UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Máster en Automática e Informática Industrial

Trabajo Final de Máster

Desarrollo de un sistema de navegación para barcos de regata

Alumno: Juan Carlos Almeida

Director: José Luis Navarro

Valencia, Septiembre 2017

Índice General

Índice de Tablas

Índice de Figuras

RESUMEN

Los sistemas embebidos, el Internet de las cosas (IoT) y las tecnologías inalámbricas están en constante desarrollo y uso, la interacción de dispositivos digitales conectados aumenta día a día. Es importante desarrollar sistemas integrales que permitan la comunicación de sistemas analógicos independientes, dichos sistemas generarán grandes cantidades de datos que nos permitirán refinar y mejorar nuestros sistemas y procesos. Este trabajo de tesis consiste en la programación e implementación de un sistema de navegación para barcos de regata, utilizando una BeagleBone Black que es una plataforma de desarrollo de bajo coste, logrando una migración del sistema actual de un velero a un sistema más moderno e inalámbrico. También se ha diseñado una interfaz gráfica en la cual el cliente podrá conocer el estado de las principales variables para la navegación.

En la presente memoria se expone de forma generalizada el hardware y software utilizado, así como el proceso de la inicialización, calibración y funcionamiento para el GPS (Sistema de posicionamiento global), la I.M.U (Unidad de medición inercial), el conversor analógico digital, el módulo eQEP (Enhanced Quadrature Encoder Pulse) y el API de programación de sockets. Además se describen los cálculos necesarios para obtener los valores de TWS (Velocidad de viento verdadero) y TWA (Ángulo de viento verdadero), que son necesarios para la navegación.

Con el desarrollo de este trabajo se pretende crear una alternativa más económica y con una interfaz de usuario más completa y gráfica que los sistemas de navegación actuales en barcos de regata. Permitiéndose una configuración flexible al momento de implementar el dispositivo, proporcionando información en tiempo real y que este desarrollo puede ser compatible para otros medios de transporte que requieran un sistema de navegación más moderno y económico, aprovechando la aplicabilidad y valor añadido que poseen los sistemas embebidos.

ABTRACT

The Embedded systems, Internet of Things (IoT) and wireless technologies are in constant development and use, the interaction of digital connected devices increases day by day. It is quite important to develop integrated systems that allow the communication of independent analog systems, these mentioned systems will generate large amounts of data that will allow us to refine and improve our systems and processes. This thesis consists on the programming and implementation of a navigation system for racing boats, using a BeagleBone Black, which is a platform of low cost development, achieving a switch from the current system of a sailboat to a more modern and wireless system. A graphical interface has also been designed in which the customer can be familiarized with the state of the main variables for navigation.

In the present memory, the hardware and software used, as well as the initialization, calibration and operation process for the GPS (Global Positioning System), the IMU (Inertial Measurement Unit), the analogue digital converter, the eQEP (Enhanced Quadrature Encoder Pulse) module, and the Sockets Programming API are exposed in a general way. In addition, the calculations necessary to obtain the TWS (true wind speed) and TWA (true wind angle) values, which are required for navigation, are described as well.

With the development of this project, it is intended to create a more economical alternative which contains a more complete user interface and graphic than the current navigation systems in racing boats. Allowing a flexible configuration when implementing the device, providing information in real time and this development may be compatible for other means of transport that require a more modern and economical navigation system, taking advantage of the applicability and added value of embedded systems .

1. INTRODUCCIÓN

Todos estamos familiarizados con la idea de un ordenador de escritorio o portátil, y el increíble procesamiento que pueden realizar. Estos ordenadores son de uso general, por lo tanto podemos hacer que ejecuten diferentes programas en distintos momentos dependiendo de la aplicación o requerimiento. En el núcleo de dichos componentes encontramos un microprocesador, un circuito electrónico minúsculo y complejo, que contiene las características centrales de una computadora (Toulson, 2017).

Lo que es menos familiar para muchas personas es la idea de que en lugar de poner un microprocesador en un ordenador de uso general, también este puede colocarse dentro de un producto que no tiene nada que ver con la informática, como una lavadora, tostadora o cámara. El microprocesador se personaliza para controlar ese producto. A esto se le conoce como un sistema embebido, es decir, un sistema que tiene un software incorporado en el hardware de la computadora, lo que hace que un sistema esté dedicado a una aplicación o a partes específicas de la misma, a un producto o parte de un sistema más grande.

En contraste con las computadoras de uso general, los sistemas empotrados realizan un rango estrecho de tareas predefinidas. Por lo general no tienen ninguno de los típicos dispositivos periféricos como un teclado, monitor de pantalla, conexiones en serie, almacenamiento masivo (por ejemplo, unidades de disco duro), etc. o cualquier tipo de software de interfaz de usuario, a menos que lo requiera el producto en el que se utilizan. Esto puede hacer posible reducir en gran medida la complejidad, el tamaño y el coste, y en contraste aumentar la robustez de los sistemas embebidos en comparación con los sistemas de uso general. (Linfo, 2017)

Teniendo en cuenta la aplicabilidad y valor añadido que poseen los sistemas embebidos, en este trabajo se va a desarrollar un sistema para la navegación de barcos de regata con un sistema embebido de bajo coste, como es BeagleBone Black, que permitirá tener una navegación más completa, gráfica y flexible, que el sistema propio de estas embarcaciones. Se han utilizado las siguientes funcionalidades, de la BeagleBone Black como: UART, I2C, TCP/IP, conversor analógico digital y el módulo eQEP. Para poder llevar a cabo una visualización de la información de navegación se va a diseñar una interfaz gráfica en JAVA mediante el paradigma de comunicación cliente – servidor. Los principales sensores que se utilizarán para la obtención de información de navegación del velero serán el GPS y la I.M.U.

El alcance de este proyecto abarca desde la portabilidad para el uso correcto del GPS e I.M.U., procesamiento de señales analógicos y digitales, programación del servidor y cliente mediante TCP/IP, desarrollo de una interfaz gráfica hasta la implementación y montaje de la BeagleBone Black, sensores y circuitos de protección que han sido desarrollados para este fin.

En esta memoria se plasmará el proceso de inicialización, calibración e implementación de cada una de las partes antes mencionadas de manera general, así como de las características técnicas del hardware empleado. Además de los cálculos necesarios para obtener los valores de TWS y TWA. Finalmente se detallará el presupuesto que conlleva el diseño de este sistema.

Para concluir, este proyecto está abierto a futuras mejoras como podría ser la implementación de un circuito impreso para evitar ruidos en señales, desplazamiento de las componentes, entre otros. Agregar más funcionalidades a la interfaz gráfica y su desarrollo para dispositivos móviles.

2. OBJETIVOS

2.1. OBJETIVO GENERAL

- Desarrollar un sistema de navegación para barcos de regata, mediante una placa programable de bajo coste.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar la portabilidad de librerías para el GPS e I.M.U.
- Diseñar un proceso de inicialización y calibración para GPS, I.M.U, conversor analógico digital y módulo eQEP.
- Desarrollar un software para la gestión de la información de GPS, Unidad de medición inercial-I.M.U, anemómetro y corredera del barco regata, que permita la visualización de datos de navegación.
- Diseñar e implementar un modelo de comunicación cliente-servidor mediante una placa BeagleBone Black.
- Analizar el comportamiento de la I.M.U., GPS, y módulo eQEP.

3. MARCO TEÓRICO

3.1. SISTEMAS EMBEBIDOS

Un sistema embebido es un artefacto de ingeniería e informática que está sujeto a restricciones físicas. Las restricciones físicas surgen a través de dos tipos de interacciones de procesos computacionales con el mundo físico: (1) reacción a un entorno físico, y (2) ejecución en una plataforma física. En consecuencia, los dos tipos de restricciones físicas son restricciones de reacción y limitaciones de ejecución. Las restricciones de reacción comunes especifican los plazos, el rendimiento y la variación en el retardo de los paquetes recibidos (jitter); se originan a partir de los requisitos de comportamiento del sistema. Las limitaciones de ejecución comunes limitan las velocidades del procesador disponible, la potencia y las tasas de error de hardware; se originan en los requisitos de implementación del sistema. Obtener el control de la interacción de la computación con ambos tipos de restricciones para satisfacer un conjunto de requisitos, es la clave para el diseño de sistemas embebidos (Henzinger, 201X).

Estos se pueden programar directamente en el lenguaje ensamblador del microprocesador incorporado sobre el mismo o también, utilizando los compiladores específicos. Pueden utilizarse lenguajes como C o C++; en algunos casos, cuando el tiempo de respuesta de la aplicación no es un factor crítico, también pueden usarse lenguajes como JAVA. (UNED, 2014).

Los sistemas embebidos consisten en hardware, software y un entorno, lo común como la mayoría de los sistemas informáticos. Sin embargo, existe una diferencia esencial entre los sistemas empotrados y otros sistemas informáticos: dado que los sistemas embebidos implican una computación sujeta a restricciones físicas, la poderosa separación de la capacidad de cómputo (software) de la física (plataforma y entorno), que ha sido una de las ideas centrales de la informática, no funciona para los sistemas embebidos. En cambio, el diseño de sistemas embebidos requiere un enfoque holístico que integre paradigmas esenciales desde el diseño de hardware, el diseño de software y la teoría de control de una manera consistente (Henzinger, 201X).

Para la próxima generación de sistemas embebidos se puede anticipar tres niveles principales de complejidad derivados de esta nueva situación (IDC, 2012):

1. La proliferación de dispositivos: Un mundo de alta densidad de cómputo, de 7 billones de dispositivos que estuvieron conectados a internet en el 2010, 5 billones no fueron ordenadores.
2. La proliferación de datos: Un mundo de datos y análisis almacenados por el sistema, el volumen de datos generados por los sistemas embebidos aumentará aún más rápido con cada conexión de los mismos.
3. La diversidad y riqueza de funciones y servicios, la interoperabilidad y redes: Estos aspectos continuarán desarrollándose debido al efecto demanda de la sociedad y como resultado del factor de diferenciación y competitividad para la industria.

Es complejo definir estándares de diseño para sistemas embebidos porque cada aplicación específica conduce a diferentes opciones de diseño. Las funciones típicas de estos sistemas pueden ser las siguientes:

- Procesamiento: Capacidad de procesar las señales analógicas / digitales.

- Comunicación: Capacidad de transferir señales desde / hacia el mundo exterior.
- Almacenamiento: La capacidad de preservar la información temporal dentro del sistema embebido.

Cada aplicación específica hecha por un sistema embebido tiene requisitos diferentes para procesamiento, suministro de energía, almacenamiento y comunicación. Además, según sus características comerciales se pueden describir en los siguientes puntos:

- Costo final: El costo del producto final es un parámetro muy importante para las opciones de diseño.
- Tiempo de comercialización: En el diseño de un sistema embebido siempre debe tener en cuenta el momento en que desea que el producto sea lanzado en el mercado.
- Tiempo de vida: Otro factor importante es el tiempo de vida esperado para el producto; que puede variar de unos pocos días a varios años o décadas.

Además de los parámetros implicados en el mercado, el hardware, las características de software y los sistemas embebidos se utilizan para ser confiables. En realidad, en la fase de diseño es necesario considerar los siguientes aspectos:

- Confiabilidad: Evaluación realista de la probabilidad de que el sistema falla.
- Mantenimiento: El sistema puede ser reparado o reemplazado dentro de un cierto intervalo de tiempo.
- Disponibilidad: La probabilidad de que el sistema esté funcionando; depende esencialmente de la fiabilidad y la facilidad de mantenimiento.
- Seguridad: Poder de recuperación del sistema contra el uso no autorizado.

El sistema en tiempo real, es un sistema diseñado para operar dentro de los parámetros de tiempo bien definidos. Prácticamente, un sistema en tiempo real funciona correctamente únicamente si cada configuración de entrada es producida por la salida correcta respetando restricciones de tiempo bien definidas. (Di Paolo, 2015)

En la Tabla 1, se resume las diferentes opiniones que se tiene para los sistemas embebidos y su impacto tecnológico en diferentes sectores.

Tabla 1. Sistemas embebidos en diferentes sectores

SECTOR	SISTEMA/TECNOLOGÍA
Transporte como un servicio	**Carga inteligente, vehículo-a-red (V2G) en tecnología de coches (software, comunicación)** ▪ Infraestructura de alimentación eléctrica. ▪ Red de amplia cobertura, GPS. **Servicios al conductor/consumidor** ▪ Movilidad. ▪ Redes sociales.
Salud	**Telemedicina y monitorización remota de pacientes** ▪ Dispositivos médicos inteligentes/conectados. ▪ Redes de cobertura. ▪ Registros médicos electrónicos (EMR). ▪ Registros de salud personal (PHR).
Energía	**Redes Inteligentes (Smart Grids)** ▪ Contadores inteligentes, sensores de red. ▪ Sistema de gestión de los datos de medición.

	Eficiencia Energética ▪ Termostatos y aplicaciones inteligentes, para el hogar. ▪ Redes de cobertura para el hogar.
Distribución	**Sistemas de Almacenamiento en tiempo-real** ▪ Quioscos inteligentes. ▪ Asistentes digitales de compras personales. **Sistemas de gestión de energía** ▪ Iluminación. ▪ Refrigeración.

Fuente: (IDC, 2012)

La Tabla 2 especifica las características requeridas para los sistemas embebidos con una perspectiva para el año 2020, en la que se indican los requerimientos con respecto a:

- Exigencias de criticidad: Seguridad y certificaciones.
- Gestión de la arquitectura distribuida y autonomía del sistema.
- Necesidades del usuario: Interfaz Humano-Máquina y conexión e interoperabilidad sin fallos.
- Controladores y desafíos tecnológicos: Procesadores multinúcleo y software de virtualización, y el manejo de la energía para dispositivos pequeños.

Tabla 2. Características requeridas para sistemas embebidos para el 2020.

	Exigencias Criticas				**Necesidades del Usuario**		**Manejadores tecnológicos**	
	Garantía	**Seguridad**	**Certificación**	**Arquitectura Distribuida**	**HMI**	**Conexión sin fallos**	**Multi-núcleo y virtualización**	**Consumo de energía**
Automóviles								
Aeroespacial								
Industria automatización								
Consumo energético								
Salud médica								
Comunicaciones								
Consumidor								

Fuente: (IDC, 2012)

3.2. INTERNET DE LAS COSAS (IoT)

Si tuviéramos computadoras que supieran todo lo que hay que conocer sobre las cosas, utilizando la información que recopilaron sin ninguna ayuda de los humanos, seríamos capaces de rastrear y contar cada cosa. Sabríamos cuando las cosas necesitaban ser reemplazadas o reparadas reduciendo en gran medida desperdicios, pérdidas y gastos (CEA, 2015).

El IoT puede realizarse en tres paradigmas: Internet orientado (middleware), orientado a las cosas (sensores) y semántico orientado (conocimiento). Aunque este tipo de delineación se requiere debido a la naturaleza interdisciplinaria, la utilidad de IoT puede ser desencadenada sólo en un dominio de aplicación donde los tres paradigmas se cruzan (Véase la Figura 1) (Atzori, 2010).

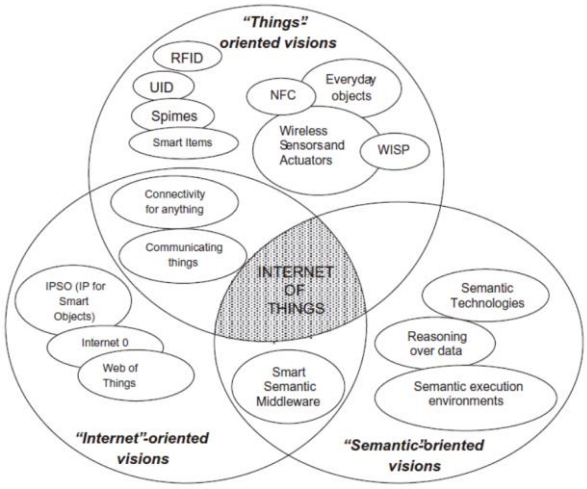


Figura 1. Paradigma del Internet de las cosas (IoT)

El IoT permite la interconexión de dispositivos de detección y actuación que proporcionan la capacidad de compartir información entre plataformas a través de un marco unificado, desarrollando una imagen operativa común para permitir aplicaciones innovadoras.

Internet de las cosas ha sido identificado como una de las tecnologías emergentes como se señala en el Ciclo de Hype IT de Gartner (ver Fig. 2). Un Ciclo de Hype es una forma de representar la aparición, adopción, madurez e impacto en las aplicaciones de tecnologías específicas. Se ha pronosticado que IoT tomará 5-10 años para la adopción del mercado.

