UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Interfaz sensorial de un brazo humano

Proyecto final de grado

desarrollado y entregado en

Escuela Técnica de Ingeniería de Telecomunicaciones de Barcelona

Universidad Politécnica de Catalunya

realizado por

Sergi Baños Lara

En cumplimiento de los requisitos para el

Grado de ciencias y tecnologías de la telecomunicación

Tutor: Vicente Jiménez

Barcelona, Febrero 2015

Abstract

With this project, has been created an interface which can capture, process and transmit the motion of a human arm using three sensors boards and one microcontroller. The project has been divided in two blocks, a sensor capture block and a processing signal block. The first one captures the acceleration, rotation rate and the orientation respect the earth on three arm articulations. The second part is about the signal processing, two types of filters (the AHRS and the IMU) have been implemented based on the gradient descent algorithm. Those two filters use the minimum arithmetic operations that have been possible, 374 operations for the AHRS and 149 for the IMU. Furthermore, a module has been added in order to compensate the magnetic earth's distortion on the AHRS filter. The filters use the information provided by the sensors and return back a quaternion, for each arm's articulation, which contains the articulation's position regarding the terrestrial globe. Finally, the data is sent to the PC in order to recreate the arm's movement.

Resum

En aquest projecte s'ha creat una interfase capaç de captar, processar i transmetre el moviment d'un braç humà mitjançant sensors i un micro-controlador. El projecte es pot dividir en dos grans blocs, la part electrònica y la part de processat de senyal. En la primera part, es captura l'acceleració, la velocitat de rotació i la orientació respecte la Terra de tres articulacions del braç. La segona part consta del processat de senyal, on s'ha realitzat un filtre iteratiu basat en l'algoritme de gradient descendent, pel qual s'ha pogut implementar dos tipus de filtres, el AHRS i el IMU. Aquests dos filtres s'han implementat amb el mínim nombre d'operacions aritmètiques que ha sigut possible, 374 operacions pel AHRS i 149 pel IMU. Més a més, s'ha afegit al filtre AHRS, un mòdul per la compensació de la distorsió magnètica del camp de la terra. Els filtres utilitzen la informació que provenen dels sensors i en retornen un quaternió per a cada articulació, el qual conté la informació de l'orientació de l'articulació respecte el globus terraqüi. Per últim, les dades s'envia al PC amb la finalitat de recrear el moviment del braç.

Resumen

En este proyecto se ha creado una interfaz capaz de captar, procesar y transmitir el movimiento de un brazo humano mediante sensores y un micro controlador. El proyecto se puede dividir en dos grandes partes, la parte electrónica y la parte de procesado de señal. En la primera parte se ha hecho uso del protocolo de comunicación I2C y de tres sensores, los cuales proveen los datos de la orientación de cada articulación al micro controlador. La segunda parte consta del procesado de señal, donde se ha realizado un filtro iterativo basado en el algoritmo de gradiente descendiente, a partir del cual se ha podido implementar dos tipos de filtros, el AHRS y el IMU. Ambos se han implementado con el mínimo número de operaciones aritméticas posibles, 374 y 149 operaciones aritméticas, respectivamente. Además, al filtro AHRS, se le ha añadido un módulo para la compensación de la distorsión magnética del campo de la tierra. Los filtros utilizan la información que proviene de los sensores y devuelven un cuaternión por cada articulación, el cual contiene la información de cuál es la orientación de cada sensor respecto al globo terráqueo. Finalmente, el los datos se envía al PC con la finalidad de recrear el movimiento.

Historial de revisions del documento

Revisión	Fecha	Objetivo
0	24/12/2014	Creación del documento
1	03/02/2015	Revisión del documento
2	05/02/2015	Revisión del documento

Lista de distribución del documento

Nombre	e-mail
Sergi Baños	15sergi1991@gmail.com
Vicente Jiménez	vicente.jimenez@upc.edu

Escrito por:		Revisado y aprobado por:	
Fecha	05/02/2015	Fecha	06/0272015
Nombre	Sergi Baños Lara	Nombre	Vicente Jiménez
Posición	Autor del proyecto	Posición	Supervisor del proyecto

Índice

Abstract 1
Resum 1
Resumen 1
Historial de revisions del documento 2
Índice 3
Lista de figuras 5
Lista de tablas 6
1 Introducción 7
1.1. Objetivos y requisitos 7
1.2. Hitos en el proyecto 7
1.3. Metodología 8
1.4. Dificultades 8
2 Trasfondo teórico: 9
2.1. Representación con cuaterniones 9
2.2. Filtraje derivativo 10
2.2.1. Orientación a partir de la velocidad angular 10
2.2.2. Orientación a partir de los vectores de campos magnético y gravitatorio 11
2.3. Algoritmo final 14
2.4. Compensación de la distorsión magnética 16
2.5. Ganancia del filtro 17
3 Experimentación: 18
3.1. Comunicación I2C 18
3.1.1. Protocolo de transmisión 18
3.1.2. Transmitiendo datos 19
3.1.3. Direcciones de 7-bits 19
3.1.4. Resistencias Pull-up 19
3.2. Sensores 20
3.2.1. Altimu-10 v.4 20
3.3. Microprocesador 21
3.3.1. STM32F4 21
3.3.2. Sistema operativo (ChibiOS/RT) 22
3.4. Procedimiento experimental 22
4 Resultados 24

4.1. Rendimiento estático y dinámico 24
4.2. Rendimiento de software 25
5 Conclusiones y futuros desarrollos: 26
6 Bibliografía 27
7 Apéndices 28
7.1. I2C specifications 28
7.2. Altimu -10 v.4 (Información extra) 28
7.2.1. Sensibilidad de los sensores 28
7.2.2. Esquema 28
7.3. Algoritmo (C-Script) 29
7.3.1. AHRS 29
7.3.2. IMU 30
8 Glosario 32

Lista de figuras

Figura 1: Vectores unitarios del instante A y B 9

Figura 2: Diagrama de bloques del algoritmo IMU 15

Figura 3: Diagrama de bloques del algoritmo AHRS con la compensación de distorsión magnética 16

Figura 4: Condición de inicio y finalización del bus I2C [6] 18

Figura 5: Configuración del hardware, la placa STM32F4 a la izquierda y las placas de los sensores a la derecha 21

Figura 6: Bloques del programa en C 22

Figura 7: Modelo 3D del brazo humano en Unity 23

Figura 8: Esquemático de la placa AltIMU v4 28

Lista de tablas

Tabla 1: Direcciones de los sensores en función del estado de SA0 20
Tabla 2: Error RMS de los ángulos de Euler en estático y en dinámico 24
Tabla 3: Rangos de medida de los sensores 28

1 Introducción

1.1. Objetivos y requisitos

El principal objetivo de este trabajo es la creación de una interfaz capaz de medir la actividad y el movimiento del brazo humano para poder enviarlo a un ordenador. Para lograr este fin, la interfaz debe estar dotada de un conjunto de sensores capaz de medir la posición y orientación. Este tipo de sensores se suelen utilizar en proyectos aeroespaciales, robótica, navegación y análisis del movimiento humano. Para saber la posición en que se encuentra un brazo humano basta con saber la orientación en la que se encuentra cada una de las tres articulaciones que lo componen, hombro, codo, y muñeca. Para ello, se puede situar una placa de sensores en cada una de estas articulaciones. Para poder cumplir este primer objetivo del proyecto, la interfaz debe cumplir unos requisitos específicos. Primero, la interfaz ha de estar dotada de sensores locales. Por locales se entiende que toda la instrumentación que requiera la interfaz no esté condicionada a un espacio en concreto, sino que todos los sensores deben estar sobre el brazo a monitorizar. Segundo, toda la captación de información que proviene de los sensores y todo el procesado de señal deben estar programados en el micro controlador. De este modo, se logra poder enviar los datos ya procesados al ordenador. Como consecuencia de este requisito, el micro controlador elegido tiene que tener una capacidad de cómputo apropiada. Finalmente, los datos procesados por el micro controlador deben enviarse al ordenador con la menor latencia para poder reproducir el movimiento del brazo a tiempo real.

Como segundo objetivo del proyecto, la interfaz tiene que dar la máxima precisión para poder hacer el seguimiento del brazo con la menor desviación posible respecto a la posición de brazo real. Esto significa implementar un algoritmo capaz de usar la información proporcionada por los diferentes sensores usados y eliminar las interferencias externas a la información. Con el fin de llevar a cabo este objetivo, un requisito indispensable es tener en cuenta el número de operaciones aritméticas del algoritmo durante la fase de implementación de este. De no ser así aumentaría la carga computacional del micro controlador y aumentaría la latencia. Por tanto, estaríamos incumpliendo el primer objetivo marcado para el proyecto. Un requisito directo de este segundo objetivo del proyecto es el uso de sensores de sensibilidad configurable. De este modo se consigue optimizar la resolución de los sensores para poder cumplir las especificaciones temporales requeridas.

1.2. Hitos en el proyecto

En un principio, se había pensado hacer uso de placas de sensores que tuvieran un giroscopio y un acelerómetro por cada articulación a monitorizar. A su vez, se había planteado usar el filtro de Kalman extendido [3] en la parte de procesado de señal. Pero los algoritmos que solo están dotados de un giroscopio y un acelerómetro carecen de posicionamiento global. Además, un movimiento rápido hace que el algoritmo pierda el posicionamiento de los ejes. A causa de este hecho, se decidió ampliar el alcance del proyecto incluyendo un nuevo sensor, el magnetómetro. El magnetómetro dota al algoritmo de procesado de la posición global de la placa de sensores respecto a la tierra. Por consecuencia, el algoritmo de procesado había de variar dado que los algoritmos que se habían planteado solo involucraban el giroscopio y el acelerómetro. Se cambió de un algoritmo de unidad de medición inercial o IMU (del inglés: *inertial measurement unit*) a un algoritmo de procesado de señal de sistema de referencia de actitud y rumbo o ARHS (del inglés: *Attitude and heading reference system*). Aún así, también se ha desarrollado un algoritmo IMU para utilizarlo en caso de que los datos del magnetómetro fueran nulos o por si se implementara el sistema con placas de

sensores que no dispongan de magnetómetro. En el segundo apartado de este documento se explica el trasfondo teórico empleado para desarrollar ambos algoritmos.

1.3. Metodología

El procedimiento que se ha seguido durante la elaboración del proyecto ha sido eliminar lo antes posible las múltiples incertidumbres del proyecto para poder dedicarle el máximo tiempo posible a la experimentación. Éstas consistían en cuáles iban a ser los sensores, el micro controlador y el sistema operativo a usar. Finalmente, se optó por unos sensores de Pololu (Altimu-10 v.4), el micro controlador STM32F406, de la compañía STMicroelectronics, con un *SO embedded RTOS*, llamado ChiOS/RT desarrollado por Giovanni Di Sirio y distribuido bajo licencia GPL. Todos ellos comentados y explicados en el apartado de experimentación de este proyecto.

1.4. Dificultades

A lo largo de todo el proyecto se han ido presentando varias dificultades. Por ejemplo, la falta de *stock* hizo que se tuviese que cambiar el modelo de sensor. Por otro lado, tanto el micro procesador como el sistema operativo no tienen una comunidad online muy extensa y ha sido difícil encontrar solución a los problemas. Además, la complejidad de las matemáticas del proyecto ha sido un reto clave. Sin embargo, se agradece la gran ayuda brindada por el tutor de este proyecto frente a todas las dificultades encontradas a lo largo del mismo y el soporte del departamento de ingeniería electrónica por prestar el material para poder llevar a cabo este proyecto.

