UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO E IMPLEMENTACIÓN DEL CONTROL DE LA NAVEGACIÓN AUTÓNOMA DE UN MODELO A ESCALA DE EMBARCACIÓN

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

María Francés Pérez

TUTORIZADO POR

José Vicente Busquets Mataix

Javier Busquets Mataix

CURSO ACADÉMICO: 2020/2021

RESUMEN

El objetivo principal de este proyecto es el desarrollo del control de una embarcación ya dada para que pueda cumplir dos funciones: alcance de un punto de coordenadas o el control manual de la misma embarcación de forma remota.

Se realizará el control mediante la plataforma Arduino, que contiene un microcontrolador y un entorno de desarrollo libre. Este proyecto parte de una maqueta de una embarcación que ha sido cedida para la realización del proyecto. Esta maqueta contiene parte del hardware que se requiere ya instalado.

Respecto al alcance de unas coordenadas, se utilizará un módulo GPS. La embarcación conocerá las coordenadas y tendrá que alcanzarlas calculando la trayectoria y realizando tareas de control. Este, tendrá que corregir posibles agentes externos como viento.

Para el control de la embarcación de forma remota se utilizará un módulo LoRa, ya que tiene más alcance. En este caso la embarcación no tiene que comportarse de forma autónoma.

Palabras clave: Autonomous, Navigation, Path, Control, GPS, Arduio, ASV.

ABSTRACT

The main objective of the project is the development of the control of a boat which must fulfil two functions: autonomous navigation to reach a gps coordinate or manual navigation by control remote of the same boat.

The control will be carried out through the Arduino platform, which contains a microcontroller and a free software developer. This project is based on a boat model which has been loaned for the realization of the project. This model contains part of the hardware that is required for the project.

Regarding the range of coordinates, a GPS module will be used. he ship will detect its position and the coordinates and will have to reach them, calculate the execution and perform control tasks. This will have to correct possible external agents such as wind.

For controlling the boat remotely, a LoRa module will be used in order to communicate with the user, since it has a longer range. In this case the boat will not have to work autonomously.

Key words: Autonomous, Navigation, Path, Control, GPS, Arduio, ASV.

DOCUMENTOS

1. MEMORIA
2. ESQUEMAS
3. PLIEGO CONDICIONES
4. PRESUPUESTO

DISEÑO E IMPLEMENTACIÓN DEL CONTROL DE LA NAVEGACIÓN AUTÓNOMA DE UN MODELO A ESCALA DE EMBARCACIÓN

1. MEMORIA

Autora: María Francés Pérez

Tutor: José Vicente Busquets Mataix

Tutor externo: Javier Busquets Mataix

Tabla de contenidos

Tabla de figuras

1. INTRODUCCIÓN

Embarcaciones no tripuladas

Las embarcaciones no tripuladas también llamadas vehículos de superficie no tripulados, en inglés USV [1], muchas veces llamadas drones de superficie o marítimos son aquellas que operan sin una tripulación a bordo. El desarrollo de estos dispositivos fue impulsado por la invención del radiocontrol.

En julio de 1898[2] Nikola Tesla creo una patente llamada "Método y Aparato para Controlar el Mecanismo de Movimiento de Buques o Vehículos" que describía un dispositivo llamado Teleautomaton [3] que funcionaba por ondas de radio. En septiembre del mismo año Tesla hizo la demostración de la patente en la primera Electrical Exhibition, que tuvo lugar en el Madison Square Garden. La demostración consistió en que una pequeña embarcación navegó por una piscina que fue instalada por Nikola Tesla dirigida desde un puesto de control.

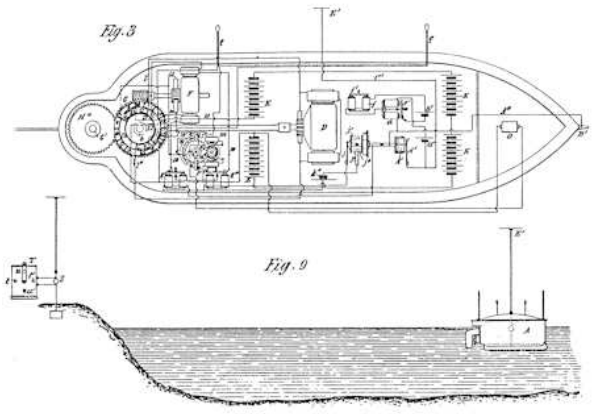


Figura 1: Nikola Tesla 1898 Esquema Teleautomaton [4]

Posteriormente, el radiocontrol, se fue incluyendo en nuevos dispositivos como drones o vehículos. Con el comienzo de la Guerra Mundial apareció, en el año 1917 [5], el primer dron teledirigido. Realmente eran bombas guiadas a control remoto. Posteriormente se incorporó el radiocontrol en aviones y vehículos, todo esto teniendo en cuenta que eran dispositivos a escala. También cabe destacar el Telekino que fue creado por un ingeniero español y fue el primer control remoto que se inventó.

Figura 2: Telekino

Los USV junto con estas dos tecnologías mencionadas han ido evolucionando hasta el punto de que los vehículos de superficie no tripulados se dividen en distintos grupos: los que operan de forma remota, de forma parcialmente autónoma o de forma completamente autónoma. Estos últimos son conocidos como los ASV y hoy en día se están desarrollando para muchas aplicaciones futuras. Un proyecto de gran repercusión fue cuando un ASV Saildrone [6] navegó de forma autónoma durante 7 meses recopilando información sobre el entorno.

Figura 3 ASV Saildrone [7]

En cuanto a los USV en general son muy utilizados en el campo de oceanografía porque debido a su autonomía y su coste sin la mejor opción para recoger datos. Se han desarrollado de tal forma que son capaces de obtener energía a través de las olas y la energía solar. Un proyecto que se ha desarrollado recientemente es el USV Vendaval [8]. Se trata de una pequeña embarcación creada para velar por la protección del mar Mediterráneo. Esta embarcación combina el funcionamiento con radiocontrol con el autónomo y estos, pueden trabajar de forma independiente en distintos tiempos.

Figura 4: Navantia, USV Vendaval 2019 [9]

2. OBJETIVOS

Los objetivos de este proyecto son desarrollar el hardware y software de una embarcación a escala. Esta embarcación a escala deberá responder ante dos modos de funcionamiento. El primero de ellos consiste en que la embarcación funcione de manera teledirigida por un usuario. El otro modo de funcionamiento será el seguimiento de una trayectoria marcada por unas coordenadas gps. En este caso, la embarcación deberá de corregir los pequeños errores que vayan produciéndose.

Para el desarrollo de este proyecto se va a utilizar un control con Arduino junto con otros módulos como LoRa, bluetooth o GPS. Para este proyecto se ha proporcionado una embarcación a escala. En el resto de la memoria la palabra embarcación se referirá a esta embarcación a escala.

Figura 5: Maqueta embarcación cedida

3. MATERIALES Y SOFTWARE

3.1. Maqueta embarcación

Para el desarrollo del proyecto se va a utilizar una embarcación cuyo modelo es NQD Storm Engine PX-16. En esta se implementará todo el desarrollo. Ha sido cedida por el profesor Jose Vicente Busquets Mataix para realizar el proyecto.

Figura 6: Maqueta embarcación

Esta embarcación consta internamente de dos motores eléctricos para el control de su velocidad junto con dos ESC (controladores de velocidad electrónicos). En cuanto a los motores son del modelo 390, es importante saber que pueden llegar hasta 20 km/h.

En cuanto al ESC o variador, el modelo es el 320A Brushed. Este se va a utilizar ya que es capaz de controlar la velocidad de un motor e incluso su sentido de giro. Se han de tener en cuenta una serie de características:

Alimentación	6 – 7.2 V
Salida BEC	5.6 V
	2A
Corriente de salida	300A

La embarcación también contiene unas hélices que están unidas al motor a través de una varilla. Estas hélices sobresalen por la parte posterior de la embarcación y son las encargadas de ejecutar el movimiento y la dirección. Al ir unidas al motor girarán a la vez que este.

Figura 7: Motor y varilla

3.2. Microcontroladores

Los microcontroladores que se van a utilizar van a ser de la familia Arduino. Específicamente se va a utilizar el modelo Arduino UNO junto con el Arduino Mega. La placa Arduino Uno está basado en un controlador ATmega328P, tiene 14 pines de entrada/salidas digitales, de estos pines 6 son PWM, y 6 entradas analógicas.

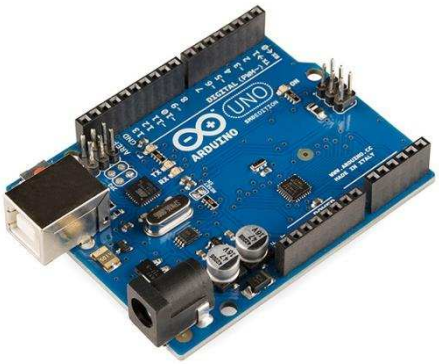


Figura 8: Arduino UNO

La placa Arduino Mega es una ampliación del Arduino Uno. Contiene 54 pines de entrada/salida digitales, de estos 14 pueden ser PWM y 16 entradas analógicas. Para este proyecto era interesante el mega ya que contiene 4 UARTs, puertos serial, y para comunicarse con los distintos dispositivos era conveniente esta ampliación.

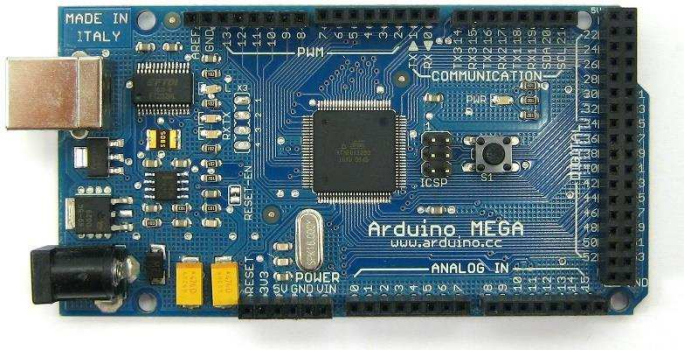


Figura 9: Arduino Mega

3.3. Dispositivo de conexión LoRa

Para poder establecer la conexión de la embarcación con el usuario se establecerá una conexión LoRa, debido a su gran capacidad de enviar paquetes en distancias medias. Se van a utilizar dos módulos TTGO LoRa 32 OLED V1. Debido a la normativa europea el modelo escogido es el que emite a 686 MHz de frecuencia.

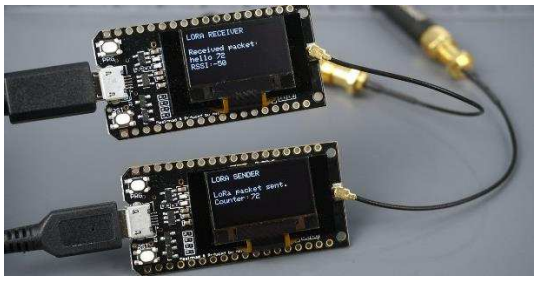


Figura 10: TTGO LoRa 32 OLED V1 [10]

Uno de los módulos trabajara como receptor y otro como emisor. Los dos módulos contienen una pantalla para poder visualizar en todo momento la información que se está compartiendo. También tiene una ESP32 que controlará como enviar la información y qué hacer con ella y una antena que deberá de estar conectada en todo momento, ya que en caso contrario se podría quemar el chip. Se ha de tener en cuenta de que el dispositivo se alimenta a 5V.

3.4. GPS

Se va a utilizar un módulo GPS para poder conocer y controlar en todo momento la posición de la embarcación. Esto hará que la embarcación pueda corregir en todo momento posibles errores de trayectoria. El dispositivo elegido es el Skylab SKM53 ya que es un dispositivo compatible con Arduino.

Figura 11: Skylab SKM53 [11]

El módulo tiene 6 pines de conexión de los cuales se van a utilizar 4, El de alimentación (Vcc), el de GND, el TXD y RXD. Estos dos últimos serán los encargados de comunicarse con el Arduino.

Figura 12: Pines conexión del Skylab SKM53 [12]

El pin GND se ha de conectar con el correspondiente pin GND del Arduino. Los pines Tx y Rx se conectarán con el puerto serial que se quiera. De la hoja de características se obtiene el valor del rango de la alimentación. En este caso puede variar entre los 3.3V y los 5V. Aunque sea el valor límite se podría conectar en el pin de 5V del propio Arduino.

Mechanical requirements	
Dimension	30mm x20mm x 8.5mm
Weight	9g
Power consumption	
VCC	3.3V~5V
Current	50mA(typical)

Figura 13: Hoja de características Skylab SKM53 [13]

Este dispositivo utiliza el protocolo NMEA nace de la Asociación Nacional de Fabricantes de Electrónica, en inglés las siglas son NMEA. Se desarrolló para poder estandarizar la comunicación entre varios dispositivos. Este protocolo define la velocidad de trasmisión que es en torno a 4800 baudios y el formato de los datos trasmitidos. El primer protocolo se llamó NMEA 0183. Actualmente hay uno actualizado que se denomina NMEA 2000

3.5. Dispositivo de conexión bluetooth

Dado que la embarcación se controlará desde un teléfono móvil se va a utilizar un módulo bluetooth para que recoja la señal y se comunique con el módulo LoRa a través de un microcontrolador. El dispositivo que se va a utilizar va a ser un HC-06, que es compatible con Arduino.

