UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Diseño e implementación del sistema de control de un ROV

Autor: Sánchez Durán, Federico

Director: Garrigós Guerrero, Francisco Javier

Codirector: Aguirre Martínez, José Luis

Universidad Politécnica de Cartagena

Índice

Capítulo 1
Introducción

El desarrollo de vehículos submarinos no tripulados (UUVs) ha experimentado en los últimos años una importante expansión En muchos casos, estos vehículos, se han convertido en un sustituto ideal del ser humano a la hora de llevar a cabo trabajos bajo el agua en los que existen condiciones extremas o peligrosas.

Actualmente se están realizando importantes esfuerzos con el fin de desarrollar sistemas que permitan a los UUVs cumplir misiones de forma autónoma, o que ayuden al operador del vehículo en la navegación, para que éste pueda concentrarse en los objetivos principales del trabajo, y no en el guiado del vehículo.

Los UUVs se clasifican en dos grandes grupos, los denominados ROV (Remotely Operated Vehicle) son vehículos controlados de forma remota por un humano y los AUV (Autonomous Underwater Vehicle), vehículos que se controlan de forma totalmente autónoma.

Antecedentes

El proyecto que nos ocupa, se inició gracias al interés que mostró la empresa Navantia en el diseño y la creación de un prototipo de un nuevo sistema de recogida de vehículos submarinos autónomos. Para la realización de este proyecto se contó con el apoyo del Departamento de Ingeniería Mecánica de la Universidad Politécnica de Cartagena.

Durante los últimos años se han diseñado gran cantidad de modelos de vehículos autónomos, sin embargo, ahora se pretende construir un prototipo para comprobar los modelos. Con esta aspiración, se decidió crear un vehículo autónomo, de diseño sencillo, capaz de realizar maniobras complejas, movimientos y giros en las tres dimensiones espaciales, y con cabida para modificaciones, tanto del modelo informático, como del prototipo real. La realización de una simulación previa del comportamiento del prototipo nos permitirá optimizar su diseño.

A lo largo de los últimos años en el Departamento de Ingeniería Mecánica de la UPCT se han realizado modelos dinámicos de diferentes vehículos submarinos no tripulados. También se han desarrollado algoritmos que permiten el control de dichos vehículos durante las misiones simuladas. Con este proyecto se pretende desarrollar, mediante hardware específico, el sistema de control de uno de los ROV diseñado y modelado.

Este proyecto constituye la continuación de algunos proyectos anteriores como los son el de Jorge Juan García. Su proyecto consistió en el desarrollo de una herramienta informática para la simulación dinámica de vehículos submarinos no tripulados, de tal forma que se obtuvo un software en MATLAB incluyendo la formulación que desarrolló del movimiento para los seis grados de libertad de vehículos submarinos, logrando su resolución.

Otro de los proyectos que han servido de antecedente al nuestro es el del diseño de un prototipo de vehículo autónomo submarino desarrollado por Alberto Bonillo Legaz. Alberto consiguió la obtención de un diseño hidrodinámico óptimo, con una resistencia al avance de valores muy bajos. Así mismo, su diseño permitió obtener una fiabilidad estructural, de tal forma que el vehículo es capaz de soportar unas cotas de inmersión muy elevadas sin afectar a las características hidrodinámicas del mismo. Finalmente, fue capaz de obtener unos resultados de autonomía elevados y otorgarle al dispositivo la capacidad de disponer de módulos auxiliares para realizar diferentes misiones.

Antonio Garrido Pellicer, llevó a cabo la realización de un proyecto de estimación de los coeficientes hidrodinámicos de vehículos autónomos submarinos mediante CFD. Con este proyecto se realizó un estudio básico de mercado en el que se eligió el vehículo autónomo REMUS 100 para someterlo a estudio. Mediante el software CFD Tdyn se elaboró una estimación de la resistencia al avance del vehículo elegido, simulando, para ello, el movimiento en 2D del vehículo, posteriormente el movimiento 3D sin apéndices, y, finalmente, el movimiento 3D con apéndices en el vehículo para obtener un estudio lo más veraz posible. Para finalizar, se llevó a cabo la definición de los coeficientes hidrodinámicos para los seis grados de libertad de los vehículos que estudiaba y la estimación de los mismos mediante el software de CFD.

Javier de la Red Calvo, consiguió diseñar cuatro modelos distintos del vehículo, con el añadido de dos programas que permiten cambiar las dimensiones básicas de dos de ellos. Así mismo, en ese proyecto, se modelaron las hélices y los motores de propulsión, además se obtuvo el centro de gravedad, la matriz de inercia, el peso y el volumen de dos de los modelos. También se calcularon de forma aproximada algunos de los parámetros. Finalmente, se llevó a cabo el diseño de un controlador proporcional derivativo para el guiado del vehículo hacia puntos de paso. Todo fue realizado mediante programación en MATLAB y C++.

Álvaro García Foz, realizó el diseño de un vehículo subacuático de bajo coste y con materiales al alcance de usuarios no expertos, acercando dicha tecnología al gran público, uso de electrónica y materiales de bajo coste.

El proyecto de Eloy Yaguë Martínez, en el que llevó a cabo la creación, mediante lenguaje Visual Basic, de una interfaz capaz de realizar varias tareas y comunicarse en tiempo real mediante conexión UDP con otro host. La interfaz diseñada se emplea para realizar el control de un vehículo submarino manipulado de forma remota o ROV y recibe las instrucciones de un joystick de control que se comunica directamente con la interfaz, y esta última, realiza el envío de sus comandos al otro host.

El presente proyecto desarrollado queda muy ligado al de Javier de la Red y Eloy Yagüe. Del proyecto realizado por Javier de la Red Calvo se ha utilizado el modelo que desarrollo en 2014 bajo entorno Matlab del UUV para diseñar y optimizar el control del vehículo, de otro modo, del proyecto de Eloy Yaguë Martínez se colaboró en el diseño de aplicaciones de apoyo a la Interfaz Gráfica que fue desarrollada en 2016.

Objetivos

Este proyecto pretende desarrollar, a través de dispositivos electrónicos específico de bajo coste (Ordenador de placa reducida Raspberry Pi 2 B y una placa basada en microcontrolador chipKIT Max32 o Arduino UNO), el sistema de control de un UUV de 6 grados de libertad, gobernado por 6 propulsores que integra además el siguiente equipamiento e instrumentación: 2 focos de 2000 lm, 1 cámara de video, micrófono, IMU, medidor de presión, sensores de temperatura y humedad. Se pretende realizar la implementación electrónica de todas las partes que conforman el sistema de monitorización y control del vehículo, incluyendo la programación necesaria para, en una primera fase, poder visualizar los diferentes parámetros captados con los instrumentos a bordo y realizar el control manual del vehículo con estabilización autónoma, y en una segunda fase, obtener un control totalmente autónomo del UUV.

Fases del proyecto

1. Búsqueda de información sobre los diferentes equipos y sistemas que integra el vehículo, así como de los lenguajes de programación más adecuados para la integración de los diferentes dispositivos y la implementación del software de comunicación y control.
2. Estudio de la instrumentación y de los sistemas de gobierno del ROV a controlar.
3. Diseño e integración de los diferentes dispositivos para conformar un sistema de control manual del ROV que permita realizar maniobras con un joystick conectado a un PC externo.
4. Programación de herramientas de apoyo al desarrollo de la interfaz gráfica.
5. Mejora del código de programación y entorno de la interfaz gráfica.
6. Diseño e implementación del sistema de control para la estabilización del ROV.
7. Verificación del funcionamiento de los sistemas desarrollados
8. Análisis de los resultados.
9. Redacción del texto del proyecto final de carrera.

Capítulo 2

Estado del Arte

Los vehículos subacuáticos no tripulados (UUV), a veces conocidos como drones submarinos, son vehículos que son capaces de operar bajo el agua sin un ocupante humano. Estos vehículos pueden dividirse en dos categorías: los vehículos submarinos operados remotamente (ROV), que son controlados por un operador humano remoto, y vehículos autónomos submarinos (AUVs), que funcionan independientemente del operador humano. Esta última categoría constituiría una especie de robot.

ROV

Un ROV (acrónimo del inglés Remote Operated Vehicle, (Vehículo operado remotamente) es un robot submarino no tripulado y conectado a una unidad de control en la superficie por medio de un cable umbilical. Las órdenes se envían mediante un mando de control a través del cable al ROV. La energía del vehículo puede ser enviada a través del cable de unión o puede provenir de unas baterías que tenga el propio ROV

A través del cable umbilical se transmiten también los datos de las cámaras de video del ROV, los datos de los sensores y de los sonares a la unidad de control en la superficie. Los ROVS pueden llevar una gran variedad de brazos manipuladores, herramientas y sensores para realizar trabajos en las profundidades, o simplemente una cámara de video con el fin de captar las imágenes del fondo del mar.

Una vez en el ROV la energía eléctrica se divide y se distribuye entre los diferentes componentes del ROV. Sin embargo, en aplicaciones que requieren alta potencia, la mayor parte de la energía eléctrica se utiliza para accionar un motor de alta potencia eléctrica que a su vez acciona una bomba hidráulica. La bomba hidráulica se utiliza para alimentar equipos tales como propulsores, herramientas de torsión y brazos manipuladores.

La mayoría de los ROVs están equipados con al menos una cámara de vídeo y luces. El equipo adicional se agrega comúnmente para ampliar las capacidades del vehículo. Estos pueden incluir sonares, magnetómetros, cámara fotográficas, un brazo manipulador, herramienta de corte, sistemas para toma de muestras, e instrumentos para medir parámetros.

Un ROV puede lanzarse "en vuelo libre" en el que el vehículo se opera a través del umbilical (Tether) a la unidad de control de superficie o pueden ser operados desde planta a través del umbilical. El garaje se baja desde el buque. Ambas técnicas tienen sus pros y sus contras. En operaciones muy profundas, el ROV trabaja desde un 'garaje submarino' al que está conectado el ROV y que a su vez está conectado a una plataforma en superficie

Un sistema ROV básico incluirá los siguientes módulos:

- El vehículo.
- Un sistema de lanzamiento y recuperación.
- Una unidad de suministro de energía.
- Una consola para el control.
- Monitor.
- El equipo humano de ROV.
-

Tipos de ROV

Actualmente, los ROV's son clasificados en cinco grandes grupos de acuerdo con las tareas que desempeñan:

- Clase 1: vehículos solamente de observación
- Clase 2: vehículos para observación y transporte de pequeñas cargas
- Clase 3: vehículos para trabajos generales a nivel de intervención
- Clase 4: Tractores sumergibles o enterradores de líneas o cables
- Clase 5:.Prototipos y vehículos en desarrollo

Breve repaso histórico

Existen dos a los cuales se les acredita el desarrollo del primer ROV, el PUV (Programmed Underwater Vehicule) que fue un torpedo desarrollado por Luppis- Whitehead Automobile en Austria en 1864 y el llamado POODLE, que fue desarrollado por Dimitri Rebikoff en 1953.

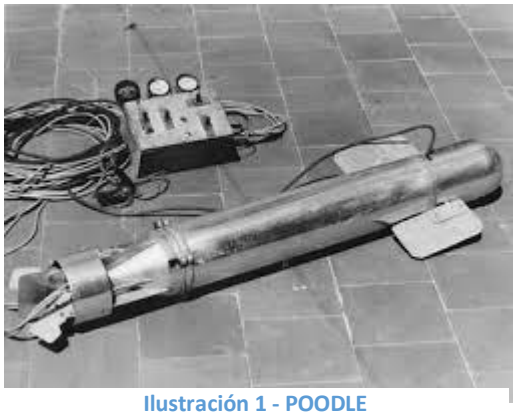


Ilustración 1 - POODLE

Se le atribuye a la Marina de los Estados Unidos el avance de la tecnología a un estado más avanzado para desarrollar los robots para recuperar los artefactos subacuáticos perdidos durante las pruebas en el océano. Financio la mayor parte del desarrollo de la tecnología ROV en los años 60 en lo que entonces se llamó CURV (Cable Controlled Underwater Recovery Vehicule) dando la capacidad de realizar operaciones de rescate en agua profundas y recuperar objetos del fondo del océano como el caso de la bomba nuclear perdida en el Mar Mediterráneo después del accidente de Palomares B-52 de 1966.

Ilustración 2 – Bomba Nuclear

Ilustración 3 - CURV

El siguiente paso en la evolución de la tecnología ROV fue llevado a cabo por las empresas comerciales que vieron futuro en ROV de apoyo a las operaciones petroleras en alta mar. La industria del petróleo y el gas creó la clase de trabajo ROV para ayudar al desarrollo de campos de petróleo en alta mar. Los dos primeros ROV para esta finalidad fueron desarrollados por HydroProducts en Estados Unidos entre los que destacaron el RCV-225 y el RCV-150.

Muchas otras empresas desarrollaron una línea similar de pequeños vehículos de inspección subacuática.

Ilustración 4 – RVC-150

A mediados de la década de 1980, la industria marítima ROV sufrió un grave estancamiento en el desarrollo tecnológico, causado un su mayor parte a la caída del precio del petróleo y una crisis económica mundial.

Desde entonces el desarrollo tecnológico en la industria ROV se ha acelerado y hoy se pueden encontrar ROVs realizando numerosas tareas en muchos campos. Sus tareas abarcan desde la simpe inspección de estructuras submarinas, tuberías y plataformas hasta la conexión de tuberías y la colocación de colectores submarinos.

AUV

Dentro del estado de arte se tiene que hacer mención de los AUVs, ***Autonomous Underwater Vehicles*** por sus siglas en inglés, es un robot que desempeña tareas debajo del agua. Esto significa que no son tripulados, sino que son controlados por un ordenador instalado en el dispositivo o también a bordo de un vehículo de apoyo, estos son accionados por baterías o celdas de combustible y pueden operar alrededor de los 6000 m. de profundidad. Los avances en sistemas de propulsión y fuentes de energía le dan a estos robots mayor duración en cuanto a distancias y tiempos de operación.

En la actualidad la tecnología en los AUV ha llegado a puntos muy importantes ya que ahora es cuando empiezan a ser más conocidos y están llegando a ser una opción comercial más rentable. Últimamente, se ha mostrado mucho interés de la industria petrolera y de gas, ya que al emplear este tipo de tecnologías, se libran de muchos problemas, costos y riesgos que conllevan las actividades de inspección y muestreo subacuáticas. Esto ha llevado a que las empresas privadas, así como los equipos de las organizaciones mundiales principales, han hecho esfuerzos conjuntos para hacer AUVs como parte operativa de la industria del petróleo y del gas, de esta forma se aumentara la comercialización y demanda de este tipo de tecnologías.

Las tareas realizadas por estos dispositivos son principalmente de inspección, investigación y recolección de muestras. En ellos se tiene una mayor autonomía de navegación debido al planeamiento de las trayectorias donde son escogidas para atender puntos específicos basados en curvas batimétricas, pero, la autonomía está condicionada al rendimiento de las baterías de a bordo, para este tipo de vehículos existen dos tipos de navegación:

- Navegación autónoma sin referencia inercial
- Navegación autónoma con referencia inercial

Los sistemas sin referencia inercial tienen aplicaciones de tipo furtivo, son utilizados principalmente en el ámbito militar, trabajan con sistemas de navegación que son orientados basándose en los movimientos del vehículo.

Los sistemas con referencia inercial necesitan tener sistemas auxiliares externos para hacer una constante comparación y reducir el error. Estos vehículos poseen los siguientes sistemas:

- Sistemas de navegación
- Sistemas de sensores
- Sistemas de Energía
- Sistemas de Control
- Sistemas de Comunicaciones

Sensores

Los vehículos submarinos llevan sensores para poder navegar de forma autónoma y mapas de las características del océano.

Ilustración 5 – Cámara submarina

Como sensores más utilizados pueden destacar las brújulas, los sensores de profundidad, los sonares, magnetómetros, sensores de presión, cámaras, sensores de temperatura y sondas de conductividad.

Algunos vehículos están equipados con sensores biológicos incluyendo fluorómetros, sensores de turbidez y sensores para medir el pH y cantidades de oxígeno disuelto.

Sistema de posicionamiento USBL (Ultra Short Line Base)

Es un método de posicionamiento acústico submarino.

Un sistema completo USBL consiste en un transmisor - receptor, que se monta en un poste en un barco, y un transpondedor / respuesta sobre el fondo del mar o en un ROV.

Una computadora, o "unidad de la superestructura", se utiliza para calcular la posición mediante la medición de los rangos por el transmisor-receptor.

Para calcular una posición submarina, la USBL calcula tanto una distancia y un ángulo del transmisor-receptor con el faro submarino. Los ángulos se miden por el transmisor-receptor, que contiene una serie de transductores. El maestro transmisor-receptor contiene normalmente tres o más transductores separados por una línea de base de 10 cm o menos. Un método llamado " fase de diferenciación "dentro de este arsenal del transductor se utiliza para calcular el ángulo para el transpondedor submarino.

Un Pulso acústico es transmitido por el transmisor-receptor y detectado por el transpondedor submarino, que responde con su propio pulso acústico. Este pulso de retorno es detectado por el transmisor-receptor a bordo. El tiempo de la transmisión del pulso acústico inicial hasta que la respuesta es detectada se mide por el sistema USBL y se convierte en un rango.

USBL sistemas ofrecen la ventaja de no exigir un fondo marino matriz transponedor. La desventaja es que la precisión de posicionamiento y solidez no es tan buena como para los sistemas LBL (Long Line Base). La razón es que el ángulo fijo resuelto por un sistema de USBL se traduce en un margen de error más grande a mayor distancia. Además, los múltiples sensores

necesarios para la posición del transductor USBL polo y la indemnización de orientación hace introducir errores adicionales. Por último, la falta de uniformidad de las refracciones y reflexiones en el entorno acústico submarino tienen un mayor impacto en el posicionamiento USBL que en el caso de la geometría LBL.

Ventajas

- Únicamente un transceptor solo en la superficie, es decir en uno de los polos.
- Buen nivel de precisión con sistemas de vuelo en tiempo.

Desventajas

- Requiere un sistema de calibración detallado, por lo general no completada rigurosamente.
- La precisión de posición absoluta depende de sensores adicionales, por ejemplo giroscopios en el buque y una unidad de referencia vertical.

Integración de los sistemas de posicionamiento

La integración de sistemas no es más que la combinación o fusión de los sistemas de posicionamiento en la superficie del mar, los sistemas de posicionamiento acústico submarino y los sistemas de mapeo acústico submarino utilizado en diversas aplicaciones. Es importante en la navegación marítima para el sistema o persona que gobierna la nave, conocer la posición del buque cuando esté en mar abierto, en los puertos congestionados y en las vías fluviales. Si bien en el mar, la posición exacta, velocidad y rumbo son necesarios para garantizar que el buque llegue a su destino de la manera más segura, más económica y oportuna que las condiciones lo permitan. La necesidad de información precisa de la posición se vuelve aún más crítico cuando el buque salga o llegue a puerto. El tráfico de buques y otros peligros de navegación durante la maniobra de atraque hace que el riesgo de accidentes aumente.

Sonar de haz simple

Los sensores de sonar de haz simple recopilan las mediciones del fondo del mar. Estos sensores recogen punto o mapa de bits de datos derivados de la fuerza y el momento de la devolución acústico. Sensores de haz único compuesto por un transductor, montado sobre o remolcado por

un vehículo, que se alimenta en un procesador de señales y dispositivo de visualización. La onda de sonido rebota en el fondo del mar y el retorno es capturado por el transductor.

Emiten en una sola frecuencia, típicamente 200 KHz, por encima del sonido audible por el ser humano (ultrasónicas). En trabajos de cartografía ya no se emplean, pero por su facilidad de manejo y fiabilidad hasta los 1000 metros se emplean aun en localización de bancos de peces por ejemplo.

Sonar Multihaz

Se trata de un conjunto de sondas que emiten en varias direcciones a una determinada frecuencia, cubriendo así una mayor zona y posibilitando la corrección de errores mediante la interpolación de los resultados obtenidos. A demás de precisión se gana rapidez y por tanto un ahorro significativo en el gasto que supone cartografiar una zona.

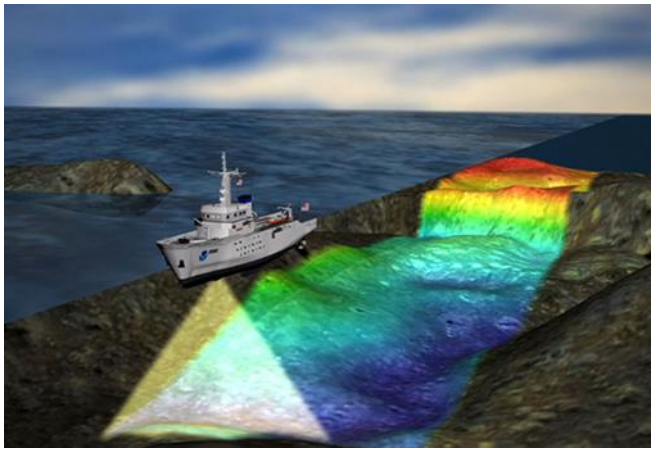


Ilustración 6 - Barrido con sonar multihaz

La siguiente tabla muestra las características típicas de las sondas multihaz:

Tabla 1 – Características sonar multihaz

Frecuencia de muestreo	13 KHz
Profundidad de muestreo (z)	De 50 a 11000 m
Longitud de cobertura	de 5 a 7 veces la altura de agua
Apertura de haz	150º
Numero de haces	162
Resolución por pixel	2,4 m
Velocidad de adquisición máximo	10 nudos
Frecuencia de adquisición	de 2 a 20 segundos
Escala de trabajo	1:100 000 – 1:500 000

Estos sistemas son ampliamente utilizados para los levantamientos hidrográficos de aguas poco profundas en apoyo de la cartografía de navegación. Ecosondas multihaz también son utilizadas para la investigación geológica y oceanográfica, y desde la década de 1990 para el petróleo y la exploración de gas y el fondo marino enrutamiento de cables.

Sondas laterales o Side SCAM Sonar:

Las sondas laterales son, quizás, los dispositivos más versátiles para la realización de batimetrías en un amplio rango de profundidades y resoluciones. La mayoría de las sondas laterales van montadas en dispositivos sumergibles que son arrastrados por un barco, evitando en gran medida la problemática asociada al movimiento de la nave. Este tipo de dispositivos sumergibles se denomina deep tow. En muchos casos, el dispositivo incorpora además de la sonda lateral, numerosos sensores para medir las propiedades del agua, y la naturaleza geológica del terreno. El principio de funcionamiento es muy sencillo: la sonda emite lateralmente ecos en una banda de anchura constante, que se va desplazando con el avance del vehículo. La emisión de este eco caracteriza las irregularidades del terreno permitiendo crear una batimetría de gran precisión, y escalas de hasta 1:10000 1:5000. Pero además, permite el almacenamiento de información acerca de la reflectividad del fondo, para su posterior caracterización.

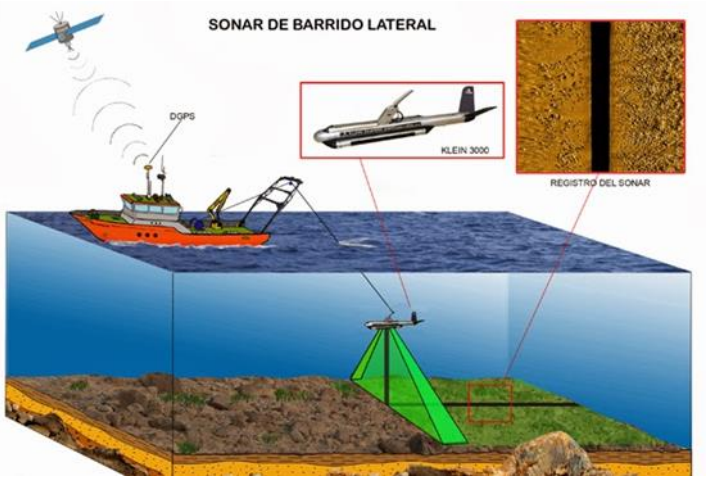


Ilustración 7 - Sonar de barrido lateral

Puede ser utilizado para llevar a cabo estudios acerca de la arqueología marítima, junto con muestras del fondo marino es capaz de proporcionar una comprensión de las diferencias en el material y tipo de textura del fondo marino.

Las imágenes de sonar de barrido lateral son también una herramienta de uso general para detectar puntos de escombros y otros obstáculos en el fondo marino que pueden ser peligrosos para la navegación o al fondo del mar las instalaciones de la industria de petróleo y gas. Además, el estado de los cables y tuberías en el fondo marino pueden ser inspeccionados con sonar de barrido lateral.

El Sonar de barrido lateral también se utiliza para la investigación pesquera, las operaciones de dragado y estudios ambientales. También tiene aplicaciones militares, incluyendo la detección de minas.

Capítulo 3
Descripción del ROV a controlar y Sistema Propuesto

El ROV sobre el que se ha diseñado el sistema de control se denominado *CUBO* y es un vehículo con estructura resistente de acero inoxidable con forma cubica, dotado con 6 propulsores dispuestos en cada una de sus caras de forma simétrica, y emparejados en dirección aquellos que se encuentran colocados en caras opuestas. Esta distribución confiere al vehículo una importante simetría y la capacidad de tener control sobre los 6 grados de libertad.

A la estructura resistente del vehículo se le han añadido 4 cilindros, 2 a cada lado en las que se encuentra la mayor parte de sistemas auxiliares (focos, cámaras de video, micrófonos, sensores de presión, temperatura, salinidad, etc…). Estos tubos están fabricados PVC y sus tapas de cierre en ABS y metacrilato.

Los propulsores son los T-100 de la marca BlueRobotic, cuyas características más importantes se muestran en la tabla 0. Su forma y dimensiones aparecen en la figura 0.

Tabla 2 – Principales características del propulsor T-100

Máximo empuje positivo	23.15 N
Máximo empuje negativo	18.15 N
Mínimo empuje	0.1 N
Velocidad de rotación	300-4200 rpm
Eléctricas	
Tensión de operación	6-16 V
Max corriente	12.5 A
Max potencia	135 w
Físicas	
Longitud	101 mm
Diámetro	97 mm

Diámetro de la hélice	76 mm

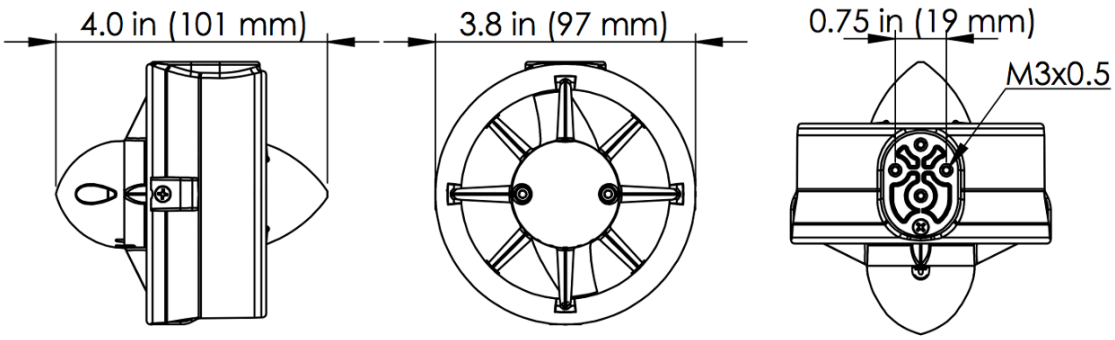


Ilustración 8 - Forma y dimensiones de los propulsores T-100 integrados en CUBO

El diseño de cada uno de los componentes y el ensamblaje final se ha realizado con la herramienta *SolidWorks®*. Esta herramienta ha permitido, mediante el análisis por elementos finitos, calcular la resistencia y deformaciones que tendría el vehículo en función de la profundidad a la que se sumerja. De este análisis se ha extraído que la profundidad máxima para operar de forma segura con este ROV estará algo por encima de los 100 metros.

En el diseño ha primado la sencillez, la economía, la simetría y la movilidad.

En la figura 1 se muestra el diseño completo del ROV, en la figura 2 un esquema con las tres vistas y las cotas dimensionales básicas y en la figura 3 una imagen del estado actual en el que se encuentra su fabricación.

Las principales características físicas del vehículo se muestran en la Tabla 1.

Tabla 3 - Principales características físicas del ROV

Dimensiones	Solo estructura de acero: 290×290×290 mm Con los tubos auxiliares: 290×422,43×291 mm De punta a punta de hélice: 560×560×560 mm
Peso en aire	20.9 Kg
Volumen	21 dm^3
C.D.G respecto al C.D.C (Centro De Carena)	X: 0 mm Y: 0 mm Z: 20 mm
Momentos de inercia principales	Ixx: 0.77 Kgm^2 Iyy: 0.73 Kgm^2

	Izz: 0.75 Kgm^2
Materiales de fabricación:	Acero inoxidable AISI 316L ABS Metacrilato PET

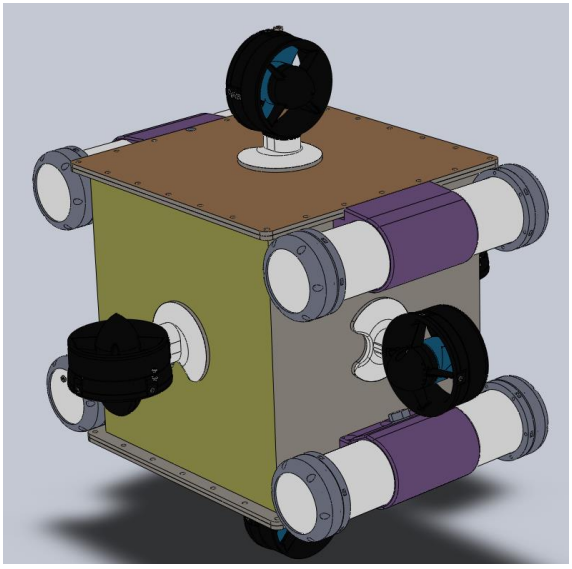


Ilustración 9 - Ensamblaje completo del ROV en *SolidWorks®*.