2 Trasfondo teórico:

2.1. Representación con cuaterniones

Un cuaternión es un número complejo de cuatro dimensiones que puede ser usado para representar la orientación de un cuerpo rígido en un espacio tridimensional. La rotación arbitraria de un objeto en un instante *B* respecto a un instante *A* puede representarse como la rotación del ángulo θ alrededor del eje ${}^{A}\hat{\boldsymbol{r}}$ definido en el instante *A*. Se puede ver la representación gráfica en la Figura 1, dónde los vectores unitarios ortogonales $\hat{\boldsymbol{x}}_{\boldsymbol{A}}$, $\hat{\boldsymbol{y}}_{\boldsymbol{A}}$ y $\hat{\boldsymbol{z}}_{\boldsymbol{A}}$, y $\hat{\boldsymbol{x}}_{\boldsymbol{B}}$, $\hat{\boldsymbol{y}}_{\boldsymbol{B}}$ y $\hat{\boldsymbol{z}}_{\boldsymbol{B}}$ definen los principales ejes en cada uno de los instantes. El cuaternión describe la orientación, ${}^{A}_{B}\hat{\boldsymbol{q}}$, mediante la ecuación (1), dónde r_x, r_y,y r_z definen las componentes unitarias del vector ${}^{A}\hat{\boldsymbol{r}}$ en los ejes x, y, z del instante *A*.

$$ {}^{A}_{B}\hat{\boldsymbol{q}} = [q_1 \;\; q_2 \;\; q_3 \;\; q_4] = \left[\cos\frac{\theta}{2} \quad -r_x \sin\frac{\theta}{2} \quad -r_y \sin\frac{\theta}{2} \quad -r_z \sin\frac{\theta}{2}\right] \qquad (1) $$

La conjugación de un cuaternión denotada por * puede usarse como un intercambio entre los diferentes instantes *A* y *B*. Por ejemplo, ${}^{A}_{B}\hat{\boldsymbol{q}}$ tiene como conjugado ${}^{B}_{A}\hat{\boldsymbol{q}}$ que describe la orientación del instante *B* respecto al instante *A*. La conjugación de un cuaternión está definida por la ecuación (2).

$$ {}^{A}_{B}\widehat{\boldsymbol{q}}^{*} = {}^{B}_{A}\hat{\boldsymbol{q}} = [q_1 \;\; q_2 \;\; q_3 \;\; q_4] \qquad (2) $$

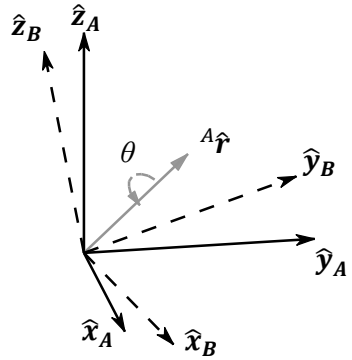

Figura 1: Vectores unitarios del instante A y B

El producto de cuaterniones está denotado por $\otimes$, se utiliza para definir la composición de orientaciones. Por ejemplo, se define ${}^{A}_{B}\hat{\boldsymbol{q}}$ como la orientación del instante *B* respecto al instante *A* y también se define ${}^{B}_{C}\hat{\boldsymbol{q}}$ como la orientación del instante *C* respecto al *B*, es posible demostrar que la orientación ${}^{A}_{C}\hat{\boldsymbol{q}}$ viene definida por la ecuación (3).

$$ {}^{A}_{C}\hat{\boldsymbol{q}} = {}^{B}_{C}\hat{\boldsymbol{q}} \otimes {}^{A}_{B}\hat{\boldsymbol{q}} \qquad (3) $$

Para dos cuaterniones, a y b, el producto viene determinado por el uso de la regla de Hamilton definida en la ecuación (4). El producto entre cuaterniones no es conmutativo, $\boldsymbol{a} \otimes \boldsymbol{b} \neq \boldsymbol{b} \otimes \boldsymbol{a}$

$$ \boldsymbol{a} \otimes \boldsymbol{b} = [a_1 \;\; a_2 \;\; a_3 \;\; a_4] \otimes [b_1 \;\; b_2 \;\; b_3 \;\; b_4] = \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix}^{T} \qquad (4) $$

Un vector tridimensional puede rotarse con un cuaternión usando la relación descrita en la ecuación (5) [1]. ${}^{A}\boldsymbol{v}$ y ${}^{B}\boldsymbol{v}$, son el mismo vector en el instante *A* y en el instante *B* respectivamente. Cada uno de ellos contiene 0 como primer elemento del vector a fin de convertirlo en un vector de cuatro elementos.

$$ {}^{B}\boldsymbol{v} = {}^{A}_{B}\widehat{\boldsymbol{q}} \otimes {}^{A}\boldsymbol{v} \otimes {}^{A}_{B}\widehat{\boldsymbol{q}}^{*} \qquad (5) $$

La rotación de ${}^{A}_{B}\widehat{\boldsymbol{q}}$ puede representarse como la matriz de rotación ${}^{A}_{B}\boldsymbol{R}$ definida por la equación (6)[1].

$$ {}^{A}_{B}\boldsymbol{R} = \begin{bmatrix} 2q_1^2 - 1 + 2q_2^2 & 2(q_2q_3 - q_1q_4) & 2(q_2q_4 - q_1q_3) \\ 2(q_2q_3 - q_1q_4) & 2q_1^2 - 1 + 2q_3^2 & 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_3q_4 - q_1q_2) & 2q_1^2 - 1 + 2q_4^2 \end{bmatrix} \qquad (6) $$

Los ángulos de Euler ψ, θ *y* $\emptyset$ describen la orientación del instante *B* alineada con el instante *A*, ψ alrededor de $\widehat{\boldsymbol{z}}_{\boldsymbol{B}}$, θ alrededor de $\widehat{\boldsymbol{y}}_{\boldsymbol{B}}$ y $\emptyset$ alrededor de $\widehat{\boldsymbol{x}}_{\boldsymbol{B}}$. Para calcular los ángulos de Euler a partir de un cuaternión, ${}^{A}_{B}\widehat{\boldsymbol{q}}$, se utilizan las ecuaciones (7), (8) y (9).

$$ \psi = \text{Atan2}\ (2q_2\, q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1) \qquad (7) $$

$$ \theta = -\sin^{-1}(2q_2q_4 + 2q_1q_3) \qquad (8) $$

$$ \emptyset = \text{Atan2}\ (2q_3\, q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1) \qquad (9) $$

2.2. Filtraje derivativo

2.2.1. Orientación a partir de la velocidad angular

Un giroscopio de tres ejes mide la velocidad angular en cada instante para cada uno de los ejes x, y, z denominados w_x, w_y y w_z respectivamente. Tal como se indica en la ecuación (10) las diferentes velocidades angulares pueden ser introducidas en un vector llamado ${}^{S}\boldsymbol{w}$, así este puede ser usado para el cálculo del cuaternión derivativo que describe el cambio de orientación de la tierra relativo al sensor. Este cuaternión, ${}^{S}_{E}\dot{\boldsymbol{q}}$, puede ser calculado tal y como se indica en la ecuación (11).

$$ {}^{S}\boldsymbol{w} = [0\ \ w_x\ w_y\ \ w_z] \qquad (10) $$

Si se mide la orientación de la tierra relativa al sensor en tiempo, ${}^{S}_{E}\boldsymbol{q}_{w,t}$, se puede computar numéricamente haciendo uso del cuaternion ${}^{S}_{E}\dot{\boldsymbol{q}}_{w,t}$ este representa la velocidad angular que hay que aplicar al cuaternion debido a la información aportada por el giroscopio, que se puede observar en las ecuaciones (12) y (13). Para realizar este cálculo numérico, se tienen que conocer las condiciones iniciales. En este caso se hace uso del cuaternión identidad [1 0 0 0]. En las siguientes ecuaciones se ha discretizado la medición del sensor, por lo tanto, se obtiene ${}^{S}\boldsymbol{w}_t$, que es la velocidad angular del sensor en el instante *t*. Se ha añadido el periodo de muestreo de la velocidad angular Δt y, por último, ${}^{S}_{E}\widehat{\boldsymbol{q}}_{w,t-1}$, es la estimación previa a la estimación actual. El sub-índice *w* sirve para indicar que el cuaternion está calculado desde velocidades angulares.

$$ {}^{S}_{E}\dot{\boldsymbol{q}} = \frac{1}{2}\,{}^{S}_{E}\widehat{\boldsymbol{q}} \otimes {}^{S}\boldsymbol{w} \qquad (11) $$

$$ {}^{S}_{E}\dot{\boldsymbol{q}}_{w,t} = \frac{1}{2}\,{}^{S}_{E}\widehat{\boldsymbol{q}}_{w,t-1} \otimes {}^{S}\boldsymbol{w} \qquad (12) $$

$$ {}^{S}_{E}\boldsymbol{q}_{w,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{w,t-1} + {}^{S}_{E}\dot{\boldsymbol{q}}_{w,t}\Delta t \quad (13) $$

2.2.2. Orientación a partir de los vectores de campos magnético y gravitatorio

Un acelerómetro de tres ejes mide la magnitud y dirección del campo gravitatorio de la tierra conjuntamente con las contribuciones de las aceleraciones aplicadas al sensor. Paralelamente el magnetómetro mide la magnitud y la dirección del campo magnético de la tierra conjuntamente con las distorsiones magnéticas locales. En el contexto de los filtros de orientación, se asume que el acelerómetro medirá solo la gravedad y el magnetómetro medirá solo el campo magnético de la tierra. Si la dirección del campo magnético de la tierra es conocida por el sensor, entonces la medición dada por éste podrá ser relativa al campo magnético de la tierra. Sin embargo, para cualquier medida dada por el sensor no habrá una única solución,sino que hay infinitas soluciones proporcionadas por todas las orientaciones obtenidas por la rotación del sensor alrededor de un eje tangente al campo terrestre. La representación mediante cuaterniones requiere una única solución. Esta solución se estimará mediante la optimización del problema donde la orientación del sensor, ${}^{S}_{E}\hat{\boldsymbol{q}}$, se encuentra como la solución que se alinea con una dirección predefinida del campo terrestre de la tierra, ${}^{E}\hat{\boldsymbol{d}}$, conjuntamente a la medida del campo en el sensor, ${}^{S}\hat{\boldsymbol{s}}$, utilizando la operación de rotación descrita en la ecuación (5). La estimación está definida en la ecuación (14) y la función en la ecuación (15).

$$ \min_{{}^{S}_{E}\hat{\boldsymbol{q}} \in R^4} \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) \quad (14) $$

$$ \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) = {}^{S}_{E}\widehat{\boldsymbol{q}^{*}} \otimes {}^{E}\hat{\boldsymbol{d}} \otimes {}^{S}_{E}\hat{\boldsymbol{q}} - {}^{S}\hat{\boldsymbol{s}} \quad (15) $$

$$ {}^{S}_{E}\hat{\boldsymbol{q}} = [q_1 \;\; q_2 \;\; q_3 \;\; q_4] \quad (16) $$

$$ {}^{E}\hat{\boldsymbol{d}} = [0 \;\; d_x \;\; d_y \;\; d_z] \quad (17) $$

$$ {}^{S}\hat{\boldsymbol{s}} = [0 \;\; s_x \;\; s_y \;\; s_z] \quad (18) $$

Existen muchas posibilidades de optimización de algoritmos. En este trabajo se va a hacer uso del algoritmo de gradiente descendiente, dado que es un algoritmo simple en lo que a implementación se refiere y la carga computacional es menor que la que generan otros algoritmos. Además, respecto al filtro de Kalman tiene la ventaja que no se debe conocer previamente la estadística del señal y no tiene una gran dependencia de las condiciones iniciales. En la ecuación (19) está descrito el algoritmo de gradiente descendiente para "*n*" iteraciones. El resultado que proporciona es la predicción del próximo cuaternión, ${}^{S}_{E}\hat{\boldsymbol{q}}_{n+1}$, basado en unas condiciones iniciales ${}^{S}_{E}\hat{\boldsymbol{q}}_{0}$ y la variable de paso del algoritmo μ. En la ecuación (20) se puede ver computado el error de dirección sobre la superficie definida por la función objetivo, $\boldsymbol{f}$, y su Jacobiano, $\boldsymbol{J}$.

