Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Sistema de detección de "hombre al agua" basado en tecnología GPS

Grado en Ingeniería Mecánica

ALUMNO: Javier Yrayzoz Goenechea

DIRECTOR: Rafael Asorey Cacheda

CURSO ACADÉMICO: 2015-2016

Universidade Vigo

Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Sistema de detección de "hombre al agua" basado en tecnología GPS

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

Este trabajo Fin de Grado presenta un sistema de detección que permite saber cuando un miembro de la dotación de un buque de la Armada ha caído accidentalmente al agua.

Para su realización, el sistema se basa en microcontroladores Arduino, dos placas GPS, un módulo Bluetooth del tipo HC-06 y otro módulo Bluetooth del tipo HC-05. Para facilitar la conexión entre la placa Arduino y el módulo Bluetooth se emplea una placa de prototipado.

El proyecto se resume en tres fases. La primera consiste en la instalación del módulo GPS y la obtención de la posición del dispositivo. La segunda trata la conexión entre sí de los módulos Bluetooth. Finalmente, la última desarrolla el intercambio de posiciones entre las placas Arduino a través de dichos módulos Bluetooth.

Para determinar cuando una persona cae al agua se establece un perímetro de seguridad alrededor de la plataforma cuyo diámetro se corresponde con la eslora del buque. Para comprobar el funcionamiento de este, se acude a una lancha de adiestramiento de la Escuela Naval Militar, tipo Rodman 66. Los resultados recogidos durante la prueba son más que satisfactorios, por lo que se demuestra que podría ser puesto en marcha de manera eficaz.

Durante el proyecto se especifican las diferentes dificultades encontradas con el uso de estos materiales, al igual que también se indican las líneas a seguir para su posible mejora en el futuro.

PALABRAS CLAVE

Detección de hombre al agua, Posición GPS, Comunicación Bluetooth, Arduino, Marina española

AGRADECIMIENTOS

Este proyecto no hubiera sido posible sin el apoyo que he recibido de mi familia desde que decidí el camino que quería seguir, por lo que me gustaría agradecer a todos y cada uno de ellos su granito de arena aportado. A su vez, agradecer a mi padre que desde pequeño ha sido un referente para mi.

Agradezco a la Señora Martinazzo, mi profesora de matemáticas de bachiller, la gran base que me aporto y su gran apoyo incondicional que siempre me mostró.

También quiero agradecer al director del presente trabajo fin de grado D. Rafael Asorey, por confiar desde el principio en la viabilidad del proyecto. Sin nunca olvidar que siempre fue un gran apoyo durante la realización del trabajo.

Finalmente, y no por ello menos importante, a la Escuela Naval Militar por poner a mi disposición las Lanchas de Instrucción, ya que sin ellas no se podría a ver llevado a cabo las pruebas empíricas.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Descripción del proyecto

Este trabajo es producto del análisis de una problemática que existe actualmente en todos los buques de la Armada, como es detectar rápidamente el momento en que un hombre cae al agua accidentalmente. Durante el periodo de formación en las unidades, el alumno detectó que esta ineficiencia podría resolverse con un sistema como el que se explica en esta memoria.

El avance de la tecnología ha permitido que el entorno y los ordenadores estén conectados. De esta manera, los ordenadores son capaces de analizar y decidir a partir de los datos que se obtienen del medio ambiente. Gracias a esto, se desarrollan nuevas tecnologías para vivir en un mundo más seguro. Entre estas, se encuentra la seguridad de las personas. La sociedad vive en un mundo en el que la muerte de una persona por un fallo humano es inaceptable, y menos si puede evitarse con la ayuda de la tecnología. Desde este punto de partida, se propone una solución que permite detectar cuando una persona pueda haber caído al agua desde un barco.

1.2 Objetivos

El objetivo de este proyecto es diseñar un sistema con microcontroladores Arduino comunicados por Bluetooth, que midan la distancia que entre ellos y así determinar cuando alguien cae al agua. Para calcular la distancia se utilizan las posiciones obtenidas mediante dispositivos GPS.

En el prototipo desarrollado, uno de los microcontroladores Arduino se coloca en el puente de la plataforma, actuando como punto estático. Este microcontrolador obtiene la posición GPS del centro del buque, mientras el segundo es portátil y va junto a las personas que trabajan en el exterior del buque. De esta manera, se obtiene la distancia entre las dos posiciones. Dicha distancia, en caso de ser mayor al radio en el que se pueda encontrar la persona que trabaja en exteriores, genera una alarma para que el personal del buque sepa que una persona se ha caído al agua y así empezar la maniobra de rescate.

1.3 Organización de la memoria

El resto de la memoria se organiza el trabajo en cuatro bloques:

- En el capítulo 2 se estudian tecnologías similares al objetivo de este trabajo, como puede ser cualquier sistema de control de personal. A continuación, se explican las tecnologías empleadas en este proyecto como Bluetooth, GPS o Arduino, entre otras.

- En el capítulo 3 se pressentan los resultados obtenidos con el prototipo desarrollado. En este apartado se analizan en detalle estos resultados y las dificultades encontradas durante el proceso de desarrollo.

- Finalmente, en el capítulo 4 se ofrecen las conclusiones finales y se proponen líneas futuras de mejora y continuación de este proyecto.

2 ESTADO DEL ARTE

2.1 Introducción

En el mundo, todo tiende a estar controlado. Cuando uno entra o sale de una tienda, de un campo de fútbol o de la oficina, es casi siempre identificado por diferentes sistemas. Sin embargo, en el ámbito de la Armada podríamos hablar de la falta cierto control de personal durante las navegaciones.

Desde la época del hundimiento del buque "TITANIC", la "Organización Internacional Marítima" está interesada en mejorar los sistemas de control de personas en los buques. Este interés, se debe al aumento de buques tipo "crucero" que transportan miles de pasajeros. Por ello, el desarrollo, la investigación e implantación de los sistemas de control de pasajeros representa un gran capítulo en los gastos de desarrollo de los buques comerciales. Así, las empresas que trabajan para los astilleros de cruceros desean disponer de sistemas de control de personal que permitan saber, en caso de emergencia, donde se encuentran todos y cada uno de los pasajeros (Murias, 2007).

El tiempo es un factor muy importante en situaciones de emergencia. Reducir el tiempo desde que ocurre una incidencia hasta que se detecta y se avisa de ella, puede ser clave de cara a evitar una catástrofe o un daño mayor. Por ello, se están desarrollando muchísimas técnicas automatizadas que permiten detectar cualquier emergencia. La que se encuentra más a la orden del día en el mundo es la detección de incendios. La empresa alemana "BOSCH" ha desarrollado un sistema buscapersonas y de detección de incendios, que según ellos reduce el pánico y reduce el tiempo de respuesta de las alarmas. Este sistema, cuando detecta un incendio salta una alarma que solo escuchan el personal responsable de la evacuación, ayudando así a que en un primer momento solo ellos puedan saber si es una falsa alarma. Si no es así, llevan a cabo la evacuación de manera más ordenada evitando el caos. Esta personas, al igual del resto, disponen de unos "móviles de seguridad" que les permiten recibir avisos con una alarma discreta y, a su vez, les permiten mandar hasta tres mensajes al resto del personal y también almacenar hasta 10 mensajes. (Bosch, 2015).

En otras marinas, como la Marina Australiana, se utiliza un sistema de tarjetas desde el cual se puede saber donde se encuentra el personal en todo momento. A esta idea, se le une la desarrollada por unos investigadores de la Universidad de Salamanca, que pretenden desarrollar un sistema que sepa donde se encuentra todo el personal y a su vez tener el control de las cámaras desde otro sitio fuera del buque (Garcia del Valle, 2016). El protocolo de comunicación ZigBee, inalámbrico, sería el usado en este sistema. Los creadores aseguran que este sistema sería muy útil para combatir la piratería ya que desde los dispositivos inalámbricos se podría dar la señal de alarma. Una vez se sabe que el buque se

encuentra en problemas, desde la estación terrestre pueden controlar las cámaras y ciertos accesos. Así, sabiendo donde se encuentra el personal, podrán cerrar ciertas puertas a los que se desea prohibir la entrada a los posibles piratas.

En la Marina Española se ha implantado, en la última serie de buques, un sistema que controla donde se encuentran cada uno de los miembros de la dotación. Esto puede ser práctico aunque genera ciertos problemas, ya que el dispositivo es una tarjeta que cualquiera puede dejar en su destino o camarote y puede llegar a confundir al usuario. Además, si este sistema se implementara en otro tipo de buques que esté muchas horas en la mar, sería perjudicial ya que eliminaría la poca privacidad que queda en un espacio tan reducido. Por ello, se plantea usar un sistema que con la ayuda de unas pulseras se pueda saber cuando alguien, que esté en exteriores, se cae al agua y reducir el tiempo de detección de la incidencia.

2.2 Microcontralodores Arduino

El sistema de microcontroladores es una plataforma abierta para la creación de prototipos como el que se desarrolla en este proyecto. En conjunto, se trata de un *software*, una placa microcontroladora y accesorios que se le añaden a la placa para la realización de alguna de las funciones que se desee.

El *software* permite la programación de dicha placa. Utiliza un lenguaje llamado "Processing", que esta basado en "C" y "C++" con pequeñas variaciones. Este lenguaje permite diseñar lo que se desea obtener, medir o calcular con ella. El *software* de desarrollo se descarga de la pagina del fabricante (33pág. www.arduino.cc; 33) y esta disponible para los sistemas operativos más conocidos. Dicho *software* dispone de multitud de ejemplos, que se pueden usar para proyectos propios. Este programa dispone también de un terminal llamado "monitor serial" en el que se puede ver el resultado de cualquier operación gracias a diferentes comandos que se detallan más adelante. Además, esta plataforma dispone de un sistema de detección de errores que informa al usuario de errores en el código. Si no sabe solucionarlos por él mismo, puede solicitar ayuda en la *web* mencionada anteriormente (Web arduino).

2.2.1 Arduino UNO

Existen multitud de modelos de placas Arduino que, en función de sus características técnicas serán un modelo u otro. El más básico, que es con el que se ha trabajado en este proyecto, es la placa "Arduino uno".

2.2.1.1 Partes de la placa Arduino uno

El microcontrolador "Arduino UNO" (Figura 2-1) está compuesto por:

1. *Microprocesador*: se encarga de ejecutar todas las instrucciones del programa de manera cíclica.

2. *Puerto USB*: a través de él se cargan las instrucciones a ejecutar y se permite la comunicación serial entre el ordenador y la placa Arduino.

3. *Puerto de alimentación*: es el puerto por el que se le da energía en caso de no estar conectado por USB. En este puerto es donde se puede conectar una pila o batería.

4. *Chip controlador de frecuencia*: es el reloj del sistema.

5. *Pines de entrada y salida*: permiten colocar los elementos para interactuar con el medio.

6. *"LED 13"* : LED integrado conectado al pin 13 de la placa.

7. *"LED Tx y Rx"*: un LED que se enciende o se apaga cuando se transimiten o reciben datos.

8. *Pines de transmisión y recepción*: pines en los que se conectaran las entradas o salidas.

9. *"LED indicador ON / OFF"*: pequeño LED que se enciende cuando a la placa le llega corriente.

10. *Botón de* reinicio: este botón permite reiniciar la placa. Es bastante útil ya que muchas veces queda información del anterior programa cargado.

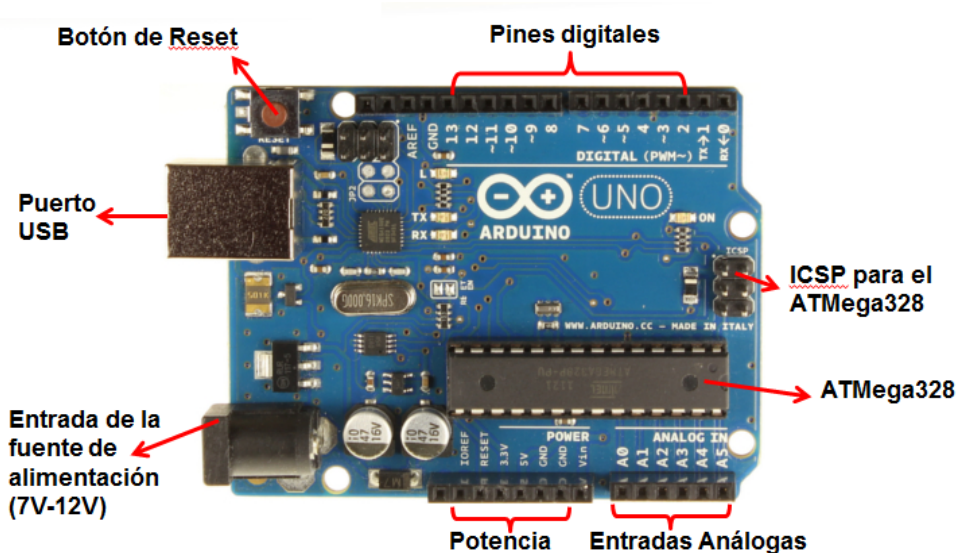

Figura 2-1 Arduino UNO (Arduino).

