Acceleration at 2g
//    Clock source at internal 8MHz  //    The device is in sleep mode.
  error = MPU6050_read (MPU6050_WHO_AM_I, &c, 1);
// According to the datasheet, the 'sleep' bit should read a '1'. But I read a '0'.
// That bit has to be cleared, since the sensor is in sleep mode at power-up. Even if the bit reads '0'.
  error = MPU6050_read (MPU6050_PWR_MGMT_2, &c, 1);
// Clear the 'sleep' bit to start the sensor.
  MPU6050_write_reg (MPU6050_PWR_MGMT_1, 0);
// ==== ==== FIN. Configuracion MPU-6050. Codigo realizado por "Krodal" ==== ==== \\
```

```
266
267
268 // **** **** Ajuste de ESC. Fase 1 **** **** \\
269   motor1.attach(3); //Localiza el servo en el pin 6. D3
270   motor2.attach(4,pulso_min,pulso_max); //Localiza el servo en el pin 7. D4
271   motor3.attach(5,pulso_min,pulso_max); //Localiza el servo en el pin 8. D5
272   motor4.attach(6,pulso_min,pulso_max); //Localiza el servo en el pin 9. D6
273   delay(100);
274   Serial.println("Write Microseconds pulso_max");
275   motor1.writeMicroseconds(pulso_max);
276   motor2.writeMicroseconds(pulso_max);
277   motor3.writeMicroseconds(pulso_max);
278   motor4.writeMicroseconds(pulso_max);
279   espera=millis();
280 // ==== ==== FIN. Ajuste de ESC. Fase 1 ==== ==== \\
281
282
283 // **** **** Inicialización del giróscopo **** **** \\
284 //  initialize values of angulo_gyr
285   for (int i=0; i<3; i++) {
286     angulo_gyr[i]=0;
287   }
288
289 // set the sensitivity of the gyros
290   S_gyro[0]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
291   S_gyro[1]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
292   S_gyro[2]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
293 // ==== ==== FIN. Inicialización del giróscopo ==== ==== \\
294
295
296 // **** **** Ajuste de offsets **** **** \\
297   int totalGyroXValues = 0;
298   int totalGyroYValues = 0;
299   int totalGyroZValues = 0;
300   int totalAccelXValues = 0;
301   int totalAccelYValues = 0;
302   int totalAccelZValues = 0;
303   int i;
304
305 // Calculo de los offsets
306   delay(100); //wait for gyro
307
308   for (i = 0; i < 50; i += 1)
309   {
310     // Subrutina del código de "Krodal". Lectura de valores del sensor.
311     error = leerdatos (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro, sizeof(accel_t_gyro));
312     totalGyroXValues += readings[3];
313     totalGyroYValues += readings[4];
314     totalGyroZValues += readings[5];
315     totalAccelXValues += readings[0];
316     totalAccelYValues += readings[1];
317     totalAccelZValues += readings[2];
318     delay(50);
319   }
320   offsetGyroX = totalGyroXValues / 50;
321   offsetGyroY = totalGyroYValues / 50;
322   offsetGyroZ = totalGyroZValues / 50;
323   offsetAccelX = totalAccelXValues / 50;
324   offsetAccelY = totalAccelYValues / 50;
325   offsetAccelZ = (totalAccelZValues / 50) - 256;
326
327   Serial.println(offsetGyroX);
328   Serial.println(offsetGyroY);
329   Serial.println(offsetGyroZ);
330   Serial.println(offsetAccelX);
331   Serial.println(offsetAccelY);
```

```
  Serial.println(offsetAccelZ);
// ==== ==== FIN. Ajuste de offsets ==== ==== \\


// **** **** Ajuste de ESC. Fase 2 **** **** \\
  while(millis()-espera<=2500);
  Serial.println("Write Microseconds pulso_min");
  motor1.writeMicroseconds(pulso_min);
  motor2.writeMicroseconds(pulso_min);
  motor3.writeMicroseconds(pulso_min);
  motor4.writeMicroseconds(pulso_min);
// ==== ==== FIN. Ajuste de ESC. Fase 1 ==== ==== \\


// **** **** Envio de datos a excel mediante el macro de PLX_DAQ, Parallax Data Acquisition tool **** **** \\
  //Configura captura de datos con Parallax_DAQ
  //Mostrará en Excel: Tiempo|ACC_X|ACC_Y|ACC_Z|GYRO_X|GYRO_Y|GYRO_Z|
  Serial.println("CLEARDATA");
  Serial.println("LABEL,TIME,ACC_X,ACC_Y,ACC_Z,GYRO_X,GYRO_Y,GYRO_Z,ANG_ACC_X,ANG_ACC_Y,ANG_GYRO_X,ANG_GYRO_Y,ANG_GYRO_Z,ANG_CMP_X,ANG_CMP_Y,ANG_CMP_Z");
// ==== ==== FIN. Envio de datos a excel mediante el macro de PLX_DAQ, Parallax Data Acquisition tool ==== ==== \\


}


//############################ LOOP ##############################\\

void loop()
{
  int error;
  // Subrutina del código de "Krodal". Lectura de valores del sensor.
  error = leerdatos (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro, sizeof(accel_t_gyro));

// **** **** Configuracion módulo de comunicación XBee **** **** \\
   if(Serial.available())
   {
     xbee();
     Serial.print("Parametro : ");
     Serial.print(parametro[0]);
     Serial.print(parametro[1]);
     Serial.print(" : ");
     Serial.println(valor,6);
     Serial.print("Alfa : ");
     Serial.println(alfa);
     Serial.print("Altura : ");
     Serial.println(altura);
     Serial.print("Roll Reference : ");
     Serial.println(roll_referencia);
     Serial.print("Pitch Reference : ");
     Serial.println(pitch_referencia);
     Serial.print("Yaw Reference : ");
     Serial.println(yaw_referencia);
     Serial.print("Ganancia proporcional Kp : ");
     Serial.println(Kp,6);
     Serial.print("Ganancia integral Ki : ");
     Serial.println(Ki,6);
     Serial.print("Ganancia derivativa Kd : ");
     Serial.println(Kd,6);
   }
// ==== ==== FIN. Configuracion módulo de comunicación XBee ==== ==== \\

// **** **** Obtención de ángulos **** **** \\
//Sustracción de los offsets previamente calculados
   readings[0]=readings[0]-offsetAccelX;
   readings[1]=readings[1]-offsetAccelY;
   readings[2]=readings[2]-offsetAccelZ;
   readings[3]=readings[3]-offsetGyroX;
```

```
  readings[4]=readings[4]-offsetGyroY;
  readings[5]=readings[5]-offsetGyroZ;

//Cálculo de los ángulos a partir de los acelerómetros
  angulo_acc[0]=atan2(readings[1],readings[2]);   //Tangente del angulo que forma el sensor con el eje X
  angulo_acc[0]*=57.29577;                        //Ajuste de radianes a grados sexagesimales.
  angulo_acc[1]=atan2(-readings[0],readings[2]);  //Tangente del angulo que forma el sensor con el eje Y
  angulo_acc[1]*=57.29577;                        //Ajuste de radianes a grados sexagesimales.

//Cálculo de los ángulos a partir de los giróscopos.
  dT=millis()-last_t;
  last_t=millis();
  for (int i=0; i<3; i++)
  {
    angulo_gyr[i]=angulo_gyr[i]+S_gyro[i]*readings[i+3]*dT;
  }
// ==== ==== FIN. Obtención de ángulos ==== ==== \\


// **** **** Filtrado de valores. Filtro complementario **** **** \\
//Fitrado complementario
  angulo_cmp[0] = alfa*(angulo_cmp[0]+(S_gyro[0]*readings[3])*dT)+(1-alfa)*angulo_acc[0];
  angulo_cmp[1] = alfa*(angulo_cmp[1]+(S_gyro[1]*readings[4])*dT)+(1-alfa)*angulo_acc[1];
  angulo_cmp[2] = angulo_gyr[2];
// ==== ==== FIN. Filtrado de valores. Filtro complementario ==== ==== \\


//// **** **** Filtrado de valores. Filtro Kalman **** **** \\
//  timer = millis();
//  getGyroscopeReadings(gyroResult);
//  getAccelerometerReadings(accelResult);
//
//  pitchAccel = atan2((accelResult[1] - offsetAccelY) / 256, (accelResult[2] - offsetAccelZ) / 256) * 360.0 / (2*PI);
//  pitchGyro = pitchGyro + ((gyroResult[0] - offsetGyroX) / 14.375) * timeStep;
//  pitchPrediction = pitchPrediction + ((gyroResult[0] - offsetGyroX) / 14.375) * timeStep;
//
//  rollAccel = atan2((accelResult[0] - offsetAccelX) / 256, (accelResult[2] - offsetAccelZ) / 256) * 360.0 / (2*PI);
//  rollGyro = rollGyro - ((gyroResult[1] - offsetGyroY) / 14.375) * timeStep;
//  rollPrediction = rollPrediction - ((gyroResult[1] - offsetGyroY) / 14.375) * timeStep;
//
//  Pxx += timeStep * (2 * Pxv + timeStep * Pvv);
//  Pxv += timeStep * Pvv;
//  Pxx += timeStep * giroVar;
//  Pvv += timeStep * deltaGiroVar;
//  kx = Pxx * (1 / (Pxx + accelVar));
//  kv = Pxv * (1 / (Pxx + accelVar));
//
//  pitchPrediction += (pitchAccel - pitchPrediction) * kx;
//  rollPrediction += (rollAccel - rollPrediction) * kx;
//
//  Pxx *= (1 - kx);
//  Pxv *= (1 - kx);
//  Pvv -= kv * Pxv;
//
//  Serial.print(pitchGyro);
//  Serial.print("\t");
//  Serial.print(pitchAccel);
//  Serial.print("\t");
//  Serial.print(pitchPrediction);
//  Serial.print("\t");
//  Serial.print(rollGyro);
//  Serial.print("\t");
//  Serial.print(rollAccel);
//  Serial.print("\t");
//  Serial.print(rollPrediction);
//  Serial.print("\n");
```