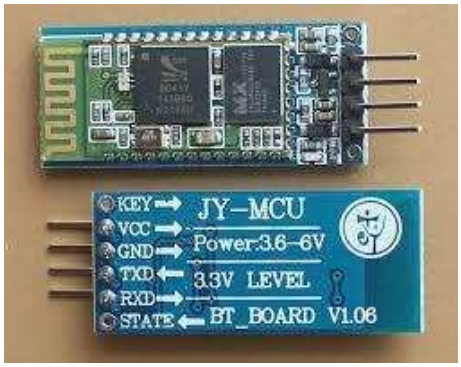


Figura 14: HC-06

Este dispositivo contiene 4 pines, dos se encargan la de la alimentación y los otros dos de enviar la información al dispositivo bluetooth. Como se puede observar en la parte posterior de la propia placa, o en la hoja de características, el rango de voltaje en la alimentación debe oscilar entre los 3.6 V y los 6V.

3.6. Unidad de medición inercial (IMU)

Para obtener la orientación de la embarcación se va a implementar una brújula digital con Arduino. Para esto se va a utilizar el módulo GY-273. Este módulo contiene el chip QMC5883L que es un magnetómetro de 3 ejes.

Figura 15: GY-273 IMU [14]

El dispositivo se alimenta a un voltaje de entre 2.16 V a 3.6V y su sensibilidad es de 230-1370 LSB / Gs. El módulo contiene 5 pines de los cuales se van a utilizar los dos destinados a la alimentación y los pines SCL y SDA para la trasmisión de la información. Con el uso de la librería "MechaQMC5883.h "se obtendrán las posiciones.

3.7. Batería

Para la alimentación de los motores se van a utilizar varias baterías, en este caso el modelo será Ni-cd Sc 18000 mah, que ofrece un voltaje de 7.2 V.

Figura 16: Batería 7.2 V [15]

Para la alimentación del Arduino Uno se va a utilizar una batería de litio de 3.7V y 48000 MAh. Dado que la pila es de 3.7V y el Arduino se alimenta a 5V se va a utilizar conversor de tensión de la cual se podrá obtener una tensión de 5V o de 3V.

Figura 18: Puerto de batería

Figura 17: Batería 3.7 V

3.8. Arduino IDE

Para el desarrollo del código se va a utilizar la aplicación de Arduino que permite incluir librerías y distintas tarjetas.

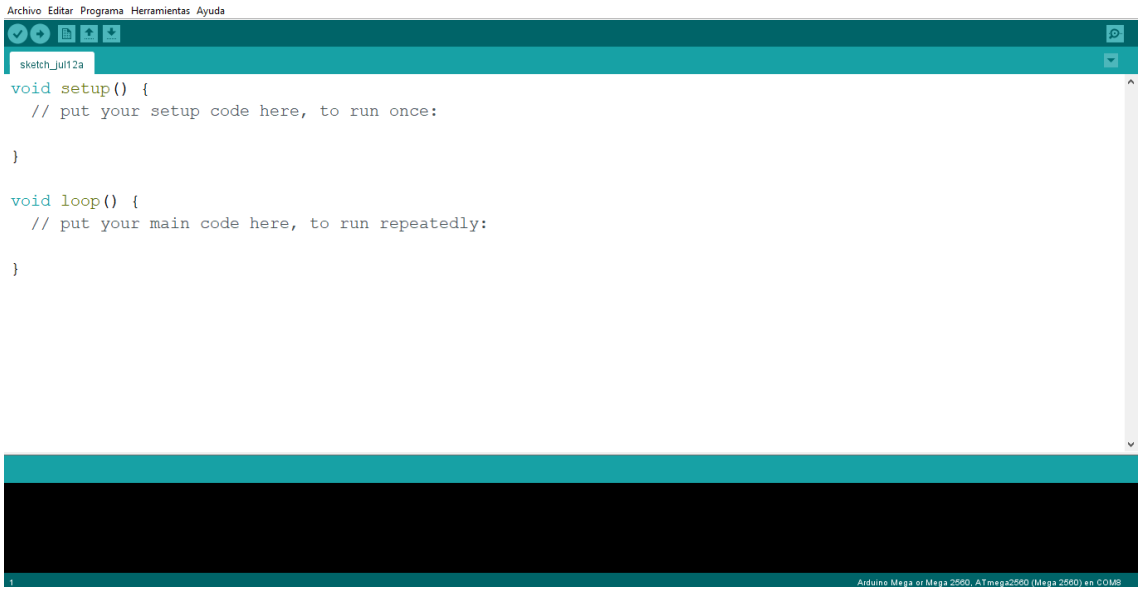


Figura 19: Software Arduino

3.9. Eagle

Para el desarrollo de los diagramas de conexión se va a utilizar el software Eagle. El programa tiene un panel de control inicial desde el cuál se pueden crear nuevos proyectos en los que se pueden incluir estos esquemas.

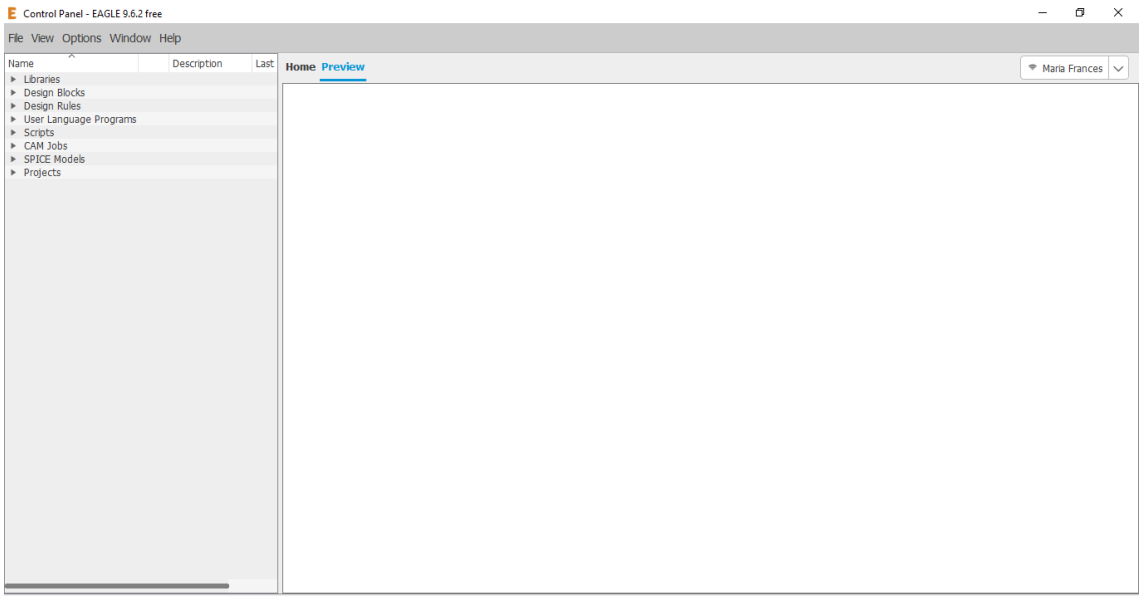


Figura 20: Software Eagle

La ventaja de utilizar este programa frente a otros es que el usuario se puede desarrollar sus propias librerías o incluir librerías desarrolladas por otros usuarios. En internet se pueden encontrar una gran variedad de librerías en las cuales se incluyen componentes de diferentes clases. Por lo que representar los esquemas de conexiones del proyecto es más sencillo.

4. Análisis y diseño

Para resolver este objetivo se ha de diseñar un controlador en bucle cerrado que controle la orientación. Para ello, se utilizará un PID [16]. Se partirá de unas coordenadas objetivo a las que se pretende que se dirija la embarcación. A partir de estas coordenadas objetivo y de las coordenadas de la embarcación del momento, que se medirán a través de un dispositivo GPS, se obtendrá la orientación o rumbo al que debe dirigirse la embarcación para alcanzarlas.

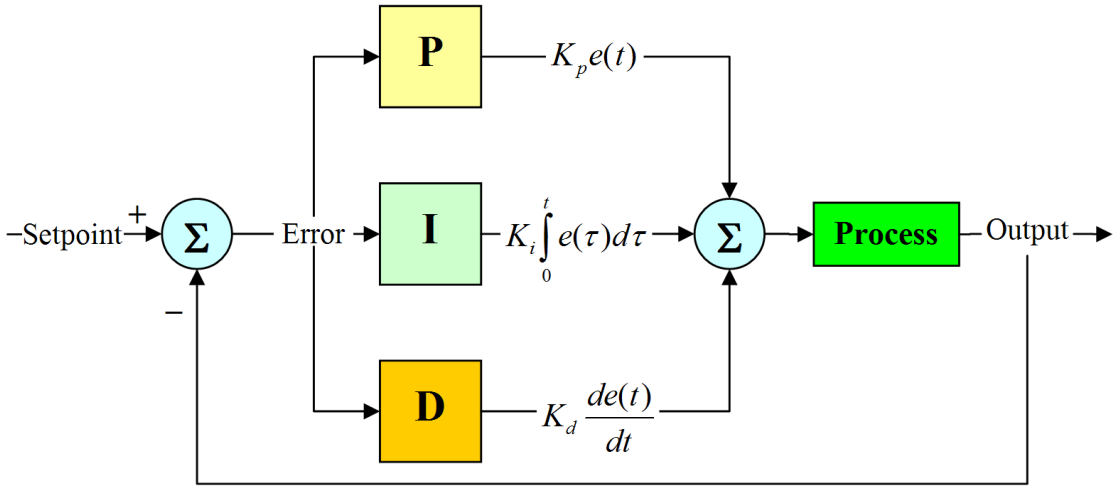


Figura 21: PID

El control en bucle cerrado será el encargado de mantener la embarcación en el rumbo correcto controlando la potencia de los motores de la embarcación. Para saber el rumbo que lleva la embarcación en todo momento se utilizará un módulo magnetómetro de 3 ejes que se utilizará como brújula electrónica. Con el dato del rumbo actual y el inicial se desarrollará el PID.

Tras el cálculo del PID y la actuación sobre los motores se medirán las coordenadas de la embarcación y se compararán con las coordenadas objetivo. Si estos dos valores fueran iguales o próximos la embarcación se parará, si no, continuará con el funcionamiento PID. El diagrama de como actuará la lógica del controlador es el siguiente:

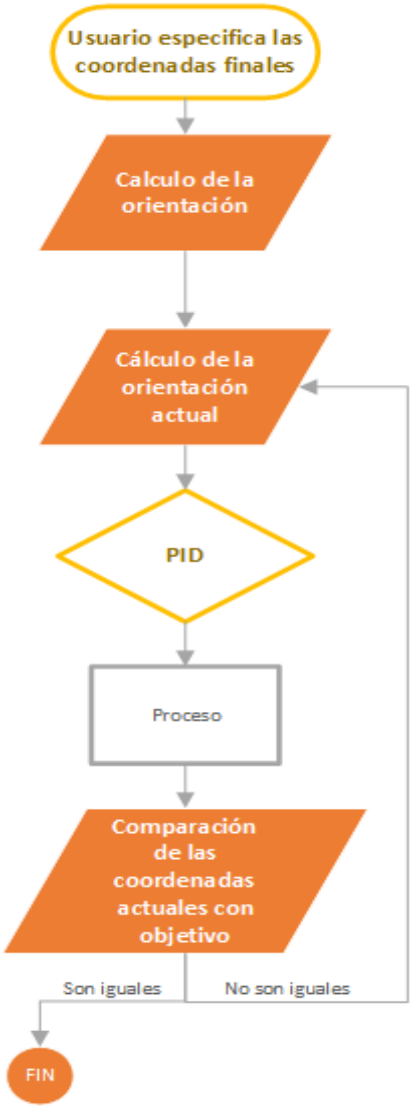


Figura 22: Diagrama proceso PID

La parte de los actuadores va a funcionar de forma que inicialmente a la embarcación se le van a enviar a los dos motores los valores para que inicie la navegación hacia delante. El motor de la izquierda a partir de este punto no cambiará su valor, a no ser que se quiera detener la embarcación, y todo el control se hará sobre el motor de la derecha. El PID se programará de forma que devuelva un valor que luego se sumará o restará a la velocidad que ya tenía el motor de la derecha, por lo que irá corrigiendo la trayectoria.

5. Desarrollo de los módulos

5.1. Módulo comunicaciones

Para establecer la comunicación del usuario con la embarcación se va a utilizar una conexión bluetooth y una LoRa. La conexión bluetooth servirá para captar la información que le envía el usuario a la embarcación desde un dispositivo móvil y la LoRa para enviársela a la embarcación. Se ha elegido este método de comunicación en vez de únicamente la bluetooth para que se abarque más distancia, ya que utilizando solo la conexión bluetooth la distancia es muy reducida y, por tanto, no se considera suficiente.