2.3 Bluetooth

La tecnología Bluetooth es un sistema estándar de comunicación inalámbrica que conecta dispositivos separados por una cierta distancia, como la unión de un teléfono móvil con unos cascos inalámbricos. Permite eliminar cables y conectores entre dispositivos y facilita la creación de pequeñas redes inalámbricas que sincronizan datos entre equipos. Existen infinidad de objetos que pueden ser conectados entre sí, por ello se debe destacar la importancia del "Internet de las cosas" (IOT).

2.3.1 El internet de las cosas

El "Internet de las cosas" es un concepto que propuso Kevin Aston en el MIT. Trata básicamente de dar acceso a Internet a todos los objetos del mundo. Su creador dijo que si todos los objetos del mundo estuvieran conectados a Internet y equipados con un dispositivo de identificación, el cual, podría utilizar el sistema de comunicación Bluetooth, se podría llegar a la inexistencia de cosas fuera de *stock* o carencia de medicinas. También sabríamos la ubicación y como se consumen y venden los productos en todo el mundo. Esta tecnología eliminaría la pérdida de objetos, ya que en todo momento se podrían localizar. En nuestro caso, perder una persona en un buque sería algo del pasado.

2.3.2 Funcionamiento del Bluetooth

Los dispositivos Bluetooth son capaces de conectarse entre sí mediante un enlace por radio en la banda ISM de 2,4 GHz. Cualquier equipo Bluetooth contiene siempre un pequeño *chip* con una radio y un *software* que facilita la conexión entre dispositivos.

Cuando dos dispositivos Bluetooth desean comunicarse, antes de nada, deben buscarse el uno al otro y sincronizarse. La comunicación entre dispositivos es siempre de corto alcance. A estas redes se las llama "piconets", como se puede ver en la Figura 2-2 (Web bluetooth). Estas redes pueden tener de dos a siete dispositivos y hay siempre un maestro y el resto son esclavos. El maestro se encarga de elegir el canal correcto para el enlace y establece conexiones en las que un paquete de datos ocupa una ranura para la emisión y otra para la recepción. Esta técnica sirve para transformar un canal *símplex* en un canal *dúplex*, separando las señales enviadas y recibidas en intervalos de tiempo diferentes.

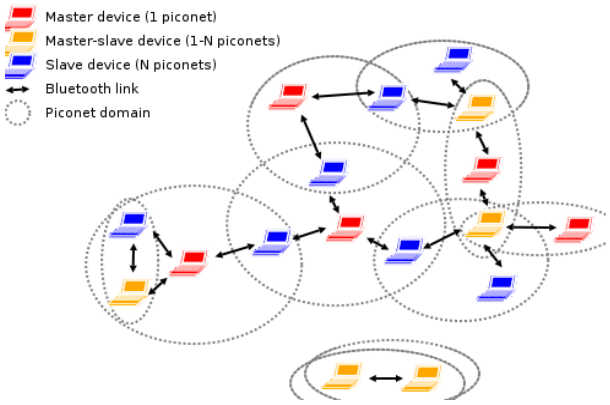

Figura 2-2 Los dispositivos Bluetooth se relacionan formando piconets. (Web bluetooth).

2.3.3 ¿Por qué elegir la tecnología Bluetooth?

Bluetooth , en nuestros días, se encuentra en casi cualquier aparato, utiliza poca energía, es fácil de usar y su coste de implantación es bajo. A continuación se detallan estas afirmaciones (Bluetooth SIG):

1. **Está en todos los sitios**: cualquier ordenador, *tablet* o teléfono móvil dispone de esta tecnología. Desde el punto de vista actual, muchos dispositivos necesitan este sistema para funcionar ya que las conexiones con cable forman casi parte del pasado. No sería aceptable que un ordenador no pudiera tener un ratón inalámbrico o que un teléfono móvil no se pudiera conectar un altavoz inalámbrico.

2. **Utiliza poca energía**: gracias a la creación de Bluetooth de baja energía (BLE, *Bluetooth Low Energy*) se ha podido incorporar en dispositivos más pequeños.

Es fácil de usar: el usuario de esta tecnología solo tiene que ir a los ajustes de su dispositivo, encender su Bluetooth para buscar el otro dispositivo y, una vez encontrado, enlazarlo. Por ello ha tenido tanta acogida. Además de tener contraseña si se desea, Bluetooth contiene, aunque la mayoría de usuarios no lo saben, un cifrado tipo "Advanced Encryption Standard" (AES) que permite el intercambio seguro de datos.

3. **Bajo coste**: la empresas pueden añadir la tecnología Bluetooth por un coste mínimo, ya que solo tienen que pagar el coste del módulo Bluetooth que se desea y una pequeña tasa administrativa para el uso de esta tecnología. Como curiosidad, esta tasa es diferente en función de la magnitud de la empresa contratante.

2.3.4 Wi-Fi HaLow, ¿el futuro Bluetooth?

"Wi-Fi Hallow" es una nueva tecnología anunciada por la "Wi-Fi Alliance" en enero de 2016. La nueva conexión inalámbrica tendrá un bajo consumo energético y mejorará el alcance con respecto a Bluetooth (Figura 2-2). Esta tecnología opera en la banda de 900 MHz, mientras que un equipo actual trabaja en la de 2,4 GHz o 5 GHz (Wi-Fi Alliance).

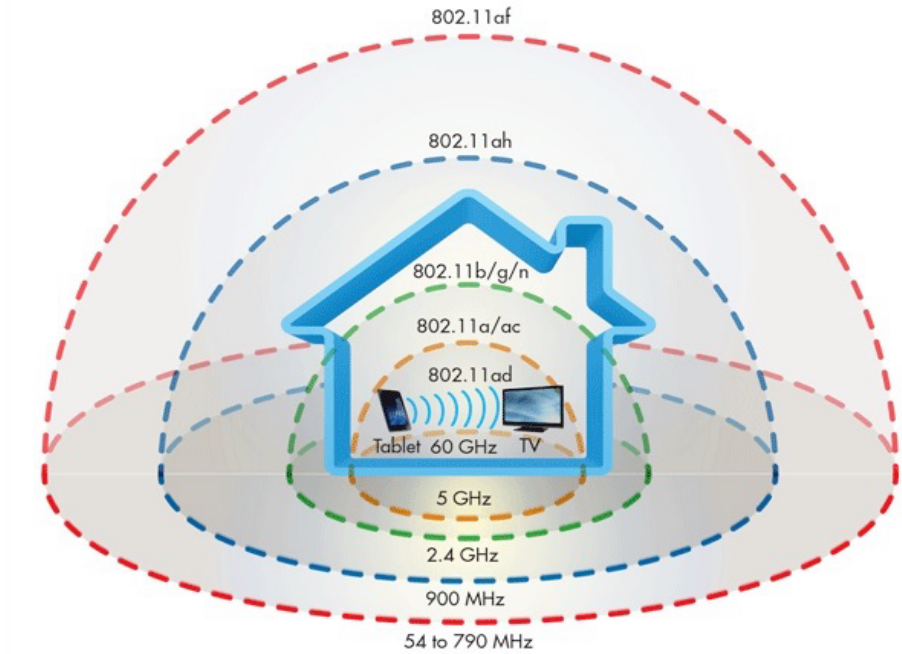

Figura 2-2 Esquema de diversas tecnolgías de transmisión inalámbrica (Objets, 2016).

La mejora con respecto a Bluetooth es que los dispositivos podrán estar conectados directamente a Internet. En cambio, con Bluetooth tienen que conectarse a un dispositivo con acceso a Internet. Sin embargo, la desventaja es que los pequeños dispositivos, que antes eran ajenos a Internet, ahora serán más vulnerables a problemas de seguridad. Este avance es otro paso mas hacia el "Internet de las cosas". De hecho, ya se pueden encontrar muchos dispositivos como altavoces que se conectan por WiFi, en vez de por Bluetooth.

2.3.5 Comandos AT

En 1977, Dennis Hayes inventó un sistema de comandos AT como un interfaz de comunicación con un módem, para así poder configurarlo o transmitir instrucciones (Bluehack web, 2005). Se podría decir que los comandos AT son instrucciones codificadas que forman un lenguaje de comunicación entre el hombre y el terminal de una máquina o entre dos máquinas.

2.3.6 Bluetooth HC-05

El módulo Bluetooth HC-05 (Figura 2-3) es uno de los más conocidos para aplicaciones con microcontroladores Arduino. Estos dispositivos triunfan ya que son muy baratos y, además, se venden en un formato que permite conectarlos a una placa de desarrollo y cablearlos a cualquier microcontrolador.

Existen módulos parecidos a este, pero el HC-05 puede configurarse tanto de maestro como de esclavo. Por lo tanto, puede recibir conexiones de otros dispositivos (esclavos), o bien generar una conexión con otro dispositivo (maestro).

Figura 2-3 Foto realizada del módulo HC-05.

2.3.7 Bluetooth HC-06

Este módulo Bluetooth es físicamente igual al HC-05 excepto que lleva cuatro pines en vez de seis (Figura 2-4). Una de la ventajas de este dispositivo, además de sus dimensiones, de su buen alcance en transmisión y en recepción , es el bajo consumo de corriente. El HC-06 no consume prácticamente nada en funcionamiento pero tampoco en espera. Es decir, alimentado con energía pero sin conexión o enlace a otro dispositivo gasta mucha menos energía que estando enlazado. Otra gran característica que posee este módulo es que es una vez realiza una conexión es capaz de guardarla en su memoria y no solicita ningún tipo de validación. Esto ayuda a no perder tiempo con los comandos AT enlazando los dispositivos.

Desde el punto de vista de la idoneidad para este proyecto, es muy útil que su tensión de alimentación sea 3,3 V, lo que le hace ideal para trabajar con microcontroladores que trabajan con la misma tensión.

En nuestro caso, este módulo Bluetooth se pretende que sea portátil. Por lo tanto, tendrá que ser alimentado por una pila o batería recargable. Su bajo consumo se convierte en un aspecto crucial ya que cuanto menos consuma funcionará más tiempo se o mas pequeña será la pila o batería instalada. Esta última consideracion facilita un diseño de menor tamaño.

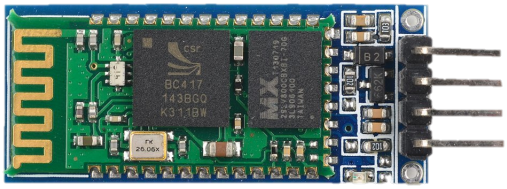

Figura 2-4 Modulo HC-06 (Prometec).

2.4 GPS

El Sistema de Posicionamiento Global o GPS es un sistema que permite posicionar cualquier objeto (persona, vehículo, calle, etc.) en el mundo. Si se utiliza el GPS diferencial se puede saber la posición con centímetros de exactitud.

2.4.1 Historia del GPS

El GPS fue inventado por el Departamento de Defensa de Estados Unidos. En concreto, se inventó para mejorar los métodos de navegación de la Armada estadounidense. Antes de existir el GPS, los buques se posicionaban mediante mediciones de alturas a diferentes astros. El objetivo de la Armada americana fue obtener un sistema de posicionamiento automático como el que se tiene ahora. Antes de llegar al sistema GPS actual se utilizaba otro, llamado "TRANSIT"(Figura 2-5), que obligaba a un buque estar parado 40 minutos para obtener su posición. Con la invención del reloj atómico, en una operación conjunta con el ejercito del aire americano, se creó el NAVSTAR GPS. Así, entre 1978 y 1983 se lanzaron once satélites de prueba y a partir de 1995 se consideraron de "capacidad total operacional (FOC)" (GPS, 2016).

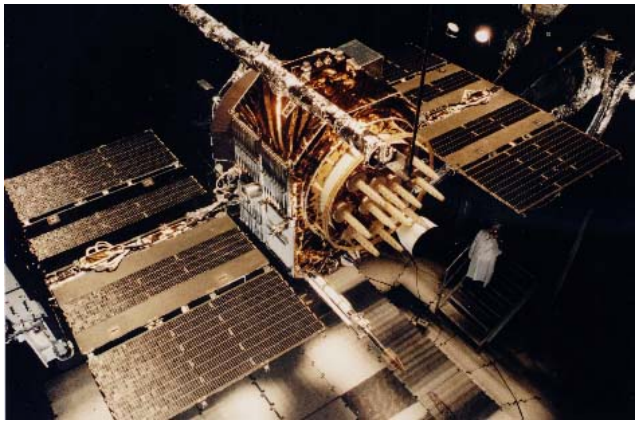

Figura 2-5 Receptor Satélite NAVSTAR (Air Force, 2005).

2.4.2 Funcionamiento

Para determinar la posición, el sistema tiene 24 satélites repartidos alrededor del planeta Tierra. Estos satélites están en órbita, a una altura de 20.200 kilómetros, con trayectorias dirigidas de manera que cubren siempre todo el planeta.