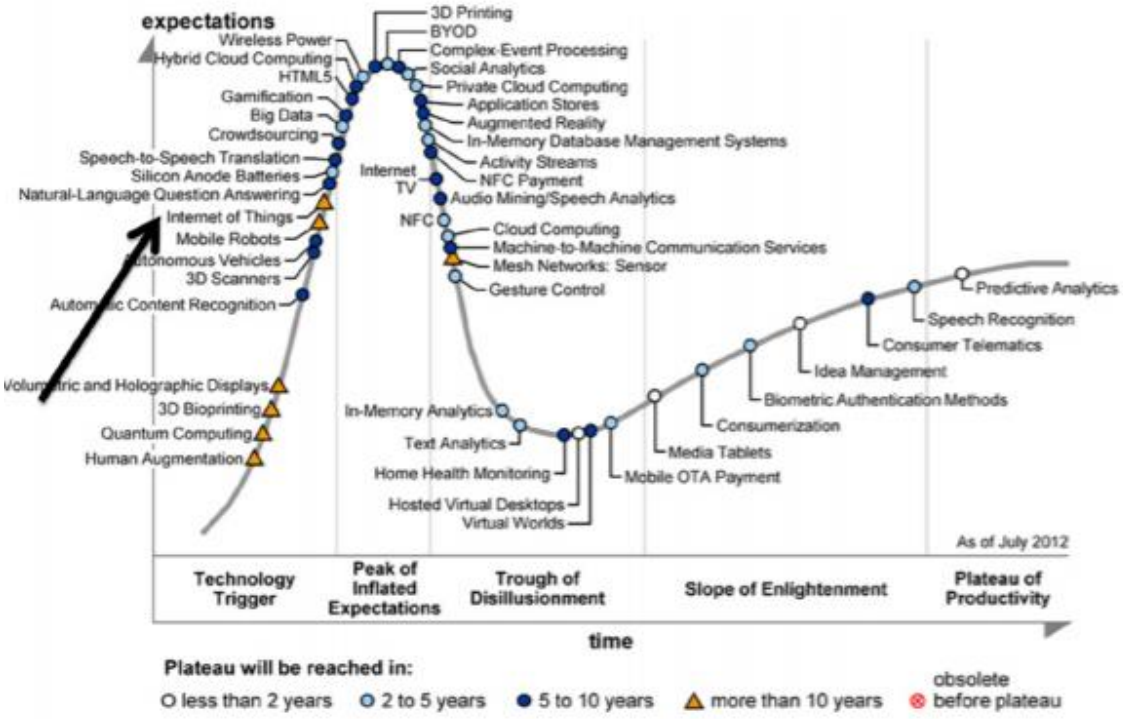


Figura 2. Ciclo Hype de Garner sobre tecnologías emergentes.

Fuente: (Gubbi, 2013)

Existen tres componentes de IoT que permiten una integración ininterrumpida (Gubbi, 2013):

a) Hardware: Compuesto de sensores, actuadores y hardware de comunicación incorporado.
b) Middleware: Es la demanda de almacenamiento y herramientas de computación para análisis de datos
c) Presentación: Fácil de entender la visualización y herramientas de interpretación que pueden ser ampliamente accesibles en diferentes plataformas y que pueden ser diseñadas para diferentes aplicaciones.

Uno de los ámbitos donde estas tecnologías provocarán una transformación profunda es en la industria. El paradigma de la fábrica con su proceso productivo conectado, teniendo información detallada en tiempo real de cada uno de las etapas en la manufactura, el consumo de insumos, energía y materias primas, situación de inventarios, logística y distribución, permitirá mejorar la eficiencia y la productividad, reducir costes, mejorar la calidad del proceso y de los productos y abrirá la posibilidad a la personalización en el proceso productivo.

Para la consecución adecuada y desde un punto de vista industrial del IoT y de los servicios interconectados, deben basarse en los siguientes principios básicos de diseño (EOI, 2015):

- **Interoperabilidad**: Ofrecer la capacidad de interconexión de todos sus elementos, materiales y humanos, mediante el uso del IoT y sus servicios.
- **Virtualización**: La fábrica inteligente ha de tener una copia virtual mostrando toda la información de sensores y sistemas, además de modelos de simulación.
- **Descentralización**: Dado que los objetos conectados en las fábricas inteligentes deberán tener capacidades de decisión autónoma.

- **Capacidades de tiempo real**: Mediante la captura de datos, su análisis y toma de decisiones en tiempo real, incorporando la inteligencia de negocio necesaria.
- **Orientación al servicio**: Mediante la capacidad de ofrecer un catálogo de servicios que permita la interacción y la creación de nuevas aplicaciones y, por ende, mayor valor añadido.
- **Modularidad**: Con la flexibilidad máxima en la fábrica inteligente para la adición, sustracción o sustitución de cualquiera de sus elementos.

Las potencialidades ofrecidas por el IoT posibilitan el desarrollo de un gran número de aplicaciones, de las cuales sólo una parte muy pequeña está actualmente disponible para nuestra sociedad.

Muchos son los dominios y los ambientes en los que las nuevas aplicaciones probablemente mejorarían la calidad de nuestras vidas: en casa, durante un viaje, por motivos de enfermedad, en el trabajo, solo por citar algunos. Estos entornos ahora están equipados con objetos con inteligencia primitiva, la mayoría de las veces sin ninguna capacidad de comunicación. Dando a estos objetos la posibilidad de comunicarse unos con otros y elaborar la información percibida desde el entorno implican tener diferentes entornos donde se puede desplegar una amplia gama de aplicaciones. En la Figura 3 se puede observar la evolución que tendrá el IoT en nuestra sociedad, en la que se pueden ver aplicaciones a corto, mediano plazo e inclusive aplicaciones futurísticas. (Atzori, 2010).

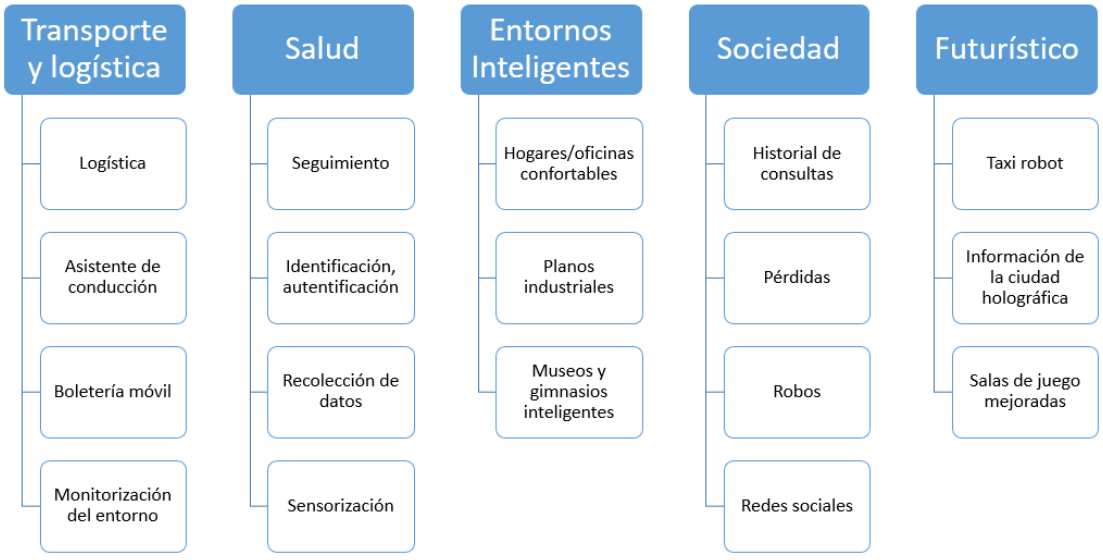


Figura 3. Dominios de aplicaciones y escenarios principales relevantes

Fuente: (Atzori, 2013)

3.3. SISTEMAS EMBEBIDOS E INTERNET DE LAS COSAS

En períodos anteriores, el crecimiento de los datos se debió en gran medida al aumento de la computadora personal y al consumo de entretenimiento digital. El mundo de hoy contiene más dispositivos de consumo (PC, teléfonos, consolas de videojuegos y reproductores de música) que los seres humanos, y todos estos dispositivos necesitan datos para operar (IDC, 2017).

La incorporación de la potencia de cálculo en un gran número de dispositivos de punto final se ha convertido en un factor clave para el crecimiento de los datos en nuestra era actual. Hoy en día, el número de dispositivos de sistemas embebidos que alimentan a los centros de datos es inferior a uno por persona en todo el mundo, y en los próximos 10 años, ese número aumentará a más de cuatro por persona. Mientras que los datos de los sistemas embebidos tienden a ser muy eficientes en comparación con los datos del entretenimiento y el uso de otros consumidores, el número de archivos generados será muy grande, midiendo en los quintillones por año. Para poner ese número en perspectiva, tomaría Niagara Falls 210.000 años mover un quintillón galones de agua.

Dado que hay muchos tipos de dispositivos que generan datos, la esfera mundial de datos se ha segmentado en cuatro clasificaciones principales (véase la figura 4). Las categorías de tipo de datos son:

- Entretenimiento: Contenido de imagen y video creado o consumido con fines de entretenimiento.
- Imagen / video sin entretenimiento: Contenido de imagen y video para propósitos no relacionados con el entretenimiento, como imágenes de video vigilancia o publicidad.
- Datos de productividad: Datos tradicionales basados en la productividad, como archivos en Pc y servidores, archivos de registro y metadatos.
- Embebido: Datos creados por dispositivos incrustados, máquina a máquina, y IoT.

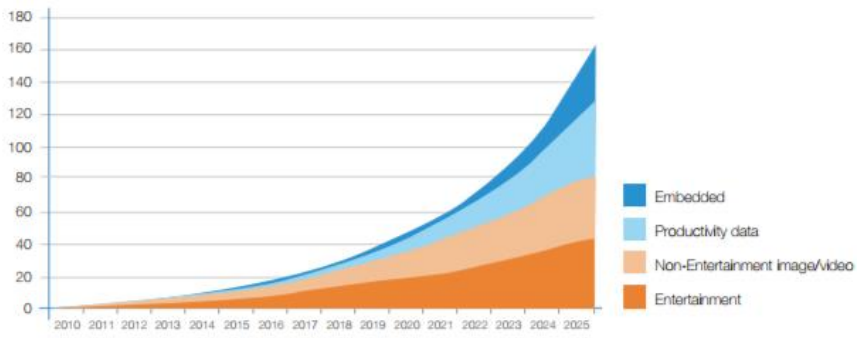


Figura 4. Creación de datos por tipo

Fuente: (IDC, 2017)

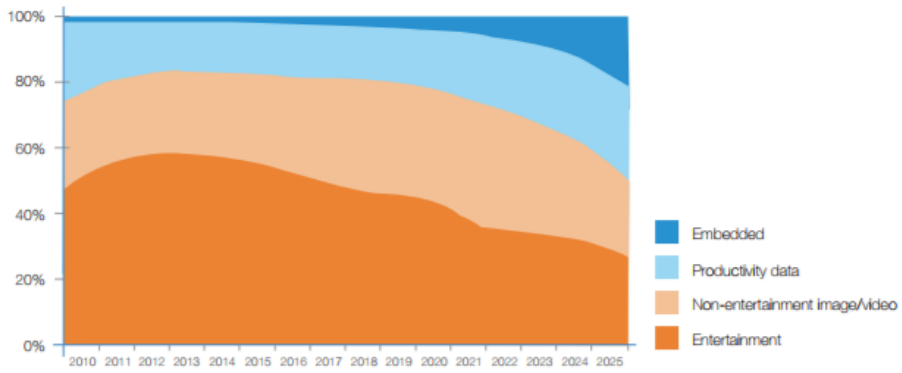


Figura 5. Creación de datos compartidos por tipo.

Fuente: (IDC, 2017).

Para el año 2025, los datos de sistemas embebidos constituirán casi el 20% (véase la figura 5) de todos los datos creados. Los datos de productividad provienen de un conjunto de plataformas informáticas tradicionales como PC, servidores, teléfonos y tabletas. Los datos embebidos, por otro lado, provienen de una amplia variedad de tipos de dispositivos, incluyendo:

- Cámaras de seguridad
- Tarjetas chip
- Lectores RFID
- Estaciones de alimentación
- Automatización de edificios
- Infraestructura inteligente
- Herramientas de máquina
- Automóviles, botes, aviones, autobuses y trenes
- Máquinas expendedoras
- Señalización digital
- Casinos
- Implantes médicos
- Juguetes

Todos estos dispositivos embebidos aumentarán radicalmente el nivel de interacción de la persona promedio con los datos, cambiando la experiencia del usuario. Se espera que la tasa promedio per cápita de interacciones basadas en datos por día se incremente 20 veces en los próximos 10 años a medida que nuestros hogares, lugares de trabajo, electrodomésticos, vehículos, portátiles e implantes lleguen a estar habilitados (ver Figura 6).

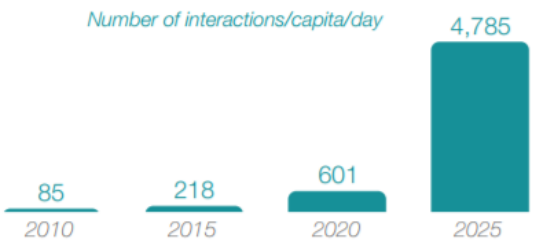


Figura 6.Interacción por persona conectada por día

Fuente: (IDC, 2017)

Los sistemas embebidos están siendo definidos hoy por la migración a sistemas más inteligentes. Se prevé que para el año 2020 la evolución del sistema embebido se centrará en la integración de tecnologías de hardware y software de nivel superior que permitirán la reconfiguración del usuario, permitiendo un funcionamiento autónomo, acceso a Internet y la ampliación del modelo de uso del sistema. (IDC, 2012)

A medida que los dispositivos analógicos independientes dan paso a dispositivos digitales conectados, estos últimos generarán grandes cantidades de datos que, a su vez, nos permitirán refinar y mejorar nuestros sistemas y procesos de formas previamente inimaginables. Datos grandes y metadatos (datos sobre datos) eventualmente tocarán casi todos los aspectos de nuestras vidas. Para el 2025, una persona en cualquier parte del mundo interactuará con dispositivos conectados casi 4.800 veces al día, básicamente una interacción cada 18 segundos. (IDC, 2017)

Se espera que el mercado mundial de sistemas embebidos alcance los € 180 billones en 2020. La creciente adopción y evolución de la Internet de las Cosas (IoT), impulsada por el avance tecnológico, alimentará la industria en los próximos años. (Billore, 2015).

Se estima que el potencial económico de IoT oscila entre € 1,09 trillones por año a €11,20 trillones en todos los sectores a nivel mundial. Asimismo, la venta de dispositivos y servicios conectados llegará a alrededor de €1,94 trillones en 2020. Además la conexión de 100 billones de dispositivos globalmente conectados indica inversiones acumuladas a 2025 de al menos €2 trillones a precios actuales. Por ejemplo, China ya ha destinado €625 millones a la inversión de IoT (Rand Europe, 2012).

3.4. UNIDAD DE MEDIDA INERCIAL - I.M.U

La IMU es un componente electrónico basado en sensores de aceleración (acelerómetro), velocidad angular (giróscopos) y campo magnético (magnetómetro), la cual aporta información acerca del movimiento y orientación que sufre dicha unidad. Es el componente principal de sistemas de guía inercial usados en vehículos aéreos, espaciales, marinos y aplicaciones robóticas.

A continuación, se detallarán los principales componentes:

Acelerómetro: Instrumento capaz de medir aceleración en uno, dos o tres ejes. Existen varios tipos de acelerómetros, dependiendo de su fabricación y funcionamiento. Las I.M.Us incorporan acelerómetros integrados en silicio para reducir el tamaño total de la unidad. La mayoría de éstos son capacitivos, y calculan la aceleración mediante el voltaje obtenido entre dos placas una de las cuales varía su posición dependiendo del movimiento del acelerómetro. Se caracterizan por ser muy precisos en situaciones estables y tener un gran error en situaciones vibratorias o movimientos muy inestables.

Giróscopo: Dispositivo que mide la orientación, basándose en los principios de la conservación del momento angular. La salida de dicho sensor es un voltaje, la variación del cual nos indica en grados por segundo la velocidad angular sufrida por el sensor. Se caracterizan por tener un error constante y lineal.

Magnetómetro: Algunas unidades de medida inercial incluyen también sensores magnetómetros. Estos dispositivos miden la fuerza y/o dirección de los campos magnéticos

que los afectan respecto al campo magnético terrestre. Aunque cabe la posibilidad de que se vean afectados por la variación de otros campos magnéticos en algunas zonas.

Protocolo de comunicación I2C (Inter-Integrated Circuit): Es un bus de comunicación muy utilizado para comunicar circuitos integrados, uno de sus usos más comunes es la comunicación entre un microcontrolador y sensores periféricos. El I2C es un bus multimaestro (el que inicia la comunicación), es decir permite que haya múltiples maestros y múltiples esclavos en el mismo bus. El bus I2C cuenta con dos líneas SDA(datos) y SCL(clock) además de masa. Cada flanco de SCL, marca la aparición de un bit de datos en la línea SDA. Además, hay que tener en cuenta que la línea SDA solo puede cambiar de valor en caso de que la línea SCL este a 0.

3.5. SISTEMA DE POSICIONAMIENTO GLOBAL – GPS

Es un Sistema Global de Navegación por Satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros (usando GPS diferencial), aunque lo habitual son unos pocos metros.

Es un sistema de radionavegación basado en satélites desarrollado y controlado por el Departamento de Defensa de Estados Unidos de América que permite a cualquier usuario saber su localización, velocidad y altura, las 24 horas del día, bajo cualquier condición atmosférica y en cualquier punto del globo terrestre.

Protocolo de comunicación UART: La universal asynchronous receiver/transmitter (UART), es un periférico bastante común en los sistemas empotrados. Históricamente ha sido usado, y sigue usándose, para conectar sensores y dispositivos en una comunicación punto a punto. De este modo una UART no es estrictamente un bus de comunicaciones ya que solo conecta a dos nodos. La UART gestiona el nivel lógico de comunicaciones serie, es decir, que los bits correspondientes a una palabra, se trasmiten secuencialmente por el cable como trenes de pulsos de nivel TTL, de forma que la UART que recibe estos bits tiene un registro de desplazamiento que recompone la palabra transmitida en bits paralelos que se almacenan en la memoria de destino.