El módulo requiere de una aplicación para su control desde el móvil. La aplicación es la siguiente:

Figura 23: Aplicación consola

Esta aplicación tendrá el siguiente aspecto:

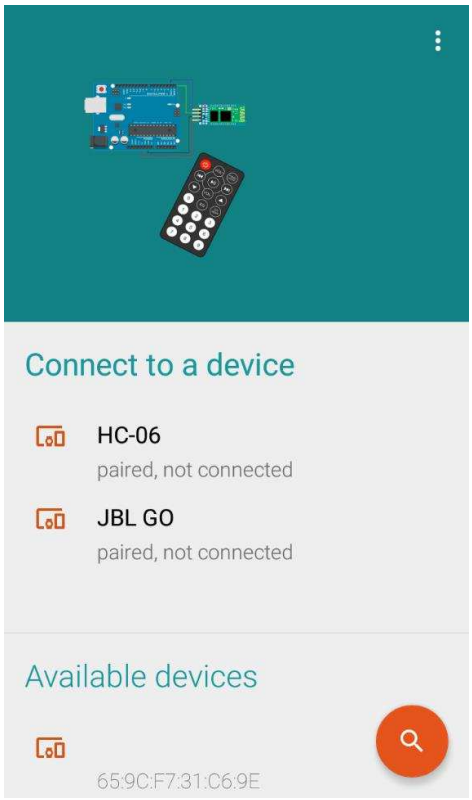


Figura 24: Pantalla inicial consola

En la pantalla principal se selecciona el dispositivo al que se debe de conectar. Une vez conectado aparece la siguiente pantalla donde se escoge el modo de funcionamiento:

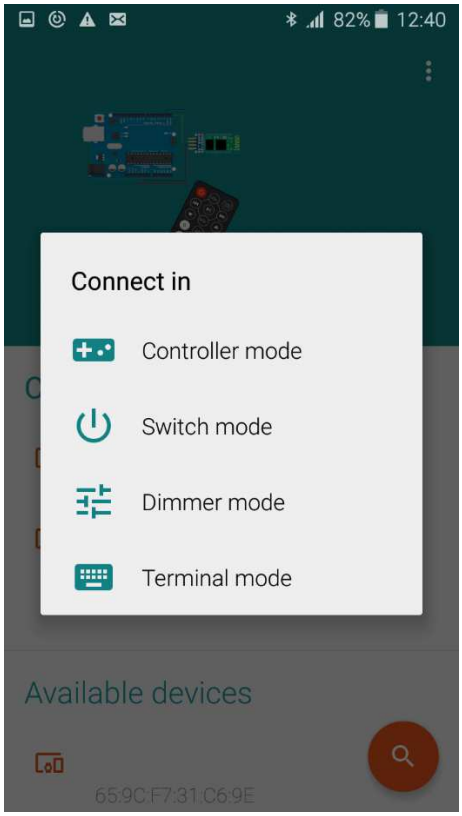


Figura 25: Aplicación consola

Para este caso se va a escoger el primero," controller mode", y aparecerá la siguiente consola:

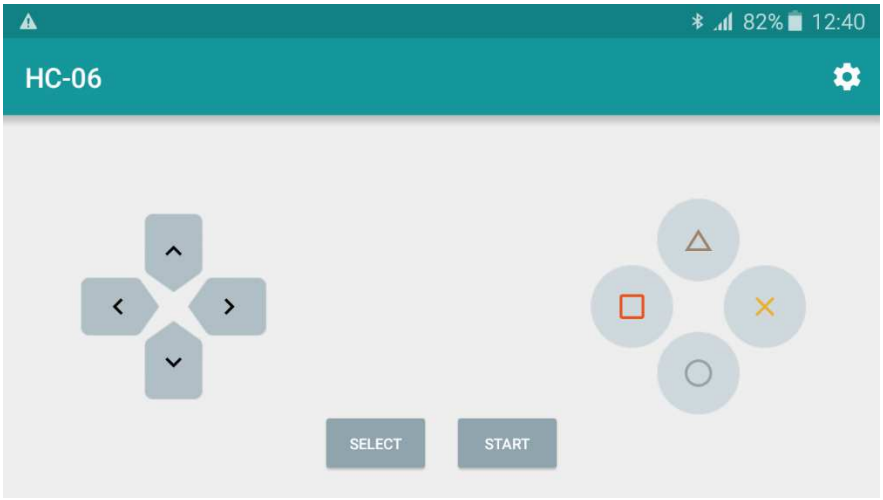


Figura 26: Aplicación consola

Las flechas se usarán para dirigir la embarcación y los botones inferiores para escoger la manera en la que funcionará la embarcación. Se asocia a los botones los siguientes números:

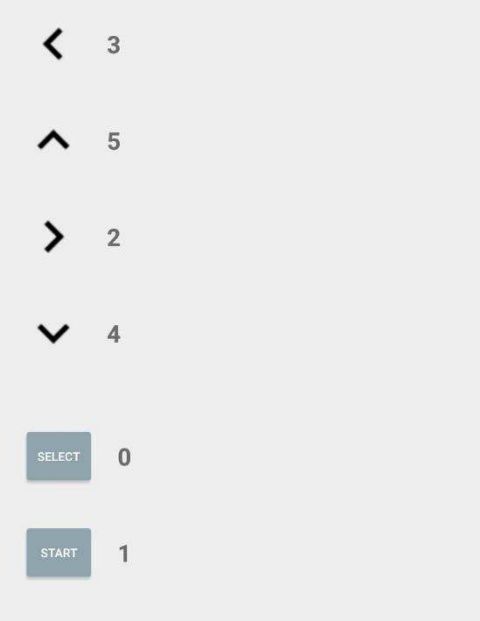


Figura 27: Parámetros aplicación consola

Estos valores se tendrán en cuenta posteriormente en el código.

5.1.1. Módulo bluetooth

Una vez configurada la aplicación se realiza una prueba de funcionamiento del módulo bluetooth, para esto se va a utilizar un dispositivo HC-06 y un Arduino UNO. El dispositivo HC-06 se conectará de la siguiente forma:

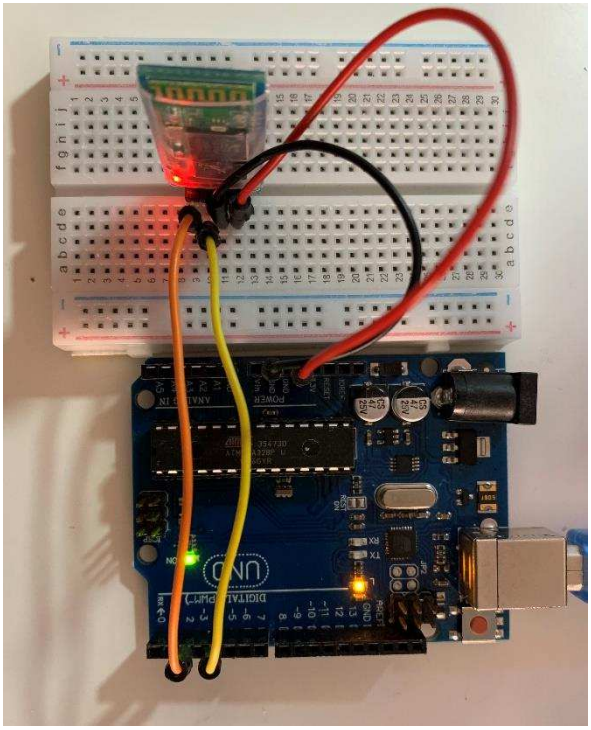


Figura 28: Montaje bluetooth

Una vez que está conectado, el LED que contiene parpadeará hasta que un dispositivo se conecte. Cuando esto suceda el LED de la placa dejará de parpadear y se quedará encendido permanentemente. Esto significará que ya está listo para recibir información.

Para la prueba se utilizará la consola explicada anteriormente y se le enviarán distintos valores pulsando las diferentes flechas. Para ver que funciona correctamente se muestran estos valores por el monitor serie. La prueba se realiza sin ninguna complicación, se obtiene el siguiente monitor serie:

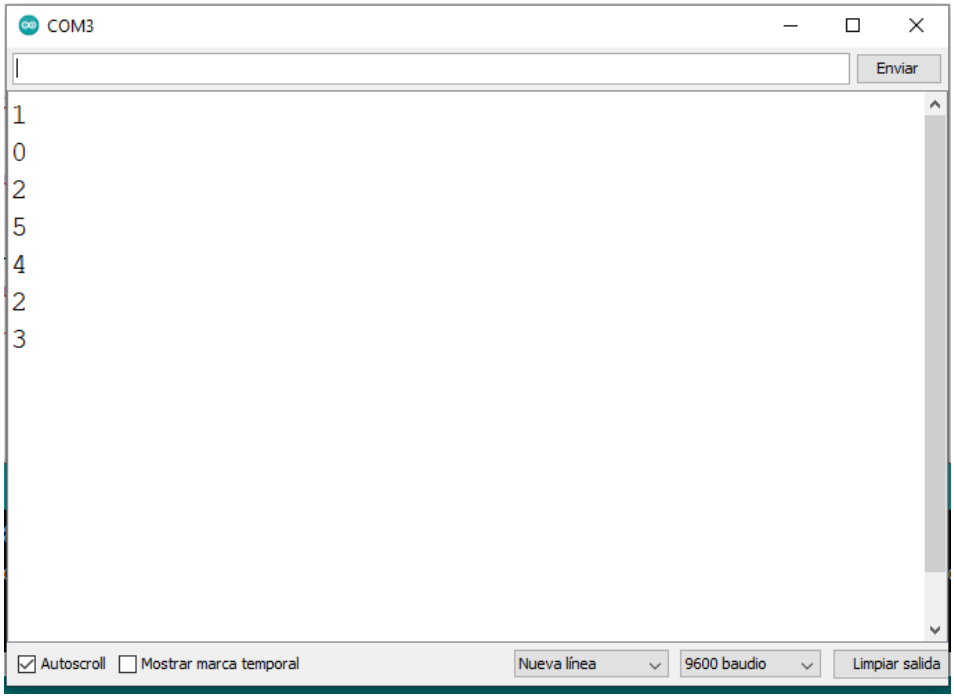


Figura 29: Valores obtenidos por el monitor serie

5.1.2. Módulo LoRa

Para la prueba del módulo LoRa se van a utilizar los dos dispositivos de conexión que son los módulos TTGO LoRa32 oled v1. Uno de ellos se usará como receptor y otro como emisor de la información. Al disponer de una pantalla LED se podrá tener certeza de lo enviado en cada momento. Para el desarrollo de este módulo se desarrolla un ejemplo [17].

Se configura el sender (emisor) para que envíe un valor deseado, en este caso se enviará el número cuatro. Para utilizar estos módulos se utilizan varias librerías tanto en el dispositivo emisor como en el receptor. Para la conexión LoRa se utilizan la SPI.h y LoRa.h. La primera librería es la encargada de la configuración de los pines usados por la trasmisión LoRa y la segunda de la trasmisión del paquete que se pretenda enviar de un LoRa a otro. Los pines que se utilizan para la conexión LoRa se definen como constantes y son los siguientes:

Nombre	Pin
SCK	5
MISO	19
MOSI	27
SS	18
RST	14
DIO0	26

Para la configuración y uso de la pantalla LED se utilizan las librerías: Wire.h, Adafruit_GFX.h y Adafruit_SSD1306.h. La librería Wire.h es utilizada para que el Arduino se conecte con dispositivos con el protocolo I2C/TWI. Este se basa en dos trasmisiones; la SDA que es la

encargada de los datos y la SCL que es la del reloj serie. Las otras dos librerías son las encargadas del estilo de la pantalla OLED. Para la utilización de la pantalla LED se van a utilizar distintos pines que se declararán como constantes:

Nombre	Pin
OLED_SDA	4
OLED_SCL	15
OLED_RST	16

Como se puede comprobar en la imagen en la pantalla del sender se comprueba que el paquete enviado es el número 4, que corresponde con el recibido.

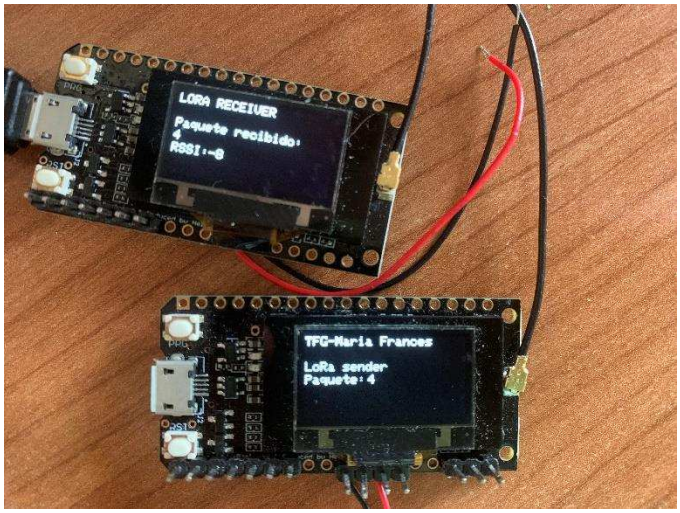


Figura 30: TTGO LoRa 32 OLED v1

5.1.3. Módulo bluetooth +módulo LoRa

Se van a implementar los dos módulos de comunicaciones para que trabajen de forma conjunta. Para esto se va a utilizar un Arduino UNO para que haga de intermediario entre los dos.

Como se va a utilizar un dispositivo Arduino se va a tener que usar la comunicación UART disponible en la placa. Para esto en el Arduino se van a declarar dos puertos serie auxiliares, en este caso no se va a utilizar el puerto serie 0 de la placa (TX0 y RD) por si se hacen pruebas con el monitor serie que no interfiera. El puerto que se comunicará con el HC-06 estará en los pines 4 y 2 (RX y TX). Y el puerto que se comunicará con el dispositivo LoRa estará en los pines 8 y 9 (RX y TX). Para la configuración de estos puertos extra se necesita la librería SoftwareSerial.h.

Puerto bluetooth	RX	4
	TX	2
Puerto LoRa	RX	8
	TX	9

Para que la ESP32 reciba la información del Arduino también se ha de configurar un puerto adicional en la TTGO LoRa32 OLED. En este caso se ha de tener en cuenta que muchos de los pines están siendo utilizados o por la pantalla LED o por la comunicación LoRa, por esto se mira el diagrama de conexiones interno de este dispositivo:

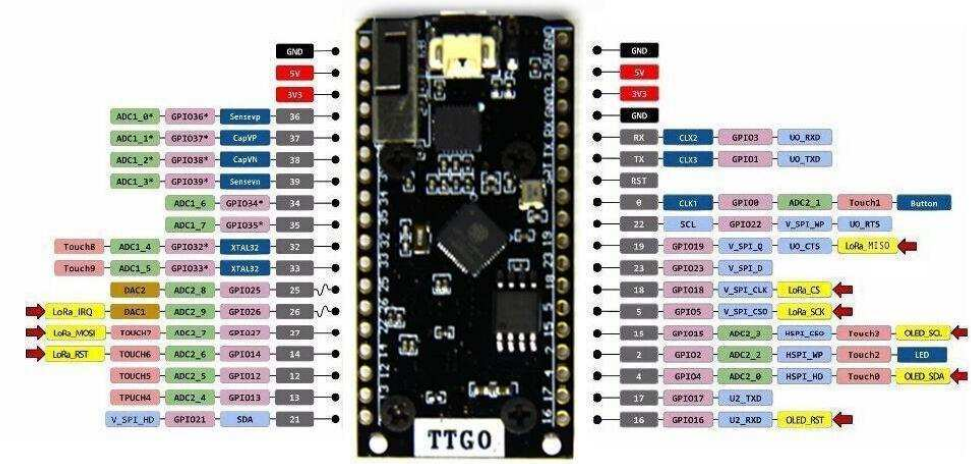


Figura 31: Conexiones internas [17]

En el diagrama se pueden ver que los pines 17 y 23 no son utilizadas por ninguna de las configuraciones realizadas anteriormente, por lo tanto, se escogen estos. El pin 17 corresponderá al TX y el 23 al RX. Si se utilizasen otros pines la pantalla se bloquearía o la conexión LoRa no se establecería ya que internamente la mayoría del resto de pines están comunicados con la pantalla LED o se utilizan para establecer la conexión.