```
//  Serial.print("\t");
//  Serial.print(rollPrediction);
//  Serial.print("\n");
//
//  timer = millis() - timer;
//  timer = (timeStep * 1000) - timer;
//// ==== ==== FIN. Filtrado de valores. Filtro Kalman ==== ==== \\


// **** **** Implementación del control PID **** **** \\
  dT2=millis()-last_t2;
  last_t2=millis();

  // **** ROLL PID ****
  //Proporcional
  roll_error_prop=angulo_cmp[0]-roll_referencia;
  //Integral
  roll_error_int+=roll_error_prop*dT2/1000;
  //Anti WindUP
  if(roll_error_int>=180)
  {
    roll_error_int=180;
  }
  if(roll_error_int<=0)
  {
    roll_error_int=0;
  }
  //Derivativo
  roll_error_der=(roll_error_prop-roll_error_prop_last)/(dT2/1000);
  //Resultado
  roll_output=Kp*roll_error_prop+Ki*roll_error_int-Kd*roll_error_der;
  //Anti WindUP 2
  //Anti WindUP 2
  if(roll_output>=180)
  {
    roll_output=180;
  }
  if(roll_output<=0)
  {
    roll_output=0;
  }
  //Otros valores
  roll_error_prop_last=roll_error_prop;

  // **** PITCH PID ****
  //Proporcional
  pitch_error_prop=angulo_cmp[1]-pitch_referencia;
  //Integral
  pitch_error_int+=pitch_error_prop*dT2/1000;
  //Anti WindUP
  if(pitch_error_int>=180)
  {
    pitch_error_int=180;
  }
  if(pitch_error_int<=0)
  {
    pitch_error_int=0;
  }
  //Derivativo
//  pitch_error_der=(pitch_error_prop-pitch_error_prop_last)/(dT2/1000);
  //Resultado
  pitch_output=Kp*pitch_error_prop+Ki*pitch_error_int; //-Kd*pitch_error_der;
  //Anti WindUP 2
  if(pitch_output>=180)
  {
```

```
    pitch_output=180;
  }
  if(pitch_output<=-180)
  {
    pitch_output=-180;
  }
  //Otros valores
  pitch_error_prop_last=pitch_error_prop;
// ==== ==== FIN. Implementación del control PID ==== ==== \\


// **** **** Acción de control **** **** \\
//Acción de control para posicion X
  accion_M[0]=altura-pitch_output-roll_output;
  accion_M[1]=altura+pitch_output-roll_output;
  accion_M[2]=altura-pitch_output+roll_output;
  accion_M[3]=altura+pitch_output+roll_output;

  //Acción de control para posicion + (plus)
  accion_M[0]=altura+pitch_output;
  accion_M[1]=altura+roll_output;
  accion_M[2]=altura-roll_output;
  accion_M[3]=altura-pitch_output;
  for(int i=0;i<=4;i+=1) //Anti WindUP 2
  {
    if(accion_M[i]>=180)
    {
      accion_M[i]=180;
    }
    if(accion_M[i]<=0)
    {
      accion_M[i]=0;
    }
  }

  //Activacion de los motores
  motor1.write(accion_M[0]);
  motor2.write(accion_M[1]);
  motor3.write(accion_M[2]);
  motor4.write(accion_M[3]);
// ==== ==== FIN. Acción de control ==== ==== \\


// **** **** Medida de bateria y control de descenso **** **** \\
 lectura_analog_divisor=analogRead(1);
 delay(100);
 voltaje_divisor=lectura_analog_divisor*5/1024; //3V sera la entrada máxima al pin analógico.
 voltaje_bateria=voltaje_divisor/0.248;         //Cuando se lean 3V significará que hay 12V

 if(voltaje_bateria<=9)
 {
   if(altura>-180)
   {
     --altura;
   }
 }
 Serial.print("Altura : ");
 Serial.println(altura);
// ==== ==== FIN. Medida de bateria y control de descenso ==== ==== \\


// **** **** Escritura de datos **** **** \\
// Se utiliza la subrutina print_RAW
  print_RAW();
// ==== ==== FIN. Escritura de datos ==== ==== \\
//    Serial.print("Parametro : ");
//    Serial.print(parametro[0]);
//    Serial.print(parametro[1]);
//    Serial.print(" : ");
```

```
//    Serial.println(alfa);
//    Serial.print("Altura : ");
//    Serial.println(altura);
//    Serial.print("Roll Reference : ");
//    Serial.println(roll_referencia);
//    Serial.print("Pitch Reference : ");
//    Serial.println(pitch_referencia);
//    Serial.print("Yaw Reference : ");
//    Serial.println(yaw_referencia);
//    Serial.print("Ganancia proporcional Kp : ");
//    Serial.println(Kp,6);
//    Serial.print("Ganancia integral Ki : ");
//    Serial.println(Ki,6);
//    Serial.print("Ganancia derivativa Kd : ");
//    Serial.println(Kd,6);

// **** **** Captura de datos con Parallax_DAQ **** **** \\
  //Mostrará en Excel: Tiempo|ACC_X|ACC_Y|ACC_Z|GYRO_X|GYRO_Y|GYRO_Z|...
  Serial.print(F("DATA,TIME, "));
  Serial.print(accel_t_gyro.value.x_accel,DEC);
  Serial.print(F(", "));
  Serial.print(accel_t_gyro.value.y_accel,DEC);
  Serial.print(F(", "));
  Serial.print(accel_t_gyro.value.z_accel,DEC);
  Serial.print(F(", "));
  Serial.print(accel_t_gyro.value.x_gyro,DEC);
  Serial.print(F(", "));
  Serial.print(accel_t_gyro.value.y_gyro,DEC);
  Serial.print(F(", "));
  Serial.print(accel_t_gyro.value.z_gyro,DEC);
  Serial.print(F(", "));
  Serial.print(angulo_acc[0],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_acc[1],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_gyr[0],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_gyr[1],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_gyr[2],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_cmp[0],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_cmp[1],DEC);
  Serial.print(F(", "));
  Serial.print(angulo_cmp[2],DEC);
  Serial.print(accion_M[1]);
  Serial.print(";");
  Serial.print(accion_M[2]);
  Serial.print(";");
  Serial.print(accion_M[3]);
  Serial.print(";");
  Serial.print(accion_M[4]);
  Serial.print(";");
  Serial.print(roll_output);
  Serial.print(";");
  Serial.print(pitch_output);
  Serial.print(";");
  Serial.print(pitch_referencia);
  Serial.print(";");
  Serial.print(pitch_error_prop);
  Serial.print(";");
  Serial.print(pitch_error_der);
  Serial.print(";");
  Serial.print(pitch_error_int);
  Serial.print(";");
```

```
663   Serial.print(pitch_error_prop_last);
664   Serial.print(";");
665   Serial.print(pitch_output);
666   Serial.println(";");
667 
668 // Con esto realiza la sobreescritura de los valores al llegar al número 1000
669   row++;
670   if (row > 1000)
671   {
672     row=0;
673     Serial.println("ROW,SET,2");
674   }
675   delay(50);
676 // ==== ==== FIN. Captura de datos con Parallax_DAQ ==== ==== \\
677 
678 }
679 
680 
681 
```

3.8.3 Anexo 1. Lectura del sensor MPU 6050.

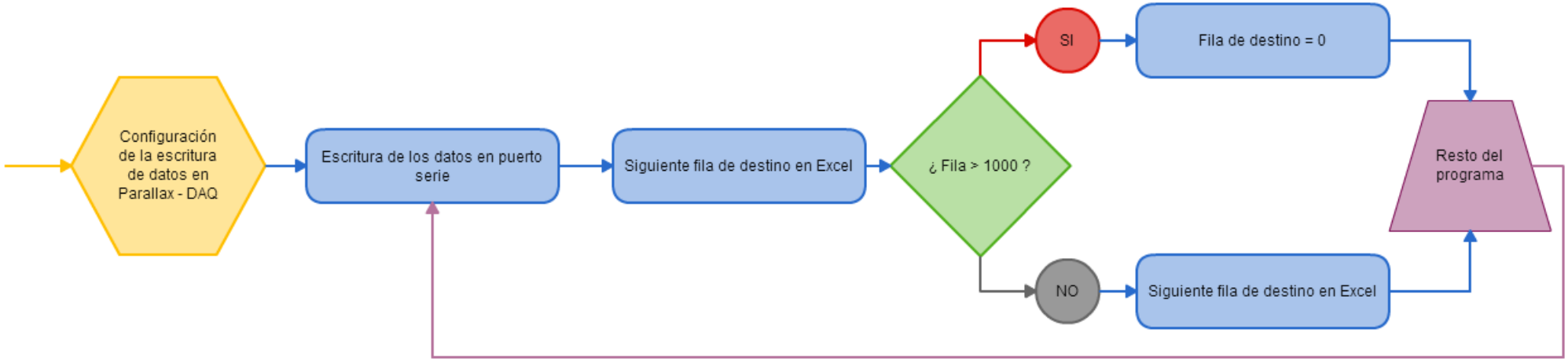
```
//****************************************************************************\\
// **** **** Configuracion MPU-6050. Codigo realizado por "Krodal" **** **** \\
#define MPU6050_ACCEL_XOUT_H       0x3B   // R
#define MPU6050_PWR_MGMT_1         0x6B   // R/W
#define MPU6050_PWR_MGMT_2         0x6C   // R/W
#define MPU6050_WHO_AM_I           0x75   // R


// Defines for the bits, to be able to change // between bit number and binary definition.
// By using the bit number, programming the sensor // is like programming the AVR microcontroller.
// But instead of using "(1<<X)", or "_BV(X)", // the Arduino "bit(X)" is used.#define MPU6050_D0 0
#define MPU6050_D1 1
#define MPU6050_D2 2
#define MPU6050_D3 3
#define MPU6050_D4 4
#define MPU6050_D5 5
#define MPU6050_D6 6
#define MPU6050_D7 7