$$ {}^{S}_{E}\hat{\boldsymbol{q}}_{k+1} = {}^{S}_{E}\hat{\boldsymbol{q}}_{k} - \mu \frac{\nabla \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right)}{\left\|\nabla \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right)\right\|}, k = 0, 1, 2 \ldots n \quad (19) $$

$$ \nabla \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) = \boldsymbol{J}^{T}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}\right)\boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) \quad (20) $$

$$ \boldsymbol{f}\left({}^{S}_{E}\hat{\boldsymbol{q}}_{k}, {}^{E}\hat{\boldsymbol{d}}, {}^{S}\hat{\boldsymbol{s}}\right) = \begin{bmatrix} 2d_x\left(\frac{1}{2} - q_3^2 - q_4^2\right) + 2d_y(q_1q_4 - q_2q_3) + 2d_z(q_2q_4 - q_1q_3) - s_x \\ 2d_x(q_2q_3 - q_1q_4) + 2d_y\left(\frac{1}{2} - q_2^2 - q_4^2\right) + 2d_z(q_1q_2 - q_3q_4) - s_y \\ 2d_x(q_1q_3 - q_2q_4) + 2d_y(q_3q_4 - q_1q_2) + 2d_z\left(\frac{1}{2} - q_2^2 - q_3^2\right) - s_z \end{bmatrix} (21) $$

$$\boldsymbol{J}\left({}^{S}_{E}\widehat{\boldsymbol{q}}_k, {}^{E}\widehat{\boldsymbol{d}}\right) = \begin{bmatrix} 2d_y q_4 - 2d_z q_3 & 2d_y q_3 + 2d_z q_4 & -4d_x q_3 + 2d_y q_2 - 2d_z q_1 & -4d_x q_4 + 2d_y q_1 + 2d_z q_2 \\ -2d_x q_4 + 2d_z q_2 & 2d_x q_3 - 4d_y q_2 + 2d_z q_1 & 2d_x q_2 + 2d_z q_4 & -2d_x q_1 - 4d_y q_4 + 2d_z q_3 \\ 2d_x q_3 - 2d_y q_2 & 2d_x q_4 - 2d_y q_1 - 4d_z q_2 & 2d_x q_1 + 2d_y q_4 - 4d_z q_3 & 2d_x q_2 + 2d_y q_3 \end{bmatrix} \quad (22)$$

Respecto a la explicación se ha cambiado la variable de iteración "*n*" por la variable "*k*" en las ecuaciones (19) y (20) para poder preservar la notación estándar.

Las ecuaciones (19) y (20) describen la forma general del algoritmo aplicable al campo predefinido en cualquier dirección. Sin embargo, si la dirección de referencia del campo se la define con solo 1 o 2 componentes de los ejes de la tierra, la ecuación se simplifica significativamente. Un convenio apropiado seria asumir que la gravedad de la tierra está en el eje "z*"* como muestra la ecuación (23). Substituyendo ${}^{E}\widehat{\boldsymbol{g}}$ y normalizando la medida proporcionada por el acelerómetro, ${}^{S}\widehat{\boldsymbol{a}}$, para ${}^{E}\widehat{\boldsymbol{d}}$ y ${}^{S}\widehat{\boldsymbol{s}}$, respectivamente en las ecuaciones (21) y (22). De esta forma, se obtiene el objetivo propuesto de simplificar la función y el Jacobiano para poder simplificar el problema y a su vez la carga computacional. La función simplificada y el Jacobiano están descritas en las ecuaciones (25) y (26) respectivamente.

$${}^{E}\widehat{\boldsymbol{g}} = [0 \;\; 0 \;\; 0 \;\; 1] \quad (23)$$

$${}^{S}\widehat{\boldsymbol{a}} = [0 \;\; a_x \;\; a_y \;\; a_z] \quad (24)$$

$$\boldsymbol{f_g}\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{S}\widehat{\boldsymbol{a}}\right) = \begin{bmatrix} 2(q_2 q_4 - q_1 q_3) - a_x \\ 2(q_1 q_2 - q_3 q_4) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix} \quad (25)$$

$$\boldsymbol{J_g}\left({}^{S}_{E}\widehat{\boldsymbol{q}}\right) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \quad (26)$$

Por lo que respecta al campo magnético de la tierra, se puede considerar que tiene dos componentes una componente horizontal y otra componente vertical. La componente vertical respecto a la horizontal forman el ángulo de inclinación del campo magnético (en España estaría entre de 55º y 60º [5]). El campo magnético vendría representado por la ecuación (27). Sustituyendo ${}^{E}\widehat{\boldsymbol{b}}$ y normalizando la medición proporcionada por el magnetómetro del sensor ${}^{S}\widehat{\boldsymbol{m}}$ para ${}^{E}\widehat{\boldsymbol{d}}$ y ${}^{S}\widehat{\boldsymbol{s}}$ respectivamente en las ecuaciones (21) y (22), igual que se ha hecho para la medida proporcionada por el acelerómetro, se obtienen las ecuaciones (29) y (30)

$${}^{E}\widehat{\boldsymbol{b}} = [0 \;\; b_x \;\; 0 \;\; b_z] \quad (27)$$

$${}^{S}\widehat{\boldsymbol{m}} = [0 \;\; m_x \;\; m_y \;\; m_z] \quad (28)$$

$$\boldsymbol{f_b}\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{E}\widehat{\boldsymbol{b}}, {}^{S}\widehat{\boldsymbol{m}}\right) = \begin{bmatrix} 2b_x\left(\frac{1}{2} - q_3^2 - q_4^2\right) + 2b_z(q_2 q_4 - q_1 q_3) - m_x \\ 2b_x(q_2 q_3 - q_1 q_4) + 2b_z(q_1 q_2 - q_3 q_4) - m_y \\ 2b_x(q_1 q_3 - q_2 q_4) + 2b_z\left(\frac{1}{2} - q_2^2 - q_3^2\right) - m_z \end{bmatrix} \quad (29)$$

$$\boldsymbol{J_b}\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{E}\widehat{\boldsymbol{b}}\right) = \begin{bmatrix} -2b_z q_3 & 2b_z q_4 & -4b_x q_3 - 2b_z q_1 & -4b_x q_4 + 2b_z q_2 \\ -2b_x q_4 + 2b_z q_2 & 2b_x q_3 + 2b_z q_1 & 2b_x q_2 + 2b_z q_4 & -2b_x q_1 + 2b_z q_3 \\ 2b_x q_3 & 2b_x q_4 - 4b_z q_2 & 2b_x q_1 - 4b_z q_3 & 2b_x q_2 \end{bmatrix} \quad (30)$$

Tal y como se ha expuesto anteriormente, la medida del campo magnético por sí sola no provee solo una única orientación del sensor. Para conseguir obtener solo una única orientación, la medida y dirección proporcionada por el campo magnético y por el campo gravitatorio deben ser combinadas. Para combinarlas, se debe declarar la función definida en la ecuación (31) y (32). La solución que proporcionada por la ecuación (25) y (29) está formada por una superficie donde el mínimo está proporcionado por una línea, mientras que la solución provista por la ecuación (31) tiene el mínimo localizado en un solo punto particularizado por $b_x \neq 0$

$$\boldsymbol{f}_{g,b}\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{S}\boldsymbol{a}, \widehat{{}^{E}\boldsymbol{b}, {}^{S}\boldsymbol{m}}\right) = \begin{bmatrix} \boldsymbol{f}_g\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{S}\widehat{\boldsymbol{a}}\right) \\ \boldsymbol{f}_b\left({}^{S}_{E}\widehat{\boldsymbol{q}}_k, {}^{E}\widehat{\boldsymbol{b}}, {}^{S}\widehat{\boldsymbol{m}}\right) \end{bmatrix} \quad (31)$$

$$\boldsymbol{J}_{g,b}\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{E}\widehat{\boldsymbol{b}}\right) = \begin{bmatrix} \boldsymbol{J}_g\left({}^{S}_{E}\widehat{\boldsymbol{q}}\right) \\ \boldsymbol{J}_b\left({}^{S}_{E}\widehat{\boldsymbol{q}}, {}^{E}\widehat{\boldsymbol{b}}\right) \end{bmatrix} \quad (32)$$

Con el fin de poder obtener una optimización ajustada al óptimo, se debería iterar múltiples veces la ecuación (19) para cada nueva orientación y sus correspondientes medidas proporcionadas por el sensor. Esto provocaría una latencia, que incumpliría uno de los requisitos del proyecto. Por tanto se ha de hacer uso de una variable paso μ muy ajustada para poder ajustar cada iteración al valor óptimo. Normalmente se obtiene utilizando la segunda derivada de la función objetivo, es decir se debería efectuar el Hessiano de la función. En este caso sería la función representada en la ecuación (31). Sin embargo, crear un algoritmo que hiciese eso no sería eficiente y no se podría cumplir el primer objetivo marcado en el proyecto: la medición del movimiento del brazo debe ser a tiempo real con la mínima latencia posible. El cálculo del Hessiano ocasionaría una gran carga computacional al micro controlador. Se podría minimizar este impacto haciendo que el ordenador ejecutara el filtro, pero se estaría incumpliendo el requisito de que la interfaz efectuase todos los cálculos con el fin de dotar al ordenador de los datos ya filtrados. Teniendo en cuenta lo anteriormente mencionado, se ha de considerar si computar solo una iteración del algoritmo por muestra proporcionada por el sensor será suficiente. Esta consideración se puede realizar siempre y cuando se tenga en cuenta que la tasa de convergencia proporcionada por μ_t es igual o mayor a la velocidad física del cambio de orientación, de no ser así el algoritmo no convergiría.

Con la ecuación (33) se puede calcular la orientación estimada ${}^{S}_{E}\boldsymbol{q}_{\nabla,t}$ en el instante t, basándose en la anterior estimación ${}^{S}_{E}\boldsymbol{q}_{\mathrm{est},t-1}$ y en el gradiente descendiente de la función $\nabla\boldsymbol{f}$ definida por el acelerómetro y el magnetómetro en el mismo instante, ${}^{S}\widehat{\boldsymbol{a}}_{\boldsymbol{t}}$ y ${}^{S}\widehat{\boldsymbol{m}}_{\boldsymbol{t}}$ respectivamente. Llegados a este punto, se puede definir la función $\nabla\boldsymbol{f}$ en función de los sensores utilizados, tal y como muestra la ecuación (34). La primera ecuación solo tiene la contribución del acelerómetro y la segunda tiene la contribución del acelerómetro y el magnetómetro. Estas dos opciones se han obtenido de la particularización de la ecuación (20) con las funciones y sus respectivo Jacobiano obtenidos en las ecuaciones (25) y (26) o (31) y (32).

$${}^{S}_{E}\boldsymbol{q}_{\nabla,t} = {}^{S}_{E}\boldsymbol{q}_{\mathrm{est},t-1} - \mu_t \frac{\nabla f}{\|\nabla f\|} \quad (33)$$

$$\nabla \boldsymbol{f} = \begin{cases} \boldsymbol{J}_{\boldsymbol{g}}^{T}\left({}_{E}^{S}\boldsymbol{q}_{\nabla,t-1}\right)\boldsymbol{f}_{\boldsymbol{g}}\left({}_{E}^{S}\boldsymbol{q}_{\nabla,t-1}, {}^{S}\widehat{\boldsymbol{a}}_{\boldsymbol{t}}\right) \\ \boldsymbol{J}_{\boldsymbol{g},\boldsymbol{b}}\left({}_{E}^{S}\boldsymbol{q}_{\nabla,t-1}, {}^{E}\widehat{\boldsymbol{b}}\right)\boldsymbol{f}_{\boldsymbol{g},\boldsymbol{b}}\left({}_{E}^{S}\boldsymbol{q}_{\nabla,t-1}, {}^{S}\widehat{\boldsymbol{a}}_{t}, {}^{E}\widehat{\boldsymbol{b}}, {}^{S}\widehat{\boldsymbol{m}}_{t}\right) \end{cases} \quad (34)$$

Para encontrar un valor óptimo de μ_t se debería asegurar la convergencia de ${}_{E}^{S}\boldsymbol{q}_{\nabla,t}$, la cual está limitada por la velocidad de giro física. Un posible cálculo de μ_t es el muestrado en la ecuación (35), donde Δt es el periodo de muestreo, ${}_{E}^{S}\dot{\boldsymbol{q}}_{w,t}$ es la velocidad física medida por el giroscopio y α es una ganancia para paliar el ruido generado por las medidas del acelerómetro y del magnetómetro.

$$\mu_t = \alpha \left\| {}_{E}^{S}\dot{\boldsymbol{q}}_{w,t} \right\| \Delta t, \ \alpha > 1 \quad (35)$$

Llegados a este punto, se puede observar cómo se han obtenido dos posibles implementaciones del filtro, una que hace uso de la información que proviene del giroscopio, compensada con la información que proviene del acelerómetro IMU, y otra que también hace uso de la información del giroscopio pero no solo utiliza la información que proviene del acelerómetro, sino que también la que proviene del magnetómetro AHRS.

