EVALUACIÓN DE LA SOLUCIÓN ARDUPILOT MEGA 2.5 PARA LA IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL EN AEROPLANOS A ESCALA NO TRIPULADOS (UAV).

CESAR JAVIER SEPÚLVEDA PEÑA

LEONARDO MIGUEL JAIMES SÁNCHEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2013

EVALUACIÓN DE LA SOLUCIÓN ARDUPILOT MEGA 2.5 PARA LA IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL EN AEROPLANOS A ESCALA NO TRIPULADOS (UAV).

CESAR JAVIER SEPÚLVEDA PEÑA

LEONARDO MIGUEL JAIMES SÁNCHEZ

Trabajo de Grado para optar al título de
Ingeniero Electrónico

Director

M.Sc ALFREDO RAFAEL ACEVEDO PICÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA

2013

A Dios por permitirme superar esta etapa de mi vida, y por darme salud y fuerza a lo largo de ella.

A mi padre JOSÉ JOAQUÍN SEPÚLVEDA, por su apoyo incondicional, consejos, y el gran amor que me ha demostrado durante toda mi vida.

A mi madre MARIA ANA PEÑA, que desde el cielo me ha brindado la fuerza necesaria para afrontar los retos y perseverancia para alcanzarlos.

A mis Hermanos: Luz, Edgar, Enrique, Nelson y José, por su apoyo y sus palabras cargadas de buena energía las cuales hicieron las cosas más fáciles.

Finalmente a mis amigos: Yohan, Tatiana, Leonardo, Antonio, Mayerli, Angela, Angelica, Yeny, Ferney, y ha mi novia Evelyn Julitza y sus padres Julio y Ligia, por su incondicionalidad brindada a lo largo de esta etapa.

Al director de proyecto de grado, pues sin su apoyo ni los recursos que nos brindó hubiese sido posible culminar este trabajo.

Cesar

En primera instancia al Dios todo poderoso de Abraham, Isaac y Jacob, quien en su infinita misericordia me brindó de su respaldo y veo su promesa cumplida al permitirme culminar con este ciclo de preparación en mi vida.

A mi madre DENIS SÁNCHEZ CARDOZO, quien con su apoyo incondicional me brindó de toda la ayuda necesaria para seguir adelante y nunca desfallecer, sin importar los problemas presentados.

A mi Hermano: Iván quien siempre ha estado ahí acompañándome en todo proceso que llevo, apoyándome en su conocimiento también.

Al director de proyecto de grado, pues sin su apoyo ni los recursos que nos brindó hubiese sido posible culminar este trabajo.

Finalmente pero no menos importantes, a aquellas personas que de una u otra forma estuvieron conmigo en todo este proceso, y a aquellos primero me apoyaron, Samir, Cristian, Nathaly, Clara, y los que conocí en esta ciudad, Juan Antonio, Cesar, Evelyn, y en especial, Laura Ojeda, pues todos ellos me aceptaron sin cuestionarme.

Leonardo

TABLA DE CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: EVALUACIÓN DE LA SOLUCIÓN ARDUPILOT MEGA 2.5, PARA LA IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL EN AEROPLANOS A ESCALA NO TRIPULADOS (UAV)*

AUTORES: SEPULVEDA PEÑA, Cesar Javier
JAIMES SANCHEZ, Leonardo Miguel**

PALABRAS CLAVES: UAV, Ardupilot, DCM, Piloto automático, Ángulos de Navegación, MEMS, Matrices de rotación.

Descripción: En el presente documento se desea evaluar la funcionalidad de la tarjeta Ardupilot Mega 2.5 como piloto automático en modelos de aeronaves no tripuladas, siguiendo rutas predefinidas sobre el mapa establecido. Para ello fue necesario consultar el manual de operación de la misma, su compra, el estudio del software base de configuración y la obtención de un aeroplano a escala con capacidades de maniobrabilidad y vuelo manual. Las pruebas fueron hechas en campo abierto, utilizando el software proporcionado por la casa fabricante para asignar la ruta automática con variación de altura que debía seguir el aeromodelo. Considerando los trabajos futuros del proyecto general del cual hace parte el presente, financiado por la VIE-UIS y desarrollado por el grupo CEMOS de la UIS, se desarrollaron e implementaron algoritmos para el cálculo de los ángulos de navegación, utilizando los sensores inerciales de la tarjeta Ardupilot estudiada, tales como el magnetómetro, el giróscopo y el acelerómetro. Para este objeto, fue necesario establecer rutinas de inicialización y calibración de los sensores, así como el desarrollo de un algoritmo general que permitiera obtener una matriz de cosenos directores de la cual posteriormente, se calcularon los ángulos de Euler, verificando el orden de la matriz desarrollada.

* Proyecto de Grado

** Facultad de Ingenierías Físico Mecánicas. Ingeniería Electrónica. Director: M.Sc. Alfredo Rafael Acevedo Picon

ABSTRACT

TITLE: EVALUATION OF THE ARDUPILOT MEGA 2.5 BOARD, TO IMPLEMENT CONTROL ALGORITHMS IN UNMANNED AHERIAL VEHICLE (UAV)*

AUTHORS: SEPULVEDA PEÑA, Cesar Javier
JAIMES SANCHEZ, Leonardo Miguel**

KEYWORDS: UAV, Ardupilot, DCM, Autopilot, Euler angles, MEMS, Rotation matrix

Description: In this document is desired to evaluate the functionality of the board Ardupilot Mega 2.5 as an auto pilot in unmanned aero plane models following predefined paths over the established map. In order to accomplish such labor it was necessary to read the board's operation manual, the acquisition of the board itself, the acquaintance about using the base configuration software and the obtainment of a model aircraft capable of maneuverability and manual flight. The tests were made in open field, using the builder's software to assign the automatic route varying the altitude followed by the aircraft. Considering future projects covered by the general project funded by VIE-UIS and developed by the research group CEMOS-UIS, were developed and implemented algorithms to calculate the Euler angles, using the inertial sensors presented on the Ardupilot board, such as the magnetometer, accelerometer and gyroscope. To this purpose, it was necessary to establish some sensors' initialization and calibrations routines, as well as the development of a general algorithm which led to obtain the direction cosine matrix from which the Euler angles were calculated, verifying the order of the matrix developed.

* Graduation project

** Physics-Mechanical Faculty. Electronic Engineering. Director: M.Sc. Alfredo Rafael Acevedo Picón

INTRODUCCIÓN

Los UAV (Unmanned Aerial Vehicle por sus siglas en inglés) o “Vehículo Aéreo no Tripulado”, son modelos de aeronaves a escala utilizados ampliamente en diferentes sectores, especialmente para reconocimiento del terreno [1]. La implementación de algoritmos de control en estos dispositivos requiere de un estudio avanzado en las estrategias empleadas, métodos óptimos, sistemas MIMO, control digital, acciones que requieren en un principio el conocimiento íntegro del hardware necesitado para ser llevado a cabo. En algunas ocasiones la documentación al respecto no se encuentra estructurada dentro de un contexto matemático, sino de experiencias que a priori se basan en la experimentación, en conjunto con el método de prueba y error, modelos que no incluyen la fundamentación teórica del sistema, por ende, permite que el desarrollo de nuevas aplicaciones inicien invirtiendo tiempo innecesario en la evaluación de sistemas físicos, hardware y procesadores que eventualmente, limitan el mismo en forma física, por lo que una actualización posterior se haría necesaria.

Dentro de lo propuesto por el presente proyecto, se encuentra la caracterización de la tarjeta de desarrollo para modelos aeronáuticos Ardupilot, tarjeta open source, bajo la plataforma de desarrollo del Arduino, evaluando en forma cualitativa el empleo de la misma, con algoritmos ya propuestos por la comunidad existente alrededor de la tarjeta, determinando los requerimientos mínimos que en esta, u otra tarjeta de prestaciones similares o superiores debe poseer, para luego indicar los posibles pasos a seguir en el caso de implementar un algoritmo personalizado con técnicas de control avanzadas, optimizadas, en el campo de vuelos autónomos para modelos de avión a escala.

En el presente documento se describen los elementos utilizados y el desarrollo de las actividades que se realizaron a lo largo del proyecto. En la primera sección se tratará sobre la formulación y justificación del problema, seguido del sistema de

desarrollo, en donde se caracteriza la tarjeta Ardupilot Mega 2.5 realizando algunas pruebas para verificar su funcionamiento.

Los ángulos de navegación (ángulos de Euler) se pueden obtener mediante la DCM, esta matriz se encuentra descrita en la sección 3, utilizándose en la siguiente sección para realizar rutinas de lecturas de datos desde los sensores, y finalmente se tienen las conclusiones, observaciones, trabajo futuro, recomendaciones y anexos.

1. DESCRIPCION DEL PROYECTO

1.1 FORMULACIÓN DEL PROBLEMA

El problema objeto específico del proyecto es la evaluación cualitativa de la tarjeta de desarrollo Ardupilot, caracterizando sus prestaciones a nivel de hardware y software, para la posterior indicación en la viabilidad de su uso como tarjeta de control de vuelo autónomo presente en modelos aeronáuticos no tripulados, con estrategias de control moderno que proporcionen las acciones correctivas necesarias para el seguimiento fiel de la trayectoria, bien sea rectilínea, o no uniforme con obstáculos.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

- Determinar la potencialidad de la tarjeta digital Ardupilot Mega 2.5 como sistema de seguimiento de trayectorias de un UAV.

1.2.2 OBJETIVOS ESPECIFICOS

- Evaluar de forma cualitativa el funcionamiento del sistema Ardupilot Mega 2.5, como herramienta para implementar algoritmos de control moderno digital en modelos de aeroplanos no tripulados para vuelos autónomos.
- Especificar rutinas, procedimientos y funciones que serán necesarias para el desarrollo de futuros códigos que implementen estrategias de control moderno.

1.3 JUSTIFICACIÓN

La humanidad siempre se encuentra al pendiente de los recursos que se puedan utilizar para minimizar situaciones en donde se coloquen en peligro vidas humanas, bien sea por desastres naturales, seguridad de un lugar, reconocimiento de espacios o vigilancia de elementos de gran valor comercial, entre otros.

En determinados casos la presencia de un ser humano es imposible, ya sea por el difícil acceso al terreno o por las condiciones inseguras del lugar, en estos casos el UAV, juega un papel primordial, puesto que solo necesita una condición inicial y un algoritmo con una estrategia de control que conlleve al éxito de la misión, esto con el fin de proteger las vidas humanas.

La base de todos los proyectos es el conocer los elementos con los que se realiza su ejecución. Con este trabajo se pretende obtener la información necesaria para conocer la plataforma de desarrollo con la que se efectuará la prueba de algoritmos ya existentes, y a su vez conocer la potencialidad y flexibilidad (en forma cualitativa) para implementar algoritmos de otros autores en dicha plataforma por medio de la tarjeta digital Ardupilot Mega 2.5.

2. SISTEMA DE DESARROLLO

Las tarjetas digitales que se utilizan para el control de navegación de UAV en su mayoría pueden hacer funcionar el sistema de forma autónoma, es decir, que partiendo de una condición inicial, estos pueden realizar una misión determinada sin la necesidad de un operador que la controle en tierra mediante radiocontrol. La tarjeta Ardupilot es de la familia de los dispositivos que funcionan como pilotos automáticos, además poseen un software tipo código abierto, al igual que su hardware.

2.1 ARDUPILOT MEGA 2.5

La tarjeta digital Ardupilot versión Mega 2.5, es un piloto automático con un software de código abierto en capacidad de controlar un UAV en forma autónoma, la cual puede manejar puntos de referencia en 3 dimensiones, integrando los factores de estabilidad y navegación, eliminando de esta manera la necesidad de un tercer co-piloto. Entre los sensores de la tarjeta principal del Ardupilot Mega 2.5 se encuentran, un MPU-6000 de 6 ejes, el cual en su interior contiene un acelerómetro de 3 ejes, y un giroscopio de 3 ejes [2]; un compás digital que encuentra la desviación del punto móvil en grados con respecto al norte magnético [3]; un sensor de presión barométrica [4] además de estos sensores posee una memoria interna tipo flash para el registro de datos, un microcontrolador Atmel, entre otros. Además de estas características, también cuenta con periféricos tales como una antena GPS que ubica el punto móvil con respecto a un punto global (punto de referencia global), un kit de telemetría (transmisor-receptor) y diferentes interfaces entre las cuales pueden ser conectados dispositivos I^2C [5], existen diferentes autores cuyos desarrollos se basan en el uso exclusivo de esta tecnología. Su desarrollo, en algunos casos empíricos ha logrado generar patrones de rutas autónomas para modelos de UAV, usando incluso, versiones

antiguas y obsoletas de la plataforma del Ardupilot, no a nivel de programación, sino a nivel físico, en el diseño de la tarjeta y la cantidad de sensores y características que posee. Estas rutas seguían pistas pre-definidas a nivel de programación, establecidas por el desarrollador, vigilando cada uno de los sensores presentes en la tarjeta de desarrollo (acelerómetro, giróscopo, GPS), encargándose de direccionar el aeroplano en el sentido correcto, en algunos casos, con aterrizaje automático.

2.1.1 MPU6000 Giróscopo + Acelerómetro + Temperatura

Este circuito integrado esta compuesto de: un acelerómetro, un giróscopo y un sensor de temperatura. Las principales características del MPU6000 son:

- ✓ Alimentación: 3.3 [V]
- ✓ Bits: 16 bits
- ✓ Interfaz de comunicación: SPI
- ✓ $\overline{CS}$: 53 (Pin en el Arduino)

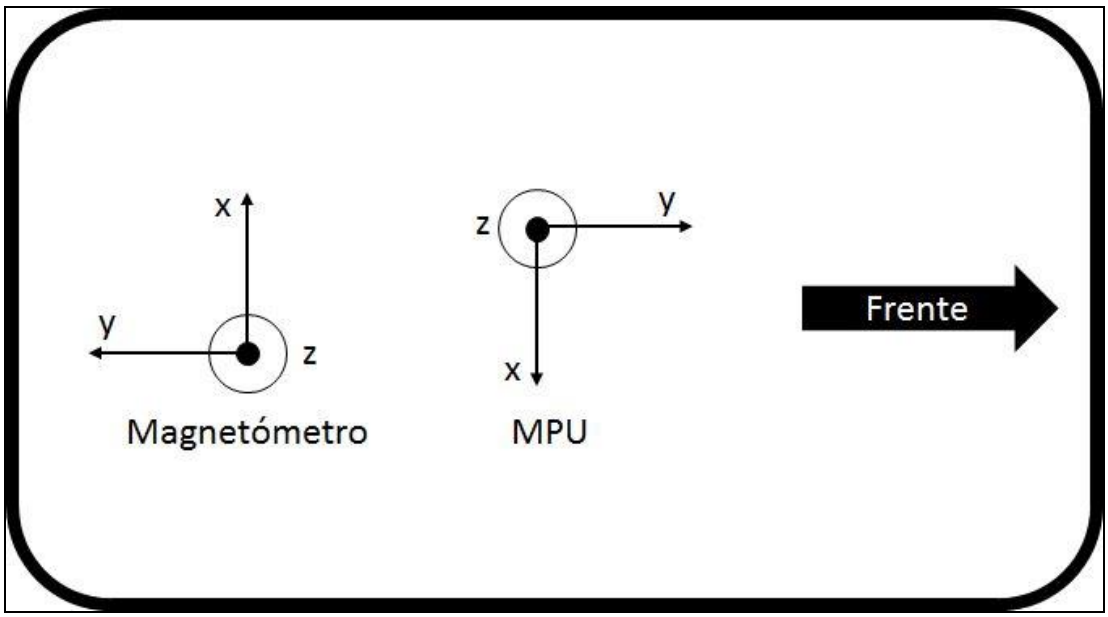

Figura 1 Disposición de ejes del Magnetómetro y del MPU6000 al interior del Ardupilot. Fuente: autores

El MPU6000 tiene un desfase de -90º con respecto al frente de la tarjeta principal, como se observa en la Figura 1. Esto desfase debe ser compensado como se indica en la ecuación 1.

$$\begin{aligned} x &= y \\ y &= -x \end{aligned} \qquad (1)$$

2.1.1.1 Giróscopo

El giróscopo es un sensor basado en la conservación del momento angular, capaz de medir velocidades angulares a la que rota un objeto sobre su(s) eje(s) cartesiano(s), para esto se usa el efecto coriolis. Al adquirir la magnitud de velocidad angular es posible obtener el giro al cual se ve sometido el cuerpo en determinado intervalo de tiempo. El giróscopo de la tarjeta en evaluación [2] puede medir la velocidad angular en sus tres dimensiones y es de tecnología MEMS.