La información que permite que el receptor GPS calcule su posición se llama efeméride. Cada satélite en cobertura indica al receptor cual es su posición con respecto a cada uno de estos satélites. Si se obtiene la información de dos satélites se determina una circunferencia resultado de intersecar dos esferas en algún punto de la cual se encuentra el receptor. Con un tercer satélite se elimina el problema de la sincronización entre los relojes de los receptores y los satélites. A partir de ese momento es cuando el receptor (Figura 2-6) puede calcular la posición 3D exacta con latitud, longitud y altitud.

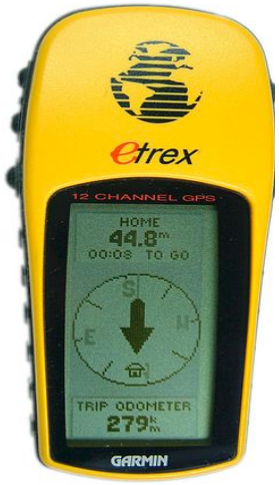

Figura 2-6 Receptor convencional (GPS) de la marca Garmin. (Etrex).

Hasta el año 2000, el Departamento de Defensa de los Estados Unidos se guardó la llamada "disponibilidad selectiva". Consistía en inducir un cierto grado de error, que podía variar la posición de cualquier usuario de 15 a 100 metros (GPS, 2016). Actualmente, la posición se obtenida de la visión de 7 a 9 satélites. Esto consigue un error de menos de 1,5 metros el 97% de tiempo. Existen tres tipos de fuentes de error posibles:

1. *Errores propios del satélite*: de reloj, orbitales o de configuración geométrica.

2. *Errores provenientes del medio de propagación*: de refracción ionosférica, troposférica o multicamino.

3. *Errores en la recepción*: Ruido o centro de fase de las antenas.

2.4.3 Ultimate GPS logger shield

En este proyecto se utilizó un módulo GPS de la casa "Adafruit" (Figura 2-7), preparado para adaptarlo a una placa Arduino. Este tipo de GPS se diseñó para que pueda almacenar datos en una tarjeta SD, lo cual abre un amplio campo de posibilidades. Además, este GPS tiene la opción de instalar una antena para una mayor precisión y cobertura del GPS.

Esta placa dispone de unos pequeños LED que facilitan la comprobación del trabajo que realiza el GPS. El LED rojo cuando parpadea continuamente indica que está intentando geolocalizar el dispositivo. Cuando baja su cadencia a un encendido cada 6 segundos significa que ya ha obtenido la posición. A partir del momento que tiene la posición el GPS continúa actulizando la posición cada segundo con objeto de ser más preciso.

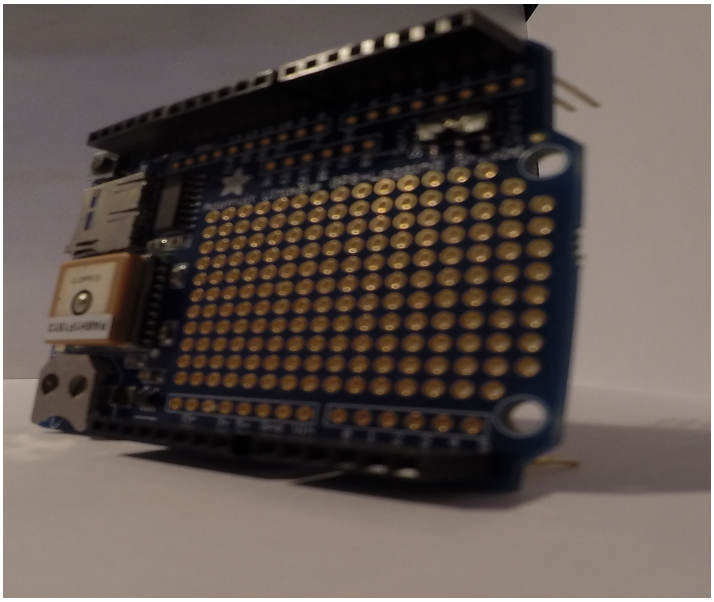

Figura 2-7 Foto de la placa GPS utilizada.

3 MONTAJE DEL DISPOSITIVO DE DETECCIÓN

3.1 Generalidades

La Armada dispone actualmente de muchos medios o técnicas para recuperar un "hombre al agua" pero, como ya se dijo anteriormente, no dispone de métodos de detección avanzados.

En la Marina Española podemos encontrar infinidad de buques distintos, ya que cada clase tiene funciones diferentes. Por lo que en este proyecto, se buscará encontrar un sistema estándar, el cual sea válido para todos y cada uno de los buques, simplemente modificando datos como puedan ser la manga y eslora del buque. Para hacerse una idea de las diferentes dimensiones de cada clase de buque se muestra la Tabla 3-1.

CLASE	ESLORA-MANGA
Juan Carlos I	231-32 m
Clase Álvaro de Bazán	146,7-18,6 m
B.A.M	93,9-14,2 m
Clase Descubierta	88,9-10,4 m
J.S de Elcano	113,1-13,1 m
Lancha de instrucción	20,5-4,9 m

Tabla 3-1 Dimensiones de algunas clases de buques de la Armada española.

Como se puede observar, las dimensiones de los buques pueden variar bastante. Durante el desarrollo del proyecto se baraja si el área de detección será un círculo o una forma ovalada. Este dispositivo de detección requiere ciertos conocimientos básicos y un estudio de nuevas tecnologías como Bluetooh, GPS o la plataforma Arduino. Desde el primer momento se intenta entender lo que en principio es más desconocido, el Arduino UNO, plataforma que va a ser la base de todo el proyecto. El Arduino requiere una fase de familiarización, que facilite su programación y conocimiento con el objetivo de obtener los resultados deseados. Por ello, se ha trabajado progresivamente desde lo más básico hasta donde se quería llegar. Es decir, a un sistema de detección que avisa cuando un hombre se

cae accidentalmente al agua. A lo largo de este apartado se describe secuencialmente como ha sido todo el desarrollo de este sistema.

3.2 Familiarización con la plataforma Arduino UNO

Con la plataforma Arduino(Figura 3-1) se hicieron proyectos de menor a mayor magnitud hasta llegar a los resultados deseados.

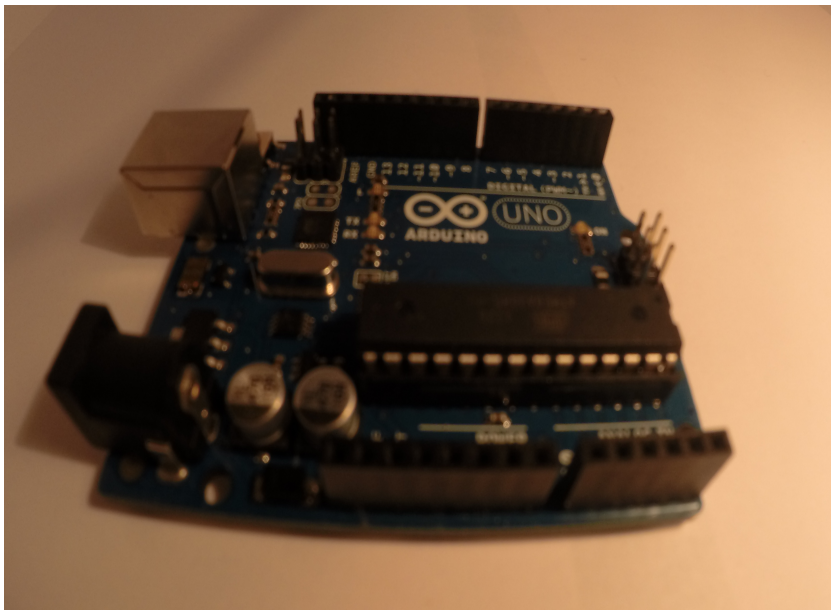

Figura 3-1 Placa Arduino UNO utilizada.

3.2.1 Aplicación "Hola mundo"

El primer trabajo que se lleva a cabo es el típico programa "Hola mundo", mas abajo se vera la Figura 3-1 que muestra como se obtiene por pantalla la frase "Hola mundo". El proceso duró bastante ya que el alumno nunca había utilizado el *software* de Arduino. Así, la configuración básica del sistema resultó un poco complicada hasta lograr conocer su funcionamiento, avanzando rápidamente a partir de ese momento.

Una vez superados los problemas de configuración, se necesitaba encontrar cómo comunicarse con el Arduino. Para ello, se utilizó un libro de uno de los creadores de esta plataforma, Massimo Banzi (Banzi). Este libro fue de gran ayuda para aprender a utilizar de manera eficiente la plataforma, ya que en el se podían encontrar varios tutoriales en los que se explicaban las funciones básicas o el uso del terminal.

La configuración de este "Hola mundo" se complicó un poco debido a que no se obtenía al principio la frase deseada a través del monitor serie. Esta operación fue muy útil ya que permitía comprobar que el compilador, el entorno de ejecución y el entorno de desarrollo estuvieran bien instalados y funcionaran correctamente. Al final, tras no saber manejar muy bien el Arduino se consigue esta primera operación que, aunque complicada, no era de difícil ejecución.

Figura 3-2 Arduino Mega con pantalla configurada con "Hola mundo".

3.2.2 Operaciones aritméticas con "processing"

Una vez entendida la plataforma (Figura 3-2), llega el momento de aprender todo el código que utiliza Arduino para realizar cálculos. Para ello, se empieza con operaciones básicas, como una suma, que son inmediatamente resueltas por el microprocesador e impresas por pantalla a través del monitor serie. El objetivo de hacer operaraciones aritméticas con Arduinos es que en un futuro tendrá que hacerlo con las posiciones GPS.

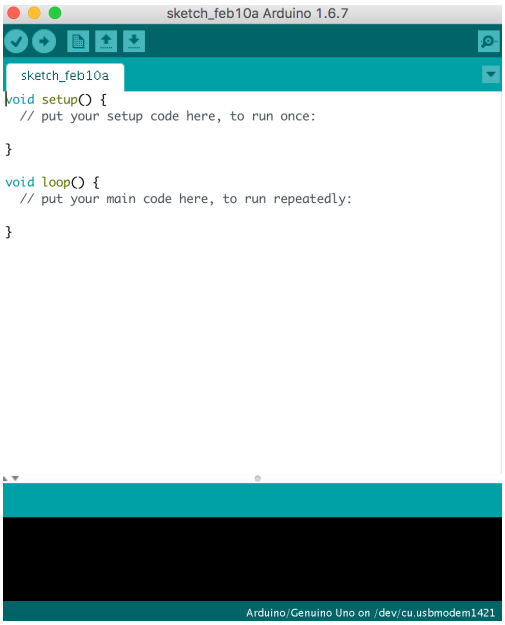

Figura 3-3 Interfaz del software Arduino para OS X.

Para simplificar el cálculo, mientras no llega el resto del material, se decide desarrollar una función de cálculo para usarla más adelante. Estas operaciones son la distancia entre dos puntos, en un espacio de dos dimensiones. Como no se va a disponer de los módulos "GPS Adafruit", se introduce la posición ficticia de dos puntos A y B. Gracias a esta operación, se puede decir que el Arduino es capaz de calcular distancias entre dos puntos.

3.2.3 Conexión Bluetooh

Para poder transmitir la posición entre Arduinos se utiliza un módulo Bluetooth en cada uno de ellos. Como ya se explicó anteriormente, los módulos utilizados son diferentes.

Inicialmente, se adquieren dos módulos Bluetooth de tipo "HC-06", queriendo configurar uno como esclavo y otro como maestro. Aquí llega el primer problema no esperado, debido a que estos módulos no pueden configurarse como maestros, ya que tienen la característica de ser siempre esclavos. Por ello, se pidió un solo modulo "HC-05" para las funciones de maestro, pudiendo así mandar información al otro módulo Bluetooth. Esta decisión no limita el prototipo, ya que el futuro dispositivo portátil (el que portaría el náufrago) llevaría el modulo "HC-05" pudiendo mandar continuamente su posición GPS al otro Arduino.

Ambos módulos pueden configurarse en distintos modos, generalmente como búsqueda o enlazado/emparejado. El "HC-05" dispone de otro modo más, el modo AT. Para detectar en qué modo se encuentra cada dispositivo existe una luz incorporada que según la frecuencia de parpadeo indicará como se encuentra.

Una vez recibido el nuevo modulo "HC-05", se procede a desarrollar el código que conecte a los dos Arduinos. El módulo "HC-06" resulta algo complicado de enlazar. Por ello, se utiliza un código que emplea comandos AT para forzar el enlace entre los módulos. El comando "CMODE" permite enlazar estos dos módulos fácilmente, ya que si se configura el HC-05 como maestro y en modo "emparejar con otro" (comando "AT+CMODE=1") los dos módulos deberían conectarse automáticamente.

El prototipo se desarrolla con un módulo como maestro y otro como esclavo para facilitar el desarrollo. Sin embargo, podría tener varios esclavos para el mismo maestro.

Estado	LED
Buscando/Esperando Conexión	Parpadea cada segundo
Modo "AT COMAND"	2 segundos encedido – 2 segundos apagado
Conectado	2 parpadeos por segundo – 2 segundos apagado

Tabla 3-2 Módulo "HC-05".