La comunicación serie por medio de la UART puede ser:

- "Full duplex": si permite la comunicación en ambos sentidos (del nodo A al B y del B al A) simultáneamente.
- "Half duplex": si permite la comunicación en ambos sentidos, pero no simultáneamente.
- "Simplex": si solo permite la comunicación en un sentido.

Protocolo de comunicación NMEA: NMEA es la abreviatura de National Marine Electronics Association. Es una asociación fundada en1957 por un grupo de fabricantes de electrónica para obtener un sistema común de comunicación entre las diferentes marcas de electrónica naval.

NMEA consiste en mensajes por oraciones o frases, cuya primera palabra, llamada un tipo de datos, define la interpretación del resto de la línea de mensaje. Cada tipo de datos tendría su propia interpretación única y se define en la norma NMEA. En el estándar NMEA no hay comandos para indicar que los GPS deben hacer algo diferente. En su lugar, cada receptor sólo envía todos los datos y espera que gran parte de ella sea ignorada. Algunos receptores

tienen comandos dentro de la unidad que pueden seleccionar un subconjunto de todas las frases o, en algunos casos, incluso las oraciones individuales para enviar. No hay manera de indicar nada de nuevo a la unidad en cuanto a si la frase se está leyendo correctamente o para solicitar un reenvío de algunos datos que no recibió. En su lugar, la unidad receptora sólo verifica la suma de comprobación e ignora los datos si la suma de comprobación es mala calculando que los datos se enviarán de nuevo algún tiempo después.

Existen 19 oraciones interpretadas, que son las siguientes:

```
$GPBOD - Bearing, origin to destination
$GPBWC - Bearing and distance to waypoint, great circle
$GPGGA - Global Positioning System Fix Data
$GPGLL - Geographic position, latitude / longitude
$GPGSA - GPS DOP and active satellites
$GPGSV - GPS Satellites in view
$GPHDT - Heading, True
$GPR00 - List of waypoints in currently active route
$GPRMA - Recommended minimum specific Loran-C data
$GPRMB - Recommended minimum navigation info
$GPRMC - Recommended minimum specific GPS/Transit data
$GPRTE - Routes
$GPTRF - Transit Fix Data
$GPSTN - Multiple Data ID
$GPVBW - Dual Ground / Water Speed
$GPVTG - Track made good and ground speed
$GPWPL - Waypoint location
$GPXTE - Cross-track error, measured
$GPZDA - Date & Time
```

Las que nos interesan para el desarrollo de este trabajo son GPGGA y GPRMC, que se las detallan a continuación:

- GPGGA : Global Positioning System Fixed Data

```
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18
```

Tabla 3. Formato del dato GGA

Nombre	Ejemplo	Unidades	Descripción
ID mensaje	$GPGGA		Encabezado del protocolo GGA
Tiempo UTC	161229.487		hhmmss.sss
Latitud	3723.247		dddmm.mmmm
Indicador N/S	N		N: Norte o S: Sur
Longitud	12158.3416		dddmm.mmmm
Indicador E/W	W		W:Oeste o E: Este
Indicador posición corregida	1		Observar tabla adjunta
Satélites usados	07		Rango 0 a 12
HDOP	1.0		Dilución horizontal de precisión
MSL Altitud	9.0	metros	
Unidades	M	metros	
Separación Geoide		metros	
Unidades	M	metros	
Age of Diff Corr		segundos	

Ref. ID estacion	0000		
Checksum	*18		
<CR> <LF>			Final del mensaje

Tabla 4. Indicador posición corregida

Valor	Descripción
0	Corrección no disponible o inválida
1	Modo GPS SPS, corrección válida
2	GPS Diferencial, Modo SPS, corrección válida
3-5	No soportada

- GPRCM: Recommended minimum specific GPS/Transit data

```
$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10
```

Tabla 5. Formato del dato RMC

Nombre	Ejemplo	Unidades	Descripción
ID mensaje	$GPRMC		Encabezado del protocolo RMC
Tiempo UTC	161229.487		hhmmss.sss
Estado	A		A: Datos válidos o V: Datos no válidos
Latitud	3723.247		dddmm.mmmm
Indicador N/S	N		N: Norte o S: Sur
Longitud	12158.3416		dddmm.mmmm
Indicador E/W	W		W:Oeste o E: Este
SOG (Velocidad sobre tierra)	0.13	nudos	
COG(Curso sobre tierra)	309.62	grados	Verdadero
Fecha	120598		ddmmyy
Variación magnética		grados	E: Este o W: Oeste
Modo	A		A: Autónomo, D: DGPS, E:DR
Checksum	*10		
<CR> <LF>			Final del mensaje

3.6. PROTOCOLO TCP/IP

Las siglas TCP/IP se refieren a un conjunto de protocolos para comunicaciones de datos. Este conjunto toma su nombre de dos de sus protocolos más importantes, el protocolo TCP (Transmission Control Protocol) y el protocolo IP (Internet Protocol).

El protocolo TCP/IP fue creado antes que el modelo de capas OSI, así que los niveles del protocolo TCP/IP no coinciden exactamente con los siete que establece el OSI. Existen descripciones del protocolo TCP/IP que definen de tres a cinco niveles. La Figura 7 representa un modelo de cuatro capas TCP/IP y su correspondencia con el modelo de referencia OSI. Los datos que son enviados a la red recorren la pila del protocolo TCP/IP desde la capa más alta de aplicación hasta la más baja de acceso a red. Cuando son recibidos, recorren la pila de protocolo en el sentido contrario. Durante estos recorridos, cada capa añade o sustrae cierta información de control a los datos para garantizar su correcta transmisión.

Como esta información de control se sitúa antes de los datos que se transmiten, se llama cabecera (header). El encapsulado es el proceso cuando cada capa añade una cabecera a los datos que se envían a la red. Si en vez de transmitir datos se trata de recibirlos, el proceso sucede al revés. Cada capa elimina su cabecera correspondiente hasta que quedan sólo los datos. En teoría cada capa maneja una estructura de datos propia, independiente de las demás, aunque en la práctica estas estructuras de datos se diseñan para que sean compatibles con las de las capas adyacentes. Se mejora así la eficiencia global en la transmisión de datos.

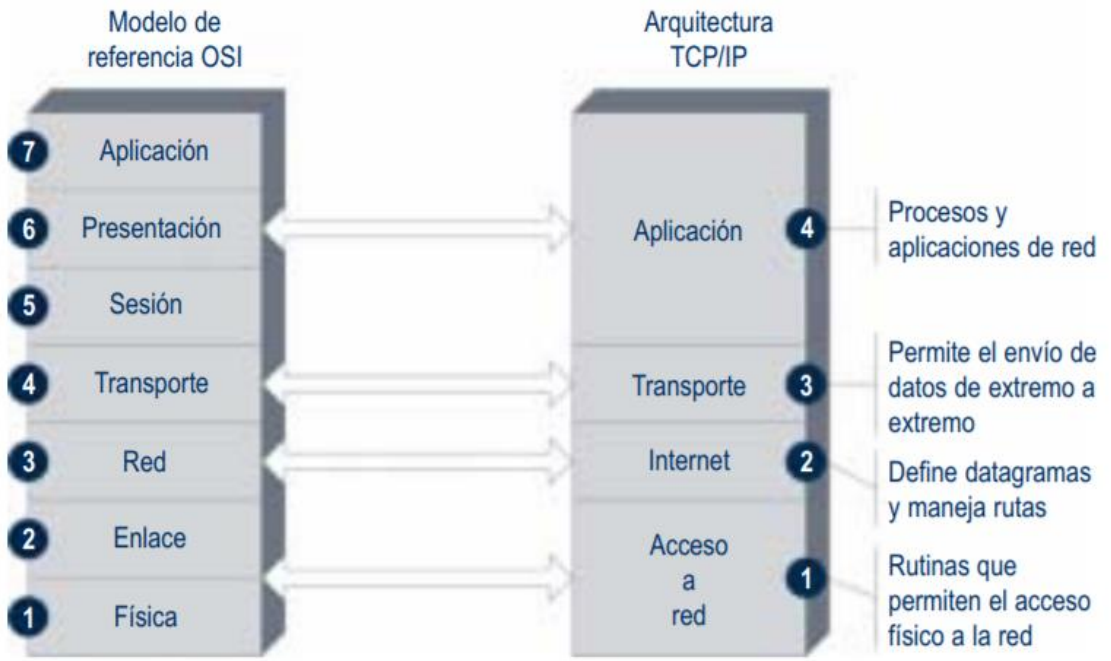


Figura 7. Correspondencia del modelo OSI con TCP/IP

3.7. ESTADO DE VELERO

3.7.1. DEFINICIONES

- TWD: True Wind Direction (Dirección real de viento), típicamente en sentido de las agujas del reloj de 0 a 360 grados, 0 el viento viene del norte
- AWA: Apparent wind angle (Angulo aparente del viento), típicamente en sentido de las agujas del reloj de -180 a 180 grados, ángulo medido (sin corregir) entre la línea central del bote y el viento entrante.
- TWA: True Wind Angle (Angulo real del viento), típicamente en sentido de las agujas del reloj de -180 a 180 grados, ángulo verdadero (correcto) entre la línea central del bote y el viento entrante.
- AWS: Apparent wind speed (Velocidad aparente del viento), medido y sin corregir de la velocidad del viento en nudos.
- TWS: True Wind Speed (Velocidad real del viento), medida correcta de la velocidad del viento en nudos.
- SOG : Speed Over Ground (Velocidad sobre tierra)
- COG : Course Over Ground (Curso sobre tierra)

3.7.2. DIRECCIÓN Y ÁNGULO DEL VIENTO REAL Y APARANTE

Un velero en movimiento experimentará un viento aparente que es una combinación del viento verdadero y el viento inminente creado por el movimiento del barco. Para un velero el cálculo del TWS y TWA es importante para tener una mejor navegación.

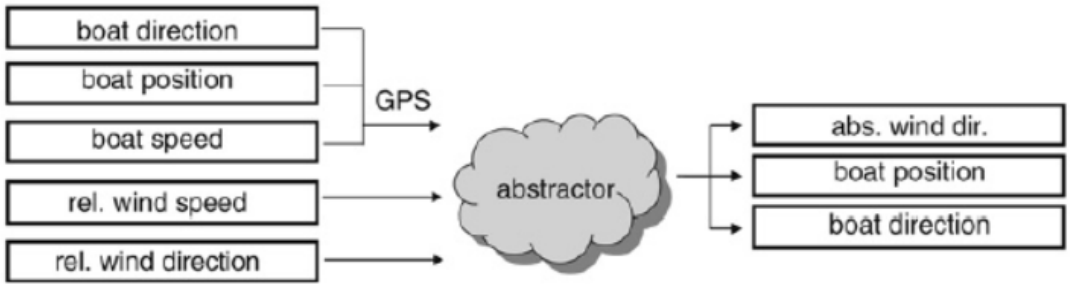


Figura 8. Procesamiento de dato de los sensores.

Considerando la figura 9, el bote se mueve con dirección hacia la izquierda a la velocidad SB. Un viento aparente viene con un ángulo θ de la proa, su velocidad es indicada por la longitud del vector SAW. Si el barco no se mueve, el viento aparente coincidirá con el viento verdadero, pero a medida que el barco gana velocidad, el viento aparente cambia para que venga de una dirección más cercana al movimiento del barco hacia delante.

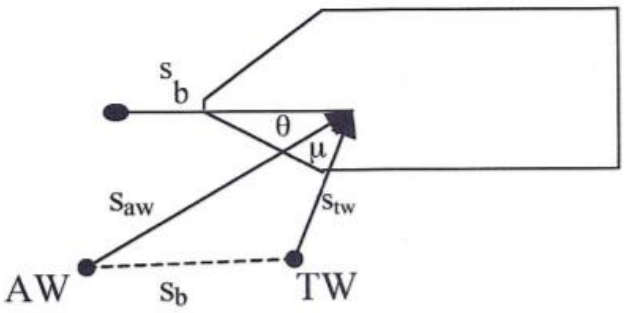


Figura 9. Dirección del viento verdadero y el viento aparente en el velero.

La figura 10 muestra el cálculo de la velocidad y dirección del viento real. La posición del barco se muestra en el origen. Los ángulos se calculan en relación con el movimiento del barco. El barco se está moviendo a velocidad SB creando un viento de barco (BW) en la dirección de la popa. El viento aparente (AW) fluye de θ hasta babor a la velocidad de SAW. El viento verdadero (aún desconocido) está fluyendo desde α hasta babor a la velocidad STW.

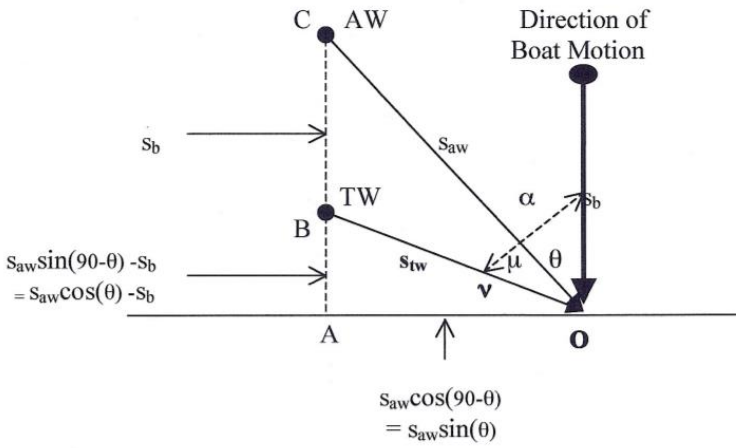


Figura 10. Análisis trigonométrico para viento aparente y viento verdadero

En la figura 10 las flechas muestran la dirección del viento con respecto a la dirección de avance. El barco se mueve verticalmente, creando un flujo de viento hacia abajo sobre él. El origen (O) es la posición actual del barco. El triángulo OCA representa el estado conocido de viento aparente; AC = SAW sin (90 - θ) y AO = SAW cos (90 - θ) son los lados de ese triángulo.

3.7.3. CÁLCULO DE VIENTO REAL

Para la obtención de la información básica para la navegación, se debe desarrollar un cálculo proveniente de la información de los sensores que se especifican a continuación:

- GPS: COG y SOG
- IMU: Magnetómetro
- Sensor de Viento: AWS y AWA

Vector Velocidad de Bote (BS): Está determinada por el GPS (SOG) y Magnetómetro (Hdg)

- SOG consisten en STW (velocidad a través del agua) y la suma de la velocidad de marea porque ambos causan AWS.
- Hdg refleja el punto de la vela y esta es la base de AWA.

$$X_{BS} = Cos\ (90° - Hdg\ ) * SOG$$

$$Y_{BS} = Sin\ (90° - Hdg\ ) * SOG$$

Vector del ángulo de inclinación (LWY): En la Figura 11 sé que diferenciar las dos situaciones que se pueden presentar debido al ángulo de inclinación.

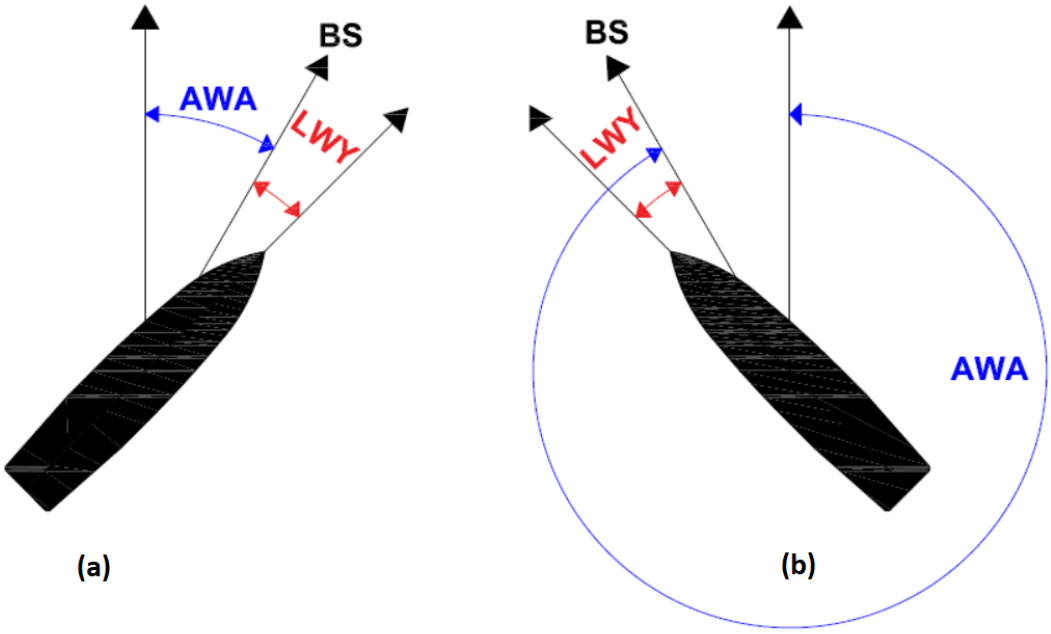


Figura 11. Ángulo de inclinación a) Viento a Estribor y b) Viento a babor

- Viento a estribor (0° > AWA > 180°)
- Viento a babor (180° > AWA > 360°)

$$Estribor \rightarrow LWY = AWA + 90°$$

$$Babor \rightarrow LWY = AWA - 90°$$

$$X_{LWY} = Cos(90° - AWA + LWY) * Tan(LWY) * SOG$$

$$Y_{LWY} = Sin(90° - AWA + LWY) * Tan(LWY) * SOG$$

Vector AWS

$$X_{AWS} = Cos(90° - Hdg + AWA + 180°) * AWS$$

$$Y_{AWS} = Sin(90° - Hdg + AWA + 180°) * AWS$$

Vector TWS: Está determinado por la suma de los vectores AWS, BS y LWY.

$$TWS = \sqrt{(X_{AWS} + X_{BS} + X_{LWY})^2 + (Y_{AWS} + Y_{BS} + Y_{LWY})^2}$$

$$TWA = Tan\left(\frac{Y_{AWS} + Y_{BS} + Y_{LWY}}{X_{AWS} + X_{BS} + X_{LWY}}\right)$$

4. HARDWARE UTILIZADO

4.1. PLACA COMPUTADORA: BEAGLEBONE BLACK – BBB

La BeagleBone Black es una placa programable de bajo coste que encuentra su punto fuerte en su gran versatilidad y en su capacidad para convertirse en un ordenador de bajo consumo de potencia con un sistema operativo en base Linux. El procesado de la BBB incorpora un acelerador de hardware de operaciones de punto flotante. Esto es clave para ejecutar de forma eficiente el núcleo de Linux y abre un nuevo abanico de posibilidades de desarrollo de sistemas empotrado sobre Linux.