Entre las principales características del giróscopo que utiliza la tarjeta a evaluar se tienen:

Escala completa [°/s]	Resolución [ADC/(°/s)]
±250	131
±500	655
±1000	32.8
±2000	16.4

Tabla 1 Resolución a escala completa del giróscopo. Fuente: [2].

El offset depende de la lectura inicial. De acuerdo a [2], la lectura en estado estable debe ser cero. Se sugiere crear una rutina de inicialización, o bien durante un tiempo inicial (estando en reposo la tarjeta) realizar el promedio de los datos leídos y considerar este valor como offset.

La forma de obtener el valor traducido en la escala real a la que fue conFigurado, en lugar del valor en bytes producto del ADC interno del chip, es con la ecuación (2).

$$G_{eje} = \frac{eje - offset_{eje}}{resolucion} \ [°/s] \qquad (2)$$

Donde el *eje* es el seleccionado, bien sea *X*, *Y* ó *Z*.

2.1.1.2 Acelerómetro.

Cualquier dispositivo que pueda medir aceleraciones lineales sufridas en un eje en particular se denomina acelerómetro, estas medidas se constituyen de una magnitud y una dirección. Un acelerómetro es un transductor que convierte una señal mecánica en una señal eléctrica. Existen varios tipos de acelerómetros.

Los acelerómetros de tecnología MEMS son basados en traspaso de energía térmica, por convención natural, esto significa que realizan una medida de cambios internos en la transferencia de calor causados por aceleraciones sobre moléculas de gas.

El acelerómetro de la tarjeta es de tecnología MEMS [2], mide la aceleración en las 3 dimensiones, es decir en 3 direcciones ortonormales del espacio. Entre las principales características de este sensor se tienen:

Escala completa [g]	Resolución [ADC/g]
±2	16384
±4	8192
±8	4096
±16	2048

Tabla 2 Resolución a escala completa del acelerómetro. Fuente: [2].

El offset depende del valor inicial en estado estable. De acuerdo a [2], sobre una superficie plana debe indicar 0 [g] en el eje *X* y *Y*, y 1 [g] en el eje *Z*. Luego, así

como para el giróscopo, debe considerarse la forma en que será tomado el offset, cumpliendo que en *X* y *Y* los valores serán la muestra tomada, a diferencia del eje Z, el cual, dependiendo de la escala escogida, tendrá distintos valores, es decir, el offset será el *valorMedido* - *resolución*. Esto es, si la escala escogida fue de ±4 [g], el offset es igual al valorPromediado - 8192. El valor promediado, dependiendo del caso, puede ser igual al valor inicial tomado.

La conversión del valor provisto por el chip está dado de acuerdo a (3).

$$A_{eje} = \frac{eje - offset_{eje}}{resolucion} \text{ [g]} \qquad (3)$$

Siendo *eje* el seleccionado, bien sea *X*, *Y* ó *Z*.

2.1.1.3 Temperatura.

Es obtenida a la misma tasa de transferencia que el giróscopo. Al igual que las otras dos magnitudes, es un valor de 16 bits dividido en 2 bytes. La temperatura en grados Celsius está dada por la ecuación (4).

$$T = \frac{dato}{340} + 36.53 \text{ [°C]} \qquad (4)$$

2.1.2 Compás Digital (HMC5883)

Este tipo de sensor interviene en la navegación y estabilización de un cuerpo, pues se necesita un dato fiable para encontrar el giro en el eje Z o ángulo de guiñada. Además es capaz de detectar el campo magnético terrestre, definiendo el norte magnético.

Características:

Alimentación: 3.3 [V]

Bits: 12 bits

Interfaz de comunicación: I^2C

Dirección I^2C: 0x1E

DRDY: 31 (Pin en el Arduino)

Escala completa [Ga]	Resolución [ADC/Ga]
±0,88	1370
±1,3	1090
±1,9	820
±2,5	660
±4,0	440
±4,7	390
±5,6	330
±8,7	230

Tabla 3 Resolución a escala completa del Compas Digital (Magnetómetro) Fuente: [3].

Este sensor mide el campo magnético de la tierra, en unidades de *Gauss*. Es un sensor que mide tal variación en 3 dimensiones (*X, Y y Z*). Se considera que no posee offset. No obstante, el valor obtenido debe ser ajustado, calibrado y compensado para obtener una lectura acorde a la esperada.

En primera instancia, se debe considerar la declinación magnética de la ciudad local donde se encuentra en funcionamiento el sensor. Esta declinación puede ser encontrada en páginas web como la disponible en [6], y convertir la respectiva lectura en un valor decimal para ser tratado por el Ardupilot. Este dato será utilizado más adelante para la obtención del norte de la tarjeta. De no ser disponible dicho valor, deberá crearse una rutina que calcule la inclinación.

El sensor se encuentra dispuesto a un ángulo de +90° respecto a la posición frontal de la tarjeta. Esto debe ser tenido en cuenta si se requiere calcular el ángulo de guiñada.

Un factor de temperatura debe ser calculado con el fin de compensar el dispositivo, si no está disponible la temperatura, puede asumirse una temperatura base de 25°C.

El sensor posee un autoajuste. Se puede recalibrar a partir de los valores leídos en este modo, no obstante, este factor debe tenerse en cuenta en conjunto con la temperatura y el desfase de +90° para obtener la medida calibrada.

Para obtener el dato en unidades de Gauss del sensor, se debe seguir la ecuación (5).

$$M_{eje} = \frac{eje}{resolucion} \qquad (5)$$

Esta fórmula de por si no indica la dirección del norte de la tarjeta respecto a la tierra. Para poder obtener la inclinación real (considerando que la tarjeta se encuentra estable y sobre una superficie horizontal) deben considerarse factores, como la declinación magnética local.

Para lograr esto, habiendo aplicado (5) en primera instancia, se utiliza la función *atan2();* para el eje *Y* y el eje *X* (6).

$$\operatorname{atan2}(eje_y, eje_x); \qquad (6)$$

Esta función no posee gran diferencia de la función atan(*eje*), pues, el cálculo se realiza considerando el cuadrante del vector dado por (eje_x, eje_y), realizando la división respectiva eje_x/eje_y.

No obstante, debido a que el eje del magnetómetro se encuentra modificado, la ecuación debe resultar diferente también. En la Figura 1 se puede apreciar la orientación real del eje del magnetómetro.

Entonces, para poder indicar en forma correcta la disposición de ejes, se debe indicar que:

$$\begin{aligned} x &= -y \\ y &= x \end{aligned} \qquad (7)$$

Por lo tanto, modificando (6) de acuerdo a (7) queda expresado en (8):

$$\operatorname{atan2}(eje_x, -eje_y); \qquad (8)$$

De esta forma, se arregla el ajuste de +90° que posee el sensor. El siguiente paso, es adicionar la declinación magnética local. Para el ejemplo mostrado, se tomará la declinación en la ciudad de Bucaramanga, la cual es -7°37', convertidas en grados resultan en aproximadamente -7.6166°, y estas en radianes resultan en aproximadamente -0.133. La fórmula queda como en (9)

$$\operatorname{atan2}(eje_x, -eje_y) + (-0.133); \qquad (9)$$

Adicional a esto, y dado como sugerencia por [7], se debe realizar un ajuste agregándole 2π al valor leído, verificando luego que el valor final, no sea menor a cero (0), ni mayor a 2π. Si luego del ajuste el valor está por debajo de cero (0), debe agregarse 2π a dicho valor; si por el contrario, es mayor a 2π, debe realizarse una resta de 2π.

$$\operatorname{atan2}(eje_x, -eje_y) + (-0.133) \mp 2\pi; \qquad (10)$$

Posteriormente se convierte este valor en grados y se obtiene el dato de la dirección del avión con respecto al norte de la tierra. Esto es válido si se encuentra en forma horizontal el sensor, para valores diferentes de alabeo y cabeceo deben considerarse otros aspectos no descritos hasta ahora. Esta conversión de los valores se da asumiendo, que los mismos están calibrados.

2.1.2.1 Calibrando el magnetómetro.

Es posible utilizar dos métodos diferentes para realizar la calibración del magnetómetro. Uno de ellos consiste en la compensación mediante la conFiguración de la auto-prueba del sensor, el otro en la compensación por temperatura.

La conFiguración de auto-prueba posee una serie de pasos específicos, que dadas las condiciones de trabajo del dispositivo, generan un retardo en la conFiguración inicial del mismo que es perceptible para una persona. El no realizar esta calibración en tiempos prudentes, puede resultar en la obtención de ángulos erróneos de guiñada en cualquier momento durante el funcionamiento de la tarjeta. Al final de la calibración, se obtendrán unos coeficientes que generan pesos sobre los valores leídos del magnetómetro, con el fin de determinar la desviación que posee la lectura de los datos reales esperados.

La compensación por temperatura genera valores que corrigen la desviación de los datos producto del calentamiento del dispositivo, bien sea por las condiciones de trabajo, o por el calentamiento interno al que se ve sometido por su funcionamiento normal. La idea general consiste en tomar una lectura a una temperatura dada, y luego, por el criterio que tome el autor (bien sea, por cambio de temperatura, intervalo de tiempo de trabajo, u otro que sea considerado pertinente) se recalculan los valores generando coeficientes por los cuales se compensarán las mediciones realizadas en el dispositivo.

En la Figura 2 se puede encontrar el algoritmo general para la calibración del dispositivo en modo de auto-prueba. Los valores esperados por cada uno de los

ejes es de 1580 para el eje X, y de 1500 para el eje Y y Z. Estos valores fueron determinados experimentalmente, puesto que los resultados observados no fueron acordes en la tarjeta respecto a los leídos. Esto es, sin importar la ganancia conFigurada, los valores esperados ya mencionados son exactamente los mismos en la tarjeta de pruebas obtenida.

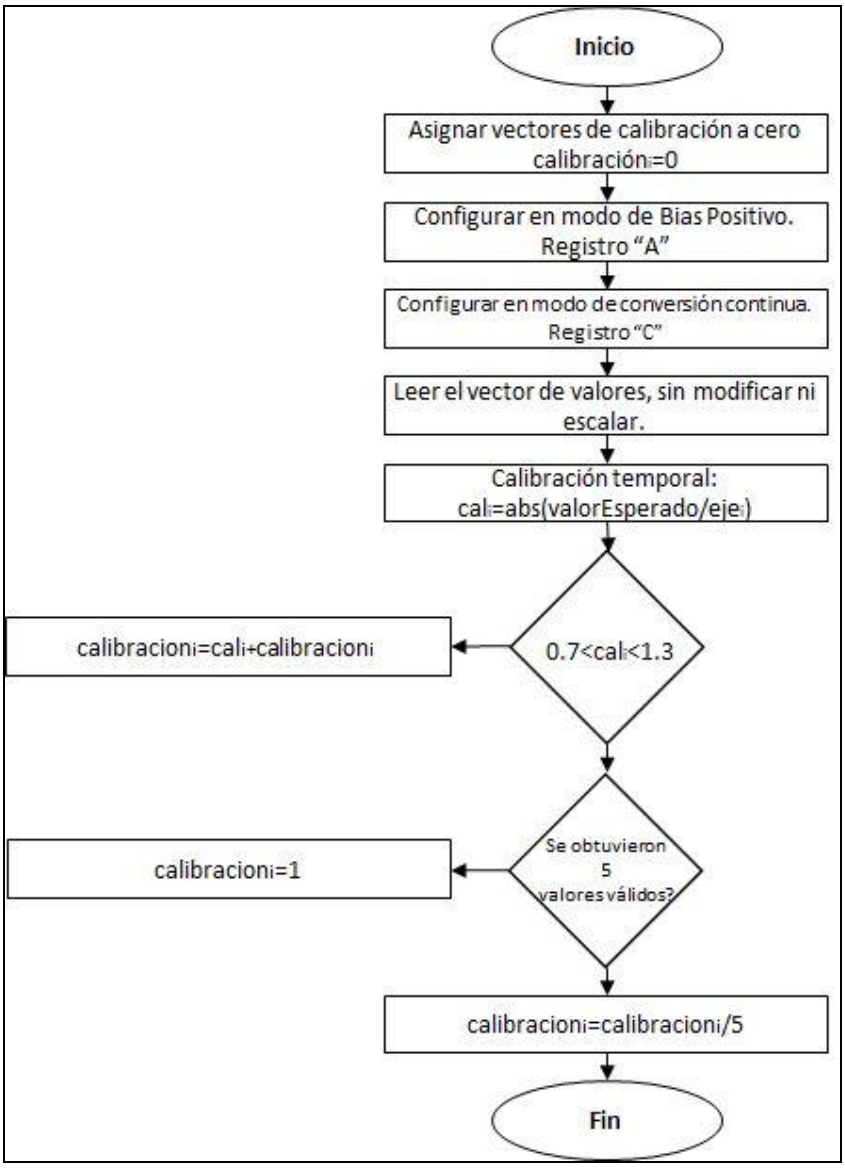

Figura 2. Algoritmo general de calibración del magnetómetro. Fuente [7]. Adaptador por autores

2.1.3 Sistema de Posicionamiento Global (GPS).

El GPS tiene como objetivo principal localizar lo posición de un cuerpo en el espacio partiendo del cálculo de las distancias de al menos 4 satélites. La latitud, altitud y longitud son calculadas en forma continua por el receptor utilizando triangulación. Este sensor mide distancias, que resultan de multiplicar el tiempo de vuelo de las ondas de radio por la velocidad a la que viajan que por lo general se acercan a la velocidad de la luz. Una desventaja que se presenta es que el reloj del receptor (sensor GPS) no posee la precisión del reloj de los satélites pues para medir el tiempo de vuelo es necesario que estos dos relojes se encuentren sincronizados, por tal motivo se hace necesario medir la distancia como mínimo con 4 satélites. Dado que con mayor número de satélites se aumenta la exactitud de la medida.

2.1.4 Altímetro (sensor de presión barométrica)

Un altímetro es un sensor que indica la diferencia de la altura del cuerpo donde se encuentra localizado, y un punto de referencia, en la mayoría de los casos se pretende encontrar dicha distancia con respecto al nivel del mar.