El código que se utiliza en el dispositivo LoRa que se utiliza como emisor está desarrollado en el Anexo I. Hay que tener en cuenta que el Arduino enviará el dato como un entero y el Lora lo envía como una cadena de caracteres, por lo que habrá que hacer esta conversión. En cuanto al dispositivo LoRa que recibe la información también ha de tener en cuenta este cambio de formato del texto recibido como cadena de caracteres a número entero. El código es similar al del emisor, se desarrolla en el Anexo II.

La ejecución consiste en que cuando haya información en el puerto bluetooth esta será leída por el Arduino la y enviada al puerto donde está conectada la ESP32. Las conexiones quedan de la siguiente forma:

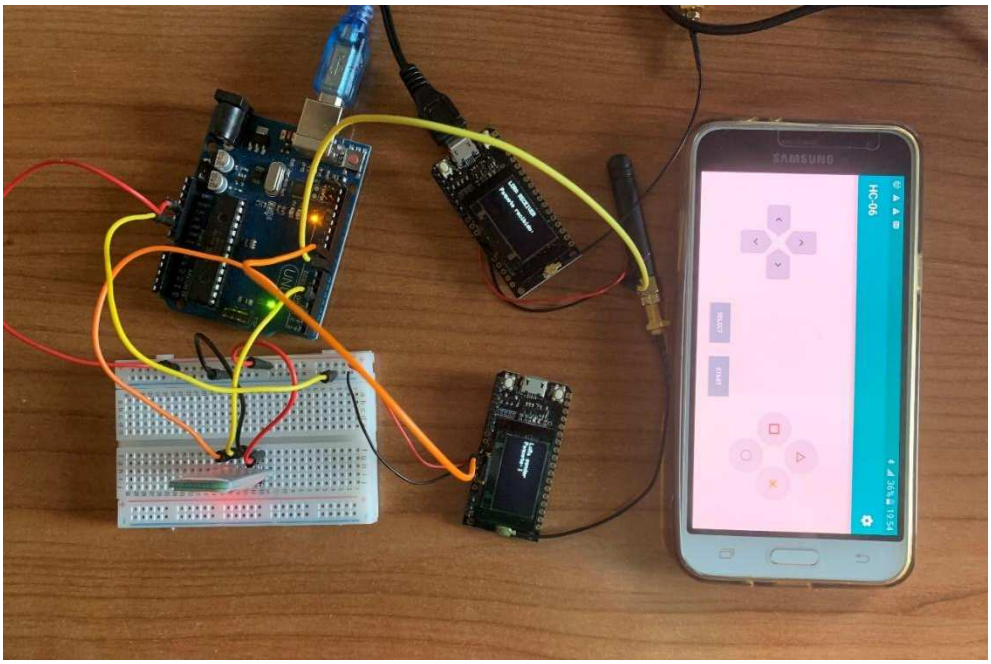


Figura 32: Conexiones módulo bluetooth + LoRa

Al realizar la prueba se comprueba que el funcionamiento es el esperado, todo ello gracias a que se puede controlar gracias a la pantalla LED, el valor que se envía en cualquier momento y así comprobar si coincide con el enviado por el móvil gracias a la pantalla LED.

5.2. Módulo control de velocidad

Este módulo estará formado por todo lo necesario para el uso de los motores. Entre los elementos que lo integran estarán el ESC 320 Brushed, los motores 390, una batería para proporcionar una alimentación en torno a 7 V y un Arduino para controlar la velocidad.

Para controlar el ESC con el Arduino se va a utilizar la librería Servo [18]. Dado que la embarcación podrá moverse hacia delante o hacia atrás o a distintas velocidades el servo transmitirá valores en un rango. El valor máximo que podrá enviar el servo será de 180 º y el mínimo de 0º. Esto corresponderá a la máxima velocidad en sentido positivo y negativo del motor respectivamente. Esto es controlado por el ancho de pulso.

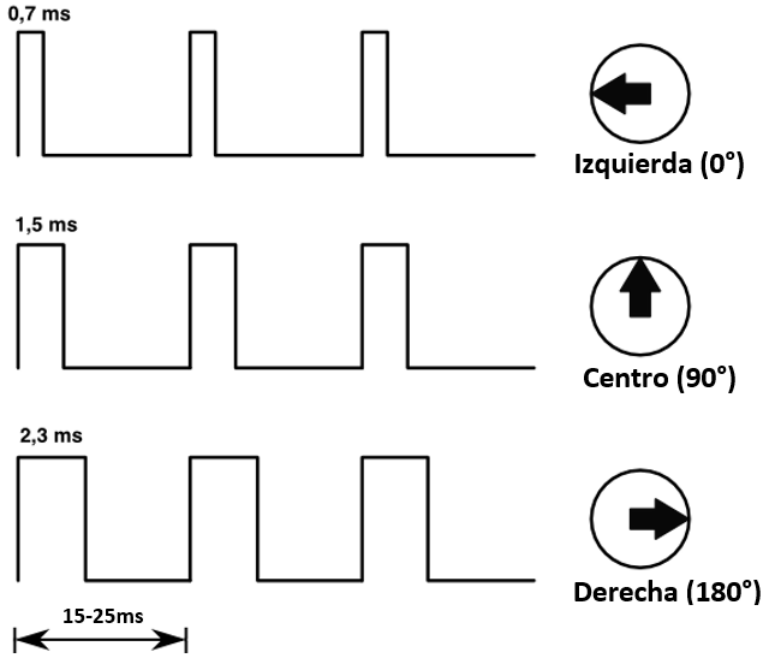


Figura 33: Funcionamiento del motor

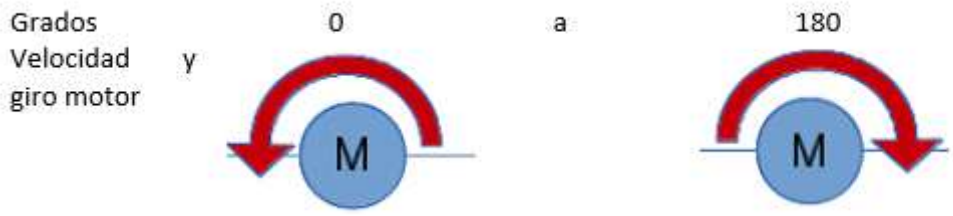


Figura 34: Funcionamiento del motor con Servo.h

En los extremos el motor alcanza el máximo de velocidad y va disminuyendo hasta que en el centro se detiene (valor aproximado de 90 º) y al ir sobrepasando este rango cambia de sentido el movimiento del motor y va aumentando su velocidad hasta alcanzar el máximo otra vez.

Para la prueba se le van a enviar diversos valores dentro del rango para comprobar el correcto funcionamiento en todas las direcciones. El diagrama de conexiones será el siguiente [19]:

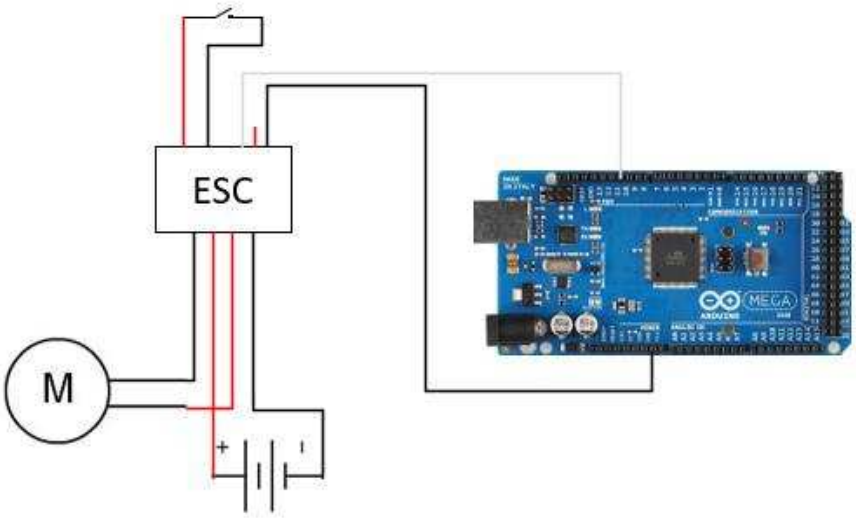


Figura 35: Esquema de conexiones prueba motores

También hay que tener en cuenta la posición de las hélices ya que para que funcione de forma correcta las hélices deben girar en sentido contrario, y para ello la pala de la hélice debe estar montada en sentido contrario de la otra. El montaje real es el siguiente:

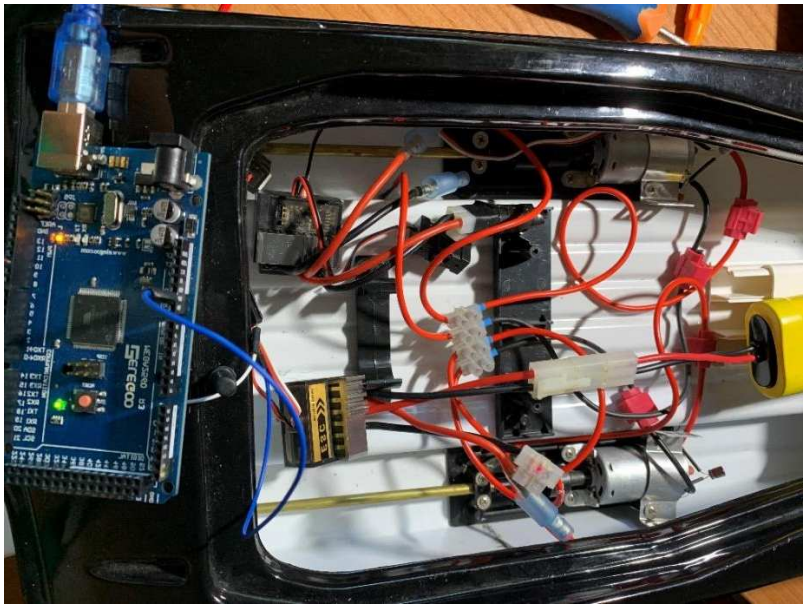


Figura 36: Montaje real prueba motores

Para la prueba se le va a crear un bucle que transmita a la embarcación valores entre el 20 al 150, de forma que los mismo se vayan incrementando y posteriormente decrementando. De esta prueba se sacan varias conclusiones. La primera de ellas es que las baterías no pueden oscilar significativamente de los 7 V. Es decir, el voltaje no puede ser inferior a 6.5

aproximadamente, ya que en caso contrario el ESC junto con el motor no funcionarán de forma idónea.

Otra observación de la prueba es que los valores de cambio de sentido no son en torno a 90. En realidad, aproximadamente para que gire en sentido horario el valor debe de ser como mínimo en torno a 100 y para que gire en sentido antihorario el valor ha de ser de al menos de 50. Si no se alcanzan estos valores el motor se mantendrá parado.

5.3. Módulo GPS

Para la prueba del GPS se ha de tener en cuenta que se tiene que hacer en exteriores ya que, en otro caso, la señal no es captada. También, si el dispositivo ha estado más de una semana sin usar debe estar previamente encendido 15 minutos al menos, para recibir señal.