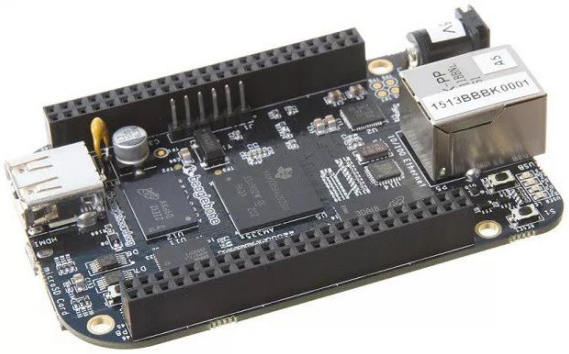


Figura 12. BeagleBone Black Rev C

Su procesador ARM-A8 de 1GHz y su memoria RAM DDR3 de 512 MB la convierten en una unidad de proceso realmente potente en relación con su tamaño. Sin embargo, su mayor potencial se encuentra en su capacidad de conexión gracias a sus 92 pines de entrada/salida, compatibles con buses de comunicación como I2C, UART y SPI, y a sus puertos Ethernet, USB y mini HDMI.

La BBB también ofrece una unidad de memoria flash interna de 4GB y la posibilidad de ampliar su capacidad mediante una ranura para tarjetas microSD. Las principales características de la placa podemos observarlas en la Figura 13:

Feature		
Processor	Sitara AM3359AZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	2GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	
Power	Refer to Section 6.1.7	

Figura 13. Características BeagleBone Black

4.2. CONVERSOR ANALÓGICO DIGITAL (ADC)

El desarrollo de los microprocesadores y procesadores digitales de señal (DSP), ha permitido realizar tareas que durante años fueron hechas por sistemas electrónicos analógicos. El objetivo básico de un ADC es transformar una señal eléctrica análoga en un número digital equivalente. De la misma forma, un conversor digital analógico (DAC) transforma un número digital en una señal eléctrica análoga.

El diagrama de bloques de la Figura 14 muestra la secuencia desde que la variable física entra al sistema hasta que es transformada a señal digital (código binario). Para dicha señal ingrese al convertidor análogo - digital, ésta debe ser muestreada, es decir, se toman valores discretos en instantes de tiempo de la señal análoga. Matemáticamente es el equivalente a multiplicar la señal análoga por una secuencia de impulsos de periodo constante. Como resultado se obtiene un tren de impulsos con amplitudes limitadas por la envolvente de la señal analógica.

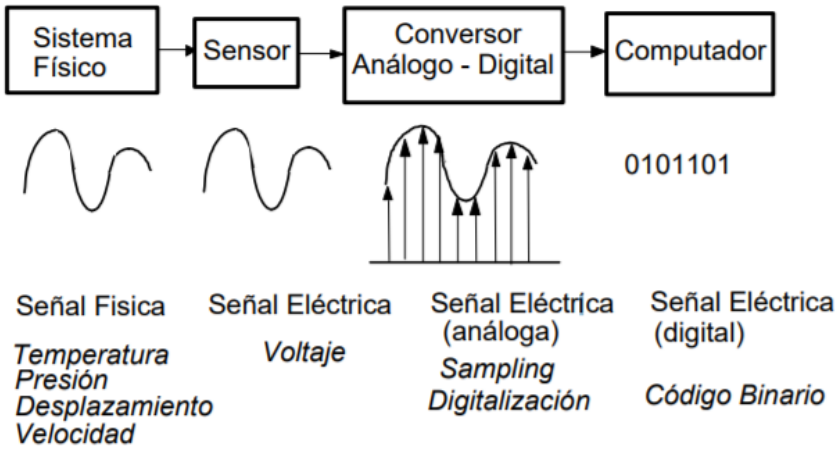
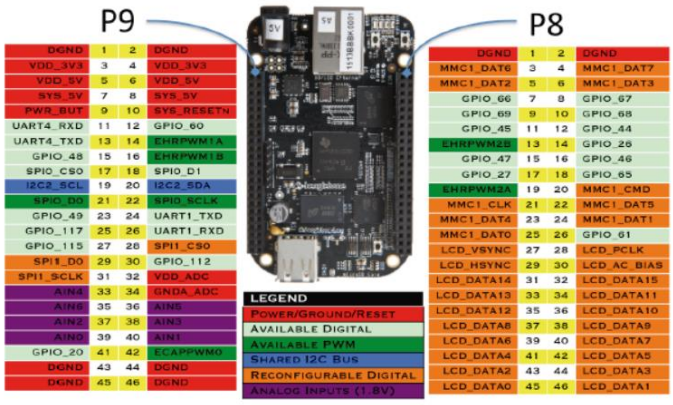


Figura 14. Conversión analógico – digital

Fuente: (Huircan, 20XX)

En la Figura 15 observamos la disposición de los pines dedicados a entrada analógica (desde el P9_32 al P9_40). Estos pines soportan únicamente la función (modo) de conversión analógico/digital, es decir, no hay asignadas diferentes funciones en el mismo pin, por lo que no será necesario configurar el modo en el "pinmux" (multiplexado de los pines) para su utilización, aunque sí que es necesario cargar un "overlay" específico para que los pines estén disponibles y los conversores A/D habilitados.

La BBB tiene 7 canales de conversión analógico/digital de 12 bits que trabajan en el rango 0V y 1.8V. Esto quiere decir que para representar una señal analógica disponemos de 2^12 =4096 posibilidades numéricas, que en el rango de operación corresponde a una resolución de unos 0.44mV.



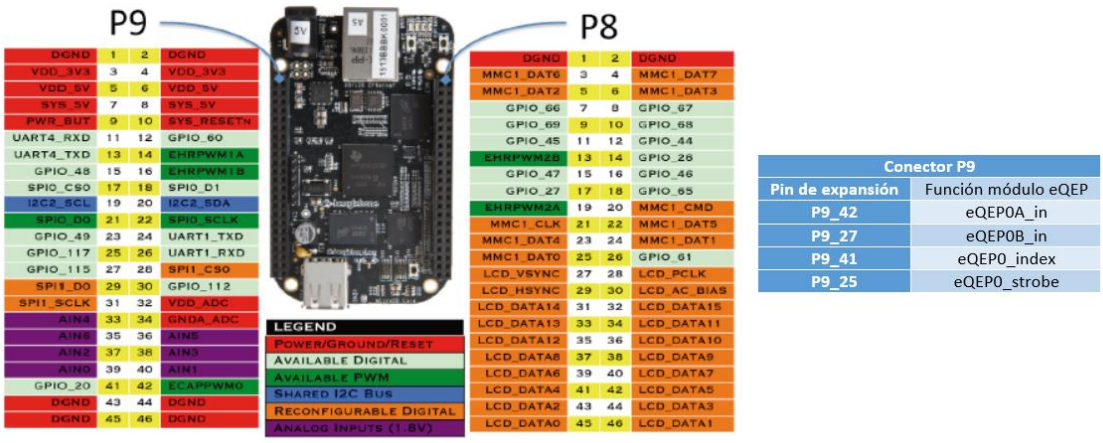
Conector P9	
Pin de expansión	Función ADC
P9_32	VDD_ADC (1.8V)
P9_33	AIN_4
P9_34	GND_ADC
P9_35	AIN_6
P9_36	AIN_5
P9_37	AIN_2
P9_38	AIN_3
P9_39	AIN_0
P9_40	AIN_1

Figura 15. Asignación de los pines de expansión de la conversión analógico-digital

4.3. MÓDULO eQEP

El módulo eQEP **(**Enhanced Quadrature Encoder Pulse) se utiliza para la interfaz directa con encoder incrementales lineales o rotativos para obtener información de posición, dirección y velocidad de una máquina giratoria para su uso en un sistema de movimiento y control de posición de alto rendimiento. Se puede utilizar el módulo eQEP que incluye el chip Sitara, en un modo simple que sirve para medir frecuencia.

En la figura 16 observamos la disposición de los pines que pueden ser utilizados por el módulo eQEP, para su utilización es necesario cargar un "overlay" específico para que los pines estén disponibles y el módulo eQEP se encuentre habilitado.



Conector P9	
Pin de expansión	Función módulo eQEP
P9_42	eQEP0A_in
P9_27	eQEP0B_in
P9_41	eQEP0_index
P9_25	eQEP0_strobe

Figura 16. Asignación de los pines de expansión para el módulo eQEP

4.4. I.M.U.: BNO055

La IMU BNO055 se trata de un "System in Package" (SiP), integrando un acelerómetro triaxial de 14-bits, un giróscopo con un rango de ± 2000 grados por segundo, un sensor geomagnético triaxial y un microcontrolador M0+ de 32-bit, siendo ejecutados en un software de fusión de BOSCH y un empaquetado único. Para una integración óptima de la IMU BNO055 se ha equipado con una interfaz de comunicación I2C y UART.

A continuación, se detallan sus especificaciones técnicas:

- Acelerómetro de tres ejes con un fondo de escala variable entre: ±2g, ±4g, ±8g y ±16g.
- Giróscopo de tres ejes con un rango variable entre ±125 °/s a ± 2000 °/s.
- Magnetómetro de tres ejes con un fondo de escala de ±1300 µT (eje x e y) y ±2500 µT (eje z).
- Rango de tensión de alimentación: 2.4V – 3.6V.

Figura 17. I.M.U. BNO055

4.5. GPS: ULTIMATE GPS

El dispositivo GPS, de la empresa Addafruit, está construido a base del chipset MTK3339, un módulo GPS de alta calidad, que puede rastrear hasta 22 satélites en 66 canales, posee un excelente receptor de alta sensibilidad de (-165 dB tracking), lo que hace que tenga una gran capacidad de registro de datos, además se le puede incorporar una antena externa y una pila para guardar la información de satélites e inicialización del módulo.

A continuación, se detalla sus especificaciones técnicas:

- Satélites: 22 de seguimiento y 66 de búsqueda.
- Frecuencia de actualización de 1 a 10 Hz.
- Precisión de posición: 1.8 metros.
- Precisión de velocidad: 0.1 metros/seg.
- Sensibilidad de adquisición: -145 dBm.
- Sensibilidad de seguimiento: -165 dBm.
- Voltaje de alimentación: 3V – 5.5 V.
- Consumo de corriente: 25 mA en seguimiento y 20 mA durante la navegación.
- Salida: NMEA 0183,9600 baudios por defecto.

Figura 18. Ulimate GPS V3

5. COMUNICACIÓN DE LA BEAGLEBONE BLACK CON LOS DISPOSITIVOS

Para poder realizar la comunicación de la IMU y GPS con la BeagleBone Black y utilizar todas las librerías del fabricante (Adafruit), se ha partido de la documentación proporcionada por GitHub, y realizado una portabilidad de dichas librerías y establecer las funciones equivalentes para nuestra plataforma. De caso muy similar se ha realizado el uso de las librerías desarrolladas para el módulo eQEP que ya venían definidas para la BeagleBone Black del sitio GitHub.

5.1. BNO055

La plataforma GitHub, dispone de la librería para el uso de la IMU que ha sido definida para la plataforma de código abierto Arduino. Desarrollar la portabilidad requirió el uso adicional de la librería I2C provisionada en la asignatura de Sistemas Empotrados del máster, logrando así configurar correctamente el bus I2C y la definición correcta de la plataforma para el desarrollo del entorno de programación.

La portabilidad consistió en la creación de la clase BNO055 que presenta varias funciones para inicializar, calibrar, obtener los valores del acelerómetro, giróscopo, magnetómetro, cuaterniones y de manera similar contiene varias estructuras de datos que nos permiten almacenar los valores requeridos por el usuario y que puedan ser utilizados por otras clases. En este documento se detallarán las funciones más importantes que han sido utilizadas, en los anexos de la memoria se dispone de todas las funciones y variables que se pueden utilizar.

5.1.1. Inicialización de la I.M.U

Para poder realizar la inicialización y calibración del sensor, se ha utilizado las siguientes funciones:

Clase BNO055
➔ BNO055(I2C *i2c); ➔ bool initialize(); ➔ void setMode(unsigned char Opr_Mode); ➔ void getSensor(sensor_t *sensor); ➔ void setSensorOffset(BNO055_offsets *offsets_type); ➔ bool isFullCalibrated();
Clase I2C
➔ I2C(int bus); ➔ unsigned char readByte(unsigned char slave_addr, unsigned char registerAddr); ➔ bool writeByte(unsigned char slave_addr, unsigned char registerAddr, unsigned char value);

El procedimiento para una correcta calibración se la describe en los siguientes pasos:

No.	Función	Descripción
1	BBB::I2C i2cBus(2);	Constructor de la clase I2C, indicando el bus de comunicación que se usará.
2	BBB::BNO055 imu(&i2cBus);	Constructor de la clase BNO055, que proporciona el objeto de comunicación I2C.
3	if (!imu.initialize()) {printf("Ooops, no BNO055 detected, Check your wiring or I2C ADDR!\n"); }	Secuencia de inicialización de la IMU, se comprueba si el bus i2c es válido y si se ha detectado a la IMU, además se hacen configuraciones para la IMU.
4	imu.setSensorOffset(&offsets_cal); printf("\n\nCalibration data loaded into BNO055\n");	Se setean los valores de calibración, previamente obtenidos, para el acelerómetro, magnetómetro y giróscopo.
5	if (foundCalib && !imu.isFullCalibrated())	Se realiza una comprobación interna de los valores de calibración a la IMU, para

		poder determinar si fueron asignados correctamente.