Estado	LED (Encendido/Apagado)
Esperando conexión	1 segundo encendido – 1 segundo apagado
Conexión	Luz encendida

Tabla 3-3 Módulo "HC-06".

El módulo "HC-05" no se pone solo en el modo AT ya que su activación es manual. Para ello, posee un botón en el mismo módulo que facilita la puesta en marcha de dicho modo. Puede ser complicado conseguirlo, ya que el módulo no siempre capta el botón. El módulo "HC-06" tiene una pequeña memoria que, una vez conseguida la primera conexión, la guarda como predeterminada. Así la próxima vez que estén conectados ambos Arduinos con los Bluetooth en modo "búsqueda" se conectarán automáticamente sin tener que poner el módulo "HC-05" en modo AT. Para conseguir este objetivo se debe seguir una serie de pasos:

1. Desconectar el cable de alimentación que une VCC del modulo con el pin 5V de la placa Arduino.
2. Desconectar la fuente de alimentación de la placa Arduino.
3. Pulsar el botón del módulo Bluetooth.
4. Conectar la alimentación del Arduino.
5. Conectar el cable de alimentación del módulo Bluetooth.
6. Soltar el botón una vez el LED aparezca sin parpadeo.
7. Observar que tiene el parpadeo correspondiente al modo AT.
8. Introducir en el terminal AT (contestará "OK" si está en dicho modo).
9. Posteriormente se escribe en el terminal "AT+CMODE=1" (se obtendrá el parpadeo correspondiente a estar enlazados los módulos Bluetooth).

Una vez conseguido el modo AT y su posterior conexión, es necesario no volver a poner el módulo "HC-05" en modo AT para que cuando se encienda se conecte al "HC-06".

El *software* Arduino solo deja presentar un terminal (llamado monitor seri). Sin embargo, se utilizan dos Arduinos con diferente código, por lo que se necesita poder ver los dos terminales en pantalla. Los comando para hacerlo en OS X (donde "cu.usbmodemXXXX" debe susitituirse por el puerto específico del equipo):

- `screen /dev/cu.usbmodem1421 9600`
- `screen /dev/cu.usbmodem1411 9600`

Una vez se consigue visualizar los dos terminales, se procede a comprobar si llegan datos o no. Se verifica que se obtienen datos aunque llegue la información en forma de caracteres desconocidos,

que serán traducidos posteriormente. La traducción en esta fase no es muy importante, ya que lo que se pretendía conseguir era la conexión de los módems Bluetooth.

Para conectar estos Arduinos con los módulos bluetooth se utiliza una *protoboard* (Figura 3-2), la cual permite conectar los pines de los módulos bluetooth a sus correspondiente pines en el microprocesador.

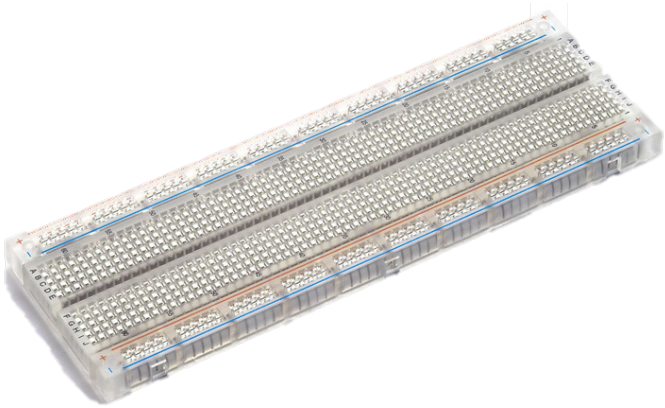

Figura 3-4 Placa *protoboard* utilizada .

Cada Arduino lleva un código "processing" diferente ya que uno de los dos Arduino (el que va con el módulo Bluetooth “HC-05”) tiene que mandar datos y la placa que vaya con el módulo Bluetooth “HC-06” tiene que recibir los datos que le mande el Arduino transmisor.

3.2.3.1 Código “Processing” para la conexión de módulos Bluetooth “HC-05”

A diferencia del “HC-06”, el “HC-05” trabaja a una velocidad de hasta 38400 baudios en lugar de los 9600 baudios. Sin embargo, la única diferencia que tienen estos dos códigos es que este mandará datos y el HC-06 los recibirá e interpretará los resultados. Para que los dos módulos se puedan comunicar se decide poner el “HC-05” a una velocidad más baja y se iguala a la del “HC-06”. A continuación se presenta el código “processing” para esta prueba :

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(2, 3); // RX | TX
//
char c = '';

void setup()
{
      Serial.begin(9600);
      Serial.println("Arduino esta listo");
      Serial.println("selecciones Both NL & CR en el monitor serial");

      // HC-05 la velocidad preestablecida parael modo AT es 38400
      BTserial.begin(9600);
}

void loop()
{

      // lee el HC-05 y va mandando por el monitor serial
      if (BTserial.available())
      {
            c = BTserial.read();
```

```
        Serial.write(c);
    }

    // lee sobre el monitor serial y manda al Arduino HC-05
    if (Serial.available())
    {
        c =  Serial.read();
        BTserial.write(c);
    }
```

3.2.3.2 Código "processing" para la conexión de módulos Bluetooth "HC-06"

Como se ve a continuación, es necesario cargar la biblioteca "SoftSerial". Se puede ver como este módulo Bluetooth trabaja a 9600 baudios en vez de 38400.

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3); // RX | TX

void setup()
{
  Serial.begin(9600);
  Serial.println("Introduzca los comandos AT")


  BTSerial.begin(9600);
}

void loop()
{

  // lee el  HC-06 y manda Arduino Serial Monitor
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // lee de  Arduino Serial Monitor y manda a  HC-06
  if (Serial.available())
  BTSerial.write(Serial.read());
}
```

3.2.4 Prueba con el módulo GPS

Es la fase previa al ensamblaje, que consiste en obtener las posiciones GPS de los Arduinos. Lo primero es instalar los módulos GPS en las placas Arduino, para lo que se necesita un soldador. El soldador ayuda a fijar los pines del módulo GPS a su placa base, para así poder conectar el modulo "Adafruit GPS" (Figura 3-4) al Arduino.

El ensamblaje deja una nueva estructura que es algo más voluminosa, con los mismos pines (de entradas y salidas) que cuando estaba la placa Arduino sin el modulo GPS. Gracias a esto, se pueden utilizar las mismas conexiones de los módulos Bluetooth sin tener que cambiar el código.

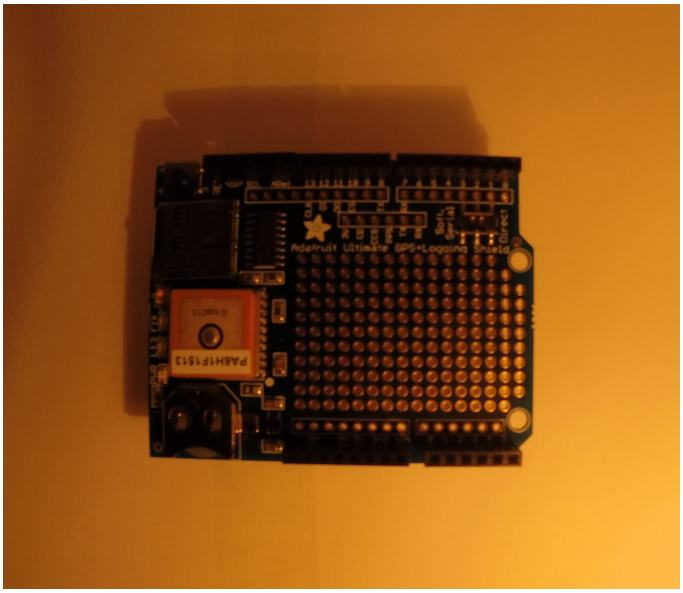

Figura 3-5 *Shield* "GPS Adafruit".

Una vez acabada la parte física, comienza la parte de programación para que el Arduino obtenga su posición GPS. Para ello, lo primero es acudir al manual (Ladyada, 2014) que facilita la propia página del producto (Adafruit).

Las posiciones se obtienen en espacios cerrados, lo cual es buena señal, ya que quiere decir que el módulo posee grandes capacidades. Esto también depende de la cobertura que haya en el lugar de estudio en concreto, siendo en algunos casos complicado que fije su posición.

En cuanto al código desarrollado, se aplican algunas modificaciones con respecto al código inicial, ya que no se desea tanta información como la que el módulo facilita. En el código, solo se necesitan dos variables llamadas "latitud" y "longitud" que reflejan, como su propio nombre indica, la posición del Arduino. Estas posiciones son las que se transmiten con un intervalo de tiempo pequeño para actualizar rápidamente la posición del Arduino en movimiento y saber lo antes posible cuando se rebasa el límite de distancia respecto al Arduino base.

3.2.5 Transmitisión de la posición GPS de un Arduino a otro

El siguiente paso es conseguir que el Arduino móvil, el que tiene instalado el módulo Bluetooth "HC-05"(Figura 3-6), transmita sus datos de "latitud" y "longitud" continuamente al Arduino fijo (con módulo "HC-06"(Figura 3-7).

Para ello, se usa de base el código ya desarrollado anteriormente, que permitía conectar los módulos Bluetooth. Como ya se comentó anteriormente, los módulos utilizados disponen de una memoria que guarda la conexión anterior. Así, conectar dichos módulos es fácil. De hecho, no hay que hacer nada, sambos módulos Bluetooth se conectan rápidamente (unos 4 segundos).

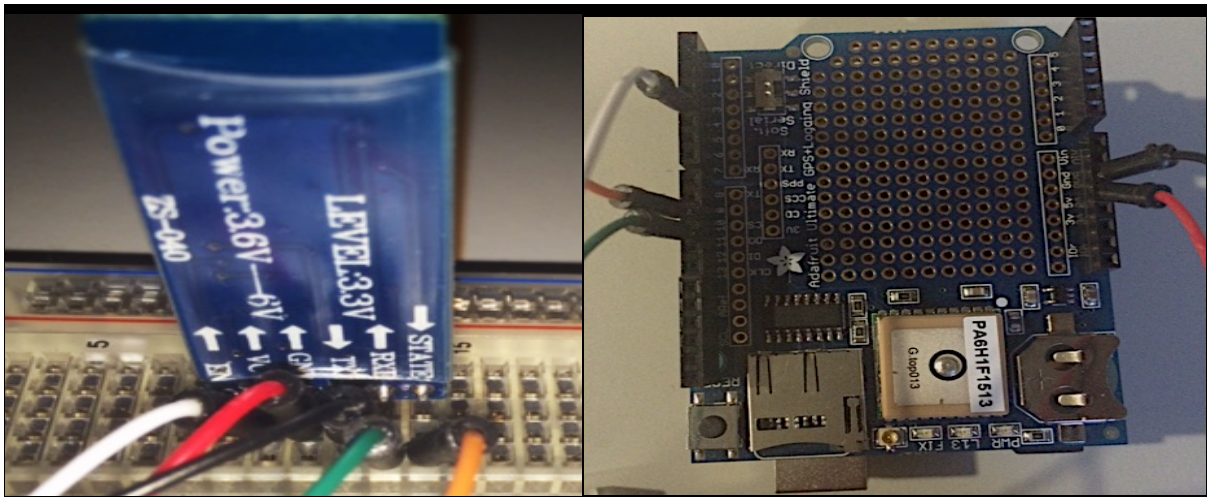

Figura 3-6 Montaje utilizado del "HC-05".

Una vez escrito todo el código, se aprecia que los caracteres recibidos en el "Arduino fijo" no son reconocibles debido a que se mandan datos en formato de coma flotante y, en este momento, errores en el ajuste de las velocidades hacían que se procesase la información mal en el receptor.

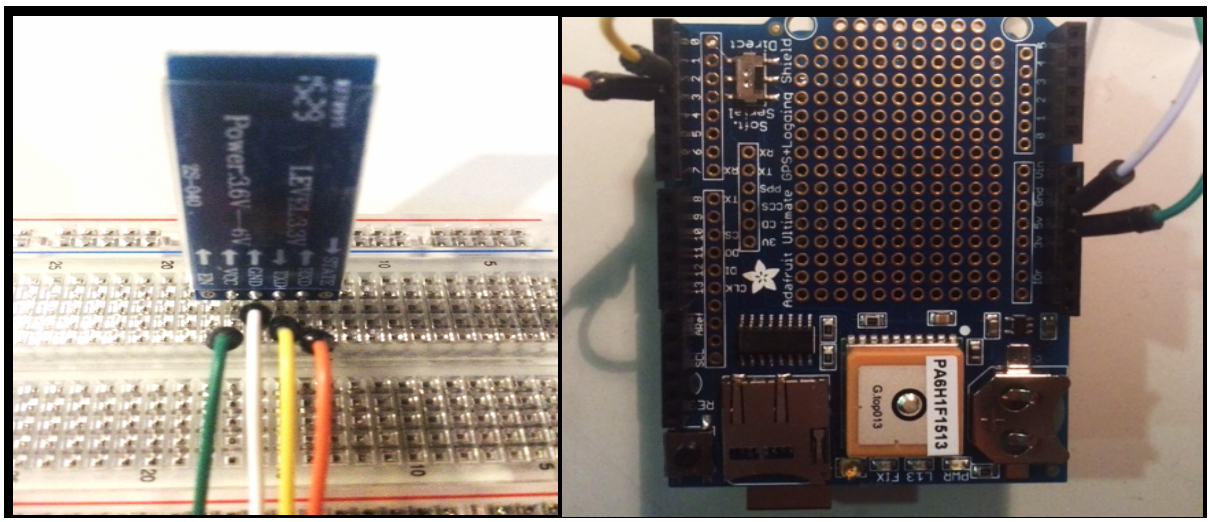

Figura 3-7 Montaje utilizado del "HC-06".