Para el funcionamiento de este módulo se necesita la librería TinyGPS.h [20] [21]. Esta librería es la encargada de obtener el valor y de descodificarlo, ya que en un principio el dispositivo devuelve un número largo que incluye distintos datos seguidos como las coordenadas de longitud y latitud, pero también la fecha en la que se miden los valores. Las conexiones serán:

Pin GPS	Pin Arduino
Vcc	5V
GND	GND
TX	RX2
RX	TX2

El montaje es el siguiente:

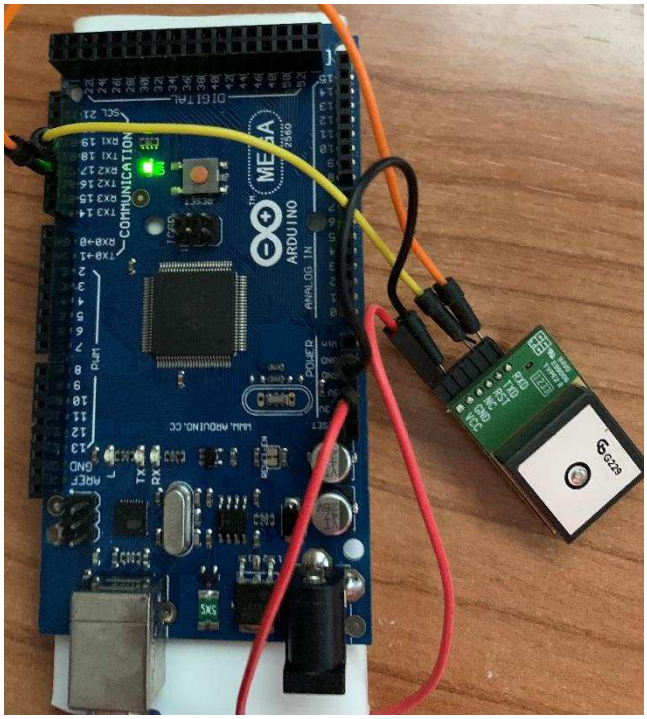


Figura 37: Montaje real GPS

Para comprobar el funcionamiento del GPS se ha de entender cómo funcionan las coordenadas geográficas para saber qué valores son los adecuados. Las coordenadas geográficas funcionan de la siguiente manera [22]:

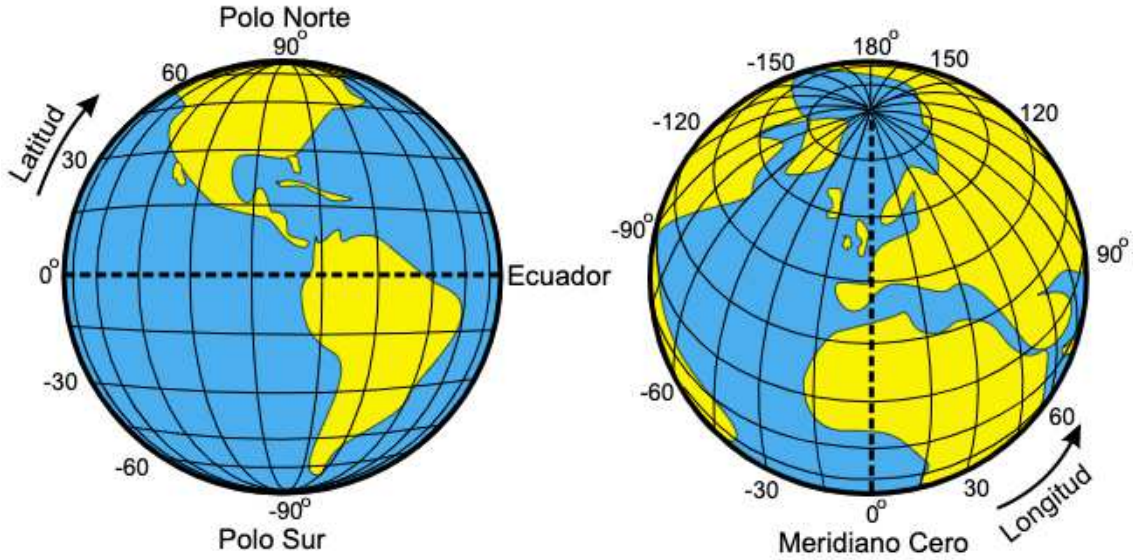


Figura 38: Coordenadas geográficas

Por lo tanto, al hacer la prueba en un pueblo de Alicante situado al oeste del Meridiano de Greenwich, el valor de la latitud debe de ser positivo ya que nos encontramos al norte del ecuador y el de la longitud negativo dado que por convención se consideran negativos los valores hacia el Oeste. También se comprueba que aumenta o disminuya correctamente al moverse el módulo gps a la derecha o la izquierda. Los datos de posición obtenidos se han mostrado por el monitor serie y son los siguientes:

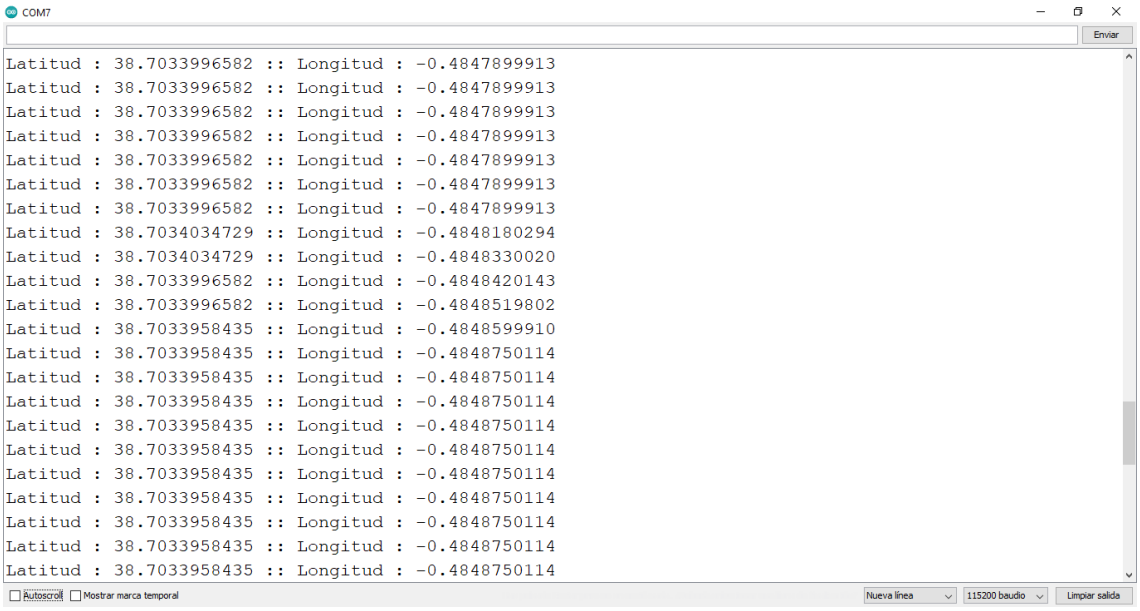


Figura 39: Datos obtenidos por el monitor serie

5.4. Módulo brújula electrónica

Para la prueba de este módulo se va a obtener el valor de los 3 ejes del sensor y a calcular la orientación. Para esto se van a utilizar las librerías Wire.h y MechaQMC5883.h. La primera librería sirve para la comunicación del módulo con el Arduino y la segunda para la trasmisión de la información. Se ha de tener en cuenta que los módulos GY-273 suelen tener de forma general un chip diferente del QMC5883L que es el que contiene el módulo obtenido. Se ha de tener en cuenta ya que todos los ejemplos y librerías se basan en el otro.

Para este ejemplo se va a utilizar el siguiente código de ejemplo [23], si bien incluyendo las modificaciones correspondientes para el chip de la placa [24]**.** También se debe de tener en cuenta que en la placa Arduino Mega existen los pines SCL y SDA [25] para la comunicación del bus I^2C, por lo tanto, la conexión de los pines es la siguiente:

Arduino Mega	GY-273
5V	Vcc
GND	GND
SCL	SCL
SDA	SDA

También se ha de tener en cuenta la diferencia del norte magnético al norte real por lo que en el código se adjunta el cálculo para que se muestre el norte real. Para realizar este cálculo se debe de obtener la diferencia del norte real [26] y el magnético según el año en curso y posición donde se vaya a desarrollar la prueba. Para obtener este dato se ha consultado en el Instituto Geográfico nacional. Una vez se prueba el código se obtiene la orientación respecto al norte por el monitor serie según se gira.

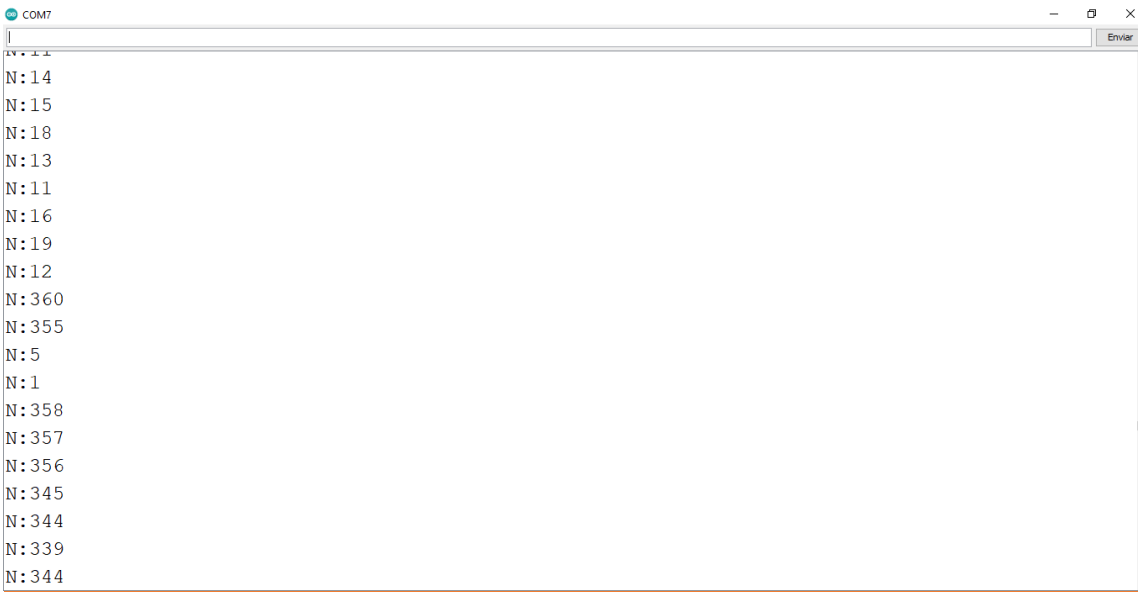


Figura 40: Monitor serie brújula

En la prueba se puede ver que la brújula es muy sensible a los cambios sobre todo cuando no está en una posición horizontal al suelo. Para comprobar si los datos obtenidos son los correctos se utiliza un dispositivo móvil que haga de aguja y se comprueban que los resultados coinciden.

6. Implementación e integración

Para el desarrollo de la implementación e integración se va a dividir su ejecución en dos módulos, que se corresponderán con los dos modos de funcionamiento: Modo radiocontrol y modo seguimiento de trayectoria. No obstante, hay que indicar que, aunque se trabaje de dos formas distintas las conexiones serán comunes para los dos bloques. En cuanto al esquema de conexiones se divide en la parte de control remoto y la de los actuadores.

La parte de control remoto constará del LoRa, bluetooth y el Arduino Uno y será la encargada de recibir la información del dispositivo del usuario y enviarla a la embarcación. El esquema de conexiones será el siguiente:

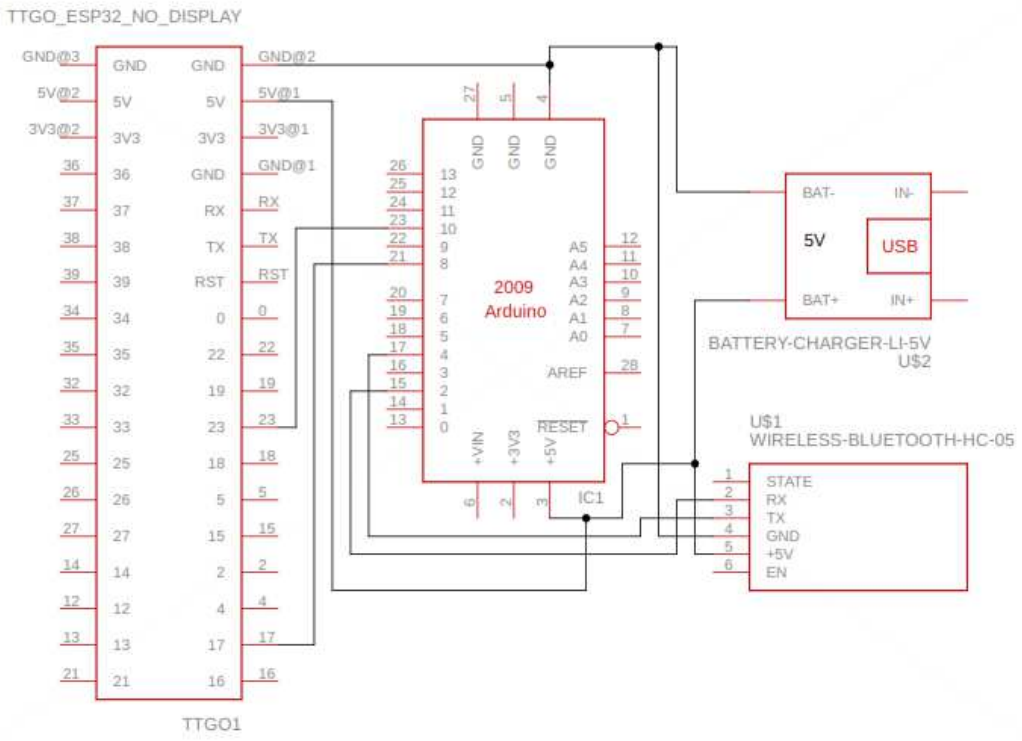


Figura 41: Esquema conexiones control remoto

La programación de esta parte también es común para los dos modos de funcionamiento. El flujo de funcionamiento será el siguiente:

Figura 42: Flujo control remoto

El código se encuentra desarrollado en el Anexo III.