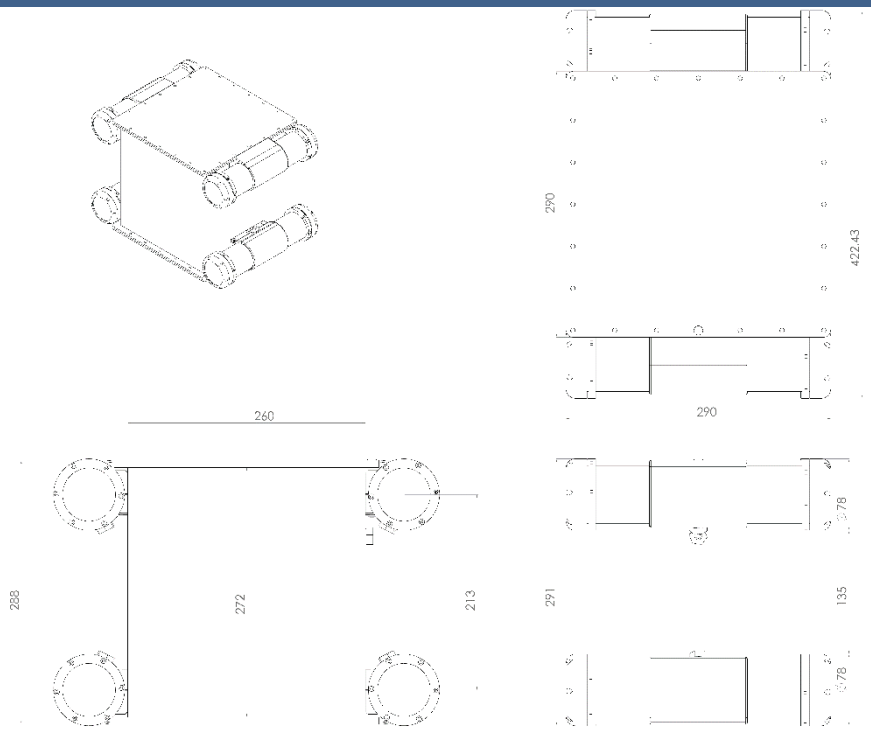


Ilustración 10 - Esquema 3D de CUBO junto a sus vistas acotadas de forma básica

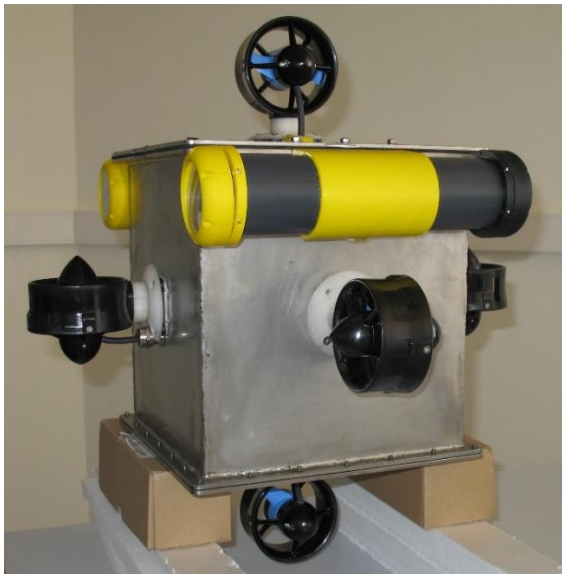


Ilustración 11 - Estado actual del ROV diseñado

Modelo del ROV

Para obtener el comportamiento dinámico del vehículo sumergido en agua y sometido a corrientes submarinas se han utilizado la ecuación de la cinemática de un sólido rígido en el espacio (1) y la ecuación de la dinámica para un objeto propulsado dentro de un fluido no compresible (2). En la Tabla 3 se describen cada uno de los términos de estas dos ecuaciones.

$$\dot{\boldsymbol{\eta}} = J \cdot \boldsymbol{\upsilon} \tag{1}$$

$$M_{SR}\dot{\boldsymbol{\upsilon}} + M_A\dot{\boldsymbol{\upsilon}} + C_{SR}(\boldsymbol{\upsilon})\boldsymbol{\upsilon} + C_A(\boldsymbol{\upsilon}_r)\boldsymbol{\upsilon}_r + D(\boldsymbol{\upsilon}_r)\boldsymbol{\upsilon}_r + g(\boldsymbol{\eta}) - A_c(\upsilon_c)\boldsymbol{\upsilon}_r = \tau \tag{2}$$

Tabla 4 - Descripción de términos de la ecuación cinemática y dinámica de un vehículo sumergido en agua

Término	Descripción
$\boldsymbol{\eta}$	**Vector de posición (respecto a un sistema fijo a tierra)**
$\boldsymbol{\upsilon}$	**Vector de velocidad (respecto a un sistema fijo al vehículo)**
$\boldsymbol{J}$	**Matriz de transformación de coordenadas**
$\boldsymbol{M_{SR}}$	**Matriz de masas**
$\boldsymbol{M_A}$	**Matriz de masa añadida**
$\boldsymbol{C_{SR}()}$	**Matriz de Coriólis del sólido rígido**
$\boldsymbol{C_A()}$	**Matriz de Coriólis de la masa añadida**
$\boldsymbol{D()}$	**Matriz de amortiguamiento hidrodinámico**
$\boldsymbol{g()}$	**Vector de fuerzas hidrostáticas**
$\boldsymbol{A_c}$	**Matriz de fuerzas por cambio de dirección dentro de una corriente**
$\boldsymbol{\tau}$	**Fuerzas externas (propulsión)**
$\boldsymbol{\upsilon_r}$	**Vector de velocidad relativa del vehículo respecto al fluido**
$\boldsymbol{\upsilon_c}$	**Vector de velocidad de la corriente de agua por la que navega el vehículo**

A continuación se desarrollan cada uno de los términos descritos en la Tabla 3.

El vector de posición respecto de un sistema de referencia fijo a tierra se define como:

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

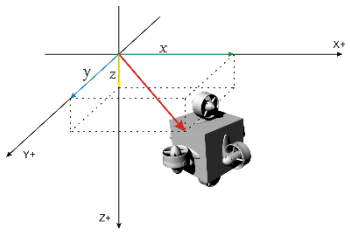


Ilustración 13 – Sistema de referencai ROV (II)

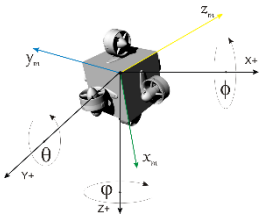


Ilustración 12 – Sistema de referencia ROV (I)

$X+, Y+, Z+$, corresponden a los ejes positivos del sistema de referencias fijo a tierra, y x_m, y_m, z_m corresponden a los ejes positivos del sistema de referencias fijo a ROV.

El vector de velocidad respecto del sistema de referencia fijo al vehículo será:

$$\boldsymbol{v} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}$$

Donde $\vec{v}_{SR} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$ son las velocidades del ROV en las direcciones x_m, y_m, z_m y $\vec{\omega}_{SR} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$ son las velocidades angulares respecto a esos mismos ejes.

La matriz de transformación de coordenadas tiene el siguiente aspecto:

$$J = \begin{bmatrix} cos\theta \cdot cos\psi & sin\phi \cdot sin\theta \cdot cos\psi - cos\phi \cdot sin\psi & cos\phi \cdot sin\theta \cdot cos\psi + sin\phi \cdot sin\psi & 0 & 0 & 0 \\ cos\theta \cdot sin\psi & sin\phi \cdot sin\theta \cdot sin\psi + cos\phi \cdot cos\psi & cos\phi \cdot sin\theta \cdot sin\psi - sin\phi \cdot cos\psi & 0 & 0 & 0 \\ -sin\theta & sin\phi \cdot cos\theta & cos\phi \cdot cos\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & sin\phi \cdot tan\theta & cos\phi \cdot tan\theta \\ 0 & 0 & 0 & 0 & cos\phi & -sin\phi \\ 0 & 0 & 0 & 0 & sin\phi \cdot sec\theta & cos\phi \cdot sec\theta \end{bmatrix}$$

La matriz de masas toma la forma:

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & Ix & -Ixy & -Ixz \\ mz_G & 0 & -mx_G & -Ixy & Iy & -Iyz \\ -my_G & mx_G & 0 & -Ixz & -Iyz & Iz \end{bmatrix}$$

Donde *m* es la masa del vehículo, $\vec{r}_G = \begin{bmatrix} x_G \\ y_G \\ z_G \end{bmatrix}$ es la posición de su C.D.G. respecto al sistema móvil, y la submatriz $TI = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix}$ es el denominado tensor de inercia.

La matriz de masa añadida es de la forma:

$$M_A = \begin{pmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{pmatrix}$$

Esta matriz se compone con una serie de coeficientes que dependen de la masa de fluido que arrastra el vehículo en su movimiento debido a su forma y que, al estar el vehículo completamente sumergido, podemos considerar constantes.

La matriz de Coriolis del sólido rígido se representa de la siguiente forma:

$$C_{SR}(\boldsymbol{v}) = \begin{pmatrix} m(\vec{\omega}_{SR} \times \vec{v}_{SR}) + m(\vec{\omega}_{SR} \times \vec{\omega}_{SR} \times \vec{r}_G) \\ \vec{\omega}_{SR} \times (TI \cdot \vec{\omega}_{SR}) + m(\vec{r}_G \times (\vec{\omega}_{SR} \times \vec{v}_{SR})) \end{pmatrix}$$

La matriz de Coriolis de la masa añadida se calcula como:

$$C_A(\boldsymbol{v})\boldsymbol{v} = \begin{pmatrix} \vec{\omega}_{SR} \times (M_{A1} \cdot \vec{v}_{SR}) + \vec{\omega}_{SR} \times (M_{A2} \cdot \vec{\omega}_{SR}) \\ \vec{v}_{SR} \times (M_{A1} \cdot \vec{v}_{SR}) + \vec{v}_{SR} \times (M_{A2} \cdot \vec{\omega}_{SR}) + \vec{\omega}_{SR} \times (M_{A3} \cdot \vec{v}_{SR}) + \vec{\omega}_{SR} \times (M_{A4} \cdot \vec{\omega}_{SR}) \end{pmatrix}$$

Donde se ha dividido la matriz de masa añadida en 4 submatrices:

$$M_A = \begin{pmatrix} M_{A1} & M_{A2} \\ M_{A3} & M_{A4} \end{pmatrix}$$

$$M_{A1} = \begin{pmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} \end{pmatrix} \quad M_{A2} = \begin{pmatrix} X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \end{pmatrix} \quad M_{A3} = \begin{pmatrix} K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} \end{pmatrix} \quad M_{A4} = \begin{pmatrix} K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{pmatrix}$$

La matriz de amortiguamiento hidrodinámico se compone de tres partes, el amortiguamiento lineal D_l, el no lineal $D_{nl}(\boldsymbol{v})$ y los amortiguamientos que se incrementan con la velocidad en cada una de las direcciones de movimiento D_u, D_v y D_w quedaría como:

$$D(\boldsymbol{v}) = D_l + u \cdot D_u + v \cdot D_v + w \cdot D_w D_{nl}(\boldsymbol{v})$$

$$D_l = -\begin{pmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{pmatrix}$$

$$D_u = -\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{uv} & Y_{uw} & Y_{up} & Y_{uq} & Y_{ur} \\ 0 & Z_{uv} & Z_{uw} & Z_{up} & Z_{uq} & Z_{ur} \\ 0 & K_{uv} & K_{uw} & K_{up} & K_{uq} & K_{ur} \\ 0 & M_{uv} & M_{uw} & M_{up} & M_{uq} & M_{ur} \\ 0 & N_{uv} & N_{uw} & N_{up} & N_{uq} & N_{ur} \end{pmatrix}$$

$$D_v = -\begin{pmatrix} X_{vu} & 0 & X_{vw} & X_{vp} & X_{vq} & X_{vr} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ Z_{vu} & 0 & Z_{vw} & Z_{vp} & Z_{vq} & Z_{vr} \\ K_{vu} & 0 & K_{vw} & K_{vp} & K_{vq} & K_{vr} \\ M_{vu} & 0 & M_{vw} & M_{vp} & M_{vq} & M_{vr} \\ N_{vu} & 0 & N_{vw} & N_{vp} & N_{vq} & N_{vr} \end{pmatrix}$$

$$D_w = -\begin{pmatrix} X_{wu} & X_{wv} & 0 & X_{wp} & X_{wq} & X_{wr} \\ Y_{wu} & Y_{wv} & 0 & Y_{wp} & Y_{wq} & Y_{wr} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ K_{wu} & K_{wv} & 0 & K_{wp} & K_{wq} & K_{wr} \\ M_{wu} & M_{wv} & 0 & M_{wp} & M_{wq} & M_{wr} \\ N_{wu} & N_{wv} & 0 & N_{wp} & N_{wq} & N_{wr} \end{pmatrix}$$

$$D_{nl}(\boldsymbol{v}) = -\begin{pmatrix} X_{|u|u}|u| & X_{|u|v}|u| & X_{|u|w}|u| & X_{|u|p}|u| & X_{|u|q}|u| & X_{|u|r}|u| \\ Y_{|v|u}|v| & Y_{|v|v}|v| & Y_{|v|w}|v| & Y_{|v|p|v|} & Y_{|v|q}|v| & Y_{|v|r}|v| \\ Z_{|w|u}|w| & Z_{|w|v}|w| & Z_{|w|w}|w| & Z_{|w|p}|w| & Z_{|w|q}|w| & Z_{|w|r}|w| \\ K_{|p|u}|p| & K_{|p|v}|p| & K_{|p|w}|p| & K_{|p|p}|p| & K_{|p|q}|p| & K_{|p|r}|p| \\ M_{|q|u}|q| & M_{|q|v}|q| & M_{|q|w}|q| & M_{|q|p}|q| & M_{|q|q}|q| & M_{|q|r}|q| \\ N_{|r|u}|r| & N_{|r|v}|r| & N_{|r|w}|r| & N_{|r|p}|r| & N_{|r|q}|r| & N_{|r|r}|r| \end{pmatrix}$$

Pr otra parte, el vector de fuerzas hidrostáticas $g(\boldsymbol{\eta})$ se calcula a partir de dos fuerzas que intervienen en la flotación del vehículo, la fuerza de la gravedad (W), que actúa sobre el centro de gravedad, y la fuerza de flotación (B), que actúa sobre el centro de carena. Ambas fuerzas tienen dirección vertical y se calculan según las expresiones:

$$W = m \cdot \mathrm{g}$$

$$B = -\rho_{agua} \cdot V \cdot \mathrm{g}$$

Siendo g la gravedad, ρ_{agua} la densidad del agua y *V* el volumen del vehículo.

Suponiendo que el centro de carena se encuentra ubicado respecto de nuestro sistema de referencia móvil en la posición $\vec{r}_C = \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}$, el vector de fuerzas hidrostáticas queda de la siguiente forma:

$$g(\boldsymbol{\eta}) = \begin{pmatrix} -(W+B) \cdot sen(\theta) \\ (W+B) \cdot \cos(\theta) \cdot sen(\phi) \\ (W+B) \cdot \cos(\theta) \cdot cos(\phi) \\ (y_G \cdot W - y_C \cdot B) \cdot \cos(\theta) \cdot \cos(\phi) - (z_G \cdot W - z_C \cdot B) \cdot \cos(\theta) \cdot sen(\phi) \\ -(z_G \cdot W - z_C \cdot B) \cdot \mathrm{sen}(\theta) - (x_G \cdot W - x_C \cdot B) \cdot \cos(\theta) \cdot cos(\phi) \\ (x_G \cdot W - x_C \cdot B) \cdot \cos(\theta) \cdot \mathrm{sen}(\phi) + (y_G \cdot W - y_C \cdot B) \cdot \mathrm{sen}(\theta) \end{pmatrix}$$

La matriz de fuerzas por cambio de dirección dentro de una corriente, expresa las fuerzas que aparecen en el vehículo cuando este intenta girar y al mismo tiempo se encuentra en una corriente de agua, su expresión depende del vector que define la velocidad de la corriente $\upsilon_c = \begin{bmatrix} u_c \\ v_c \\ w_c \end{bmatrix}$ y de los coeficientes de la masa añadida:

$$A_c(\upsilon_c)\boldsymbol{\upsilon}_r = M_A \cdot S_c(\vec{\omega}_{SR}) \cdot \upsilon_c$$

Donde $S_c(\vec{\omega}_{SR})$ se define como:

$$S_c(\vec{\omega}_{SR}) = \begin{pmatrix} 0 & -r & q & 0 & 0 & 0 \\ r & 0 & -p & 0 & 0 & 0 \\ -q & p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

El vector de velocidad relativa del vehículo respecto al fluido $\boldsymbol{v}_r$ viene definido por;

$$\boldsymbol{v}_r = \begin{bmatrix} \vec{v}_r \\ \vec{\omega}_{SR} \end{bmatrix} = \begin{bmatrix} u - u_c \\ v - v_c \\ w - w_c \\ p \\ q \\ r \end{bmatrix}$$

Por último, el vector τ de fuerzas externas queda definido por las fuerzas de los propulsores mediante la expresión:

$$\tau = \sum_{i=1}^{m} (F_{hi} \cdot V_{Dhi} + M_{hi})$$

Donde *m* es el número total de propulsores, y el subíndice *i* apunta a cada uno de los propulsores en concreto. Cada uno de los términos de la expresión anterior, para cada hélice se desarrolla según las siguientes ecuaciones:

$$F_h = K_{Ta} \cdot \rho_{agua} \cdot D_h^4 \cdot n \cdot |n|$$

$$M_h = \eta_h \cdot F_h \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vec{u}_h \end{pmatrix}$$

$$V_{Dh} = \begin{pmatrix} \vec{u}_h \\ \vec{r}_h \times \vec{u}_h \end{pmatrix}$$

$$K_{Ta} = \frac{\pi}{2}(k_1 \cdot J_o^2 + k_2 \cdot J_o + k_3)$$

$$J_o = \frac{(\vec{v}_r + \vec{\omega}_{SR} \times \vec{r}_h) \cdot \vec{u}_h}{D_h \cdot n}$$

Tabla 5- Descripción de términos de la ecuación que definen las fuerzas de propulsión y valores que toman para el ROV CUBO

Término	Descripción	Valor
D_h	Diámetro de la hélice en metros	0.075
n	Velocidad de la hélice en rad/s	-440 a 440
η_h	Rendimiento de la hélice	1
k_1 k_2 k_3	Coeficientes de la hélice	-10^{-10} $5 \cdot 10^{-8}$ $2.5 \cdot 10^{-3}$
$\vec{r}_h$	Vector con la posición del centro de la hélice respecto del sistema de referencias fijo al ROV	$\begin{matrix} 0 & 0.2315 & 0 \\ 0 & -0.2315 & 0 \\ 0 & 0 & 0.2315 \\ 0 & 0 & -0.2315 \\ 0.2315 & 0 & 0 \\ -0.2315 & 0 & 0 \end{matrix}$
$\vec{u}_h$	Vector unitario en la dirección del propulsión de la hélice cuando gira en sentido positivo, según el sistema de referencias fijo al ROV	$\begin{matrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{matrix}$

Como se aprecia en las ecuaciones (1) y (2), y en el desarrollo posterior de estas ecuaciones, para realizar el modelo del ROV no solo aparecen las características físicas principales del vehículo, y que se han mostrado en la Tabla 1, sino también una serie de coeficientes hidrodinámicos que son fundamentales para la simulación dinámica: amortiguamiento hidrodinámica y masa añadida.

La forma más exacta de obtener estos coeficientes es la realización de ensayos del vehículo en un canal de experimentación. Pero puesto que el ROV no se encuentra todavía totalmente construido, estos coeficientes se han tenido que obtener mediante simulación por CFD y regresión cuadrática (caso de la resistencia hidrodinámica) y mediante cálculo analítico de un objeto de forma parecida (caso de la masa añadida).

Para el cálculo mediante CFD se ha usado el módulo *Flow Simulation* de *SolidWorks®*.

Para comprobar la validez de los resultados, antes de simular con *CUBO*, se ha simulado con *Flow Simulation* otro vehículo submarino, concretamente el *REMUS 100* [1]. De éste vehículo ya se tenía información de sus coeficientes por dos fuentes diferentes. Los resultados obtenidos fueron muy satisfactorios y llevan a pensar que los valores obtenidos mediante los CFD para *CUBO* son cercanos a los reales.

En la figura 3 se muestra la distribución de velocidades del fluido para uno de los ensayos realizados con el *REMUS 100*.

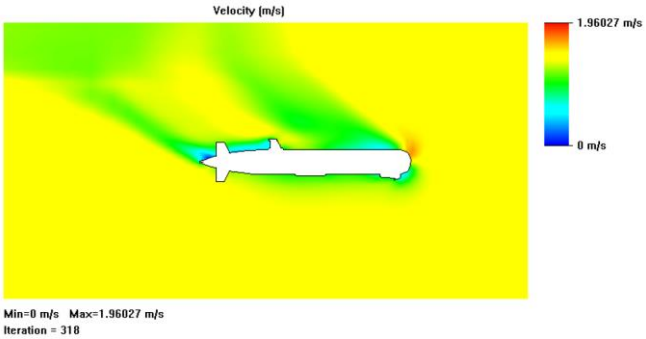


Ilustración 14 - Distribución de velocidades cruzadas en X y Z para el REMUS 100 a 1 m/s

En la figura 4 se muestra una comparativa de los coeficientes de amortiguamiento cruzado obtenidos para el *REMUS 100* mediante dos estudios por CFD (Azul-CFD de *SolidWorks®* y Naranja-Garrido CFD con Tdyn) y por métodos experimentales (Gris-Empíricas).

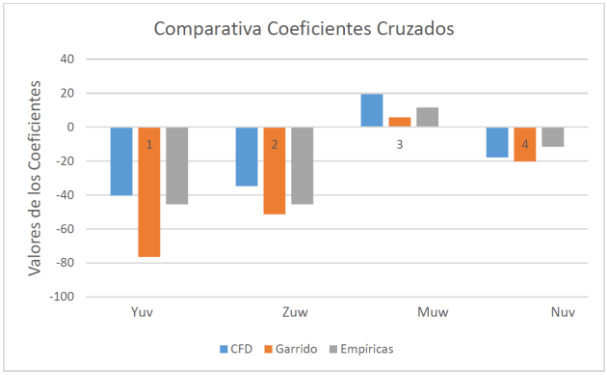


Ilustración 15 - Comparativa de resultados de coeficientes de amortiguación cruzados para el REMUS 100

Figura 4: Comparativa de resultados de coeficientes de amortiguación cruzados para el *REMUS 100*.

En la figura 5 se observa la distribución de velocidades obtenidas con *SolidWorks®* para uno de los ensayos realizados de forma simulada con *CUBO*, concretamente cuando este se mueve a 0.5m/s en la dirección X. En la Tabla 4 se presentan los coeficientes de amortiguamiento para *CUBO*.

Tabla 6 - Coeficientes de amortiguamiento hidrodinámico para CUBO

X_u	-0.134	*Kg/s*	Z_w	-0.1716	*Kg/s*
$X_{u\|u\|}$	-38.55	*Kg/m*	$Z_{w\|w\|}$	-41.314	*Kg/m*
Z_u	-0.009	*Kg/s*	M_w	0.0009	*Kg·m/s*
$Z_{u\|u\|}$	-0.082	*Kg/m*	$M_{w\|w\|}$	-0.0597	*Kg*
M_u	-0.015	*Kg·m/s*	K_p	-0.0161	*Kg·rad/s*
$M_{u\|u\|}$	-0.060	*Kg*	$K_{p\|p\|}$	-0.119	*Kgm²/rad²*
Y_v	0.1767	*Kg/s*	M_q	-0.01	*Kg·rad/s*
$Y_{v\|v\|}$	-44.49	*Kg/m*	$M_{q\|q\|}$	-0.1041	*Kgm²/rad²*
K_v	-0.019	*Kg·m/s*	Y_r	0.0112	*Kg·rad/s*
$K_{v\|v\|}$	0.1244	*Kg*	$Y_{r\|r\|}$	-0.0403	*Kgm²/rad²*
N_v	-0.015	*Kg·m/s*	N_r	0.0199	*Kg·rad/s*
$N_{v\|v\|}$	-0.028	*Kg*	$N_{r\|r\|}$	0.149	*Kgm²/rad²*
Y_u	0.1017	*Kg·rad/s*	Y_{uv}	-27.04	*Kgm²/rad²*
$Y_{u\|u\|}$	-0.316	*Kgm²/rad²*	N_{uv}	-1.3122	*Kgm²/rad²*
N_u	0.0293	*Kg·rad/s*	Z_{uw}	-7.9959	*Kgm²/rad²*
$N_{u\|u\|}$	0.0063	*Kgm²/rad²*	M_{uw}	0.1932	*Kgm²/rad²*

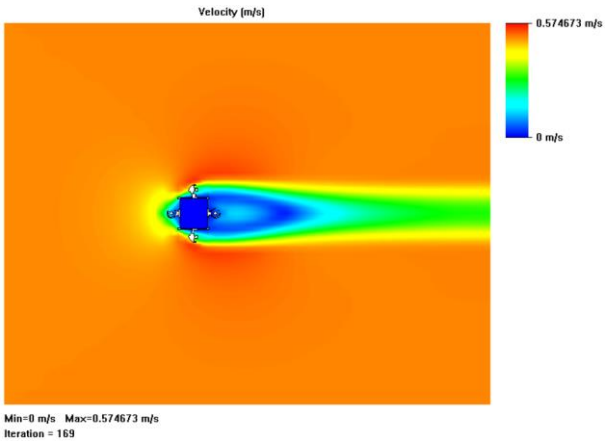


Ilustración 16 - Distribución de velocidades del ROV moviéndose a 0,5 m/s en el eje X.

Los coeficientes de la masa añadida no ha sido posible calcularlos mediante CFD con los softwares disponibles por lo que se van a estimar mediante la formulación analítica disponible suponiendo que el vehículo tiene una forma de elipsoide, y teniendo en cuenta la simetría en las tres direcciones espaciales.

La aproximación de un vehículo con forma de cubo a un elipsoide puede parecer algo burda, pero se aproxima bastante en los ejes Y e Z, de los cuales extrapolamos los resultados para el eje X. En cualquier caso, el mejor método de cálculo de estas componentes es de manera experimental, lo cual se podrá hacer una vez construido el modelo.

Las ecuaciones usadas son:

$$Y_{\dot{v}} = Z_{\dot{w}} = \frac{\beta_0}{2-\beta_0} \cdot m$$

$$M_{\dot{q}} = N_{\dot{r}} = -\frac{1}{5} \cdot \frac{(b^2-a^2)^2 \cdot (\alpha_0 - \beta_0)}{2 \cdot (b^2-a^2) + (b^2+a^2) \cdot (\beta_0 - \alpha_0)} \cdot m$$

Siendo α_0 y β_0

$$\alpha_0 = \frac{2 \cdot (1-e^2)}{e^3} \cdot \left(\frac{1}{2} \cdot \ln\left(\frac{1+e}{1-e} \right) - e \right)$$

$$\beta_0 = \frac{1}{e^2} - \frac{1-e^2}{2 \cdot e^3} \cdot \ln\left(\frac{1+e}{1-e} \right)$$

Con

$$e = 1 - \left(\frac{b}{a} \right)^2$$

Introduciendo las características básicas del ROV se obtiene la siguiente matriz de masa añadida:

$$M_A = \begin{pmatrix} 5.493 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.493 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5.493 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01086 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01086 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01086 \end{pmatrix}$$

En la figura se muestra un diagrama de bloques de forma general que ayudara a describir la estructura del sistema desarrollado

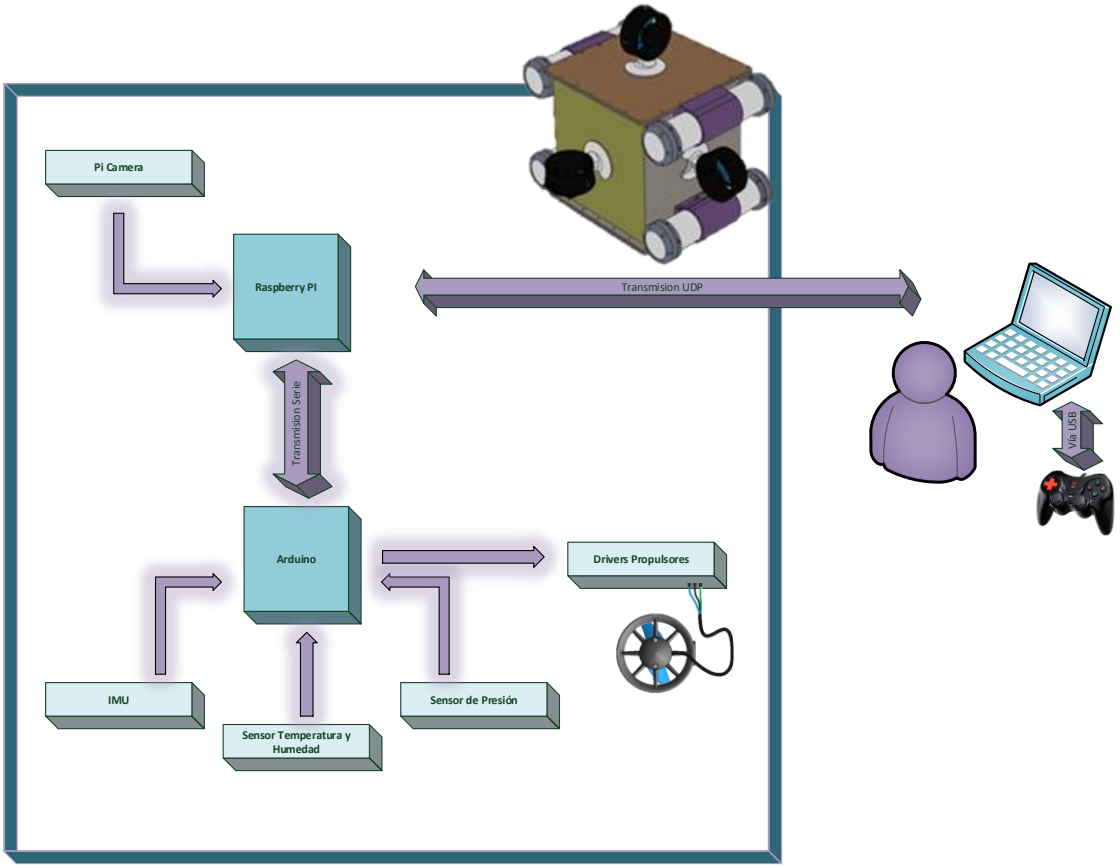


Ilustración 17 - Diagrama de bloques del sistema propuesto

Se muestran dos bloques principales, son Raspberry Pi y Arduino. El primero de ellos se podría calificar como el cerebro del sistema encargado de procesar la información adquirida, realizar numerosos cálculos, coordinar el resto de subsistemas y comunicar con el exterior o planta. El bloque Arduino no resta de importancia, es el encargado de adquirir la información del entorno a través de una serie de sensores, así como la de controlar la señales de control que se envían hacia los propulsores del vehículo. Ambos bloques principales realizan comunicación dúplex a través de un puerto serie, el bloque Raspberry Pi se comunica con planta vía UDP.