// AUX_VDDIO Register
#define MPU6050_AUX_VDDIO MPU6050_D7  // I2C high: 1=VDD, 0=VLOGIC

// Default I2C address for the MPU-6050 is 0x68.// But only if the AD0 pin is low.
// Some sensor boards have AD0 high, and the// I2C address thus becomes 0x69.
#define MPU6050_I2C_ADDRESS 0x68


// Declaring an union for the registers and the axis values.
// The byte order does not match the byte order of // the compiler and AVR chip.
// The AVR chip (on the Arduino board) has the Low Byte // at the lower address.
// But the MPU-6050 has a different order: High Byte at // lower address, so that has to be corrected.
// The register part "reg" is only used internally, // and are swapped in code.
typedef union accel_t_gyro_union
{
  struct
  {
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  } reg;
  struct
  {
    int x_accel;
    int y_accel;
    int z_accel;
    int temperature;
    int x_gyro;
    int y_gyro;
    int z_gyro;
  } value;
};
// ==== ==== FIN. Configuracion MPU-6050. Codigo realizado por "Krodal"==== ==== \\
//============================================================================\\
```

```
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {
250 // **** **** Configuracion MPU-6050. Codigo realizado por "Krodal" **** **** \\
251   int error;
252   uint8_t c;
253   // Initialize the 'Serial' class for the XBee reading.
254   Serial.begin(57600);
255   // Initialize the 'Wire' class for the I2C-bus.
256   Wire.begin();
257 // default at power-up:  //    Gyro at 250 degrees second  //    Acceleration at 2g
258 //    Clock source at internal 8MHz  //    The device is in sleep mode.
259   error = MPU6050_read (MPU6050_WHO_AM_I, &c, 1);
260 // According to the datasheet, the 'sleep' bit should read a '1'. But I read a '0'.
261 // That bit has to be cleared, since the sensor is in sleep mode at power-up. Even if the bit reads '0'.
262   error = MPU6050_read (MPU6050_PWR_MGMT_2, &c, 1);
263 // Clear the 'sleep' bit to start the sensor.
264   MPU6050_write_reg (MPU6050_PWR_MGMT_1, 0);
265 // ==== ==== FIN. Configuracion MPU-6050. Codigo realizado por "Krodal" ==== ==== \\
354 }
357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
361   int error;
362   // Subrutina del código de "Krodal". Lectura de valores del sensor.
363   error = leerdatos (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro, sizeof(accel_t_gyro));
678 }
  1 // ----------------------------------------------------------
  2 // MPU6050_read
  3 //
  4 // This is a common function to read multiple bytes
  5 // from an I2C device.
  6 //
  7 // It uses the boolean parameter for Wire.endTransMission()
  8 // to be able to hold or release the I2C-bus.
  9 // This is implemented in Arduino 1.0.1.
 10 //
 11 // Only this function is used to read.
 12 // There is no function for a single byte.
 13 //
 14 int MPU6050_read(int start, uint8_t *buffer, int size)
 15 {
 16   int i, n, error;
 17
 18   Wire.beginTransmission(MPU6050_I2C_ADDRESS);
 19   n = Wire.write(start);
 20   if (n != 1)
 21     return (-10);
 22
 23   n = Wire.endTransmission(false);    // hold the I2C-bus
 24   if (n != 0)
 25     return (n);
 26
 27   // Third parameter is true: relase I2C-bus after data is read.
 28   Wire.requestFrom(MPU6050_I2C_ADDRESS, size, true);
 29   i = 0;
 30   while(Wire.available() && i<size)
 31   {
 32     buffer[i++]=Wire.read();
 33   }
 34   if ( i != size)
 35     return (-11);
 36
 37   return (0);  // return : no error
 38 }
```

```
// --------------------------------------------------------
// MPU6050_write_reg
//
// An extra function to write a single register.
// It is just a wrapper around the MPU_6050_write()
// function, and it is only a convenient function
// to make it easier to write a single register.
//
int MPU6050_write_reg(int reg, uint8_t data)
{
  int error;

  error = MPU6050_write(reg, &data, 1);

  return (error);
}
// --------------------------------------------------------
// MPU6050_write
//
// This is a common function to write multiple bytes to an I2C device.
//
// If only a single register is written,
// use the function MPU_6050_write_reg().
//
// Parameters:
//   start : Start address, use a define for the register
//   pData : A pointer to the data to write.
//   size  : The number of bytes to write.
//
// If only a single register is written, a pointer
// to the data has to be used, and the size is
// a single byte:
//   int data = 0;        // the data to write
//   MPU6050_write (MPU6050_PWR_MGMT_1, &c, 1);
//
int MPU6050_write(int start, const uint8_t *pData, int size)
{
  int n, error;

  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
  n = Wire.write(start);        // write the start address
  if (n != 1)
    return (-20);

  n = Wire.write(pData, size);  // write data bytes
  if (n != size)
    return (-21);

  error = Wire.endTransmission(true); // release the I2C-bus
  if (error != 0)
```

```
int leerdatos(int start, uint8_t *buffer, int size)
{
  int error;

  // Read the raw values. Read 14 bytes at once, containing acceleration, temperature and gyro.
  // With the default settings of the MPU-6050, there is no filter enabled, and the values are not very stable.
  error = MPU6050_read (start, buffer, size);
  // Swap all high and low bytes. After this, the registers values are swapped,
  // so the structure name like x_accel_l does no longer contain the lower byte.

  uint8_t swap;
  #define SWAP(x,y) swap = x; x = y; y = swap

  SWAP (accel_t_gyro.reg.x_accel_h, accel_t_gyro.reg.x_accel_l);
  SWAP (accel_t_gyro.reg.y_accel_h, accel_t_gyro.reg.y_accel_l);
  SWAP (accel_t_gyro.reg.z_accel_h, accel_t_gyro.reg.z_accel_l);
  SWAP (accel_t_gyro.reg.t_h, accel_t_gyro.reg.t_l);
  SWAP (accel_t_gyro.reg.x_gyro_h, accel_t_gyro.reg.x_gyro_l);
  SWAP (accel_t_gyro.reg.y_gyro_h, accel_t_gyro.reg.y_gyro_l);
  SWAP (accel_t_gyro.reg.z_gyro_h, accel_t_gyro.reg.z_gyro_l);

  readings[0]=accel_t_gyro.value.x_accel;
  readings[1]=accel_t_gyro.value.y_accel;
  readings[2]=accel_t_gyro.value.z_accel;
  readings[3]=accel_t_gyro.value.x_gyro;
  readings[4]=accel_t_gyro.value.y_gyro;
  readings[5]=accel_t_gyro.value.z_gyro;

  return (0);         // return : no error
}
```

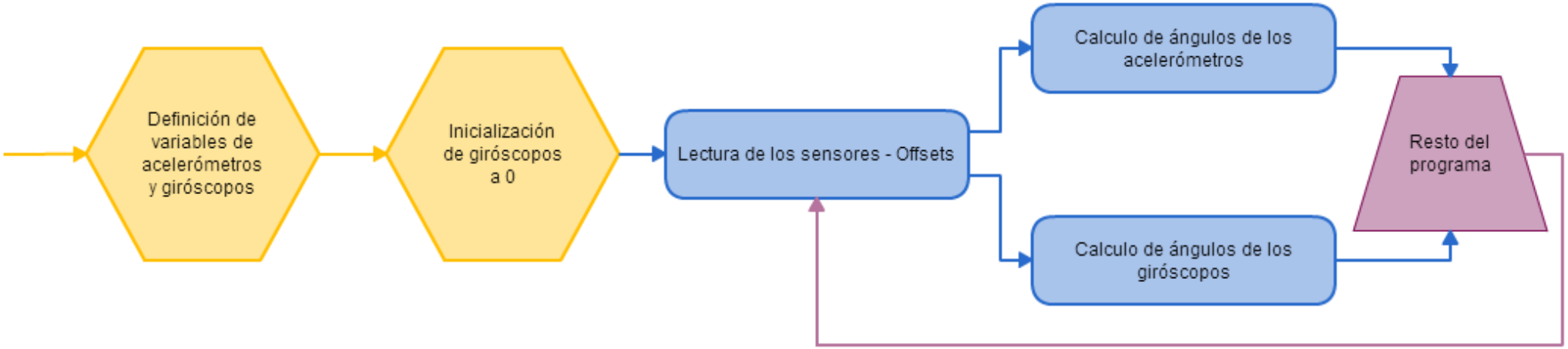
3.8.4 Anexo 2. Código de escritura en Parallax-DAQ tool.