En cuanto a la parte de los actuadores el esquema de conexiones contiene los motores, los ESC, y el Arduino mega. El esquema es el siguiente:

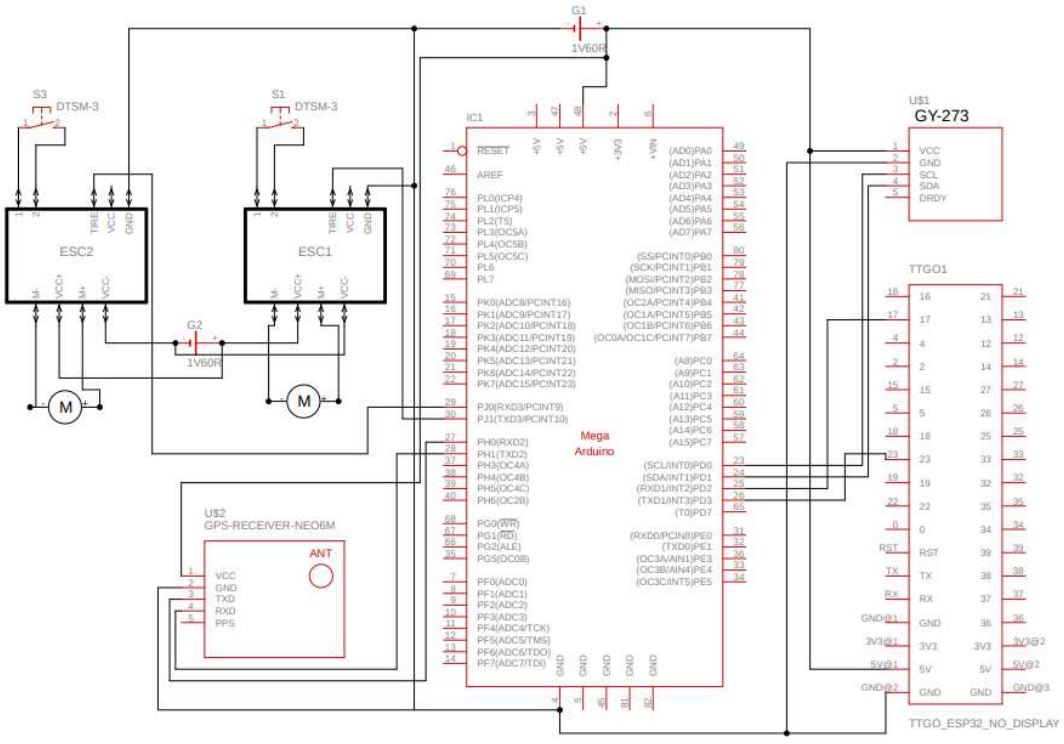


Figura 43: Esquema conexiones actuadores

El diagrama de flujo del código de la parte de los actuadores es el siguiente que, más adelante se verá con más detalle:

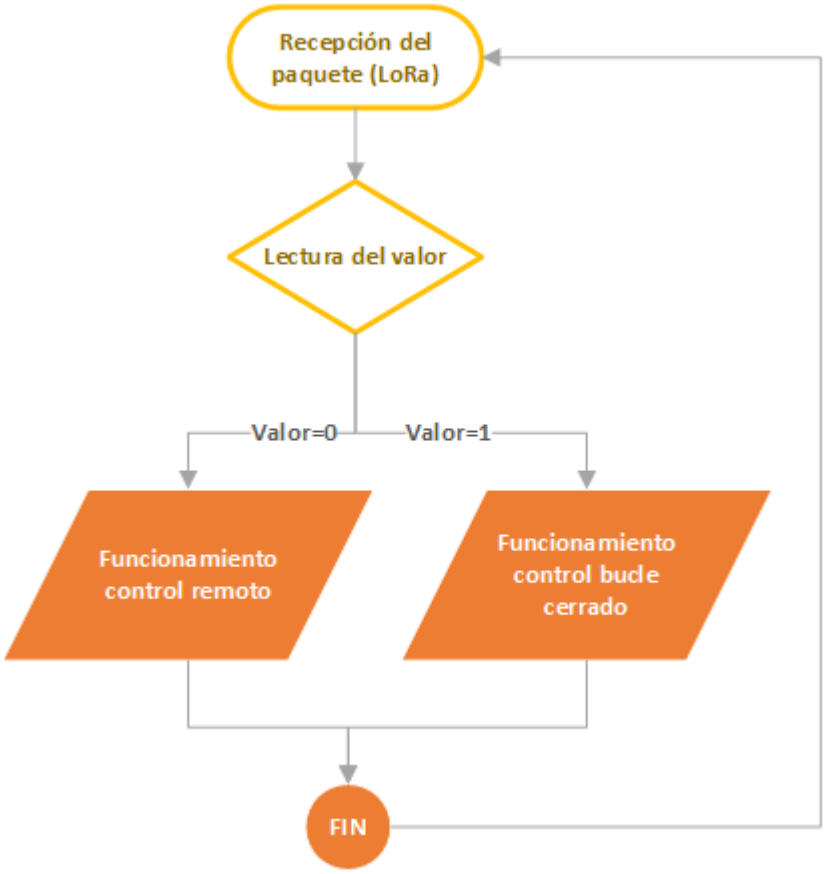


Figura 44: Flujo actuadores

Como se puede ver en el diagrama lo primero es comprobar si el usuario ha seleccionado algún modo, o en el caso de que ya estuviese funcionando la embarcación si ha cambiado de modo. Tras esto ya entra en el funcionamiento según el modo seleccionado. Una vez acabado el bucle vuelve a leer la variable recibida y comprobar el modo. El código completo está en el Anexo IV.

6.1. Modo radiocontrol

En este modo de funcionamiento la embarcación se moverá dependiendo de lo que el usuario ordene. Una vez el usuario haya seleccionado este modo junto con la aplicación de consola tendrá cuatro posibles opciones y dependiendo de lo escogido actuará.

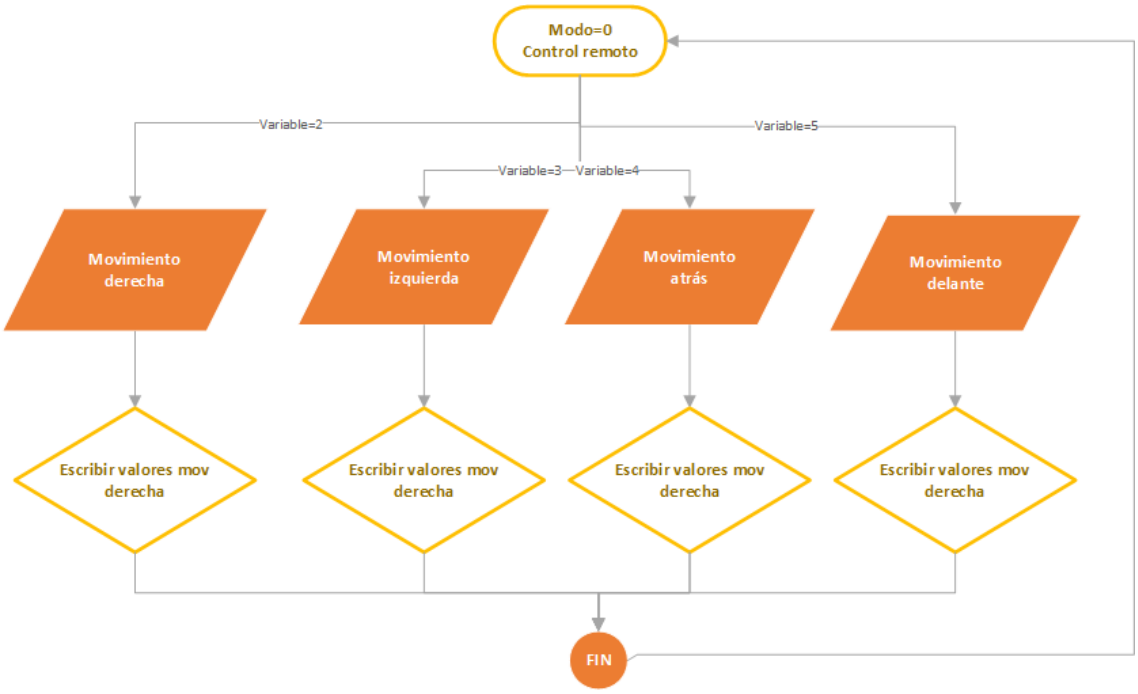


Figura 45: Flujo control remoto

Se ha de tener en cuenta que los TTGO LoRa mandan los datos en forma de cadena de caracteres (String) entre ellos y como se quiere obtener un numero en el código del TTGO que recibe se debe prever el cambio a número entero de dicho dato con el objetivo de que Arduino envíe un numero entero (int). También, se ha de comprobar que la velocidad de trasmisión entre todos los dispositivos sea la misma, para que no haya fallos. Para el desarrollo de esta prueba el diagrama de conexiones es el mostrado anteriormente.

Para que los motores funcionen según lo esperado se tendrá que mandar los valores que se quiere que sean aplicados, realizar una espera de un segundo, y mandarles un 90 para que los motores se paren.

Para comprobar que la embarcación funciona se van a realizar dos pruebas. La primera será que se conectará el Arduino por el puerto USB para que por el monitor serie muestre lo que está sucediendo en cada momento. Los resultados son los esperados ya que cambia de modo según se lo indique el usuario a través del dispositivo y cambia de dirección. Los resultados son los siguientes:

Figura 46: Resultados obtenidos modo radiocontrol

La otra prueba se realizará sobre el agua y se observará el comportamiento y la respuesta de la embarcación en cada momento y como responde a las peticiones del usuario.

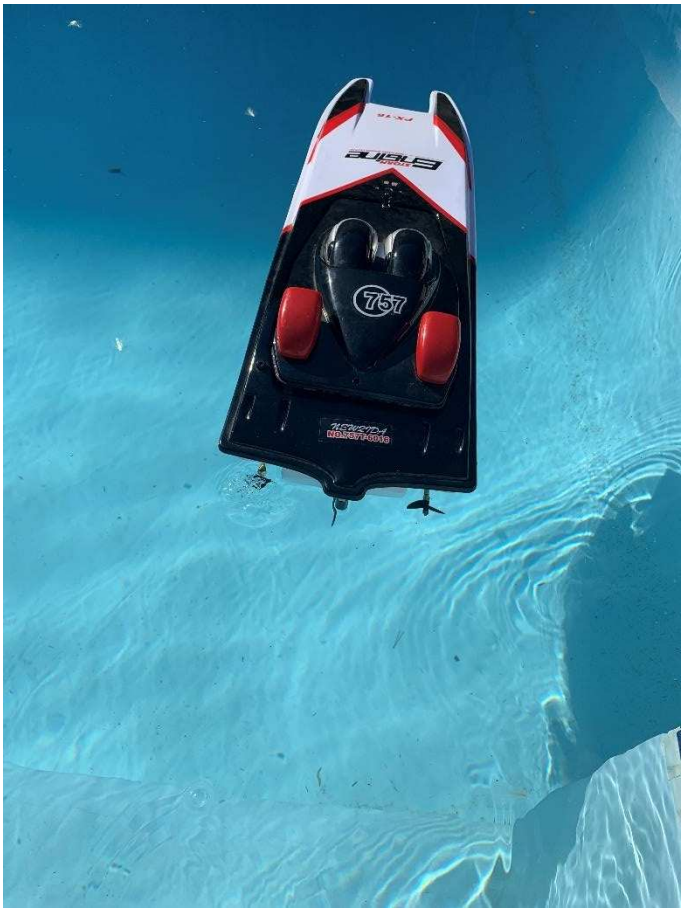


Figura 47: Embarcación en funcionamiento

6.2. Modo control bucle cerrado

Cuando trabaje en modo PID se desplazará automáticamente la embarcación. Se le tendrán que indicar las coordenadas a las que se quiere ir previamente. Una vez indicadas y estando el modo seleccionado se comenzará a acercar al objetivo. El diagrama del flujo que se realizará es el explicado en el Apartado 4 del documento. El desarrollo de este modo de funcionamiento se va a dividir en los siguientes pasos.

Primero. Cálculo del rumbo que debería de llevar la embarcación para alcanzar el objetivo. El rumbo es el ángulo de la orientación medido de 0 a 360 grados respecto a un eje, en este caso respecto al norte. Para esto primero se deberán convertir las coordenadas geográficas objetivo y las de la embarcación en el momento de la prueba en coordenadas UTM (Sistema de coordenadas universal transversal de Mercator). Estas coordenadas están basadas en la proyección cilíndrica de la tierra sobre el plano. Para esta conversión se va a utilizar un código ya desarrollado [27] con los cambios necesarios.

Una vez convertidos los dos puntos a coordenadas UTM se procede al cálculo del rumbo [28]. Para esto se desarrolla una función en Arduino que lo calcule. Esta calculará las diferencias entre los puntos X e Y anteriormente calculados y obtendrá el arco tangente. Dependiendo de si el cálculo de las diferencias es positivo o negativo se ajusta el valor para obtener la orientación desde el norte.

Segundo. Medición de la orientación con la brújula. Con el sensor GY-273, anteriormente programado en el apartado 5.4. se obtiene la orientación de la embarcación. Se utiliza el mismo trozo de código que en el desarrollo del módulo.

Tercero. Cálculo del PID. Para el control en bucle cerrado se va a utilizar una librería de Arduino llamada PIDController [29]. El PID tendrá como entrada el ángulo calculado en el segundo paso, que es el recogido por el sensor. Y tendrá como objetivo el calculado en el primer paso. Para la obtención de las constantes del propio PID se va a realizar una primera aproximación en la que se estimen las constantes. La estimación óptima de los coeficientes queda fuera de los objetivos principales del proyecto.

Cuarto. Envío de datos a los motores. Una vez el PID ha devuelto el valor de la salida este se debe de enviar al motor de la derecha. Se deberá comprobar a qué lado de la embarcación se encuentra el objetivo y dependiendo de ello se aplicará la corrección con signo positivo si está a la derecha y signo negativo si se encuentra a la izquierda.

Cabe destacar que durante el desarrollo del control automático el bucle incluye una comprobación de las coordenadas de la embarcación a través del GPS, para proceder a compararlas con las coordenadas objetivo y en el caso de que coincidiesen o fuesen muy próximas se procedería a trasmitir la orden de parada de los dos motores. Una vez finalizada esta implementación, se ha procedido a realizar la prueba.

7. Conclusiones

Se han cumplido los objetivos del proyecto que eran aplicar tanto el control PID de la embarcación como el control remoto por un usuario de forma que manejarla a distancia.