2.2 MODELO DEL AVIÓN DE PRUEBA

Para poder realizar la evaluación cualitativa de la tarjeta Ardupilot Mega 2.5, es necesario realizar una prueba de los distintos modos de vuelo que ofrece la empresa fabricante para su implementación de forma predefinida. Basados en esta premisa, se hace necesario primero obtener un modelo de avión de ala fija en el cual instalar el Ardupilot. Construyendo un avión en material de balso, en forma artesanal, se realiza en primera instancia una prueba de vuelo en forma manual

(sin piloto automático a bordo), estas fueron realizadas utilizando el modelo de avión presentado en la Figura 3. Videos con la prueba realizada pueden ser encontrados en el anexo 5.

Figura 3. Modelo de avión de ala fija utilizado para la prueba aérea. Fuente: autores

Las características que presenta el avión de prueba se pueden observar en la Tabla 4.

Características del avión de prueba	
Peso del avión sin batería [Kg]	1.1
Envergadura [cm]	113
Largo del fuselaje [cm]	70
Máximo amperaje del motor [A]	14
Referencia del motor	C2830-1050
Referencia de la batería	Turnigy nano-tech 2650mah 3S 25~50C Lipo Pack
Referencia de los servomotores	Hobbyking HK15148B

Tabla 4 Características del avión con el que se realizaron las pruebas de vuelo. Fuente autores.

Al interior de este modelo se ubica la electrónica necesaria para evaluar el Ardupilot, entre esas se encuentra el Ardupilot, el receptor del control RC, la unidad de telemetría, el GPS, actuadores de los alerones y los cables para la

conexión de los distintos dispositivos con el piloto automático. La Figura 4 muestra los componentes principales sin la conexión.

Es de considerar que la disposición del Ardupilot debe estar conforme al modelo del avión, no puede ubicarse de lado, en sentido inverso, ni diagonal, sino acoplando los frentes tanto del modelo, como de la tarjeta, de forma que el firmware presente en el Ardupilot pueda determinar la mejor decisión al momento de establecer un rumbo.

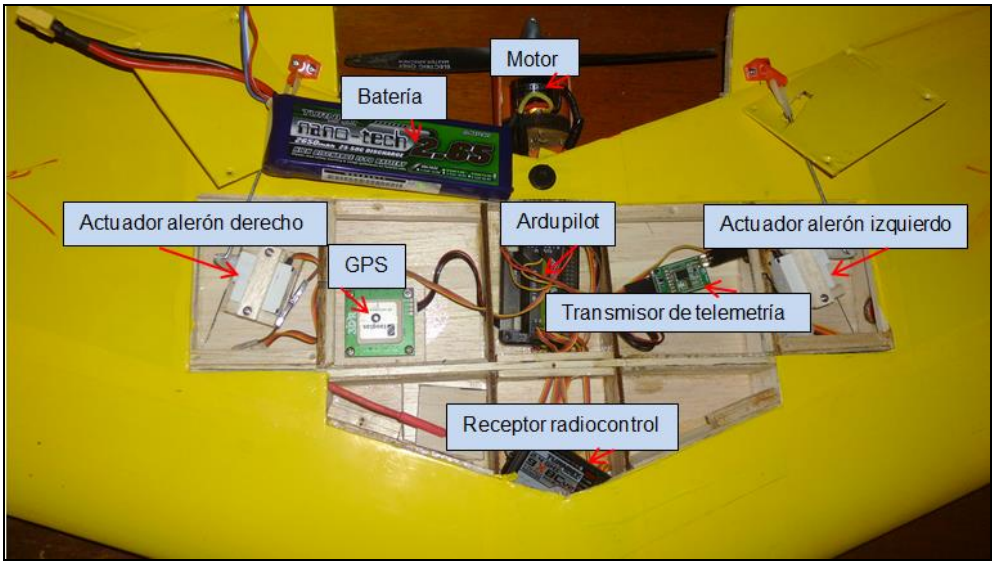

Figura 4. Componentes electrónicos principales al interior del modelo de avión para efectuar la evaluación del Ardupilot como piloto automático. Fuente: autores

En la Figura 5 se muestran todos los componentes que se utilizaron en las pruebas de vuelo, tanto en forma manual (mando de control en tierra) como las tomadas con los diferentes modos de vuelo que posee la tarjeta Ardupilot Mega 2.5. Los componentes son: software Mission Planner Mav, (toma de datos inalámbricos a través del receptor del kit de telemetría y conFiguración de los modos de vuelo) el modelo del avión con la electrónica que posee en su interior y el radiocontrol turnigy TGY 9X.

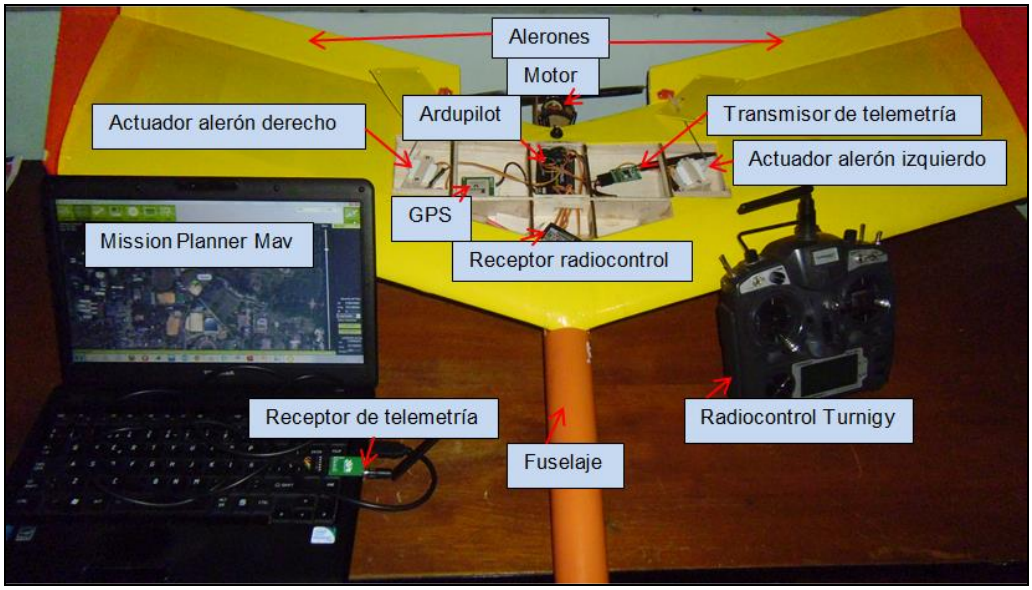

Figura 5. Componentes utilizados durante las pruebas de vuelo. Fuente: autores

2.3 MODOS DE VUELO DE LA TARJETA ARDUPILOT.

Los modos de vuelo provistos por la tarjeta Ardupilot se pueden encontrar en [8]. Entre estos se encuentra el modo manual, que permite que no exista control por parte de la tarjeta sino, es el piloto en tierra quien se encarga de maniobrar el avión. Además de este modo, se posee:

- Stabilize (Estabilizar): Se encarga de estabilizar el avión en forma automática. Puede verse su acción al intentar girar la aeronave y este tratará de devolverlo a la posición por defecto estable.
- RTL (Return to launch - Volver al lanzamiento): Modo en el cual el avión regresa al punto de despegue. Una vez ubicado allí, comienza a sobrevolar en círculos alrededor del punto hasta que el modo manual es activado.
- Auto: La aeronave intentará seguir la trayectoria definida por el usuario a través de la aplicación (Mission Planner Mav). Es posible intentar desestabilizar el avión en este modo para observar cómo intenta regresar a su trayectoria original.

- Loiter (Holgazanear/ocio): Vuela en círculo alrededor del punto donde fue activado el modo. También es posible desestabilizar el avión en forma manual estando dentro de este modo de vuelo.
- FBW-A, B: Modo de aprendizaje de vuelo, permanecerán bloqueados el giro (alabeo) y el levantamiento (cabeceo) del avión, la aceleración es manual. Los límites de giro son establecidos por software.
- Take-off, Land (Modos avanzados): Modo de despegue y aterrizaje respectivamente, son modos avanzados y deben ser programados mediante scripts por el usuario. Depende del modelo del avión.

El procedimiento para hacer la prueba de vuelo consiste de los siguientes pasos:

1. Realizar el vuelo manual del modelo y ajustar los valores de offset del control remoto.
2. Posicionar el interruptor de cambio de modo de vuelo en “Manual”, de esta forma el Ardupilot no tomará ninguna acción de control.
3. Nivelar la tarjeta. Existen dos métodos, uno es por software, mediante el Mission Planner Mav ajustar el nivel en la conFiguración avanzada. El otro, es antes de cada vuelo, dejar el avión lo más nivelado y quieto posible, de esta forma los sensores se ajustan al nivel del avión. Esto se puede verificar pues los 3 leds indicadores del Ardupilot se encuentran parpadeando durante el tiempo de calibración.
4. Verificar que el GPS haya ubicado la posición correcta del avión, tanto en latitud y longitud, como en orientación.
5. Verificar que los controles no se encuentran en reversa. En el modo de estabilizar, verificar que los alerones se mueven en la dirección correcta, de esta forma se garantiza que regresarán a la posición real y evitar accidentes. Esto debe hacerse antes de cada vuelo a ser efectuado. Una medida de seguridad implementada, es que la aceleración no será activada

en tierra a no ser que se encuentre en modo manual, de estabilizar, o en el modo automático al realizar un despegue, solo si ya se encuentra en movimiento en el aire. En el modo de estabilizar es posible mover el avión en tierra sin volar verificando que la superficie intentará corregir el rumbo del avión para estabilizarlo en posición horizontal. El modelo del avión juega un papel fundamental en este control, por ejemplo, el tipo de avión Elevon utilizará sólo dos canales para maniobrar el avión, y esta conFiguración debe establecerse en el Mission Planner para garantizar un control correcto.

6. Para el primer vuelo, luego de estar en el aire, utilizar únicamente los modos de estabilizar o FBW (cualquiera, sea A o sea B), y verificar que el avión se estabilice en el aire luego de maniobrar los controles. Se pueden realizar ajustes en el Mission Planner Mav para alcanzar la estabilidad deseada. Luego de haber verificado los parámetros, es posible pasar al paso 7, para el segundo vuelo.
7. En el segundo vuelo, es posible cambiar al modo RTL, en el cual el punto de despegue debe ser establecido en el Mission Planner Mav, esto con el objeto de probar la navegación, el avión debe retornar al punto indicado y volar en círculos alrededor del mismo.
8. Verificado los pasos 6 y 7, se puede realizar una ruta automática ajustándola desde el Mission Planner Mav, evaluando el desempeño como piloto automático de la tarjeta.

2.4 EVALUACIÓN CUALITATIVA DEL ARDUPILOT MEGA 2.5

En primera medida, para conocer cualitativamente el desempeño del avión desde tierra, se realiza la prueba de telemetría (sin necesidad del avión). De acuerdo a las especificaciones, el módulo posee un alcance de varios kilómetros, sin

especificar si debe poseer línea de vista, por ello, en una cuadra normal de barrio se desplazó el Ardupilot alimentado independientemente en conjunto con el módulo transmisor de telemetría, y se verificó que al perder la línea de vista el receptor no podía obtener los datos del transmisor, esto fue, al dar la vuelta en la esquina, ubicando el receptor en la terraza de un tercer piso. No obstante, logrando que la línea de vista se mantuviera, la potencia alcanzada a una distancia de aproximadamente 80 [m] se mantuvo en 98%, y se puede observar en la Figura 6 la traza del recorrido hecho, la línea morada tenue indica el recorrido realizado con la tarjeta conectada al kit de telemetría. De acuerdo a datos tomados del GPS la localización de la casa del avión se encuentra a 7.135045º latitud norte, -73.119421º longitud oeste, y altura absoluta de 1028 msnm. La demostración de la prueba de telemetría puede ser encontrada en el anexo 5.

Figura 6. Prueba de telemetría vista desde el simulador del Ardupilot (Mission Planner Mav). Fuente: Autores

Siguiendo el procedimiento propuesto, las pruebas tuvieron lugar en la pista seleccionada. Como primera medida, y de acuerdo a como está especificado en la guía de inicio en [8], la conFiguración de los canales para los alerones debe ser

estricta, garantizando control y estabilización, en lugar de provocar una caída en picada que destruya el modelo.

La primera prueba de vuelo consiste entonces en el modo de estabilizar, a modo que bajo las perturbaciones indicadas, el aeroplano sea capaz de estabilizarse, la perturbación seleccionada para probar la funcionalidad de la tarjeta consiste en dirigir el avión en descenso vertical, y con éxito, al soltar el control RC, el piloto automático restableció la maniobra de vuelo, logrando un vuelo suave horizontal.

La siguiente prueba realizada en el vuelo, es el retorno a casa, bajo la cual el avión sin importar su ubicación u orientación, retornaba de inmediato a casa (la coordenada de home establecida en el Mission Planner), lugar donde se encontraba orbitando alrededor hasta el cambio de modo de vuelo.

En los modos automáticos descritos, se observa como el piloto automático incluso realiza control de velocidad, disminuyendo la velocidad de giro del motor. La estabilización del aeroplano era evidente, generando un vuelo suave, sin perturbaciones o cambios bruscos de giro y/o rotación. La maniobrabilidad del piloto respecto al avión fue considerada excelente, navegando mejor de lo que el piloto experimentado en tierra pudo lograr, pues es de resaltar, que debido a lo artesanal de la construcción del avión, presenta perturbaciones en el vuelo que el piloto debe compensar con su experiencia para lograr un vuelo suave, y aun así, el Ardupilot estuvo en la capacidad de controlarlo, concluyendo que sin importar la planta (en este caso, el avión) ni las perturbaciones es capaz de efectuar el control necesario para estabilizar el vuelo.

Luego de lo mencionado, la prueba realizada fue el vuelo por ruta predefinida. Como se muestra en la Figura 7, la ruta generada alrededor del complejo permite evaluar si efectivamente la trayectoria está siendo definida. El cambio de modo de vuelo puede ser establecido “en caliente”, esto es, mientras se encuentra en vuelo

el avión sin necesidad de aterrizar, permitiendo así ahorro de batería y logística para el despegue.