```
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {
346 // **** **** Envío de datos a excel mediante el macro de PLX_DAQ, Parallax Data Acquisition tool **** **** \\
347   //Configura captura de datos con Parallax_DAQ
348   //Mostrará en Excel: Tiempo|ACC_X|ACC_Y|ACC_Z|GYRO_X|GYRO_Y|GYRO_Z|
349   Serial.println("CLEARDATA");
350   Serial.println("LABEL,TIME,ACC_X,ACC_Y,ACC_Z,GYRO_X,GYRO_Y,GYRO_Z,ANG_ACC_X,ANG_ACC_Y,ANG_GYRO_X,ANG_GYRO_Y,ANG_GYRO_Z,ANG_CMP_X,ANG_CMP_Y,ANG_CMP_Z");
351 // ==== ==== FIN. Envío de datos a excel mediante el macro de PLX_DAQ, Parallax Data Acquisition tool ==== ==== \\

357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
613 // **** **** Captura de datos con Parallax_DAQ **** **** \\
614   //Mostrará en Excel: Tiempo|ACC_X|ACC_Y|ACC_Z|GYRO_X|GYRO_Y|GYRO_Z|...
615   Serial.print(F("DATA,TIME, "));
616   Serial.print(accel_t_gyro.value.x_accel,DEC);
617   Serial.print(F(", "));
618   Serial.print(accel_t_gyro.value.y_accel,DEC);
619   Serial.print(F(", "));
620   Serial.print(accel_t_gyro.value.z_accel,DEC);
621   Serial.print(F(", "));
622   Serial.print(accel_t_gyro.value.x_gyro,DEC);
623   Serial.print(F(", "));
624   Serial.print(accel_t_gyro.value.y_gyro,DEC);
625   Serial.print(F(", "));
626   Serial.print(accel_t_gyro.value.z_gyro,DEC);
627   Serial.print(F(", "));
628   Serial.print(angulo_acc[0],DEC);
629   Serial.print(F(", "));
630   Serial.print(angulo_acc[1],DEC);
631   Serial.print(F(", "));
632   Serial.print(angulo_gyr[0],DEC);
633   Serial.print(F(", "));
634   Serial.print(angulo_gyr[1],DEC);
635   Serial.print(F(", "));
636   Serial.print(angulo_gyr[2],DEC);
637   Serial.print(F(", "));
638   Serial.print(angulo_cmp[0],DEC);
639   Serial.print(F(", "));
640   Serial.print(angulo_cmp[1],DEC);
641   Serial.print(F(", "));
642   Serial.print(angulo_cmp[2],DEC);
643   Serial.print(accion_M[1]);
644   Serial.print(";");
645   Serial.print(accion_M[2]);
646   Serial.print(";");
```

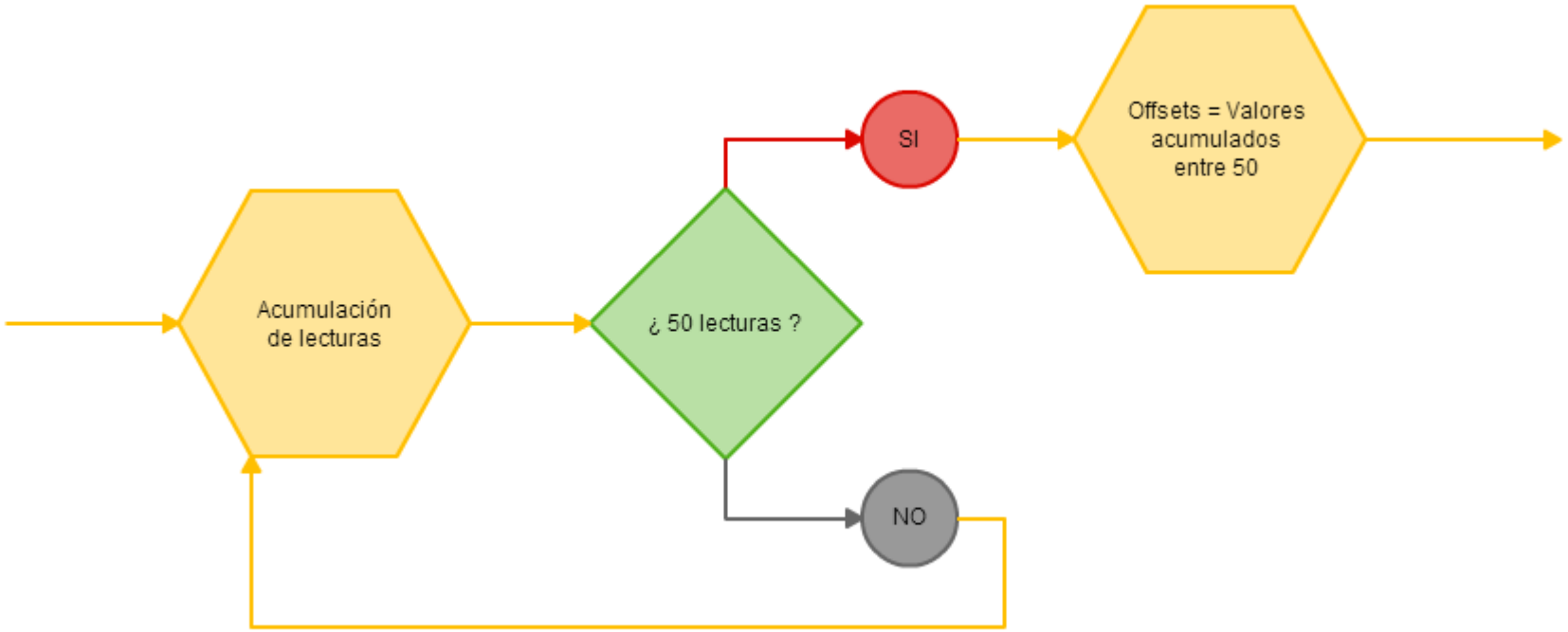
```
647   Serial.print(accion_M[3]);
648   Serial.print(";");
649   Serial.print(accion_M[4]);
650   Serial.print(";");
651   Serial.print(roll_output);
652   Serial.print(";");
653   Serial.print(pitch_output);
654   Serial.print(";");
655   Serial.print(pitch_referencia);
656   Serial.print(";");
657   Serial.print(pitch_error_prop);
658   Serial.print(";");
659   Serial.print(pitch_error_der);
660   Serial.print(";");
661   Serial.print(pitch_error_int);
662   Serial.print(";");
663   Serial.print(pitch_error_prop_last);
664   Serial.print(";");
665   Serial.print(pitch_output);
666   Serial.println(";");
667
668 // Con esto realiza la sobreescritura de los valores al llegar al número 1000
669   row++;
670   if (row > 1000)
671   {
672     row=0;
673     Serial.println("ROW,SET,2");
674   }
675   delay(50);
676 // ==== ==== FIN. Captura de datos con Parallax_DAQ ==== ==== \\
677
678 }
```

3.8.5 Anexo 3. Código programación acelerómetro y giróscopo



```
//*************************************************************************\\
// **** **** Definicion de lecturas de acelerometro y giroscopo **** **** \\
int readings[6];                                // Lecturas

float angulo_acc[3];                            // Angulo de cada acelerometro
float offsetAccelX, offsetAccelY, offsetAccelZ;   // Offsets acelerometro

float S_gyro[3];                                // Sensibilidad del giroscopo
float angulo_gyr[3];                            // Angulo de cada giroscopo
float offsetGyroX, offsetGyroY, offsetGyroZ;      // Offsets giroscopo
int dT=0;                                       // Tiempo de muestreo del giroscopo
double last_t=0;                                // Tiempo acumulado desde el inicio, para uso en el calculo del angulo del giroscopo
// ==== ==== FIN. Definicion de lecturas de acelerometro y giroscopo ==== ==== \\
//=========================================================================\\
//############################ SETUP ############################\\

void setup()
{
// **** **** Inicialización del giróscopo **** **** \\
//  initialize values of angulo_gyr
  for (int i=0; i<3; i++) {
    angulo_gyr[i]=0;
  }

// set the sensitivity of the gyros
  S_gyro[0]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
  S_gyro[1]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
  S_gyro[2]=0.00000762939453125;     // Sensibilidad= 250/(32768*1000)
// ==== ==== FIN. Inicialización del giróscopo ==== ==== \\

//############################ LOOP ############################\\

void loop()
{
// **** **** Obtención de ángulos **** **** \\
//Sustracción de los offsets previamente calculados
  readings[0]=readings[0]-offsetAccelX;
  readings[1]=readings[1]-offsetAccelY;
  readings[2]=readings[2]-offsetAccelZ;
  readings[3]=readings[3]-offsetGyroX;
  readings[4]=readings[4]-offsetGyroY;
  readings[5]=readings[5]-offsetGyroZ;

//Cálculo de los ángulos a partir de los acelerómetros
  angulo_acc[0]=atan2(readings[1],readings[2]);   //Tangente del angulo que forma el sensor con el eje X
  angulo_acc[0]*=57.29577;                        //Ajuste de radianes a grados sexagesimales.
  angulo_acc[1]=atan2(-readings[0],readings[2]);  //Tangente del angulo que forma el sensor con el eje Y
  angulo_acc[1]*=57.29577;                        //Ajuste de radianes a grados sexagesimales.