Uno de los problemas que se ha tenido a la hora de desarrollar el proyecto ha estado relacionado con los motores ya que estos eran muy sensibles al voltaje que les proporcionaba y también debido a que la batería que incorporaba la embarcación no estaba en buen estado y tenía problemas consistentes en que se descargaba fácilmente o se excedía del voltaje. También los ESC no funcionaban de igual manera, por lo tanto, la implementación completa ha sido difícil de probar. Para asegurarse de que funcionaba todo el desarrollo del código se ha utilizado el monitor serie como ayuda.

Una mejora para este proyecto sería contar con un modelo distintos de motores junto con su ESC para que respondan de manera más rápida y fiable. Porque con los actuales, aunque la respuesta en software sea rápida la física es más lenta. Otra mejora podría ser el desarrollo del envío de coordenadas desde el dispositivo móvil.

Concluyendo, se podría decir que pese a los aspectos negativos que se han observado desarrollando el proyecto, se ha logrado cumplir con los objetivos marcados habiendo completado el código necesario para el control de todos los dispositivos implicados y la comunicación entre los mismos.

8. Bibliografía

Introducción

[1] https://dronespain.pro/usv-barco-no-tripulado/

[2] https://teslauniverse.com/nikola-tesla/timeline/1898-tesla-demonstrates-worlds-first-wireless-remote-controlled-boat#goto-291

[3] https://vadebarcos.net/2017/02/18/teleautomaton-tesla-primer-dron-marino/

[4] http://www.tfcbooks.com/patents/0613809.htm

[5]] https://es.hobbyteam.net/blog/la-historia-del-radiocontrol/

[6] https://www.saildrone.com/news/global-carbon-ocean-observation-network

[7] https://www.plocan.eu/la-flota-de-plocan-recorre-18-000-millas-en-misiones-cientificas-y-de-observacion-en-2020/

[8] https://www.navantia.es/es/usv-vendaval/

[9] https://www.infodefensa.com/es/2019/12/13/noticia-navantia-integra-sistemas-primer-operaciones-espana.html

Materiales

[10] https://randomnerdtutorials.com/ttgo-lora32-sx1276-arduino-ide/

[11] http://www.skylab.com.cn/en/productview-82.html

[12] https://beetlecraft.blogspot.com/2015/10/tutorial-de-uso-skylab-skm53.html

[13] http://www.skylab.com.cn/uploadfile/Download/202107281554025203.pdf

[14] https://www.diarioelectronicohoy.com/blog/magnetometro-hmc5883l

[15] https://www.radiostyrda-odeller.se/default.asp?mod=product&cat_id=72,40,79&product_id=468

Análisis y diseño

[16] Libro: Control Automático. Tiempo Continuo y Tiempo Discreto (2016); Julian J. Salt Llobregat, Ángel Cuenca Lacruz, Vicente Casanova Calvo, Antonio Correcher Salvador

Desarrollo de los módulos

[17] https://randomnerdtutorials.com/ttgo-lora32-sx1276-arduino-ide/

[18] https://arduproject.es/motores-esc-y-su-programacion-en-arduino/

[19] http://techvalleyprojects.blogspot.com/2012/06/arduino-control-escmotor-tutorial.html

[20] http://arduiniana.org/libraries/tinygpsplus/

[21] https://www.engineersgarage.com/gps-module-with-arduino/

[22] https://beetlecraft.blogspot.com/2015/10/tutorial-de-uso-skylab-skm53.html

[23] https://www.luisllamas.es/brujula-magnetica-con-arduino-compass-digital-hmc5883/

[24] https://solectroshop.com/es/content/77-como-hacer-una-brujula-digital-con-arduino-y-un-compas-magnetometro

[25] http://robots-argentina.com.ar/didactica/descripcion-y-funcionamiento-del-bus-i2c/

[26] http://www.ign.es/web/ign/portal/gmt-declinacion-magnetica

[27] https://forum.arduino.cc/t/convertir-coordenadas-gps-a-utm/179485

[28] Libro: Patrón de embarcaciones de recreo, José de Simón Quintana

[29] https://www.luisllamas.es/como-implementar-un-controlador-pid-en-arduino/

9. Anexos

Tabla de Anexos

Anexo I: Código dispositivo LoRa emisor

```
//Libraries for LoRa

#include <stdlib.h>

#include <SPI.h>

#include <LoRa.h>

#include <HardwareSerial.h>


//Libraries for OLED Display

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


//define the pins used by the LoRa transceiver module

#define SCK 5

#define MISO 19

#define MOSI 27

#define SS 18

#define RST 14

#define DIO0 26

#define BAND 866E6


//OLED pins

#define OLED_SDA 4

#define OLED_SCL 15

#define OLED_RST 16

#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);

HardwareSerial seri(2);

#define RXD2 23
```

```
#define TXD2 17

void setup() {
seri.begin(115200,SERIAL_8N1,RXD2,TXD2);  //RD TX
//reset OLED display via software
pinMode(OLED_RST, OUTPUT);
digitalWrite(OLED_RST, LOW);
delay(20);
digitalWrite(OLED_RST, HIGH);
//initialize OLED
Wire.begin(OLED_SDA, OLED_SCL);
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for 128x32
for(;;); // Don't proceed, loop forever
}
display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0,0);
display.print("LORA SENDER ");
display.display();
//SPI LoRa pins
SPI.begin(SCK, MISO, MOSI, SS);
//setup LoRa transceiver module
LoRa.setPins(SS, RST, DIO0);

if  (!LoRa.begin(BAND)) {
while (1);
}
Serial.println ("LoRa Initializing OK!");
display.setCursor(0,10);
display.print("LoRa Initializing OK!");
```

```
display.display();
delay(2000);
}

void loop() {
int contador;
if( seri.available()){
char valor=seri.read();
switch (valor){
case '0':
contador=0;
break;
case '1':
contador=1;
break;
case '2':
contador=2;
break;
case '0':
contador=3;
break;
case '4':
contador=4;
break;
case '5':
contador=5;
break;
}
//Send LoRa packet to receiver
LoRa.beginPacket();
LoRa.print (contador);
```

```
LoRa.endPacket ();

display.clearDisplay();
display.setCursor(0,0);
display.println("TFG-Maria Frances");
display.setCursor(0,20);
display.setTextSize(1);
display.print("LoRa sender");
display.setCursor (0,30);
display.print ("Paquete:");
display.setCursor (50,30);
display.print (contador);
display.display();
delay(100);
}
}
```

Anexo II: Código dispositivo LoRa receptor

```
//Libraries for LoRa
#include <SPI.h>
#include <LoRa.h>
#include <HardwareSerial.h>

//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26
#define BAND 866E6

//OLED pins
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
HardwareSerial seri(2);
#define RXD2 23
#define TXD2 17
String LoRaData;
```

```
void setup() {
seri.begin(115200,SERIAL_8N1,RXD2,TXD2);  //RD TX
//reset OLED display via software
pinMode(OLED_RST, OUTPUT);
digitalWrite(OLED_RST, LOW);
delay(20);
digitalWrite(OLED_RST, HIGH);
//initialize OLED
Wire.begin(OLED_SDA, OLED_SCL);
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for 128x32
for(;;); // Don't proceed, loop forever
}
display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0,0);
display.print("LORA RECEPTOR");
display.display();
//SPI LoRa pins
SPI.begin(SCK, MISO, MOSI, SS);
//setup LoRa transceiver module
LoRa.setPins(SS, RST, DIO0);

if  (!LoRa.begin(BAND)) {
while (1);
}
Serial.println ("LoRa Initializing OK!");
display.setCursor(0,10);
display.print("LoRa Initializing OK!");
display.display();
delay(2000);
```

```
}

void loop() {
int packetSize = LoRa.parsePacket();
if (packetSize) {
//read packet
while (LoRa.available()) {
LoRaData = LoRa.readString();
}
//print RSSI of packet
int rssi = LoRa.packetRssi();
// Display information
display.clearDisplay();
display.setCursor(0,0);
display.print("LORA RECEIVER");
display.setCursor (0,20);
display.print ("Paquete recibido:");
display.setCursor (0,30);
display.print (LoRaData);
display.setCursor (0,40);
display. print ("RSSI:");
display.setCursor (30,40);
display.print (rssi);
display.display ();
int aux;
if (LoRaData=="0")
aux=0;
if (LoRaData=="1")
aux=1;
if (LoRaData=="2")
aux=2;
```

```
if (LoRaData=="3")
aux=3;
if (LoRaData=="4")
aux=4;
if (LoRaData=="5")
aux=5;
seri.write(aux);
}
}
```

Anexo III: Código módulo conexiones

```
#include <SoftwareSerial.h>

SoftwareSerial puertoblu (4,2); //RX y TX

SoftwareSerial puertowifi (8,9); //RX y TX

char valor;


void setup() {

puertoblu.begin(9600);

puertowifi.begin(115200);

}


void loop() {

puertoblu.listen();


if(puertoblu.available()){

valor=puertoblu.read();

puertowifi.print(valor); }


}
```

Anexo IV: Código módulo actuadores

```
//Puerto serial 2 - GPS

//Puerto serial 1 - wifi

//pin 10 motorderecha y pin 11 motor izquierda

// TFG Maria Frances Perez

#include <TinyGPS.h> // Libreria de manejo del GPS

#include <HardwareSerial.h> // Libreria de manejo de comunicacion serial alterna

#include <Servo.h>

#include <math.h>

#include <PIDController.hpp>

#include <Wire.h>

#include <MechaQMC5883.h>


Servo ESCiz;

Servo ESCde;

TinyGPS GPS;

MechaQMC5883 aguja;


double xOrig = 0, yOrig = 0, latOrig,lonOrig,x,y,Arad;

const float declinacion = 0.48;

int mx,my,mz;


PID::PIDParameters<double> parameters(2.0, 0.5, 1);

PID::PIDController<double> pidController(parameters);



long lat, lon;  // Variables auxiliares para lectura de longitud y latitud

float LAT, LON; // Variables auxiliares para latitud y longitud con punto decimal

char datorecogido;

int variableusuario, modo=3;

float POSLAT=38.7034113, POSLON=-0.4850416; //longitud y latitud objetivo
```

```
double Kp=200000, Ki=5, Kd=5,Input1, Output1, Setpoint1, Input2, Output2, Setpoint2,
Output;


unsigned long currentTime, previousTime;
double elapsedTime, error, lastError, cumError, rateError;
unsigned long currentTime2, previousTime2;
double elapsedTime2, error2, lastError2, cumError2, rateError2;
float anguloobj;
void setup(){
  Serial2.begin(9600);    // GPS
  Serial.begin(115200); // Velocidad de comunicacion con la computadora
  Serial1.begin (115200); //Wifi
  ESCiz.attach(9);
  ESCde.attach(10);
  Wire.begin();
  aguja.init();
  lectura_serial();
  latOrig=LAT;
  lonOrig=LON;
  conversionUTM(latOrig, lonOrig);
  xOrig=x;
  yOrig=y;
  anguloobj=atan2(yOrig,xOrig);
  anguloobj = anguloobj * RAD_TO_DEG;
  pidController.Input = 80;
  pidController.Setpoint = anguloobj;
  delay(2000);
  pidController.TurnOn();
}


void loop(){
```

```
 if(Serial1.available()){
  Serial.println("Empezamos");
  variableusuario=Serial1.read();
  Serial.println(variableusuario);

  if(variableusuario==0){
   modo=0; //funciona como control remoto
   Serial.println("control");
   }
  else if(variableusuario==1){
    modo=1; //funciona como PID
    Serial.println("bucle cerrado");
    }

  if(modo==0){
    //comandos para dirección según usuario
    motores(variableusuario);
  }

  if (modo==1){
   lectura_serial();
   calculoPID();
  }
 }

 delay(100);
}

void motores(int direccion){
 //si la direccion es 2 significará derecha, 3 izquierda, 4 atrás, 5 delante
 switch (direccion){
```

```
case 2:
Serial.println("Hacia la derecha");
  ESCiz.write(40);
  ESCde.write(90);
  delay(1000);
  ESCiz.write(90);
  ESCde.write(90);
 break;
 case 3:
 Serial.println("Hacia la izquierda");
  ESCiz.write(90);
  ESCde.write (40);
  delay(1000);
  ESCiz.write (90);
  ESCde.write (90);
 break;
 case 4:
 Serial.println("Hacia atrás");
 ESCiz.write(40);
  ESCde.write (120);
  delay(1000);
  ESCiz.write(90);
  ESCde.write(90);
 break;
 case 5:
 Serial.println("Hacia delante");
  ESCiz.write(40);
  ESCde.write(110);
  delay(1000);
  ESCiz.write(90);
  ESCde.write(90);
```

```
    break;
  }
}
void lectura_GPS(){
  bool newdata = false;            // Variable auxiliar para ingreso de datos
  unsigned long start = millis();    // Variable auxiliar para refresh y update datos

  while (millis() - start < 1000){   // Update de lectura cada segundo
    if (feedgps ()){newdata = true;}  // Validacion de datos en la pila
  }
  if (newdata){informacion_gps(GPS);} // Ingreso de datos
}

bool feedgps(){
  while (Serial2.available()){     // Mientras exista informacion en la pila de datos
    if (GPS.encode(Serial2.read())) // Validacion de existencia de datos en la pila de datos
      return true;
  }
  return 0;
}

void informacion_gps(TinyGPS &GPS)
{
  GPS.get_position(&lat, &lon); // Byte de datos de latitud y longitud
  LAT = lat; // Adquicion de datos en decimales
  LON = lon; // Adquicion de datos en decimales
  {
    feedgps(); // Sin esta función, puede causarse un error de suma
  }
}
```

```
void lectura_serial(){
 GPS.get_position(&lat, &lon);    // Adquicion de datos de latitud y longitud
 lectura_GPS();                   // Funcion de rutina de lectura del GPS
}
void calculoPID(){

 aguja.read(&mx,&my,&mz);
 float difx, dify;
 difx=abs(mx-lat);
 dify=abs(my-lat);
 if(difx<0.05&&dify<0.05){
  ESCiz.write(90);
   ESCde.write(90);
 }
 else{

 float angulo = atan2(my, mx);
 angulo = angulo * RAD_TO_DEG;
 angulo = angulo - declinacion;
 if(angulo < 0) angulo = angulo + 360;
 pidController.Input = angulo;
 pidController.Update();
  int jol=pidController.Output;
 }
}
void conversionUTM (double lati, double longi)
{
 /*!
  * Transformación de las coordenadas geográficas a UTM
  */
 /// Sobre la geometría del delipsoide WGS84
```

```
double a = 6378137.0;
double b = 6356752.3142;

//  float e = sqrt((a*a) + (b*b))/a; ///< Excentricidad.
double e = sqrt((a*a) - (b*b))/b; ///< Segunda excentricidad.
double e2 = e * e; ///< al cuadrado. Usaremos esta directamente.

double c = a*a / b; ///< Radio Polar de Curvatura.

/// Sobre la longitud y latitud. Conversión de grados decimales a radianes.

/*!
 * Cálculo del signo de la longitud:
 *     - Si la longitud está referida al Oeste del meridiano de Greenwich,
 *       entonces la longitud es negativa (-).
 *     - Si la longitud está referida al Este del meridiano de Greenwich,
 *       entonces la longitud es positiva 8+).
 */

double latRad = lati * PI / 180.0; ///< Latitud en Radianes.
double lonRad = longi * PI / 180.0; ///< Longitud en Radianes.

/// Sobre el huso.
float huso = (longi/6)+31;
int h=int(huso); //< Nos interesa quedarnos solo con la parte entera.
int landa0 = h * 6 - 183; ///< Cálculo del meridiano central del huso en radianes.
double Dlanda = lonRad - (landa0 * PI / 180.0);
///< Desplazamiento del punto a calcular con respecto al meridiano central del huso.

/*!
 * Ecuaciones de Coticchia-Surace para el paso de Geográficas a UTM (Problema directo);
```

```
 */

/// Cálculo de Parámetros.
double coslatRad = cos (latRad);
double coslatRad2 = coslatRad * coslatRad;

double A = coslatRad * sin (Dlanda);
double xi = 0.5 * log ((1 + A) / (1 - A));
double n = atan(tan(latRad) / cos(Dlanda)) - latRad;
double v = (c / sqrt(1 + e2 * coslatRad2)) * 0.9996;
double z = (e2/ 2.0) * xi * xi * coslatRad2;
double A1 = sin (2 * latRad);
double A2 = A1 * coslatRad2;
double J2 = latRad + (A1 / 2.0);
double J4 = (3.0 * J2 + A2) / 4.0;
double J6 = (5.0 * J4 + A2 * coslatRad2) / 3.0;
double alf = 0.75 * e2;
double bet = (5.0 / 3.0) * alf * alf;
double gam = (35.0 / 27.0) * alf * alf * alf;
double Bfi = 0.9996 * c * (latRad - alf * J2 + bet * J4 - gam * J6);

/*!
 * Cálculo final de coordenadas UTM
 */
Serial.println (" Coordenadas UTM actuales: ");
x = xi * v * (1 + (z / 3.0)) + 500000; /*!< 500.000 es el retranqueo que se realiza en cada huso sobre el origen de  coordenadas en el eje X con el objeto de que no existan coordenadas negativas. */
Serial.print (" X = ");  Serial.print (x,5);  Serial.print (" (m)");
y = n * v * (1 + z) + Bfi;  /*!< En el caso de latitudes al sur del ecuador, se sumará al valor de Y 10.000.000  para evitar coordenadas negativas. */
 Serial.print (" Y = ");  Serial.print (y,5);  Serial.println (" (m)");  }
```