Los sensores y el resto de subsistemas se detallan en el capítulo 4.

Capítulo 4
Herramientas Hardware y Software

En este capítulo se abordan los dispositivos electrónicos empleados, tanto placas de desarrollo basadas en microcontrolador y microprocesador como sensores y sus características; herramientas de software y los lenguajes de programación empleados. Se finaliza detallando como se realiza la calibración del magnetómetro y el acelerómetro.

Arduino

Según Massimo Banzi, uno de sus creadores, es una plataforma de computación en código abierto basada en una simple placa de entradas y salidas simples (I/O) y un entorno de desarrollo que implementa el lenguaje de procesamiento.

Ilustración 18 – Massimo Banzi

Arduino nació en el *Ivrea Interaction Design Institute* como una herramienta fácil para el prototipado rápido, dirigido a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de simples placas de 8 bits a productos para aplicaciones IoT, portátiles, impresión 3D y entornos incrustados.

Arduino también simplifica el proceso de trabajo con los microcontroladores, pero ofrece algunas ventajas para los profesores, estudiantes y aficionados interesados en otros sistemas:

- **Económico**

  Los tableros Arduino son relativamente baratos en comparación con otras plataformas de microcontroladores. La versión menos costosa del módulo Arduino se puede montar a mano, e incluso los módulos Arduino pre-ensamblados pueden llegar a costar menos de 5€
- **Cross-platform**

  El software Arduino (IDE) se ejecuta en sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los sistemas de microcontroladores están limitados a Windows.
- **Entorno de programación sencillo y claro**

  El software Arduino (IDE) es fácil de usar para los principiantes, pero lo suficientemente flexible como para que los usuarios avanzados también se beneficien. Para los

profesores, está convenientemente basado en el entorno de programación de Procesamiento, por lo que los estudiantes que aprenden a programar en ese entorno estarán familiarizados con el funcionamiento del IDE de Arduino.

- **Software de código abierto y extensible**

  El software Arduino se publica como herramientas de código abierto, disponibles para la extensión por programadores experimentados. El lenguaje se puede expandir a través de las bibliotecas C y las personas que quieran comprender los detalles técnicos pueden dar el salto desde Arduino al lenguaje de programación AVR C en el que se basa. Del mismo modo, puede agregar el código AVR-C directamente a sus programas Arduino si lo desea.

- **Código abierto y hardware extensible**

  Los esquemas de las placas Arduino se publican bajo una licencia Creative Commons, por lo que los diseñadores de circuitos experimentados pueden hacer su propia versión del módulo, ampliarla y mejorarla. Incluso los usuarios relativamente inexpertos pueden construir la versión de paneles del módulo con el fin de entender cómo funciona y ahorrar dinero.

Todas las placas de Arduino son totalmente de código abierto, lo que permite a los usuarios construirlas independientemente y eventualmente adaptarlas a sus necesidades particulares. El software, también, es de código abierto, y está creciendo a través de las contribuciones de los usuarios de todo el mundo.

Raspberry Pi 2B

Raspberry Pi, es un es un ordenador de tamaño de tarjeta de crédito que se conecta a un televisor o monitor y un teclado y/o mouse. Es una placa que soporta varios componentes necesarios en un ordenador común. Es un pequeño ordenador, que puede ser utilizado por muchas de las tareas que un PC de escritorio es capaz de realizar.

Ilustración 19 - Raspberry Pi 2

Raspberry Pi 2 B representa un gran aumento de rendimiento con respecto a sus antecesores basados en un núcleo. Este modelo es hasta seis veces más rápido puesto que está basado en un procesador Cortex-A7 de cuatro núcleos. Ofrece 1 GB de memoria RAM para las aplicaciones con más requisitos de memoria o cálculo. El núcleo del sistema operativo se ha actualizado para aprovechar al máximo la última tecnología ARM Cortex-A7.

Algunas de sus características

- Procesador Broadcom BCM2836 de 900 MHz ARM Cortex-A7 de cuatro núcleos con GPU VideoCore IV de doble núcleo
- GPU proporciona una tecnología Open GL ES 2.0, hardware acelerado OpenVG y admite imágenes de alta resolución 1080p30 H.264
- GPU tiene una capacidad de 1 Gpixel/s, 1,5 Gtexel/s o 24 GFLOPs con filtrado e infraestructura DMA
- SDRAM LPDDR2 de 1 GB
- Salida de vídeo: HD 1080p
- Salida de vídeo compuesto (PAL/NTSC)
- Salida de audio estéreo
- Conector hembra Ethernet RJ45 10/100 BaseT
- Conector hembra de vídeo/audio HDMI 1.3 y 1.4
- Conector hembra de salida de vídeo compuesto/audio de 3,5 mm 4 polos
- 4 conectores hembra USB 2.0

- Conector MPI CSI-2 de 15 vías para cámara de vídeo HD Raspberry Pi (775-7731)
- Conector de interfaz serie de display de 15 vías
- Conector para tarjeta MicroSD
- Conector macho de 40 pines para buses serie y GPIO (compatible con el conector macho de 26 pines Raspberry Pi 1)
- Fuente de alimentación: +5 V a 2 A a través de conector hembra microUSB
- Dimensiones: 86 x 56 x 20 mm

Pi Camera

La placa de cámara RPI se conecta directamente al conector CSI de la Raspberry Pi. Es capaz de generar una imagen nítida con resolución de 5MP o grabar vídeo HD de 1080p a 30fps con la última versión v1.3. La placa cuenta con un sensor Omnivision 5647 de 5MP (2592 × 1944 píxeles) en un módulo de enfoque fijo. El módulo se conecta mediante un cable plano de 15 pines a la interfaz serie para cámaras (CSI) de 15 pines de la Raspberry Pi, diseñada especialmente para conectarle cámaras. El bus CSI bus es capaz de alcanzar velocidades de datos extremadamente altas y traslada datos de píxeles exclusivamente al procesador BCM2835.

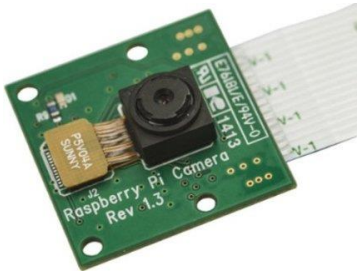


Ilustración 20 - Pi Camera

- Compatibilidad con todos los modelos de Raspberry Pi 1 y 2
- Módulo de cámara Omnivision 5647 de 5MP
- Resolución de imagen inmóvil de 2592 x 1944
- Admite grabación de vídeo de 1080p a 30fps, 720p a 60fps y 640x480p 60/90
- Interfaz serie para cámaras MIPI de 15 pines para conexión directa a la placa Raspberry Pi
- Tamaño de 20mm x 25mm x 9mm
- Peso de 3 g

Actualmente el modulo Pi Camera v2 ha reemplazado al anterior. Pi Camera v2 posee un sensor Sony IMX219 de 8MP en comparación con el de 5MP.

AltIMU 10 v4

Pololu AltIMU 10 v4 es una unidad de medición inercial (IMU) y altímetro que cuenta con el mismo acelerómetro L3GD20H y LSM303D y el magnetómetro como MinIMU-9 v3, y agrega un barómetro digital LPS25H. Una interfaz I²C tiene acceso a diez mediciones independientes de presión, rotación, aceleración y magnéticas que pueden usarse para calcular la altitud del sensor y la orientación absoluta. El módulo trabaja con una tensión de 2.5 a 5.5 V y tiene un espacio entre pines de 0.1

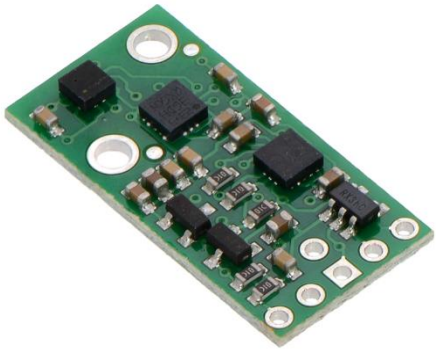


Ilustración 21 - AltIMU 10 v4

Los circuitos integrados LPS25H, L3GD20H y LSM303D tienen muchas opciones configurables, incluyendo resoluciones seleccionables para el barómetro y sensibilidad seleccionable dinámicamente para el giroscopio, el acelerómetro y el magnetómetro. Cada sensor también tiene una selección de velocidades de datos de salida. A los tres se puede acceder a través de una interfaz I²C compartida, permitiendo que los sensores se aborden individualmente a través de una sola línea de reloj y una única línea de datos. Además, permite a los usuarios cambiar las direcciones de los esclavos y tener dos AltIMUs conectados en el mismo bus I²C.

Las nueve lecturas independientes de rotación, aceleración y magnética proporcionan todos los datos necesarios para hacer un sistema de referencia de actitud y rumbo AHRS, y las lecturas del sensor de presión absoluta se pueden convertir fácilmente en altitudes, lo que le da un total de diez mediciones independientes, a veces llamado 10DOF. Con un algoritmo apropiado, un microcontrolador u ordenador se pueden utilizar los datos para calcular la orientación y la altura de la placa AltIMU. El giroscopio puede utilizarse para rastrear la rotación de forma muy precisa en una escala de tiempo corta, mientras que el acelerómetro y la brújula pueden ayudar a compensar la desviación del giroscopio con el tiempo proporcionando un marco de referencia absoluto. Los respectivos ejes de los dos chips están alineados en la placa para facilitar estos cálculos de fusión de sensores.

Tabla 7 - Característica del módulo AltIMU 10 v4

Dimensiones	25mm x 13mm x 3mm
Tensión de trabajo	2.5 a 5.5V

Corriente de trabajo	6 mA
Formato de salida	I^2C
Sensibilidad giroscopio	±245, ±500, ±2000 °/s
Sensibilidad acelerómetro	±2, ±4, ±8, ±16 g
Sensibilidad magnetómetro	±2, ±4, ±8, ±12 gauss
Sensibilidad barómetro	260 a 1260mbar

Sensor de presión DIGITEN PS-25

Ilustración 22 - Sensor de Presión

Tensión de trabajo: 5VDC

G1/2'' Ø20.8mm

Tensión de salida: 0.5 - 4.5 VDC

Material: Aleación de acero al carbono

Corriente de trabajo ≤ 10mA

Rango de presión de medida: 0 – 2.5 MPa

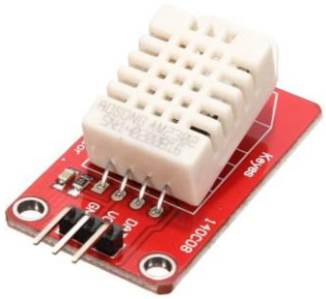
Tiempo de respuesta ≤ 2.0 ms

Error de medida ±1.5% FSO

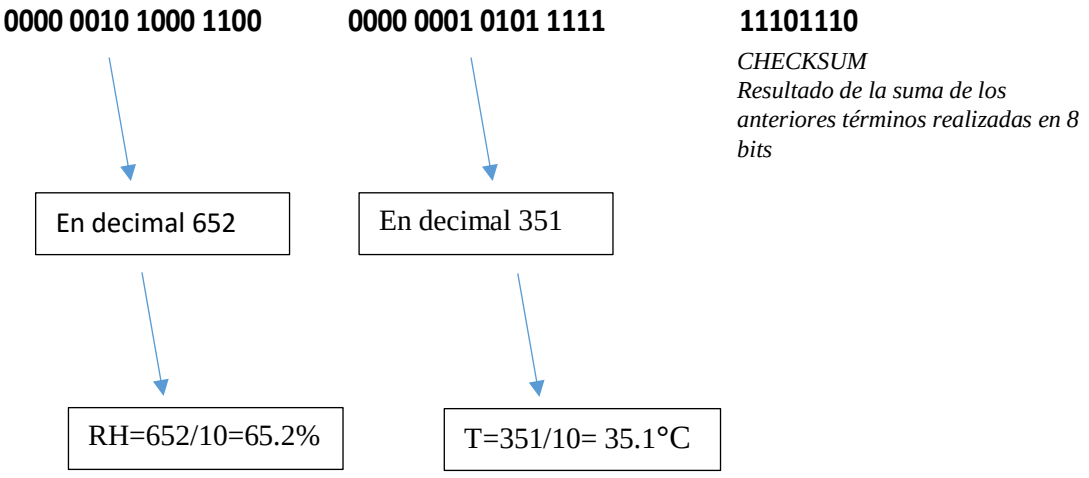
Sensores de temperatura y humedad AM2302 DHT22

Tabla 8- Características del sensor de humedad y temperatura

Modelo	AM2302
Tensión de trabajo	3.3-5.5V DC
Señal de salida	Señal digital vía 1-wire bus
Sensor	Polímero condensador de humedad
Rango de operación	Humedad 0-100%RH Temperatura -40-80 °C
Precisión	Humedad ±2%RH Temperatura ±0.5 °C
Resolución	Humedad ±1%RH Temperatura ±0.2°C
Histéresis de humedad	±0.3%RH
Estabilidad (largo plazo)	±0.5%RH/año



La trama transmitida está compuesta por 16 bits que componen los datos RH, 16 bits de datos temperatura y 8 bits de checksum. A continuación, se muestra un ejemplo de una trama transmitida:



**Cuando el bit más significativo de temperatura es 1, significa que la medida es bajo cero*

AFRO ESC 30A

Basic ESC es un controlador de velocidad simple, preprogramado con firmware personalizado que permite el funcionamiento en ambos sentidos de giro. El firmware es de código abierto y está disponible en la web del fabricante

Tabla 9 - Especificaciones y característica de AFRO ESC 30A

Tabla de Especificaciones	
Eléctricas	
Voltaje	6-16.8V
Corriente Máxima	30 A
Físicas	
Longitud	50mm
Ancho	25mm
Altura	11mm
Conectores de Potencia	Macho 3.3mm (tipo bullet)
Conectores de Motor	Hembra 3.5mm (tipo bullet)
Conector de Señal	Conector de 3 pines (0.1'' pitch)
Señal Pulse Width	
Voltaje	3.3-5V
Tasa máxima de actualización	400 Hz
Parada	1500 us
Máximo sentido adelante	1900 us
Máximo sentido atrás	1100 us
Banda de histéresis	±25 us (centrado en torno a 1500 us)

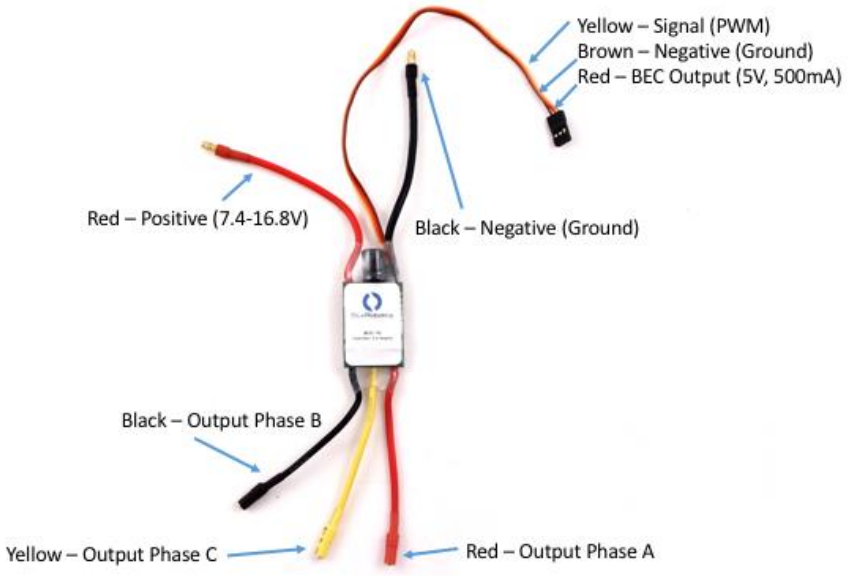


Ilustración 23 - Diagrama de conexiones con Arduino y Propulsores T100

C#

Pronunciado C Sharp, es actualmente, junto al lenguaje de programación Java, uno de los más utilizados, además está disponible para el desarrollo de aplicaciones de propósito general, aplicaciones que muestren una interfaz gráfica, aplicaciones para internet y aplicaciones para dispositivos móviles.

En los últimos años, C y C++ han sido los lenguajes más utilizados en el desarrollo de aplicaciones. Otros lenguajes como Visual Basic ofrecen, además de facilidad, una elevada productividad en el desarrollo de aplicaciones, con el inconveniente de sacrificar la flexibilidad que C y C++ nos ofrece. La solución que Microsoft da a este problema es el lenguaje de programación C#, un lenguaje moderno orientado a objetos que junto a la plataforma Microsoft .NET se caracteriza por proporcionar utilidades y servicios para obtener un elevado provecho tanto de la informática como de las comunicaciones.

Microsoft .NET extiende las ideas de Internet y sistema operativo haciendo de la propia Internet la base de un nuevo sistema operativo. En última instancia, esto permitirá a los desarrolladores crear programas que transciendan los límites de los dispositivos y aprovechen por completo la conectividad de Internet y sus aplicaciones. Para ello proporciona una plataforma que incluye los siguientes componentes básicos:

- Herramientas de programación para crear servicios Web XML con soporte multilingüe: Visual Studio .NET y .NET Framework.

- Infraestructura de servidores, incluyendo Windows y .NET Enterprise Servers.

Un conjunto de servicios que actúan como bloques de construcción para el sistema operativo de Internet que incluirán autentificación del usuario (servicio Passport .NET), servicios para almacén de datos, administración de preferencias de los usuarios, calendario, y otros muchos.

Visual Studio 2015

Microsoft Visual Studio 2015 es un conjunto de herramientas para crear software, desde la fase de diseño pasando por la fases de diseño de la interfaz de usuario, codificación, pruebas, depuración, análisis de la calidad y el rendimiento del código, implementación en los clientes y recopilación de telemetría de uso. Estas herramientas están diseñadas para trabajar juntas de la forma más eficiente posible y todas se exponen a través del Entorno de desarrollo integrado (IDE) de Visual Studio.

De forma predeterminada, Visual Studio proporciona compatibilidad con C#, C y C++, JavaScript, F # y Visual Basic. Visual Studio funciona y se integra bien con aplicaciones de terceros como Unity y Apache Cordova.

La siguiente imagen muestra el IDE de Visual Studio con un proyecto abierto, la ventana Explorador de soluciones para desplazarse por los archivos de proyecto y la ventana de Team Explorer para navegar por el código fuente y controlar los elementos de trabajo.

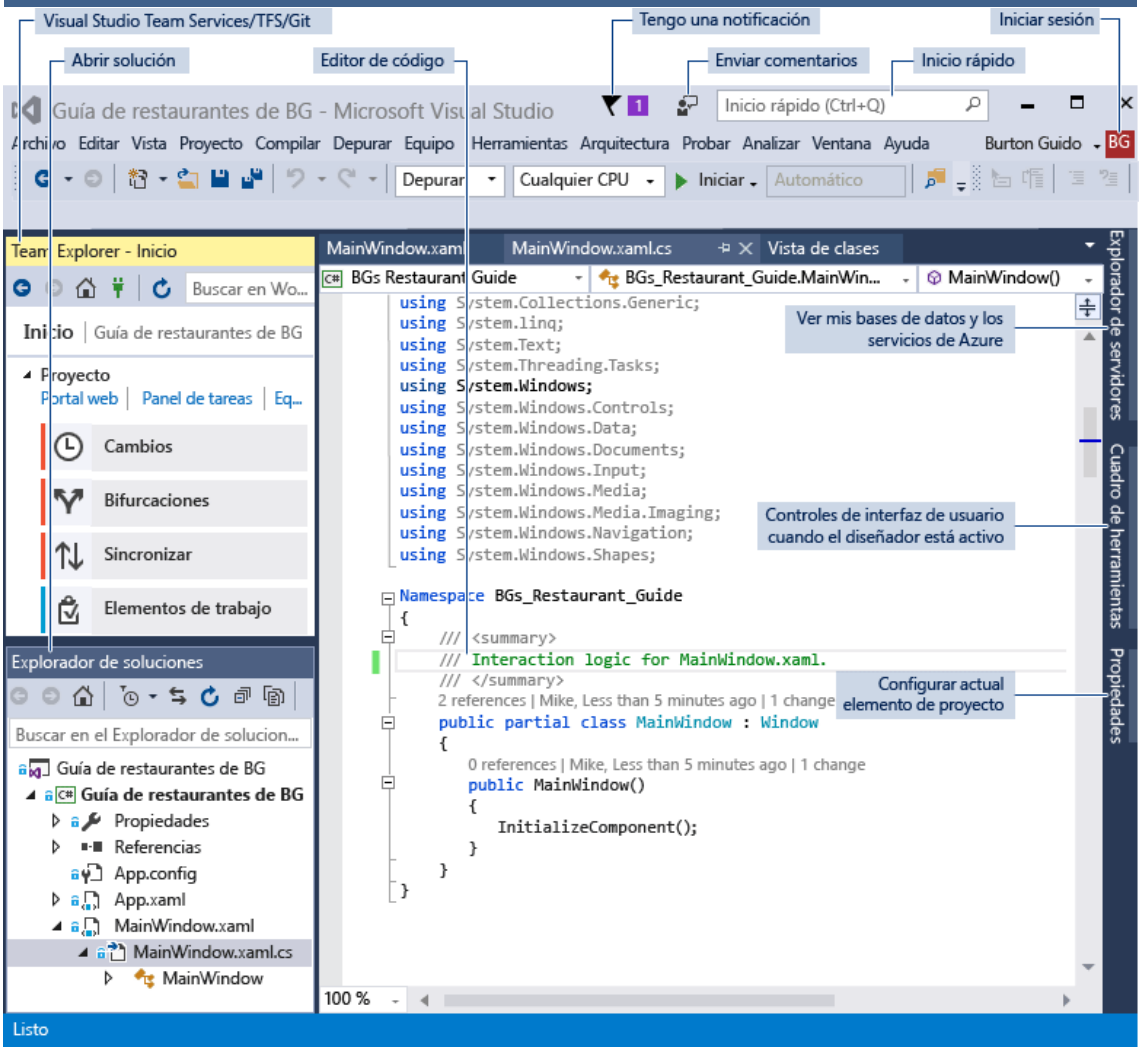
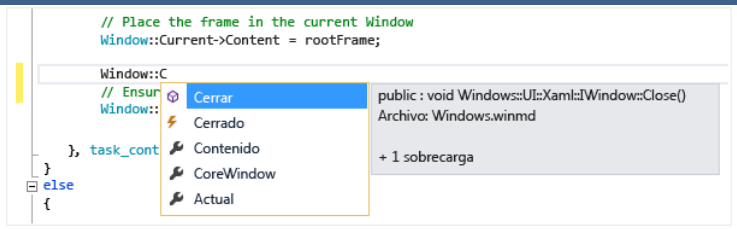


Ilustración 24 - Microsoft Visual Studio

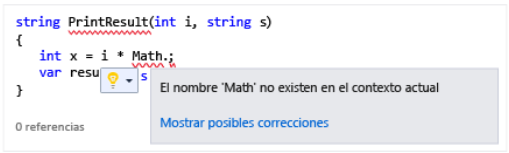
El editor de texto es sumamente interactivo con muchas características de productividad que le ayudarán a escribir código mejor y más rápidamente. Las características varían según el lenguaje y no tiene que usar todas ellas. Escriba "Editor" en Inicio rápido para activar o desactivar características. Algunas de las características de productividad más comunes son:

- **Refactorización** incluye operaciones tales como el cambio inteligente de nombre de las variables, mover líneas seleccionadas de código a una función diferente, mover código a otras ubicaciones, reordenar los parámetros de una función y mucho más.
- ***IntelliSense*** es un término que aglutina un conjunto de características muy populares que muestran información escritura sobre el código directamente en el editor y, en algunos casos, escriben pequeños fragmentos de código automáticamente.

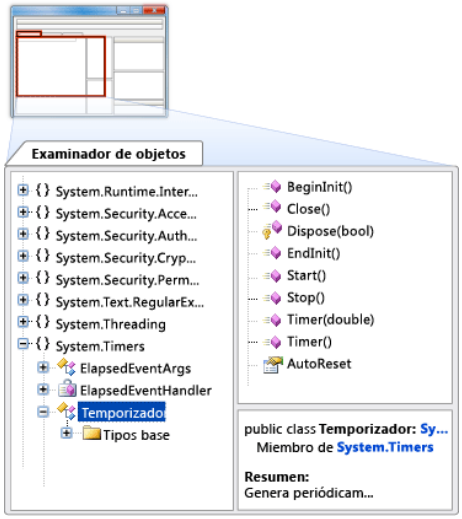


Básicamente, IntelliSense es como tener documentación básica insertada en el editor, lo que evita tener que buscar información de escritura en una ventana de ayuda independiente.

- Los **subrayados ondulados** le avisan de errores o posibles problemas en el código en tiempo real a medida que escribe, lo que permite corregirlos inmediatamente sin esperar a que el error se detecte en tiempo de compilación o de ejecución



- Los **marcadores** permiten ir rápidamente a líneas específicas en los archivos en los que está trabajando activamente.
- En el menú contextual del editor de texto, puede invocar la ventana **Jerarquía de llamadas** para mostrar los métodos que llaman a y son llamados por el método situado por debajo del símbolo de intercalación.
- **Code Lens** permite buscar referencias y cambios en el código, errores vinculados, elementos de trabajo, revisiones de código y pruebas unitarias, todo sin salir del editor.
- La ventana **Ojear la definición** muestra un método o definición de tipo en línea, sin salir del contexto actual.
- La opción de menú contextual **Ir a definición** le lleva directamente al lugar donde se definen la función o el objeto.
- Una herramienta relacionada, el **Examinador de objetos**, permite inspeccionar ensamblados .NET o Windows en tiempo de ejecución en el sistema para ver qué tipos contienen y qué métodos y propiedades contienen esos tipos.



Lenguaje de programación en Arduino

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen declaraciones, estamentos o instrucciones.

```
void setup() //Primera Parte
{
  estamentos;
}
void loop() //Segunda Parte
{
  estamentos;
}
```

En donde **setup()** es la parte encargada de recoger la configuración y **loop()** es la que contiene el programa que se ejecutará cíclicamente. Ambas funciones son necesarias en el programa.

La función de configuración **setup** debe contener la declaración de las variables. Es la primera función a ejecutar en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar el modo de trabajo de las entradas y salidas **pinMode,** configuración de la comunicación en serie y otro tipo de características.

La función bucle **loop** contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc.) Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

Arduino está basado en C y soporta todas las funciones del estándar C y algunas de C++.

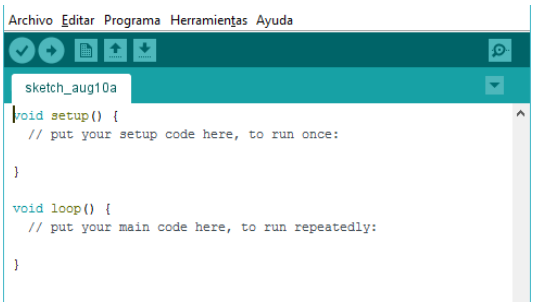


Ilustración 25 - Entorno Arduino

Funciones

Existen multitud de funciones que implementa el lenguaje de programación de Arduino, entre las más utilizadas se encuentran:

pinMode(pin, mode)

Función usada para configurar un pin dado para comportarse como INPUT u OUTPUT

digitalRead(pin)

Lee el valor desde un pin digital específico. Devuelve un valor HIGH o LOW. El pin puede ser especificado con una variable o una constante (0-13).

digitalWrite(pin, value)

Introduce un nivel alto (HIGH) o bajo (LOW) en el pin digital especificado. De nuevo, el pin puede ser especificado con una variable o una constante 0-13.

analogRead(pin)

Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT u OUTPUT.

analogWrite(pin, value)

Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM. Esta función está activa para los pines 3, 5, 6, 9, 10, 11. Puede especificarse un valor de 0 – 255. delay(ms). Realiza una pausa en el programa la cantidad de tiempo en milisegundos especificada en el parámetro (máximo 1000, mínimo 1).

millis()

Devuelve el tiempo en milisegundos que lleva la placa Arduino ejecutando el programa actual.

min(x,y)

Devuelve el mínimo respectivamente de entre sus parámetros.

max(x,y)

Devuelve el máximo respectivamente de entre sus parámetros.