Después de varias modificaciones del código se consigue que los datos se transmitan de manera correcta. Una vez realizada la prueba de transmisión de datos GPS entre ambos Arduinos, se comprueba que llegan correctamente los datos. Posteriormente, se realizan ciertos cambios en su presentación para facilitar futuras operaciones. Lo primero que se hace es cambiar la posición de grados, minutos y segundos a simples grados. Toda la información se transmite en una única unidad.

3.2.6 Iintroducción de los cálculos de distancia

Como ya se ha explicado, el Arduino fijo necesita hacer ciertas operaciones para calcular su distancia al otro Arduino (Arduino móvil). Estas operaciones consisten en calcular la diferencia de la latitud y de la longitud de ambos Arduinos.

Si se quisiera calcular la posición con muchísima exactitud, se tendría que tener en cuenta la curvatura de la tierra. Sin embargo, para posiciones tan cercanas no es necesario ya que lo máximo que se quiere medir son 20 metros, en condiciones óptimas. Para pasar de grados a metros, se convierte todo a segundos y después se utiliza la igualdad de:

1”≈ 30,88 metros

Para pasar de grados a minutos y de minutos a segundos se utiliza una simple conversión de unidad :

1º= 60´
1´=60”

El objetivo de tener las posiciones en metros es para calcular de manera rápida a qué distancia física se encuentran los dos Arduinos. Una vez se tiene la diferencia de latitudes y longitudes en metros se puede obtener la distancia entre ambos en metros.

Una vez alcanzado este punto, se detecta un error que podría poner punto y final a este proyecto. Así, estando los dos Arduinos en la misma posición se obtienen valores distintos de latitud y longitud. Tras varias pruebas en lugares diferentes se observa que el error tiene muy poca variabilidad entre las medidas, por lo que se deduce que se podría resolver mediante calibración. La solución tomada es hacer un corrección inicial a la diferencia de latitudes y longitudes. Así, se consigue que el error no supere los 2,5 metros, el cual es un error aceptable.

Un error de 2,5 metors es aceptable porque un barco avanza a una velocidad que hace que 2,5 metros de error sea apenas nada si lo traducimos a tiempo. Para demostrar esta afirmación se muestran a continuación los cálculos que se realizan:

1. Velocidad media de un buque = 12 nudos.
2. Un buque a 12 nudos avanza 12 millas náuticas en 1 hora.
3. 1 milla náutica = 1852 metros.
4. El buque avanza 22224 metros en 60 minutos.
5. Cada minuto avanza 370 metros a 12 nudos.
6. Avanza 1 metro en 0,16 segundos.
7. Avanza 2,5 metros en menos de medio segundo (0,40 segundos).

Después de realizar estos cálculos se demuestra que es un error asumible, ya que medio segundo es un tiempo que el ser humano es casi incapaz de percibir. Cierto es que se pierde la capacidad de detectar a la persona antes de caer al agua en un barco de pequeña eslora. Habría que ver que ese medio segundo de margen de reacción que tiene el hombre para evitar pasar por debajo del sistema de propulsión, detalle que puede ser vital para su salvamento.

En el caso de un buque más grande se podría detectar perfectamente, ya que aunque suelen ir a menor velocidad y la eslora es mayor. Así, se detectaría el náufrago antes de llegar al sistema de

propulsión trasero. En función de esto, el equipo de trabajo entonces acepta el error y además considera un error aceptable de hasta de 6 metros, equivalemte a un segundo. Si el error fuera mayor que 1 segundo habría que volver a recalibrar ambos Arduinos para que no superen el segundo a 12 nudos de velocidad.

Sin errores, la idea inicial era crear un perímetro alrededor del buque. Así, se cubre toda la eslora del buque. Como no se llega a tanta precisión se decide crear un círculo como área de detección. Si se usa un círculo de detección con un diámetro igual a la eslora (Figura 3-8), una persona que cae al agua se detecta en cuanto pase por la popa o proa del buque.

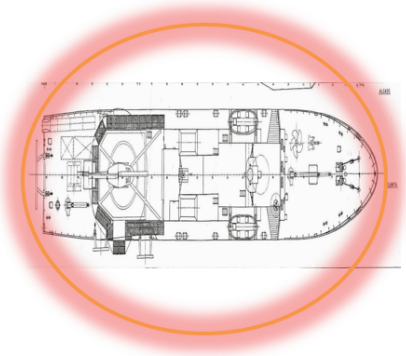

Figura 3-8 Plano de un buque con área de detección (Sacaluga Rodriguez, 2016).

Lo ideal sería usar una elipse como área de detección. Al existir cierto error se decide dejar usar un círculo porque la manga del buque suele ser un tercio de la eslora y esas dimensiones pueden llegar a ser muy pequeñas para el error que de calibración y podría producir falsas alarmas. Si el buque fuera de muy pequeñas dimisiones, para evitar una falsa alarma, su usaría un circulo de detección con diámetro mayor que la eslora.

Finalmente, sería inviable tener a una persona delante del monitor serie durante una navegación completa. En consecuencia, se decide añadir a la plataforma Arduino un LED. A nivel de código, simplemente se configura una salida más en los pines y en el caso de que la distancia exceda el perimétro se encenderá indicando que alguien se cayó al agua.

3.3 Prueba del sistema de detección

Lo primero que se hace una vez se finaliza el prototipo es ir a una plataforma para ponerlo a prueba y recoger datos. La plataforma elegida inicialmente es el patrullero "Tabarca" (Figura 3-9), que actualmente tiene como puerto base Marín y se encuentra atracado en el muelle de la Escuela Naval Militar.

Figura 3-9 Patrullero "Tabarca" (Armada.mde).

Debido a la falta de disponibilidad para acceder a él (los días en los que se pretendía realizar las pruebas) se decide hacer las pruebas en la Lancha de Instrucción "Guardiamarina Rull" (Figura 3-10). Esta lancha de instrucción tiene una estructura de fibra, una eslora de 20,5 metros y una manga de 5,1 metros. Estas dimensiones se consideran más que suficientes para llevar a cabo las pruebas con el prototipo. Así, el área de detección tendrá un diámetro de 21 metros.

Figura 3-10 Lancha de Instrucción RODMAN 66 (Armada.mde).

Durante las pruebas iniciales se detectó que la placa GPS instalada tenía ciertas dificultades para obtener rápidamente la posición en interiores. Por ello se instaló una antena(Figura 3-9), con mayor ganancia. Gracias a esta antena se podía tener el Arduino en interiores del buque y la antena a la mitad de la eslora del buque.

Figura 3-11 Imagen de la antena tomada el día de las pruebas.

La antena en la Figura 3-11 sale desplazada a la banda de babor aunque, en realidad, durante la jornada de pruebas se puso en el centro, justo a la mitad del ancho del buque. Es decir, a 2,5 metros de ambas bandas.

Una vez instalada la antena, se colocó un ordenador portátil ya que no se había instalado ninguna pantalla por la que se vieran los resultados. Se decide instalarlo en el puente del barco ya que es donde más personal hay y es donde está protegido de las condiciones ambientales. Obviamente, el portátil se utiliza este día de pruebas pero no sería necesario ya que el LED se enciende cuando un hombre cae al agua.

Figura 3-12 Imagen realizada en el puente de la lancha "GM RULL".

Con todas las partes del prototipo instaladas en el buque, la primera acción es calibrar ambas placas GPS. Para ello, situamos la antena GPS del Arduino fijo ("HC-05") en el mismo sitio que el Arduino que se utilizará como dispositivo móvil. Esta acción se toma para poder calibrar el error y tener una diferencia de latitudes y longitudes casi nula. Los resultados obtenidos en esta prueba son muy satisfactorios ya que no se obtiene casi ningún error. La calibración secundaria que se hace es mínima. De hecho, en esta prueba solo se tuvo que modificar en 1,5 metros la latitud, ya que la longitud salió exactamente la misma para ambos Arduinos.

Ya calibrados los Arduinos, se procede a obtener resultados reales a lo largo del buque. Como se podrán preguntar si el dispositivo móvil está conectado a la red eléctrica. Pues bien, para poder hacer autónomo el Arduino que lleva el Bluetooth "HC-05" (por lo que también sería el portado por el personal) es necesario instalarle alguna fuente de energía que le permita desplazarse de manera fácil. La opción más sencilla es poner una batería portátil que suministre energía durante un largo periodo de tiempo. Como es un prototipo y no se le da importancia al tamaño que pueda ocupar el dispositivo, se instala una batería de 10400 mAh (Figura 3-13). Una vez se quieren obtener los resultados por el monitor serie del Arduino fijo, el equipo lee como posición del Arduino móvil cero. La latitud que recibe es 0,000000 y la longitud que obtiene es también 0,000000, por lo que se deduce que hay un error en la transmisión de información.

Figura 3-13 Imágenes del Arduino móvil realizadas en la "GM RULL".

Para solucionar este problema, se comprueban inicialmente las cosas mas básicas:

1. Lo primero que se hace es comprobar si están enlazados ambos Bluetooth, ya que podría ser que no tuvieran cobertura Bluetooth a dicha distancia. Ambos Arduinos están enlazados, así que no es fallo de cobertura Bluetooth.

2. Se procede a realizar una segunda comprobación, ya que la anterior no fue efectiva, donde se comprueba si el Arduino móvil tiene el programa cargado. Para ello, se vuelve a cargar el programa y se vuelve a comprobar que el "HC-06" recibe datos. El resultado de esta prueba es negativo ya que este no es el error.

3. Se baraja la opción de si el Arduino está perdiendo información cuando se desconecta del ordenador portátil. Esta opción se descarta, ya que anteriormente se había desconectado y vuelto a conectar y funcionaba perfectamente.

4. Por último, revisando el código se observa que tenía muchas funciones de envío de información al terminal serie, que no existe en modo autónomo ya que no está conectado a ningún monitor serie para sacar en pantalla los datos. Esto provocaba que se bloqueara el sistema y no se pudiera mandar la información. La solución fue comentarlos en el código. Una vez hecho esto se solucionaron los problemas y los resultados obtenidos fueron más que satisfactorios después de todas las complicaciones.

Figura 3-14 El alumno posicionando el Arduino móvil en la proa de la "GM RULL".

Una vez todo funciona se procede a obtener resultados colocando el Arduino móvil en diferentes zonas del buque. Como el resultado es positivo, se va reduciendo la distancia para ver si se puede llegar a una discriminación aceptable. Se llega a poner una separación de hasta dos metros y se obtienen los resultados que pueden ver en la Figura 3-15.

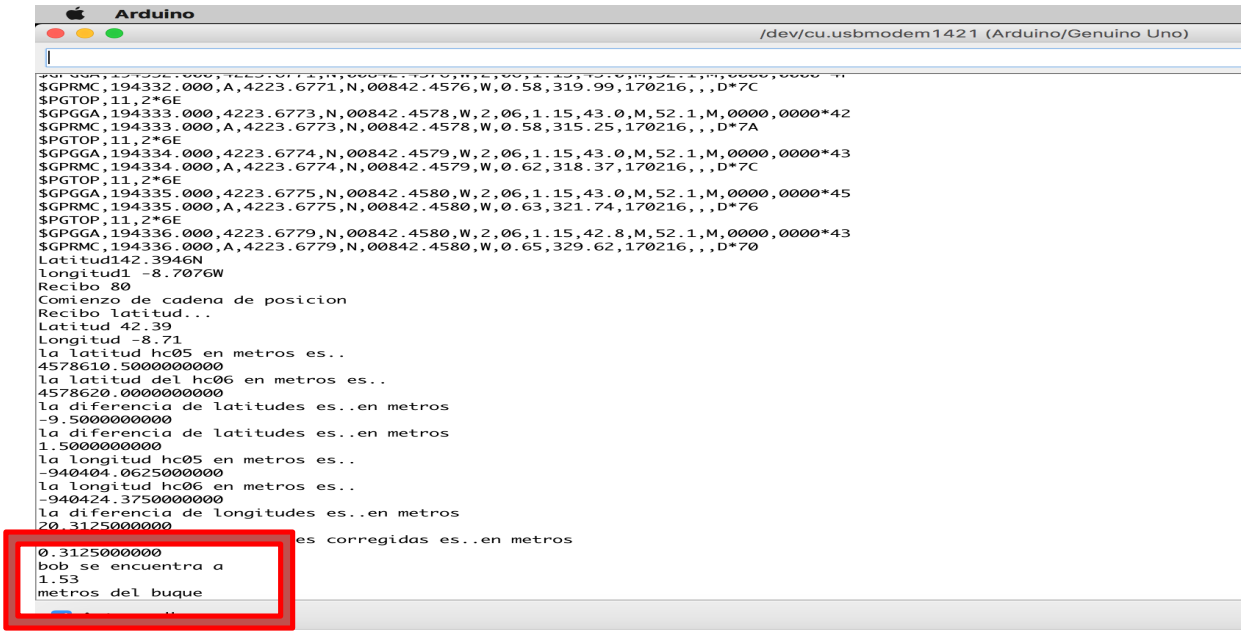

Figura 3-15 Resultados obtenidos con los Arduinos a una distacia de 2 metros.