//Cálculo de los ángulos a partir de los giróscopos.
  dT=millis()-last_t;
  last_t=millis();
  for (int i=0; i<3; i++)
  {
    angulo_gyr[i]=angulo_gyr[i]+S_gyro[i]*readings[i+3]*dT;
  }
// ==== ==== FIN. Obtención de ángulos ==== ==== \\
}
```

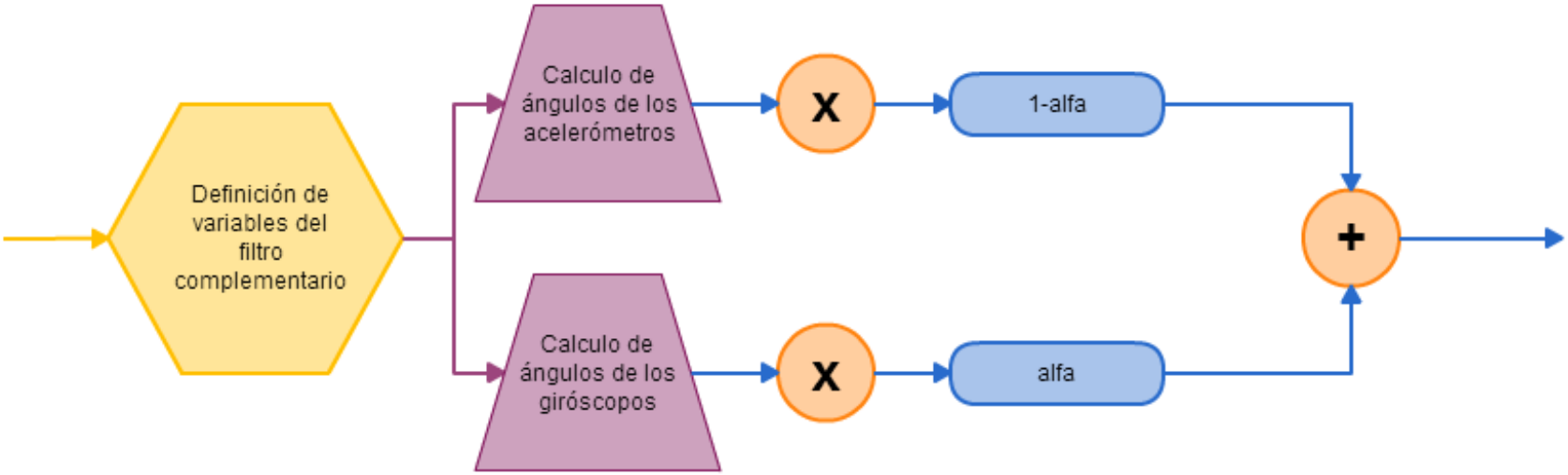
3.8.6 Anexo 4. Calculo automático de offsets.



Definición de las variables en el anexo anterior.

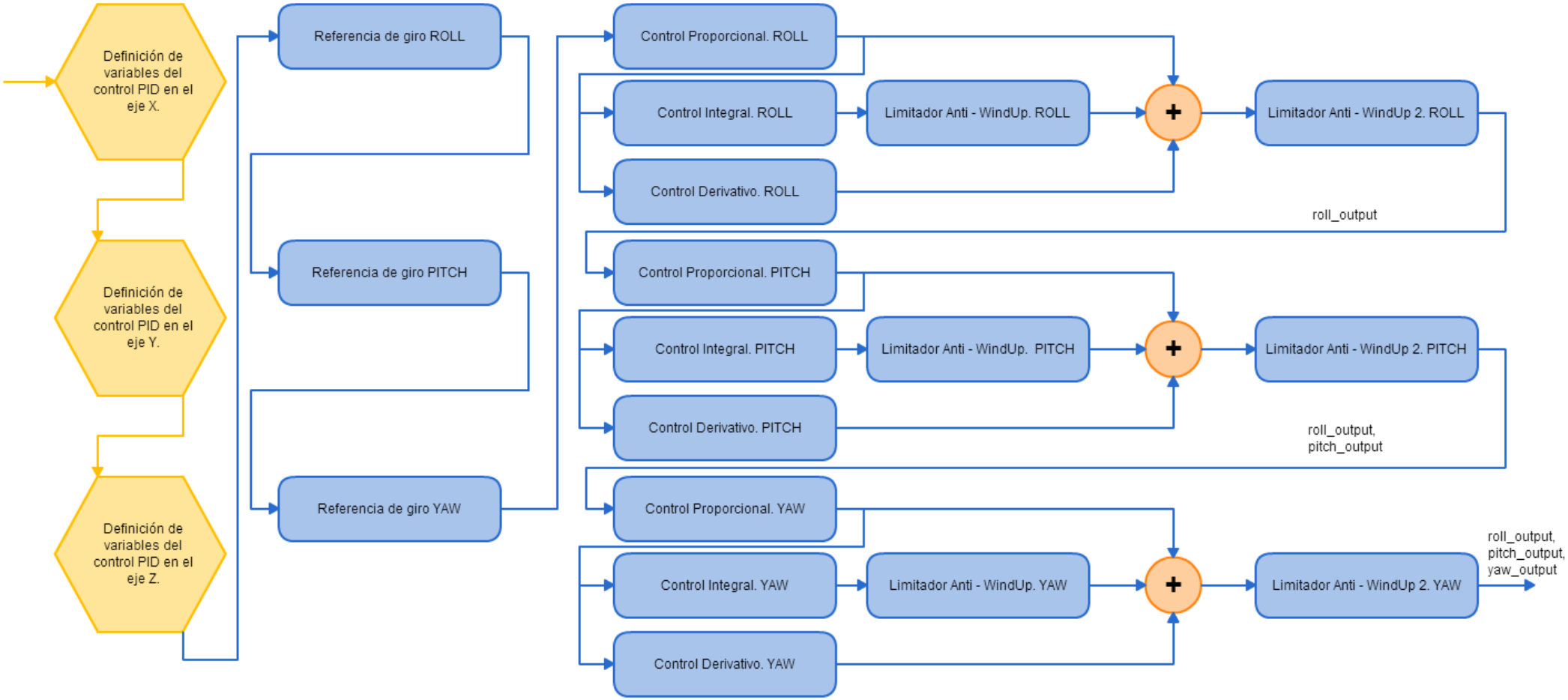
```
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {
305 // Calculo de los offsets
306   delay(100); //wait for gyro
307
308   for (i = 0; i < 50; i += 1)
309   {
310     // Subrutina del código de "Krodal". Lectura de valores del sensor.
311     error = leerdatos (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro, sizeof(accel_t_gyro));
312     totalGyroXValues += readings[3];
313     totalGyroYValues += readings[4];
314     totalGyroZValues += readings[5];
315     totalAccelXValues += readings[0];
316     totalAccelYValues += readings[1];
317     totalAccelZValues += readings[2];
318     delay(50);
319   }
320   offsetGyroX = totalGyroXValues / 50;
321   offsetGyroY = totalGyroYValues / 50;
322   offsetGyroZ = totalGyroZValues / 50;
323   offsetAccelX = totalAccelXValues / 50;
324   offsetAccelY = totalAccelYValues / 50;
325   offsetAccelZ = (totalAccelZValues / 50) - 256;
326
327   Serial.println(offsetGyroX);
328   Serial.println(offsetGyroY);
329   Serial.println(offsetGyroZ);
330   Serial.println(offsetAccelX);
331   Serial.println(offsetAccelY);
332   Serial.println(offsetAccelZ);
333 // ==== ==== FIN. Ajuste de offsets ==== ==== \\

357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
678 }
```

3.8.7 Anexo 5. Filtro complementario



```
123 // **** **** Definicion de variables del Filtro Complementario **** **** \\
124 float angulo_cmp[3];                          // Angulo resultante del filtro complementario de cada eje
125 float alfa = 0.90;                            // Parametro de asignacion de valores al filtro paso alto y paso bajo
126 // ==== ==== FIN. Definicion de variables del Filtro Complementario ==== ==== \\
127 //=============================================================================\\
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {

357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
418 // **** **** Filtrado de valores. Filtro complementario **** **** \\
419 //Fitrado complementario
420   angulo_cmp[0] = alfa*(angulo_cmp[0]+(S_gyro[0]*readings[3])*dT)+(1-alfa)*angulo_acc[0];
421   angulo_cmp[1] = alfa*(angulo_cmp[1]+(S_gyro[1]*readings[4])*dT)+(1-alfa)*angulo_acc[1];
422   angulo_cmp[2] = angulo_gyr[2];
423 // ==== ==== FIN. Filtrado de valores. Filtro complementario ==== ==== \\
678 }
```

3.8.8 Anexo 6. Control PID

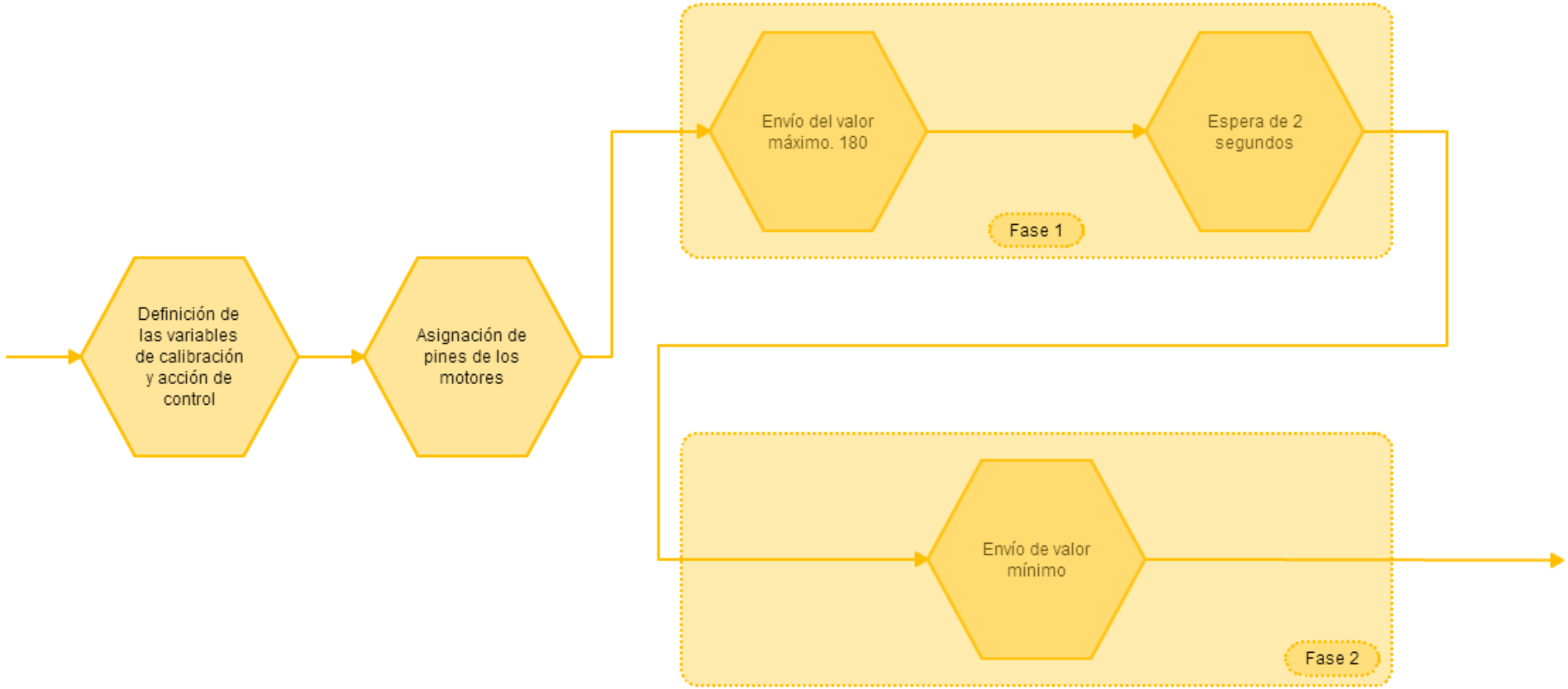