Serial.begin (rate)

Abre un Puerto serie y especifica la velocidad de transmisión. La velocidad típica para comunicación con el ordenador es de 9600 aunque se pueden soportar otras velocidades.

Serial.println(data)

Imprime datos al puerto serie seguido por un retorno de línea automático. Este comando tiene la misma forma que **Serial.print()** pero este último sin el salto de línea al final. Este comando puede emplearse para realizar la depuración de programas.

Serial.begin(rate)

Configura la velocidad del puerto serie

Serial.read()

Lee o captura un byte desde el puerto serie. Devuelve -1 si no hay ningún carácter en el puerto serie.

Serial.available()

Devuelve el número de caracteres disponibles para leer desde el puerto serie.

Interrupciones en Arduino

Arduino dispone de dos tipos de eventos en los que definir interrupciones. Por un lado tenemos las **interrupciones de timers** y por otro lado, tenemos las **interrupciones de hardware,** que responden a eventos ocurridos en ciertos pines físicos.

Dentro de las interrupciones de hardware, que son las que nos ocupan en este apartado, Arduino es capaz de detectar los siguientes eventos.

- RISING, ocurre en el flanco de subida de LOW a HIGH.
- FALLING, ocurre en el flanco de bajada de HIGH a LOW.
- CHANGING, ocurre cuando el pin cambia de estado (rising + falling).
- LOW, se ejecuta continuamente mientras está en estado LOW.

Los pines susceptibles de generar interrupciones varían en función del modelo de Arduino.

Tabla 10- Interrupciones en Arduino

Modelo	INT0	INT1	INT2	INT3	INT4	INT5
UNO	2	3				
Nano	2	3				
Mini	2	3				
Mega	2	3	21	20	19	18

La función asociada a una interrupción se denomina **ISR** *(Interruption Service Routines)* y, por definición, tiene que ser una función que no recibe nada y no devuelva nada.

Dos ISR no pueden ejecutarse de forma simultánea, en caso de dispararse otra interrupción mientras se ejecuta una ISR, la función ISR se ejecuta una a continuación de otra.

Durante la ejecución de una interrupción Arduino no actualiza el valor de la función *millis* y *micros*, es decir, el tiempo de ejecución de la ISR no se contabiliza y Arduino tiene un desfase en la medición del tiempo.

Mono

Raspberry Pi es capaz de ejecutar una distribución completa de Linux, entonces su potencial como herramienta de desarrollo es enorme. No sólo apoya los principales lenguajes compilados como C, C ++ y Java, sino que también soporta lenguajes de scripting populares como Python. También es capaz de compilar lenguajes tales como Google's Go Language y C#. El uso de este último gracias al proyecto *open source* Mono.

Mono es un conjunto de herramientas (incluyendo un compilador C # y Common Language Runtime) utilizado para crear programas compatibles con ".NET" basados en los estándares ECMA publicados. En esencia, le permite compilar y ejecutar código C # en Linux, siendo los ejecutables resultantes totalmente compatibles con Microsoft.NET.

Instalación de Mono en Raspbian

Para llevar a cabo la instalación de Mono desde la consola:

```
sudo apt-get install mono-complete
```

También es posible escribir programas GUI usando GTK. Pero primero debes instalar los enlaces entre Mono y GTK:

```
sudo apt-get install gtk-sharp2
```

Mono tiene un proyecto hermano IDE de la plataforma cruzada, conocido como **MonoDevelop** que facilita la escritura de aplicaciones de escritorio y de Web ASP.NET en Linux, Windows y Mac OSX. También está disponible para Raspberry Pi.

```
sudo apt-get install monodevelop
```

Herramientas para la calibración de los sensores de posición y movimiento

Dos parámetros fundamentales en navegación son la posición real y la dirección del movimiento.

Desde el origen de los tiempos se han utilizado gran variedad de herramientas para identificar dichos parámetros, desde piedras o montañas para recordar e identificar durante los viajes que realizaban, pasando por objetos celestes que les permitieran orientarse incluso herramientas de orientación creadas por el hombre.

Un elemento natural como el campo magnético de la tierra, ha sido utilizado durante siglos para estimar la orientación en función de los polos magnéticos Norte y Sur. Una de las primeras brújulas fueron inventadas por los chinos hace miles de años, hechas de un recipiente lleno de agua, usado como una plataforma niveladora, y una piedra magnética colocada en un plato flotando en el agua.

Gracias a los avances en la fabricación de Circuitos Integrados (IC) y la tecnología de sensores, los magnetómetros son hoy en día miniaturizados e integrados en dispositivos portátiles. Sin embargo, las mediciones del campo magnético obtenidas con sensores de bajo coste están corrompidas por varios errores debidos a problemas de fabricación de sensores y desviaciones magnéticas inducidas por el propio hardware donde está implementado. Por tanto, la calibración de los magnetómetros es necesaria para lograr mediciones de alta precisión.

Análisis de errores en magnetómetros

1. ***Debidos a la fabricación***

   Errores de instrumentación, que pueden considerarse únicos y constantes para un magnetómetro específico.

   Factor de escala, se debe al error correspondiente a los coeficientes de proporcionalidad que relacionan la entrada con la salida.

   No ortogonalidad.

   El offset del sensor puede introducir un sesgo en la salida.

2. *Debidos al entorno*

Desviación magnética, debido a elementos ferromagnéticos en el hardware donde esta implementado el sensor. Está compuesta por un magnetismo permanente y otros inducidos conocidos como “hard iron effect” y “soft iron effect”.

Hard Iron Effect, es el resultado de los imanes permanentes y la histéresis magnética, es decir, la remanencia de materiales de hierro magnetizados y es equivalente a un sesgo.

Soft Iron Effect, es debido a la interacción de compuestos ferromagnéticos con un campo externo inducido. Como consecuencia se produce un cambio de la intensidad y de la dirección del campo detectado.

El software ***Magneto*** encuentra el elipsoide que mejor se ajusta a las medidas del magnetómetro haciendo uso del método conocido como "*Li's ellipsoid specific fitting algorithm".*

Un elipsoide puede ser determinado por su ecuación general:

$$\boxed{ax^2 + By^2 + Cz^2 + 2Dxy + 2Exy + 2Fyz + 2Gx + 2Hy + 2Iz + J = 0}$$

En forma matricial:

$$X = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad X^T = [x, y, z, 1] \qquad Q = \begin{bmatrix} A & D & E & G \\ D & B & F & H \\ E & F & C & I \\ G & H & I & j \end{bmatrix}$$

Donde ***(x, y, z)*** son las coordenadas de un punto, tal que:

$$X^T Q X = 0$$

El centro del elipsoide puede ser calculado a través de vector:

$$b = -Q^{-1}u$$

Los semi-ejes del elipsoide ***a, b y c***, son obtenidos mediante:

$$a = \left(\frac{b^T Q b - J}{\lambda_1}\right)^{\frac{1}{2}} \qquad b = \left(\frac{b^T Q b - J}{\lambda_2}\right)^{\frac{1}{2}} \qquad c = \left(\frac{b^T Q b - J}{\lambda_3}\right)^{\frac{1}{2}}$$

$$Donde\ \lambda_1, \lambda_2\ y\ \lambda_3 son\ los\ valores\ propios\ de\ Q$$

Los correspondientes *vectores propios normalizados* $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ y $\boldsymbol{v_3}$ asociados a la matriz Q describen la dirección del *eje principal* del elipsoide. La matriz V formada a partir de los vectores columna v_1, v_2 y v_3 determina la *matriz de rotación* que describe la orientación del elipsoide.

Asignación de un elipsoide general a una esfera de radio R centrada en el origen

En primer lugar se realiza una traslación del elipsoide para que su centro coincida con el origen, el punto x se hace x_1:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - B = \begin{bmatrix} x - b_x \\ y - b_y \\ z - b_z \end{bmatrix}$$

Alineación del eje principal del elipsoide con los ejes X, Y y Z, el punto x_1 se hace x_2

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = V^T \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = V^{-1} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

Escalado de la longitud del semieje *a*, transformando el elipsoide en una ***esfera de radio a***, el punto x_2 se hace x_3

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{a}{b} & 0 \\ 0 & 0 & \frac{a}{c} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{\frac{\lambda_2}{\lambda_1}} & 0 \\ 0 & 0 & \sqrt{\frac{\lambda_3}{\lambda_1}} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \sqrt{\frac{1}{\lambda_1}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 \\ 0 & 0 & \sqrt{\lambda_3} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

Rotación de la esfera hacia la orientación del elipsoide por medio de la matriz de rotación **V**, el punto x_3 se hace x_4

$$\begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} = V \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

Cambio del radio de la esfera de **a** a **R**, el punto x_4 se hace x_{map}

$$\begin{bmatrix} x_{map} \\ y_{map} \\ z_{map} \end{bmatrix} = \frac{R}{a} \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix}$$

En resumen:

$$\left(\begin{bmatrix} x_{map} \\ y_{map} \\ z_{map} \end{bmatrix}\right) = \frac{R}{a} V \sqrt{\frac{1}{\lambda_1}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 \\ 0 & 0 & \sqrt{\lambda_3} \end{bmatrix} V^{-1} \begin{bmatrix} x - b_x \\ y - b_y \\ z - b_z \end{bmatrix}$$

A continuación se detallan los pasos que se siguen a la hora de calibrar el magnetómetro y el acelerómetro.

Calibración del magnetómetro

El primer paso es descargar las librerías LSM303 desde https://github.com/pololu/lsm303-arduino/archive/master.zip y descomprimir los archivos en la carpeta *libraries* de Arduino.

Se utiliza el siguiente código para recoger una serie de valores del sensor:

```
#include <Wire.h>
#include <LSM303.h>
LSM303 compass;

void setup()
{
Serial.begin(250000);
Wire.begin();
compass.init();
compass.enableDefault();
Serial.println("Magnetometro sin calibrar (Unidad Nanotesla)");
}

void loop()
{
compass.read();
float Xm_print, Ym_print, Zm_print;

Xm_print = compass.m.x*(100000.0/1100.0); // Gain X [LSB/Gauss] for selected sensor input field range (1.3 in these case)
Ym_print = compass.m.y*(100000.0/1100.0); // Gain Y [LSB/Gauss] for selected sensor input field range
Zm_print = compass.m.z*(100000.0/980.0 );  // Gain Z [LSB/Gauss] for selected sensor input field range

Serial.print(Xm_print, 10); Serial.print(" "); Serial.print(Ym_print, 10); Serial.print(" "); Serial.println(Zm_print, 10);
delay(125);
}
```

En el archivo LSM303.cpp, el valor predeterminado para el registro CRB_REG_M es 0x20. Si se convierte ese valor hexadecimal en binario, se obtiene 00100000. Comprobando la tabla 75 en la página número 38 en la hoja de datos, el rango de campo de entrada del sensor predeterminado es ±1,3 gauss.

```
242     // Magnetometer
243
244     // 0x0C = 0b00001100
245     // DO = 011 (7.5 Hz ODR)
246     writeMagReg(CRA_REG_M, 0x0C);
247
248     // 0x20 = 0b00100000
249     // GN = 001 (+/- 1.3 gauss full scale)
250     writeMagReg(CRB_REG_M, 0x20);
251
252     // 0x00 = 0b00000000
253     // MD = 00 (continuous-conversion mode)
254     writeMagReg(MR_REG_M, 0x00);
```

Tabla 11 - Registros Magnetómetro

CRB_REG_M (01h)

Table 73. CRB_REG_M register

GN2	GN1	GN0	0(1)	0(1)	0(1)	0(1)	0(1)

1. This bit must be set to '0' for correct operation of the device.

Table 74. CRB_REG_M description

GN[2:0]	Gain configuration bits. The gain configuration is common for all channels (refer to *Table 75*)

Table 75. Gain setting

GN2	GN1	GN0	Sensor input field range [Gauss]	Gain X, Y, and Z [LSB/Gauss]	Gain Z [LSB/Gauss]	Output range
0	0	1	±1.3	1100	980	0xF800–0x07FF (-2048 to +2047)
0	1	0	±1.9	855	760	
0	1	1	±2.5	670	600	
1	0	0	±4.0	450	400	
1	0	1	±4.7	400	355	
1	1	0	±5.6	330	295	
1	1	1	±8.1	230	205	

Se debe abrir el monitor en serie y mover la placa durante varios minutos en todas las direcciones. Una vez tenemos las lecturas realizadas, se copian los valores en un archivo de texto y se guarda.

El siguiente paso sería ejecutar la aplicación Magneto y cargar los datos. Para encontrar la norma del campo magnético, se puede hacer a través del enlace http://www.ngdc.noaa.gov/geomag-web/#igrfwmm, se introduce la localización y el servidor nos devuelve el dato que se buscaba.

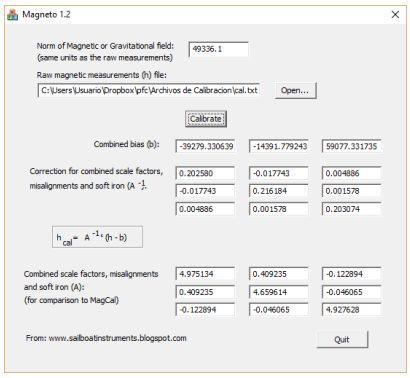


Ilustración 26 - Software Magneto

Llegado a este punto, solo hay que pulsar en el botón *Calibrate* y el software Magneto dará una serie valores que se utilizaran para el siguiente sketch de Arduino.

```
#include <Wire.h>
#include <LSM303.h>
LSM303 compass;

void setup()
{
Serial.begin(250000);
Wire.begin();
compass.init();
compass.enableDefault();
Serial.println("Magnetometer Calibrated (Units in Nanotesla)");
}

void loop()
{
compass.read();
float Xm_off, Ym_off, Zm_off, Xm_cal, Ym_cal, Zm_cal;

Xm_off = compass.m.x*(100000.0/1100.0) - 39279.330639; //X-axis combined bias (Non calibrated data - bias)
Ym_off = compass.m.y*(100000.0/1100.0) - 14391.779243; //Y-axis combined bias (Default: substracting bias)
Zm_off = compass.m.z*(100000.0/980.0 ) + 59077.331735; //Z-axis combined bias

Xm_cal =  0.179781*Xm_off + -0.015747*Ym_off + 0.004337*Zm_off; //X-axis correction for combined scale factors (Default: positive factors)
Ym_cal =  -0.015747*Xm_off + 0.191855*Ym_off + 0.001401*Zm_off; //Y-axis correction for combined scale factors
Zm_cal =  0.004337*Xm_off + 0.001401*Ym_off + 0.180220*Zm_off; //Z-axis correction for combined scale factors

Serial.print(Xm_cal, 10); Serial.print(" "); Serial.print(Ym_cal, 10); Serial.print(" "); Serial.println(Zm_cal, 10);
delay(125);
}
```

De nuevo los pasos a seguir será cargar el código, abrir el monitor serie y mover la placa en todas las direcciones.

Copiar valores en un archivo de texto y guardar el archivo.

Cargar en Magneto archivo de texto con valores calibrados.

Hacer clic en *Calibrate*.

Combined Bias (b) debe ahora estar cerca de cero

Calibración del acelerómetro

De un modo muy similar al realizado con el magnetómetro se realiza la calibración del acelerómetro.

En este caso se carga el siguiente código en Arduino:

```
#include <Wire.h>
#include <LSM303.h>
LSM303 compass;

void setup()
{
Serial.begin(250000);
Wire.begin();
compass.init();
compass.enableDefault();
Serial.println("Accelerometer Uncalibrated (Units in mGal)");
}

void loop()
{
compass.read();
float Xa_print, Ya_print, Za_print;

Xa_print = compass.a.x/16.0; //Acceleration data registers contain a left-aligned 12-bit number, so values should be shifted right by 4 bits (divided by 16)
Ya_print = compass.a.y/16.0;
Za_print = compass.a.z/16.0;

Serial.print(Xa_print); Serial.print(" "); Serial.print(Ya_print); Serial.print(" "); Serial.println(Za_print);
delay(125);
}
```

En el archivo LSM303.cpp, el valor predeterminado para el registro CTRL_REG4_A es 0x08.

Si convierte ese valor hexadecimal en binario, se obtiene 00001000.

Tabla 12 - Registros Acelerómetro

Comprobando la tabla 27 en la hoja de datos, notará que el acelerómetro tiene una escala de selección de ± 2G.

Y comprobando la tabla 3 para esa selección de escala, la sensibilidad de aceleración lineal es de 1 mg / LSB.

CTRL_REG4_A (23h)

Table 26. CTRL_REG4_A register

BDU	BLE	FS1	FS0	HR	0(1)	0(1)	SIM

1. This bit must be set to '0' for correct operation of the device.

Table 27. CTRL_REG4_A description

BDU	Block data update. Default value: 0 (0: continuous update, 1: output registers not updated until MSB and LSB have been read
BLE	Big/little endian data selection. Default value 0. (0: data LSB @ lower address, 1: data MSB @ lower address)
FS[1:0]	Full-scale selection. Default value: 00 (00: ±2 *g*, 01: ±4 *g*, 10: ±8 *g*, 11: ±16 *g*)
HR	High-resolution output mode: Default value: 0 (0: high-resolution disable, 1: high-resolution enable)
SIM	SPI serial interface mode selection. Default value: 0 (0: 4-wire interface, 1: 3-wire interface).

Tabla 13 - Características de Sensores

Table 3. Sensor characteristics

Symbol	Parameter	Test conditions	Min.	Typ.(1)	Max.	Unit
LA_FS	Linear acceleration measurement range(2)	FS bit set to 00		±2		*g*
		FS bit set to 01		±4		
		FS bit set to 10		±8		
		FS bit set to 11		±16		
M_FS	Magnetic measurement range	GN bits set to 001		±1.3		gauss
		GN bits set to 010		±1.9		
		GN bits set to 011		±2.5		
		GN bits set to 100		±4.0		
		GN bits set to 101		±4.7		
		GN bits set to 110		±5.6		
		GN bits set to 111		±8.1		
LA_So	Linear acceleration sensitivity	FS bit set to 00		1		mg/LSB
		FS bit set to 01		2		
		FS bit set to 10		4		
		FS bit set to 11		12		
M_GN	Magnetic gain setting	GN bits set to 001 (X,Y)		1100		LSB/ gauss
		GN bits set to 001 (Z)		980		
		GN bits set to 010 (X,Y)		855		
		GN bits set to 010 (Z)		760		
		GN bits set to 011 (X,Y)		670		
		GN bits set to 011 (Z)		600		
		GN bits set to 100 (X,Y)		450		
		GN bits set to 100 (Z)		400		
		GN bits set to 101 (X,Y)		400		
		GN bits set to 101 (Z)		355		
		GN bits set to 110 (X,Y)		330		
		GN bits set to 110 (Z)		295		
		GN bits set to 111(2) (X,Y)		230		
		GN bits set to 111(2) (Z)		205		

Se debe abrir el monitor en serie y mover la placa durante varios minutos en todas las direcciones. Una vez tenemos las lecturas realizadas, se copian los valores en un archivo de texto y se guarda.

Ejecutar Magneto y cargar archivo de texto con lecturas sin procesar. Para la norma del campo gravitatorio, se introduce 1000 (ya que las unidades brutas están en mGal).

Se obtienen los valores combined bias y los factores de escala, quedando el codigo Arduino como se muestra en la imagen.

```
#include <Wire.h>
#include <LSM303.h>
LSM303 compass;

void setup()
{
Serial.begin(9600);
Wire.begin();
compass.init();
compass.enableDefault();
Serial.println("Accelerometro Calibrado (Unidades en mGal)");
}

void loop()
{
compass.read();
float Xa_off, Ya_off, Za_off, Xa_cal, Ya_cal, Za_cal;

Xa_off = compass.a.x/16.0 - 7.837247; //X-axis combined bias (Non calibrated data - bias)
Ya_off = compass.a.y/16.0 + 6.828846; //Y-axis combined bias (Default: substracting bias)
Za_off = compass.a.z/16.0 + 79.337329; //Z-axis combined bias

Xa_cal =  0.956712*Xa_off -0.026425*Ya_off +0.029607*Za_off; //X-axis correction for combined scale factors (Default: positive factors)
Ya_cal =  -0.026425*Xa_off + 0.990613*Ya_off -0.003651*Za_off; //Y-axis correction for combined scale factors
Za_cal = 0.029607*Xa_off -0.003651*Ya_off + 0.958057*Za_off; //Z-axis correction for combined scale factors

Serial.print(Xa_cal); Serial.print(" "); Serial.print(Ya_cal); Serial.print(" "); Serial.println(Za_cal);
delay(125);
}
```

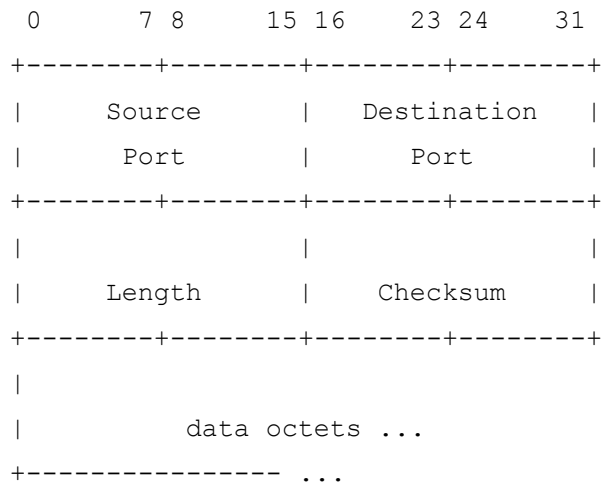
Se carga el código, repitiendo el procedimiento y se comprueba que el sesgo ha disminuido.

Protocolo UDP

En redes de ordenadores, el protocolo UDP es uno de principales protocolos de Internet. El protocolo fue diseñado por David P. Reed en 1980 y formalmente definido en RFC 768. Con UDP, las aplicaciones informáticas pueden enviar mensajes, en este caso denominados datagramas, a otros hosts en una red IP. No se requieren comunicaciones previas para configurar canales de comunicación o rutas de datos.

UDP utiliza un modelo de comunicación simple sin conexión con un mínimo de mecanismo de protocolo. UDP proporciona sumas de comprobación para la integridad de datos y números de puerto para dirigir diferentes funciones en la fuente y el destino del datagrama. No tiene diálogos de handshaking, y por lo tanto expone el programa del usuario a cualquier falta de fiabilidad de la red subyacente, no posee garantía de entrega, ordenación o protección duplicada. El Protocolo de control de transmisión (TCP) o el Protocolo de transmisión de control de flujo (SCTP) diseñados para este propósito.

UDP es adecuado para fines en los que la comprobación y corrección de errores no son necesarias o se realizan en la aplicación; UDP evita la sobrecarga de dicho procesamiento en la pila de protocolos. Las aplicaciones sensibles al tiempo usan a menudo UDP porque la caída de paquetes es preferible a la espera de paquetes retrasados debido a la retransmisión, lo cual puede no ser una opción en un sistema en tiempo real.



User Datagram Header Format

Source Port, es un campo opcional, cuando es significativo, indica el puerto del proceso de envío, y se puede suponer que es el puerto al que la respuesta se debe abordar en ausencia de cualquier otra información. Si no se utiliza, se inserta un valor de cero.

Destination Port, tiene un significado dentro del contexto de una dirección de destino de Internet particular.

Length, es la longitud en octetos de este datagrama de usuario que incluye este encabezado y los datos. (Esto significa que el valor mínimo de la longitud es ocho.)

Checksum, es el complemento de 16 bits de la suma de complemento de una pseudo encabezado de información de la cabecera IP, la cabecera UDP y los datos, rellenados con cero octetos al final (si es necesario) para hacer un múltiplo de dos octetos .

Esta información da protección contra los datos perdidos. Este procedimiento de suma de comprobación es el mismo que se utiliza en TCP.

UDP, es generalmente, el protocolo usado en la transmisión de vídeo y voz a través de una red. Esto es debido a que no hay tiempo para enviar de nuevo paquetes perdidos cuando se está transmitiendo audio o viendo un vídeo en tiempo real.

Ya que tanto TCP como UDP circulan por la misma red, en muchos casos ocurre que el aumento del tráfico UDP daña el correcto funcionamiento de las aplicaciones TCP. Por defecto, TCP pasa a un segundo lugar para dejar a los datos en tiempo real usar la mayor parte del ancho de banda.

Capítulo 5
Modelo de vehículo en Matlab

Al estar en proceso de desarrollo y no disponer del vehículo físico, el modelo del vehiculo que se ha utilizado en este proyecto, es el punto a abarcar en este capítulo. Fue desarrollado por *Javier de la Red Calvo* en la elaboración de su proyecto fin de carrera y está realizado en entorno Matlab. En este modelo, se programan en Matlab las ecuaciones presentadas en el capítulo 3.

El modelo del ROV CUBO viene integrado en una función denominada ROV2.

```
function [Vecf,Vecder] = ROV2(x,Nprop,j)
```

Se trata de una función con tres argumentos de entrada y dos de salida. Como entradas son necesarias introducir el vector de estado (x), el vector de parámetros de control (Ni) y el estado de iteración (j)

x = [u v w p q r x y z phi theta psi]

Tabla 14 - Vector de Estado

u	Velocidad de Avance	(m/s)
v	Velocidad de Deriva	(m/s)
w	Velocidad de Arfada	(m/s)
p	Velocidad de Balance	(rad/s)
q	Velocidad de Cabeceo	(rad/s)
r	Velocidad de Guiñada	(rad/s)
x	Posición en la dirección de x	(m)
y	Posición en la dirección de y	(m)
z	Posición en la dirección de z	(m)
phi	Ángulo de Balanceo	(rad)
theta	Ángulo de Cabeceo	(rad)
psi	Ángulo de Guiñada	(rad)

Ni = [N1 N2 N3 N4 N5 N6]

Tabla 15 - Vector de parámetros de control

N1	Revoluciones de la Hélice 1 (estribor) (rpm)
N2	Revoluciones de la Hélice 2 (babor) (rpm)
N3	Revoluciones de la Hélice 3 (proa) (rpm)
N4	Revoluciones de la Hélice 4 (popa) (rpm)
N5	Revoluciones de la Hélice 5 (abajo) (rpm)
N6	Revoluciones de la Hélice 6 (arriba) (rpm)

Los argumentos de salida son el vector de fuerzas y momentos hidrodinámicos Vecf y la derivada del vector de estado Vecder

Vecf = [X Y Z K M N]

Tabla 16 - Vector de Fuerzas

X	Fuerza en la dirección X (N)
Y	Fuerza en la dirección Y (N)
Z	Fuerza en la dirección Z (N)
K	Momento en la dirección X (N m)
M	Momento en la dirección Y (N m)
N	Momento en la dirección Z (N m)