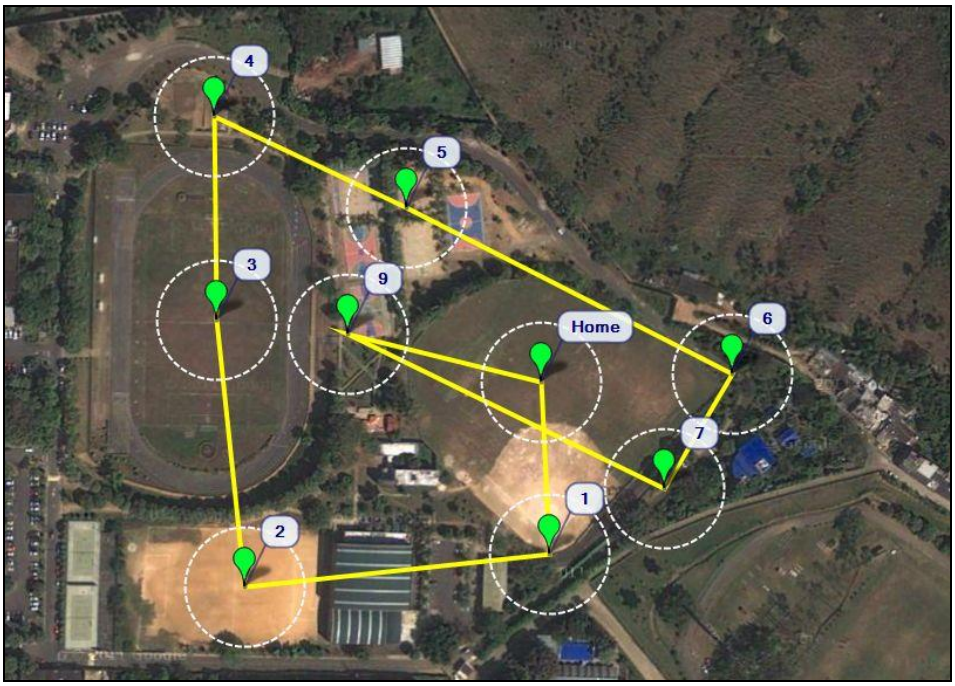

Figura 7. Definición de ruta de sobrevuelo para la prueba del piloto automático del Ardupilot, tomado del software Mission Planner MAV. Fuente: Autores

El aeroplano se encontraba siguiendo la ruta en forma fiel, cuando la conFiguración de repetición no estaba definida, este orbitaba el avión alrededor del punto de casa al terminar el recorrido, habiendo conFigurado la repetición infinita el recorrido se hacía indefinidamente. La telemetría no se perdió durante el recorrido, y el Mission Planner graba automáticamente los datos para ser analizados luego de la prueba de vuelo. Aunque las perturbaciones dadas por el viento eran presentes, el Ardupilot controlaba sin problema alguno la ruta establecida, incluso, activando el modo de vuelo invertido, el Ardupilot fue capaz de estabilizar el aeroplano sin ningún inconveniente, generando una trayectoria suave sin perturbaciones bajo este modo.

Esto genera la conclusión más importante del trabajo. La solución Ardupilot Mega 2.5 posee todas las características a nivel de hardware y software para efectuar control automático de vuelo en un aeroplano a escala no tripulado. Un video de las pruebas de vuelo realizadas se encuentra en el anexo 5.

3. DESARROLLO DE RUTINAS Y PROCEDIMIENTOS NECESARIOS PARA IMPLEMENTACIÓN DE ESTRATEGIAS DE CONTROL

El objetivo de un piloto automático en un aeromodelo es el seguimiento de una ruta predefinida establecida. Para implementarlo es necesario controlar la posición actual de la aeronave. La posición de la misma incluye determinar geográficamente, respecto a una referencia dada, las coordenadas actuales en el espacio, esto es, la altura, la distancia en X y en Y. Teniendo como referencia la tierra, es tomada como distancia en X la longitud y como distancia en Y a la latitud. Es fácil determinar esta coordenada si se posee de un GPS, pero el determinar la altura solo lo es con ayuda de un sensor de presión para compensar el valor obtenido con el GPS, en muchos casos, es suficiente un sensor de presión barométrica, determinando la altura respecto al nivel del mar. Si se desea establecer una ruta, el GPS puede indicar la posición actual en la que se encuentra, pero el piloto debe determinar si se encuentra en la dirección deseada, de lo contrario, deberá corregir su curso y re calcular el error. La dirección deseada es un vector compuesto por diferentes ángulos, estos son conocidos como los ángulos de navegación.

3.1 INTRODUCCIÓN A LAS TÉCNICAS EMPLEADAS PARA LA OBTENCIÓN DE VARIABLES DE CONTROL

Los ángulos de navegación [9] (ángulos de Euler: Ψ, θ y Φ) son utilizados para describir la orientación de un cuerpo rígido en el espacio tridimensional (espacio euclidiano) de un sistema de coordenadas que se encuentran en un marco móvil con respecto a un sistema de coordenadas de referencia que normalmente está fijo. En este documento el cuerpo móvil (sistema de coordenadas móvil) es el UAV el cual puede girar en tres ejes ortogonales ya definidos, como se observa en la Figura 8. Estos ángulos de navegación son algunas de las variables de control que

deben ser tenidas en cuenta al momento de implementar estrategias bien sea de estabilización, bien sea de posición.

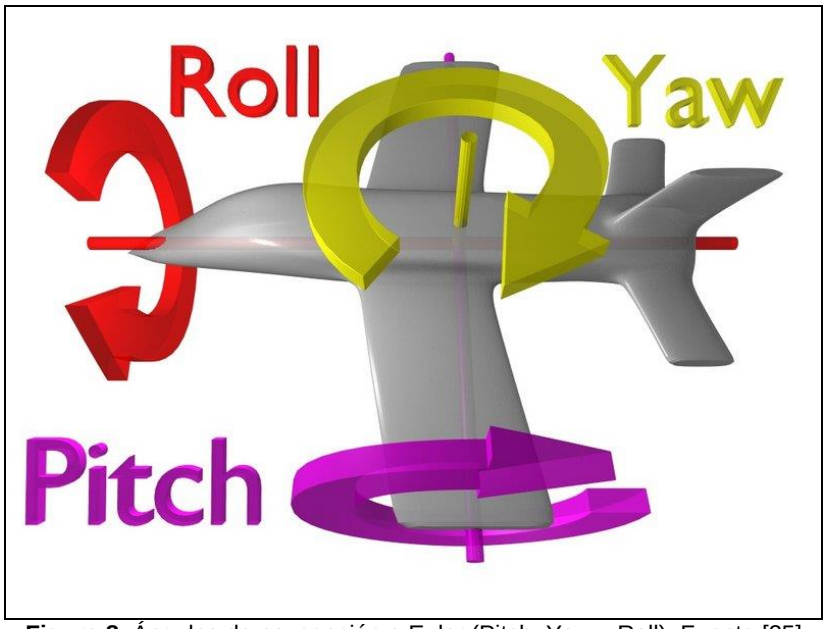

Figura 8. Ángulos de navegación o Euler (Pitch, Yaw y Roll). Fuente [25]

Los ángulos de Euler pueden obtenerse mediante diferentes métodos. Un método existente y mencionado por [17] es el uso de cuaterniones. Los cuaterniones son una extensión de los números reales que se obtienen en forma similar a los números complejos. Cuando son representados como elementos matriciales pueden expresarse en términos de 4 términos (una matriz de 2x2). De acuerdo a [24], los cuaterniones poseen ventajas numéricas como lo es la rapidez de la concatenación de matrices, estabilidad, pocos elementos (pues, son solo 4), la extracción de ángulos y ejes de rotación es simple, entre otras. El método de mayor uso [11], [17] y [19] entre otras razones, por encajar en forma natural al control y la navegación [11], es el uso de las matrices de cosenos directores (DCM por sus siglas en inglés).

3.2 MATRIZ DE COSENOS DIRECTORES (DCM)

Una Matriz de Cosenos Directores (DCM) [9], [10] y [11] es el resultado del cálculo de las proyecciones de un marco de referencia en tierra sobre un marco de referencia móvil (avión), a través de las matrices de rotación, en términos de los ángulos de Euler (dicha proyección se puede observar en la Figura 9). La DCM contiene suficiente información del sistema analizado para expresar la orientación (vectores: Velocidad, Aceleración, Traslación, Dirección) de la aeronave. Como el interés de este documento es obtener los ángulos de navegación usando los sensores de la tarjeta a evaluar, se transforman los datos leídos por los mismos a través de la matriz DCM.

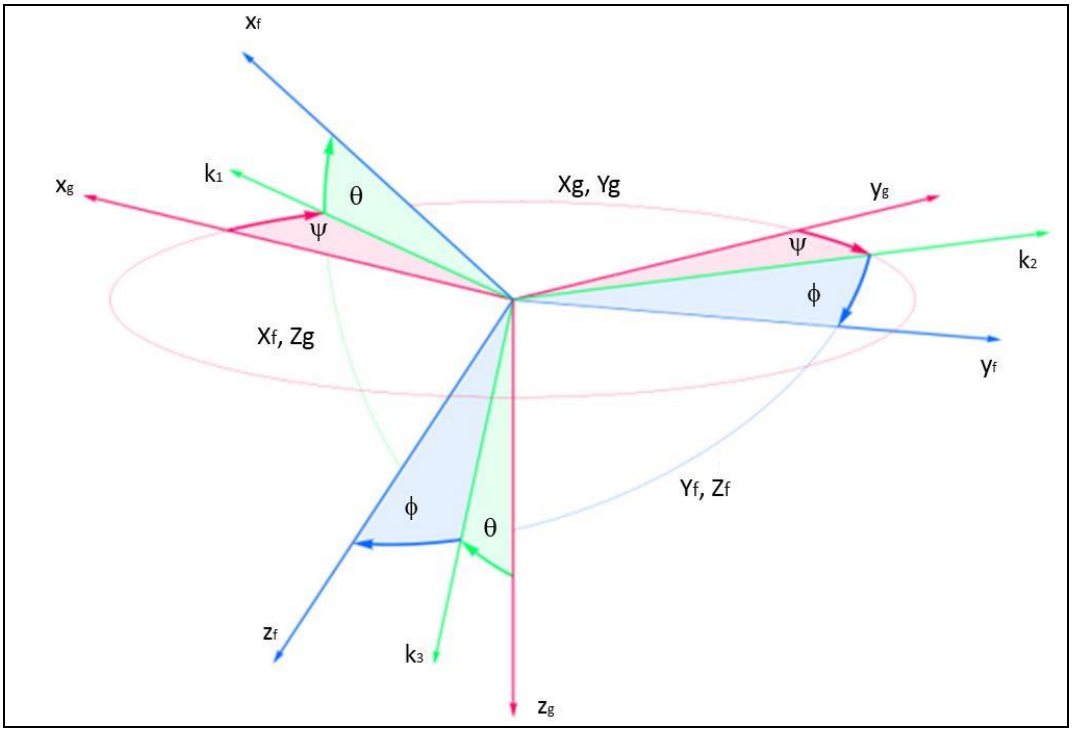

Figura 9. Proyección del marco del avión sobre el marco de referencia en tierra. Fuente [26]

En la Figura 9 se puede observar que tanto el sistema de coordenadas en tierra como las coordenadas que se toman en el marco de referencia del avión tienen el mismo origen.

En la ecuación (11) se muestra el producto de las tres matrices de rotación de acuerdo al orden utilizado (Guiñada, Alabeo y Cabeceo) expresadas en términos de los ángulos de Euler, que corresponde a la matriz de cosenos directores.

$$R = \begin{pmatrix} \cos\theta\cos\Psi & \sin\Phi\sin\theta\cos\Psi - \cos\Phi\sin\Psi & \cos\Phi\sin\theta\cos\Psi + \sin\Phi\sin\Psi \\ \cos\theta\sin\Psi & \sin\Phi\sin\theta\sin\Psi + \cos\Phi\cos\Psi & \cos\Phi\sin\theta\sin\Psi - \sin\Phi\cos\Psi \\ -\sin\theta & \sin\Phi\cos\theta & \cos\Phi\cos\theta \end{pmatrix} \quad (11)$$

De la ecuación (11), se pueden encontrar el valor de los ángulos Euler como se muestra en (12), (13) y (14).

$$\theta = -\sin^{-1}(R_{31}) \quad (12)$$

$$\Psi = \sin^{-1}\left(\frac{R_{21}}{\cos\theta}\right) \quad (13)$$

$$\Phi = \sin^{-1}\left(\frac{R_{32}}{\cos\theta}\right) \quad (14)$$

Donde R es la matriz descrita en la ecuación (11) y los subíndices representan las filas y columnas respectivamente.

Una limitación que presentan las ecuaciones (13) y (14) es cuando el ángulo θ, toma un valor de: $\theta = \left(\frac{\pi}{2}\right) \pm n\pi \ (n = 0,1,2,\dots)$, dado que en ese punto se produce una condición en la que no se pueden calcular los ángulos Euler, por ser una solución indeterminada con infinitas soluciones que producen la misma matriz de rotación [9].

3.3 RUTINAS DE LECTURAS DE LOS SENSORES EN EL ARDUPILOT

Para realizar el cálculo de la DCM en dispositivos digitales, se hace necesaria la lectura de datos de los sensores inerciales presentes en la tarjeta [18] y [19]. Cada sensor presente en el Ardupilot posee una rutina de inicialización y conFiguración de la cual dependerá la escala, la tasa de transferencia, el modo de trabajo, muestreos, filtros digitales, interrupciones externas, entre otras características de los mismos.

La información para la inicialización fue tomada de las librerías disponibles en [12], las librerías disponibles en [13] y de las hojas de datos de los sensores disponibles en [2] y [3].

3.3.1 EL MAGNETÓMETRO

El magnetómetro utiliza el protocolo de comunicación I^2C, los autores del presente texto asumen que el lector ya está familiarizado con este y por lo tanto no se entrará en detalles de cómo debe enviar la información al dispositivo. La secuencia de inicialización consta de los siguientes pasos:

1. Solicitar escritura al dispositivo.
2. Enviar la dirección del registro “A” (0x00 en hexadecimal).
3. Enviar byte de conFiguración del registro (conforme a [3]). Este registro se conFigura el muestreo y el modo de trabajo.
4. Enviar la dirección del registro “B” (0x01 en hexadecimal).
5. Enviar byte de conFiguración del registro (conforme a [3]) para establecer la escala del dispositivo.

6. Enviar la dirección del registro del modo de trabajo (0x02 en hexadecimal).
7. Enviar byte de conFiguración del registro (conforme a [3]), estableciendo el modo de lectura del sensor y la conFiguración del bus I^2C.

Para conocer si el dispositivo se encuentra o no disponible para requerir datos del mismo, el integrado posee físicamente un pin de interrupción por hardware conocido como DDRY, el cual se encontrará activo en caso de existir un nuevo valor en el registro de datos de salida del sensor. Esta característica debe tenerse en cuenta si se requiere que el algoritmo obtenga el dato a una tasa de muestreo lo más corta posible. De no ser utilizado este pin por hardware, deben esperarse al menos 6 [ms] [3] antes de poder obtener un dato fiable. La secuencia para leer los datos consta de:

1. Enviar la dirección del registro inicial de datos (0x03 en hexadecimal).
2. Solicitar 6 bytes de lectura.
3. Obtener los 6 bytes de lectura.
4. Esperar 6 [ms] o a que DDRY se encuentre en estado alto.

De esta forma se garantiza que los datos puedan ser recibidos y procesados.

3.3.2 ACELERÓMETRO, GIRÓSCOPO Y TEMPERATURA.

Este dispositivo trabaja mediante el protocolo SPI. De acuerdo a [2], el protocolo posee una línea dedicada al pin de *select* para establecer comunicación con el dispositivo, en el caso del Ardupilot, existen al menos dos sensores que se encuentran conectados al bus, el otro es el medidor de presión barométrica. Se hace necesario deshabilitar el sensor de presión barométrica para poder trabajar con el MPU6000. La tarjeta trabaja como Arduino, entonces deshabilitar el sensor se puede realizar mediante dos líneas de código (15).

```
pinMode(40, OUTPUT);                    (15)
digitalWrite(40, HIGH);
```

El pin 40 es el $\overline{CS}$ del sensor de presión barométrica [16].