En los resultados de la última línea de la Figura 3-15, se puede leer una frase que dice "bob se encuentra a 1,53 metros del buque". Esta se debe a una separación de dos metros por la misma banda (babor, donde esta encuentra la antena). Se podría decir que el error en esta prueba es muy satisfactorio, ya que se estima inferior a 0,3 metros.

En alguna de las pruebas anteriores fue necesario recalibrar el módulo GPS debido a que inicialmente obtenía una posición que no era del todo exacta. A lo largo de la mañana, se anotan los diferentes errores que se obtienen para intentar llegar a un patrón de error y calibrar el dispositivo. Como ya se dijo anteriormente, esta precisión no tiene una gran relevancia, debido a la decisión que se tomó al elegir un circulo como área de detección. De cara a la posible realización de un a elipse (como área de detección), se sabrá que este GPS puede llegar a una precisión bastante correcta.

Para que sea más fácil de entender el código, se muestra un diagrama de flujo en la siguiente Figura:

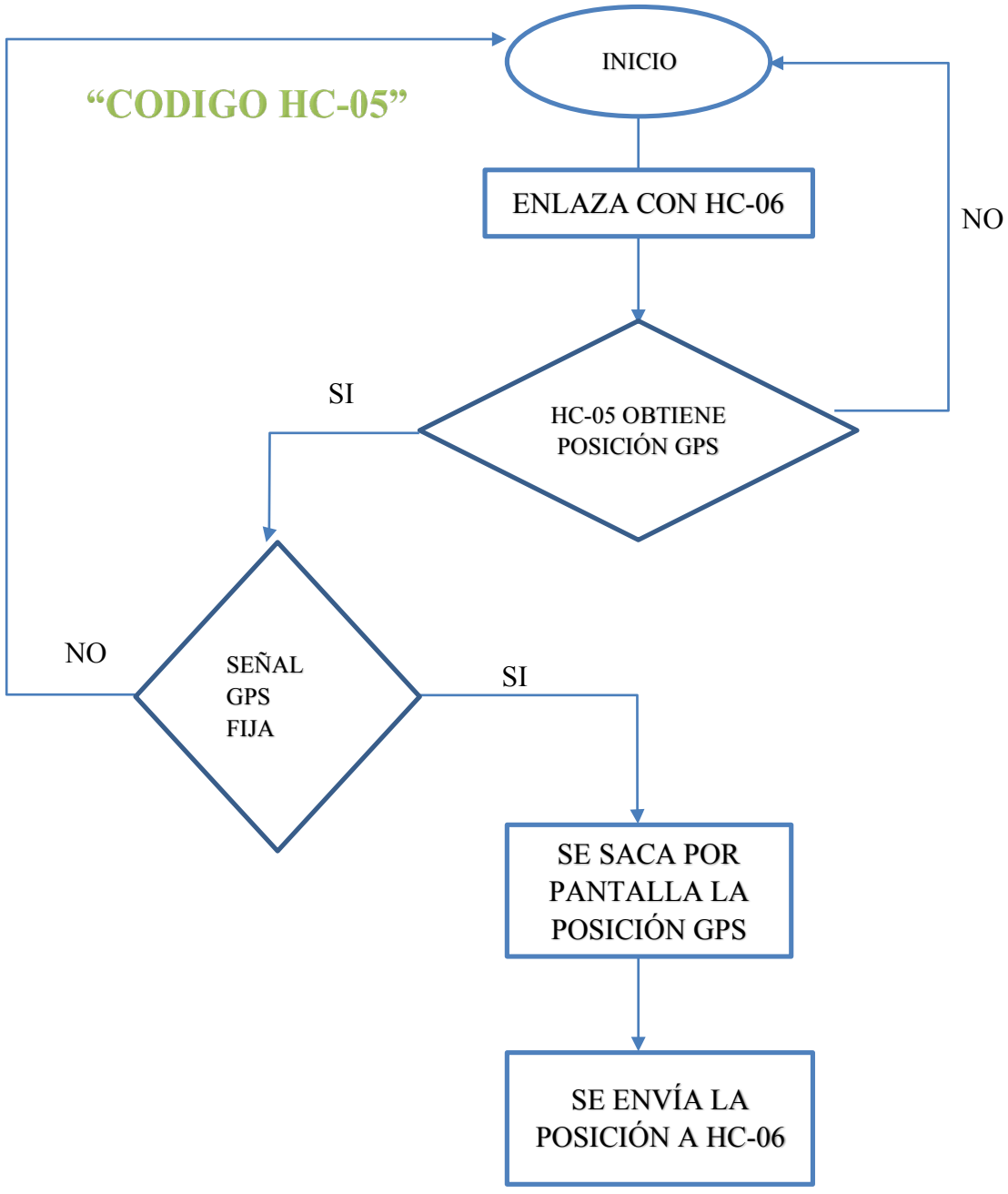

Figura 3-16 Diagrama de flujo del funcionamiento del módulo con el HC-05

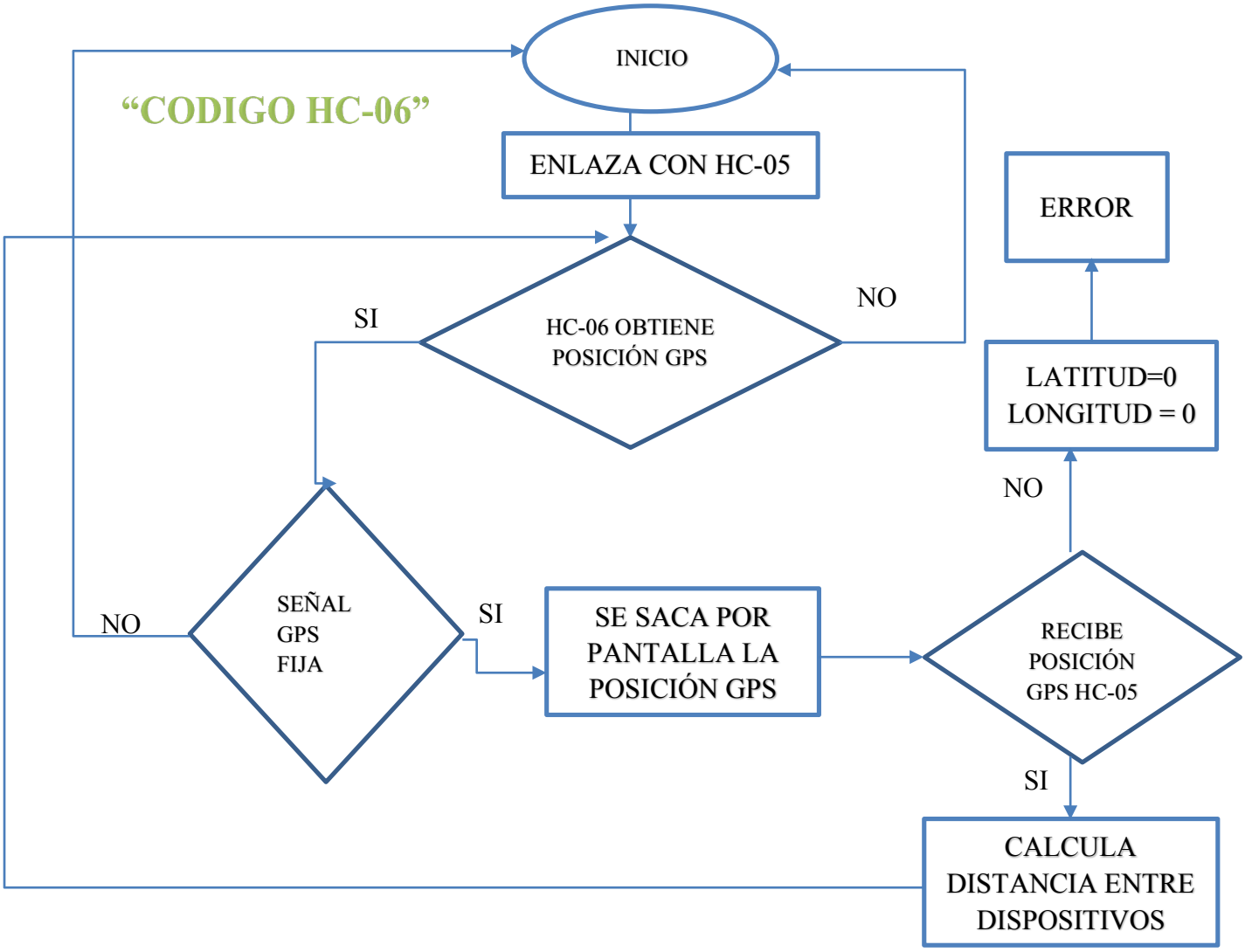

Figura 3-17 Diagrama de flujo del funcionamiento del módulo con el HC-05

4 CONCLUSIONES Y LÍNEAS FUTURAS

4.1 Introducción

Después del análisis de los resultados, se puede decir que este diseño es válido para buques con una eslora inferior a 30 metros, debido a las limitaciones del enlace Bluetooth. Si embargo, la idea que se saca es que se ha desarrollado un prototipo capaz de detectar cuando un hombre cae al agua, algo que podría ser adaptarse a los buques. El sistema es válido, a falta de fabricar la caja en la que se desease instalar.

Los Arduinos se han comportado de manera eficaz durante todo el proceso, aunque es cierto que es un microcontrolador que en ciertos momentos puede dar algunos fallos de leve relevancia.

En cuanto a los módulos GPS, son totalmente válidos aunque se recomienda usarlos con las antenas adaptables que traen. Si se fuera a implantar desde ahora se aconseja usar otro módulo más preciso y con mejor antena para no tener que añadir ninguna, ya que esto puede causar problemas en las conexión con la placa, que es bastante delicada.

Al no existir un referente previo, se fueron observando las limitaciones que tenían los materiales seleccionados para su desarrollo. Esto fue un impulso positivo para la construcción del prototipo. Una vez sabido que es posible, se plantean unas líneas futuras que servirán de ayuda a la persona que desee continuar este prototipo.

4.2 Conclusiones y líneas futuras

Se presentan varias conclusiones obtenidas y se explica el nivel de alcance de los objetivos establecidos al inicio del TFG. Se presentan, asimismo, las posibles líneas futuras en las que se podría seguir trabajando para mejorar y ampliar el TFG presentado.

4.2.1 Habitáculo para el prototipo

El mismo prototipo desarrollado parece algo grande pero con esto se quiere decir que puede usarse tal y como está. En el casco de maniobra que se usa en exteriores del buque se podría adaptar una caja de plástico estanca donde iría todo (GPS, Arduino y módulo Bluetooth) introducido en su interior. Esta caja iría fijada con cuatro tornillos que podrían quitarse en caso de avería. La carga se

haría en la salida por el interior del casco(Figura 4-1), ya que este puesto permanece relativamente estanco y se evita que entre agua. Esta caja debería ser intercambiable en otros cascos de diferente modelo, ya que no siempre se dispone del mismo.

Figura 4-1 Casco de maniobra en el que seria instalado en prototipo actual.

Además, cabe la posibilidad de que se esté trabajando en exteriores y no sea para una maniobra. La "Force Protection", utilizada en la guerra asimétrica, está a la orden del día, por lo que debe ensayarse de manera habitual. Este tipo de operaciones implica tener mucho personal en exteriores, el cual no lleva el casco de maniobra sino que utiliza uno especifico para "Force Protection". En consecuencia, sería de gran utilidad que este casco (Figura 4-2) pudiera adaptarse para llevar la caja con el dispositivo. Habría que analizar ambos cascos para conseguir unas dimensiones comunes con la misma tornillería.

Figura 4-2 Posible casco para "Force Protection".

4.2.2 Pulsera

Desde el primer momento se pensó que este proyecto tendría que tener un final con una propuesta de futuro, el diseño de una pulsera. La pulsera podría servir para que todo el personal que saliese a exteriores se la pusiese y así estar controlodo.

La pulsera debería fabricarse de manera que pudiera instalarse la placa Arduino, el módulo GPS y el Bluetooth. Para que esta pulsera no fuese gigantesca se aconseja utilizar un micro Arduino (Feng, 2013) que tiene un tamaño muy reducido, además de unas buenas características técnicas. En este proyecto se utiliza una placa GPS bastante grande por su facilidad de adaptar al Arduino, pero podría reducirse a un tamaño tan pequeño como el del Arduino micro (Figura 4-3).