La implementación de la función se muestra en el siguiente código:

```
function [Vecf,Vecder] = ROV2(x,Nprop,j)

%%Coef ROV

g = 9.81; % m/s^2

% Valor de la Densidad del Agua de Mar

rho = 1.03e3; % kg/m^3

% Propiedades Fisicas del ROV:

% Distancia propulsores centro geométrico

L = 0.155; % m

% Coordenadas del Centro de Gravedad (c.d.g)

xG = 0;       yG = 0;       zG = 1.6e-2; % m
% zG debe ser positivo para que el centro de gravedad se encuentre por
debajo del centro de flotación.

% Peso y Empuje del Vehículo

W = 84;  B = 86.72; % N

% Masa del Vehículo

m = W/g; %kg

% Momentos de Inercia del Vehículo

Ix = 1.0104e-1;      Iy = 1.0104e-1;    Iz = 9.1442e-2; % kg/m^2
(consideramos nulos los productos de inercia)

% Coeficientes hidrodinámicos del ROV
```

```
Kma1 = 0.5; % relación de los parámetros de masa añadida lineales
(Xudot, Yvdot ,Zwdot) con la masa del sólido
Kma2 = 0.2; % relación del parámetro Nrdot con la masa del sólido

Xudot = -Kma1*m;
Xuu = -35;
Xu = -7;

Yvdot = -Kma1*m;
Yvv = -35;
Yv = -7;

Zwdot = -Kma1*m;
Zww = -35;
Zw = -7;

Kpdot = -Kma2*m;
Kpp = -2;
Kp = -1;

Mqdot= -Kma2*m;
Mqq = -2;
Mq = -1;

Nrdot = -Kma2*m;
Nrr = -2;
Nr = -1;

% Limite máximo de las Revoluciones de la Hélice (2000 rpm)

mxrpm = 3800;

% Coeficientes de las hélices

Tnn = 6.5e-5;
```

```
% Dimensiones de Estado

u    = x(1);  v     = x(2);  w    = x(3);
p    = x(4);  q     = x(5);  r    = x(6);
phi = x(10); theta = x(11); psi = x(12);

%Matriz de masa (masa del sólido rígido más la masa añadida)



MMass = [m-Xudot  0         0         0          m*zG       -m*yG
         0        m-Yvdot   0         -m*zG      0          m*xG
         0        0         m-Zwdot   m*yG       -m*xG      0
         0        -m*zG     m*yG      Ix-Kpdot   0          0
         m*zG     0         -m*xG     0          Iy-Mqdot   0
         -m*yG    m*xG      0         0          0          Iz-Nrdot];

MMassInv = inv(MMass);

% Saturaciones del Eje

for i=1:1:6,
   if abs(Nprop(j,i))>mxrpm
     Nprop(j,i)=sign(Nprop(j,i))*mxrpm;end

end

% Cálculo de las fuerzas de propulsión
    Np = zeros(6);
    T = zeros(6);

for i=1:1:6,
   Np(i) = Nprop(j,i)/60*2*pi;
   T(i) = Tnn*Np(i)*abs(Np(i)); % el efecto de los propulsores se
considera unicamente cuadrático
end
```

```
% Fuerzas y Momentos Hidrodinámicos

X = - (W - B)*sin(theta)+ Xuu*u*abs(u) + Xu*u - m*(w*q - v*r -
xG*(q^2+r^2) ...
    + yG*p*q + zG*p*r) + T(1) + T(2);

Y = (W - B)*cos(theta)*sin(phi) + Yvv*v*abs(v) + Yv*v - m*(u*r - w*p
- yG*(p^2+r^2) ...
    + xG*p*q + zG*q*r) + T(5) + T(6);

Z = (W - B)*cos(theta)*cos(phi) + Zww*w*abs(w) +Zw*w - m*(v*p - u*q -
zG*(p^2+q^2) ...
    + xG*p*r + yG*q*r) + T(3) + T(4);

K = yG*W*cos(theta)*cos(phi) - zG*W*cos(theta)*sin(phi) + Kpp*p*abs(p)
+ Kp*p ...
    + (Iy - Iz)*q*r + m*yG*(u*q - v*p) + m*zG*(u*r-w*p) + L*(T(6) -
T(5));

M = -xG*W*cos(theta)*cos(phi) - zG*W*sin(theta) + Mqq*q*abs(q) + Mq*q
...
    + (Iz - Ix)*r*p + m*zG*(v*r - w*q) ...
    + m*xG*(v*p - u*q) + L*(T(4) - T(3));

N = xG*W*cos(theta)*sin(phi) + yG*W*sin(phi) + Nrr*r*abs(r) + Nr*r ...
    +(Ix - Iy)*p*q + m*yG*(w*q - v*r) + m*xG*(w*p-u*r) ...
    + L*(T(2) - T(1));

% Derivada del vector x

udot                                                                =
MMassInv(1,1)*X+MMassInv(1,2)*Y+MMassInv(1,3)*Z+MMassInv(1,4)*K+MMas
sInv(1,5)*M+MMassInv(1,6)*N;
vdot                                                                =
MMassInv(2,1)*X+MMassInv(2,2)*Y+MMassInv(2,3)*Z+MMassInv(2,4)*K+MMas
sInv(2,5)*M+MMassInv(2,6)*N;
```

```
K = yG*W*cos(theta)*cos(phi) - zG*W*cos(theta)*sin(phi) + Kpp*p*abs(p)
+ Kp*p ...
    + (Iy - Iz)*q*r + m*yG*(u*q - v*p) + m*zG*(u*r-w*p) - m*zG*vdot +
L*(T(6) - T(5));

M = -xG*W*cos(theta)*cos(phi) - zG*W*sin(theta) + Mqq*q*abs(q) + Mq*q
...
    + (Iz - Ix)*r*p + m*zG*(v*r - w*q) - m*zG*udot ...
    + m*xG*(v*p - u*q) + L*(T(4) - T(3));

Vecder                                                              =
[MMassInv(1,1)*X+MMassInv(1,2)*Y+MMassInv(1,3)*Z+MMassInv(1,4)*K+MMa
ssInv(1,5)*M+MMassInv(1,6)*N

MMassInv(2,1)*X+MMassInv(2,2)*Y+MMassInv(2,3)*Z+MMassInv(2,4)*K+MMas
sInv(2,5)*M+MMassInv(2,6)*N

MMassInv(3,1)*X+MMassInv(3,2)*Y+MMassInv(3,3)*Z+MMassInv(3,4)*K+MMas
sInv(3,5)*M+MMassInv(3,6)*N

MMassInv(4,1)*X+MMassInv(4,2)*Y+MMassInv(4,3)*Z+MMassInv(4,4)*K+MMas
sInv(4,5)*M+MMassInv(4,6)*N

MMassInv(5,1)*X+MMassInv(5,2)*Y+MMassInv(5,3)*Z+MMassInv(5,4)*K+MMas
sInv(5,5)*M+MMassInv(5,6)*N

MMassInv(6,1)*X+MMassInv(6,2)*Y+MMassInv(6,3)*Z+MMassInv(6,4)*K+MMas
sInv(6,5)*M+MMassInv(6,6)*N
          cos(psi)*cos(theta)*u    +    (cos(psi)*sin(theta)*sin(phi)-
sin(psi)*cos(phi))*v                                                +
(sin(psi)*sin(phi)+cos(psi)*cos(phi)*sin(theta))*w
          sin(psi)*cos(theta)*u                                     +
(cos(phi)*cos(psi)+sin(phi)*sin(theta)*sin(psi))*v                  +
(cos(phi)*sin(theta)*sin(psi)-cos(psi)*sin(phi))*w
          -sin(theta)*u        +        cos(theta)*sin(phi)*v       +
cos(phi)*cos(theta)*w
          p + sin(phi)*tan(theta)*q + cos(phi)*tan(theta)*r
          cos(phi)*q - sin(phi)*r
          sin(phi)/cos(theta)*q + cos(phi)/cos(theta)*r];
```

```
Vecf = [X Y Z K M N]';
```

Capítulo 6
Lógica Difusa

Introducción

Como indicó una vez Platón, había una tercera región más allá de lo verdadero y lo falso, donde los contrarios se desplomaban.


Ilustración 27 – Jan Lukasiewicz

En el siglo XX, Jan Lukasiewicz propuso una lógica de tres valores (verdadero, posible, falso).

En 1965, un reconocido matemático, Lofti A. Zadeh, de la Universidad de California en Berkeley propuso Fuzzy Logic (FL) en un documento que detalla las matemáticas de la teoría de conjuntos difusos.


Ilustración 28 - Lofti A. Zadeh

Zadeh profundizó en sus ideas en un artículo de 1973 que introdujo el concepto de variables difusas que pertenecen a un conjunto difuso. Una aplicación industrial de sistemas borrosos implementada en un horno de cemento en Dinamarca, que entró en línea en 1975.

Aunque FL tuvo su origen en América, los científicos estadounidenses y europeos consideraron la teoría como no matemática e infantil, y nunca obtuvo una amplia aceptación.

En Japón, surgió un interés por la lógica difusa llevada a cabo por *Seiji Yasunobu* y *Soji Miyamoto* de *Hitachi*, quienes en el año 1985 demostraron la supremacía de los sistemas de control difusos en la gestión de la aceleración, frenado y detención de trenes en Sendai.

Dos años más tarde, durante una reunión internacional en Tokio, *Takeshi Yamakawa* llevó a cabo un experimento de péndulo invertido utilizando sencillo circuito de lógica difusa basada en chips, donde instaló una copa de vino que contenía un ratón encima del péndulo. Como resultado directo de la demostración, FL se hizo popular en


Ilustración 29 - Takeshi Yamakawa

Japón para varias aplicaciones industriales y de consumo.En 1988, 48 empresas japonesas establecieron el *Laboratory for International Fuzzy Engineering* (LIFE). Hoy en día, los productos japoneses que utilizan la lógica difusa van desde lavadoras hasta cámaras de enfoque automático y dispositivos de aire acondicionado industriales.

El cerebro humano interpreta la información sensorial imprecisa e incompleta proporcionada por los órganos perceptivos. La teoría de conjuntos difusos proporciona un cálculo sistemático para tratar dicha información lingüísticamente, y realiza cálculos numéricos utilizando etiquetas lingüísticas estipuladas por funciones de pertenencia. Un sistema de inferencia difusa (FIS) cuando se selecciona adecuadamente puede efectivamente modelar la experiencia humana en una aplicación específica.

En contraste con un conjunto clásico, un conjunto difuso, como su nombre lo indica, es un conjunto sin un límite nítido. La transición de "*pertenece a un conjunto*" a "*no pertenece a un conjunto*" es gradual y esta transición suave se caracteriza por funciones de pertenencia que dan flexibilidad de conjuntos difusa en el modelado de expresiones lingüísticas de uso común.

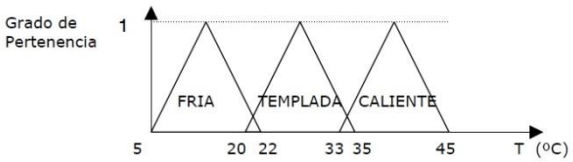


Ilustración 30- Conjunto Difuso

Sea $\boldsymbol{X}$ un espacio de objetos y x un elemento genérico de X. Un conjunto clásico A, $\boldsymbol{A \subseteq X}$ se define como una colección de elementos u objetos x∈ X, de tal manera que cada x puede pertenecer o no pertenecer al conjunto A.

Definiendo una función característica para cada elemento x en X, podemos representar un conjunto clásico A por un conjunto de pares ordenados (x, 0) o (x, 1), el cual, respectivamente, indica x∉ A o x∈ A.

A diferencia del conjunto clásico, un conjunto difuso expresa el grado al cual un elemento pertenece a un conjunto. Por lo tanto, se permite que la función característica de un conjunto difuso tenga valores entre 0 y 1, lo que denota el grado de pertenencia de un elemento en un conjunto dado.

Si X es una colección de objetos denotados genéricamente por x, entonces un conjunto difuso A en x se define como un conjunto de pares ordenados:

A = {(x, $\mu_A(x)$) , tal que x∈ X

, donde ***$\mu_A(x)$*** se denomina ***función de pertenencia*** (*membership function, MF*) para el conjunto difuso A. La MF asigna cada elemento de x a un valor de pertenencia entre 0 y 1. Es obvio que si el valor de la función de pertenencia ***$\mu_A(x)$*** está restringido a 0 o 1, entonces A se reduce a un conjunto clásico y ***$\mu_A(x)$*** es la función característica de A. X puede ser un espacio continuo o un espacio discreto.

En un espacio continuo, por lo general, se particiona X en varios conjuntos difusos cuyas MF abarcan *x* de un modo uniforme. Estos conjuntos difusos poseen nombre que se relacionas con los adjetivos que normalmente utilizamos en la vida cotidiana como *caliente, rápida, negativa, grande, etc.,* se denominan "***valores lingüísticos***" o "***etiquetas lingüísticas***".

Membership Functions (MF)

Las funciones de pertenencia MF se expresan con la ayuda de una fórmula matemática que puede ser parametrizadas según la complejidad requerida. Estas pueden ser unidimensionales o multidimensionales. Los tipos más utilizados son:

Triangular

Una función de pertenencia triangular viene dada por tres parámetros (a, b, c), tal que:

$$MF_{triangular} = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \end{cases}$$

Usando min y máx., existe una expresión alternativa para dicha ecuación.

$$MF_{triangular} = \max\left(\min\left\langle\frac{x-a}{b-a}, \frac{c-x}{c-b}, 0\right\rangle\right)$$

Los parámetros (a, b, c), con a <b <c, determinan las coordenadas x de las tres esquinas de la MF triangular.

Trapezoidal

Una función de pertenencia trapezoidal viene dada por los parámetros (a, b, c, d), tal que:

$$MF_{trapezoidal} = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases}$$

Una expresión alternativa para dicha ecuación usando min y máx.

$$MF_{trapezoidal} = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}, 0\right)\right)$$

Los parámetros (a, b, c, d), con a <b <c<d, determinan las coordenadas x de las cuatro esquinas de la MF trapezoidal.

Gaussiana

Una función de pertenencia gaussiana (campana) viene dada por los parámetros (a, b, c), tal que:

$$MF_{gaussiana} = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

**Donde el parámetro b es normalmente positivo.*

También se llama MF de Cauchy.

Sigmoide

Una función de pertenencia sigmoide definida por:

$$MF_{sigmoide} = \frac{1}{1+e^{-a(x-c)}}$$

Donde a controla la pendiente en el punto de cruce x = c.

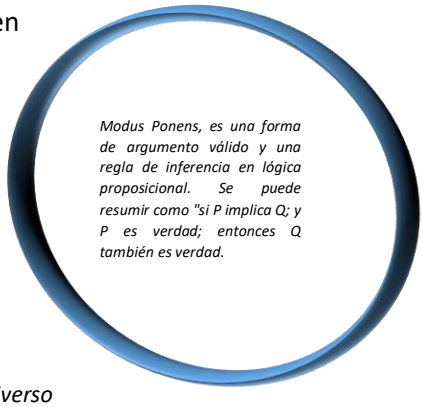
Las funciones sigmoides son ampliamente utilizadas en redes neuronales artificiales.

Variables lingüísticas y reglas difusas de If-Then

En 1973, el profesor Lotfi Zadeh propuso el concepto de variables lingüísticas o "difusas". Pensando en ellas como objetos lingüísticos o palabras, en lugar de números.

La entrada del sensor es un nombre, por ejemplo "temperatura", "desplazamiento", "velocidad", "flujo", "presión", etc. Dado que el error es sólo la diferencia, puede pensarse de la misma manera que las variables difusas son adjetivos que modifican la variable (por ejemplo, error "grande positivo", error "pequeño positivo", error "cero", error "pequeño negativo" y error "grande negativo"). Como mínimo, podríamos simplemente tener variables "positivas", "cero" y "negativas" para cada uno de los parámetros, aunque también se pueden añadir rangos adicionales tales como "muy grande" y "muy pequeño" para extender la respuesta a condiciones excepcionales o muy no lineales, pero no serían necesarios en un sistema básico.

Una vez definidas las variables y valores lingüísticos, se pueden formular las reglas del sistema de inferencia fuzzy. Estas reglas asignan las entradas difusas a las salidas difusas. Esta situación tiene lugar a través de la regla de inferencia composicional que se basa en la extensión de Zadeh de *modus ponens* que no es nada más que la forma If-Then condicional.



Una regla fuzzy if-then (también conocida como regla difusa) asume la forma:

$$If\ x\ is\ A\ then\ y\ is\ B$$

Donde A y B son valores lingüísticos definidos por conjuntos difusos en el universo X e Y, respectivamente. "X es A" se denomina antecedente o premisa, mientras que "y es B" se denomina consecuencia o conclusión. Esta regla queda abreviada como $A \rightarrow B$

El antecedente consiste normalmente en alguna combinación de las variables de entrada y la consecuencia consiste en variables de salida.

La variable lingüística y su valor lingüístico se combinan con las otras variables y valores sobre el antecedente por los operadores AND u OR difusos. La elección depende del sistema de inferencia deseado, el operador AND corresponde a la intersección de los conjuntos difusos y el operador OR corresponde a la unión de los conjuntos difusos.

Generar las reglas para el sistema de inferencia fuzzy es a menudo el paso más difícil en el proceso de diseño. Por lo general requiere un conocimiento experto de la dinámica de la planta.

Este conocimiento podría ser en forma de una comprensión intuitiva adquirida a partir de la experimentación, o podría provenir de un modelo de planta que luego se utiliza en una simulación por ordenador, este último es el caso de este proyecto fin de carrera.

Sistemas de Inferencia Difusa

Existen tres sistemas principales de inferencia de lógica: tipo Mamdani, tipo Sugeno y tipo Tsukamoto.

El sistema de inferencia difusa Mamdani, en el que se centrara principalmente la atención se utiliza como se muestra en la Ilustración 31, el controlador contiene cuatro partes principales, dos de las cuales realizan transformaciones.

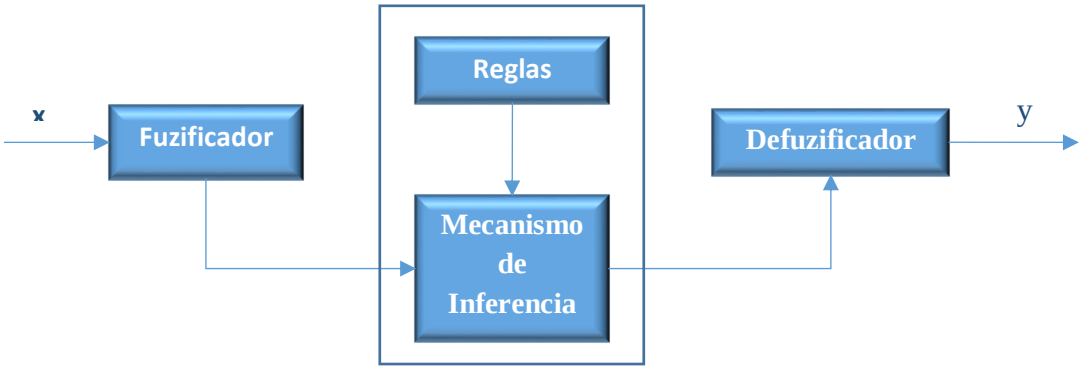


Ilustración 31 - Sistema Mamdani

El **fuzzificador** realiza mediciones de las variables de entrada (señales de entrada, variables reales), mapeo de escala y fuzzificación (transformación 1). Así, todas las señales monitorizadas son escaladas, y la fuzzificación significa que las señales medidas (entradas que tienen valores numéricos) se transforman en cantidades difusas (que también se denominan variables lingüísticas). Esta transformación se realiza mediante funciones de pertenencia. En un controlador de lógica difusa convencional, el número de funciones de pertenencia y las formas de éstas se determinan inicialmente por el usuario. Una función de pertenencia tiene un valor entre 0 y 1, e indica el grado de pertenencia de una cantidad a un conjunto difuso. Si es absolutamente seguro que la cantidad pertenece al conjunto difuso, entonces su valor es 1, pero si es absolutamente seguro que no pertenece a este conjunto entonces su valor es 0.

La base de conocimientos consiste en la base de datos y la base de reglas de control lingüístico. La base de datos proporciona la información que se utiliza para definir las reglas de control

lingüístico y la manipulación de datos difusos en el controlador de lógica difusa. El conjunto de reglas define específicamente las acciones de la meta de control mediante un conjunto de reglas lingüísticas, como las que proporcionaría un experto, las cuales contiene un conjunto de bloques If-Then.

Los principales métodos para desarrollar la base de reglas son:

- Utilizar la experiencia y el conocimiento de un experto para la aplicación y los objetivos de control.
- Modelar la acción de control del operador.
- Modelar el proceso.
- Utilizar un controlador fuzzy.
- Uso de un controlador artificial.
- Uso de redes neuronales artificiales.

Cuando las reglas iniciales se obtienen usando consideraciones físicas expertas, éstas pueden ser formadas considerando los tres objetivos principales:

- Eliminación de errores significativos en la salida del proceso mediante ajustes adecuados de la salida de control
- Asegurar una acción de control suave cerca del valor de referencia
- Evitar que la salida del proceso exceda los valores especificados por el usuario.

El **motor de inferencia** (mecanismo de razonamiento) es el núcleo de FLC y tiene la capacidad de simular la toma de decisiones humana basada en conceptos difusos e inferir las acciones de control difusas mediante el uso de implicaciones difusas y las reglas de inferencia de la lógica difusa. Una vez que todas las variables de entrada monitorizadas se transforman en sus respectivas variables lingüísticas, el motor de inferencia evalúa el conjunto de reglas If-Then (dadas en la base de reglas) y así se obtiene un resultado que es de nuevo un valor para la variable lingüística. El resultado lingüístico tiene que ser entonces transformado en un valor de salida numérica del FLC y es por eso que hay una segunda transformación en el FLC. La segunda transformación es realizada por el defuzzificador. El defuzzificador produce una acción de control no fuzzy, usando las funciones de pertenencia consecutivas de las reglas.

Existen muchas técnicas de defuzzificación, algunas de las cuales se comentan a continuación:

Centroide

$$COA = \frac{\int \mu_A(z) z dz}{\int \mu_A(z) dz}$$

Donde $\mu_A\ (z)$ es la salida agregada MF y z es la cantidad de salida. Esta es la estrategia de defuzzificación es la más adoptada.

Para valores discretos, la ecuación anterior se expresa de la forma:

$$COA = \frac{\sum_{k=1}^{n} \mu_A(Z_k) Z_k dz}{\sum_{k=1}^{n} \mu_A(Z_k) dz}$$

Donde $\mu_A(Z_k)$ son los k = 1,2, ..., n valores muestreados de la función de membresía de salida.

Centro de sumas (CoS, Center of Sums)

Este método es similar al del centroide, pero su implementación puede resultar mucha mas eficiente ya que no se descartan las áreas solapadas al evaluar las salidas agregadas, por lo que es posible que estas se consideren más de una vez.

$CoS = \frac{\int z \sum_{i=1}^{n} \mu_{A_i}(z) dz}{\int \sum_{i=1}^{n} \mu_{A_i}(z) dz}$, donde n es la cantidad de reglas activadas para esa consecuencia

En caso de valores discretos:

$$CoS = \frac{\sum_{k=1}^{l} z_k \sum_{i=1}^{n} \mu_{Z_i}(Z_k)}{\sum_{k=1}^{l} \sum_{i=1}^{n} \mu_{Z_i}(Z_k)}$$

Método Mean-Max

Mediante esta técnica de defuzzificación el valor medio de salida $z = \frac{z_1 + z_2}{2}$ es obtenido siendo $z_1$el primer valor y $z_2$el último valor, donde la función de pertenencia $\mu_A\ (z)$ es máxima.

Método FOM (First of Maxima)

Cuando se utiliza esta técnica de defuzzificación, selecciona el menor valor que alcance el máximo grado de pertenencia en el conjunto difuso.

Método LOM (Last of Maxima)

A diferencia del anterior, selecciona el mayor valor que alcance el máximo grado de pertenencia en el conjunto difuso.

El control PID requiere, en general, menos tiempo de procesador para calcular señales de control (una de las ventajas de PID) que un FLC comparable. El tiempo de procesamiento requerido cuando se utiliza el control de lógica difusa depende del número de reglas que se deben evaluar. Los sistemas grandes con muchas reglas requieren procesadores muy potentes y rápidos para calcular en tiempo real. Cuanto menor sea la base de reglas, menor será la potencia computacional necesaria. Para reducir el tiempo de procesamiento, se puede usar una tabla de búsqueda estática para generar la acción de control FLC. En algunas aplicaciones, que puede reducir considerablemente el tiempo de procesamiento en comparación con la realización de inferencia difusa.

Los sistemas de control difusos utilizados en este proyecto poseen dos entradas: error y derivada del error, en cada una de sus variables a controlar, como pueden ser los ángulos de orientación del dispositivo o la profundidad. De este modo los FLC son sistemas de dos entradas y dos salidas (motores correspondientes).

El sistema Sugeno en comparación con Mamdani es una representación más compacta y eficiente desde el punto de vista computacional, hace uso de técnicas adaptativas para construir modelos difusos las cuales se pueden utilizar para personalizar las funciones de pertenencia. El modelo Mamdani es más intuitivo y tiene amplia aceptación.

Se utilizó *MATLAB's Fuzzy Logic Toolbox* para ayudar en el diseño FLC. Este toolbox contiene funciones, interfaces gráficas de usuario y estructuras de datos que permiten al usuario diseñar, probar, simular y modificar un sistema de inferencia difusa. Nos proporciona funciones para muchos métodos comunes, incluyendo el agrupamiento borroso y el aprendizaje adaptativo *neurofuzzy*. El toolbox permite modelar comportamientos complejos del sistema usando reglas lógicas sencillas, e implementar estas reglas en un sistema de inferencia difusa.

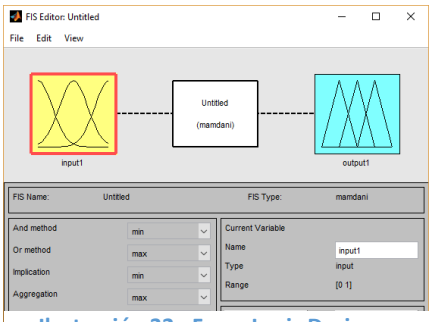


Ilustración 32 - Fuzzy Logic Designer

La aplicación *Fuzzy Logic Designer* nos permite diseñar y probar sistemas de inferencia difusos para modelar comportamientos complejos del sistema. Esta aplicación ofrece la capacidad de:

- Diseñar sistemas de inferencia fuzzy de Mamdani y Sugeno.
- Agregar o eliminar variables de entrada y salida.
- Especificar las funciones de pertenencia de entrada y salida.
- Definir reglas difusas If-Then.
- Seleccionar las funciones de inferencia difusa para operaciones AND y OR
- Implicación
- Agregación
- Defuzzificación
- Ajustar los valores de entrada y vea los diagramas de inferencia fuzzy asociados.
- Ver mapas de superficie de salida para sistemas de inferencia difusa.
- Exportar sistemas de inferencia fuzzy al espacio de trabajo MATLAB®.

En la realización de este proyecto se ha empleado la herramienta anteriormente mencionada, pero también se realizaron script basados en las funciones que incluye a modo de línea de comandos. Las funciones que se utilizaron se detallan en la siguiente tabla:

Tabla 17- Funciones Toolbox Fuzzy

newfis	Crea un nuevo sistema de inferencia fuzzy
genfis	Genera estructura del sistema de inferencia fuzzy a partir de unos datos
genfisOptions	Establece opciones para el comando genfis
addvar	Añade variable al sistema de inferencia fuzzy
rmvar	Elimina variable del sistema de inferencia fuzzy
mam2sug	Transforma sistema Mamdani a Sugeno
getfis	Obtiene las propiedades del sistema de inferencia fuzzy
setfis	Establece las propiedades del sistema de inferencia fuzzy
mfedit	Abre el editor de funciones de pertenencia
addmf	Añade función de pertenencia
rmmf	Elimina función de pertenencia
ruledit	Abre el editor de reglas

ruleview	Abre el visualizador de reglas
addrule	Añade una regla
showrule	Muestra reglas del sistema de inferencia fuzzy
evalfis	Realiza los cálculos de inferencia fuzzy
plotfis	Realiza un plot del sistema de inferencia fuzzy
surfview	Abre el visualizador de superficie del sistema de inferencia fuzzy
gensurf	Genera la superficie de salida del sistema de inferencia fuzzy
gensurfOptions	Establece opciones para el comando gensurf
showfis	Muestra el contenido del sistema de inferencia fuzzy
readfis	Carga sistema de inferencia fuzzy desde archivo
writefis	Guarda sistema de inferencia fuzzy a archivo
trimf	Función de membresía en forma triangular
trapmf	Función de membresía en forma trapezoidal
gaussmf	Función de membresía en forma de curva gaussiana
sigmf	Función de membresía en forma de sigmoide
plotmf	Realiza un plot de todas las funciones de membresía para una variable

La estructura de un archivo FIS contiene toda la información del sistema de inferencia difusa, incluyendo los nombres de variables, definiciones de función de pertenencia y métodos de inferencia difusa. Esta estructura es en sí una jerarquía de estructuras, como se muestra en el siguiente árbol:

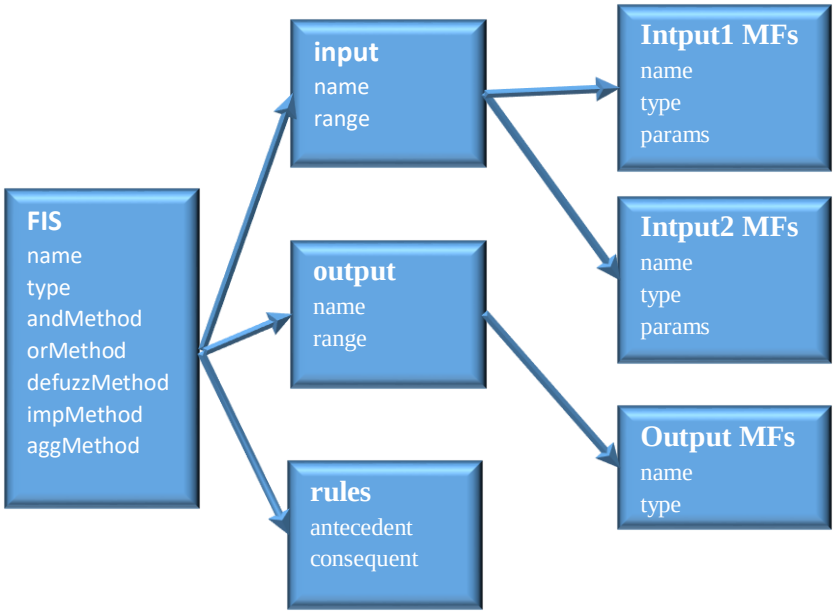


Ilustración 33- Estructura FIS

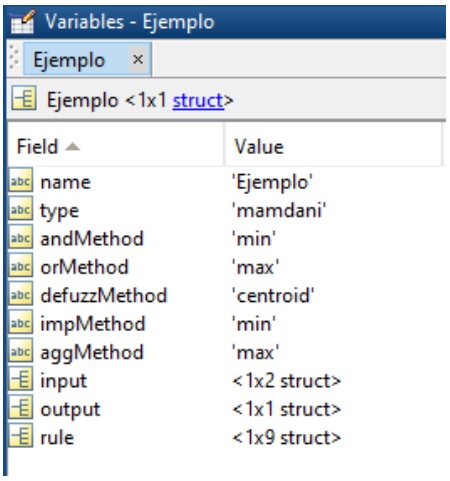


Ilustración 34 - Campos de FIS

Se muestra un ejemplo simple para un FIS de dos entradas y una salida, donde se puede ver la estructura y cada uno de los campos que la definen

Este ejemplo ha sido generado con el FIS Editor de Matlab

Se clasifican las entradas y salida según 3 funciones de pertenencia, con las variables lingüísticas pequeña, mediana y grande

MF1: pequeño, MF2: mediano, MF3: grande

La tabla de reglas del sistema de inferencia difusa, según las variables lingüísticas, quedaría según se muestra en la tabla

Tabla 18 - Ejempo de reglas I

Input2 \ Input1	Pequeño	Mediano	Grande
Pequeño	Pequeño	Grande	Pequeño
Mediano	Grande	Mediano	Pequeño
Grande	Grande	Pequeño	Grande

Si se trasladan a funciones de pertenencia:

Tabla 19 - Ejemplo de reglas II

Input2 \ Input1	MF1	MF2	MF3
MF1	MF1	MF3	MF1
MF2	MF3	MF2	MF1
MF3	MF3	MF1	MF3

Se define en Matlab las funciones de pertenecia del sistema

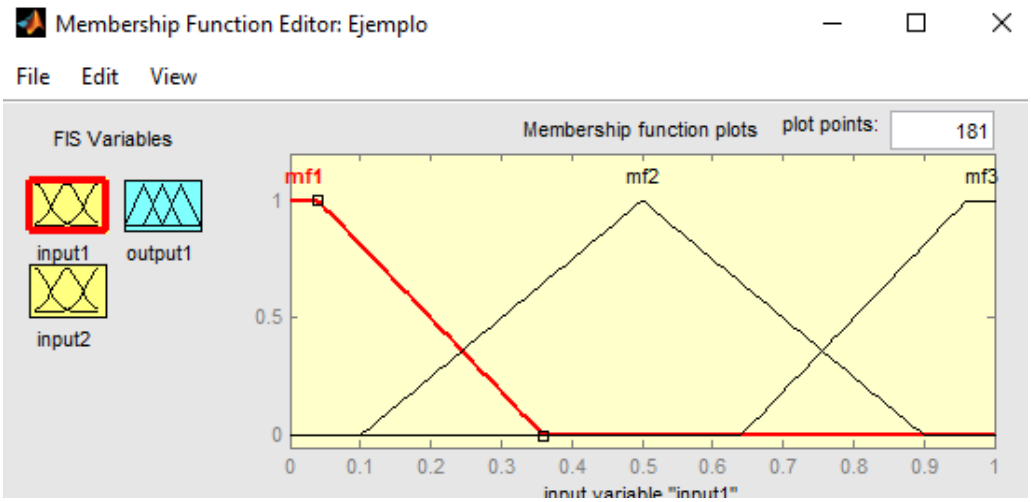


Ilustración 35 – Creación de funciones de pertenencia en Matlab

Con la función *plotmf*, se pueden representar las funciones de pertenencia de cada una de las entradas y salidas.