Debido a que el MPU6000 es un sensor con mayores prestaciones, su inicialización es más elaborada comparada con la del magnetómetro. Los pasos para inicializarlo son:

1. Deshabilitar todos los dispositivos del bus SPI.
2. Resetear por software el dispositivo (Registro POWER_MANAGEMENT_1, 107 en decimal), activando el bit 7. Este bit se desactiva automáticamente al terminar el reset.
3. Seleccionar el modo de trabajo del dispositivo y el reloj (Registro POWER_MANAGEMENT_1). De conformidad con [2] se sugiere usar el reloj del giróscopo para brindar un mejor desempeño o usar uno externo.
4. Deshabilitar el bus auxiliar I^2C, de esta forma se habilita el bus SPI.
5. ConFigurar la razón de muestreo (Registro SMPRT_DIV).
6. ConFigurar el filtro pasa-baja digital, puede ser conFigurado con sincronización de tramas (FSYNC) (Registro CONFIG).
7. ConFigurar las escalas del giróscopo y del acelerómetro.
8. (Opcional) Habilitar la interrupción por hardware que indica el dato listo (Registro INT_ENABLE).
9. (Opcional) ConFigurar el “clear” de la interrupción al leer el dato (Registro INT_PIN_CFG).

Para un mayor detalle de las direcciones y uso de los registros dirigirse a [2] o [13].

Para establecer la lectura del dispositivo se debe seguir la siguiente secuencia:

1. Establecer el *msb* del registro leído en 1.
2. Establecer en bajo lógico el pin $\overline{CS}$ del dispositivo.
3. Enviar la dirección del registro con el *msb* en alto.
4. Enviar un byte de ceros, leyendo el valor que devuelva el dispositivo.
5. Poner en alto el pin $\overline{CS}$ del dispositivo.

La secuencia para escribir datos en el dispositivo es:

1. Establecer en bajo el pin $\overline{CS}$ del dispositivo.
2. Enviar la dirección del registro.
3. Enviar el dato a establecer.
4. Establecer en alto el pin $\overline{CS}$ del dispositivo.

El dispositivo utiliza registros de 16 bits de resolución ADC para almacenar las lecturas de los sensores. Cada lectura realizada en el proceso descrito anteriormente entrega datos de 8 bits, luego los datos de los sensores se encuentran divididos en dos registros, uno con el byte alto, y el otro con el byte bajo. Para obtener el valor real medido, deben almacenarse estos dos registros en una variable de 16 bits, que luego puede ser tratada para establecer su significado físico, y no el valor digital equivalente.

3.4 IMPLEMENTACIÓN DEL CÁLCULO DE LA DCM EN LA TARJETA ARDUPILOT

Existen varias fuentes acerca de las cuales se pueden referenciar la construcción de la DCM en aeronaves no tripuladas [9], [10] y [11], no obstante, la matemática detrás de dichas fuentes está alejada de su implementación en un dispositivo digital, logrando de esta forma que no solo se requiera de un conocimiento avanzado en álgebra matricial y física, así como también de cálculo, sino también

de habilidades destacadas para el desarrollo de algoritmos a partir de datos obtenidos en la tarjeta micro controlada que se desea utilizar. No obstante, existe una fuente que se ha encargado de estudiar esta matemática pues, todas llevan por referencia [17], y establece aproximaciones para la implementación del cálculo de la DCM en una tarjeta digital. En [19] se especifica la forma en que los sensores en una tarjeta funcionan, y el método que debe seguirse (dependiendo del chip) para poder transformar los datos entregados por los dispositivos en mediciones físicas. En [18] se centra el estudio de esta sección, a partir de la cual se establecerá el algoritmo general para la construcción de la DCM. Este algoritmo se encuentra descrito en la Figura 10.

La lectura inicial de los sensores proporcionan el vector $\vec{K}$ y el vector $\vec{I}$ obtenidos del acelerómetro y el magnetómetro respectivamente normalizando los valores. El acelerómetro establece una lectura positiva hacia el centro de la tierra, el cenit se encuentra en dirección opuesta y por ello, $\vec{K}$ es el valor inverso leído del acelerómetro.

$$\vec{K} = \frac{-\vec{A}}{\left|\vec{A}\right|} \qquad (16)$$

La dirección del vector $\vec{I}$ se encuentra dirigida al norte magnético de la tierra, pero no siempre es ortogonal al vector $\vec{K}$ y por lo tanto, debe corregirse.

$$\vec{I} = \left(\vec{K} \times \vec{I}\right) \times \vec{K} \qquad (17)$$

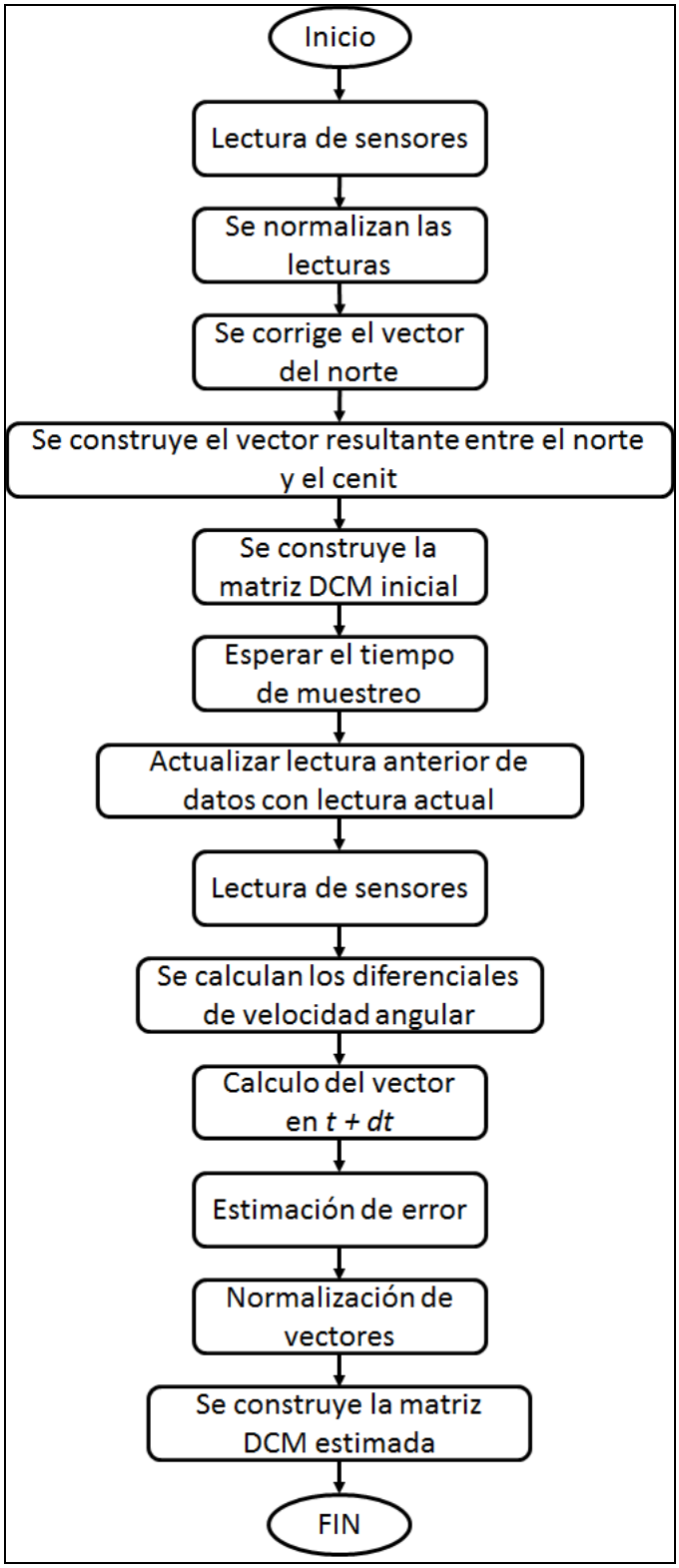

Figura 10. Diagrama de flujo general para la obtención de una matriz de DCM en un dispositivo IMU con 9 grados de libertad. Fuente [18]. Adaptado por los autores.

El vector $\vec{J}$ se construye a partir de los vectores $\vec{I}$ y $\vec{K}$.

$$\vec{J} = \vec{K} \times \vec{I} \qquad (18)$$

La DCM inicial es entonces el arreglo matricial de los vectores $\vec{I}$, $\vec{J}$ y $\vec{K}$.

$$DCM_0 = \begin{bmatrix} \vec{I} \\ \vec{J} \\ \vec{K} \end{bmatrix} \qquad (19)$$

El diferencial de velocidad angular se construyen a partir de la suma ponderada entre la actualización de los vectores $\vec{I}$ y $\vec{K}$ y el diferencial de velocidad angular provisto por los giróscopos.

$$d\vec{\theta}_g = T_s \cdot \vec{W} \qquad (20)$$

$$d\vec{\theta}_m = \vec{I} \times \left[\vec{M} - \vec{I} \right] \qquad (21)$$

$$d\vec{\theta}_A = \vec{K} \times \left[\left(-\vec{A} \right) - \vec{K} \right] \qquad (22)$$

$$d\vec{\theta} = \frac{s_a d\vec{\theta}_A + s_g d\vec{\theta}_g + s_m d\vec{\theta}_m}{s_a + s_g + s_m} \qquad (23)$$

Siendo $\vec{W}$ la lectura normalizada y en radianes sobre segundos ([rad/s]) del giróscopo, $\vec{M}$ el vector que contiene la lectura normalizada obtenida del magnetómetro, s_a s_g y s_m pesos asignados a los diferenciales de velocidad angular del acelerómetro, giróscopo y magnetómetro respectivamente.

Los vectores estimados en el tiempo *t* + *dt* se obtienen del diferencial de velocidad angular.

$$\vec{I}_{est} = \vec{I} + \left[d\vec{\theta} \times \vec{I} \right] \qquad (24)$$

$$\vec{J}_{est} = \vec{J} + \left[d\vec{\theta} \times \vec{J} \right] \qquad (25)$$

Estos vectores $\vec{I}$ y $\vec{J}$ deben ser normalizados. La estimación del error se realiza asumiendo que ambos se encuentran igualmente erróneos [18].

$$err = \frac{\left[\vec{I}_{est} \bullet \vec{J}_{est} \right]}{2} \qquad (26)$$

Los vectores entonces ortogonales son calculados, luego deben ser normalizados.

$$\vec{I} = \vec{I}_{est} - err \cdot \vec{J}_{est} \qquad (27)$$

$$\vec{J} = \vec{J}_{est} - err \cdot \vec{I}_{est} \qquad (28)$$

$$\vec{K} = \vec{I} \times \vec{J} \qquad (29)$$

La DCM estimada se construye igual que en (19).

El tiempo de muestreo T_s debe ser el más corto posible, esto debido a que ante variaciones menores ($dt \rightarrow 0$) la velocidad lineal es perpendicular al vector actual en la posición del cuerpo, permitiendo de esta forma que la variación del ángulo pueda tender a cero, y por lo tanto, sin importar el orden de la rotación, se pueda estimar con el mínimo error la posición actual del vector a partir de la velocidad angular [18], [19] y [20]. De [2] y [3] se puede deducir que aquel sensor con mayor latencia a la hora de entregar datos nuevos es el magnetómetro, implicando que el T_s mínimo es de 6 [ms].

Este algoritmo fue implementado en el Ardupilot. Considerando la memoria del mismo y que los algoritmos no se encuentran optimizados, el conjunto de estas operaciones, incluyendo un debug por puerto serial ocupa menos del 10% de la memoria Flash que posee el dispositivo para almacenar los programas. La evidencia de esto se encuentra en la Figura 11.

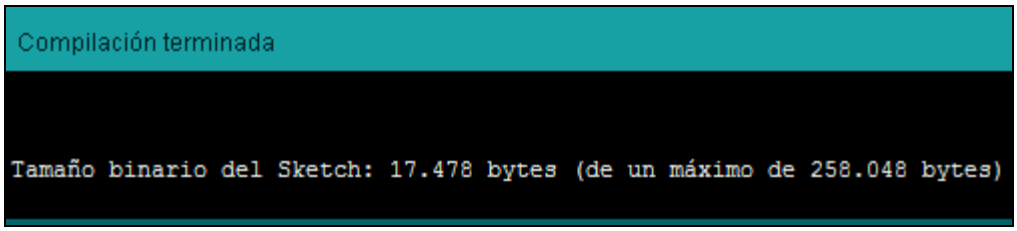

Figura 11. Compilación del programa del cálculo de la DCM en el software de Arduino. Fuente: Autores.

3.5 VERIFICACIÓN DE RESULTADOS

Enfocándose en la utilidad de la DCM, los ángulos de rotación del aeroplano se calculan con elementos específicos de la misma. En [9], [10] y [11] indican que el orden del giro afecta la DCM, por lo que de conformidad con la sección 3.2, se calculan los ángulos de rotación a partir de la DCM obtenida. Dado que la DCM no fue obtenida mediante las referencias mencionadas, no se conoce el orden de la misma y por ende, las ecuaciones para obtener los ángulos de navegación. Por esto, se implementa en el Ardupilot el cálculo de los ángulos de navegación de acuerdo a las ecuaciones (12), (13) y (14), y a [21].

Utilizando las ecuaciones descritas en la sección 3.2, se observa en la Figura 12 el cálculo de los ángulos de Euler en la tarjeta. Durante la toma de datos presentada, la tarjeta se encontraba en posición horizontal sin desplazamiento o perturbación alguna. Es posible observar que la DCM inicial proporciona valores para el cálculo de los ángulos cercanos a los reales (pues, la tarjeta solo se encuentra desplazada en el eje de guiñada), no obstante, es de esperarse que si la tarjeta se

mantiene detenida, los ángulos se mantengan constantes sin variación. En el caso presentado implementando el algoritmo, los valores de los ángulos divergen del valor esperado, siendo el valor del ángulo de cabeceo con mayor divergencia, el cual luego de unas muestras, se ha elevado por encima de ángulos que no son considerados despreciables. Los otros dos ángulos aparentemente no se encuentran afectados, no obstante, en el resultado presentado puede apreciarse como el incremento se hace a menor magnitud. Esto luego de un tiempo transcurrido, provoca que el valor del ángulo calculado regrese a su punto original, generando la interpretación de una rotación en todos sus ejes al mismo tiempo, y luego regresará a la posición original.