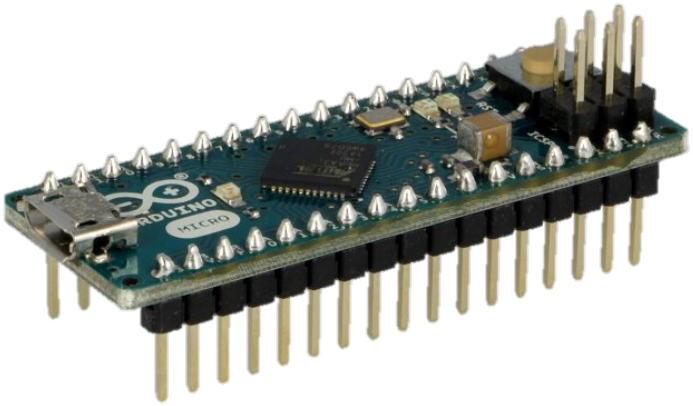

Figura 4-3 Imagen del Arduino micro (T.M.E).

Las dimesiones del Arduino UNO son 68,66x53,4 mm frente a las diminutas dimensiones de la versión micro que tiene 25,4x27,9 mm. Las dimensiones del Arduino micro son compatibles con una pulsera de mano. Para introducir los cuatro elementos esenciales (GPS, Bluetooth, Arduino y batería) bastaría con dejar un espacio hueco, con una salida para poder cargarla por la entrada USB. La batería no duraría mucho tiempo debido a su limitación de espacio físico pero, si en las salidas a exteriores de buque hubiera unos cargadores donde las pulseras (Figura 4-4) estarían siempre operativas. De hecho, las guardias en un buque suelen durar cuatro horas y no siempre hay gente fuera. Obviamente, habría que hacer un estudio para saber el número de pulseras que puede necesitar cada tipo de buque y el número de repuestos.

Figura 4-4 Imagen de diseño de una posible pulsera.

En cuanto al uso de pulseras, el personal que las utilizara solo tendría que encender el dispositivo una vez equipadas. En el caso de estropearse alguna, el servicio de comunicaciones del buque debería de estar preparado para repararlas. Para preparar al personal, se aconseja impartir un curso a un miembro de la dotación por buque. Así, en caso de salir a navegar durante un largo periodo de tiempo (unos meses) la dotación podría utilizar este sistema.

Existe la posibilidad de utilizar un Arduino diferente a la versión micro para la realización de esta pulsera. La casa Adafruit desarrolló unos años un novesoso Arduino llamado "Seeeduino Film" (Szczys, 2010). Esta versión tiene la singularidad de que han utilizado un material de resina para producir un circuito impreso flexible (FPC). El objetivo de este diseño es, además, mantenerlo modular. Como se puede ver en la Figura 4-3, cada uno de los cuadrados posee un componente diferente del Arduino. A la derecha se puede ver el modulo USB, a continuación el regulador de potencia, el microprocesador y, finalmente, las conexiones de los periféricos restantes. Cada una de estas partes puede cortarse en función de lo que se necesitase.

Figura 4-5 Seeeduino Film (Szczys, 2010).

4.2.2.1 Pulsera con botón de emergencia

Con el avance del proyecto surge la cuestión de si la seguridad que da este dispositivo puede provocar la despreocupación de la dotación por saber si está toda el mundo a bordo. Por ello, se propone la opción de adaptar a la pulsera un botón de emergencia para que la propia persona accidentada pueda pulsarlo y activar una alarma en el puente del buque.

4.2.3 Alarma de error

Se deja abierta la posibilidad de instalar otro LED (Figura 4-6) de color diferente al de alarma, que informe de errores de comunicación. Si la información recibida por el "HC-06" tiene valor 0, se encendería esta luz, detectándose así la existencia de un error.

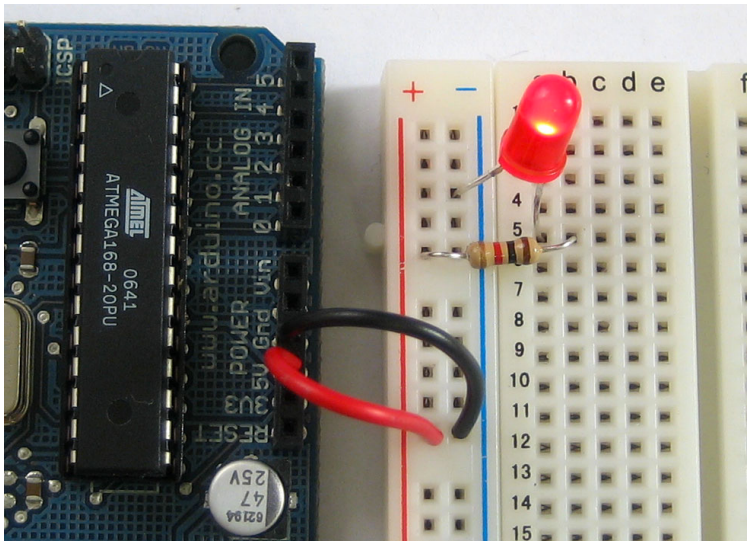

Figura 4-6 Arduino con LED.

4.2.4 Uso de la posición GPS del buque

Para mayor precisión del GPS, se propone utilizar la posición que obtiene el buque. Esta posición GPS, es utilizada por casi todos los software que ayudan a la navegación. La posición usada por estos software(proviene siempre de la misma fuente), así todos los dispositivos pueden ser coordinados en tiempo real.

En un futuro, lo ideal sería que este dispositivo de hombre al agua se pudiera integrar con el resto, para ello es necesario que la fuente de la posicion del Arduino "base" sea la misma que la del resto, por eso se aconseja sustituir la placa Adafruit GPS del Arduino base y así poder usar la misma posición que el plotter del buque o que cualquier otro dispositivo.

4.2.5 Uso de la corredera del propio buque

Una vez detectado que el naufrago está en el agua, se deberá realizar su búsqueda lo antes posible, para que sea más rápida se propone instalar una entrada de la corredera en el Arduino "base".

Gracias a esta entrada el Arduino podría dar el rumbo al naufrago, ya que el Arduino del naufrago transmite su posición. Este avance facilitaría al oficial de guardia en puente la toma de decisión, y as u vez evitaría posibles errores humanos.

4.2.6 Zigbee

Una de las mayores limitaciones de este prototipo es la distancia de alcance del Bluetooth, por ello se investigan otras alternativas que puedan ofrecer un mayor alcance.

Zigbee es un protocolo de comunicaciones inalámbrico que se basa en el estandar de comunicaciones para redes inalámbricas IEEE 802.15.4.

Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Es especialmente útil para redes de sensores en entornos industriales, médicos o domóticos.

Posiblemente Zigbee sea la más adecuada para este proyecto, ya que el alcance normal con antena dipolo en visión directa suele ser aproximadamente (tomando como ejemplo el caso de MaxStream, en la versión de 1mW de portencia de 100m y en interiores de unos 30m (Blogelectrónica)

5 BIBLIOGRAFÍA

Adafruit. (s.f.). *Adafruit*. (A. S. GPS, Productor) Recuperado el 5 Febrero de 2016, de WWW.ADAFRUIT.COM.

Air Force. (5 de Septiembre de 2005). *NAVSTAR GPS.* Recuperado el 20 Enero de 2016, de arnold.af.mil: http://www.arnold.af.mil/aedc/systems/85-1156.ht

Arduino, T. (s.f.). *Slide share.* Recuperado Febrero de 2016, de http://es.slideshare.net/tefayanez/arduino-42996209

Armada.mde. (s.f.). Recuperado Febrero de 2016, de http://www.armada.mde.es/ArmadaPortal/page/Portal/armadaEspannola/multimedia_galeria/prefLang_es/07_patrulleros--03_anaga_es

Banzi, M. *Introducción a Arduino.* ANAYA MULTIMEDIA.

Bluehack web. (2005).

Bluetooth SIG. (s.f.). *bluetooth*. Recuperado el 23 de Enero de 2016, de https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics

Bosch. (2015). *www.bosch.com*. Obtenido de sistema de deteccion de incendios bosch: http://www.duranelectronica.com/docs/24_1528_Bosch_1200-5000-v01.pdf

Etrex, L. g. (s.f.). *Activa-t.* Recuperado Febrero de 2016, de https://activatexp.wordpress.com/2012/10/25/serie-garmin-etrex-analisis/

Feng, B. (16 de Septiembre de 2013). Microduino: Arduino del tamaño de una moneda. *hipertetual*.

Garcia del Valle, C. (2016). *Video Surveillance System based on Raspberry Pi and Pi Camera.* TFG, ENM, CUD.

GPS, S. d. (Enero de 2016). Recuperado el 24 de Enero de 2016, de wikipedia: https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

Ladyada. (2014). *Adafruit Ultimate GPS Logger Shield.*

Murias, L. (2007). Sistemas de seguridad y control en los buques de crucero y terminales portuarias de pasajeros. (T. T. Red, Recopilador)

Objets, W. H. (Enero de 2016). *le monde informatique.* Obtenido de http://www.lemondeinformatique.fr/actualites/lire-wifi-halow-un-wifi-dedie-a-l-internet-des-objets-63463.html

Prometec. (s.f.). *Prometec*. Recuperado el 2016 de 2 de 2, de www.prometec.net

Rob Blanco, w. (s.f.). *wikipedia*. Obtenido de bluetooth: https://es.wikipedia.org/wiki/Bluetooth_(especificaci%C3%B3n)#/media/File:Bluetooth_network_topology.png

Sacaluga Rodriguez, B. (Febrero de 2016). *Blog Benito Sacaluga*. Obtenido de http://benitosacalugarodriguez.blogspot.com.es/2013_09_01_archive.html

Sistema de Posicionamiento Global. (s.f.). *GPS.* Recuperado el enero de 2016, de wikipedia: https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

Szczys, M. (Junio de 2010). *hackaday*. Obtenido de http://hackaday.com/: http://hackaday.com/2010/06/29/fpc-arduino/

T.M.E. (s.f.). *Transfer Multisort Elektronik*. Recuperado el 16 de febrero de 2016, de http://www.tme.eu/es/: http://www.tme.eu/es/details/a000053/kits-de-arranque-arduino/arduino-micro/?brutto=es&currency=eur&gclid=CKCaupmZi8sCFesJwwodM84D1Q

Web arduino. (s.f.). Recuperado el 20 de Enero de 2016, de www.arduino.cc

Web Bluetooth.

Web de La Moncloa. (s.f.). Recuperado el 13 de Enero de 2015, de http://www.lamoncloa.gob.es

Web of the internet of things. *the internet of things.*

Wi-Fi Alliance. (s.f.). *Wi-Fi*. Recuperado el Enero de 2016, de http://www.wi-fi.org/