2.3. Algoritmo final

La estimación de la orientación en la que se encuentra el sensor se obtiene mediante la fusión de las orientaciones anteriormente calculadas ${}_{E}^{S}\boldsymbol{q}_{w,t}$ y ${}_{E}^{S}\boldsymbol{q}_{\nabla,t}$ a partir de las ecuaciones (13) y (33), respectivamente y proporcionadas por diferentes sensores. Esta fusión está descrita en la ecuación (36) donde γ_t es la ponderación que aplicamos a cada una de las orientaciones medidas.

$${}_{E}^{S}\boldsymbol{q}_{\text{est}} = \gamma_t {}_{E}^{S}\boldsymbol{q}_{w,t} + (1-\gamma_t){}_{E}^{S}\boldsymbol{q}_{\nabla,t}, \ 0 \leq \gamma_t \leq 1 \quad (36)$$

También hay que decidir cómo ponderar los diferentes estimadores. Para ello, hay que encontrar una forma óptima de calcular γ_t. Para lograrlo se define γ_t cómo la variable que debe asegurar que la divergencia de cada una de las estimaciones es la misma. Se declara β como la divergencia de ${}_{E}^{S}\boldsymbol{q}_{w}$ calculada como la magnitud del error de medida del cuaternión que corresponde al giróscopo y la divergencia de ${}_{E}^{S}\boldsymbol{q}_{\nabla}$ es $\frac{\mu_t}{\Delta t}$. La igualdad se plasma en la ecuación (37) y se puede reordenar como en la ecuación (38).

$$(1-\gamma_t)\beta = \gamma_t \frac{\mu_t}{\Delta t} \quad (37)$$

$$\gamma_t = \frac{\beta}{\beta + \frac{\mu_t}{\Delta t}} \quad (38)$$

Las ecuaciones (36) y (38) aseguran la fusión óptima de ${}_{E}^{S}\boldsymbol{q}_{w,t}$ y ${}_{E}^{S}\boldsymbol{q}_{\nabla,t}$ asumiendo que la velocidad de convergencia de ${}_{E}^{S}\boldsymbol{q}_{\nabla}$ controlada por α es igual o mayor a la velocidad de cambio de orientación. Si se asume que α tiene un valor muy grande entonces μ_t también se convierte en un valor muy grande, tal y como se puede observar mediante la ecuación (35). En el momento que se asume que μ_t tiene un valor muy grande, las ecuaciones del filtro de orientación se simplifican y en la ecuación (33), ${}_{E}^{S}\boldsymbol{q}_{\text{est},t-1}$ se vuelve insignificante ante el valor de μ_t. Esta ecuación queda simplificada como la ecuación (39) recoge.

$${}_{E}^{S}\boldsymbol{q}_{\nabla,t} \approx -\mu_t \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \quad (39)$$

Tal y como se ha definido γ_t en la ecuación (38), se puede rescribir teniendo en cuenta que el factor $\frac{\mu_t}{\Delta t}$ es mayor que β del denominador, por tanto β también se vuelve insignificante y la ecuación puede ser reescrita como muestra la ecuación (40), asumiendo que $\gamma_t \approx 0$ dado que μ_t es muy grande.

$$\gamma_t \approx \beta \frac{\Delta t}{\mu_t} \qquad (40)$$

Sustituyendo las ecuaciones (13), (39) y (40) en la ecuación (36) se obtiene la ecuación (41). Cabe destacar que las dos γ_t de la ecuación (36) han sido sustituidas por la ecuación (40) y 0 respectivamente. Esto se debe a que tal y como se ha comentado en la ecuación (40) γ_t tiende a 0 debido al gran valor de μ_t. Como la componente proporcionada por el giroscopio no se ve afectada por el valor de μ_t, $\gamma_t = 0$, pero en la otra parte de la ecuación vemos que μ_t puede ser simplificada por tanto se substituye por la ecuación (40).

$$ {}^S_E\boldsymbol{q}_{\mathrm{est,t}} = \beta \frac{\Delta t}{\mu_t}\left(-\mu_t \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|}\right) + (1-0)({}^S_E\widehat{\boldsymbol{q}}_{w,t-1} + {}^S_E\dot{\boldsymbol{q}}_{w,t}\Delta t) \qquad (41)$$

La ecuación (41) puede ser simplificada hasta llegar a la ecuación (42) donde ${}^S_E\dot{\boldsymbol{q}}_{est,t}$ es el estimador de la velocidad de cambio de orientación definido en la ecuación (43) y ${}^S_E\dot{\boldsymbol{q}}_{\epsilon,t}$ es la dirección del error de ${}^S_E\dot{\boldsymbol{q}}_{est,t}$ definido en la ecuación (44).

$$ {}^S_E\boldsymbol{q}_{\mathrm{est,t}} = {}^S_E\widehat{\boldsymbol{q}}_{\mathrm{est,t-1}} + {}^S_E\dot{\boldsymbol{q}}_{est,t}\Delta t \qquad (42)$$

$$ {}^S_E\dot{\boldsymbol{q}}_{est,t} = {}^S_E\dot{\boldsymbol{q}}_{w,t} - \beta\, {}^S_E\dot{\boldsymbol{q}}_{\epsilon,t} \qquad (43)$$

$$ {}^S_E\dot{\boldsymbol{q}}_{\epsilon,t} = \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \qquad (44)$$

Tal y como se puede ver observando las ecuaciones (42), (43) y (44) el filtro calcula la orientación ${}^S_E\boldsymbol{q}_{\mathrm{est}}$ integrando la orientación estimada ${}^S_E\dot{\boldsymbol{q}}_{est}$. El filtro computa ${}^S_E\dot{\boldsymbol{q}}_{est}$ como la velocidad del cambio de orientación medida por el giroscopio, ${}^S_E\dot{\boldsymbol{q}}_{w}$, conjuntamente con la magnitud del error de medida, β, eliminando la componente de la dirección del error, ${}^S_E\dot{\boldsymbol{q}}_{\epsilon}$, computada gracias a la medición de los sensores de orientación, acelerómetro y magnetómetro.

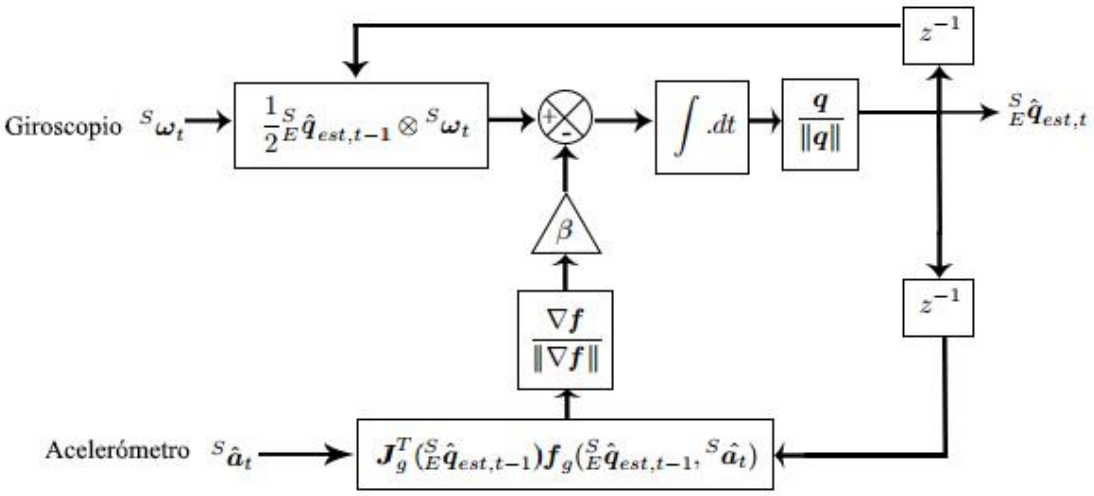

Figura 2: Diagrama de bloques del algoritmo IMU

2.4. Compensación de la distorsión magnética

El magnetómetro es uno de los sensores que contienen una mayor distorsión debido a la presencia de materiales ferromagnéticos en los alrededores del sensor. Para intentar paliar este efecto se puede modelar de una forma diferente al modelo presentado en la ecuación (27) presentada en el apartado 2.2. Esta compensación solo está presente en el algoritmo AHRS dado que el algoritmo IMU no hace uso del magnetómetro.

Para poder compensar la distorsión magnética se debe computar la dirección del campo magnético de la tierra en cada instante "*t*", ${}^{E}\widehat{\boldsymbol{h}}_t$, como la medida normalizada del magnetómetro, ${}^{S}\widehat{\boldsymbol{m}}_t$, rotada por la orientación estimada en el instante anterior, ${}^{S}_{E}\widehat{\boldsymbol{q}}_{\mathrm{est,t-1}}$, tal y como se describe en la ecuación (45). Ahora se puede cambiar el vector de referencia del campo de la tierra, ${}^{E}\widehat{\boldsymbol{b}}$, con el fin de poder corregir el efecto del error de inclinación de la medición del campo magnético de la tierra. Tal y como se describe en la ecuación (46). Llegado a este punto el vector de referencia del campo magnético de la tierra sigue teniendo dos componente pero ahora no solo se tiene en cuenta la componente "*x*" para calcular la componente horizontal del vector sino que se computa con la contribución de "*x*" y "*y*" creando así el plano horizontal tangente a la tierra.

$$ {}^{E}\widehat{\boldsymbol{h}}_t = [0 \;\; h_x \;\; h_y \;\; h_z] = {}^{S}_{E}\widehat{\boldsymbol{q}}_{\mathrm{est,t-1}} \otimes {}^{S}\widehat{\boldsymbol{m}}_t \otimes {}^{S}_{E}\widehat{\boldsymbol{q}}^{*}_{\mathrm{est,t-1}} \qquad (45) $$

$$ {}^{E}\widehat{\boldsymbol{b}} = [0 \;\; b_x \;\; 0 \;\; b_z] = [0 \;\; \sqrt{h_x^2 + h_y^2} \;\; 0 \;\; h_z] \qquad (46) $$