```
153 //************************************************************\\
154 // **** **** Definicion de variables del control PID **** **** \\
155 int dT2=0;                                    // Tiempo de ciclo del control PID
156 double last_t2=0;                             // Tiempo acumulado desde el inicio, para su uso en el control PID
157 float Kp=1,Ki=0,Kd=0;                         // Constantes proporcional, integral y derivativa respectivamente
158
159 //ROLL. Control de giro en el eje X
160 float roll_referencia=0;                        //
161 float roll_error_prop;
162 float roll_error_int;
163 float roll_error_der;
164 float roll_error_prop_last;
165 float roll_output;
166
167 //PITCH. Control de giro en el eje Y
168 float pitch_referencia=0;
169 float pitch_error_prop=0;
170 float pitch_error_int=0;
171 float pitch_error_der=0;
172 float pitch_error_prop_last=0;
173 float pitch_output=0;
174
175 //YAW. Control de giro en el eje Z
176 float yaw_referencia=0;
177 float yaw_error_prop=0;
178 float yaw_error_int=0;
179 float yaw_error_der=0;
180 float yaw_error_prop_last=0;
181 float yaw_output=0;
182 // ==== ==== FIN. Definicion de variables del control PID ==== ==== \\
183 //============================================================\\
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {
678 }
357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
473 // **** **** Implementación del control PID **** **** \\
474   dT2=millis()-last_t2;
475   last_t2=millis();
476
477   // **** ROLL PID ****
478   //Proporcional
479   roll_error_prop=angulo_cmp[0]-roll_referencia;
480   //Integral
481   roll_error_int+=roll_error_prop*dT2/1000;
482   //Anti WindUP
483   if(roll_error_int>=180)
484   {
485     roll_error_int=180;
486   }
487   if(roll_error_int<=0)
488   {
489     roll_error_int=0;
490   }
491   //Derivativo
492   roll_error_der=(roll_error_prop-roll_error_prop_last)/(dT2/1000);
493   //Resultado
494   roll_output=Kp*roll_error_prop+Ki*roll_error_int-Kd*roll_error_der;
495   //Anti WindUP 2
496   if(roll_output>=180)
497   {
498     roll_output=180;
499   }
500   if(roll_output<=0)
501   {
502     roll_output=0;
503   }
```

```
  //Otros valores
  roll_error_prop_last=roll_error_prop;

  // **** PITCH PID ****
  //Proporcional
  pitch_error_prop=angulo_cmp[1]-pitch_referencia;
  //Integral
  pitch_error_int+=pitch_error_prop*dT2/1000;
  //Anti WindUP
  if(pitch_error_int>=180)
  {
    pitch_error_int=180;
  }
  if(pitch_error_int<=0)
  {
    pitch_error_int=0;
  }
  //Derivativo
//  pitch_error_der=(pitch_error_prop-pitch_error_prop_last)/(dT2/1000);
  //Resultado
  pitch_output=Kp*pitch_error_prop+Ki*pitch_error_int; //-Kd*pitch_error_der;
  //Anti WindUP 2
  if(pitch_output>=180)
  {
    pitch_output=180;
  }
  if(pitch_output<=-180)
  {
    pitch_output=-180;
  }
  //Otros valores
  pitch_error_prop_last=pitch_error_prop;
// ==== ==== FIN. Implementación del control PID ==== ==== \\
```

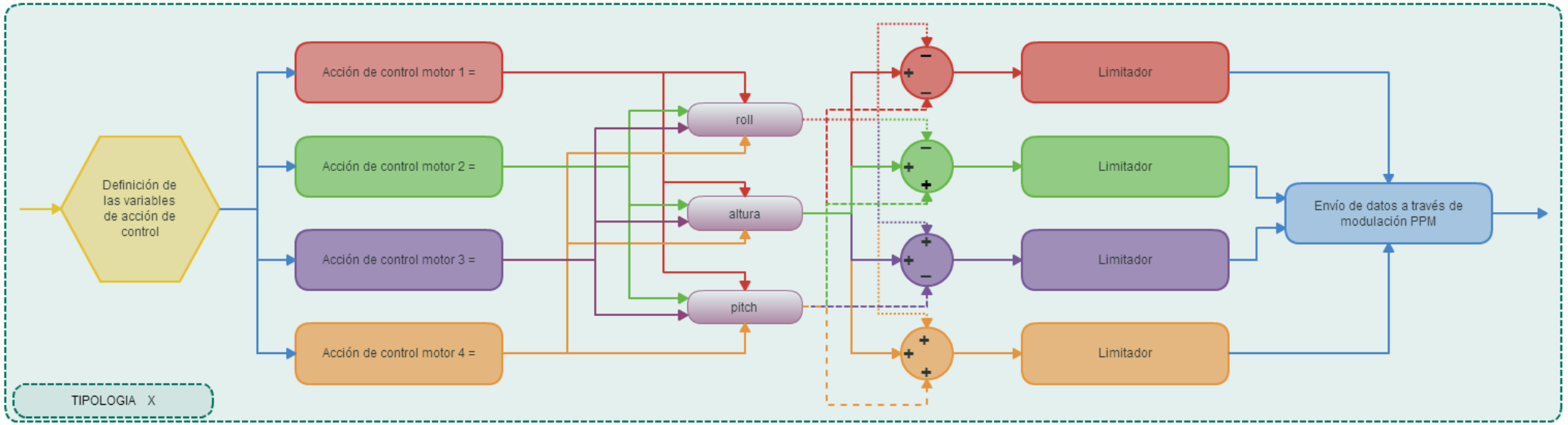
Se propone el control PID del giro respecto al ángulo Z, el esquema de programación es el mismo.

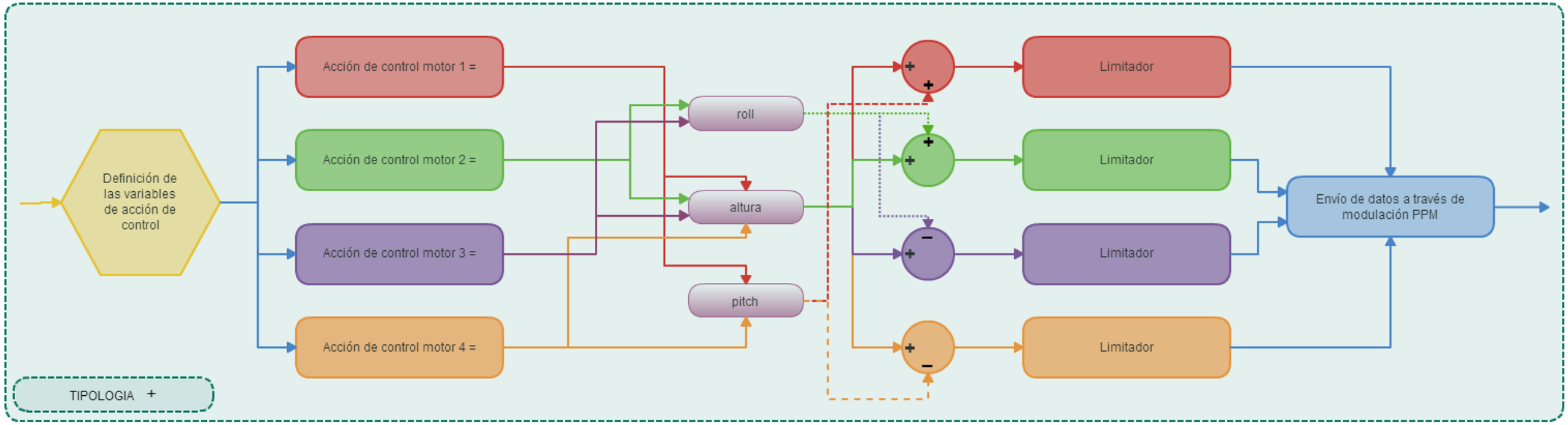
3.8.9 Anexo 7. Calibración de los ESC.



```
//*****************************************************************\\
// **** **** Definicion de variables de la Acción de control **** **** \\
//Altura
float altura=110;
int pulso_min=800;
int pulso_max=2100;
int espera=0;
int accion_M[3];                    // Accion de control del motor M1,M2,M3,M4


Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;