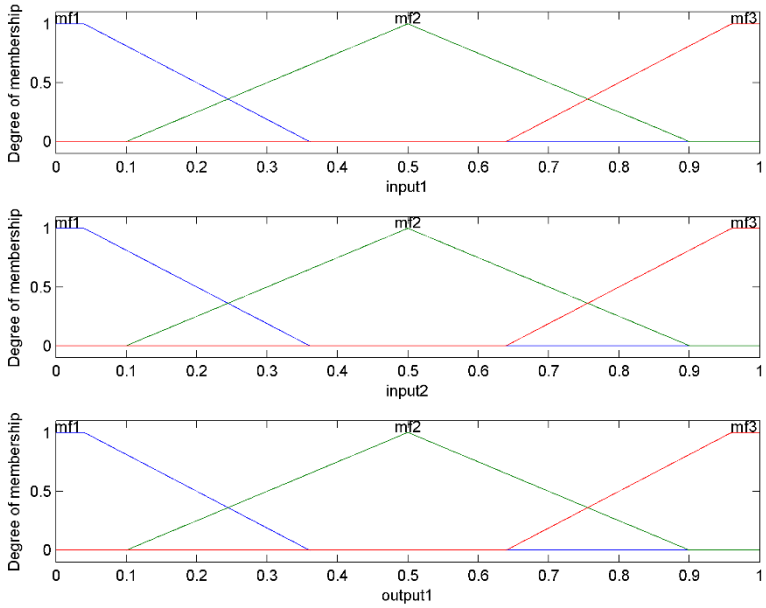


Ilustración 36 - Funciones de pertenencia del ejemplo en Matlab

El Toolbox tiene la propiedad de generar el arichivo fis. Matlab codifica los archivos .fis según se muestra en el siguiente cuadro de código

```
[System]
Name='Ejemplo'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=9
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='input1'
Range=[0 1]
NumMFs=3
MF1='mf1':'trapmf',[-0.36 -0.04 0.04 0.36]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trapmf',[0.64 0.96 1.04 1.36]
```

```
[Input2]
Name='input2'
Range=[0 1]
NumMFs=3
MF1='mf1':'trapmf',[-0.36 -0.04 0.04 0.36]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trapmf',[0.64 0.96 1.04 1.36]

[Output1]
Name='output1'
Range=[0 1]
NumMFs=3
MF1='mf1':'trapmf',[-0.36 -0.04 0.04 0.36]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trapmf',[0.64 0.96 1.04 1.36]

[Rules]
1 1, 1 (1) : 1
1 2, 3 (1) : 1
1 3, 3 (1) : 1
2 1, 3 (1) : 1
2 2, 2 (1) : 1
2 3, 1 (1) : 1
3 1, 1 (1) : 1
3 2, 1 (1) : 1
3 3, 3 (1) : 1
```

En primer lugar se define *[System]* que incluye las propiedades del FIS. Según el número de entradas y salidas, *NumInputs* y *NumOutputs*, se tendrá que declarar las N variables de entrada y M variables de salida como *[InputN]*, *[OutputM]* para cada una de las entradas y salidas, y finalmente las reglas del sistema como *[Rules]*.

La definición de las reglas sigue el siguiente modelo:

$In_1\ In_2 \dots In_N, MF_{outM}\ (Out_M) : 1$ ⟶ Peso 1

Para 2 3, 1 (1) : 1 , significa que si para la Entrada 1 lo sitúa en la MF 2 y para la Entrada 2 lo sitúa en la MF 3, se tomará la MF 1 de la Salida 1

MF: Membership Function

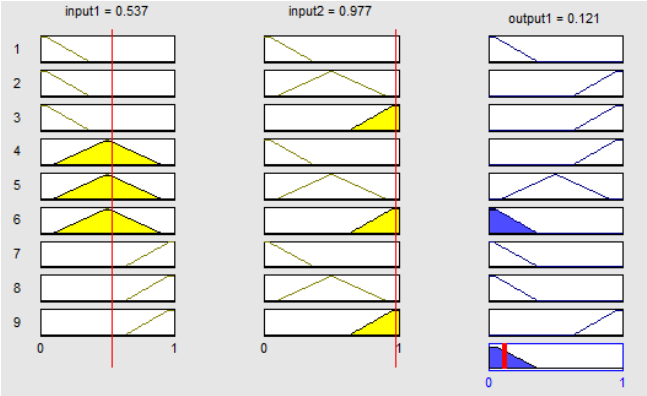


Ilustración 37 - Vista de reglas

Capítulo 7
Algorítmos Genéticos

Los algoritmos genéticos pertenecen al dominio de la computación evolutiva y representan un enfoque para la solución de problemas de optimización, donde la búsqueda se realiza en un espacio de solución y, de forma escalable, se conservan las soluciones más adecuadas en un proceso análogo a evolución biológica, por selección natural, cruzamiento y mutación

En la naturaleza, los individuos compiten por el derecho a la reproducción, a través de un proceso de habilidad-competencia. Los individuos más adecuados se reproducen generando descendientes que tienen su material genético y por lo tanto sus características de rendimiento. Finalmente, se produce una mutación en la progenie, esto no se entiende en términos de evolución como un error, sino como un intento abrupto en el proceso, para evitar el estancamiento en la generación de genes deseables.

Los algoritmos genéticos trabajan con una población de individuos generados al azar y cada uno representa una posible solución al problema. A través de una calificación asignada a cada uno de ellos llamado **aptitud**, se determina cómo es apropiado el individuo es resolver el problema. De acuerdo con esta calificación, los mejores individuos se mantienen para la próxima generación, llamada **hijos elite**. Otros individuos seleccionados aleatoriamente en la población son interdependientes por una operación matemática, dando lugar a los Hijos cruzados, mientras que cada **hijo mutante o mutado** es el resultado de una modificación aleatoria de su material genético.

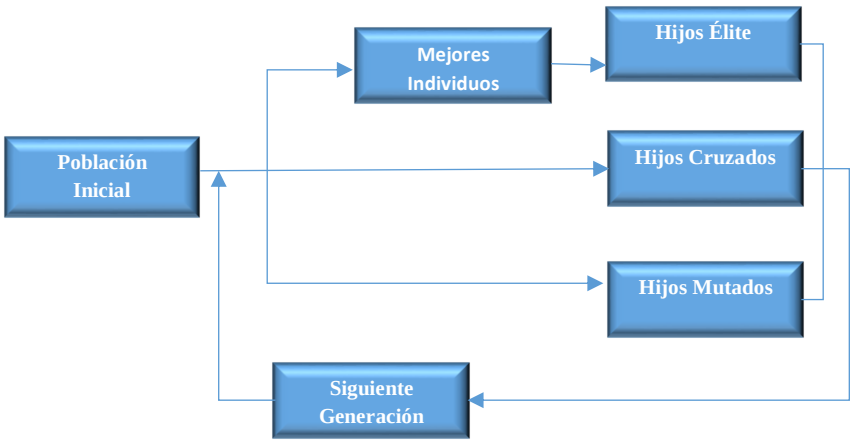


Ilustración 38 - Mecanismo de Algoritmo Genético

Dependiendo de sus características, los algoritmos genéticos difieren de los algoritmos clásicos en los que trabajan con un conjunto de soluciones para cada iteración y las soluciones que se ajusten al conjunto en la siguiente iteración se seleccionan mediante cálculos que incluyen valores aleatorios y no las funciones deterministas como en el enfoque clásico.

Tabla 20 - Tabla comparativa de algorítmos

Algoritmo Clásico	Algoritmo Genético
Genera un único punto en cada iteración. La secuencia de puntos se aproxima a una solución óptima	Genera una población de puntos en cada iteración. El mejor punto de la población se aproxima a una solución óptima
Selecciona el siguiente punto de la secuencia mediante un cálculo determinista.	Selecciona la siguiente población mediante un cálculo que emplea generadores de números aleatorios

En este proyecto los algoritmos genéticos se utilizan en entorno Matlab para optimizar paramétricamente las funciones de pertenencia de los controladores basados en sistemas de lógica difusa que se ven aplicadas el capítulo 8.

Algoritmos Genéticos con Matlab

Para usar el **Solver ga**, necesitamos proporcionar al menos dos argumentos de entrada, una función de aptitud (fitness function) y el número de variables en el problema. Los primeros dos argumentos de salida devueltos por ga son x, el mejor punto encontrado, y Fval, el valor de la función en el mejor punto. Un tercer argumento de salida, exitFlag le indica la razón por la que se detuvo. ga también puede devolver un cuarto argumento, Output, que contiene información sobre el rendimiento del solver.

X = ga (problem) encuentra el mínimo para *problem*.

Problem es una estructura que tiene los siguientes campos:

- fitnessfcn: <Función Fitness>
- nvars: <Número de variables de diseño>
- Aineq: <Una matriz para restricciones de desigualdad>
- bineq: <b vector para restricciones de desigualdad>
    - Aeq: <matriz Aeq para restricciones de igualdad>
    - beq: <vector beq para restricciones de igualdad>

- lb: <Límite inferior en X>
- ub: <Límite superior en X>

- nonlcon: <Función de restricción no lineal>
- intcon: <vector de índice para variables enteras>
- options: <Estructura de opciones creada con GAOPTIMSET>
- rngstate: <Estado del generador de números aleatorios>

Opciones para ga , Integer ga y gamultiobj

Tabla 21 - Opciones para algorítmo genético

Opción	Descripción
ConstraintTolerance	Determina la viabilidad con respecto a restricciones no lineales. Además, max(sqrt(eps),ConstraintTolerance) determina la viabilidad con respecto a las restricciones lineales. *Para gaoptimset , utilice TolCon .*
CreationFcn	Función que crea la población inicial.
CrossoverFcn	Función que el algoritmo utiliza para crear hijos cruzados
CrossoverFraction	La fracción de la población en la siguiente generación, sin incluir a los niños de élite, que se crea por la función de cruce.
Display	Nivel de visualización.
DistanceMeasureFcn	Función que calcula la medida de distancia de los individuos. El valor se aplica a la variable de decisión o espacio de diseño (genotipo) o al espacio de función (fenotipo).
EliteCount	Número entero positivo que especifica cuántos individuos en la generación actual están garantizados para sobrevivir a la siguiente generación. *No se utiliza en gamultiobj .*
FitnessLimit	Si la función de aptitud alcanza el valor de FitnessLimit , el algoritmo se detiene.
FitnessScalingFcn	Función que escala los valores de la función de aptitud. *Opción no disponible para gamultiobj .*
FunctionTolerance	El algoritmo se detiene si el cambio relativo medio del mejor valor de la función de aptitud sobre MaxStallGenerations generaciones de MaxStallGenerations es menor o igual que FunctionTolerance . Si StallTest es 'geometricWeighted' , entonces el algoritmo se detiene si el cambio relativo promedio *ponderado* es menor o igual que FunctionTolerance . *Para gaoptimset , utilice TolFun .*
HybridFcn	Función que continúa la optimización después de que ga termine. Alternativamente, una matriz de células que especifica la función híbrida y su estructura de opciones.
InitialPenalty	Valor inicial del parámetro de penalización
InitialPopulationMatrix	Población inicial utilizada para sembrar el algoritmo genético *Para gaoptimset , use InitialPopulation .*
InitialPopulationRange	Matriz o vector que especifica el rango de individuos en la población inicial. *Para gaoptimset , use PopInitRange .*
InitialScoresMatrix	Puntuaciones iniciales utilizadas para determinar la aptitud. *Para gaoptimset , use InitialScores .*
MaxGenerations	Número máximo de iteraciones antes de que el algoritmo se detenga.

	Para gaoptimset , utilizar Generations .
MaxStallGenerations	El algoritmo se detiene si el cambio relativo medio del mejor valor de la función de aptitud sobre MaxStallGenerations generaciones de MaxStallGenerations es menor o igual que FunctionTolerance . Si StallTest es 'geometricWeighted' , entonces el algoritmo se detiene si el cambio relativo promedio ponderado es menor o igual que FunctionTolerance . *Para gaoptimset , utilice StallGenLimit .*
MaxStallTime	El algoritmo se detiene si no hay mejoría en la función objetivo para MaxStallTime segundos. *Para gaoptimset , utilice StallTimeLimit .*
MaxTime	El algoritmo se detiene después de ejecutar después de segundos MaxTime. Este límite se aplica después de cada iteración, por lo que ga puede superar el límite *Para gaoptimset , utilice TimeLimit .*
MigrationDirection	Dirección de migración.
MigraciónFracción	Escalar entre 0 y 1 especificando la fracción de individuos en cada subpoblación que migra a una subpoblación diferente.
MigrationInterval	Número entero positivo que especifica el número de generaciones que tienen lugar entre migraciones de individuos entre subpoblaciones.
MutationFcn	Función que produce mutación.
NonlinearConstraintAlgorithm	Algoritmo de restricción no lineal. *Opción inalterable para gamultiobj.* *Para gaoptimset , utilice NonlinConAlgorithm .*
OutputFcn	Funciones que ga llama en cada iteración. *Para gaoptimset , use OutputFcns .*
ParetoFraction	Sólo para gamultiobj. Escalar entre 0 y 1 especificando la fracción de individuos para mantener la frontera de Pareto
PenaltyFactor	Parámetro de actualización de penalización.
PlotFcn	Funciones que trazan datos calculados por el algoritmo *Para gaoptimset , use PlotFcns .*
PlotInterval	Número entero positivo que especifica el número de generaciones entre llamadas consecutivas a las funciones de plot.
PopulationSize	Tamaño de la población.
PopulationType	Tipo de datos de la población. Debe ser 'doubleVector' para problemas enteros mixtos.
SelectionFcn	Funcion que selecciona a padres de cruce e hijos mutados. *gamultiobj utiliza solamente @selectiontournament .*
StallTest	Tipo de prueba de parada.
UseParallel	Calcule la aptitud y las funciones de restricción no lineales en paralelo.
UseVectorized	Especifica si las funciones están vectorizadas. *Para gaoptimset , utilice Vectorized con los valores 'on' o 'off' .*

Global Optimization Toolbox proporciona funciones que buscan soluciones globales a problemas que contienen múltiples máximos o mínimos. La caja de herramientas incluye búsqueda global, multistart, búsqueda de patrones, algoritmo genético, algoritmo genético multiobjetivo, recocido simulado y solucionadores de enjambres de partículas. Puede utilizar estos solucionadores para resolver problemas de optimización donde la función objetivo o de restricción es continua, discontinua, estocástica, no posee derivados o incluye simulaciones o funciones de caja negra.

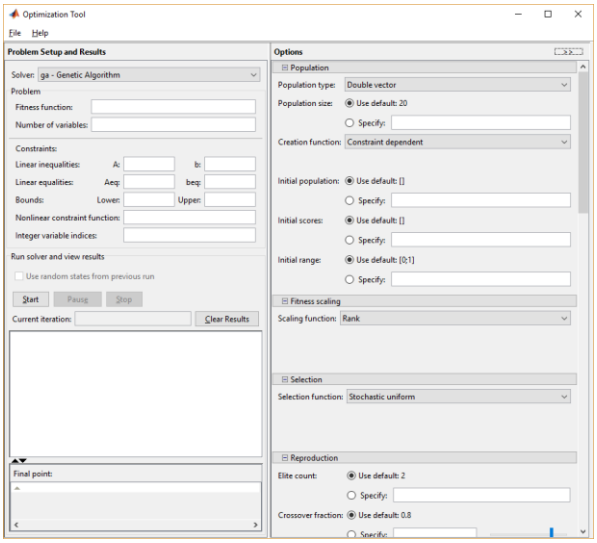


Ilustración 39 - Optimization Tool

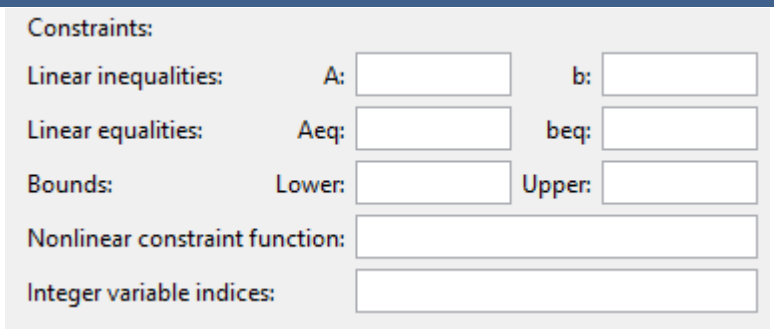
Se puede mejorar la eficacia del solucionador estableciendo opciones y personalizando las funciones de creación, actualización y búsqueda. Permite utilizar tipos de datos personalizados con el algoritmo genético y los solucionadores de recocido simulados para representar problemas que no se expresan fácilmente con tipos de datos estándar. La opción de función híbrida permite mejorar una solución aplicando un segundo solucionador después del primero.

El modo de proceder para realizar una optimización basada en algoritmos genéticos mediante *Optimization Tool* es ingresar los siguientes datos:

- **Fitness function** es la función objetivo que se desea minimizar. Esta función se especifica de la forma @objfun, donde objfun.m es un M-file que retorna un escalar.

- **Number of variables** es el número de variables de decisión

- **Constraints, l**as restricciones pueden ser lineales con signo de desigualdad, lineales con signo de igualdad y no lineales:



- **Linear inequalities** de la forma *A*x ≤ b* son definidas por la matriz A y el vector b.

  $4X(1) + 2X(2) \leq 15$, *sería :*

  A=[4 2];

  B=15;

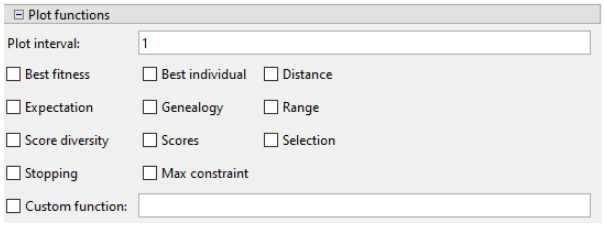
- **Linear equalities** de la forma *Aeq*x = beq* son definidas por la matriz *Aeq* y el vector *beq*
- **Bounds** son limites de las variables
- **Lower =** límite mínimo (en forma de vector).
- **Upper =** límite máximo (en forma de vector)

- **Nonlinear constraint function** define las restricciones no lineales. Se debe especificar la función de la forma @nonlcon, donde nonlcon.m es un archivo .m que retorna los vectores *c* y *ceq*. Las igualdades no lineales son de la forma *ceq = 0*, y las desigualdades son de la forma $c \leq 0$.
- **Plot Functions**, permite dibujar varias características de los AG a medida que se va ejecutando

  .

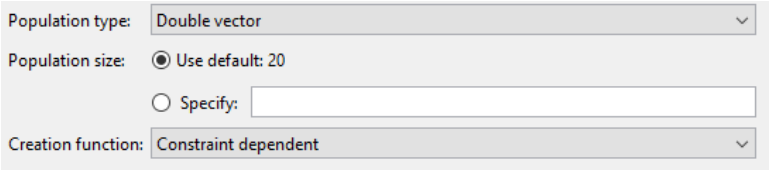
  - ***Plot interval,*** especifica el número de generaciones entre actualizaciones sucesivas de la gráfica.
  - ***Best fitness,*** dibuja el mejor valor de aptitud frente al número de iteración.
  - ***Best individual,*** dibuja los valores de x del mejor individuo la generación.
  - ***Scores,*** grafica los valores de los individuos en cada generación.
  - ***Max constraint,*** grafica la máxima violación de la restricción no lineal.

- **Range**, grafica los valores de aptitud mínima, máximo y medio en cada generación.
- ***Selection***, grafica un histograma de los padres.



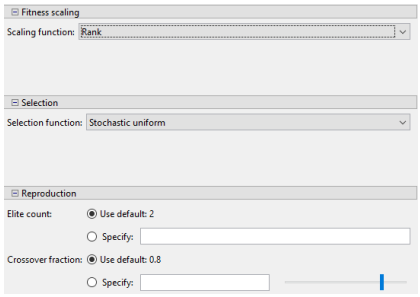
Options, permite especificar opciones especiales:

- **Population options**
  - ***Population type*** especifica el tipo de individuos. Puede ser *double vector*, o *bit string*, o *custom*.
  - ***Population size*** especifica cuántos individuos habrá en cada generación
  - ***Creation function*** especifica la función de distribución para crear la población inicial: por defecto crea la población aleatoriamente siguiendo una función uniforme.



Fitness scaling, convierte valores de aptitud puros a valores en un rango, que es usado luego por la función de selección.

- **Scaling function** especifica la función que hace el reescalamiento:
  - ***Rank*** reescala con base en el orden (rank) de cada individuo. El orden de cada individuo es su posición en la lista ordenada de aptitudes
  - ***Proportional*** reescala proporcionalmente a los valores de aptitud.



- **Selection**, elige padres para la próxima generación con base en sus aptitudes reescaladas:

- ***Roulette*** simula una ruleta circular con el área de cada segmento proporcional a su aptitud reescalada. Luego usa un número aleatorio para seleccionar una de las secciones con una probabilidad igual a su área.
- ***Tournament*** selecciona cada padre eligiendo "**Tournament size"** individuos aleatoriamente y luego eligiendo el mejor individuo de ellos como padre.

Selection, elige los padres para la próxima generación con base en sus aptitudes reescaladas:

- **Roulette** simula una ruleta circular con el área de cada segmento proporcional a su aptitud reescalada. Luego usa un número aleatorio para seleccionar una de las secciones con una probabilidad igual a su área.
- **Tournament** selecciona cada padre eligiendo *Tournament size* individuos aleatoriamente y luego eligiendo el mejor individuo de ellos como padre.

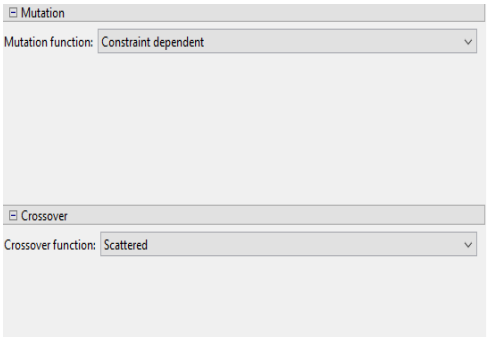
Reproduction, determina como el AG crea hijos en cada nueva generación

- **Elite count** especifica el número de individuos que se garantizan para sobrevivir a la próxima generación.
- **Crossover fraction** especifica qué fracción de la próxima generación, aparte de los elitistas, serán producidos por cruzamiento.

Mutation, especifica la función que hace la mutación:

- **Gaussian** adiciona un número aleatorio a cada término de un individuo. Este número es tomado de una distribución Gaussiana con media cero y varianza definida en **Scale** en la primera generación. El parámetro **Shrink** controla la forma en que la varianza disminuye de generación en generación. Si **Shrink** =0, la varianza es constante.
- **Uniform** es un proceso de dos pasos. Primero, el algoritmo selecciona una fracción de los genes del vector de un individuo para ser mutados, en la que cada gen tiene la misma probabilidad de ser mutado (según la probabilidad de mutación). En el segundo paso, el algoritmo reemplaza cada gen seleccionado por un número aleatorio seleccionado uniformemente de un rango de entrada.

- **Adaptive feasible** aleatoriamente genera direcciones que se adaptan respecto a los éxitos o fracasos de la última generación.



Crossover, combina 2 individuos padres para formar un nuevo hijo para la próxima generación Aquí se especifica la función a usar:

- **Scattered** crea aleatoriamente un vector binario: para cada gen selecciona de cuál padre va a formarse el hijo: si es 1 es del primer padre y si es 0 del Segundo.
- **Single point** elige aleatoriamente un entero n entre 1 y **Number of variables** para hacer el corte, y selecciona los genes enumerados menor que o igual a n del primer padre y los demás del segundo padre.

Capítulo 8

Ensayos y Resultados del Sistema Desarrollado

En este capítulo se detallan los pasos que se han dado para llevar a cabo el sistema desarrollado. Se estructura en varias partes, en primer lugar cada uno de los ensayos y optimizaciones llevadas a cabo en el desarrollo de los sistemas de control de estabilización de profundidad y posiciones angulares desarrollados en Matlab. Le sigue la parte de software realizado en Visual Basic y C#, y la programación de la placa Arduino.

Ensayos y resultados de los sistemas de control desarrollados en Matlab

Optimización para controlador estabilización de profundidad

Se diseña una función que realiza una simulación del modelo con un control a lazo abierto con una duracion de 60 segundos con un paso de 10 milisegundos. Como argumento de entrada establecemos un parametro que indica las revoluciones de los motores que afectan al eje Z. Esta función devuelve un vector con los valores obtenidos de posición Z en cada una de las iteraciones.

Codigo Fuente 1 - Función 'barreZ'

```
function x=barreZ(Motor)
tf = 60;                              % Tiempo Final de la Simulación (s)
h = 0.01;                              % Tiempo de Muestreo
Niter = round(tf/h);                   % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorZ=0; ErrorAntZ=0;dErrorZ=0; EZ=0; dEZ=0;Z=0;

for n=1:Niter

    t = n*h;  %Realiza la iteración desde t=0 hasta el tiempo final

        T100(n,3)=Motor;
        T100(n,4)=Motor;

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
```

```
Estado=Estado+h*DerivEstado;
Z  = Estado(9) ;
end
x=Z;
end
```

Para realizar un barrido del dominio posible de revoluciones por minuto (rpm) de los motores se diseña un script que asignara a la función anterior un valor de velocidad a los motores. La intención es obtener un conocimiento de las consecuencias de aplicar un valor específico de rpm en los motores de profundidad del vehículo.

Codigo Fuente 2 - Script Barrido Motores Z

```
clc
clear all
close all

M=-4000:4000;
valorF=0;
for n=1:length(M)
    valorF(n)=barreZ(M(n));
end
plot(M,valorF)
```

Se realiza un barrido desde -4000 hasta 4000 rpm y se presentan los valores finales de profundad correspondiente a la velocidad asignada en cada iteración.