5.1.2. Conexión BNO055 con BeagleBone Black

Para conectar la BNO055 al bus I2C de la BeagleBone Black hay que realizar la siguiente conexión:

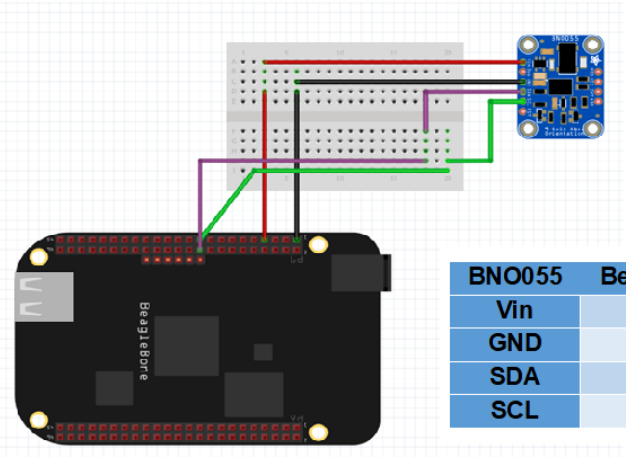


Figura 19. Conexiones entre BeagleBone Black y BNO055

Para determinar si la conexión es correcta, se ejecutó el comando i2cdetect –y –r 2. Como se puede ver en el bus i2c-2 tenemos un dispositivo con la dirección 28 que se corresponde con el módulo BNO055 que se ha conectado correctamente.

```
root@BBG:~# i2cdetect -y -r 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- 28 -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figura 20. Respuesta al comando i2cdetect

5.1.3. Funcionamiento BNO055

Dado que el objetivo del sensor es expresar la orientación del velero con respecto a un eje de referencia fijo. Partiendo de uno de los ejes coordenados que representan el sistema de

referencia del sensor BNO055, las rotaciones respecto a estos ejes provocaron el cambio en la orientación del objeto.

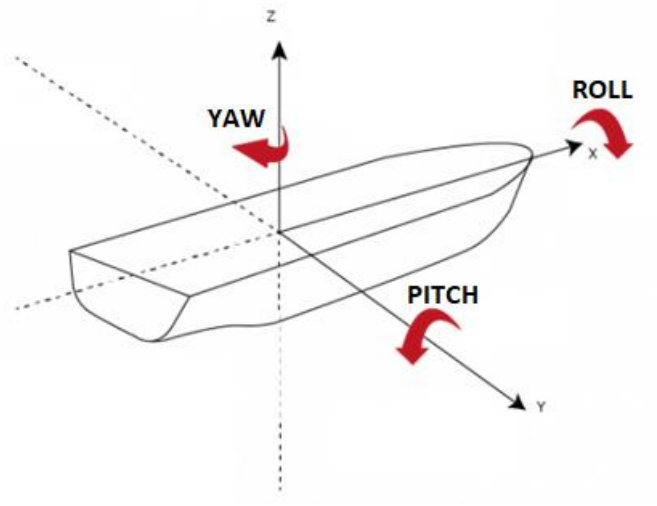


Figura 21. Ejes roll, pitch y yaw del velero

Existen varias representaciones matemáticas para definir la orientación del objeto respecto al sistema de referencia, estos se describen a continuación:

Ángulos Tait-Bryan (Roll, Pitch, Yaw) y ángulos de Euler

Una transformación de un marco coordinado a otro se define por tres rotaciones sucesivas sobre los diferentes ejes. Los ángulos roll, pitch y yaw representan las tres rotaciones sobre los ejes X, Y Z, respectivamente. Los ángulos de Euler son una representación similar, cambiando los ejes sobre los que se realizan las rotaciones y el orden en que se tienen en cuenta.

La principal diferencia reside en que para obtener los ángulos de Tait-Bryan se realizan tres rotaciones sobre tres ejes distintos, mientras que para Euler dos de las rotaciones se realizan sobre el mismo eje.

Sin embargo, como ambas son formas de expresar la orientación de un cuerpo, existe una relación entre ellos; pudiéndose expresar unos en función de otros mediante una matriz de transformación.

Estos ángulos se prefieren en aeronáutica porque le asignan un ángulo de inclinación cero a un avión en horizontal, a diferencia de los ángulos de Euler, que le asignarían PI/2. La principal ventaja de utilizar estos ángulos para realizar un control de orientación es que definen una rotación de forma única alrededor de cada uno de los ejes intrínsecos del objeto.

Cuaterniones

La representación de la orientación mediante cuaterniones es una representación de cuatro parámetros basada en la idea de que una transformación de un sistema de referencia a otro puede ser efectuada por una única rotación sobre un vector $\vec{\mu}$ definido en el sistema de referencia fijo. El cuaternión, $\overrightarrow{q}$ es un vector de cuatro elementos que son función de este vector y de la magnitud de la rotación:

$$\vec{q} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \left(\frac{\mu_x}{\mu}\right)\sin\frac{\theta}{2} \\ \left(\frac{\mu_y}{\mu}\right)\sin\frac{\theta}{2} \\ \left(\frac{\mu_z}{\mu}\right)\sin\frac{\theta}{2} \end{bmatrix}$$

Donde μ_x, μ_y y μ_z son las componentes del vector $\vec{\mu}$ y μ el módulo de dicho vector. El parámetro θ representa el valor de la rotación sobre el vector $\vec{\mu}$.

Clásicamente se ha optado por un algoritmo que va actualizando la matriz de rotación o los cuaterniones, y es este último el que más aparece en las últimas tendencias. Los ángulos RPY y de Euler presentan más inconvenientes, puesto que en la resolución de las correspondientes ecuaciones de propagación en el tiempo aparecen indeterminaciones, debido a que una misma orientación se puede expresar con distintos ángulos.

Los cuaterniones se presentan como la solución más adecuada, por ser tan sólo cuatro los parámetros a actualizar y por presentar menores errores en la computación, según diversos estudios.

Para poder trabajar con los datos de la IMU, es decir, la obtención de los valores de roll, pitch y yaw, se deberán implementar las siguientes funciones:

No.	Función	Descripción
1	Q = imu.getQuaternion();	Obtención de los cuaterniones de alfa, beta y gama.
2	rpy = imu.getRollPitchYaw(Q);	Cálculo de los ángulos para la determinación de roll, pitch y yaw.

De este modo se obtiene el valor del cuaternión, un vector de cuatro elementos $[q_1, q_2, q_3, q_4]$. Los cuaterniones permiten abstraer rotaciones y traslaciones con cierta simplicidad, permitiendo la obtención de la orientación relativa entre sistemas de coordenadas, de modo más sencillo y rápido que por ejemplo con el uso de matrices. Para poder utilizar los valores obtenidos, primero hay que normalizarlos:

$$q_{norm} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$$

$$q_{1norm} = \frac{q_1}{q_{norm}}$$

Con el cálculo de la norma, se realiza la normalización de los valores del cuaternión, llegando a tener el vector normalizado:

$$q = [q_{1_{norm}}, q_{2_{norm}}, q_{3_{norm}}, q_{4_{norm}}]$$

Con los valores normalizados, se pueden obtener los valores de roll, pitch y yaw, mediante las siguientes relaciones:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} atan\left[\frac{2*(q_0q_1 + q_2q_3)}{1-2*(q_1^2+q_2^2)}\right] \\ asin[2*(q_0q_2 - q_3q_1)] \\ \text{atan}\left[\frac{2*(q_0q_3 + q_1q_2)}{1-2*(q_2^2+q_3^2)}\right] \end{bmatrix}$$

5.1.4. Calibración BNO055

El software interno de la I.M.U. ejecuta en segundo plano un algoritmo de calibración para sus tres sensores (acelerómetro, giróscopo y magnetómetro) para eliminar los offsets, se deben efectuar algunos pasos preliminares para garantizar esta calibración automática. El acelerómetro y giróscopo son menos susceptibles a perturbaciones externas, por lo tanto, su offset es despreciable. Mientras que el magnetómetro es susceptible al campo magnético externo y por lo tanto para asegurar la exactitud del rumbo, deben tomarse en cuenta los pasos de calibración. Se ha desarrollado el archivo ***imuCal.c*** el cual servirá para realizar la calibración.

Calibración del Acelerómetro

- Se debe colocar al dispositivo en 6 diferentes posiciones estables durante un periodo de pocos segundos para permitir que el acelerómetro se calibre.
- Hay que asegurarse que hay un movimiento lento entre 2 posiciones estables.
- Las 6 posiciones estables podrían estar en cualquier dirección, pero se debe asegurar de que el dispositivo se encuentre al menos una vez perpendicular al eje X, Y y Z.
- El estado de la calibración del acelerómetro se encuentra en el registro CALIB_STAT.

Calibración del Giróscopo

- Colocar el dispositivo en una sola posición estable durante un periodo de pocos segundos para permitir que el giróscopo se calibre.
- El estado de la calibración del giróscopo se encuentra en el registro CALIB_STAT.

Calibración del Magnetómetro

- Realizar movimientos aleatorios, por ejemplo escribir el número “8” en el aire; hasta que el registro CALIB_STAT indique que está completamente calibrado.

Es necesario ejecutar el archivo ***imuCal.c***, para la calibración de los tres sensores, donde se deben seguir los pasos anteriormente descritos. Se obtiene los siguientes resultados:

```
Found Calibration for this sensor

Please Calibrate Sensor:
X: 0.000000Y: 0.000000Z: 0.000000       ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 0.000000Y: 0.000000Z: 0.000000       ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 359.937500Y: 0.000000Z: 0.000000     ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
X: 271.687500Y: -35.687500Z: -135.750000        ! Sys: 0,       G: 0,   A: 0,   M: 0
```

Figura 22. Calibración de la I.M.U.

```
Fully calibrated!
--------------------------------
Calibration Results: Accelerometer:      65533   65518   5
Gyro:     65534   65533   0
Mag:      295     65498   65288
Accel Radius:     1000
Mag Radius:       757
------------------------------------
Sensor:      BNO055
Driver Ver:    1
Unique ID:     0xA0
Max Value:     0.000000
Min Value:     0.000000
Resolution:    0.010000
------------------------------------

------------------------------------
System Status: 0x05
Self Test: 0x0F
System Error:  0x00
```

Figura 23. Datos de la correcta calibración de la I.M.U.

Los datos de la calibración, obtenidos de la Figura 23 deberán ser copiados al archivo ***tfmAPP.c*** para que sean usados por en el programa principal del proyecto.

5.1.5. Experimentos de funcionamiento de la I.M.U.

Una vez realizado la correcta calibración de la I.M.U., se ejecuta en la BeagleBone Black el programa ***imu01.c***, para observar el funcionamiento de la I.M.U. (ángulos de Euler, datos del giróscopo y magnetómetro, roll, pitch y yaw), que aparecen en la Figura 24.

```
-----------------------------------------------------------------
EULE->   x=0.125         y=-0.062        z=-5.250
GYRO->   x=2.250         y=2.000         z=0.500
MAG ->   x=-61.250       y=0.062         z=-39.500
Roll->   -5.262          Pitch->         0.128           Yaw->359.784
-----------------------------------------------------------------
-----------------------------------------------------------------
EULE->   x=0.000         y=0.000         z=-5.188
GYRO->   x=1.375         y=0.500         z=0.062
MAG ->   x=-59.750       y=-0.562        z=-39.500
Roll->   -5.219          Pitch->         0.013           Yaw->359.943
-----------------------------------------------------------------
-----------------------------------------------------------------
EULE->   x=0.062         y=0.000         z=-5.250
GYRO->   x=0.000         y=1.562         z=-0.312
MAG ->   x=-60.562       y=-0.250        z=-38.688
Roll->   -5.268          Pitch->         0.033           Yaw->359.894
-----------------------------------------------------------------
-----------------------------------------------------------------
EULE->   x=0.125         y=-0.062        z=-5.125
GYRO->   x=0.500         y=0.000         z=-4.000
MAG ->   x=-60.562       y=-0.188        z=-38.688
Roll->   -5.185          Pitch->         0.080           Yaw->359.808
-----------------------------------------------------------------
-----------------------------------------------------------------
EULE->   x=93.188        y=-0.062        z=-5.125
GYRO->   x=-1.938        y=-0.562        z=0.188
MAG ->   x=-60.562       y=-0.500        z=-37.875
Roll->   -5.154          Pitch->         1.335           Yaw->266.779
-----------------------------------------------------------------
```

Figura 24. Representación de la orientación de la I.M.U.

5.2. GPS

La plataforma GitHub, dispone de la librería para el uso del GPS que ha sido definida para la plataforma de código abierto Arduino. Desarrollar de la portabilidad de dicha requirió el uso adicional de la librería UART provisionada en la asignatura de Sistemas Empotrados del máster, logrando así configurar correctamente el bus serial y la definición correcta de la plataforma para el desarrollo del entorno de programación.

La portabilidad consistió en la creación de la clase GPS que presenta varias funciones para inicializar, calibrar a GPS, para que trabaje en varias velocidades de entrega de mensajes, poder definir qué mensajes se requiere que envíe el módulo GPS y al paso de los mensajes NMEA a valores útiles para la navegación del velero y de igual manera contiene varias estructuras de datos que nos permiten almacenar los valores requeridos por el usuario y que puedan ser utilizados por otras clases, en este documento se detallarán las funciones más importantes que han sido utilizadas, en los anexos de la memoria se dispone de todas las funciones y variables que se pueden utilizar.

5.2.1. Inicialización del GPS

Para poder realizar la inicialización del GPS, se ha seguido la siguiente secuencia de comandos para lograr una inicialización correcta del mismo:

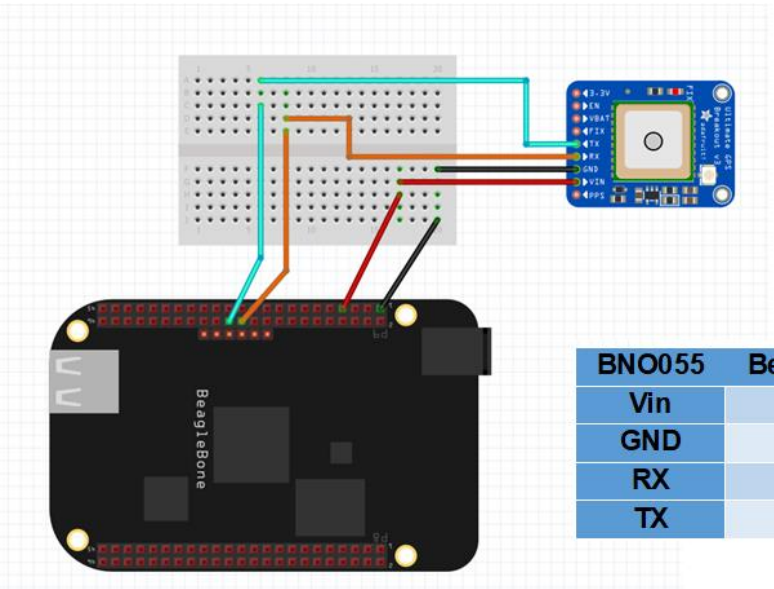
Inicialización GPS
➔ myUart->write(BAUD_9600, strlen(BAUD_9600)); ➔ myUart->write(UPDATE_200_msec, strlen(UPDATE_200_msec));

➔ myUart->write(MEAS_200_msec, strlen(MEAS_200_msec)); ➔ myUart->write(GPRMC_GPGGA, strlen(GPRMC_GPGGA)); ➔ myUart->flush(BBB::input); ➔ void common_init(void);
Class UART
➔ UART (uartName uart, baudRate uartBaud, parity uartParity, stopBits uartStopBits, characterSize uartCharSize); ➔ int write(char *writeBuffer, size_t size); ➔ bool flush(direction whichDirection);

Para la inicialización se ha determinado una velocidad de 9600 baudios para la comunicación serial, un refresco de los mensajes de 200 milisegundos y que solo requerimos los mensajes GPRMC Y GPGGA del GPS.

5.2.2. Conexión GPS con BeagleBone Black

Para conectar correctamente el módulo GPS al bus UART de la BeagleBone Black hay que realizar la siguiente conexión:



BNO055	BeagleBone Black
Vin	Pin 9_07
GND	Pin 9_01
RX	Pin 9_24
TX	Pin 9_26

Figura 25. Conexiones entre BeagleBone Black y GPS

5.2.3. Funcionamiento GPS

Una vez recibido un mensaje del GPS, se debe realizar los siguientes pasos para determinar si es un mensaje GPRCM o GPGGA, y posteriormente obtener la información que se utilizara de ese mensaje y eso se logra siguiendo los siguientes pasos:

No.	Función	Descripción
1	int count = myUart->read(buff, sizeof(buff));	Leer de la comunicación un mensaje y se determina el buffer y el tamaño del buffer recibido.
2	if (!gps.parse(buff)) { printf("Ha fallado el recibir el mensaje buffer NMEA\n");}	Determinar los distintos valores que contiene el mensaje NMEA enviado por el GPS, comprobando si el mensaje es correcto.
3	if (gps.fix || gps.fixquality == 1){ printf("Imprimir valores del GPS\n");}	Visualizar valores útiles de navegación cuando el sensor se encuentra calibrado, ya sea en el mensaje GRPMC y GPGGA.

5.2.4. Pruebas de funcionamiento de GPS

El módulo de GPS, se calibra automáticamente solo se le ha especificado que el tiempo de envió de los mensajes NMEA y que únicamente queremos el GPRMC y GPGGA para tener una navegación adecuada. En la Figura 26, se puede observar como el mensaje que se recibe la placa BeagleBone Black del GPS y se refrescan los valores de navegación dependiendo si es un mensaje GPRMC o GPGGA.