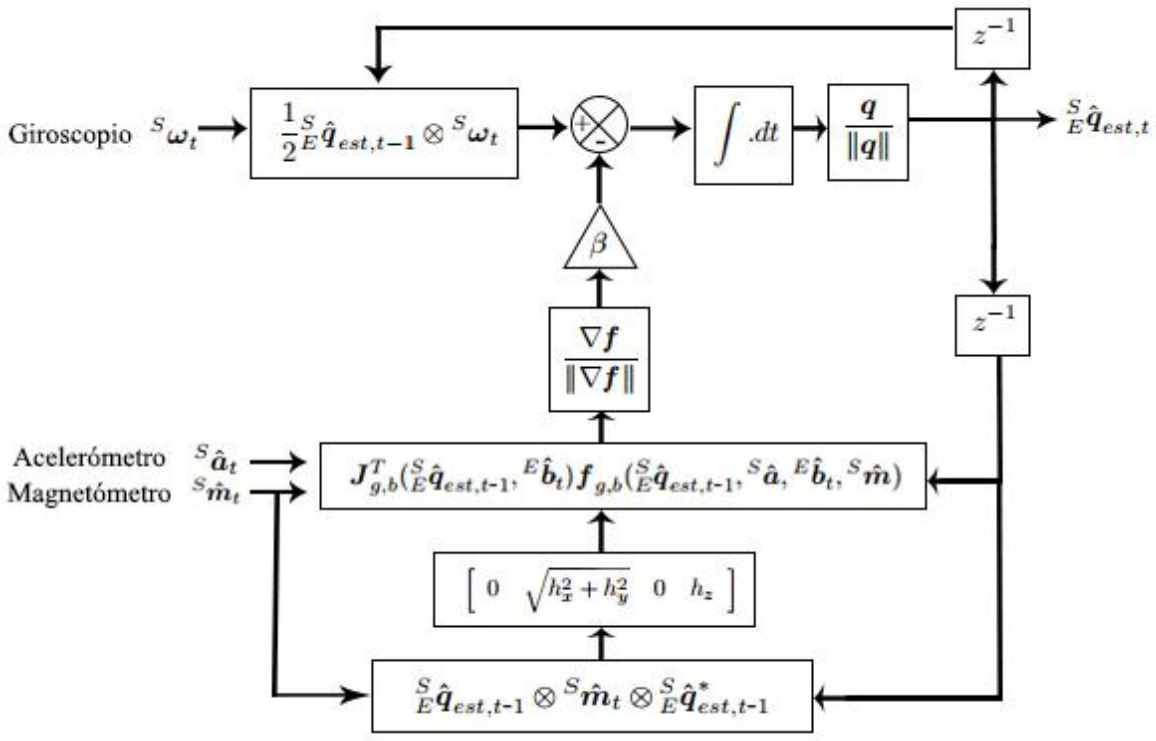

Figura 3: Diagrama de bloques del algoritmo AHRS con la compensación de distorsión magnética

2.5. Ganancia del filtro

La ganancia del filtro β es un parámetro ajustable que representa el error de medida del giroscopio y se expresa como la magnitud del cuaternión. Es conveniente definir β utilizando el error máximo de medida del giroscopio. Utilizando la relación descrita en la ecuación (11), β quedaría definida por la ecuación (47) donde $\widehat{\boldsymbol{q}}$ es cualquier cuaternión unitario.

$$\beta = \left\| \frac{1}{2} \widehat{\boldsymbol{q}} \otimes [0 \;\; \tilde{w}_{max} \;\; \tilde{w}_{max} \;\; \tilde{w}_{max}] \right\| = \sqrt{\frac{3}{4}} \tilde{w}_{max} \qquad (47)$$

3 Experimentación:

3.1. Comunicación I2C

En el proyecto se emplea un bus serie I2C para comunicar los sensores con el micro controlador. La comunicación I2C [6] utiliza dos cables: datos (SDA) y reloj (SCL). Todos los dispositivos están conectados con los mismos cables y todos pueden recibir y transmitir mediante estos cables. El bus permite que coexistan dispositivos maestros y esclavos, los maestros son los que generan en el bus SCL la señal de reloj y también son los que inician la transmisión usando el bus de datos. Los otros dispositivos son esclavos y se encargan de responder a las peticiones del dispositivo maestro. Normalmente cada línea I2C tiene un solo maestro que controla a los esclavos, pero puede haber más de dispositivo maestro en una misma línea I2C, para ello el protocolo I2C utiliza un procedimiento de arbitración que en la circunstancia en que 2 maestros hagan una petición simultánea, determina cuál de los dos maestros adquiere la oportunidad de transmitir por el bus y puede continuar con su petición. En este proyecto solo se va a hacer uso un maestro que será el micro controlador STM32F4 y nueve sensores que serán los diferentes esclavos, tres acelerómetros, tres giroscopios y tres magnetómetros.

3.1.1. Protocolo de transmisión

Por cada pulso de reloj en la línea SCL se transmite un bit por la línea SDA. La línea SDA solo puede cambiar de estado cuando la señal de reloj está a nivel bajo, mientras la señal de reloj permanezca en estado alto la línea SDA debe permanecer estable.

Tal y como se ha mencionado en el apartado anterior el dispositivo maestro es el que inicia y finaliza la transmisión. Por tanto el estándar I2C tiene una condición de inició y una condición de finalización. Tanto para iniciar como para finalizar la transmisión el estado de la línea SCL debe estar en alto y dependiendo de la transición que efectué la línea SDA significa que la transmisión empieza o termina, tal y como se puede observar en la Figura 2, el cambio de estado alto a bajo significa el inicio de la transmisión y el estado inverso, de bajo a alto significa el final de la transmisión. Una vez un maestro inicia una transmisión se considera que el estado de la línea es ocupado y no puede ser ocupado por otro maestro.

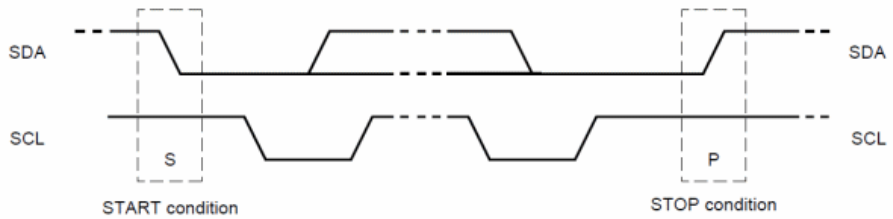

Figura 4: Condición de inicio y finalización del bus I2C [6]

3.1.2. Transmitiendo datos

El estándar I2C utiliza paquetes de 8-bits (Byte). No tiene ninguna limitación del número de bytes que se pueden transmitir sin embargo todos los bytes tienen que ir seguidos de un bit de *Acknowledge* (ACK) este bit de señalización también indica cuando el dispositivo está dispuesto a recibir el próximo byte. Para todos los bits es el dispositivo maestro el que genera los impulsos de reloj. Si un dispositivo esclavo no envía un ACK significa que no ha de transmitir más datos o que el dispositivo aún no está listo para transmitir. En estos casos el maestro debe finalizar la conexión y volver a realizar la inicialización.

3.1.3. Direcciones de 7-bits

El estándar original I2C utiliza direcciones de 7 bits y esta debe ser única entre todos los dispositivos conectados a la misma línea I2C. Tal y como se ha explicado anteriormente el protocolo se inicia con la condición de inicio por parte del dispositivo maestro y luego le sigue los 7 bits de direccionamiento junto con un bit que indica la dirección de la comunicación, es decir si el maestro quiere escribir o leer de la línea. Si es un 0 el maestro escribe al dispositivo esclavo que tiene la dirección de 7 bits, por otro lado si es un 1 el maestro leerá del dispositivo. Después, el dispositivo esclavo en cuestión debe responder con un ACK y empezar a enviar o recibir datos.

3.1.4. Resistencias Pull-up

Las dos señales que viajan haciendo uso de los dos buses (SCL y SDA) son bidireccionales. Deben estar conectados con resistencias *pull-up*, esto significa que si la línea está libre permanece en estado alto. Todos los dispositivos conectados a SDA o SCL han de tener salida en drenador (o colector) abierto de manera que únicamente puede forzar el nivel bajo. El número de dispositivos que cuelgan de una sola línea es ilimitado pero hay que tener en cuentra que la capacidad del bus no puede exceder los 400pF .

El estándar I2C puede funcionar en diferentes modos, los principales son: modo estándar que puede transmitir hasta 100kbits/s y el modo rápido que puede transmitir hasta 400kbits/s. Como nosotros necesitamos el máximo ancho de banda posible se tuvo en consideración que los sensores pudieran utilizar el modo de alta velocidad a 400kbits/s.

Para que el protocolo I2C funcione en modo rápido es necesario tener en consideración las resistencias *pull-up* y la capacidad total de la línea. La capacidad de la línea condiciona la resistencia máxima, tal y como se puede ver en las formulas (48) y (49). Considerando los limites $V_{IH} = 0.7V_{DD}$ y $V_{IL} = 0.3V_{DD}$, la constante $R_P C_b$ formada por la resistencia *pull-up* y la capacidad del cable, y por último la ecuación de carga de un condensador $V(t) = V_{DD}(1 - e^{\frac{-t}{RC}})$. Resolviendo las ecuaciones y encontrando cual sería el tiempo de cambio de estado $t_r = t_2 - t_1$ (*rise-time*) se obtiene la ecuación (50) que nos muestra cual es la resistencia máxima en función de C_b y t_r

$$V(t_1) = 0.3V_{DD} = V_{DD}(1 - e^{\frac{-t_1}{RC}}) \qquad (48)$$

$$V(t_2) = 0.7V_{DD} = V_{DD}(1 - e^{\frac{-t_2}{RC}}) \qquad (49)$$

$$R_{P(MAX)} = \frac{t_r}{0.8473 * C_b} \qquad (50)$$

Revisando las especificaciones del bus I2C [10] se observa que el $t_{r(MAX)} = 300ns$ por tanto la relación entre R_P y C_b quedaría como la ecuación (51). Al final del apartado **3.2.1** aplicaremos estas ecuaciones para calcular cuál es la capacidad máxima de la línea que podemos soportar.

$$R_{P(MAX)} = \frac{354*10^{-9}}{C_b} \quad (51)$$

3.2. Sensores

Los requisitos marcados para las placas de sensores fueron que pudiesen transmitir con el estándar I2C en modo rápido, que estuviesen integrados y que tuvieran el mínimo tamaño posible. También debían contener un giroscopio, acelerómetro y magnetómetro, este último se añadió después de haber realizado el estudio matemático y ver que con un filtro IMU no era suficiente para monitorizar el brazo humano con la precisión necesaria. Finalmente la elección fue MinIMU-9 del fabricante Pololu, debido a una falta de stock que tenía el proveedor se decidió adquirir el modelo superior que son los Atimu-10. Estos incluyen, además, un sensor de medición de presión atmosférica, pero para este proyecto no se le da uso.

3.2.1. Altimu-10 v.4

Esta placa está dotada de tres sensores: L3GD20H un giroscopio de tres ejes, LSM303D es un acelerómetro y magnetómetro de tres ejes y por último el sensor LPS25H que es un barómetro pero que no se va a usar en este proyecto. De ahora en adelante entenderemos por sensor al acelerómetro, giroscopio y magnetómetro y por placa de sensores a cada AltIMU-10.

Cada uno de los sensores puede configurarse para que aumente o disminuya la precisión con la que mide. Todos ellos proporcionan 16 bits (2 Bytes) de resolución por eje. En la tabla 3 de los anexos se puede ver los distintos rangos de medida a los cuales se les pueden configurar. La resolución del sensor vendrá dada por el rango elegido dividido entre 2^{16}. El ancho de banda de los sensores no es un factor limitante dado que pueden llegar a alcanzar 800Hz. Los sensores están preparados para poder utilizar la comunicación I2C en modo de alta velocidad.

Dado que para este proyecto es necesario hacer uso de más de una placa de sensores, es importante que puedan coexistir varios sensores en un mismo bus I2C. Para ello se emplea un bit de selección SA0 que permite elegir la dirección de cada sensor entre dos posibilidades. De esta manera podemos evitar que colisiones las direcciones de dos sensores iguales dentro de un mismo bus. En la tabla 1 se puede ver cuáles son las direcciones de los sensores en función del valor de SA0. Aun teniendo este pin lógico, como un brazo humano tiene tres articulaciones que monitorizar, se debe tener un sensor por articulación. Tal y como se ha explicado en un apartado anterior, no hay un número limitado de dispositivos esclavos que puedan colgar de una línea I2C pero todos ellos deben de tener diferentes direcciones, en este caso 2 placas coincidirían con las mismas direcciones de 7 bits. Así que se ha tenido que hacer uso de una segunda línea I2C para poder tener acceder a los tres sensores.