// ==== ==== FIN. Definicion de variables de la Acción de control ==== ==== \\
//=================================================================\\
//############################ SETUP ##############################\\

void setup()
{
// **** **** Ajuste de ESC. Fase 1 **** **** \\
  motor1.attach(3); //Localiza el servo en el pin 6. D3
  motor2.attach(4,pulso_min,pulso_max); //Localiza el servo en el pin 7. D4
  motor3.attach(5,pulso_min,pulso_max); //Localiza el servo en el pin 8. D5
  motor4.attach(6,pulso_min,pulso_max); //Localiza el servo en el pin 9. D6
  delay(100);
  Serial.println("Write Microseconds pulso_max");
  motor1.writeMicroseconds(pulso_max);
  motor2.writeMicroseconds(pulso_max);
  motor3.writeMicroseconds(pulso_max);
  motor4.writeMicroseconds(pulso_max);
  espera=millis();
// ==== ==== FIN. Ajuste de ESC. Fase 1 ==== ==== \\
// **** **** Ajuste de ESC. Fase 2 **** **** \\
  while(millis()-espera<=2500);
  Serial.println("Write Microseconds pulso_min");
  motor1.writeMicroseconds(pulso_min);
  motor2.writeMicroseconds(pulso_min);
  motor3.writeMicroseconds(pulso_min);
  motor4.writeMicroseconds(pulso_min);
// ==== ==== FIN. Ajuste de ESC. Fase 1 ==== ==== \\
//############################ LOOP ###############################\\

void loop()
{
}
```

3.8.10 Anexo 8. Asignación de la acción de control

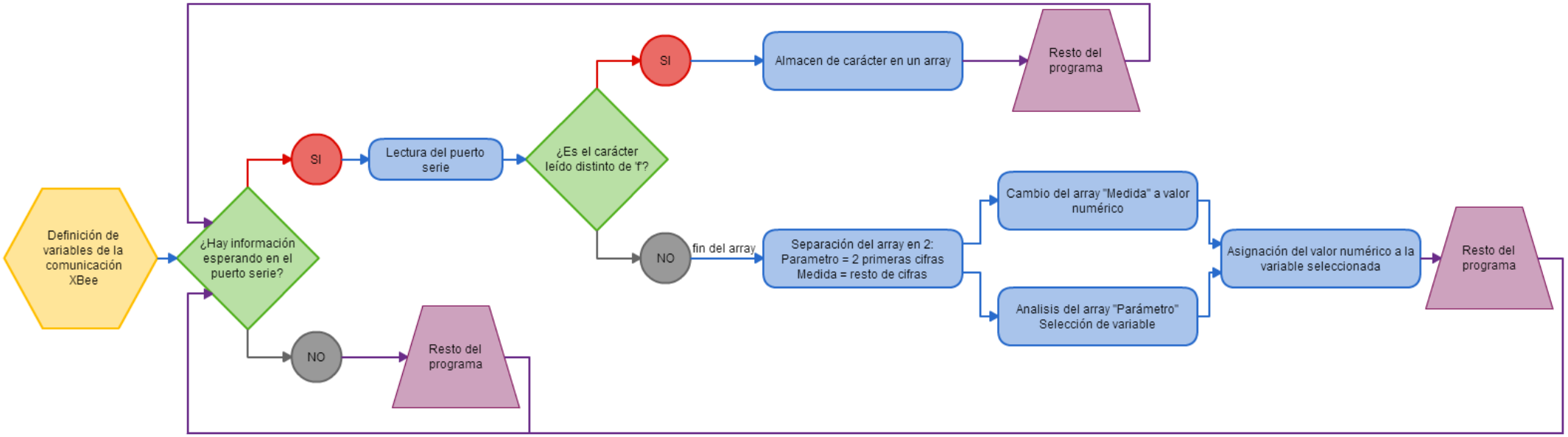




Declaración de variables en el anexo anterior.

```
//############################ SETUP ##############################\\

void setup()
{
//############################ LOOP ##############################\\

void loop()
{
// **** **** Acción de control **** **** \\
//Acción de control para posicion X
  accion_M[0]=altura-pitch_output-roll_output;
  accion_M[1]=altura+pitch_output-roll_output;
  accion_M[2]=altura-pitch_output+roll_output;
  accion_M[3]=altura+pitch_output+roll_output;

  //Acción de control para posicion + (plus)
  accion_M[0]=altura+pitch_output;
  accion_M[1]=altura+roll_output;
  accion_M[2]=altura-roll_output;
  accion_M[3]=altura-pitch_output;
  for(int i=0;i<=4;i+=1) //Anti WindUP 2
  {
    if(accion_M[i]>=180)
    {
      accion_M[i]=180;
    }
    if(accion_M[i]<=0)
    {
      accion_M[i]=0;
    }
  }

  //Activacion de los motores
  motor1.write(accion_M[0]);
  motor2.write(accion_M[1]);
  motor3.write(accion_M[2]);
  motor4.write(accion_M[3]);
// ==== ==== FIN. Acción de control ==== ==== \\
}
```

3.8.11 Anexo 9. Comunicación inalámbrica vía XBee



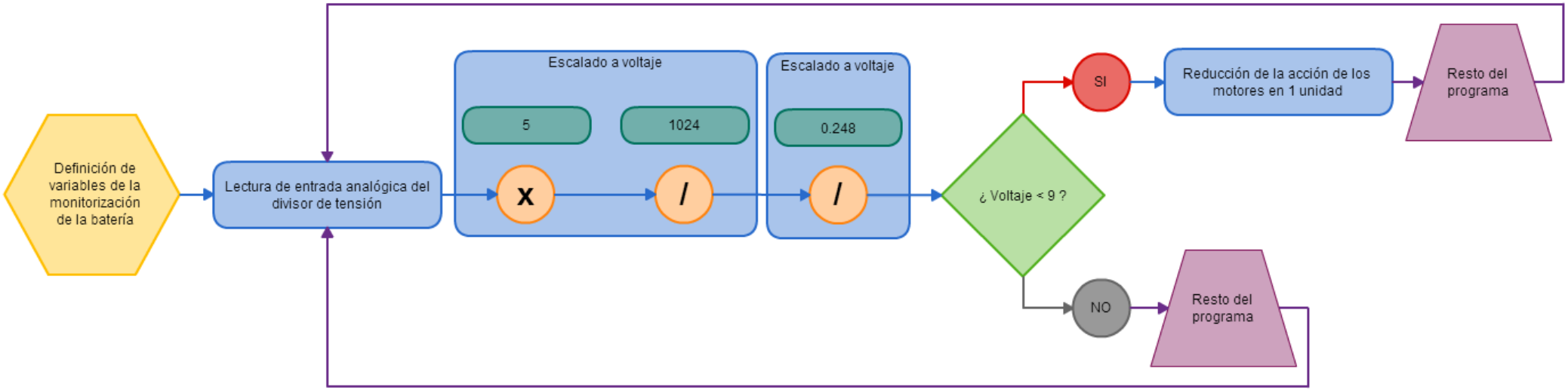
```
205 //************************************************************\\
206 // **** **** Definicion de variables comunicación XBee **** **** \\
207 char ch='0';
208 int posicion=0;
209 int terminado=0;
210 char lectura[8]={0,0,0,0,0,0,0,0};
211 char parametro[2]={0,0};
212 int poslect=0;
213 char medida[6]={0,0,0,0,0,0};
214 float valor;
215 // ==== ==== FIN. Definicion de variables comunicación XBee ==== ==== \\
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {

357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
365 // **** **** Configuracion módulo de comunicación XBee **** **** \\
366   if(Serial.available())
367   {
368     xbee();
369     Serial.print("Parametro : ");
370     Serial.print(parametro[0]);
371     Serial.print(parametro[1]);
372     Serial.print(" : ");
373     Serial.println(valor,6);
374     Serial.print("Alfa : ");
375     Serial.println(alfa);
376     Serial.print("Altura : ");
377     Serial.println(altura);
378     Serial.print("Roll Reference : ");
379     Serial.println(roll_referencia);
380     Serial.print("Pitch Reference : ");
381     Serial.println(pitch_referencia);
382     Serial.print("Yaw Reference : ");
383     Serial.println(yaw_referencia);
384     Serial.print("Ganancia proporcional Kp : ");
385     Serial.println(Kp,6);
386     Serial.print("Ganancia integral Ki : ");
387     Serial.println(Ki,6);
388     Serial.print("Ganancia derivativa Kd : ");
389     Serial.println(Kd,6);
390   }
391 // ==== ==== FIN. Configuracion módulo de comunicación XBee ==== ==== \\
589 // **** **** Escritura de datos **** **** \\
590 // Se utiliza la subrutina print_RAW
591   print_RAW();
592 // ==== ==== FIN. Escritura de datos ==== ==== \\
678 }
```

```
void xbee(void)
{
  ch=Serial.read();
    Serial.println(ch);
    if(terminado==1)
    {
      for(int i=0;i<=8;i++)
      {
        lectura[i]=0;
      }
      for(int i=0;i<=2;i++)
      {
        parametro[i]=0;
      }
      for(int i=0;i<=6;i++)
      {
        medida[i]=0;
      }
      valor=0;
      terminado=0;
    }
    if (ch!='f')
    {
      lectura[poslect]=ch;
      ++poslect;
    }
    else
    {
      lectura[poslect]='\0';
      for (int i=0;i<=1;i++)
      {
        parametro[i]=lectura[i];
      }
      for (int i=0;i<=poslect-2;i++)
      {
        medida[i]=lectura[i+2];
      }
      valor=atof(medida);
      poslect=0;
      terminado=1;
      switch(parametro[1])
      {
        case 'a':    // Alfa. Filtro complementario
          alfa=valor;
          break;
        case 'l':    // Altura
          altura=valor;
          break;
        case 'x':    // Roll Reference
          roll_referencia=valor;
          break;
        case 'y':    // Pitch Reference
          pitch_referencia=valor;
          break;
        case 'z':    // Yaw Reference
          yaw_referencia=valor;
          break;
        case 'p':    // Ganancia proporcional del control PID. Kp
          Kp=valor;
          break;
        case 'i':    // Ganancia integral del control PID. Ki
          Ki=valor;
          break;
        case 'd':    // Ganancia derivativa del control PID. Kd
          Kd=valor;
          break;

```