```
GPS Calibrandose
-------------------------------
Time: 0 : 0 : 0
Year : 0, Month: 0, Day: 0
Fix : 0
Fix quality : 0
-------------------------------
$GPRMC,165102.000,A,3928.3189,N,00020.2644,W,0.14,165.02,030917,,,A*71
-------------------------------
Time: 16 : 51 : 2
Year : 17, Month: 9, Day: 3
Fix : 1
Fix quality : 0
-------------------------------
Latitude : 3928.318848   Longitude : 20.264400
Latitude Degrees : 39.471981     Longitude Degree : -0.337740
SOG (knots) :  0.140000          COG (Degrees) : 165.020004
Altitude : 0.000000      Satellite : 0
-------------------------------------------------
-------------------------------------------------
$GPGGA,165143.000,3928.3189,N,00020.2666,W,1,05,2.29,58.7,M,51.9,M,,*44
-------------------------------
Time: 16 : 51 : 43
Year : 17, Month: 9, Day: 3
Fix : 1
Fix quality : 1
-------------------------------
Latitude : 3928.318848   Longitude : 20.266596
Latitude Degrees : 39.471981     Longitude Degree : -0.337777
SOG (knots) :  0.160000          COG (Degrees) : 298.609985
Altitude : 58.700001     Satellite : 5
-------------------------------------------------
```

Figura 26. Funcionamiento GPS

5.3. eQEP

La clase eQEP se la ha obtenido de la plataforma y cuenta con varios métodos y funciones para el uso correcto de este módulo de la BeagleBone, para nuestro caso en concreto utilizaremos una función que nos permiten la calibración del contador de bits de la posición y otra función que nos entrega el valor de flancos de subida que se han contado durante 1 segundo. Las funciones utilizadas por esta clase son las siguientes:

No.	Función	Descripción
1	void eQEP::resetPositionCounter();	Resetea la cuenta del contador asociada al valor de flancos de subida.
2	uint32_t eQEP::getPosition();	Devuelve cuantos flancos de subida han sido detectados.

5.3.1. Conexión eQEP con BEAGLEBONE BLACK

Para conectar las señales al módulo eQEP de la BeagleBone Black, se ha implementado el siguiente circuito electrónico que consta de un fototransistor para proteger a la placa de desarrollo. En la Figura 27 se dispone de las conexiones que se deben realizar.

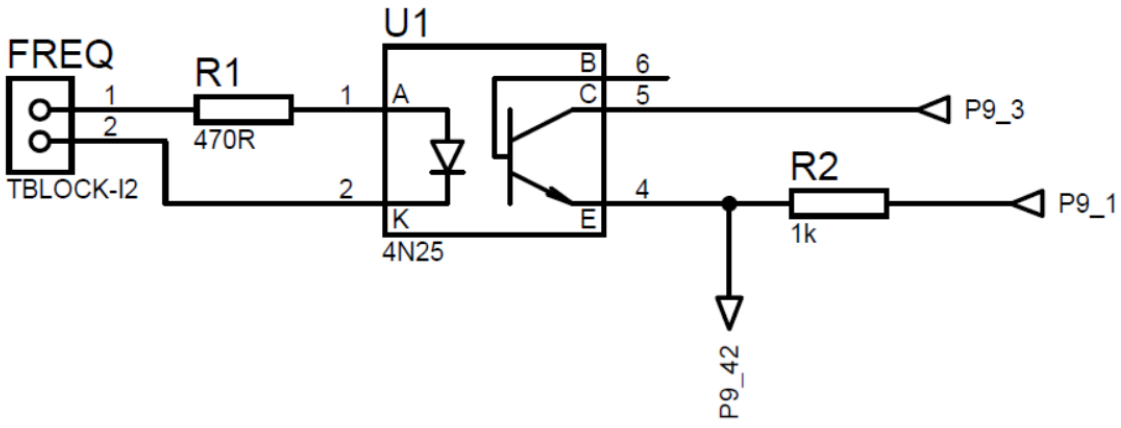


Figura 27. Conexiones entre BeagleBone Black y eQEP

5.3.2. Funcionamiento eQEP

Se realizaron pruebas del módulo eQEP, para determinar la exactitud de la medición que realizar, obteniéndose buenos resultados que se indican en las Figuras

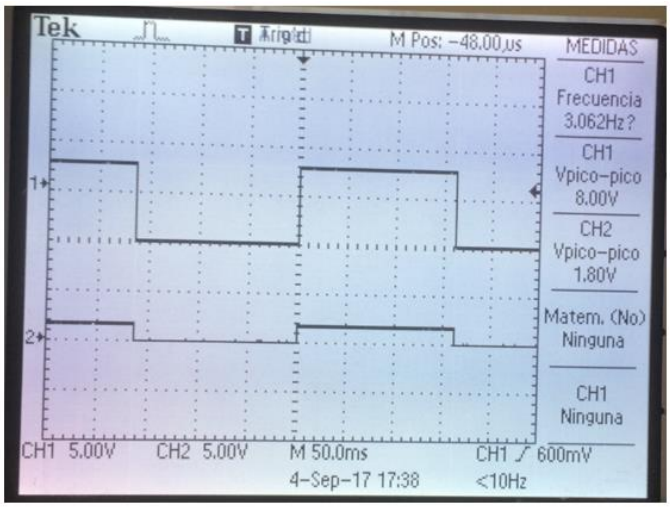


Figura 28. Funcionamiento módulo eQEP frecuencia 3Hz

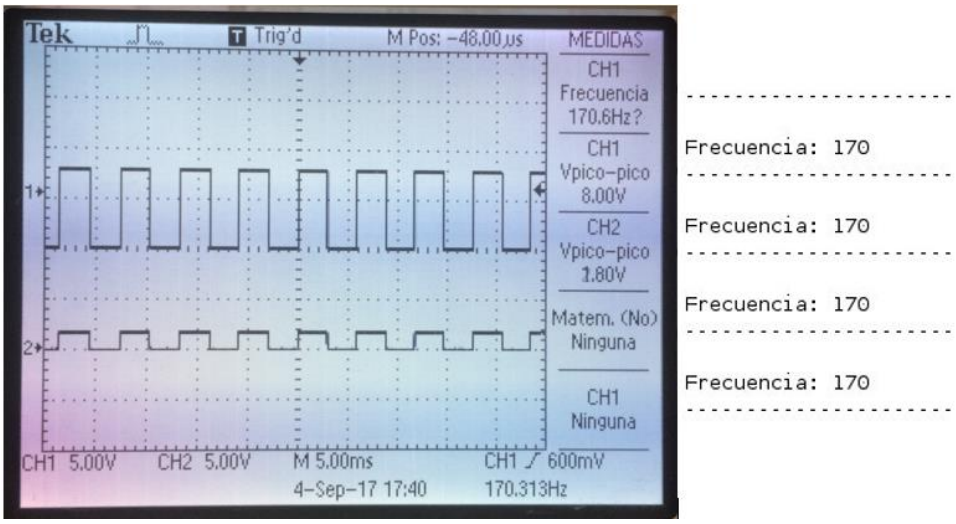


Figura 29. Funcionamiento módulo eQEP frecuencia 170Hz

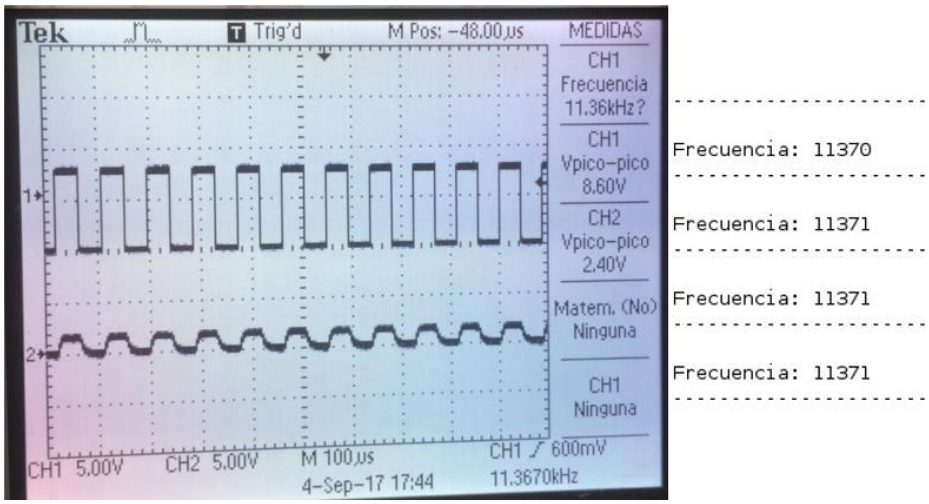


Figura 30. Funcionamiento módulo eQEP frecuencia 11.36 kHz

5.4. CONVERSOR ANALÓGICO DIGITAL

La información proveniente del anemómetro es una señal analógica, por tal motivo se requiere el uso del conversor ADC, a la que se le ha aplicado un filtrado mediante software y un cambio de escalas para poder trabajar con los ángulos de las mismas, las funciones utilizadas fueron proporcionadas por la asignatura de sistemas empotrados del máster. Logrando así un correcto uso y funcionamiento del conversor de la placa BeagleBone Black.

Para utilizar correctamente la información del conversor se han desarrollado e implementado las siguientes funciones:

No.	Función	Descripción
1	valor.sin1=readAnalog(adc1);	Retorna el valor del conversor entre 0 a 4096.
2	float ChangeScale(float x, float in_min, float in_max, float out_min,float out_max);	Con esta función se hace un mapeo de valores entre 0 a 4096 con -1 a 1.
3	void SignalFilter(void *ptr,float temp, float temp2);	Aplicación de un filtrado recursivo exponencial descrito por la siguiente ecuación: $$y[i] = \alpha * y[i-1] + (1-\alpha) * x[i]$$
4	void MoveSignal(void *ptr);	Es preciso mover la señal obtenida por la función readAnalog, llevando así a los valores centrados en 0 en el eje x.
5	void FindMaxMin(void *ptr);	Aquí se pueden encontrar los valores máximos y mínimos de las ondas senoidales.
6	void CalculateAngle(void *ptr);	Debido a que se dispone de señales desfasadas 120 grados, es necesario un análisis de los signos de las señales para poder determinar el ángulo verdadero que se tiene en la señal que no presenta desfase.

5.4.1. Conexión ADC con BeagleBone Black

Para conectar las señales analógicas a la BeagleBone Black, se ha desarrolla el circuito que se encuentra en la Figura 24, donde se limita la señal de entrada a la placa a 1.1 V, para proteger al conversor de la placa.

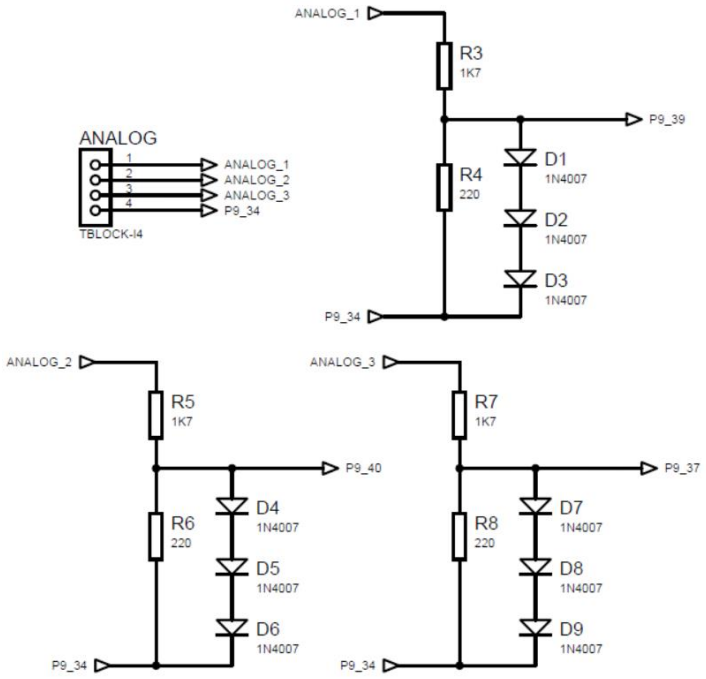


Figura 31. Conexiones entre BeagleBone Black y señales analógicas

5.4.2. Funcionamiento conversor

Para poder validar el funcionamiento del conversor y las funciones de la clase que han realizadas, se ejecutó el archivo ***adctest.c.*** Tal como se indica en el programa, el tiempo de muestro es de 2500 us, durante 5 segundos. El programa genera un fichero datasignalreal.txt con los diferentes resultados. En la Figura 32 se observa la señal senoidal que fue utilizada para comprobar el funcionamiento del conversor.

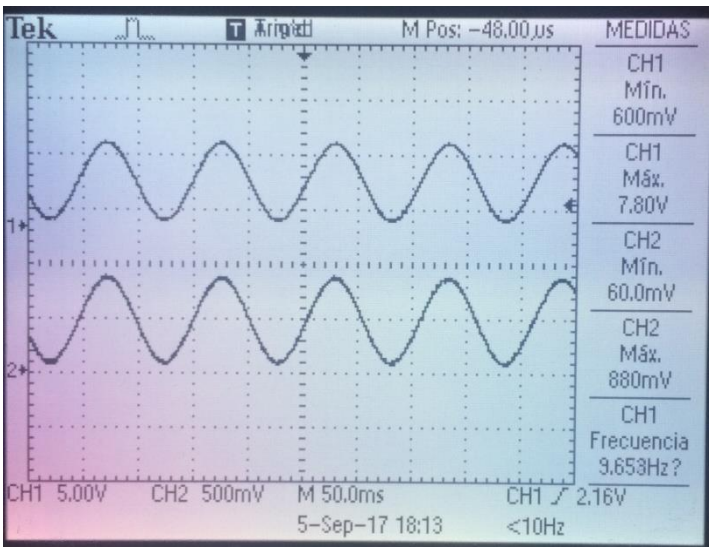


Figura 32. Señal senoidal analógica 10 Hz

Tras la adquisición de los datos, es útil realizar una gráfica de los valores que se han capturado en el fichero de salida. Con la ayuda del programa Matlab se ha desarrollado el programa *readtxtfile.m*, para la visualización del contenido del fichero.

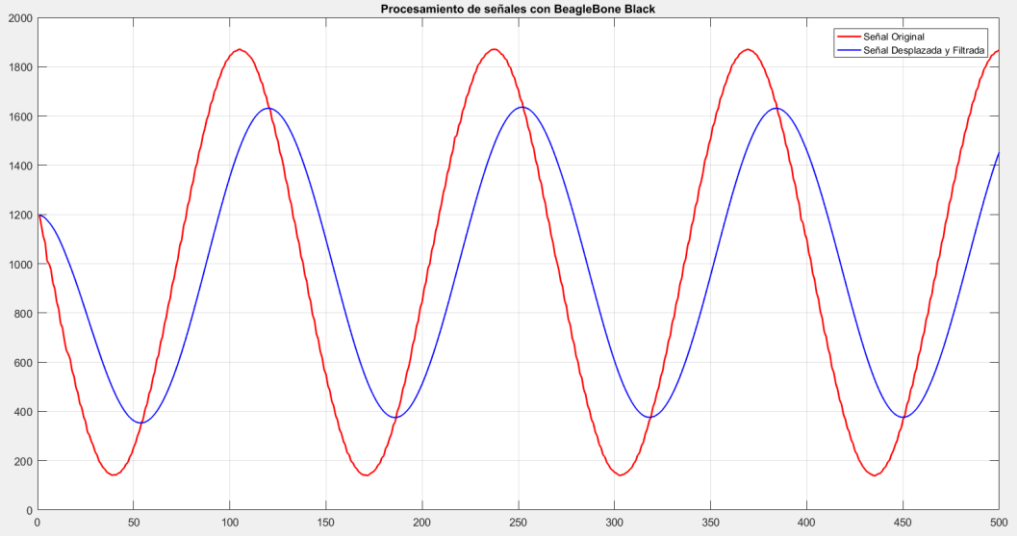


Figura 33. Visualización señales obtenidas con BeagleBone Black

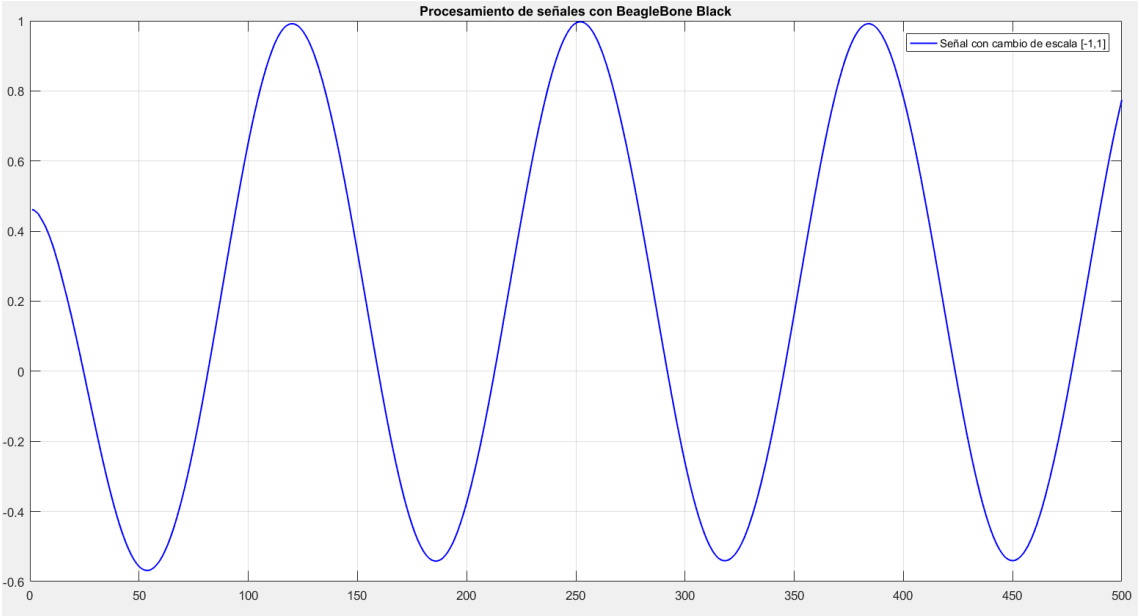


Figura 34. Visualización de señal senoidal procesada por la BeagleBone Black

Para poder determinar el correcto funcionamiento de las funciones desarrolladas para el conversor, se ha simulado en MATLAB las señales generadas por el sensor de viento, que poseen un desfase de 120°, y se les ha incluido un ruido para ver la efectividad del filtro planteado. En la Figura 35 se puede observar la señal con ruido y filtrada que ha sido procesada por la placa de desarrollo.

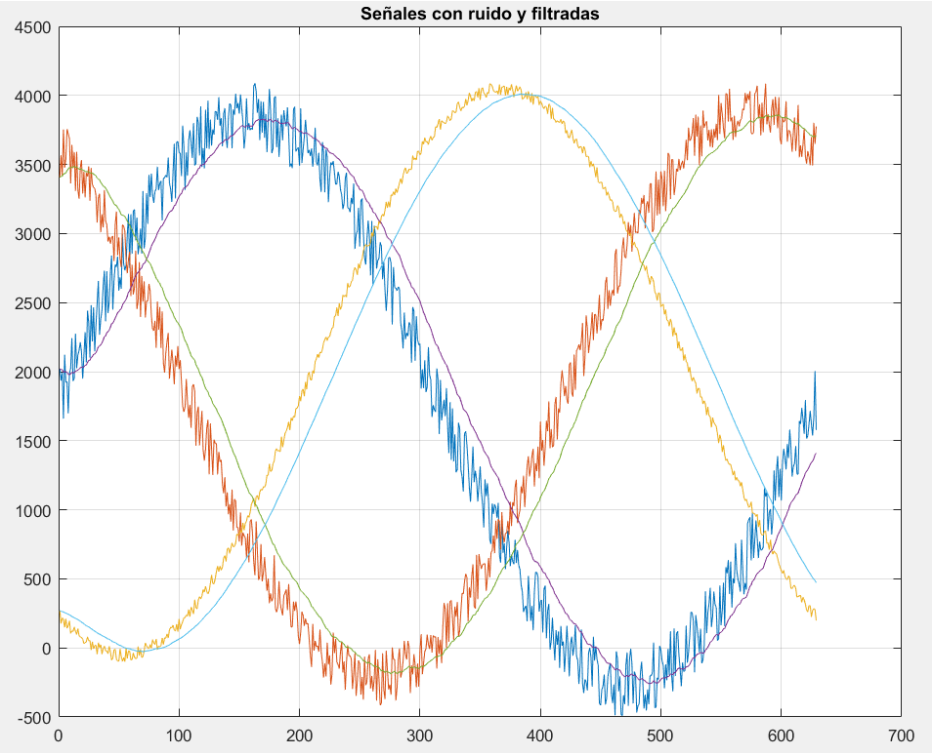


Figura 35. Señal con ruido y señal filtrada procesada por la placa de desarrollo

En la Figura 36 se puede observar las señales que han sido desplazadas en el eje vertical para que posteriormente puedan ser cambiadas de escala entre [-1, 1].

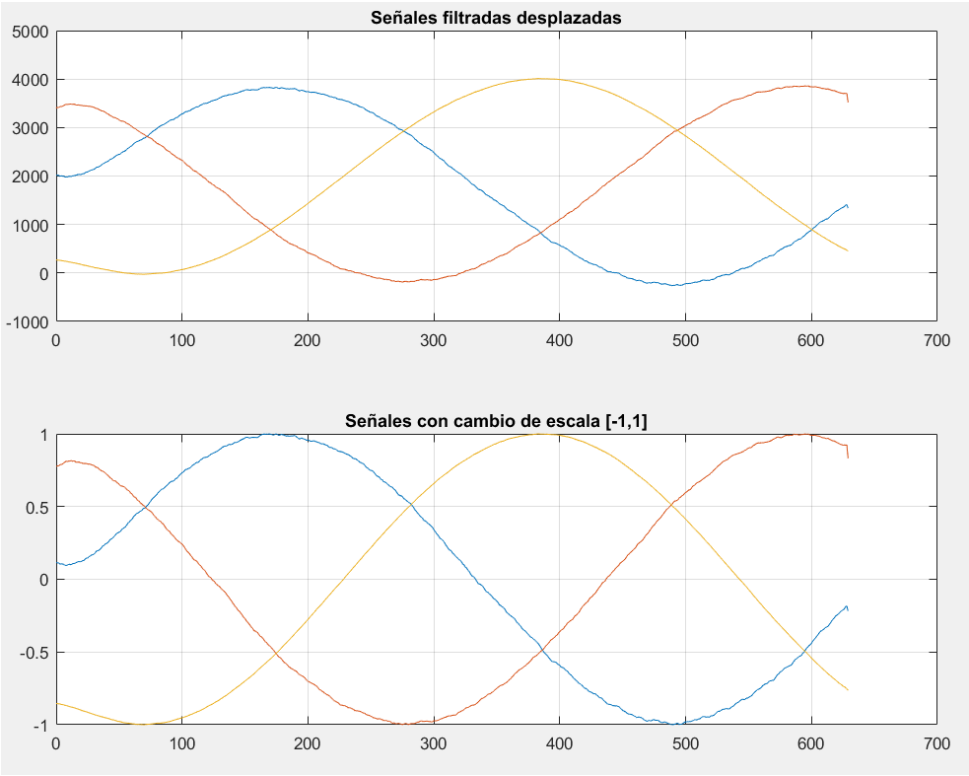


Figura 36. Señales senoidal desplazadas y con cambio de escala

Finalmente en la Figure 37 se puede observar las señales que han sido obtenidas del cálculo de las tres señales senoidales para poder determinar el ángulo de la señal que no presenta ningún desfase.

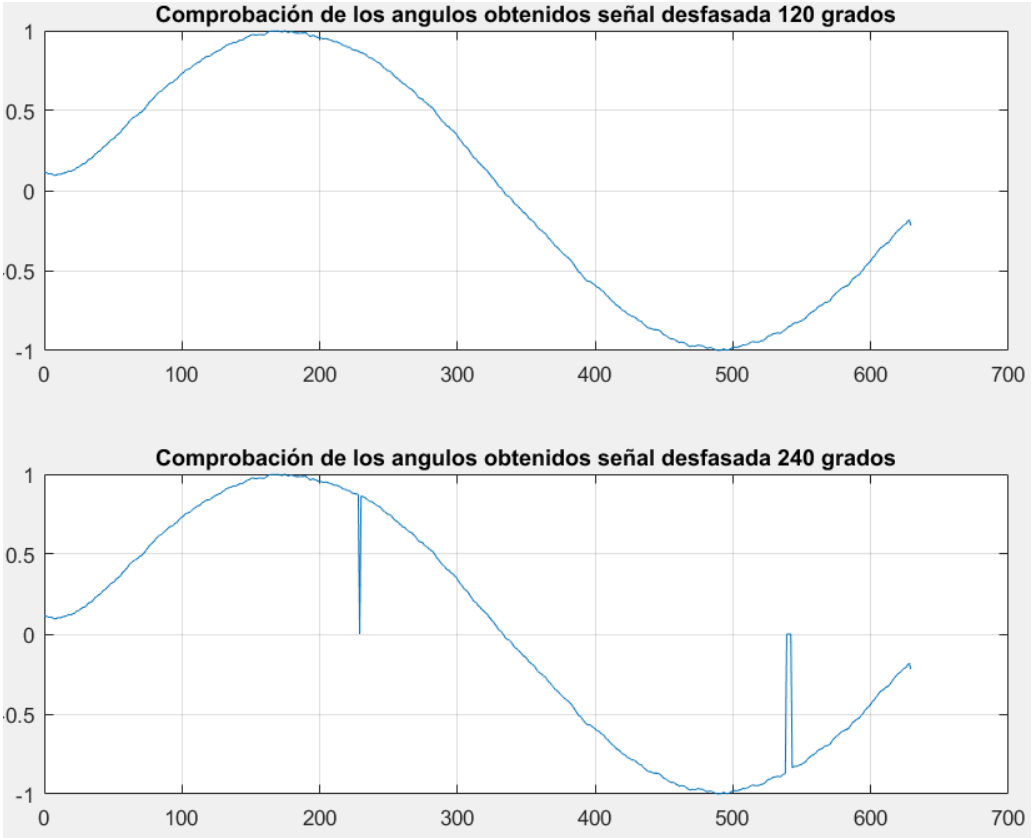


Figura 37. Comprobación de las señales senoidales procesadas

6. MANEJADORES DE DISPOSITIVOS Y SERVICIOS DE LA BEAGLEBONE BLACK

Durante el desarrollo de aplicaciones para la BBB se ha utilizado el entorno de desarrollo de compilación cruzada y el IDE Eclipse desde el computador anfitrión. Con esto se ha permitido escribir varios programas en el computador anfitrión y ejecutarlos en la BBB de una manera sencilla.

Para el manejo de los distintos dispositivos para este trabajo (GPS, eQEP y ADC) el sistema operativo nos proporciona los manejadores que nos permiten interactuar con los dispositivos, por lo tanto se debe cargar los overlay correspondientes, los mismos que queremos que se ejecuten al arrancar la BBB. Por facilidad se ha instalado un servicio que ejecute el script “my_startup_services.sh” ubicado en “/root”. Para ejecutar este servicio se han seguido los siguientes pasos:

1. Crear el archivo “/root/my_startup_service.sh” con el siguiente contenido:

```
#!/bin/sh
echo "----- Starting My Startup Service script -----" > /root/my_startup.log
echo `date` >> /root/my_startup.log

#Try to enable the internet access throght USB interface
echo "Enabling Internet access" >> /root/my_startup.log
/root/BB_Scripts/netUSB_enable_internet_access.sh >>  /root/my_startup.log
echo `route` >> /root/my_startup.log

echo "-----OVERLAY  BBB  SLOTS-------" >> /root/my_startup.log
export SLOTS=/sys/devices/platform/bone_capemgr/slots
echo BB-ADC > $SLOTS
echo "---- BB -ADC-----" >> /root/my_startup.log
cat $SLOTS >> /root/my_startup.log

echo BB-UART1 > $SLOTS
echo "---- BB - UART1 ----" >> /root/my_startup.log

echo bone_eqep0 > $SLOTS
echo "---- BB EQEP ------" >> /root/my_startup.log

echo "----- My Startup Service launched. -----" >> /root/my_startup.log

while true; do sleep 30; done
echo "----- My Startup Service killed. Child processes also killed. -----" >> /root/my_startup.log
```

Figura 38. Contenido del archivo my_startup_services.sh

2. Asignar permisos de ejecución al script.

```
root@BBG:~# nano my_startup_service.sh
root@BBG:~# chmod a+x my_startup_service.sh
```

Figura 39. Comando de ejecución para asignar permisos a my_strartup_services.sh

3. Crear el archivo “/lib/systemd/system/myStartup.service” con el siguiente contenido:

```
[Unit]
Description=Starts user-defined daemons
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/root/my_startup_service.sh

[Install]
WantedBy=multi-user.target
```

Figura 40. Contenido del fichero myStartup.service

4. Arrancar y habilitar el servicio de la siguiente manera:

```
root@BBG:~# systemctl daemon-reload
root@BBG:~# systemctl start myStartup.service
root@BBG:~# systemctl enable myStartup.service
```

Figura 41. Comandos de ejecución para arranque y habilitar servicios en la BBB

5. Reiniciar la BBB, y visualizar el contenido del fichero “my_startup.log”

```
----- Starting My Startup Service script -----
Fri Jun 30 17:14:39 UTC 2017
Enabling Internet access
The interface usb0 is UP. Adding default gateway 192.168.7.1
Kernel IP routing table Destination Gateway Genmask Flags Metric Ref Use Iface default 192.168.7.1 0.0.0.0 UG
 0 0 0 usb0 192.168.7.0 ID.txt bin boot dev etc home lib lost+found media mnt nfs-uEnv.txt opt proc root run
sbin srv sys tmp usr var 255.255.255.252 U 0 0 0 usb0
-----OVERLAY BBB SLOTS-------
---- BB -ADC-----
 0: PF----  -1
 1: PF----  -1
 2: PF----  -1
 3: PF----  -1
 4: P-O-L-   0 Override Board Name,00A0,Override Manuf,BB-ADC
---- BB - UART1 ----
---- BB EQEP ------
----- My Startup Service launched. -----
```

Figura 42. Contenido del fichero my_startup.log, después que los servicios se hayan ejecutado en la BBB

7. PROGRAMACIÓN CONCURRENTE DE LA BEAGLEBONE BLACK

Tradicionalmente un proceso puede realizar una sola actividad que se inicia en la función “main”, logrando de esta manera limitar en gran medida al gran poder de procesamiento que dispone la placa de desarrollo. Con la programación concurrente en un único programa se pueden definir varios hilos de ejecución (threads) que se ejecutan al mismo tiempo pudiéndose comunicar entre ellos para intercambiar datos y sincronizar sus líneas de flujo de control. La concurrencia es fundamental para las aplicaciones de usuario y para el sistema operativo.

7.1. Clase Buffer

Se ha desarrollado la clase BUFFER, para la gestión de todos los datos de los distintos dispositivos y su adecuada incorporación en el mensaje que será enviado a la aplicación por parte del servidor del sistema de navegación. Se ha utilizado el concepto de mutex para la sincronización de hilos y garantizar la exclusión mutua, y poder gestionar los datos de un buffer a otro buffer. Esta clase trabaja con estructura de datos para un acceso adecuado por parte de los hijos de ejecución que serán lanzados desde el servidor. Esta clase posee varias funciones que serán utilizadas por los diferentes hilos de ejecución (Thread) como se detalla a continuación.

No.	Thread	Función	Descripción
1	void *productor_g (void *ptr)	databbb.put_item_g(valor_1,ptr_struct);	En este thread se encuentra la función put_item_g que coloca los valores útiles del mensaje NMEA del GPS en el buffer asociado al GPS.
2	void *productor_IMU (void *ptr)	databbb.put_item_RPY(valor, ptr_struct);	Este thread obtiene los valores de la IMU, retornando los valores de

			roll, pitch y yaw que serán asociados al buffer correspondiente.
3	void *productor_freq (void *ptr)	databbb.put_item_freq(valor, ptr_struct);	Este thread recoge los valores del módulo eQEP y almacena el valor a su buffer correspondiente.
4	Void *productor_3_analog (void *ptr)	databbb.put_item_sin(valor, ptr_struct);	Mediante la función de lectura del ADC, y con un periodo de muestreo adecuado registra los valores de 3 canales ADC al buffer asociado.
5	void *connection_handler (void *ptr)	➔ databbb.get_item_IMU(&datorpy, &ptr_struct->buffimu); ➔ databbb.get_item_g(&datogps, &ptr_struct->buffgps); ➔ databbb.get_item_freq(&datofreq, &ptr_struct->bufffreq); ➔ databbb.get_item_sin(&datosin, &ptr_struct->buffsin);	Accede a los buffer de todos los dispositivos y los integra en un mensaje que será enviado desde el servidor.

7.2. Modelo de comunicación aplicación con BeagleBone Black

Se ha desarrollado un modelo de comunicación Cliente -Servidor con sockets. El proceso que adopta el rol de cliente se deberá conectar con el proceso servidor, para ello, necesita conocer la dirección IP de donde se ejecuta el servidor y el puerto por el que acepta las conexiones. Una vez iniciada la comunicación, queda establecido un canal de comunicación bidireccional entre cliente y servidor. Las funciones utilizadas para el desarrollo de este modelo de comunicación fueron suministradas por la asignatura sistemas empotrados del máster.

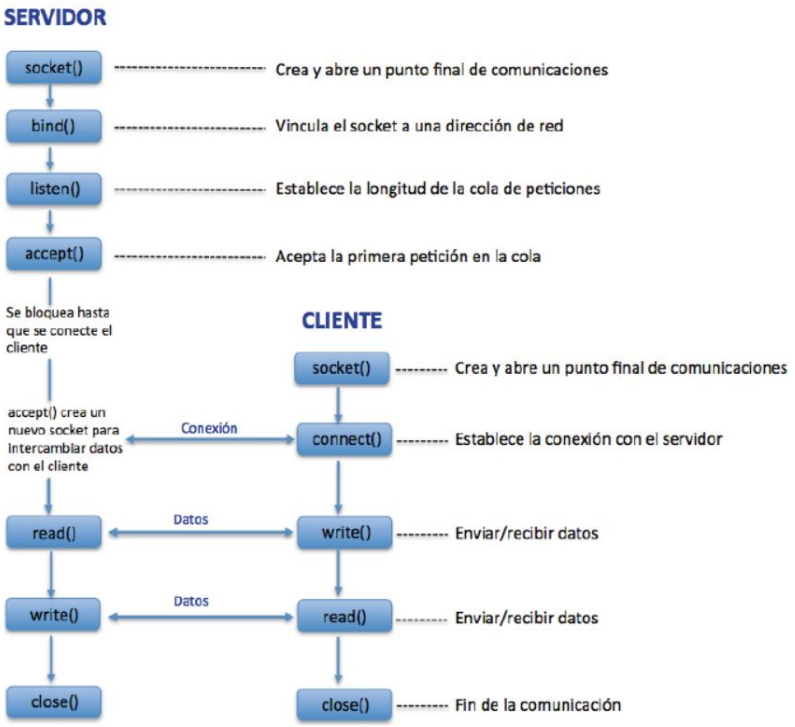


Figura 43. Modelo de comunicación cliente - servidor

7.2.1. Servidor con la BeagleBone Black

Se ha desarrollado un servidor que pueda aceptar múltiples conexiones y atenderlas concurrentemente. Para ello se crea un hilo de ejecución por cada conexión aceptada. La secuencia de uso de las funciones por parte del servidor son las siguientes:

No.	Función	Descripción
1	serverSockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);	Crea un nuevo socket y retorna un descriptor de fichero.
2	retval = bind(serverSockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr));	Vincula un socket a una dirección. Retorna 0 si se tuvo éxito y -1 si hay un error.
3	listen(serverSockfd, 5);	Prepara el socket para recibir nuevas peticiones de conexión. Máximo de conexiones 5.
4	comSockfd = accept(serverSockfd, (struct sockaddr *) &cli_addr, &clilen)	Acepta una nueva conexión, la inicializa y crea un nuevo socket a través del cual se realizará la comunicación de datos.

5	write_server = write(sock, ss.str().c_str(), ss.str().size());	Se envía en el canal de comunicación el mensaje con la información de navegación del velero.

7.2.2. Cliente interfaz en JAVA

Por su lado el cliente, está desarrollado en una aplicación con interfaz gráfica en JAVA programada en el entorno de desarrollo integrado NetBeans, que permita crear conexiones TCP para recibir mensajes de texto. La aplicación contara con los siguientes manejadores para eventos de tipo ActionEvent:

No.	Manejador	Descripción
1	Pulsación del botón "Connect"	El manejo consistirá en conectar con el servidor TCP cuya dirección y puerto se especifiquen.
2	Pulsación del botón "Disconnect"	El manejo consistirá en cerrar la conexión establecida por el botón "Connect".

Para la visualización de los valores de navegación, se ha desarrollado una aplicación en JAVA, que es la encargada de gestionar todos los datos procedentes de la placa BeagleBone Black y realizar los cálculos necesarios para obtener la información necesaria para el velero. La aplicación JAVA consta de las siguientes funciones:

No.	Función	Descripción
1	public DataAPP (String output)	Constructor de la clase
2	public boolean DataSplit ()	Divide el mensaje recibido por parte del servidor y lo almacena en un buffer para facilitar el uso de la información recibida.
3	public float getRoll() public float getPitch() public float getCompass() public float getwindDirection() public String getLatitud() public String getLongitud() public String getSpeed()	Devuelve el valor o string asociada a la variable, en ciertas funciones se realiza un cálculo adicional para obtener las mismas. Además realiza una comparación si el nuevo valor que recibe difiere del anterior.

	public float getCOG() public float getSOG()	

La interfaz que ha sido desarrollada se la encuentra en el Figura 26. La aplicación consta de 2 botones (Connect y Disconnect) que son los encargados para conectar y desconectar con el servidor de la placa BeagleBone Black, también se puede modificar el host y puerto que se debe conectar la aplicación. En esta interfaz se puede observar el COG, SOG, latitud, longitud y velocidad en forma numérica, mientras tanto la velocidad de viento, roll, pitch y yaw se puede observar en 3 calibradores (gauges) distintos.

Figura 44. Interfaz de la aplicación JAVA

8. PRUEBAS DE FUNCIONAMIENTO

Para el desarrollo de pruebas de funcionamiento del sistema de navegación, se realizó una simulación con la ayuda de la placa programable Arduino, para poder trabajar con un tren de pulso y así usar el módulo eQEP; para las señales analógicas se utilizaron diferentes divisores de tensión para simular un ángulo exacto. Las conexiones y ubicación de los distintos dispositivos se los puede ver en la Figura 45.

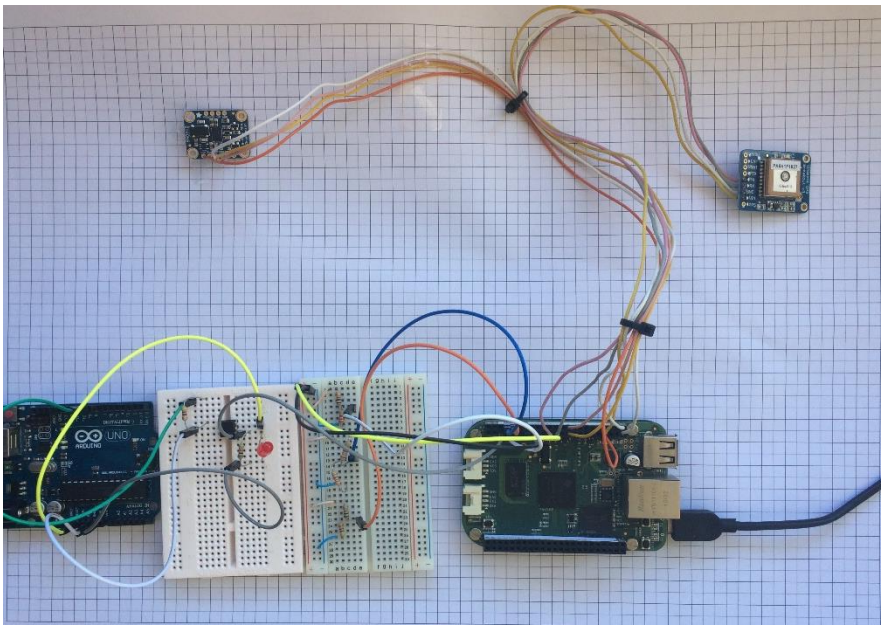


Figura 45. Disposición de los sensores para simulación del sistema de navegacion

Para establecer una comunicación de la aplicación con la placa BeagleBone Black, se debe escribir el Host y Port por donde se van a comunicar, y pulsar el botón Connect. En la Figura 46 se puede observar los mensajes de conexión y los valores que son enviados por parte del servidor.

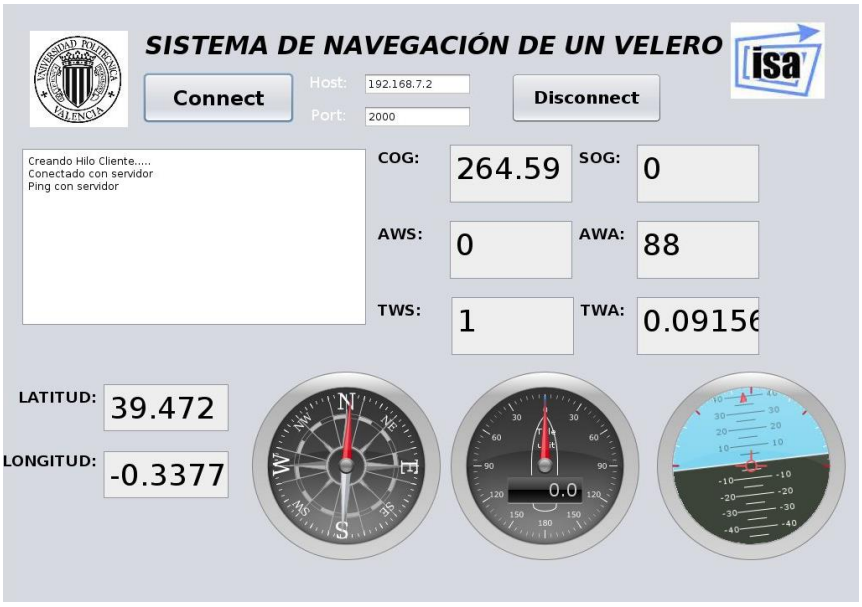


Figura 46. Ensayo de la prueba de funcionamiento

En la Figura 47 se realizó una desconexión con el servidor y enseguida se efectúo otra conexión, para poder ver si el servicio de la BeagleBone Black sigue en ejecución. Los mensajes de alerta del servidor se aprecian en la Figura 48, estos mensajes se los obtuvo ejecutando el programa desde shell del computador anfitrión, mediante el comando /root/projects/TFMApp.

Figura 47. Ensayo de desconexión y conexión del cliente con el servidor