DISEÑO E IMPLEMENTACIÓN DEL CONTROL DE LA NAVEGACIÓN AUTÓNOMA DE UN MODELO A ESCALA DE EMBARCACIÓN

2.PLANOS

Autora: María Francés Pérez

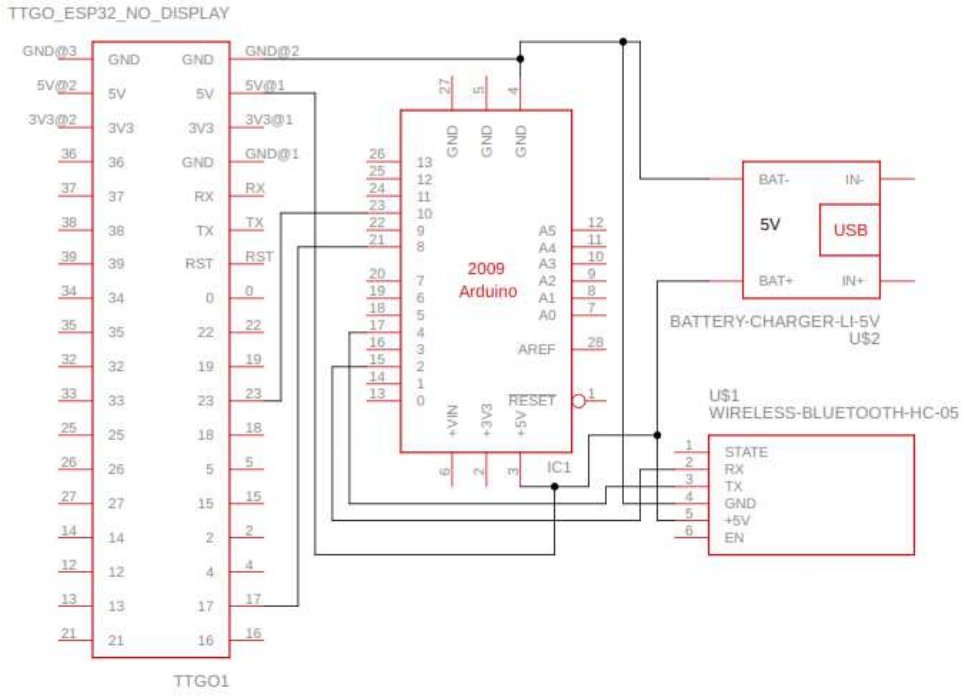
Tutor: Jose Vicente Busquets Mataix

Tutor externo: Javier Busquets Mataix

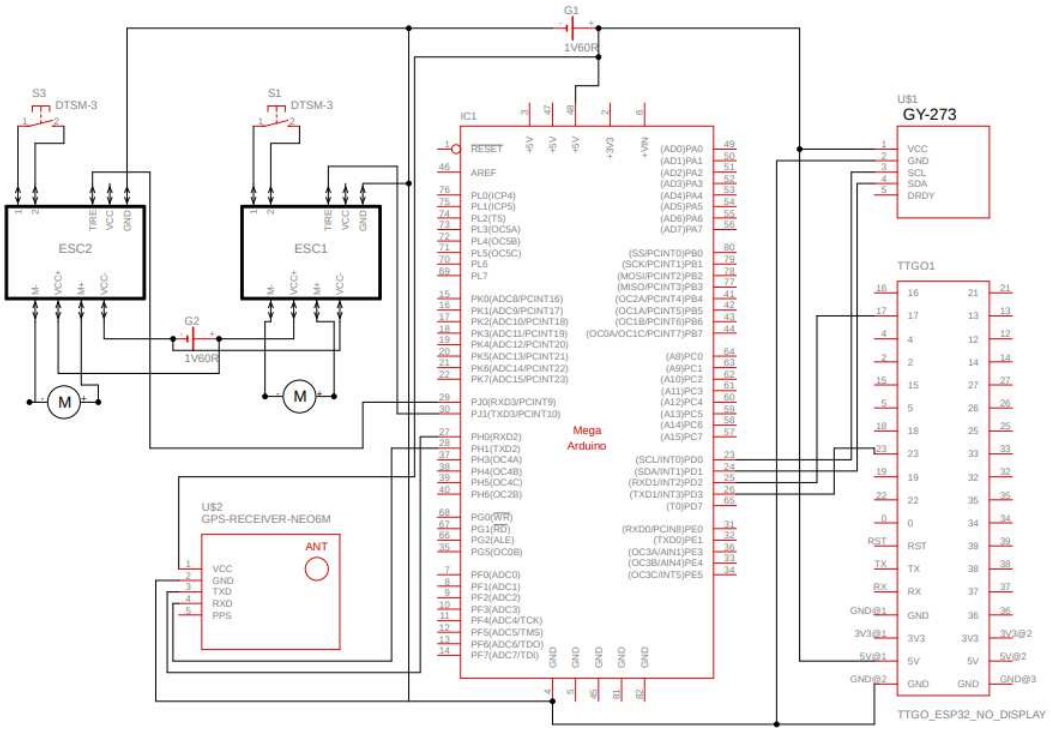
Planos

1. Plano de conexión del módulo de comunicaciones
2. Plano de conexión del módulo de actuadores

Diseño e implementación del control de la navegación autónoma de un modelo a escala de embarcación



PROYECTO: Trabajo Fin de Grado Escuela Técnica Superior de Ingeniería del Diseño Título: Diseño e implementación del control de la navegación autónoma de un modelo a escala de una embarcación		Escala S/E
Autor:María Francés Pérez	Plano: Esquema de conexiones	Plano Nº 01



PROYECTO: Trabajo Fin de Grado Escuela Técnica Superior de Ingeniería del Diseño Título: Diseño e implementación del control de la navegación autónoma de un modelo a escala de una embarcación		Escala S/E
Autor:María Francés Pérez	Plano: Esquema de conexiones	Plano Nº 02

DISEÑO E IMPLEMENTACIÓN DEL CONTROL DE LA NAVEGACIÓN AUTÓNOMA DE UN MODELO A ESCALA DE EMBARCACIÓN

3. PLIEGO DE CONDICIONES

Autora: María Francés Pérez

Tutor: Jose Vicente Busquets Mataix

Tutor: Javier Busquets Mataix

Pliego de condiciones

1. Introducción

El presente Pliego de Condiciones Técnicas es el resumen de las características que se deberán de cumplir en la ejecución del proyecto de la automatización de una embarcación que deberá cumplir una serie de requerimientos.

Los requerimientos se agrupan principalmente en dos formas de funcionamiento totalmente independientes. La embarcación deberá de ser capaz de seguir una trayectoria siguiendo unos puntos establecidos por el usuario y también podrá ser controlada por el usuario.

2. Requerimientos funcionales

Los requerimientos funcionales se agrupan según modo de funcionamiento, ya que son totalmente independientes entre ellos.

2.1. Funcionamiento remoto

1) El usuario ha de tener un control total de la embarcación y todas las acciones que esta haga.
2) El usuario ha de poder mover la embarcación hacia delante, la derecha, la izquierda o hacía atrás.

2.2. Funcionamiento autónomo

1) La embarcación ha de ser capaz de seguir una trayectoria siguiendo unas coordenadas previamente establecidas por el usuario.
2) La embarcación deberá moverse hacia delante, la derecha, la izquierda dependiendo de la trayectoria que necesite realizar.
3) La embarcación enviará información al usuario sobre el su estado, es decir, información de posición y velocidad.

3. Requisitos no funcionales

Los requisitos no funcionales del proyecto son comunes a los dos modos de funcionamiento.

1) La temperatura máxima de trabajo de la embarcación es de 50 grados centígrados.
2) La autonomía mínima de la embarcación ha de ser al menos de 20 minutos
3) La embarcación está preparada para trabajar a 20 km/h y sin lluvia.

4. Normativa

La normativa que aplica a este proyecto es la siguiente:

La ETSI EN 300 220-1 marca que en los dispositivos LoRa la frecuencia ha de ser de 868 MHz en Europa y el ciclo de trabajo no ha de ser superior al 1 % para dispositivos finales.

La ley 11981/2014 prohíbe su navegación por puertos y que presten auxilio a personas que se encuentre en el mar ni podrán informar sobre polizones o administrarles manutención o asistencia médica.

DISEÑO E IMPLEMENTACIÓN DEL CONTROL DE LA NAVEGACIÓN AUTÓNOMA DE UN MODELO A ESCALA DE EMBARCACIÓN

4.PRESUPUESTO

Autora: María Francés Pérez

Tutor: Jose Vicente Busquets Mataix

Tutor externo: Javier Busquets Mataix

Presupuesto

El presupuesto del proyecto es el siguiente:

COSTES DE MATERIALES

Descripción	Uds	Coste (€/ud)	Cantidad	Precio total (€)
NQD Storm Engine PX-16 con motor 390 y batería 7.2 V	u	70,00 €	1	70,00 €
ESC 320 Brushed	u	14,42 €	2	28,84 €
Arduino UNO	u	9,99 €	1	9,99 €
Arduino Mega	u	13,99 €	1	13,99 €
TTGO LoRa32 OLED v1	u	13,00 €	2	26,00 €
GY-273	u	1,30 €	1	1,30 €
Skylab SKM53	u	19,17 €	1	19,17 €
16340 4800mA 3.7 V	u	3,65 €	1	3,65 €
ESP8266 ESP32	u	2,00 €	1	2,00 €
TOTAL				174,94 €

COSTES MANO DE OBRA

Descripción	Uds	Precio por ud	Cantidad	Precio total (€)
Estudio del proyecto	h	20€/h	10	200,00 €
Investigación	h	20€/h	20	400,00 €
Diseño del software	h	20€/h	35	700,00 €
Implementación del software	h	20€/h	40	800,00 €
Pruebas	h	20€/h	18	360,00 €
Documentación	h	20€/h	30	600,00 €
TOTAL				3.060,00 €

COSTES EXTRAS

Descripción	Precio total (€)
Amortización del PC	100,00 €
Office 365	16,00 €
TOTAL	116,00 €

PRESUPUESTO

Descripción	Precio total (€)
Coste de materiales	174,94 €
Costes mano de obra	3.060,00 €
Costes extras	116,00 €
TOTAL	3.350,94 €