Sensor	Dirección (SA0 = HIGH)	Dirección (SA0 = LOW)
L3GD20H (giroscopio)	1101011b	1101010b
LSM303D (acelerómetro y magnetómetro)	0011101b	0011110b

Tabla 1: Direcciones de los sensores en función del estado de SA0

Todas las líneas SDA, SCL y SA0 del AltIMU-10 están por defecto conectadas a la tensión de alimentación mediante una resistencia *pull-up* de $10K\Omega$. Ello significa que la placa está preparada para funcionar por defecto en modo estándar. Para poder trabajar en modo rápido se ha tenido que poner entre la tensión de alimentación y las líneas SDA y SCL otra resistencia *pull-up* de $10K\Omega$ para así hacer que la resistencia Rp de la línea fuese $5K\Omega$. Esta modificación solo se ha efectuado en la línea I2C con un solo AltIMU-10. La línea con dos placas de sensores ya tenía en paralelo las dos resistencias internas de $10K\Omega$ de las placas. Si calculamos con la ecuación (51) cual es la capacidad máxima para poder transmitir a 400KHz obtenemos un máximo de 70,8pF por línea I2C.

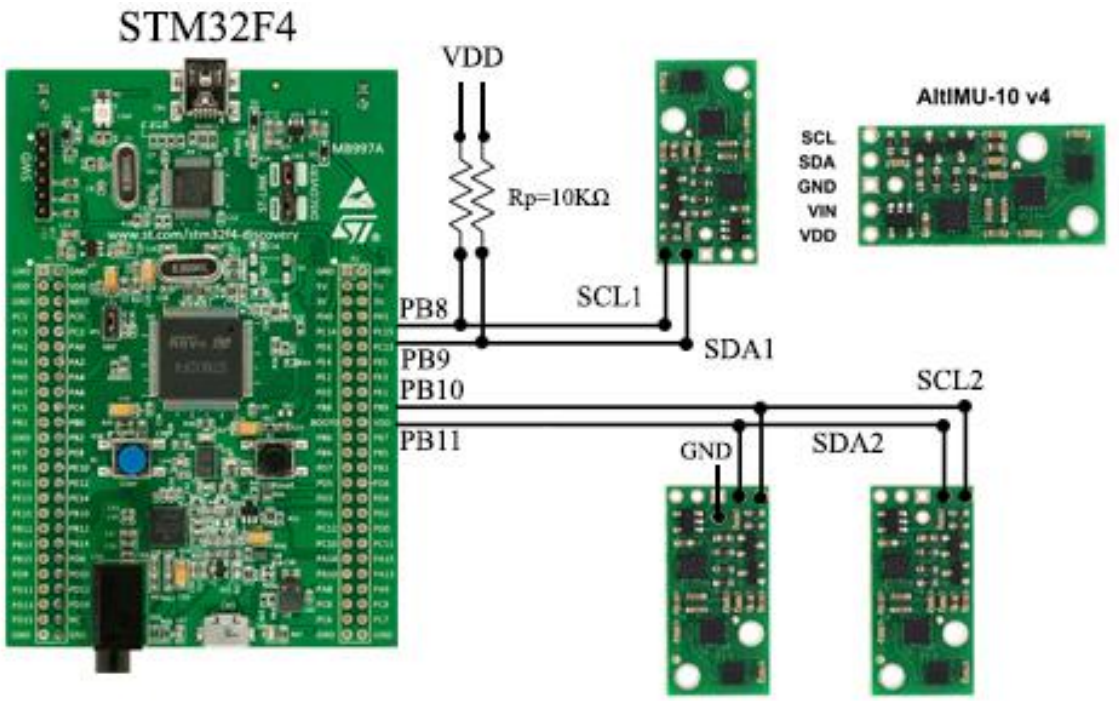

Figura 5: Configuración del hardware, la placa STM32F4 a la izquierda y las placas de los sensores a la derecha

3.3. Microprocesador

3.3.1. STM32F4

La STM32F4 fabricada por STMicroelectronics, es una de las placas más potente de la familia Discovery. El núcleo de la placa STM32F4 es un micro controlador STM32F407VGT6 con un encapsulado de 100pines que incluye 1Mbyte de memoria flash y 192 Kbytes de memoria RAM y que es capaz de operar con frecuencias de reloj de hasta 192MHz. El núcleo del micro controlador posee arquitectura ARM Cortex-M4 de 32 bits. A destacar entre todos los periféricos internos que ofrece el micro controlador, los 3 bloques de comunicación síncrona I2C, de los cuales por limitaciones de sistema operativo empleado sólo son accesible 2 de ellos en paralelo. Sería posible utilizar los tres I2C sin necesidad de utilizar un sistema operativo, pero es preferible trabajar con el sistema operativo porque ofrece los drivers de control y de comunicación ya implementados.

El entorno de trabajo utilizado ha sido Eclipse Juno con GNU-ARM GCC como compilador. Esta combinación de herramientas de código abierto GNU nos permiten compilar y enlazar programas para procesadores ARM. El código compilado no puede ser ejecutado por el PC, solo por la placa.

También se ha hecho uso de un *driver* USB para poder comunicar la placa y el servidor GDB. Así desde Eclipse se ha podido cargar y depurar el programa ejecutándose en la placa.

3.3.2. Sistema operativo (ChibiOS/RT)

ChibiOS/RT [8] es un sistema operativo que ha permitido simplificar el programa y poder centrarnos en la comunicación con los sensores. Este sistema operativo en tiempo real nos proporciona ya programada la inicialización de la placa y el sistema HAL que consiste en una serie de drivers ya programados para poder hacer uso de los periféricos del micro controlador. También ofrece soporte para crear hilos multitarea, aunque esta funcionalidad no se ha usado dado que el micro controlador tiene un solo núcleo y aunque se programara en multitarea el tiempo resultante sería el mismo. En lugar de eso se ha optado por optimizar el código. ChibiOS/RT es un sistema operativo de código abierto y con una comunidad activa pero no muy grande que lo respalda, aparte ha sido creado para poder configurarlo desde ficheros externo de tal forma que se puede seleccionar los periféricos que inicializar con la finalidad de reducir el programa final.

Principalmente en este proyecto se ha hecho uso del driver de comunicación I2C y del driver de comunicación del puerto serie sobre USB de esta forma hemos podido crear una comunicación serie con el ordenador sin necesidad de tener un puerto serie en el ordenador.

3.4. Procedimiento experimental

Tal y como se puede observar en la siguiente Figura 6, el programa desarrollado en C para la placa STM32F4 se puede dividir en 5 bloques.

Figura 6: Bloques del programa en C

En el primer bloque del programa se inicia quitar ChibiOS, el sistema HAL y se configuran el puerto I2C 1 que utiliza los puertos GPIOB 8 y 9 para las líneas SCL y SDA respectivamente y el puerto I2C 2 que utiliza el puerto GPIOB 10 y 11 para las líneas SCL y SDA respectivamente. Los dos puertos se inician con configuración de *fast-mode*. El segundo y tercer bloque se ejecutan una vez, su objetivo es configurar la resolución y activar los ejes de los sensores.

Los últimos dos bloques son los que se repiten a lo largo de la duración del programa. La lectura de cada uno de los sensores se hace por separado haciendo uso de las funciones de lectura del I2C. La iteración de estos bloques la lleva acabo un solo hilo. Al principio del proyecto se planteó la posibilidad de ejecutar la parte de procesado con la parte de captación de datos en paralelo, pero dado que el micro controlador solo tiene un núcleo y que para cada iteración del filtro de señal necesitamos nuevos datos se optó por programarlo secuencialmente, aun así en el apartado de resultados se discute porque esta decisión no fue del todo acertada. Para minimizar el impacto de un bloqueo por parte de alguno de los sensores, la comunicación I2C se ha implementado con un *timeout* y el algoritmo de procesado de señal puede efectuarse aun faltando los datos de alguno de los sensores.

Los datos a través del puerto I2C se envían por paquetes de 8 bits. Cada sensor tiene una resolución de 16 bits por eje, por tanto hay que juntar los 6 Bytes de dos en dos Bytes para poder obtener 3 enteros de 16 bits. Para hacerlo, se tienen que juntar los primeros 8bit, que contienen la parte de mayor peso, con los 8 bits consecutivos, que contienen la parte de menor peso. Por cada sensor se adquieren 3 enteros de 16 bits, por tanto por cada placa de sensores se tienen 9 enteros de 16bits. En el conjunto de las tres placas de sensores son un total de 27 enteros de 16bits. Por tanto por cada iteración se captan 432 bits. Teóricamente, haciendo uso del I2C en modo de alta velocidad, el mínimo tiempo con el que se podría llegar a obtener un conjunto de datos de todos los sensores cada 1.08ms. En este caso no se ha tenido en cuenta el *overhead* de información que necesita el bus I2C.

Por defecto ChibiOs utiliza un sistema de canales DMA para comunicarse con los periféricos. DMA es una característica que permite que ciertos subsistemas de *hardware* tengan acceso a sistemas de memoria principales independientemente de la CPU. Dado que no se puede efectuar la parte de procesado de señal sin tener los datos de los sensores solo se ha tenido en cuenta el sistema DMA para que los distintos periféricos no utilizasen los mismos canales de este sistema.

La parte del filtrado se implementó independientemente de las otras dado que un objetivo del proyecto era la creación de un algoritmo optimo, por eso se ha tratado de minimizar al máximo posible el número de operaciones a efectuar y el coste computacional del algoritmo, dado que para obtener un conjunto de resultados de todos los sensores el algoritmo se tiene que ejecutar 3 veces.

Por lo que respecta a la implementación se soldaron las tres placas de sensores y se conectaron mediante cables flexibles al micro controlador. Cada uno de ellos se ha de posicionar en una articulación, hombro, codo y muñeca, no se han de poner encima justo de la articulación sino un poco más cerca del extremo del brazo.

Los datos ya filtrados del micro controlador al PC se envían mediante USB por protocolo serie. Una vez en el ordenador ya pueden ser usados por cualquier programa que pueda leer un puerto serie. En nuestro caso vamos usamos un script que se ha implementado en C# para Unity de tal forma que podamos mover un modelo de brazo humano en tres dimensión que hemos creado previamente en Blender. El script calcula los ángulos de rotación de las articulaciones a partir de los cuaterniones que se envían. Está transformación se puede ver en las ecuaciones (7), (8) y (9).

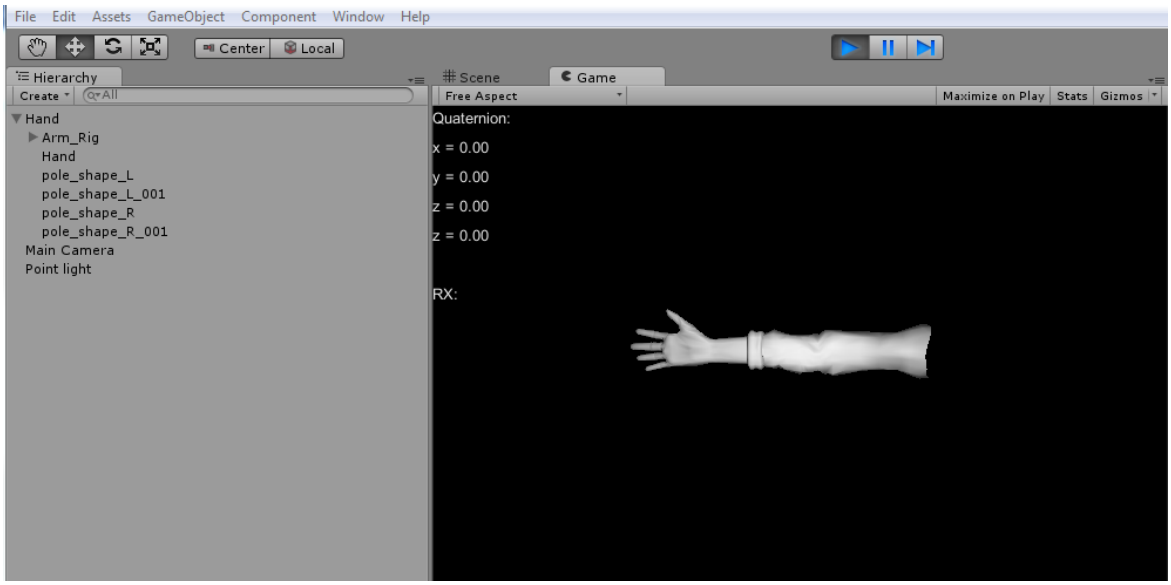

Figura 7: Modelo 3D del brazo humano en Unity

4 Resultados

El hecho de usar un filtro basado en el algoritmo de gradiente descendiente permite no tener que conocer ninguna muestra de antemano ni saber cuál es la varianza de la señal. Algoritmos como el filtro de Kalman extendido [3, 4] tienen una dependencia con las condiciones iniciales y saber cuál es la estadística de la señal. Otra ventaja que tiene este filtro es que no es necesario invertir ninguna matriz. Este hecho hace que la carga computacional de este algoritmo sea menor frente a otros algoritmos. También, se ha podido crear dos algoritmos diferentes dependiendo de los sensores que se utilicen.