ANEXO I: CÓDIGOS USADOS PARA PROTOTIPO

El código utilizado para el Arduino con el "HC-05" es el siguiente:

```
// Test code for Adafruit GPS modules using MTK3329/MTK3339 driver
//
// Este código muestra como la placa GPS obtiene la posiciónn
// Esto permite tener mas libertadad para realizar cualquier objetivo
// Cuando una frase NMEA frase esta disponible! Entonces se accede a la
// información deseada
//
// Probado y funciona bien  con el módulo de GPS Adafruit Último
// usando MTK33x9 chipset
//    ------> http://www.adafruit.com/products/746
// coja una en la Adafruit electronics shop
// y ayuda a abrir un fuente hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>

float latitud;
float longitud;

SoftwareSerial BTserial(11, 10); // RX | TX
// Conecta el HC-05 TX al Arduino pin 11 RX.
// Conecta el HC-05 RX al Arduino pin 10 TX.
//

byte * c = NULL;
byte d = 'P';


// Si usa un modulo GPS:
// Enchufa GPS Power pin al 5V
// conecta pin tierra del GPS a masa
// Si usa software serial (sketch example default):
//   conecta el pin TX del GPS al pin Digital 3
//   conecta el pin RX del GPS al pin Digital 2

// si usa el Adafruit GPS shield, cambi
// SoftwareSerial mySerial(8, 7); -> SoftwareSerial mySerial(8, 7);
// y asegurese que el botón esta en SoftSerial

// si usa software serial, guarde la línea sin tocar
// (puede cambiar los números de los pines si cambia el codigo):
SoftwareSerial mySerial(8, 7);



Adafruit_GPS GPS(&mySerial);


// resetea GPSECHO a 'false' para apagar el eco de los datos GPS hacia la consola
Serial
// resetea a 'true' si usted quiere tomar datos y escuchar la rafaga de franses
del GPS
#define GPSECHO  true
```

```
// esto guarda la pista de si usamos el interrumptor
// normalmente apagado
boolean usingInterrupt = false;
void useInterrupt(boolean); // Func prototype keeps Arduino 0023 happy

void setup()
{

  // ponga velocidad 9600
  Serial.begin(9600);
  /*Serial.println("Adafruit GPS library basic test!");*/

  // 9600 NMEA es la velocidad de transmisión en baudios de base para Adafruit MTK
el GPS - algunos 4800
  GPS.begin(9600);

  // incomente esta línea para conectar RMC (el mínimo recomendado) y GGA (fijar
datos) incluyendo la latitud
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
  // incomente esta línea para conectar sólo " el mínimo recomendado "
  //GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCONLY);
  // Para parsing data, no aconsejamos usar nada de eso RMC (sólo) o RMC+GGA desde
entonces
  // al analizador no le importan otras frases al mismo tiempo

  // cambie el ratio
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz
  // para que el código parsing funcione bien y tenga tiempo de sacar los datos
  // imprima por pantalla no se recomienda usar nada por encima de 1 Hz

  // comentar si se quiere la antena en silencio
  GPS.sendCommand(PGCMD_ANTENNA);

  // lo bueno de este código es que tiene time0 para apagarsse
  // cada 1 milisegundo, y leer los datos GPS para ti. Esto hace
  // el código del loop mucho mas facil!
  useInterrupt(true);

  delay(10000);
  // pregunte por la firmware version
  //mySerial.println(PMTK_Q_RELEASE);
  //GPS.sendCommand(PMKT_Q_RELEASE);

  // HC-05 velocidad serial base para modo AT es 38400 o menos en este caso 9600
    BTserial.begin(9600);









}


// cada milisegundo es interrumpido, busca cualquier dato nuevo del GPS, y lo
guarda
SIGNAL(TIMER0_COMPA_vect) {
  char c = GPS.read();
  // si quiere quitar fallos este es un buen momento
#ifdef UDR0
  if (GPSECHO)
```

```
    if (c) UDR0 = c;
    // escribir directamente a UDR0 es mucho mas rápido que a Serial.print
    // pero solo un carácter puede ser escrito a la vez
#endif
}

void useInterrupt(boolean v) {
  if (v) {
    // Timer0 es usado todavía por millis() - solo se interrumpirá
    // en el medio y se llamara funcion "Compare A"
    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    // ya no llame a ninguna funcion COMPA
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}

uint32_t timer = millis();
void loop()                     // se hace repetidas veces, una detrás de otra
{

  if (! usingInterrupt) {
    // read data from the GPS in the 'main loop'
    char c = GPS.read();
    // este es un buen momento para revisar el código
    if (GPSECHO)
      if (c) Serial.print(c);
  }

  // si recibe una frase puede comprobarse el checksum, parse ...
  if (GPS.newNMEAreceived()) {
    // una cosa difícil aquí es que si es la impresión de la sentencia NMEA
    // O terminamos por no escuchar y la captura de otras frases!
    // por lo que tener mucho cuidado si se utiliza OUTPUT_ALLDATA y trytng para
imprimir los datos

    //Serial.println(GPS.lastNMEA());   //esto también establece newNMEAreceived()
flag to false

    if (!GPS.parse(GPS.lastNMEA()))   // esto también establece newNMEAreceived()
flag to false
      return;  // podemos dejar de analizar una sentencia en cuyo caso deberíamos
esperar a otro
  }

  // si millis() solo tendremos que restablecerla
  if (timer > millis())  timer = millis();

  // cada 2 seconds o imprimir las estadísticas actuales
  if (millis() - timer > 2000) {
    timer = millis(); // reset the timer

    /*Serial.print("\nTime: ");
    Serial.print(GPS.hour, DEC); Serial.print(':');
    Serial.print(GPS.minute, DEC); Serial.print(':');
    Serial.print(GPS.seconds, DEC); Serial.print('.');
    Serial.println(GPS.milliseconds);
    Serial.print("Date: ");
    Serial.print(GPS.day, DEC); Serial.print('/');
    Serial.print(GPS.month, DEC); Serial.print("/20");
```

```
    Serial.println(GPS.year, DEC);
    Serial.print("Fix: "); Serial.print((int)GPS.fix);


    Serial.print(" quality: "); Serial.println((int)GPS.fixquality); */

    if (GPS.fix) {
      /*Serial.print("Latitud ");*/
      latitud=(GPS.latitudeDegrees);
     /* Serial.print(latitud,4);
      Serial.println(GPS.lat);
      Serial.print("longitud ");*/
      longitud=(GPS.longitudeDegrees);
      /*Serial.print(longitud, 4);
      Serial.println(GPS.lon);*/
     /* Serial.print("Location (en grados, funciona con google maps): ");
      Serial.print(GPS.latitudeDegrees, 4);
      Serial.print(", ");
      Serial.println(GPS.longitudeDegrees, 4);

      Serial.print("Speed (knots): "); Serial.println(GPS.speed);
      Serial.print("Angle: "); Serial.println(GPS.angle);
      Serial.print("Altitude: "); Serial.println(GPS.altitude);
      Serial.print("Satellites: "); Serial.println((int)GPS.satellites);*/

      BTserial.write(d); // envía por  BLUETOOTH "d"
      /*Serial.write(d);*/
      //c = (byte *) &latitud;
      /*Serial.println(latitud);*/
      BTserial.write((char *)&latitud,4);
      //c = (byte *) &longitud;
      /*Serial.println(longitud);*/
      BTserial.write((char *)&longitud,4);

    }
    BTserial.write('k');
  }
}
```

El código utilizado para el Arduino con el "HC-06" es:

```
// Test code for Adafruit GPS modules using MTK3329/MTK3339 driver
//
// Este código muestra como la placa GPS obtiene la posiciónn
// Esto permite tener mas libertadad para realizar cualquier objetivo
// Cuando una frase NMEA frase esta disponible! Entonces se accede a la
// información deseada
//
// Probado y funciona bien  con el módulo de GPS Adafruit Último
// usando MTK33x9 chipset
//    ------> http://www.adafruit.com/products/746
// coja una en la Adafruit electronics shop
// y ayuda a abrir un fuente hardware & software! -ada

#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>

float latitud1;
float longitud1;
float latitud_envio;
float latitud_base;
float smlat;
float smlat_2;
float long_envio;
float long_base;
float sumlon;
float sumlon_2;
float operacion;
float distancia;
float latitud;
float longitud;
float sumlon_corregida;
float smlat_corregida;
SoftwareSerial BTserial(2, 3); // RX | TX
// Conecta el HC-06 TX al pin Arduino 3.
// Conecta el HC-06 RX al pin Arduino  2.
//

byte c[4];
byte d;
int i = 0;


// Si usa un modulo GPS:
// Enchufa GPS Power pin al 5V
// conecta pin tierra del GPS a masa
// Si usa software serial (sketch example default):
//   conecta el pin TX del GPS al pin Digital 3
//   conecta el pin RX del GPS al pin Digital 2

// si usa el Adafruit GPS shield, cambi
// SoftwareSerial mySerial(8, 7); -> SoftwareSerial mySerial(8, 7);
// y asegurese que el botón esta en SoftSerial

// si usa software serial, guarde la línea sin tocar
// (puede cambiar los números de los pines si cambia el codigo):
SoftwareSerial mySerial(8, 7);
```

```
Adafruit_GPS GPS(&mySerial);


// resetea GPSECHO a 'false' para apagar el eco de los datos GPS hacia la consola
Serial
// resetea a 'true' si usted quiere tomar datos y escuchar la rafaga de franses
del GPS
#define GPSECHO  true


// esto guarda la pista de si usamos el interrumptor
// normalmente apagado
boolean usingInterrupt = false;
void useInterrupt(boolean); // Func prototype keeps Arduino 0023 happy


void setup()
{
    pinMode(12,OUTPUT); // ponga el pin 12 como salida


  Serial.begin(9600);
  Serial.println("Adafruit GPS library basic test!");


  GPS.begin(9600);

including altitude
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);

RMC+GGA since



  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);   // 1 Hz update rate


  GPS.sendCommand(PGCMD_ANTENNA);


  useInterrupt(true);

  delay(10000);

  mySerial.println(PMTK_Q_RELEASE);



    BTserial.begin(9600);



}


SIGNAL(TIMER0_COMPA_vect) {
  char c = GPS.read();

#ifdef UDR0
  if (GPSECHO)
    if (c) UDR0 = c;

#endif
```

```
}

void useInterrupt(boolean v) {
  if (v) {

    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    // do not call the interrupt function COMPA anymore
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}

uint32_t timer = millis();
void loop()                     // run over and over again
{

  if (! usingInterrupt) {
    // read data from the GPS in the 'main loop'
    char c = GPS.read();
    // if you want to debug, this is a good time to do it!
    if (GPSECHO)
      if (c) Serial.print(c);
  }

  // if a sentence is received, we can check the checksum, parse it...
  if (GPS.newNMEAreceived()) {

    //Serial.println(GPS.lastNMEA());   // this also sets the newNMEAreceived()
flag to false

    if (!GPS.parse(GPS.lastNMEA()))   // this also sets the newNMEAreceived() flag
to false
      return;  // we can fail to parse a sentence in which case we should just
wait for another
  }


  if (timer > millis())  timer = millis();


  if (millis() - timer > 2000) {
    timer = millis(); // reset the timer

    /*Serial.print("\nTime: ");
    Serial.print(GPS.hour, DEC); Serial.print(':');
    Serial.print(GPS.minute, DEC); Serial.print(':');
    Serial.print(GPS.seconds, DEC); Serial.print('.');
    Serial.println(GPS.milliseconds);
    Serial.print("Date: ");
    Serial.print(GPS.day, DEC); Serial.print('/');
    Serial.print(GPS.month, DEC); Serial.print("/20");
    Serial.println(GPS.year, DEC);
    Serial.print("Fix: "); Serial.print((int)GPS.fix);
    Serial.print(" quality: "); Serial.println((int)GPS.fixquality); */

    if (GPS.fix) {
     Serial.print("Latitud1");
     latitud1=(GPS.latitudeDegrees);
      Serial.print(latitud1,4); Serial.println(GPS.lat);
      Serial.print("longitud1 ");
```

```
      longitud1=(GPS.longitudeDegrees);
      Serial.print(longitud1, 4); Serial.println(GPS.lon);
      delay(5000);
      }
     /* Serial.print("Location (in degrees, works with Google Maps): ");
      Serial.print(GPS.latitudeDegrees, 4);
      Serial.print(", ");
      Serial.println(GPS.longitudeDegrees, 4);

      Serial.print("Speed (knots): "); Serial.println(GPS.speed);
      Serial.print("Angle: "); Serial.println(GPS.angle);
      Serial.print("Altitude: "); Serial.println(GPS.altitude);
      Serial.print("Satellites: "); Serial.println((int)GPS.satellites);*/


    if (BTserial.available()) // aqui pongo el hc05
    {
        d = BTserial.read();
        Serial.print("Recibo ");
        Serial.println(d);
        if (d == 'P') {
          Serial.println("Comienzo de cadena de posicion");
          while (BTserial.available() < 1);
          Serial.println("Recibo latitud...");
          BTserial.readBytes((char *)&latitud, 4);
          Serial.print("Latitud ");
          Serial.println(latitud);
          while (BTserial.available() < 1);
          BTserial.readBytes((char *)&longitud, 4) ;
          Serial.print("Longitud ");
          Serial.println(longitud);
        }
    }
{
          latitud_envio = latitud * 3600 *30,88 ;

          Serial.println ("la latitud hc05 en metros es.." );
          Serial.println ( latitud_envio, DEC );

          delay (5000);

          latitud_base = latitud1 * 3600 * 30,88 ;

          Serial.println ("la latitud del hc06 en metros es.." );
          Serial.println ( latitud_base, DEC );

          delay (5000);

          smlat= latitud_envio - latitud_base;
          Serial.println ("la diferencia de latitudes es..en metros" );
          Serial.println ( smlat, DEC );

          delay(5000);

          smlat_corregida= smlat  -1;
          Serial.println ("la diferencia de latitudes es..en metros" );
          Serial.println ( smlat_corregida, DEC );
          smlat_2=pow(smlat_corregida,2);



         delay (5000);
```

```
          long_envio= longitud * 3600 * 30,88;
          Serial.println ("la longitud hc05 en metros es.." );
          Serial.println ( long_envio, DEC );

          delay(5000);

          long_base=longitud1*3600*30,88;
          Serial.println ("la longitud hc06 en metros es.." );
          Serial.println ( long_base, DEC );

           delay(5000);


          sumlon= long_envio- long_base;
          Serial.println ("la diferencia de longitudes es..en metros" );
          Serial.println (sumlon , DEC );

          delay(5000);


          sumlon_corregida= sumlon - 0;
          Serial.println ("la diferencia de longitudes corregidas es..en metros"
);
          Serial.println (sumlon_corregida , DEC );
          sumlon_2= pow (sumlon_corregida, 2);

          delay(5000);



          operacion = smlat_2 + sumlon_2;


          delay(10000);

          distancia=sqrt( operacion);

          Serial.println ("bob se encuentra a");
          Serial.println(distancia);
          Serial.print ("metros del buque");

          delay(20000);
}
        if( distancia > 15 ){
          Serial.println ( "hombre al agua");
          digitalWrite(8, HIGH); //se enciende LED
        }
        else (distancia < 15);{
          digitalWrite(8, LOW); // se apaga LED
        }
  }
}
```