Una posible explicación se halla en el hecho que los sensores poseen un drift (error de acumulación) y esto implicaría un fallo crítico en el algoritmo propuesto. Más adelante se observará que este error no afecta significativamente las lecturas del acelerómetro y el magnetómetro. La explicación formal reside en la matemática del algoritmo. El diferencial de ángulo depende del vector de los giróscopos multiplicado por el tiempo de muestreo. El peso dado por s_g en la ecuación (23), es mucho mayor que el peso dado a los otros dos vectores [18]. Por observaciones empíricas, los giróscopos poseen una sensibilidad mayor que el acelerómetro o el magnetómetro, lo que provoca que sea en mayor medida susceptible a ruido, incluso, si desde el chip se implementa un filtro pasa-bajas. Las variaciones en este sensor son de mayor rapidez, en posición estática el sensor entrega datos diferentes de cero, provocando que la matriz no sea la misma en posición estática. En trabajos como el descrito por [19], [22], en la mayoría de trabajos dados por [17] e incluso en algunos apuntes de [8] y [23] proponen el uso de filtros digitales no lineales para la eliminación del ruido presente en estas lecturas. Luego, si existen valores diferentes de cero para una posición estática del dispositivo, la acumulación dada en el vector estimado para la misma matriz en posición estática incluirá este ruido, lo que se traduce en la

construcción de una DCM con datos erróneos de rotación, y por ende, se presenta la divergencia expuesta ya con anterioridad.

```
Inicio del programa
Inicializando los Sensores
Proceso de Calibracion del MPU
Offset: [Ax, Ay, Az], [Gx, Gy, Gz]
[245.00, 62.00, -1735.00], [-36.00, -5.00, -45.00]
Proceso de Calibracion del Compas
Calibracion del magnetometro [Mx, My, Mz]:
[0.99, 0.98, 1.00]
Retardo para la estabilizacion en la lectura del magnetometro. 2 segundos.
Calculo de la DCM Inicial.
|-0.4406833171  0.8976627349    0.0000559338|
|0.8976626396   0.4406833171    0.0001638122|
|0.0001223990   0.0001223990    -0.9999999046|
Pitch: -0.00
Roll: -0.01
Yaw: 116.15
[Pitch          Roll           Yaw     ]
[-3.09          -0.01          116.15  ]
[-6.59          -0.01          116.15  ]
[-9.50          -0.01          116.15  ]
[-12.78         -0.01          116.15  ]
[-15.92         -0.01          116.15  ]
[-18.51         -0.01          116.15  ]
[-21.41         -0.02          116.16  ]
[-24.16         -0.02          116.16  ]
[-26.42         -0.02          116.16  ]
[-28.91         -0.03          116.17  ]
[-31.28         -0.03          116.17  ]
[-33.20         -0.04          116.18  ]
[-35.33         -0.04          116.18  ]
[-37.34         -0.04          116.18  ]
[-38.96         -0.04          116.19  ]
[-40.75         -0.04          116.19  ]
[-42.43         -0.05          116.20  ]
[-43.79         -0.05          116.20  ]
[-45.28         -0.06          116.20  ]
[-46.68         -0.07          116.21  ]
[-47.81         -0.07          116.22  ]
[-49.05         -0.07          116.22  ]
[-50.21         -0.08          116.23  ]
```

Figura 12. Resultado obtenido del cálculo de la DCM usando el algoritmo propuesto. El encabezado de los datos presentados es el producto de la calibración de los sensores en posición horizontal. Fuente: Autores.

Se pudo entonces observar que la DCM inicial permitía calcular valores bastante cercanos a la realidad, esta DCM no utiliza el giróscopo sino únicamente el magnetómetro y el acelerómetro. Se realizó el experimento calculando únicamente

la matriz inicial, (aquella que no depende de los valores del giróscopo) tomando muestras en el tiempo de muestreo establecido, verificando si es posible obtener en tiempo real los ángulos de rotación. Los datos obtenidos comprenden datos relevantes que ayudan a calificar la viabilidad de algoritmos similares, esto es apreciable en las Figuras 13, 14 y 15.

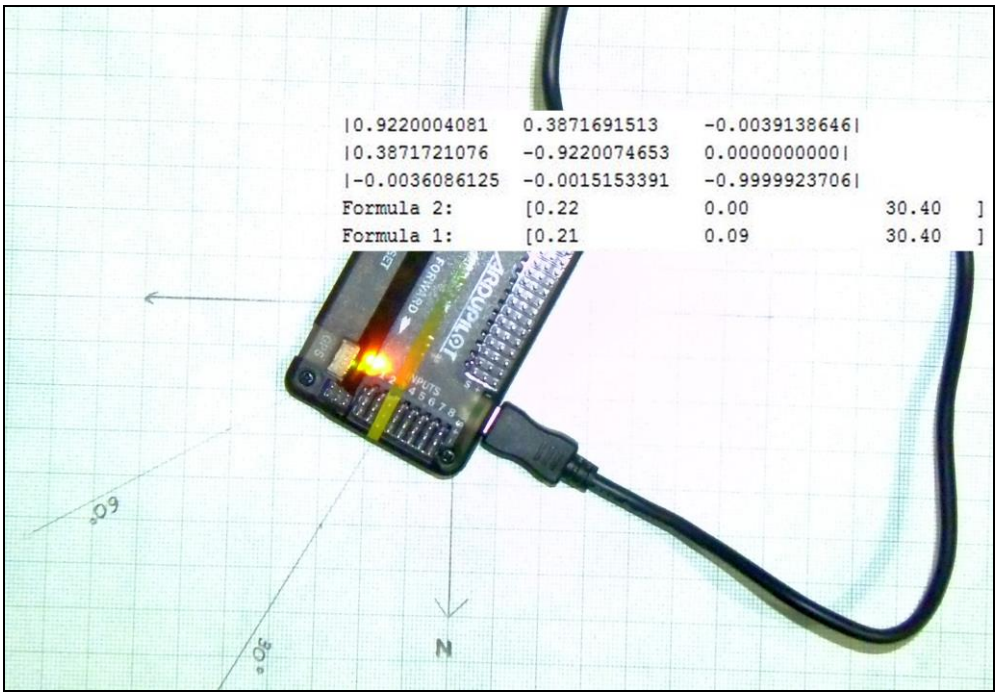

Figura 13. Verificación del cálculo del ángulo de guiñada usando los datos de la DCM, empleando dos fórmulas diferentes. El norte magnético se encuentra especificado en la superficie y una línea de referencia que indica la declinación. El ángulo de guiñada es el tercero de izquierda a derecha, indicando 30.4°. Fuente: Autores.

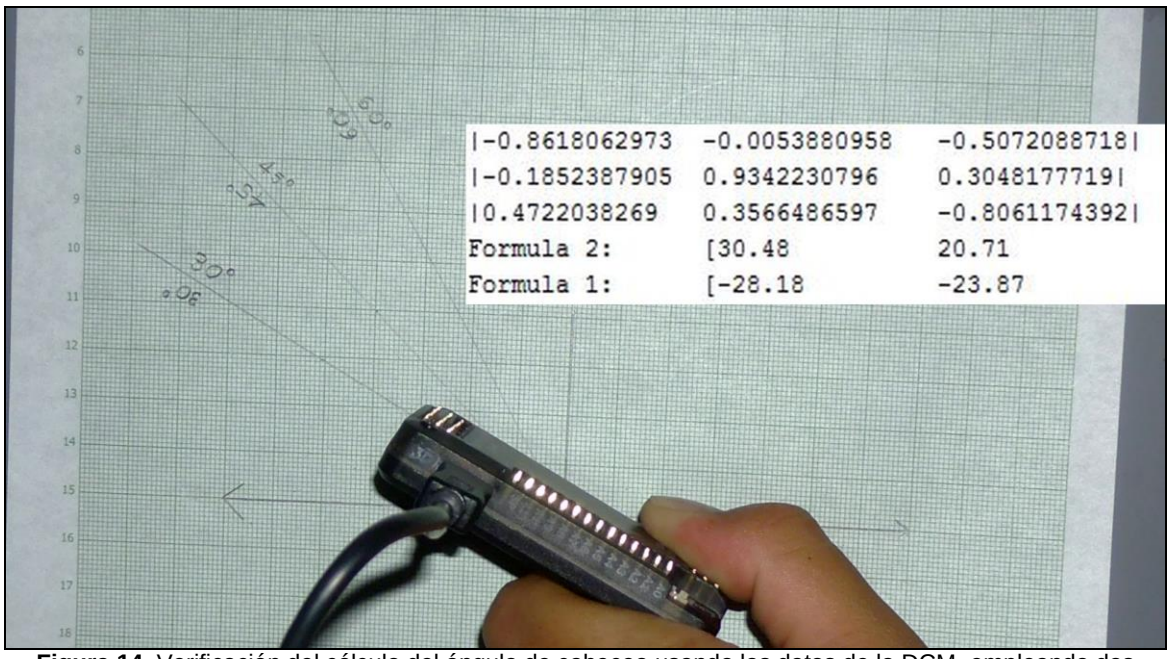

Figura 14. Verificación del cálculo del ángulo de cabeceo usando los datos de la DCM, empleando dos fórmulas diferentes. La línea horizontal se encuentra especificada en la superficie y una línea de referencia que indica la elevación. El ángulo de cabeceo es el primero de izquierda a derecha, indicando 30.48° y -28.18°. Fuente: Autores.

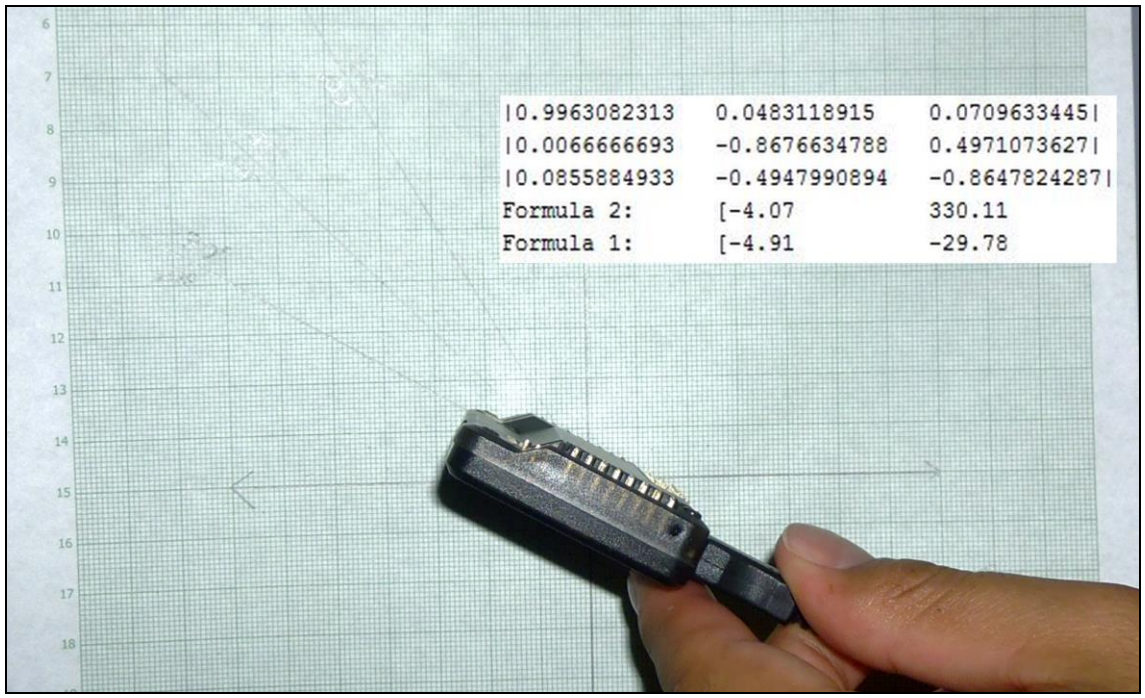

Figura 15. Verificación del cálculo del ángulo de alabeo usando los datos de la DCM, empleando dos fórmulas diferentes. La línea horizontal se encuentra trazada en la superficie y una línea de referencia que indica la inclinación. El ángulo de alabeo es el segundo de izquierda a derecha, indicando 330.11° y -29.78°. Fuente: Autores.

Es apreciable que los ángulos con cualquiera de las dos fórmulas son similares. La fórmula 1 presenta mayor error comparada con la fórmula 2, aunque su aproximación es bastante acertada con el ángulo medido, es evidente que la de mejor aproximación al ángulo medido esperado es la fórmula 2, usando las ecuaciones dispuestas en [21].

Esto arroja un resultado de gran interés, pues implica que sin importar la conFiguración de la DCM, la correspondencia de ángulos se mantiene, identificando físicamente la misma magnitud.

Datos del ArduIMU		
Alabeo [°]	Cabeceo [°]	Guiñada [°]
112.02	99.43	-170.39
112.11	100.51	-171.63
113.89	-101.78	-173.84
112.76	-101.47	-174.19
108.62	-103.17	-173.35
105.56	-103.77	-171.4
105.72	-103.96	-171.09
109.5	-103.54	-172.12
106.66	103.92	-170.97
108.19	-103.45	-171.8
105.13	-103.01	-170.4

Tabla 5. Datos obtenidos de la tarjeta ArduIMU. Puede apreciarse como los ángulos no son congruentes con la posición en la que se encuentra la tarjeta. Fuente: Autores.

Para verificar que otros autores hayan resuelto el problema de la divergencia y el cálculo de los ángulos únicamente con la determinación de la DCM, se implementó en una tarjeta de prestaciones similares un algoritmo que realiza las tareas mencionadas a partir de la DCM, esta tarjeta es el ArduIMU [23], e implementando el código disponible en [13], se imprime el resultado de los ángulos deseado en el monitor serial disponible por el programa base de Arduino,

observando en la Tabla 5, que el ángulo se mantiene constante (sin divergencia) a la inclinación dada, sin embargo, el ángulo calculado no hace correspondencia a la inclinación de la tarjeta, sostenida en forma manual. Este conjunto de resultados lleva a la conclusión que la DCM para el cálculo de ángulos de navegación es, por sus propios medios, insuficiente para llevar a cabo la tarea designada, y requiere de ayudas complementarias, que en algunas fuentes ([17], [18] y [19]) indica la necesidad de implementar filtros digitales y/o controladores digitales, a modo que la acción correctiva pueda expresar realmente el ángulo dado.

Como se encuentra dispuesto en la Figura 10, el vector del norte debe ser corregido, en la implementación se probó esta premisa, y se logró comprobar que al no obtener un vector completamente ortogonal al cenit, existe un offset en la inclinación delantera (ángulo de cabeceo), lo que genera que sea absolutamente necesaria su corrección, para obtener valores reales locales.

4. CONCLUSIONES.

La solución Ardupilot Mega 2.5 posee todas las características a nivel de hardware y software para efectuar control automático de vuelo en un aeroplano a escala no tripulado, siguiendo una ruta predeterminada por el usuario, con cambios de altura y trayectoria, utilizando el firmware base que ofrece la casa fabricante como prueba de esta premisa.

Sin importar los defectos de fabricación del aeroplano, o las perturbaciones a la que es sometido el UAV, el Ardupilot logra estabilizar el modelo en modo automático con un desempeño mayor que el logrado por un piloto con experiencia en aeronaves a escala desde tierra en forma manual.

La DCM, es determinada a partir de la secuencia de los giros (cabeceo, alabeo y guiñada) del avión, dicha matriz resulta diferente con solo realizar un cambio en la secuencia de rotación, pero, sin importar el resultado de la DCM, o el orden de la rotación realizada, los ángulos de Euler, cabeceo, alabeo y guiñada son siempre los mismos. Esto se comprobó al construir la DCM con la lectura de los sensores en la tarjeta Ardupilot, y partiendo de dicha matriz se llegaron a los mismos ángulos de Euler mediante 2 secuencias de giro diferentes.

La DCM para el cálculo de ángulos de navegación es, por sus propios medios, insuficiente para llevar a cabo la tarea de control de trayectoria designada, requiriendo de algoritmos de procesamiento complementarios que desarrollen soluciones a la singularidad de la matriz, a modo que la acción correctiva pueda converger con el mínimo error a la dirección deseada, sin embargo, una DCM corregida, sin divergencia de valores, es válida para diseñar una estrategia de control que permita estabilizar el avión durante una trayectoria uniforme e incluso bajo perturbaciones.

Las rutinas especificadas en este documento son el principio de la obtención de datos para la realimentación del sistema, a modo que futuros proyectos cuenten con una base funcional para la implementación de estrategias de control.

5. OBSERVACIONES.

Al perderse la línea de vista entre el receptor y el transmisor del kit de telemetría, se interrumpe la comunicación de los datos, necesarios para evaluar si el avión se encuentra realizando las maniobras ordenadas por el usuario. Se aconseja que todas las pruebas que se realicen sobre este tipo de sistema sean desarrolladas en campo abierto, de modo que la línea de visualización no se vea interrumpida por objetos u obstáculos de material sólido, tales como árboles o edificaciones.