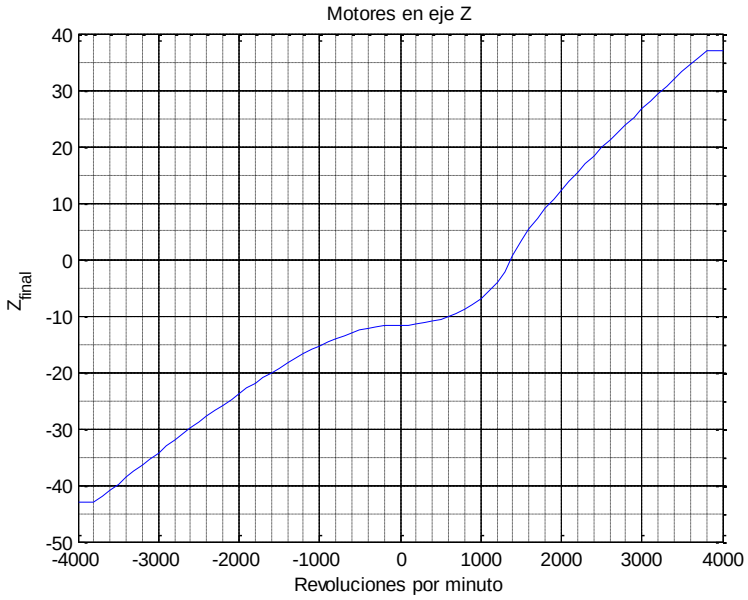


Ilustración 40 - Posicion final Z

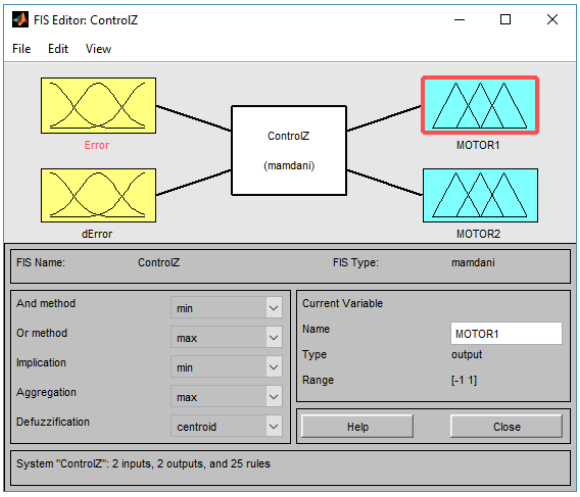
Se observó que en el valor de rpm que hace que se estabilice en cero el vehículo se puede obtener acotando:

```
M=1380:1385;
```

Ilustración 41 - Posicion final Z (zoom)

Se obtiene un valor nulo de Z, entre 1381 y 1382 RPM

Se hace uso de FIS Editor para diseñar un controlador fuzzy que sea capaz de estabilizar el vehículo a la profundidad deseada (Z). El tipo de control difuso que se emplea es incremental, es decir, las salidas dependen del estado en el que se encontraba en el instante anterior. Esto se realiza porque cuando la señal de error es nula, la salida es distinta de cero.



Se definen dos variables de salida que corresponde al error y a la derivada del error, que se definen como Error y dError; y dos salidas que corresponde al incremento de velocidad a aplicar a cada motor, que se definen como MOTOR1 y MOTOR2

El ensayo que se diseña se muestra en el siguiente código

Codigo Fuente 3 - Ensayo de control de profundidad

```
clear all
close all

C=-2;
tf = 30;                    % Tiempo Final de la Simulación (s)
h = 0.01;                   % Tiempo de Muestreo
Niter = round(tf/h);        % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorZ=0; ErrorAntZ=0;dErrorZ=0; EZ=0; dEZ=0;Z=0;


ConsignaZ=C;
for n=2:Niter

   t = n*h;       %Realiza la iteración desde t=0 hasta el tiempo final

      Mot=-LDout( ErrorZ,  dErrorZ,LDZ);
       T100(n,3)=T100(n-1,3)+Mot(1)*4000;
       T100(n,4)=T100(n-1,4)+Mot(1)*4000;
        Mot(1)
 if (T100(n,4)>4000),  T100(n,3)=4000;  T100(n,4)=4000; end
 if (T100(n,4)<-4000), T100(n,3)=-4000; T100(n,4)=-4000; end

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;
dErrorZ=ErrorZ-ErrorAntZ;
dErrorZ=dErrorZ/h;
```

```
   Zth=20;
   if dErrorZ> Zth, dErrorZ= Zth;end
   if dErrorZ<-Zth, dErrorZ=-Zth;end

    Z  = Estado(9) ;
    ErrorAntZ=ErrorZ;
    ErrorZ=ConsignaZ-Z;

EZ=[EZ ErrorZ];
end

mediaError=mean(EZ.^2);
mediaMotores=mean(diff(T100(:,3)).^2);
ts=tstab(EZ,(1:Niter-1)*h,0.01);
cost= [ts mediaError  mediaMotores];

subplot(211)
plot((1:Niter)*h,EZ)   ;
ylabel('Error')

subplot(212)
 plot((1:Niter)*h,T100(1:Niter,4))
 hold on;
 xlabel('Tiempo de simulación (s)')
 ylabel('Motores')
```

Diseño de un controlador basado en lógica difusa mediante comandos Matlab para estabilizar la profundidad del vehículo

Después de haber utilizado FIS Editor para generar sistemas de inferencia difusa, se opta por implementar una función capaz de generar un FIS mediante comandos, a partir de unos parámetros y sin utilizar la interfaz del Fuzzy Logic Toolbox.

La función genera dos entradas que corresponden al error y a la derivada del error de la posición Z (profundidad) con 5 funciones de pertenencia, las de los extremos de tipo trapezoidal y las funciones centrales de tipo triangular; y dos salidas con 9 funciones de pertenencia.

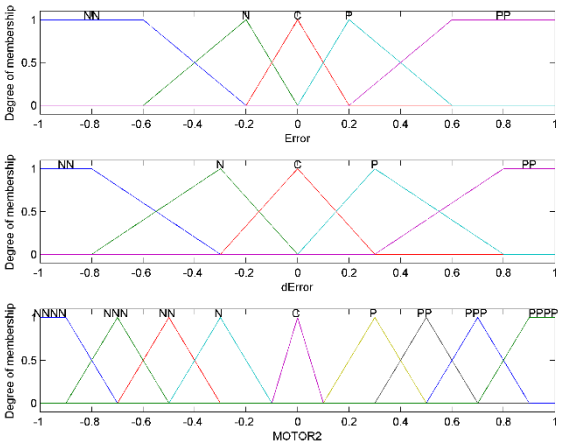


Ilustración 42 - Funciones de pertencia a optimizar

A continuación se muestra el código necesario para implementarla:

Codigo Fuente 4 - Funcion 'CrearFuzzyZPARAM'

```
function a= CrearFuzzyZPARAM(P2,P3,Q2,Q3,M2,M3,M4,M5,M6)

a=newfis('ControlZ');
P1=1;
a.input(1).name= 'ErrorZ'; a.input(1).range=[-P1 P1];
a.input(1).mf(1).name='NN'; a.input(1).mf(1).type='trapmf'; a.input(1).mf(1).params=[-P1 -P1 -P2 -P3];
a.input(1).mf(2).name='N'; a.input(1).mf(2).type='trimf'; a.input(1).mf(2).params=[-P2 -P3 0];
a.input(1).mf(3).name='C'; a.input(1).mf(3).type='trimf'; a.input(1).mf(3).params=[-P3 0 P3];
a.input(1).mf(4).name='P'; a.input(1).mf(4).type='trimf'; a.input(1).mf(4).params=[0 P3 P2];
a.input(1).mf(5).name='PP'; a.input(1).mf(5).type='trapmf'; a.input(1).mf(5).params=[P3 P2 P1 P1];

Q1=1;
a.input(2).name= 'dErrorZ'; a.input(2).range=[-Q1 Q1];
a.input(2).mf(1).name='NN'; a.input(2).mf(1).type='trapmf'; a.input(2).mf(1).params=[-Q1 -Q1 -Q2 -Q3];
a.input(2).mf(2).name='N'; a.input(2).mf(2).type='trimf'; a.input(2).mf(2).params=[-Q2 -Q3 0];
a.input(2).mf(3).name='C'; a.input(2).mf(3).type='trimf'; a.input(2).mf(3).params=[-Q3 0 Q3];
a.input(2).mf(4).name='P'; a.input(2).mf(4).type='trimf'; a.input(2).mf(4).params=[0 Q3 Q2];
a.input(2).mf(5).name='PP'; a.input(2).mf(5).type='trapmf'; a.input(2).mf(5).params=[Q3 Q2 Q1 Q1];

M1=1;
a.output(1).name= 'MOTOR1'; a.output(1).range=[-1 1];
a.output(1).mf(1).name='NNNN '; a.output(1).mf(1).type='trapmf'; a.output(1).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(1).mf(2).name='NNN '; a.output(1).mf(2).type='trimf'; a.output(1).mf(2).params=[-M2 -M3 -M4];
a.output(1).mf(3).name='NN '; a.output(1).mf(3).type='trimf'; a.output(1).mf(3).params=[-M3 -M4 -M5];
a.output(1).mf(4).name='N '; a.output(1).mf(4).type='trimf'; a.output(1).mf(4).params=[-M4 -M5 -M6];
a.output(1).mf(5).name='C '; a.output(1).mf(5).type='trimf'; a.output(1).mf(5).params=[-M6  0 M6];
a.output(1).mf(6).name='P '; a.output(1).mf(6).type='trimf'; a.output(1).mf(6).params=[M6 M5 M4];
a.output(1).mf(7).name='PP '; a.output(1).mf(7).type='trimf'; a.output(1).mf(7).params=[M5 M4 M3];
a.output(1).mf(8).name='PPP '; a.output(1).mf(8).type='trimf'; a.output(1).mf(8).params=[M4 M3 M2];
a.output(1).mf(9).name='PPPP'; a.output(1).mf(9).type='trapmf'; a.output(1).mf(9).params=[M3 M2 M1 M1];
```

```
a.output(2).name= 'MOTOR2'; a.output(2).range=[-1 1];
a.output(2).mf(1).name='NNNN '; a.output(2).mf(1).type='trapmf'; a.output(2).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(2).mf(2).name='NNN '; a.output(2).mf(2).type='trimf'; a.output(2).mf(2).params=[-M2 -M3 -M4];
a.output(2).mf(3).name='NN '; a.output(2).mf(3).type='trimf'; a.output(2).mf(3).params=[-M3 -M4 -M5];
a.output(2).mf(4).name='N '; a.output(2).mf(4).type='trimf'; a.output(2).mf(4).params=[-M4 -M5 -M6];
a.output(2).mf(5).name='C '; a.output(2).mf(5).type='trimf'; a.output(2).mf(5).params=[-M6  0 M6];
a.output(2).mf(6).name='P '; a.output(2).mf(6).type='trimf'; a.output(2).mf(6).params=[M6 M5 M4];
a.output(2).mf(7).name='PP '; a.output(2).mf(7).type='trimf'; a.output(2).mf(7).params=[M5 M4 M3];
a.output(2).mf(8).name='PPP '; a.output(2).mf(8).type='trimf'; a.output(2).mf(8).params=[M4 M3 M2];
a.output(2).mf(9).name='PPPP'; a.output(2).mf(9).type='trapmf'; a.output(2).mf(9).params=[M3 M2 M1 M1];



a.rule(1).antecedent= [1 1] ; a.rule(1).consequent=[9 9]; a.rule(1).weight=1; a.rule(1).connection=1;
a.rule(2).antecedent= [1 2] ; a.rule(2).consequent=[8 8]; a.rule(2).weight=1; a.rule(2).connection=1;
a.rule(3).antecedent= [1 3] ; a.rule(3).consequent=[7 7]; a.rule(3).weight=1; a.rule(3).connection=1;
a.rule(4).antecedent= [1 4] ; a.rule(4).consequent=[6 6]; a.rule(4).weight=1; a.rule(4).connection=1;
a.rule(5).antecedent= [1 5] ; a.rule(5).consequent=[5 5]; a.rule(5).weight=1; a.rule(5).connection=1;
a.rule(6).antecedent= [2 1] ; a.rule(6).consequent=[8 8]; a.rule(6).weight=1; a.rule(6).connection=1;
a.rule(7).antecedent= [2 2] ; a.rule(7).consequent=[7 7]; a.rule(7).weight=1; a.rule(7).connection=1;
a.rule(8).antecedent= [2 3] ; a.rule(8).consequent=[6 6]; a.rule(8).weight=1; a.rule(8).connection=1;
a.rule(9).antecedent= [2 4] ; a.rule(9).consequent=[5 5]; a.rule(9).weight=1; a.rule(9).connection=1;
a.rule(10).antecedent= [2 5] ; a.rule(10).consequent=[4 4]; a.rule(10).weight=1; a.rule(10).connection=1;
a.rule(11).antecedent= [3 1] ; a.rule(11).consequent=[7 7]; a.rule(11).weight=1; a.rule(11).connection=1;
a.rule(12).antecedent= [3 2] ; a.rule(12).consequent=[6 6]; a.rule(12).weight=1; a.rule(12).connection=1;
a.rule(13).antecedent= [3 3] ; a.rule(13).consequent=[5 5]; a.rule(13).weight=1; a.rule(13).connection=1;
a.rule(14).antecedent= [3 4] ; a.rule(14).consequent=[4 4]; a.rule(14).weight=1; a.rule(14).connection=1;
a.rule(15).antecedent= [3 5] ; a.rule(15).consequent=[3 3]; a.rule(15).weight=1; a.rule(15).connection=1;
a.rule(16).antecedent= [4 1] ; a.rule(16).consequent=[6 6]; a.rule(16).weight=1; a.rule(16).connection=1;
a.rule(17).antecedent= [4 2] ; a.rule(17).consequent=[5 5]; a.rule(17).weight=1; a.rule(17).connection=1;
a.rule(18).antecedent= [4 3] ; a.rule(18).consequent=[4 4]; a.rule(18).weight=1; a.rule(18).connection=1;
a.rule(19).antecedent= [4 4] ; a.rule(19).consequent=[3 3]; a.rule(19).weight=1; a.rule(19).connection=1;
a.rule(20).antecedent= [4 5] ; a.rule(20).consequent=[2 2]; a.rule(20).weight=1; a.rule(20).connection=1;
a.rule(21).antecedent= [5 1] ; a.rule(21).consequent=[5 5]; a.rule(21).weight=1; a.rule(21).connection=1;
a.rule(22).antecedent= [5 2] ; a.rule(22).consequent=[4 4]; a.rule(22).weight=1; a.rule(22).connection=1;
a.rule(23).antecedent= [5 3] ; a.rule(23).consequent=[3 3]; a.rule(23).weight=1; a.rule(23).connection=1;
a.rule(24).antecedent= [5 4] ; a.rule(24).consequent=[2 2]; a.rule(24).weight=1; a.rule(24).connection=1;
a.rule(25).antecedent= [5 5] ; a.rule(25).consequent=[1 1]; a.rule(25).weight=1; a.rule(25).connection=1;

a.defuzzMethod='centroid';

 end
```

La peculiaridad recae en que las funciones de pertenencia se pueden ajustar mediante unos parámetros los cuales se configuran a través de los argumentos de entrada.

La intención de parametrizar el FIS es la de optimizar cada una de las MF's mediante una función de entrenamiento y algoritmos genéticos.

En la tabla de reglas siguiente se muestra el comportamiento del sistema en función de las entradas

Tabla 22 - Tabla de reglas Z

	E --	**E -**	**E C**	**E +**	**E ++**
dE --	++++	+++	++	+	0
dE -	+++	++	+	0	-
dE C	++	+	0	-	--
dE +	+	0	-	--	---
dE ++	0	-	--	---	----

Las entradas del controlador son el error **E** y la derivada del error **dE**.

Implementación de una función de entrenamiento para la estabilización de profundidad del vehículo

Se trata de una simulación a la cual se le pasa un vector X que contiene los parámetros de las MF's comentadas en el apartado anterior, el cual genera un FIS que controla los motores encargados de propulsar el eje Z.

Codigo Fuente 5- Funcion 'evFL_Z'

```
function cost=evFL_Z(X)

C=-2;
tf = 30;                                  % Tiempo Final de la Simulación (s)
h = 0.01;                                  % Tiempo de Muestreo
Niter = round(tf/h);                          % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorZ=0; ErrorAntZ=0;dErrorZ=0; EZ=0; dEZ=0;Z=0;
LDZ=CrearFuzzyZPARAM(X(2),X(1),X(4),X(3),X(9),X(8),X(7),X(6),X(5));
ConsignaZ=C;
for n=2:Niter

    t = n*h;        %Realiza la iteración desde t=0 hasta el tiempo final

       Mot=-LDout( ErrorZ,  dErrorZ,LDZ);
        T100(n,3)=T100(n-1,3)+Mot(1)*4000;
        T100(n,4)=T100(n-1,4)+Mot(2)*4000;

 if (T100(n,3)>4000) T100(n,3)=4000; T100(n,4)=4000; end
 if (T100(n,3)<-4000) T100(n,3)=-4000; T100(n,4)=-4000; end

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;
dErrorZ=ErrorZ-ErrorAntZ;
dErrorZ=dErrorZ/h;

    Zth=20;
   if dErrorZ> Zth, dErrorZ= Zth;end
   if dErrorZ<-Zth, dErrorZ=-Zth;end

    Z  = Estado(9) ;
    ErrorAntZ=ErrorZ;
    ErrorZ=ConsignaZ-Z;

EZ=[EZ ErrorZ];
end

mediaError=mean(EZ.^2);
mediaMotores=mean(diff(T100(:,3)).^2);
ts=tstab(EZ,(1:Niter-1)*h,0.01);
cost= [ts mediaError  mediaMotores];

 end
```

La función de entrenamiento devuelve un vector con tres valores, la media del error al cuadrado, la media de la derivada de los valores al cuadrado asignados a los motores y el tiempo de estabilización.

El *tiempo de estabilización*, se calcula mediante la siguiente función que calcula el instante en que la señal queda comprendida en un intervalo [-p p].

Codigo Fuente 6 - Funcion 'tstab'

```
% E -> señal de entrada
% T -> vector de tiempos
% p -> umbral de establecimiento
% ----------------------------------
% ts -> tiempo de establecimiento
function ts=tstab(E,T,p)
c=length(E);
for n=1:length(E)
    if (E(n)<-p  | E(n)> p)
        c=n;
    end
end

ts= T(c-1);
```

Optimizacion mediante Algoritmos Genéticos Multiobjetivo

En esta sección se realiza una optimizacion haciendo uso de Optimization Tool de Matlab

El algoritmo genético se ejecuta a través de 'optimTool' del Toolbox de Matlab destinado a optimización.

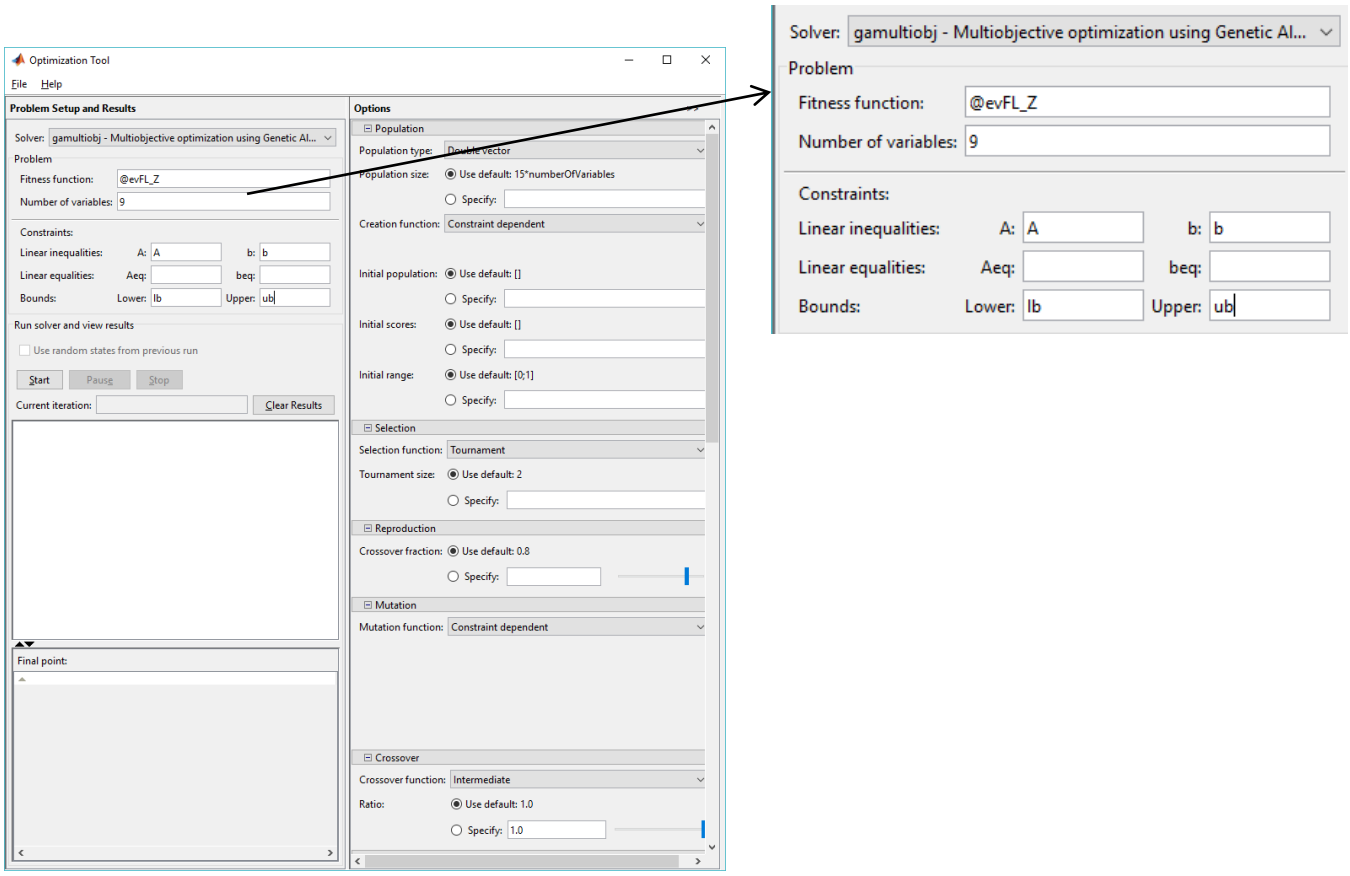
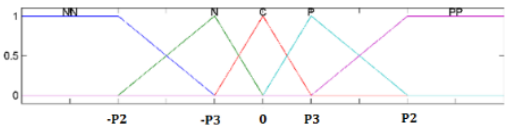


Ilustración 43- Parámetros de Optimización

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad b = \begin{pmatrix} -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \\ -0.001 \end{pmatrix}$$

Los parámetros A y b implican las siguientes restricciones:



$P_2 - P_3 \leq -0.001 \rightarrow P_2 > P_3$

$Q_2 - Q_3 \leq -0.001 \rightarrow Q_2 > Q_3$

$$M_2 - M_3 \leq -0.001 \quad \rightarrow \quad M_2 > M_3$$

$$M_3 - M_4 \leq -0.001 \quad \rightarrow \quad M_3 > M_4$$

$$M_4 - M_5 \leq -0.001 \quad \rightarrow \quad M_4 > M_5$$

$$M_5 - M_6 \leq -0.001 \quad \rightarrow \quad M_5 > M_6$$

$$M_2 > M_3 > M_4 > M_5 > M_6$$

$P_2 \leq -0.001 \quad \rightarrow \quad P_2 > 0$

$Q_2 \leq -0.001 \quad \rightarrow \quad Q_2 > 0$

$M_2 \leq -0.001 \quad \rightarrow \quad M_2 > 0$

Los parámetros lb y ub definen los valores limite inferior y superior respectivamente

$lb = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

$ub = (20 \quad 20 \quad 5 \quad 5 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$

Tras la ejecucion se obtuvo la siguiente solución:

$$Solucion = (4.5445 \quad 5.8122 \quad 1.0942 \quad 1.8221 \quad 0.095978 \quad 0.1176 \quad 0.12183 \quad 0.16927 \quad 0.33661)$$

Para verificar el funcionamiento, se muestran dos gráficas, la superior corresponde con las salidas de un barrido temporal con consignas que van desde 0 a -5 metros de profundidad. La grafica inferior corresponde a los valores de rpm que se aplican a los motores.

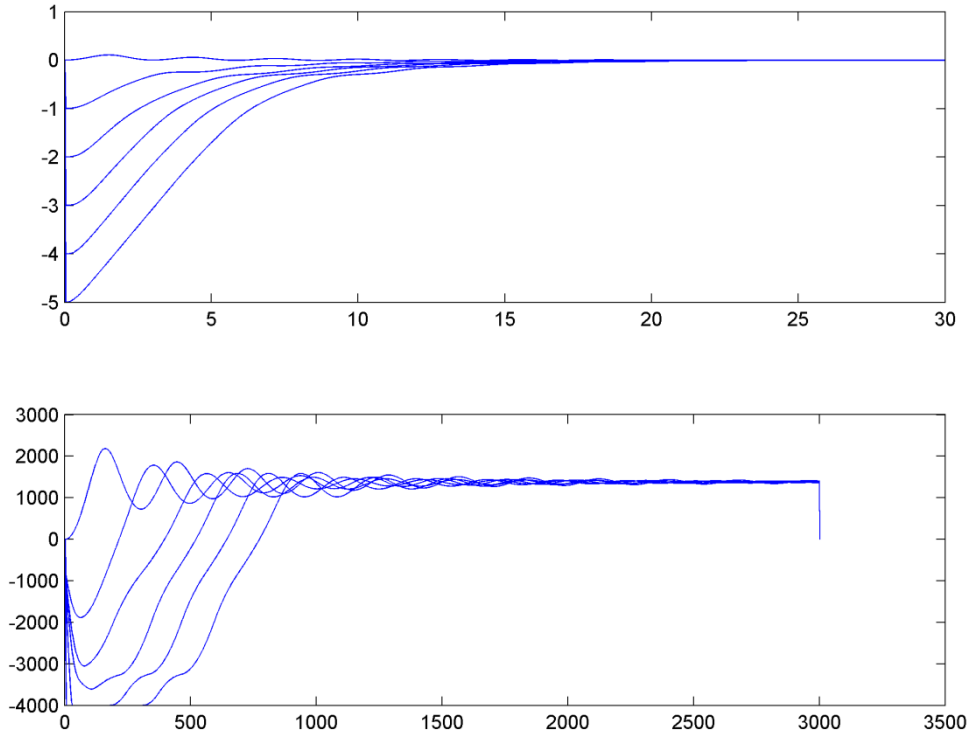


Ilustración 44 - Verificación de los parámetros obtenidos en Z

De entre los resultados, los mejores obtenidos son:

$$(4.5445 \quad 5.8122 \quad 1.0942 \quad 1.8221 \quad 0.095978 \quad 0.1176 \quad 0.12183 \quad 0.16927 \quad 0.33661)$$

Los valores aplicados a los motores correspondientes a la solución optimizada están en tanto por uno, si el valor máximo de RPM de los motores es 4000 RPM, corresponde a

383.9123 470.4135 487.3302 677.0957 (RPM), respectivamente

Las funciones de pertenencia resultantes correspondientes a los parámetros obtenidos a través de la optimización se muestras en las figuras sucesivas.