4.1. Rendimiento estático y dinámico

Normalmente [4] para cuantificar el rendimiento de un sensor de orientación se suele usar el error cuadrático medio (RMS), tanto en estático como en dinámico, de los diferentes ángulos de Euler desacoplados entre ellos. *Pich*, *roll* y *heading* cada uno corresponde a la rotación alrededor del eje $\boldsymbol{x}$, $\boldsymbol{y}$ y $\boldsymbol{z}$ respectivamente. El error en los diferentes ángulo lo vamos a denotar $\psi_\epsilon, \theta_\epsilon$ y $\emptyset_\epsilon$. La representación en ángulos de Euler tiene ventajas y desventajas, por un lado, es muy fácil interpretar gráficamente, pero por otro lado, tiene la desventaja que es que falla al describir el acoplamiento entre cada uno de los parámetros, esto provoca que a la larga se produzcan errores si la secuencia llega a una singularidad.

Para poder calcular el error cuadrático medio se ha usado Matlab. Para el cálculo hemos extraído un total 4 sets de 7000 muestras de los cuaterniones filtrados en estático y en dinámico. Se ha hecho el cómputo del RMS para cada uno de los sets. Para discriminar los datos dinámicos de los estáticos, se escogió un umbral para la velocidad angular de 5º/s. La condición estática se forma por las medidas las cuales el giroscopio no midiera más de 5º/s y las de condición dinámica eran aquellas cuya medida era superior a 5º/s. El valor del umbral ha sido escogido con la finalidad de ser significativamente superior al ruido. El valor del RMS ha sido computado en una ventana de 27 segundos. Los valores que se muestran en la siguiente tabla son la media de los valores del RMS de los 4 sets de muestras.

Ángulo de Euler	Filtro propuesto
RMS($\emptyset_\epsilon$) estático	0.601º
RMS($\emptyset_\epsilon$) dinámico	0.675º
RMS(θ_ϵ) estático	0.702º
RMS(θ_ϵ) dinámico	0.668º
RMS(ψ_ϵ) estático	1.173º
RMS(ψ_ϵ) dinámico	1.240º

Tabla 2: Error RMS de los ángulos de Euler en estático y en dinámico

Como se puede observar los resultados obtenidos están por debajo de 1º para $\emptyset$ y θ y por debajo de 2º para ψ. Estos resultados son más que satisfactorios dado que los movimientos del brazo humano son porcentualmente muy superiores a este error.

Lo ideal para encontrar el valor óptimo de β seria calcular el valor del RMS de los ángulos de Euler *Pitch*, *Roll* y *Heading*, computar la media de estos tres y repetir el experimento variando β entre 0 y 0.2, dado que a partir de 0.1 el valor de la media no decrece. Como no se ha podido realizar este

experimento, se ha cogido un valor de β igual a 0,05, este valor para el cual la media del RMS dinámico es aproximadamente el mínimo.

4.2. Rendimiento de software

La implementación de los dos algoritmos de filtrado de señal, AHRS y el IMU se han implementado con el objetivo de minimizar carga computacional sobre el micro controlador. Con ese objetivo, las ecuaciones matemáticas se han resuelto y se han implementado en forma de polinomios y no de ecuaciones vectoriales. En un principio se había implementado el filtro con la librería DSP pero los cálculos del Jacobiano así como las transposiciones de matrices creaban una latencia que incumplía los requisitos. Por ello, la versión final de los algoritmos propuestos hacen solo uso de funciones aritméticas básicas, El algoritmo AHRS tiene un total de 374 operaciones aritméticas, 209 multiplicaciones, 75 sumas, 85 restas, 4 divisiones y 1 raíz cuadrada. El algoritmo IMU tiene 149 operaciones, 97 multiplicaciones, 27 sumas, 18 restas y 4 divisiones. Con el fin de reducir el número de raíces cuadradas se ha implementado la función *invSqrt(float x)* basada en *Fast inverse square-root* de esta forma las normalizaciones de los vectores se han podido optimizar.

Con el fin de poder hacer algunos cálculos de tiempos de cómputo, se ha hecho uso de las siguientes funciones: *chTimeNow()*, retorna el tiempo actual en pulsaciones que ha ejecutado la CPU desde la inicialización del temporizador. *chTimeElapsedSince(systime_t start)*, retorna el lapso de tiempo que ha habido este el punto *start* y el actual, el valor son el número de pulsaciones de la CPU. *MS2ST(systime_t time)* retorna el valor de *time* en milisegundos.

Se ha podido calcular la duración media de la iteración del programa que ha sido de 2 milisegundos, por tanto, al PC se envían 500 sets de tres cuaterniones, un cuaternión por articulación, cada segundo al ordenador. Estos resultados es haciendo uso del filtro AHRS. Tal y como se ha calculado anteriormente si contásemos solo la transmisión por el I2C deberíamos ser capaces de obtener un dato cada 1.08ms a causa del procesado de señal este tiempo se ha visto duplicado, reduciendo a la mitad en número de sets que podríamos obtener. Esté aumento prácticamente es atribuible a los tiempos de espera de la CPU dentro de la recepción del I2C y la transmisión serie por USB. Aun así el resultado ha sido satisfactorio dado que se obtienen suficientes muestras para trabajar con el PC. Incluso se podría plantear hacer uso de un micro controlador menos potente dado que se ha podido comprobar que la capacidad de procesado de la CPU no es un factor limitante.

El hecho que el programa no se ha implementado en multihilo, hace que la CPU quede bloqueada mientras se establece la comunicación I2C con los sensores y la comunicación serie con el PC. Esto ocasiona que no se haya podido sacar la máxima potencia del micro controlador y no hayamos alcanzado el óptimo. Si se hubiese implementado con un diseño multihilo podría haberse optimizado el programa de tal forma que se ejecutaran en paralelo el procesado de las muestras anteriores y el envió de estas, con la captación de las muestras actuales. Un posible diseño seria hacer uso de 4 hilos, 2 que gestionaran cada uno de los I2C, 1 de envió de datos al PC y por último y con una prioridad de ejecución más baja el de procesado. Este diseño requeriría hacer uso de semáforos para sincronizar todos los hilos.

Se pidió ayuda a tres personas para realizar un experimento que consistía en realizar 5 series de 10 movimientos de 180º lo más rápido que pidiesen. Se midió el tiempo que tardaban en cada serie con la finalidad de calcular la velocidad angular media de un brazo. La media de tiempo que se tarda en recorrer 1800º es de 888ms, esto hace que la velocidad angular del hombro sea de unos 2.027º/ms. Con este dato aproximado podemos calcular que tendremos una resolución mínima aproximada de unos 4º/muestra.

5 Conclusiones y futuros desarrollos:

Una vez finalizado el proyecto y por finalizar entendemos el cumplimiento de todos los objetivos y requisitos marcados antes de empezar. Se ha construido la interfaz capaz de monitorizar el brazo humano, haciéndolo a tiempo real y lo hace con una gran resolución en muestras, esto permite replicar los movimientos con una mayor precisión. Otro de los requisitos cumplidos, es hacer uso únicamente de sensores locales y no depender del lugar donde se utiliza la interfaz. Por último también se ha conseguido que todo el procesado estuviera en el micro-controlador. Por tanto todos los objetivos han sido alcanzados y sus respectivos requisitos cumplidos.

Uno de los principales problemas del proyecto ha sido el dimensionamiento del mismo. El proyecto tiene mucho que dar de sí y puede mejorar en muchísimos aspectos. Con el fin de crear la interfaz se ha tenido que lidiar con muchísimas facetas de la ingeniera; programación de bajo y alto nivel y en diferentes lenguajes de programación, protocolos de comunicación, teoría de filtrado de señal... Uno de los puntos clave del proyecto ha estado la fase previa al desarrollo o implementación, que ha sido la investigación y organización del mismo. Esta fase ha hecho posible poder mejorar a tiempo el alcance del proyecto para poder cumplir los requisitos.

Un futuro desarrollo en la parte teórica del proyecto sería la compensación del sesgo del giroscopio. El sesgo del giroscopio varía en función de la temperatura y el movimiento, esto hace que prácticamente ninguna implementación práctica de IMU o AHRS haya podido compensar. Una ventaja que tiene el filtro de Kalman es que es capaz de estimar el sesgo del giroscopio como un estado adicional en el modelo del sistema. Sin embargo Mahony [2] demuestra que el sesgo del giroscopio puede ser compensado por la integral de la realimentación del error del cambio de orientación.

En la parte electrónica, se podría dar autonomía a los sensores dotándoles de alimentación propia y un sistema de trasmisión de datos inalámbrico. Con esto y dotando al micro-controlador del receptor adecuado, conseguiríamos poder monitorizar la orientación de cualquier parte del cuerpo pudiéndose mover sin necesidad de llevar cables encima. Otra posible mejora, seria cambiar el estándar de comunicación empleado con el fin de poder conseguir más muestras por segundo. También podría utilizarse I2C pero en modo de 3,4Mb/s. Si se cambiara el modo de comunicación también habría que cambiar los sensores dado que los escogidos solo pueden trabajar con I2C y, aunque limitado por la placa de sensores, SPI.

Uno de los puntos que no se ha hablado a lo largo de todo el proyecto es las aplicaciones de la interfaz, estas pueden ser muy variada. Desde una vertiente de negocio podría desarrollarse un periférico de realidad virtual con el cual se pudiese recrear los movimientos de una persona sobre un avatar virtual. Una vertiente más científica seria el uso de este dispositivo para el control de brazos mecánicos en zonas radioactivas o peligrosas. Una vertiente más social sería la implementación de un traductor de lenguaje de sordo-mudos. Estas son algunas de las muchas posibles aplicaciones de la interfaz.

6 Bibliografía

[1] KUIPERS, Jack B. *Quaternions and rotation sequences*. Princeton: Princeton university press, 1999.

[2] MAHONY, Robert; HAMEL, Tarek; PFLIMLIN, Jean-Michel. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 2008, vol. 53, no 5, p. 1203-1218.

[3] KALMAN, Rudolph Emil. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 1960, vol. 82, no 1, p. 35-45.

[4] SABATINI, Angelo M. Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. *Biomedical Engineering, IEEE Transactions on*, 2006, vol. 53, no 7, p. 1346-1356.

[5] National Geophysical Data Center. "Geomagnetic Data". [Online] Disponible: http://www.ngdc.noaa.gov/geomag/data.shtml. [Accedido: 13 Noviembre 2014].

[6] SEMICONDUCTORS, Philips. The I2C-bus specification. *Philips Semiconductors*, 2000, vol. 9397, no 750, p. 00954.

[7] MADGWICK, Sebastian OH; HARRISON, Andrew JL; VAIDYANATHAN, Ravi. Estimation of IMU and MARG orientation using a gradient descent algorithm. En *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*. IEEE, 2011. p. 1-7.

[8] ChibiOS/RT & ChibiOS Forum. [Online] Disponible: http://www.chibios.org/ http://forum.chibios.org/ [Accedido: múltiples veces a lo largo del proyecto].

[9] Manel Dominguez, Vicente Jiménez, Daniel Bardés, Clemente Pol. *Manual de practiques de SBM/SEBM*. Universidad Politécnica de Cataluña. 2014

[10] NXP Semiconductors. I2C-bus specification and manual. *NXP Semiconductors*, 2007, rev. 03.