```
I2C::writeSlaveReg: Write addressing read fail:1: Remote I/O error
ID: 1
I2C::writeSlaveReg: Write addressing read fail:1: Remote I/O error
ID: 1
ID: 160

Found Calibration for this sensor


Restoring Calibration data to the BNO055...

Calibration data loaded into BNO055
Sensor BNO055 is Full Calibrated
Fully calibrated!
GPS INICIALIZADO
-------------------------
|INICIO DE LA APLICACION|
-------------------------
Waiting for incoming connections...
Ha fallado el recibir el mensaje buffer2 NMEA
GPS Calibrandose
Connection accepted
Handler assigned
```

Figura 48. Respuesta de la placa, después de ejecutar el servidor

9. PRESUPUESTO

A continuación se procede a describir el presupuesto global estimado del proyecto realizado, con el objetivo de mostrar la aportación económica necesaria para su realización.

Costos de Mano de Obra

Tabla 6. Costos de mano de obra

Código	**Empleado**	**Salario mensual (€/mes)**	**Salario (€/h)**
MT.MAII	Graduado Master en Automática e Informática	3.400 €	
MT.MAII	Seguridad Social	272 €	
	Salario Total (€/mes)	3.672 €	40 €

Costos de Materiales

Tabla 7. Costos de materiales

Código	**Cant.**	**Denominación**	**Precio Unitario(€)**	**Total(€)**
MT.BBB	1	BeagleBone Black Wireless	87,90 €	87,90 €
MT.GPS	1	Ultimate GPS Adafruit	39,95 €	39,95 €
MT.IMU	1	IMU BNO055 Adafruit	43,56 €	43,56 €
MT.BAT	1	Alimentador para BBB	9,90 €	9,90 €
MT.RTR	1	Router inalámbrico portátil	22,95 €	22,95 €
MT.PCB	1	Placa de circuito impreso 70x80 cm	75,00 €	75,00 €
MT.CDP	1	Carcasa de protección de aluminio	35,00 €	35,00 €
MT.CDC	10	Cables UDP flexible para conexión	1,25 €	12,50 €
MT.EXT	1	Imprevistos	35,00 €	35,00 €
			Precio Total (€)	361,76 €

Costos de Maquinaria

Para el cálculo de la amortización de la maquinaria se ha supuesto un periodo de amortización de 1 año, en el cual la vida útil del equipo es 18000 horas/año, equivalente a 225 días laborales con una jornada de trabajo de 8 horas.

Tabla 8. Costos de maquinaria

Código	**Denominación**	**Precio (€)**
M. ORDP	Ordenador Portátil	1.000 €

Costos de Implementación

Tabla 9. Costos de implementación

Código	Unidad	Descripción	Precio (€)	Rendimiento	Total (€)
MT.MAII	h	Portabilidad de I.M.U. y GPS	40 €	40	1.600 €
MT.MAII	h	Programación de ADC y eQEP	40 €	24	960 €
MT.MAII	h	Programación de algoritmo necesario para la sincronización de elementos del proceso para la navegación	40 €	120	4.800 €
MT.MAII	h	Programación de la interfaz para la visualización de información	40 €	32	1.280 €
MT.MAII	h	Montaje de la BeagleBone y puesta en marcha del proyecto	40 €	16	640 €
M. ORDP	h	Ordenador Portátil	0,55 €	232	128 €
	%	Costos administrativos: renta, electricidad, agua, internet, transporte		8	753 €
			Subtotal (€)		10.160 €

Presupuesto Total

Tabla 10. Presupuesto total

Código	Descripción	Subtotal(€)
1	Dispositivos electrónicos	362 €
2	Programación, implementación y montaje	10.160,21 €
	Presupuesto de ejecución del proyecto	**10.521,97 €**
	Gastos generales 13%	**1.367,86 €**
	Beneficio Industria 6 %	**631,32 €**
	Subtotal	**12.521,14 €**
	IVA 21%	**2.629,44 €**
	TOTAL PRESUPUESTO	**15.150,58 €**

El presupuesto que presenta el desarrollo del proyecto es elevado, pero se debe considerar que el coste que este demuestre deberá ser repartido en las unidades que se puedan vender.

10. TRABAJOS FUTUROS

Una vez alcanzados los objetivos propuestos para este proyecto, se plantean las siguientes opciones para futuros trabajos:

- Mediante el software Proteus, se ha diseñado una placa de circuito impreso (PCB), que actué como shield para la placa BeagleBone Black, de esta manera se logrará minimizar el ruido electrónico, facilidad de la detección de fallo y reparación de los componentes. Además ocupar un menor espacio para su montaje y debido a que los componentes están fijos al circuito impreso por una soldadura no nos debemos preocupar por la posibilidad de desplazamiento de los mismos. En la Figura 49, se puede observar la PCB de doble cara diseñada con sus capas Top Copper y Bottom Copper, que son las necesarias para fabricar el circuito impreso.

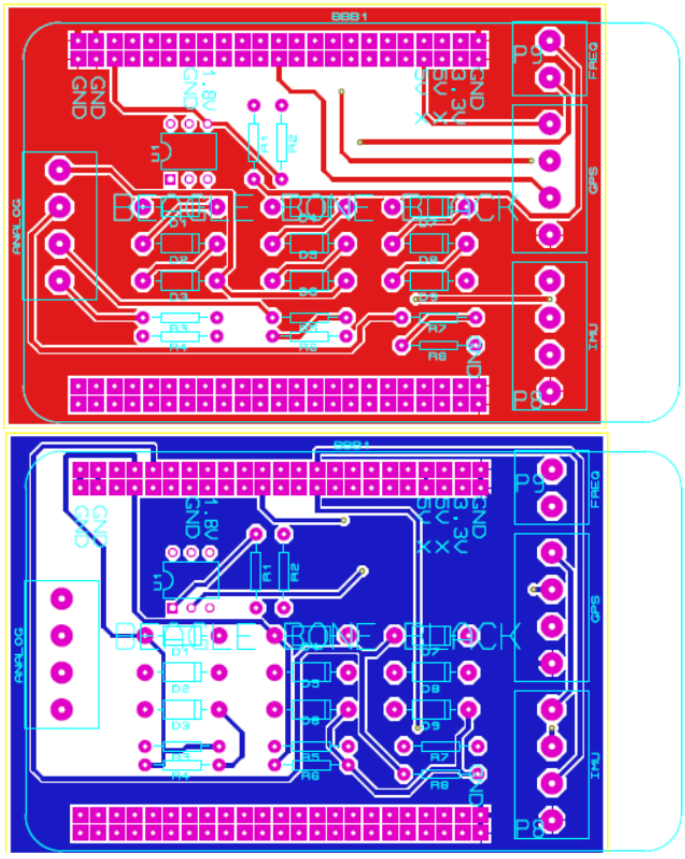


Figura 49. Prototipo de PCB para BeagleBone Black

- Desarrollar de una interfaz de navegación que sea compatible con dispositivos móviles inteligentes, adoptar la portabilidad para lenguajes Android o IOS.

- Aumentar la funcionalidad de la de la interfaz con nuevas aplicaciones, por ejemplo integrarla con ”Googlemaps” para realizar un seguimiento de la ruta que se ha tomado, guardar registros históricos de los últimos trayectos realizados, entre otros.

- Mejorar la velocidad de refresco de la información completa de los sensores para él envió del paquete por TCP/IP.

11. CONCLUSIONES

Se ha desarrolla satisfactoriamente una integración completa del sistema de sensores y una placa de bajo coste para poder determinar la información necesaria para la navegación de un velero.

Por otro lado, es importante destacar el papel fundamental de la I.M.U., tecnología que cada vez es más utilizada para muchas aplicaciones, ya que en un sistema de navegación, se convierte en un sensor fundamental por la variedad de información que puede aportar. En el mercado existen varias opciones, lo que facilita la elección de un modelo que se adapte a las necesidades de la aplicación. Además es necesario recalcar que el magnetómetro es bastante susceptible a distorsiones de las diferentes aleaciones de hierro, por lo que se debe tomar en cuenta la zona donde se coloca la I.M.U., para realizar una correcta calibración.

La portabilidad exitosa que se ha desarrollado para la gestión de información del GPS e I.M.U., hace que se puedan integrar un mayor número sensores para poder manejarlos por parte de la placa BeagleBone Black, y lograr tener un sistema operativo que gestione estos sensores, esto conlleva a tener un sistema embebido más robusto para realizar diferentes aplicaciones.

Dotar al velero con un sistema embebido que cuente con una conexión a internet, permite obtener una mayor cantidad de información de navegación, misma que es de gran utilidad para evaluar: rendimiento, marcas y trayectos. Incluso, facilita la toma de decisiones de la persona que está en el timón en cuanto al trayecto deseado.

El dispositivo GPS que se utilizó, contiene un software interno muy potente, mismo que ayudó a la programación de la clase respectiva para la determinación de toda la información que es de gran importancia para cualquier embarcación, en este caso específico un velero.

La correcta implementación de un circuito de protección para el uso del módulo eQEP, ha conllevado a la obtención de excelentes resultados en cuanto a su funcionamiento. Con las pruebas realizadas se ha logrado determinar que su rango de funcionamiento va desde frecuencias bajas hasta frecuencias altas de muestreo. Con esto también se puede ampliar el uso de este módulo que está diseñado principalmente para lectura de encoders.

12. BIBLIOGRAFÍA

Atzori, L., Iera, A., & Morabito, G. (2010). *The Internet of Things: A survey.* Cagliari: Elsevier.

Billore, D. (04 de Febrero de 2015). *Embedded Systems Market Size, Share and Trends Till 2020*. Recuperado el 03 de Agosto de 2017, de https://www.linkedin.com/pulse/embedded-systems-market-size-share-trends-till-2020-devesh-billore

CEA. (2015). *The Internet of Things: Evolution or Revolution?*

Di Paolo, E. (2015). *Embedded System Design for High-Speed Data Acquisition and Control.* Spinger.

EOI. (2015). *Las tecnologias IoT dentro de la industria conectada 4.0.* Madrid.

Europe, Rand. (2012). *Europe's policy options for a dynamic and trustworthy development of the Internet of Things.*

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). *Internet of Things (IoT): A vision, architectural elements, and future directions.* Melbourne: Elsevier.

Henzinger, T., & Sifakis, J. (201X). *The Embedded System Desing Challenge.*

Huircan, J. (20XX). *Convesores Analogico-Digital y Digital-Analogico: Conceptos Basicos.*

IDC. (2012). *Final Study Report: Design of Future Embedded Systems.* Paris.

IDC. (2017). *Data Age 2025: The Evolution of Data to Life-Critical.* Framingham.

Linfo. (03 de Agosto de 2017). *The Linux Information Project*. Obtenido de http://www.linfo.org/embedded_system.html

Toulson, R., & Wilshurt, T. (2017). *Fast and Effective Embedded System Desing* (Second Edition ed.). Elsevier.

UNED. (2014). *Ingenieria de los Sistemas Embebidos.*