Se hace necesaria la lectura completa del manual de manejo del software Mission Planner Mav, de modo que, la conFiguración del Ardupilot, los modos de vuelo, la calibración de los controles, el establecimiento de las rutas, el registro de datos de telemetría y otras conFiguraciones avanzadas, conlleven al éxito de la misión sin imprevistos, dificultades técnicas u omisión de procesos. Esto con el fin de obtener el mejor desempeño del potencial que la tarjeta ofrece.

Es importante tener en cuenta que la transmisión/recepción de datos por telemetría se realiza en forma inalámbrica, por lo que se debe considerar la frecuencia de trabajo de este dispositivo, de forma que no interfiera con la frecuencia de transmisión del radiocontrol.

La identificación del modelo de avión empleado representa en gran medida el éxito de la misión, pues de su conFiguración y de la utilización correcta de canales PWM para la maniobra del mismo, garantizarán que el piloto automático pueda estabilizar la aeronave en lugar de generar maniobras incorrectas que provoquen la colisión del modelo contra el suelo.

6. TRABAJOS FUTUROS.

En este documento se dejan plasmadas algunas bases que deben tenerse en cuenta para la realización de trabajos futuros en este campo de la ingeniería. Antes de tomar la tarjeta Ardupilot como la mejor opción para realizar diseños de algoritmos que controlan aeronaves a escala, se deja como trabajo futuro realizar una exhaustiva prueba a la tarjeta cuando esta se encuentre totalmente vertical, ($\theta = \pm\frac{\pi}{2} rad$, esto quiere decir que el frente apunte hacia arriba o hacia abajo) dado que en estos puntos (conocidos como ángulos críticos) existe una indeterminación, (sección 3.5) que en el presente trabajo de investigación no se abarca.

Las pruebas que se realicen para encontrar los ángulos de navegación (cabeceo, alabeo y guiñada), en la tarjeta a evaluar en forma física, deben ser lo suficientemente estrictos, por lo que se deja como trabajo futuro realizar estas pruebas de forma adecuada y con un margen de error mínimo, garantizando que las lecturas de los datos tomados sean los correctos, verificando en conjunto con el resultado del algoritmo.

También se deja como trabajo futuro revisar para optimizar el algoritmo que conlleva al código, (anexo 4) que se implementó para encontrar la DCM dinámica.

Es necesario que el cálculo del ángulo guiñada tenga incluida la compensación por declinación magnética local respecto del norte global. Esta solución no fue implementada en el algoritmo propuesto y se propone su revisión para futuros desarrollos sobre este modelo.

Con la información contemplada en el documento actual, es posible realizar una aproximación a una estrategia de control digital sobre la tarjeta en mención, en un principio, únicamente para estabilizar el avión durante el vuelo. El objetivo final deseado es desarrollar un control de piloto automático que siga rutas predefinidas, además de estabilizar la aeronave, controlar la posición y la dirección que toma

para seguir la ruta establecida. Se propone como trabajo futuro lograr la corrección de los datos leídos, con el ajuste pertinente en la estrategia de control para compensar el error, y de este modo, estabilizar en vuelo el aeroplano.

BIBLIOGRAFÍA

[1] Diseño, construcción, instrumentación y control de un vehículo Aéreo no tripulado (UAV), Tesis, Rafael Escamilla Núñez, Instituto politécnico nacional, México D.F. 2010. Disponible en http://itzamna.bnct.ipn.mx:8080/dspace/bitstream/123456789/9717/1/12.pdf.

Consultado el 20 de febrero de 2013.

[2] MPU-6000 and MPU-6050 Product Specification Revision 3.3, ©2011 InvenSense, Inc. All rights reserved. Disponible en http://www.invensense.com/mems/gyro/mpu6050.html. Consultado el 25 de abril de 2013.

[3] 3-Axis Digital Compass IC HMC5883L. Disponible en: http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf. Consultado el 5 de abril de 2013.

[4] MS5611-01BA03 Sensor de presión barométrica, recubierto en acero inoxidable. Disponible en: http://www.meas-spec.com/downloads/MS5611-01BA03.pdf. Consultado el 26 de abril de 2013

[5] Diy drones; Disponible en: http://www.diydrones.com/profiles/blogs/apm-2-0-release. Consultado el 22 de abril de 2013

[6] Declinación magnética de Bucaramanga. Disponible en: http://magnetic-declination.com/. Consultado el 11 de marzo de 2013

[7] Ajuste del magnetómetro (HMC5883L). Disponible en: https://www.loveelectronics.co.uk/Tutorials/8/hmc5883l-tutorial-and-arduino-library. Consultado el 12 de abril de 2013

[8] Modos de vuelo de la tarjeta Ardupilot Mega 2.5. Disponible en: https://code.google.com/p/ardupilot-mega/wiki/FlightModes. Consultado el 13 de abril de 2013

[9] Teoría de Máquinas, primera edición, Tecnun-Universidad de Navarra, Alejo Avello, 2012.

[10] Dynamics of Flight Stability and Control, tercera edición, Institute for Aerospace Studies, University of Toronto.

[11] Direction Cosine Matrix IMU: Theory, William Premerlani and Paul Bizard, 2009.

[12] Firmware base para la tarjeta Ardupilot Mega 2.5. Disponibles en: https://code.google.com/p/ardupilot-mega/downloads/detail?name=ArduPlane-2.71.zip&can=2&q=. Consultado el 12 de abril de 2013

[13] Firmware base para la tarjeta ArduIMU. Disponible en: https://code.google.com/p/ardu-imu/downloads/detail?name=ArduIMU_1.9.8.zip&can=2&q=. Consultado el 12 de abril de 2013.

[14] Especificaciones Técnicas del protocolo de comunicación SPI. Disponible en: http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf. Consultado el 15 de Abril de 2013.

[15] Protocolo SPI. Disponible en: http://www.intersil.com/content/dam/Intersil/documents/an13/an1340.pdf. Consultado el 17 de abril de 2013.

[16] Esquemas de la tarjeta Ardupilot Mega 2.5. Disponible en: http://stuff.storediydrones.com/APM_v252_RELEASE.zip. Consultado el 2 de abril de 2013.

[17] Papers escritos por: Robert Mahony. Disponibles en: http://gentlenav.googlecode.com/files/MahonyPapers.zip. Consultados el 15 de marzo de 2013.

[18] DCM Tutorial - An introduction to orientation Kinematics, Sergio Baluta fundador del proyecto STARLINO, Disponible en: http://www.starlino.com/dcm_tutorial.html, Consultado el 12 de abril de 2013.

[19] A guide to using IMU (Accelerometer and gyroscope Devices), in embedded Applications, Sergio Baluta fundador del proyecto STARLINO, Disponible en: http://www.starlino.com/imu_guide.html, Consultado el 12 de abril de 2013.

[20] DCM Tutorial - An introduction to orientation Kinematics, comentario 2, Sergio Baluta fundador del proyecto STARLINO, Disponible en: http://www.starlino.com/dcm_tutorial.html#comment-925, Consultado el 12 de abril de 2013.

[21] Direction Cosine Matrix to Euler Angles, Disponible en: http://www.weizmann.ac.il/matlab/toolbox/aeroblks/directioncosinematrixtoeulerangles.html, Consultado el 14 de abril de 2013.

[22] Open source IMU and AHRS algorithms, Disponible en: http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/, Consultado el 09 de abril de 2013

[23] ArduIMU, Disponible en: https://code.google.com/p/ardu-imu/, Consultado el 16 de abril de 2013.

[24] Cuaterniones, Disponible en: http://en.wikipedia.org/wiki/Euler_angles#Quaternions, Consultado el 08 de abril de 2013

[25] Ángulos de navegación, Disponible en: http://1.bp.blogspot.com/_nibSxQl2aCk/TBB5L_xzS1I/AAAAAAAABZc/EtJ6-FXd5G4/s1600/YawPitchRoll.jpg, Consultado el 03 de abril de 2013

[26] Proyección del marco del avión sobre el marco de referencia en tierra, Disponible en: https://pixhawk.ethz.ch/px4/dev/frames_of_reference, Consultado el 02 de abril de 2013

ANEXOS

Anexo 1. Pruebas de vuelo en forma audiovisual

Prueba de vuelo en modo manual, disponibles en:

http://www.youtube.com/watch?v=D72Wh5Q__LM

http://www.youtube.com/watch?v=n7qSYtcJ-9w

Prueba de telemetría, disponible en:

http://youtu.be/WtCAZ_7EY50

Pruebas vuelo en modo de piloto automático, disponible en:

http://youtu.be/7-zNu_E_Nag.

Anexo 2. Librería de operaciones Matriciales y Vectoriales.

```
void sumarVectores(double* sumando1, double* sumando2, double* suma, byte longitud){
	byte i;

	for(i=0;i<longitud;i++){
		*suma=*sumando1+*sumando2;
		suma++;
		sumando1++;
		sumando2++;
	}
}


void restarVectores(double* restando1, double* restando2, double* resta, byte longitud){
	byte i;

	for(i=0;i<longitud;i++){
		*resta=*restando1-*restando2;
		resta++;
		restando1++;
		restando2++;
	}
}


void productoPorEscalar(double escalar, double* vector, double* resultado, byte longitud){
	byte i;

	for(i=0;i<longitud;i++){
		*resultado=escalar*(*vector);
		resultado++;
		vector++;
	}
}


double magnitudVector(double* vector, byte longitud){
	byte i;
	double magnitud=0;

	for(i=0;i<longitud;i++){
		magnitud+=(*vector)*(*vector);
		vector++;
	}
	magnitud=sqrt(magnitud);
	return magnitud;
}
```

```
void normalizarVector(double* vector, double* resultado, byte longitud){
	signed int i;
	double magnitud;

	magnitud=magnitudVector(vector, longitud);
	magnitud=1.0/magnitud;
	productoPorEscalar(magnitud, vector, resultado, longitud);
}


void productoVectorial(double* vector1a, double* vector2a, double* resultado, byte longitud){
	//esta función asume que el producto es de un vector de 3x3. No se ha diseñado para
	// vectores generalizados
	byte i;
	double suma, resta;

	//elemento i del vector: V1yV2z-V1zV2y
	vector1a++;    //ey
	vector2a+=2;    //ez
	suma=(*vector1a)*(*vector2a);
	vector1a++;    //ez
	vector2a--;    //ey
	resta=(*vector1a)*(*vector2a);
	*resultado=suma-resta;
	//elemento y del vector: V1zV2x-V1xV2z
	resultado++;
	vector2a--;    //ex
	suma=(*vector1a)*(*vector2a);
	vector1a-=2;    //ex
	vector2a+=2;    //ez
	resta=(*vector1a)*(*vector2a);
	*resultado=suma-resta;
	//elemento z del vector: V1xV2y-V1yV2x
	resultado++;
	vector2a--;    //ey
	suma=(*vector1a)*(*vector2a);
	vector1a++;    //ey
	vector2a--;    //ex
	resta=(*vector1a)*(*vector2a);
	*resultado=suma-resta;
}
```

Anexo 3. Librerías de cálculo del magnetómetro.

HMC5883.h

```
#define inclinacionBucaramanga 0.133


byte c;
byte magnetTempCalibrado=0;


double magnetTempCal[3]={1.0, 1.0, 1.0}, magnetAjusteInterno[3]={0.0, 0.0, 0.0};
```

HMC58853.cpp

```
// configuración del sensor HMC5883 para el Ardupilot Mega 2.5


void HMC5883_init(){


        Wire.begin();
        Wire.beginTransmission(0x1E);    //dirección del dispositivo
        Wire.write(0x00);               //registro A
        Wire.write(0x78);               //8 muestras en promedio; 75 hertz;
        Wire.write(0x01);               //registro B
        Wire.write(0xA0);               //4.7 gauss
        Wire.write(0x02);               //registro mode
        Wire.write(0x00);               //modo continuo, i2c en low speed
        c=Wire.endTransmission();
}


void HMC5883_readUnscaled(){
        byte c, datos[6], i, j;
        double dato[3];


        magnet[0]=-1000000;
        magnet[1]=-1000000;
```

```
	magnet[2]=-1000000;



	dato[0]=-1000000;
	dato[1]=-1000000;
	dato[2]=-1000000;



	if(digitalRead(DRDY_PIN)==HIGH){
		Wire.beginTransmission(0x1E);
		Wire.write(0x03);    //posiciono en el primer registro de datos
		c=Wire.endTransmission();

		Wire.beginTransmission(0x1E);
		c=Wire.requestFrom(0x1E,6);
		//leo el vector de datos desde el magnetometro
		if(Wire.available() == 6){
			for(i=0;i<6;i++){
				datos[i]=Wire.read();
			}
		}
		c=Wire.endTransmission();
		//transformo los datos, pues, requiero ver su valor en un solo byte, no por
	separdado
		//dos bytes
		j=0;
		for(i=0;i<3;i++){
			dato[i]=double((datos[j]<<8) | datos[j+1]);
			j+=2;
		}
		//los guardo en el orden correcto, y sin escalar
		magnet[0]=dato[0];
		magnet[1]=dato[2];
		magnet[2]=dato[1];
	}
}


void HMC5883_read(){
	byte i;



	HMC5883_readUnscaled();



	magnet[0]=magnet[0]*escala;
	magnet[1]=magnet[1]*escala;
	magnet[2]=magnet[2]*escala;
}
```

```
void HMC5883_readCalibrated(){
	byte i;


	HMC5883_readUnscaled();


	magnet[0]=magnet[0]*escala*offset[6]*magnetTempCal[0];
	magnet[1]=magnet[1]*escala*offset[7]*magnetTempCal[1];
	magnet[2]=magnet[2]*escala*offset[8]*magnetTempCal[2];
}


float calcularInclinacion(byte compensacion){
	float inclinacion;

	//el valor de la variable compensancion es para en un futuro lograr compensar si la tarjeta
no
	//se encuentra paralela al plano XY
	inclinacion=atan2(magnet[0], -magnet[1])+inclinacionBucaramanga;
	inclinacion+=2.0*M_PI;
	if(inclinacion<0){
		inclinacion+=(2.0*M_PI);
	}else if(inclinacion>2.0*M_PI){
		inclinacion-=(2.0*M_PI);
	}
	inclinacion=inclinacion*(180.0/(M_PI));
	return inclinacion;
}


//*****************************************************************


void calibrarTemperatura(){


	byte i, conta=0, contador=0;
	double cal[3],calibrador[3],esperado[3]={1580.0, 1500.0, 1500.0};	//los	datos	de
esperado
							//son basados en
							//mediciones
							//experimentales


	//configuro el dispositivo
```

```
Wire.begin();
Wire.beginTransmission(0x1E);    //dirección del dispositivo
Wire.write(0x00);              //registro A
Wire.write(0x79);              //8 muestras en promedio; 75 hertz; self_test positivo;
Wire.write(0x01);              //registro B
Wire.write(0xA0);               //4.7 gauss, 390 LSb/gauss
Wire.write(0x02);              //registro mode
Wire.write(0x00);              //modo continuo, i2c en low speed
i=Wire.endTransmission();



//esperamos que el dispositivo se establezca
delay(50);

//leo el vector de valores, un máximo de 30 intentos, 5 valores únicamente
while(conta<30){
        HMC5883_readUnscaled();
        for(i=0;i<3;i++){
                cal[i]=double(esperado[i])/magnet[i];
                cal[i]=abs(cal[i]);
        }
        if(cal[0]>0.7 && cal[0]<1.3
                && cal[1]>0.7 && cal[1]<1.3
                && cal[2]>0.7 && cal[2]<1.3){
                for(i=0;i<3;i++){
                        calibrador[i]+=cal[i];
                }
                contador++;
        }
        //incrementamos el contador de ciclos infinitos
        conta++;
        //esperamos un tiempo prudencial para el siguiente dato, máximo pasaran 3
        //segundos calibrando
        delay(100);
}



if(magnetTempCalibrado==0){
        //leemos los valores para actualizar el registro temporal de valores leídos, dejamos
la
        //calibración intacta.
        //preguntamos si pudo calibrar, o si no
        if(contador>=5){
                for(i=0;i<3;i++){
                        //promedio de la calibracion
                        magnetAjusteInterno[i]=calibrador[i]/contador;
                }
                magnetTempCalibrado=1;
        }
        }else{
                //actualiza los valores leidos, recalculamos la calibracion
                if(contador>=5){
```

```
                                for(i=0;i<3;i++){
                                        //promedio de la calibracion, guardamos el ajuste
                                        calibrador[i]=calibrador[i]/contador;
                                        magnetTempCal[i]=magnetAjusteInterno[i]/calibrador[i];
                                        magnetAjusteInterno[i]=calibrador[i];
                                }
                        }
                }
                //re inicializamos el dispositivo
                HMC5883_init();
}


//*************************************************************
//*************************************************************


void HMC5883_calibrar(){
        byte i, conta=0, contador=0;
        double cal[3],esperado[3]={1580, 1500, 1500};           //los datos de esperado son
                                                                  //basados en mediciones
                                                                  //experimentales

        //calibramos temperatura
        if(magnetTempCalibrado==0){
                calibrarTemperatura();
        }
        //configuro el dispositivo
        Wire.begin();
        Wire.beginTransmission(0x1E);    //dirección del dispositivo
        Wire.write(0x00);               //registro A
        Wire.write(0x79);               //8 muestras en promedio; 75 hertz; self_test positivo;
        Wire.write(0x01);               //registro B
        Wire.write(0xA0);                //4.7 gauss, 390 LSb/gauss
        Wire.write(0x02);               //registro mode
        Wire.write(0x00);               //modo single, i2c en low speed
        i=Wire.endTransmission();



        //establezco a cero el vector de calibración
        for(i=6;i<9;i++){
                offset[i]=0;
        }
        //esperamos que el dispositivo se establezca
        delay(50);

        //leo el vector de valores, un máximo de 30 intentos, 5 valores únicamente
        while(conta<30){
                HMC5883_readUnscaled();
                for(i=0;i<3;i++){
                        cal[i]=double(esperado[i])/magnet[i];
```

```
                cal[i]=abs(cal[i]);
            }
            if(cal[0]>0.7 && cal[0]<1.3
                && cal[1]>0.7 && cal[1]<1.3
                && cal[2]>0.7 && cal[2]<1.3){
                for(i=6;i<9;i++){
                    offset[i]+=cal[i-6];
                }
                contador++;
            }
            //incrementamos el contador de ciclos infinitos
            conta++;
            //esperamos un tiempo prudencial para el siguiente dato, máximo pasaran 3
            //segundos calibrando
            delay(100);
        }
        //preguntamos si pudo calibrar, o si no
        if(contador>=5){
            for(i=0;i<3;i++){
                //calibracion=calibracion/muestras*GainElegida/GainAtrabajar;
                //GainAtrabajar=390
                offset[i+6]=offset[i+6]/contador*390.0/390.0;
            }
        }else{
            for(i=6;i<9;i++){
                //la mejor estimación?
                offset[i]=1.0;
            }
        }
        //re calibro a la condición inicial deseada
        HMC5883_init();
        Serial.println("Calibracion del magnetometro [Mx, My, Mz]:");
        Serial.print("[");
        for(i=6;i<9;i++){
            //la mejor estimación?
            if(i!=6){
                Serial.print(", ");
            }
            Serial.print(offset[i]);
        }
        Serial.println("]");
}
```