7 Apéndices

7.1. I2C specifications

7.2. Altimu -10 v.4 (Información extra)

7.2.1. Sensibilidad de los sensores

Sensor	Rango de medida
L3GD20H (giroscopio)	±245, ±500, o ±2000°/s
LSM303D (acelerómetro)	±2, ±4, ±6, ±8, o ±16 g
LSM303D (magnetómetro)	±2, ±4, ±8, o ±12 gauss

Tabla 3: Rangos de medida de los sensores

7.2.2. Esquema

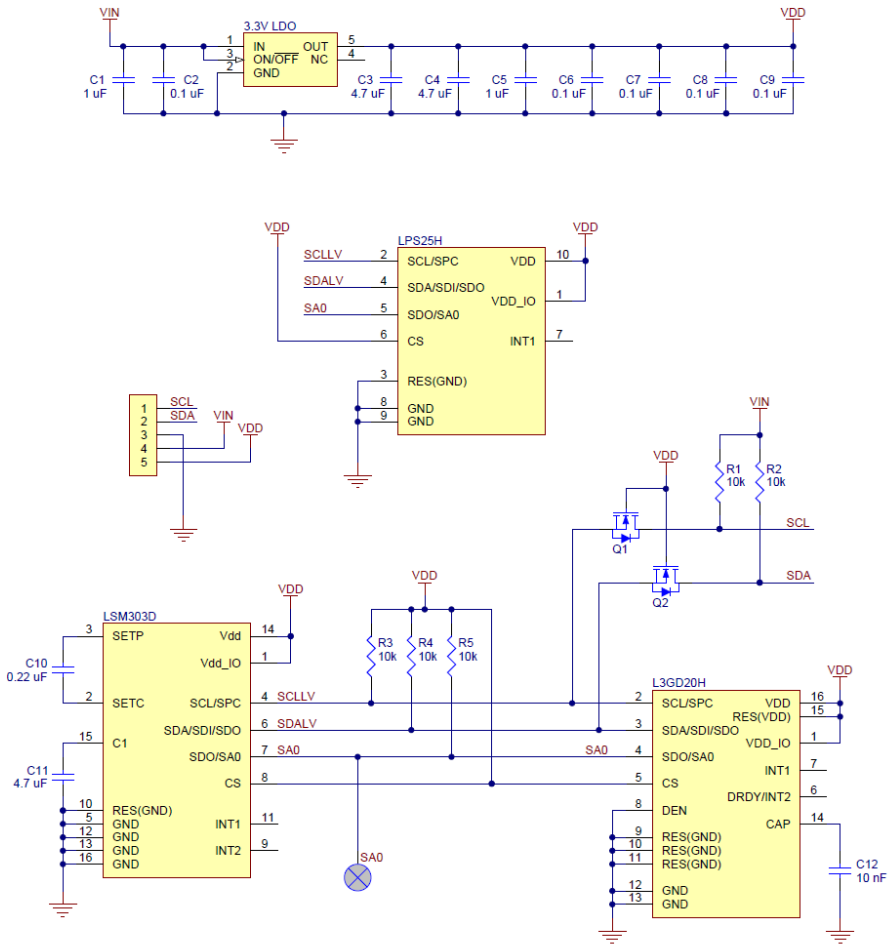

Figura 8: Esquemático de la placa AltIMU v4

7.3. Algoritmo (C-Script)

7.3.1. AHRS

```
void AHRS(float gx, float gy, float gz, float ax, float ay, float az, float mx, float my, float mz) {

	float recipNorm;
	float s1, s2, s3, s4;
	float qDiff1, qDiff2, qDiff3, qDiff4;
	float hx, hy;
	float _2q1mx, _2q1my, _2q1mz, _2q2mx, _2bx, _2bz, _4bx, _4bz, _2q1, _2q2, _2q3, _2q4, _2q1q3, _2q3q4,
q1q1, q1q2, q1q3, q1q4, q2q2, q2q3, q2q4, q3q3, q3q4, q4q4;

	// Utilizaremos el algoritmo de la IMU por si las lecturas del magnetómetro son incorrectas
	if((mx == 0.0f) && (my == 0.0f) && (mz == 0.0f)) {
		IMU(gx, gy, gz, ax, ay, az);
		return;
	}

	// Tasa de cambio del cuaternión debido al giroscopio, ecuaciones (11) y (12)
	qDiff1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz);
	qDiff2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy);
	qDiff3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx);
	qDiff4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx);

	// Calcular la contribución del acelerómetro solo si este no es invalido,
	// evitamos los NaN en la normalización de la medida del acelerómetro
	if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {

		// Normalización de la medida del acelerómetro
		recipNorm = invSqrt(ax * ax + ay * ay + az * az);
		ax *= recipNorm;
		ay *= recipNorm;
		az *= recipNorm;

		// Normalización de la medida del magnetómetro
		recipNorm = invSqrt(mx * mx + my * my + mz * mz);
		mx *= recipNorm;
		my *= recipNorm;
		mz *= recipNorm;
		// Creación de variables auxiliares para evitar
		// realizar la misma operación múltiples veces.
		_2q1mx = 2.0f * q1 * mx;
		_2q1my = 2.0f * q1 * my;
		_2q1mz = 2.0f * q1 * mz;
		_2q2mx = 2.0f * q2 * mx;
		_2q1 = 2.0f * q1;
		_2q2 = 2.0f * q2;
		_2q3 = 2.0f * q3;
		_2q4 = 2.0f * q4;
		_2q1q3 = 2.0f * q1 * q3;
		_2q3q4 = 2.0f * q3 * q4;
		q1q1 = q1 * q1;
		q1q2 = q1 * q2;
		q1q3 = q1 * q3;
		q1q4 = q1 * q4;
		q2q2 = q2 * q2;
		q2q3 = q2 * q3;
		q2q4 = q2 * q4;
		q3q3 = q3 * q3;
		q3q4 = q3 * q4;
		q4q4 = q4 * q4;

		// Calculo de la compensación de la distorsión del campo magnético de la tierra
		// Ecuaciones (45) y (46)
		hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx * q2q2 + _2q2 * my * q3 + _2q2 * mz * q4 - mx
* q3q3 - mx * q4q4;
		hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 + _2q2mx * q3 - my * q2q2 + my * q3q3 + _2q3 * mz *
q4 - my * q4q4;
		_2bz = -_2q1mx * q3 + _2q1my * q2 + mz * q1q1 + _2q2mx * q4 - mz * q2q2 + _2q3 * my * q4 - mz
* q3q3 + mz * q4q4; // = hz
		_2bx = sqrt(hx * hx + hy * hy);

		_4bx = 2.0f * _2bx;
		_4bz = 2.0f * _2bz;

		// Corrección del algoritmo de gradiente descendiente Jtran*f ecuación (34)
		s1 = -_2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2 * (2.0f * q1q2 + _2q3q4 - ay) - _2bz * q3 *
(_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (-_2bx * q4 + _2bz * q2) * (_2bx * (q2q3 - q1q4) +
_2bz * (q1q2 + q3q4) - my) + _2bx * q3 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
```

```
                s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1 * (2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q2 * (1
- 2.0f * q2q2 - 2.0f * q3q3 - az) + _2bz * q4 * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (_2bx
* q3 + _2bz * q1) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + (_2bx * q4 - _4bz * q2) * (_2bx * (q1q3
+ q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
                s3 = -_2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4 * (2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q3 * (1
- 2.0f * q2q2 - 2.0f * q3q3 - az) + (-_4bx * q3 - _2bz * q1) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 -
q1q3) - mx) + (_2bx * q2 + _2bz * q4) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + (_2bx * q1 - _4bz *
q3) * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
                s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3 * (2.0f * q1q2 + _2q3q4 - ay) + (-_4bx * q4 +
_2bz * q2) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (-_2bx * q1 + _2bz * q3) * (_2bx * (q2q3
- q1q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q2 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
                recipNorm = invSqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 * s4); // Normalización de la
contribución del giroscopio y del magnetómetro ecuación (44)
                s1 *= recipNorm;
                s2 *= recipNorm;
                s3 *= recipNorm;
                s4 *= recipNorm;

                // Aplicación de la corrección al cuaternión del giroscopio ecuación (43)
                qDiff1 -= beta * s1;
                qDiff2 -= beta * s2;
                qDiff3 -= beta * s3;
                qDiff4 -= beta * s4;
        }

        // Integración del cuaternión ecuación (42)
        q1 += qDiff1 * (1.0f / sampleFreq);
        q2 += qDiff2 * (1.0f / sampleFreq);
        q3 += qDiff3 * (1.0f / sampleFreq);
        q4 += qDiff4 * (1.0f / sampleFreq);

        // Normalización del cuaternión
        recipNorm = invSqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 * q4);
        q1 *= recipNorm;
        q2 *= recipNorm;
        q3 *= recipNorm;
        q4 *= recipNorm;
}
```

7.3.2. IMU

```
void IMU(float gx, float gy, float gz, float ax, float ay, float az) {
        float recipNorm;
        float s1, s2, s3, s4;
        float qDiff1, qDiff2, qDiff3, qDiff4;
        float _2q1, _2q2, _2q3, _2q4, _4q1, _4q2, _4q3 ,_8q2, _8q3, q1q1, q2q2, q3q3, q4q4;

        qDiff1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz);
        qDiff2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy);
        qDiff3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx);
        qDiff4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx);

        if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {

                recipNorm = invSqrt(ax * ax + ay * ay + az * az);
                ax *= recipNorm;
                ay *= recipNorm;
                az *= recipNorm;

                _2q1 = 2.0f * q1;
                _2q2 = 2.0f * q2;
                _2q3 = 2.0f * q3;
                _2q4 = 2.0f * q4;
                _4q1 = 4.0f * q1;
                _4q2 = 4.0f * q2;
                _4q3 = 4.0f * q3;
                _8q2 = 8.0f * q2;
                _8q3 = 8.0f * q3;
                q1q1 = q1 * q1;
                q2q2 = q2 * q2;
                q3q3 = q3 * q3;
                q4q4 = q4 * q4;

                s1 = _4q1 * q3q3 + _2q3 * ax + _4q1 * q2q2 - _2q2 * ay;
                s2 = _4q2 * q4q4 - _2q4 * ax + 4.0f * q1q1 * q2 - _2q1 * ay - _4q2 + _8q2 * q2q2 + _8q2 * q3q3
+ _4q2 * az;
                s3 = 4.0f * q1q1 * q3 + _2q1 * ax + _4q3 * q4q4 - _2q4 * ay - _4q3 + _8q3 * q2q2 + _8q3 * q3q3
+ _4q3 * az;
                s4 = 4.0f * q2q2 * q4 - _2q2 * ax + 4.0f * q3q3 * q4 - _2q3 * ay;
                recipNorm = invSqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 * s4);
                s1 *= recipNorm;
```

```
                s2 *= recipNorm;
                s3 *= recipNorm;
                s4 *= recipNorm;

                qDiff1 -= beta * s1;
                qDiff2 -= beta * s2;
                qDiff3 -= beta * s3;
                qDiff4 -= beta * s4;
        }

        q1 += qDiff1 * (1.0f / sampleFreq);
        q2 += qDiff2 * (1.0f / sampleFreq);
        q3 += qDiff3 * (1.0f / sampleFreq);
        q4 += qDiff4 * (1.0f / sampleFreq);

        recipNorm = invSqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 * q4);
        q1 *= recipNorm;
        q2 *= recipNorm;
        q3 *= recipNorm;
        q4 *= recipNorm;
}
```

8 Glosario

Cuaternión: número complejo de cuatro dimensiones que es usado para representar la orientación de un cuerpo sólido.

Matriz Hessiano: Matriz que contiene las segundas derivadas parciales de una function **f**

Jacobiano: Matriz formada por las derivadas parciales de primer orden de una función **f**

Gradiente descendiente: algoritmo de optimización d primer orden capaz de encontrar mínimos locales de una función **f**

Filtro de Kalman: algoritmo que usa una serie de medidas previas para predecir la siguiente medida.

AHRS: Sistema de Referencia de Actitud y Rombo, son sensores tridimensionales que proporcionan información acerca de la orientación de un cuerpo solido