```
67
68        }
69
70      Serial.print("Lectura : ");
71      Serial.print(lectura[0]);
72      Serial.print(lectura[1]);
73      Serial.print(lectura[2]);
74      Serial.print(lectura[3]);
75      Serial.print(lectura[4]);
76      Serial.print(lectura[5]);
77      Serial.print(lectura[6]);
78      Serial.print(lectura[7]);
79      Serial.println(lectura[8]);
80      Serial.print("Parametro : ");
81      Serial.print(parametro[0]);
82      Serial.println(parametro[1]);
83      Serial.print("Medida : ");
84      Serial.print(medida[0]);
85      Serial.print(medida[1]);
86      Serial.print(medida[2]);
87      Serial.print(medida[3]);
88      Serial.print(medida[4]);
89      Serial.print(medida[5]);
90      Serial.println(medida[6]);
91      Serial.print("Valor : ");
92      Serial.println(valor,6);
93      }
94      Serial.print("Poslect : ");
95      Serial.println(poslect);
96
97      Serial.println("Siguiente caracter ");
98      delay(50);
99  }
1   void print_RAW(void)
2   {
3     // Print the raw acceleration values using the ";" separator
4     Serial.println("Readings ");
5     Serial.print("Accelerometer :");
6     Serial.print(readings[0], DEC);
7     Serial.print(F(";"));
8     Serial.print(readings[1], DEC);
9     Serial.print(F(";"));
10    Serial.print(readings[2], DEC);
11    Serial.println(F(";"));
12    // Print the raw gyro values.
13    Serial.print("Gyroscope :");
14    Serial.print(readings[3], DEC);
15    Serial.print(F(";"));
16    Serial.print(readings[4], DEC);
17    Serial.print(F(";"));
18    Serial.print(readings[5], DEC);
19    Serial.print(F(";"));
20    Serial.println(F(""));
21
22    Serial.print("Ang_acc.xyz : ");
23    Serial.print(angulo_acc[0]);
24    Serial.print(";");
25    Serial.print(angulo_acc[1]);
26    Serial.print(";");
27    Serial.print(angulo_acc[2]);
28    Serial.println(";");
29    Serial.print("Ang_gyro.xyz : ");
30    Serial.print(angulo_gyr[0]);
31    Serial.print(";");
32    Serial.print(angulo_gyr[1]);
33    Serial.print(";");
34    Serial.print(angulo_gyr[2]);
```

```
  Serial.println(";");
  Serial.print("Ang_comp.xyz : ");
  Serial.print(angulo_cmp[0]);
  Serial.print(";");
  Serial.print(angulo_cmp[1]);
  Serial.print(";");
  Serial.print(angulo_cmp[2]);
  Serial.println(";");
  Serial.print("Accion M 1,2,3,4: ");
  Serial.print(";");
  Serial.print(accion_M[0]);
  Serial.print(";");
  Serial.print(accion_M[1]);
  Serial.print(";");
  Serial.print(accion_M[2]);
  Serial.print(";");
  Serial.print(accion_M[3]);
  Serial.println(";");
  Serial.print("PID output roll,  pitch: ");
  Serial.print(";");
  Serial.print(roll_output);
  Serial.print(";");
  Serial.print(pitch_output);
  Serial.println(";");
  Serial.print("PID pitch ref,prop,der,int,proplast,output: ");
  Serial.print(";");
  Serial.print(pitch_referencia);
  Serial.print(";");
  Serial.print(pitch_error_prop);
  Serial.print(";");
  Serial.print(pitch_error_der);
  Serial.print(";");
  Serial.print(pitch error int);
  Serial.print(";");
  Serial.print(pitch_error_prop_last);
  Serial.print(";");
  Serial.print(pitch_output);
  Serial.println(";");

  Serial.print("Lectura entrada analogica bateria : ");
  Serial.println(lectura_analog_divisor,DEC);
  Serial.print("Voltaje divisor de tension : ");
  Serial.println(voltaje_divisor,DEC);
  Serial.print("Voltaje bateria : ");
  Serial.println(voltaje_bateria,DEC);
  Serial.println(' ');

}
```

3.8.12 Anexo 10. Monitorización de la carga de la batería



```
219 //*************************************************************\\
220 // **** **** Definición de variables de la medición de la batería **** **** \\
221 float voltaje_divisor=0;
222 float voltaje_bateria=0;
223 float lectura_analog_divisor;
224 // ==== ==== FIN. Definición de variables de la medición de la batería ==== ==== \\
225 //=============================================================\\
244 //############################ SETUP ##############################\\
245
246 void setup()
247 {

357 //############################ LOOP ##############################\\
358
359 void loop()
360 {
571 // **** **** Medida de batería y control de descenso **** **** \\
572  lectura_analog_divisor=analogRead(1);
573  delay(100);
574  voltaje_divisor=lectura_analog_divisor*5/1024; //3V sera la entrada máxima al pin analógico.
575  voltaje_bateria=voltaje_divisor/0.248;         //Cuando se lean 3V significará que hay 12V
576
577  if(voltaje_bateria<=9)
578  {
579    if(altura>-180)
580    {
581      --altura;
582    }
583  }
584  Serial.print("Altura : ");
585  Serial.println(altura);
586 // ==== ==== FIN. Medida de batería y control de descenso ==== ==== \\
678 }
```

3.9 Referencias.

3.9.1 Tabla de Ilustraciones.

3.9.2 Referencias bibliográficas

[1] M. G. Picatoste, «Estados Unidos va a permitir el uso comercial de drones, aviones robotizados restringidos hasta ahora a fines militares.,» 2012. [En línea]. Available: http://www.abc.es/20120328/internacional/abci-drones-uso-comercial-201203272044.html.

[2] Fernando Muñóz, ABC Tecnologia, «Los Usos Más Increibles de los <Drones>,» 2014. [En línea]. Available: http://www.abc.es/tecnologia/informatica-hardware/20130714/abci-usos-diferente-drones-201307121935_1.html.

[3] S. S. A. Robinson, «FAA authorizes Predators to seek survivors,» U.S. Air Force, 2006.

[4] J. Velasco, «Virus infecta la flota de vehículos aéreos no tripulados de EE.UU.,» 2011. [En línea]. Available: http://hipertextual.com/archivo/2011/10/virus-uav-estados-unidos/.

[5] T. N. Y. T. Pir Zubair Shah, «Pakistan Says U.S. Drone Kills 13,» 2009. [En línea]. Available: http://www.nytimes.com/2009/06/19/world/asia/19pstan.html?_r=0.

[6] M. R. d. Viguri, «Nuestro drone: Un Cuadricóptero,» 2014. [En línea]. Available: www.codrone.viguri.org/es/2014-04-10-17-24-56/noticias-relacionadas/8-nuestro-drone-un-cuadricoptero.

[7] PCB Piezotronics , «Introduction to Piezoelectric Accelerometers,» [En línea]. Available: www.pcb.com/techsupport/tech_accel.aspx.

[8] R. P. Feynman, R. B. Leighton y M. Sands, The Feynman Lectures on Physics, Volume I: Mainly mechanics, radiation and heat., 2005.

[9] D. F. Pozo Espín, *Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio,* Quito, 2010.

[10] R. O. Barrachina, *Apuntes de Mecánica Clásica. Sistemas de coordenadas rotantes,* 2004.

[11] C. Acar y A. Shkel, MEMS Vibratory Gyroscopes, Structural Approaches to Improve Robustness, 2008.

[12] J. Adams, J.-D. Warren y H. Molle, Arduino Robotics, 2011.

[13] F. M. M. I. o. T. Shane Colton, The Balance Filter. A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform., 2007.

[14] K. Ogata, Ingeniería de control moderna, Pearson Educación, 2003.

[15] C. R. G. Cecilio Angulo Bahón, Tecnología de sistemas de control, Universitat Politècnica de Catalunya, 2004.

[16] A. Visioli, Practical PID Control, Springer Science & Business Media, 2006.

[17] D. Norris, Build Your Own Quadcopter. Power Up Your Designs with the Parallax Elev-8, 2014.

[18] Modeltronic Motor S.L., «Tipos de baterías.,» [En línea]. Available: http://www.modeltronic.es/baterias-cargadores-lipo-c-58.html.

[19] Atmel, «ATmega48PA ATmega88PA ATmega168PA ATmega328P,» 2009.

[20] Digi International Inc. , «XBee/XBee-PRO ZigBee RF Modules User Guide,» 2015.

[21] Digi International Inc., «Xbee Module Datasheet,» 2015.

[22] J. Boxall, «Tutorial: Arduino and the I2C bus,» 2010. [En línea]. Available: http://tronixstuff.com/2010/10/20/tutorial-arduino-and-the-i2c-bus/.

[23] Arduino , «Wire Library,» [En línea]. Available: http://www.arduino.cc/en/Reference/Wire.

[24] Arduino, «Arduino Nano,» 2014. [En línea]. Available: http://www.arduino.cc/en/Main/ArduinoBoardNano.

[25] Arduino, «ATmega168/328 - Arduino Pin Mapping,» [En línea]. Available: http://www.arduino.cc/en/Hacking/PinMapping168.

[26] Arduino, «Wire.begin(),» [En línea]. Available: http://www.arduino.cc/en/Reference/WireBegin.

[27] Martin Hebel. Parallax, Inc., «Parallax Data Acquisition for Excel. PLX-DAQ.,» 2007. [En línea]. Available: https://www.parallax.com/downloads/plx-daq.

[28] SFUptownMaker, «Sparkfun,» 2012. [En línea]. Available: https://learn.sparkfun.com/tutorials/pcb-basics.

[29] Ministerio de Educacion, «recursos.cnice.mec.es,» 2006. [En línea]. Available: http://recursos.cnice.mec.es/latingriego/Palladium/cclasica/esc432ca6.php.

[30] Real Academia de Ingeniería, «Raing.es,» [En línea]. Available: http://diccionario.raing.es/es/lema/aeronave-no-pilotada.

[31] Nosolosig, «De dónde procede el término DRON,» 2014. [En línea]. Available: http://www.nosolosig.com/articulos/270-de-donde-procede-el-termino-dron.

[32] D. F. P. Espín, «Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetro y giroscopio,» Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional de Quito, Ecuador, 2010.

[33] J. G. de la Cuesta, Aviation Terminology/Terminología Areonáutica. Inglés-Español/Español-Inglés., 2003.

[34] RC ModelsWiz, The Magic Of Speed. RC Guides., «Electronic Speed Controllers (ESC),» 2008. [En línea]. Available: http://www.rcmodelswiz.co.uk/electronic-speed-controllers-esc/.

[35] Starlino, «A Guide To Using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.,» 2009. [En línea]. Available: http://www.starlino.com/imu_guide.html.