Anexo 4. Función extra librería del MPU

```
void MPU6000_ReadWithOffset(){
        if(datoViejo!=MPU6000_newdata){

                datoViejo=MPU6000_newdata;
                MPU6000_Read();

                //el valor dividido depende de la configuración seleccionada
                acel[0]=(accelX-offset[0])/8192.0;
                acel[1]=(accelY-offset[1])/8192.0;
                acel[2]=(accelZ-offset[2])/8192.0;
                gyro[0]=(gyroX-offset[3])/16.4;
                gyro[1]=(gyroY-offset[4])/16.4;
                gyro[2]=(gyroZ-offset[5])/16.4;
                tempt=temp/((double)340.0)+36.53;
        }
}
```

Anexo 5. Código para el cálculo de la DCM.

```
void getVectorKb(double* acelerometro){
	byte i;

	MPU6000_ReadWithOffset();
	for(i=0;i<3;i++){
		*acelerometro=-accel[i];
		acelerometro++;
	}
}



void getVectorIb(double* magnetometro){
	byte i;

	HMC5883_readCalibrated();
	for(i=0;i<3;i++){
		*magnetometro=magnet[i];
		magnetometro++;
	}
}



void getVectorIbCorrected(double* magnetometro, double* acelerometro){
	double temp[3];

	productoVectorial(acelerometro, magnetometro, temp, sizeof(temp)/sizeof(double));
	productoVectorial(temp, acelerometro, magnetometro, sizeof(temp)/sizeof(double));
}



void getVectorNormalized(double* magnetometro, double* acelerometro, double* ortogonal){
	double temp[3];
	signed int i;

	normalizarVector(magnetometro, temp, sizeof(temp)/sizeof(double));
	for(i=0;i<3;i++){
		*magnetometro=temp[i];
		magnetometro++;
	}
	for(i=2;i>=0;i--){
		magnetometro--;
	}
	normalizarVector(acelerometro, temp, sizeof(temp)/sizeof(double));
	for(i=0;i<3;i++){
		*acelerometro=temp[i];
		acelerometro++;
	}
	for(i=2;i>=0;i--){
		acelerometro--;
```

```
	}
	productoVectorial(acelerometro, magnetometro, ortogonal, sizeof(temp)/sizeof(double));
}


void construirDCM(double* magnetometro,double* ortogonal, double* acelerometro){
	byte i;

	for(i=0;i<3;i++){
		_DCM[0][i]=*magnetometro;
		magnetometro++;
	}
	for(i=0;i<3;i++){
		_DCM[1][i]=*ortogonal;
		ortogonal++;
	}
	for(i=0;i<3;i++){
		_DCM[2][i]=*acelerometro;
		acelerometro++;
	}
}


double getRoll(){
	double roll;



	roll=atan2(_DCM[1][2], _DCM[2][2]);
	roll=roll*180.0/M_PI-180.0;
	return roll;
}


double getPitch(){
	double pitch;



	pitch=asin(-_DCM[0][2]);
	pitch=pitch*180.0/M_PI;
	return pitch;
}


double getYaw(){
	double yaw;



	yaw=atan2(_DCM[0][1], _DCM[0][0]);
	yaw=yaw*180.0/M_PI;
```

```
	return yaw;
}


void calcular_dTg(double ts, double* vectorW, double* dTg, byte longitud){
	productoPorEscalar(ts, vectorW, dTg, longitud);
}


void calcular_dTm(double* vectorI, double* vectorM, double* dTm, byte longitud){
	double temp[3];
	//restarVectores(vectorM, vectorI, temp, longitud);
	//productoVectorial(vectorI, temp, dTm, longitud);
	productoVectorial(vectorI, vectorM, dTm, longitud);
}


void calcular_dTa(double* vectorK, double* vectorA, double* dTa, byte longitud){
	double temp[3];
	restarVectores(vectorA, vectorK, temp, longitud);
	productoVectorial(vectorK, temp, dTa, longitud);
}


void calcular_dT(double* vectorW, double* vectorA, double* vectorM, double* resultado, double* Ia, double* Ka){
	double dTa[3], dTm[3], dTg[3], temp[3], _temp[3], Sa=0.01, Sg=1, Sm=0.0, St=Sa+Sg+Sm;
	signed int i;

	calcular_dTg(Ts/1000.0, vectorW, dTg, 3);
	calcular_dTm(Ia, vectorM, dTm, 3);
	calcular_dTa(Ka, vectorA, dTa, 3);
	//recalculo dTa
	productoPorEscalar(Sa, dTa, temp, 3);
	for(i=0;i<3;i++){
		dTa[i]=temp[i];
	}
	//recalculo dTg
	productoPorEscalar(Sg, dTg, temp, 3);
	for(i=0;i<3;i++){
		dTg[i]=temp[i];
	}
	//recalculo dTm
	productoPorEscalar(Sm, dTm, temp, 3);
	for(i=0;i<3;i++){
		dTm[i]=temp[i];
	}
	//calculo la suma
	sumarVectores(dTa, dTg, temp, 3);
	sumarVectores(temp, dTm, _temp, 3);
	productoPorEscalar(1/St, _temp, resultado, 3);
}
```

Anexo 6. Programa del cálculo de DCM Utilizando el ArduPilot Mega 2.5

```
#include "MPU6000.h";
#include "HMC5883.h";
#include "algebraMatricial.h"
#include "DCM.h"
#include <Wire.h>

#define DRDY_PIN  31

#define resolucion 390
#define escala 2.56
#define Ts 30

int datoViejo=0;
byte calibracion=0;
double gyro[3], accel[3], magnet[3], tempt;
double offset[9];
double Ib[2][3], Jb[2][3], Kb[2][3], _DCM[3][3];
long tActual, tAnterior, t1Ac, t1An;

void setup(){

        Serial.begin(9600);
        Serial.println("Inicio del programa");

        MPU6000_newdata=0;
        calibracion=0;

        //deteniendo el barómetro, pues este ocupa el bus SPI
        pinMode(40, OUTPUT);
        digitalWrite(40, HIGH);

        //Establezco pin de lectura nueva magnetómetro
        pinMode(DRDY_PIN, INPUT);

        Serial.println("Inicializando los Sensores");
        MPU6000_Init();
        HMC5883_init();

        //se calibran los dispositivos
        Serial.println("Proceso de Calibracion del MPU");
        MPU6000_calibrar();
        Serial.println("Proceso de Calibracion del Compas");
        HMC5883_calibrar();

        Serial.println("Retardo para la estabilizacion en la lectura del magnetometro. 2 segundos.");
        delay(5000);
        Serial.println("Calculo de la DCM Inicial.");
        //se lee la primera matriz DCM
        getVectorKb(Kb[0]);
```

```
	getVectorIb(Ib[0]);
	getVectorIbCorrected(Ib[0], Kb[0]);
	getVectorNormalized(Ib[0], Kb[0], Jb[0]);
	construirDCM(Ib[0], Jb[0], Kb[0]);
	//muestro el DCM
	for(int i=0;i<3;i++){
		Serial.print("|");
		for(int j=0;j<3;j++){
			if(j!=0){
				Serial.print("\t");
			}
			Serial.print(_DCM[i][j], 10);
		}
		Serial.println("|");
	}
	//calculo angulos de "Euler"
	Serial.print("Pitch: ");
	Serial.println(getPitch());
	Serial.print("Roll: ");
	Serial.println(getRoll());
	Serial.print("Yaw: ");
	Serial.println(getYaw());
	//valor actual igual al anterior.
	for(int i=0;i<3;i++){
		Ib[1][i]=Ib[0][i];
		Jb[1][i]=Jb[0][i];
		Kb[1][i]=Kb[0][i];
	}
	//obtengo el dato del tiempo actual para conocer cuando ha transcurrido mi Ts
	tAnterior=t1An=millis();
	Serial.println("[Pitch\t\tRoll\t\tYaw\t]");
}

void loop(){

	double W[3], A[3], M[3], dT[3], temp[3], Ip[3], Jp[3], err;
	byte i;

	tActual=millis();
	//solo si el tiempo de muestreo ha transcurrido
	if((tActual-tAnterior)>=Ts){
		tAnterior=tActual;
		//leemos los vectores en este instante
		/*MPU6000_ReadWithOffset();
		HMC5883_readCalibrated();
		for(i=0;i<3;i++){
			W[i]=gyro[i]*M_PI/180.0;
			A[i]=-accel[i];
			M[i]=magnet[i];
		}
		//se normalizan los vectores
		normalizarVector(A, temp, sizeof(temp)/sizeof(double));
		for(i=0;i<3;i++){
			A[i]=temp[i];
```

```
		}
		normalizarVector(M, temp, sizeof(temp)/sizeof(double));
		for(i=0;i<3;i++){
			M[i]=temp[i];
		}
		//el vector anterior es igual al actual
		for(int i=0;i<3;i++){
			Ib[0][i]=Ib[1][i];
			Jb[0][i]=Jb[1][i];
			Kb[0][i]=Kb[1][i];
		}
		//calculamos la diferencia en angulos
		calcular_dT(W, M, A, dT, Ib[0], Kb[0]);
		//calculamos los nuevos vectores I y J
		productoVectorial(dT, Ib[0], temp, 3);
		sumarVectores(Ib[0], temp, Ib[1] ,3);
		productoVectorial(dT, Jb[0], temp, 3);
		sumarVectores(Jb[0], temp, Jb[1], 3);
		//encuentro el error
		err=productoPunto(Ib[1], Jb[1], 3);
		err=err/2.0;
		productoPorEscalar(err, Jb[1], Ip, 3);
		productoPorEscalar(err, Ib[1], Jp, 3);
		restarVectores(Ib[1], Ip, temp, 3);
		for(i=0;i<3;i++){
			Ib[1][i]=temp[i];
		}
		restarVectores(Jb[1], Jp, temp, 3);
		for(i=0;i<3;i++){
			Jb[1][i]=temp[i];
		}
		productoVectorial(Ib[1], Jb[1], Kb[1], 3);
		//getVectorIbCorrected(Ib[1], Kb[1]);
		normalizarVector(Ib[1], Ib[1], 3);
		normalizarVector(Jb[1], Jb[1], 3);
		normalizarVector(Kb[1], Kb[1], 3);
		construirDCM(Ib[1], Jb[1], Kb[1]);*/
		getVectorKb(Kb[0]);
		getVectorIb(Ib[0]);
		getVectorIbCorrected(Ib[0], Kb[0]);
		getVectorNormalized(Ib[0], Kb[0], Jb[0]);
		construirDCM(Ib[0], Jb[0], Kb[0]);
}
t1Ac=millis();
if((t1Ac-t1An)>=200){
		t1An=t1Ac;
		//muestro el DCM
		for(int i=0;i<3;i++){
			Serial.print("|");
			for(int j=0;j<3;j++){
				if(j!=0){
					Serial.print("\t");
				}
				Serial.print(_DCM[i][j], 10);
```

```
                    }
                    Serial.println("|");
                }
                //calculo angulos de "Euler"
                Serial.print("Formula 2:\t[");
                Serial.print(getPitch2());
                Serial.print("\t\t");
                Serial.print(getRoll2());
                Serial.print("\t\t");
                Serial.print(getYaw2());
                Serial.println("\t]");
                //calculo angulos de "Euler"
                Serial.print("Formula 1:\t[");
                Serial.print(getPitch());
                Serial.print("\t\t");
                Serial.print(getRoll());
                Serial.print("\t\t");
                Serial.print(getYaw());
                Serial.println("\t]");
        }
}
```