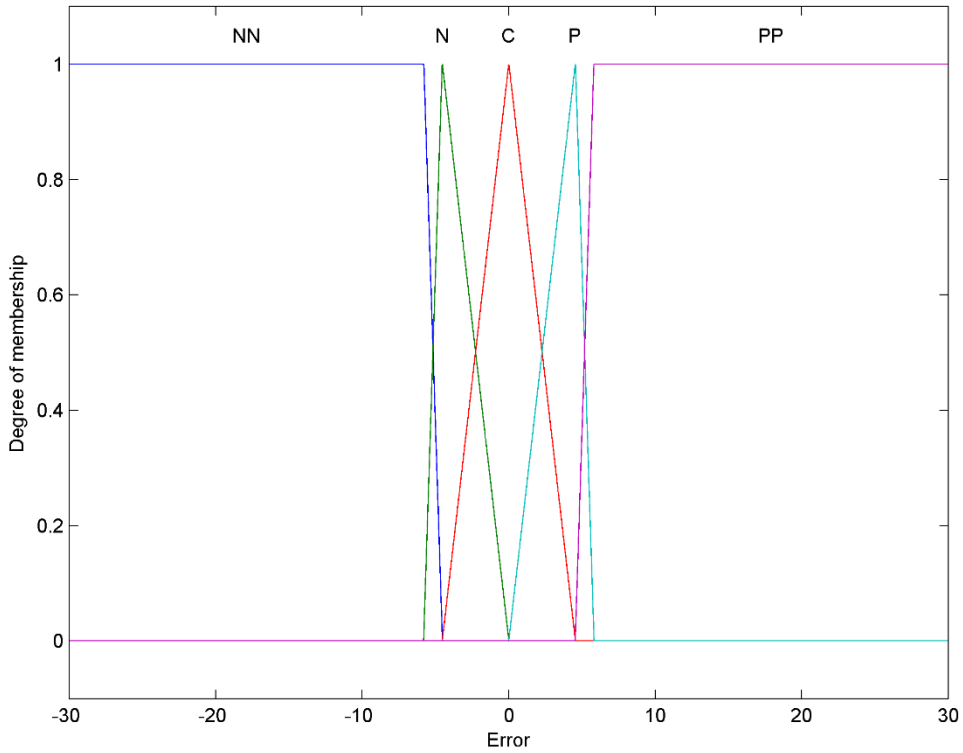


Ilustración 45 - Funciones de pertenencia para ErrorZ

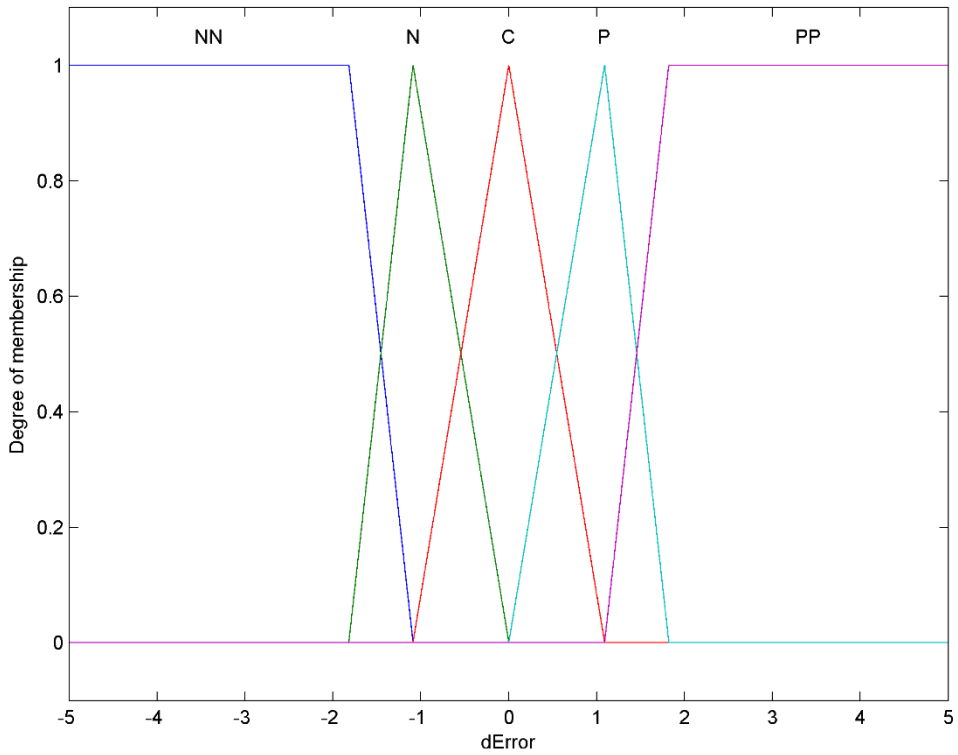


Ilustración 46 - Funciones de pertenencia para dErrorZ

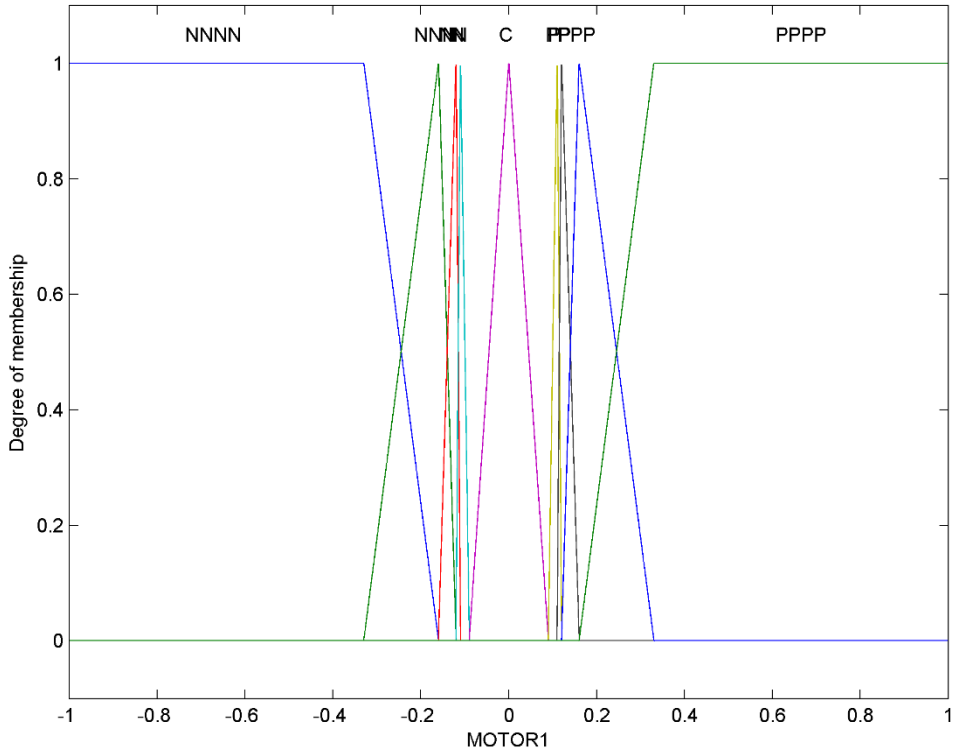


Ilustración 47 - Funciones de pertenencia para MotorZ

Optimización para controlador estabilización de ángulo THETA

Se diseña un script que realiza una simulación del modelo con un control a lazo abierto con una duracion de 60 segundos con un paso de 10 milisegundos. Realiza un barrido de la velocidad de los motores que afectan a la estabilidad del angulo \theta dando como resultado una gráfica del angulo pasado el tiempo de simulacion en funcion del valor de rpm aplicado a los motores

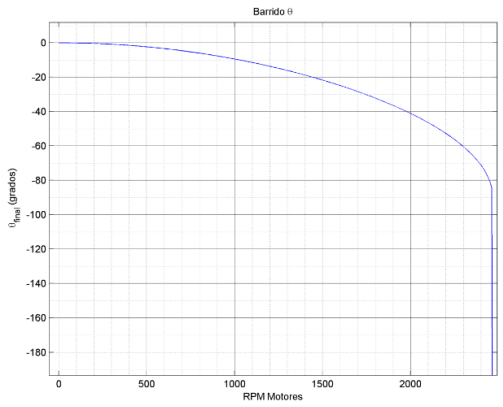


Ilustración 48 - Posición angular final de theta (zoom)

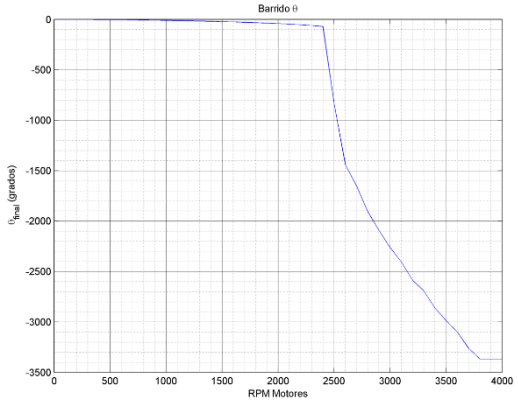


Ilustración 49 - Posición angular final de theta

Se observó que a partir de una velocidad entre 2400 y 2500 rpm el vehículo empezaba a rotar en torno a su centro

De manera análoga a la llevada a cabo en la optimización de Z, se define la función evFL_THETA, modificando algunos parámetros y teniendo en cuenta que las unidades a optimizar son unidades angulares.

Codigo Fuente 7 - Función 'evFL_THETA'

```
function cost=evFL_THETA(X)


C=150;
tf = 30;                                    % Tiempo Final de la Simulación
(s)
h = 0.05;                                      % Tiempo de Muestreo
Niter = round(tf/h);                           % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorTHETA=0;          ErrorAntTHETA=0;dErrorTHETA=0;          ETHETA=0;
dETHETA=0;THETA=0;
LDTHETA=CrearFuzzyTHETAPARAM(X(2),X(1),X(4),X(3),X(9),X(8),X(7),X(6)
,X(5));
ConsignaTHETA=C;
for n=2:Niter


    t = n*h;          %Realiza la iteración desde t=0 hasta el tiempo
final
if ErrorTHETA>360 ErrorTHETA=360;end
    if ErrorTHETA<-360 ErrorTHETA=-360;end
       Mot=-LDout( ErrorTHETA,  dErrorTHETA,LDTHETA);
        T100(n,3)=T100(n-1,3)+Mot(1)*4000;
        T100(n,4)=T100(n-1,4)-Mot(2)*4000;
```

```
 if (T100(n,3)>4000) T100(n,3)=4000; T100(n,4)=-4000; end
 if (T100(n,3)<-4000) T100(n,3)=-4000; T100(n,4)=+4000; end

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;
dErrorTHETA=ErrorTHETA-ErrorAntTHETA;
dErrorTHETA=dErrorTHETA/h;

    if ErrorTHETA>360 ErrorTHETA=360;end
    if ErrorTHETA<-360 ErrorTHETA=-360;end
    THETAth=20;
    if dErrorTHETA> THETAth, dErrorTHETA= THETAth;end
    if dErrorTHETA<-THETAth, dErrorTHETA=-THETAth;end

THETA  = rad2deg(Estado(11)) ;
    ErrorAntTHETA=ErrorTHETA;
    ErrorTHETA=ConsignaTHETA-THETA;

ETHETA=[ETHETA ErrorTHETA];
 end

mediaError=mean(ETHETA.^2);
mediaMotores=mean(diff(T100(:,3)).^2);
ts=tstab(ETHETA,(1:Niter-1)*h,0.01);
cost= [ts mediaError  mediaMotores];
end
```

La función `CrearFuzzyTHETAPARAM` genera dos entradas que corresponden al error y a la derivada del error del ángulo theta con 5 funciones de pertenencia, las de los extremos de tipo trapezoidal y las funciones centrales de tipo triangular; y dos salidas con 9 funciones de pertenencia.

Codigo Fuente 8 - Función 'CrearFuzzyTHETAPARAM'

```
function a= CrearFuzzyTHETAPARAM(P2,P3,Q2,Q3,M2,M3,M4,M5,M6)
```

```
a=newfis('ControlTHETA');
RangoE=180;
a.input(1).name= 'Error'; a.input(1).range=[-RangoE RangoE];
a.input(1).mf(1).name='NN';            a.input(1).mf(1).type='trapmf';
a.input(1).mf(1).params=[-RangoE -RangoE -P2 -P3];
a.input(1).mf(2).name='N';              a.input(1).mf(2).type='trimf';
a.input(1).mf(2).params=[-P2 -P3 0];
a.input(1).mf(3).name='C';              a.input(1).mf(3).type='trimf';
a.input(1).mf(3).params=[-P3 0 P3];
a.input(1).mf(4).name='P';              a.input(1).mf(4).type='trimf';
a.input(1).mf(4).params=[0 P3 P2];
a.input(1).mf(5).name='PP';            a.input(1).mf(5).type='trapmf';
a.input(1).mf(5).params=[P3 P2 RangoE RangoE];

Q1=10;


a.input(2).name= 'dError'; a.input(2).range=[-Q1 Q1];
a.input(2).mf(1).name='NN';            a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-Q1 -Q1 -Q2 -Q3];
a.input(2).mf(2).name='N';              a.input(2).mf(2).type='trimf';
a.input(2).mf(2).params=[-Q2 -Q3 0];
a.input(2).mf(3).name='C';              a.input(2).mf(3).type='trimf';
a.input(2).mf(3).params=[-Q3 0 Q3];
a.input(2).mf(4).name='P';              a.input(2).mf(4).type='trimf';
a.input(2).mf(4).params=[0 Q3 Q2];
a.input(2).mf(5).name='PP';            a.input(2).mf(5).type='trapmf';
a.input(2).mf(5).params=[Q3 Q2 Q1 Q1];

M1=16.5;


a.output(1).name= 'MOTOR1'; a.output(1).range=[-M1 M1];
a.output(1).mf(1).name='NNNN    ';    a.output(1).mf(1).type='trapmf';
a.output(1).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(1).mf(2).name='NNN     ';     a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[-M2 -M3 -M4];
a.output(1).mf(3).name='NN      ';     a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[-M3 -M4 -M5];
a.output(1).mf(4).name='N       ';     a.output(1).mf(4).type='trimf';
a.output(1).mf(4).params=[-M4 -M5 -M6];
a.output(1).mf(5).name='C       ';     a.output(1).mf(5).type='trimf';
a.output(1).mf(5).params=[-M6  0 M6];
a.output(1).mf(6).name='P       ';     a.output(1).mf(6).type='trimf';
a.output(1).mf(6).params=[M6 M5 M4];
a.output(1).mf(7).name='PP      ';     a.output(1).mf(7).type='trimf';
a.output(1).mf(7).params=[M5 M4 M3];
a.output(1).mf(8).name='PPP     ';     a.output(1).mf(8).type='trimf';
a.output(1).mf(8).params=[M4 M3 M2];
a.output(1).mf(9).name='PPPP';        a.output(1).mf(9).type='trapmf';
a.output(1).mf(9).params=[M3 M2 M1 M1];

a.output(2).name= 'MOTOR2'; a.output(2).range=[-M1 M1];
a.output(2).mf(1).name='NNNN    ';    a.output(2).mf(1).type='trapmf';
a.output(2).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(2).mf(2).name='NNN     ';     a.output(2).mf(2).type='trimf';
a.output(2).mf(2).params=[-M2 -M3 -M4];
```

```
a.output(2).mf(3).name='NN      ';      a.output(2).mf(3).type='trimf';
a.output(2).mf(3).params=[-M3 -M4 -M5];
a.output(2).mf(4).name='N       ';      a.output(2).mf(4).type='trimf';
a.output(2).mf(4).params=[-M4 -M5 -M6];
a.output(2).mf(5).name='C       ';      a.output(2).mf(5).type='trimf';
a.output(2).mf(5).params=[-M6  0 M6];
a.output(2).mf(6).name='P       ';      a.output(2).mf(6).type='trimf';
a.output(2).mf(6).params=[M6 M5 M4];
a.output(2).mf(7).name='PP      ';      a.output(2).mf(7).type='trimf';
a.output(2).mf(7).params=[M5 M4 M3];
a.output(2).mf(8).name='PPP     ';      a.output(2).mf(8).type='trimf';
a.output(2).mf(8).params=[M4 M3 M2];
a.output(2).mf(9).name='PPPP';        a.output(2).mf(9).type='trapmf';
a.output(2).mf(9).params=[M3 M2 M1 M1];



PINTAR='n';
    if PINTAR=='S' ,
        subplot(311)
        pintaFuzzy(a.input(1));
          subplot(312)
        pintaFuzzy(a.input(2));
          subplot(313)
        pintaFuzzy(a.output(1));
    end


a.rule(1).antecedent=    [1    1]    ;    a.rule(1).consequent=[9    9];
a.rule(1).weight=1; a.rule(1).connection=1;
a.rule(2).antecedent=    [1    2]    ;    a.rule(2).consequent=[9    9];
a.rule(2).weight=1; a.rule(2).connection=1;
a.rule(3).antecedent=    [1    3]    ;    a.rule(3).consequent=[9    9];
a.rule(3).weight=1; a.rule(3).connection=1;
a.rule(4).antecedent=    [1    4]    ;    a.rule(4).consequent=[8    8];
a.rule(4).weight=1; a.rule(4).connection=1;
a.rule(5).antecedent=    [1    5]    ;    a.rule(5).consequent=[7    7];
a.rule(5).weight=1; a.rule(5).connection=1;

a.rule(6).antecedent=    [2    1]    ;    a.rule(6).consequent=[7    7];
a.rule(6).weight=1; a.rule(6).connection=1;
a.rule(7).antecedent=    [2    2]    ;    a.rule(7).consequent=[6    6];
a.rule(7).weight=1; a.rule(7).connection=1;
a.rule(8).antecedent=    [2    3]    ;    a.rule(8).consequent=[6    6];
a.rule(8).weight=1; a.rule(8).connection=1;
a.rule(9).antecedent=    [2    4]    ;    a.rule(9).consequent=[5    5];
a.rule(9).weight=1; a.rule(9).connection=1;
a.rule(10).antecedent=   [2    5]    ;   a.rule(10).consequent=[5    5];
a.rule(10).weight=1; a.rule(10).connection=1;

a.rule(11).antecedent=   [3    1]    ;   a.rule(11).consequent=[6    6];
a.rule(11).weight=1; a.rule(11).connection=1;
a.rule(12).antecedent=   [3    2]    ;   a.rule(12).consequent=[5    5];
a.rule(12).weight=1; a.rule(12).connection=1;
a.rule(13).antecedent=   [3    3]    ;   a.rule(13).consequent=[5    5];
a.rule(13).weight=1; a.rule(13).connection=1;
a.rule(14).antecedent=   [3    4]    ;   a.rule(14).consequent=[5    5];
a.rule(14).weight=1; a.rule(14).connection=1;
```

```
a.rule(15).antecedent=   [3   5]   ;   a.rule(15).consequent=[4   4];
a.rule(15).weight=1; a.rule(15).connection=1;

a.rule(16).antecedent=   [4   1]   ;   a.rule(16).consequent=[5   5];
a.rule(16).weight=1; a.rule(16).connection=1;
a.rule(17).antecedent=   [4   2]   ;   a.rule(17).consequent=[5   5];
a.rule(17).weight=1; a.rule(17).connection=1;
a.rule(18).antecedent=   [4   3]   ;   a.rule(18).consequent=[4   4];
a.rule(18).weight=1; a.rule(18).connection=1;
a.rule(19).antecedent=   [4   4]   ;   a.rule(19).consequent=[4   4];
a.rule(19).weight=1; a.rule(19).connection=1;
a.rule(20).antecedent=   [4   5]   ;   a.rule(20).consequent=[3   3];
a.rule(20).weight=1; a.rule(20).connection=1;

a.rule(21).antecedent=   [5   1]   ;   a.rule(21).consequent=[3   3];
a.rule(21).weight=1; a.rule(21).connection=1;
a.rule(22).antecedent=   [5   2]   ;   a.rule(22).consequent=[2   2];
a.rule(22).weight=1; a.rule(22).connection=1;
a.rule(23).antecedent=   [5   3]   ;   a.rule(23).consequent=[1   1];
a.rule(23).weight=1; a.rule(23).connection=1;
a.rule(24).antecedent=   [5   4]   ;   a.rule(24).consequent=[1   1];
a.rule(24).weight=1; a.rule(24).connection=1;
a.rule(25).antecedent=   [5   5]   ;   a.rule(25).consequent=[1   1];
a.rule(25).weight=1; a.rule(25).connection=1;



a.defuzzMethod='centroid';

 end
```

Se prepara lo optimización mediante algoritmos genéticos, de manera similar a la practicada en el caso de Z. Las restricciones de las matrices A, b y lb son las mismas, cambia el caso de ub.

$$ub = (180 \quad 180 \quad 6 \quad 6 \quad 199 \quad 199 \quad 199 \quad 199 \quad 199)$$

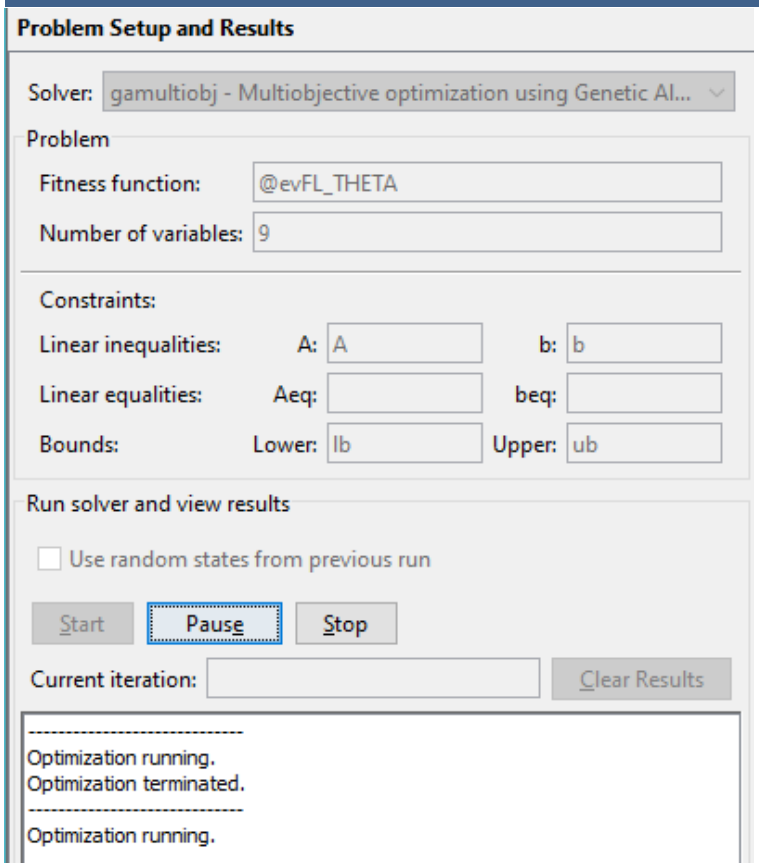


Ilustración 50 - Parametros optimizacion de theta

No se obtienen resultados que se puedan considerar óptimos con respecto a las señales de salida de los motores, por lo que se opta por buscar manualmente unos valores de iniciales para poder ejecutar el algoritmo genético.

Para ello se hace uso de un GUI diseñado en Matlab para tal fin:

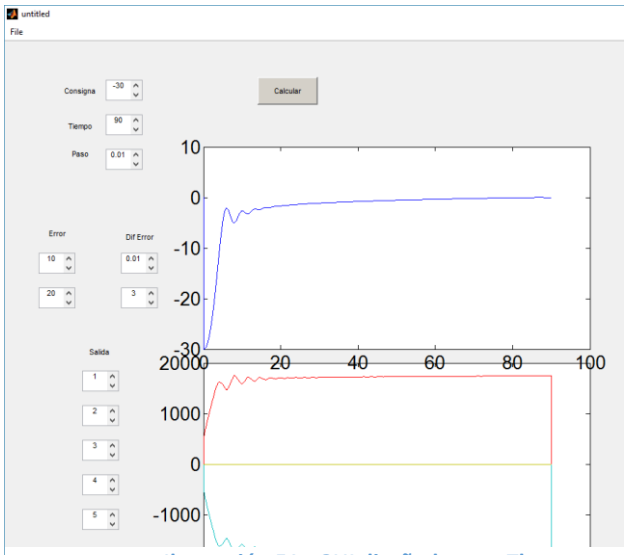


Ilustración 51 - GUI diseñado para Theta

Con esta configuración se obtienen respuestas estables en los motores, se utilizan estos valores como población inicial en la búsqueda de los valores óptimos.

```
optimosManual =
   10.0000   20.0000    0.0100    3.0000    1.0000    2.0000    3.0000    4.0000    5.0000
```

Con estos valores se obtienen una respuesta tal como se muestran en las siguientes gráficas.

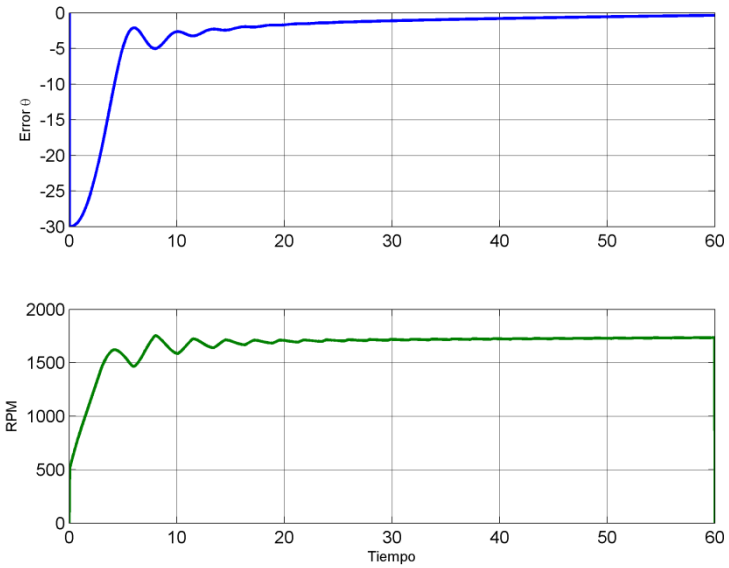


Ilustración 52 - Respuesta sistema con los valores encontrados manualmente

Con este punto de partida se vuelve a optimizar mediante algoritmo genético con una población de referencia.

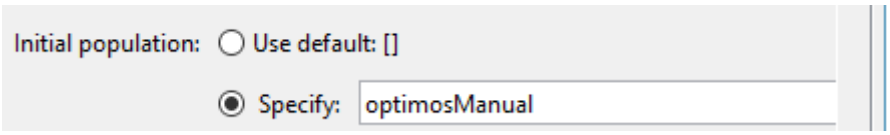


Ilustración 53 - Población Inicial en optimizacion de Theta

De los resultados obtenidos los parámetros que se consideran óptimos son:

```
mejor =

   10.2553   30.4531    0.2629    3.3259    1.8991    2.8108    2.9442    4.2545   15.4655
```

Las gráficas que se muestran a continuación muestran la respuesta del sistema y la señal de motores.

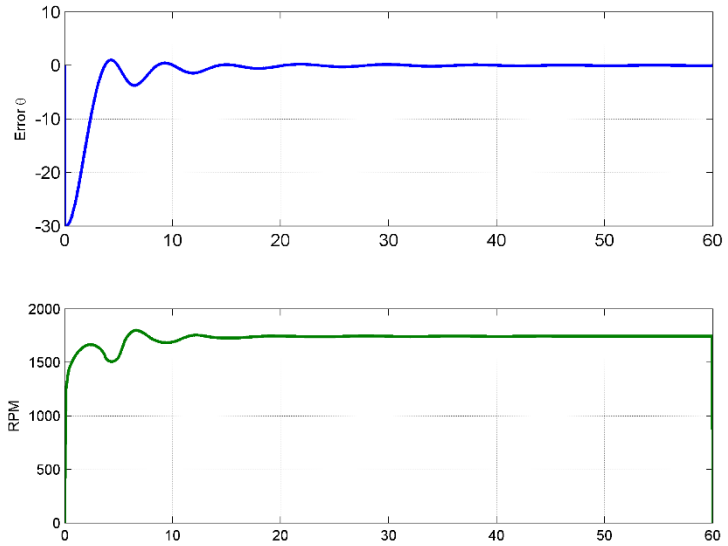


Ilustración 54 - Respuesta del sistema con parametros obtenidos

Se realiza una simulación para un conjunto de consignas, desde 0 hasta 90 grados en los que se encuentran efectos no deseables.

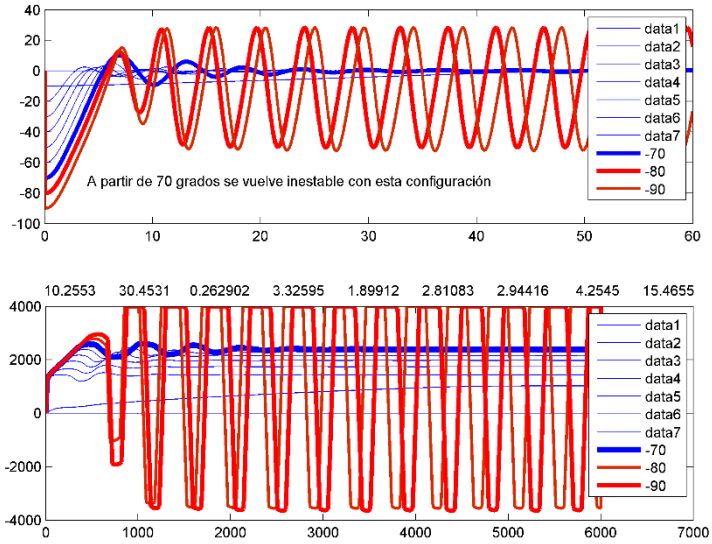


Ilustración 55 - Inestabilidades de angulo Theta

Para corregir esta inestabilidad se procede a realizar una limitación en la acción de los motores, cuyo resultado es el que sigue.

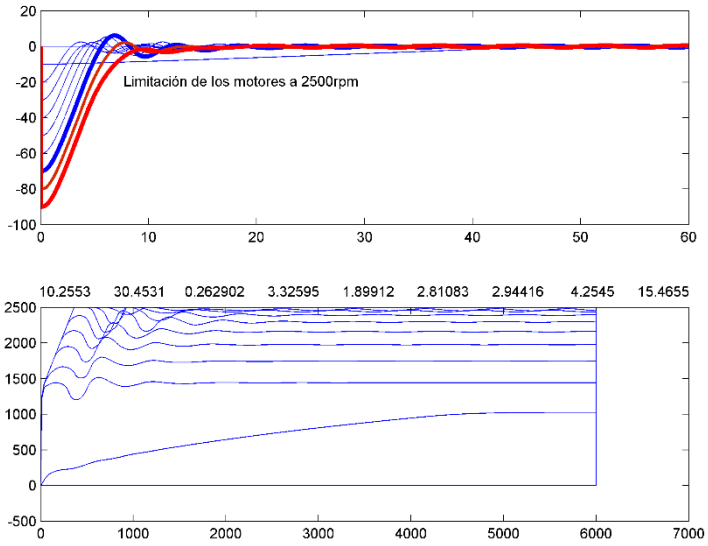


Ilustración 56 - Nueva respuesta del sistema con correcciones aplicadas

Se realiza una simulación en la que se establecen inicialmente unos valores del ángulo inicial theta, entre 0 y 180 grados, a lazo abierto con los motores parados (0 RPM), para estudiar el comportamiento dinámico del vehículo en cada uno de esos ángulos.

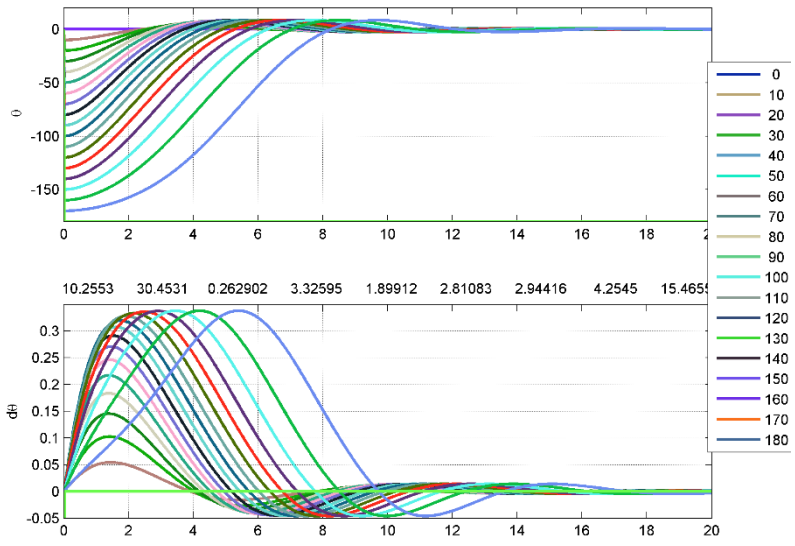


Ilustración 57 - Estudio angular en theta

Cuando el sistema se acerca a la zona de 180 grados, se vuelve inestable.

Optimización para controlador estabilización de ángulo THETA

Del mismo modo que en caso del ángulo theta se diseña un script que realiza una simulación del modelo con un control a lazo abierto con una duracion de 60 segundos con un paso de 10 milisegundos. Realiza un barrido de la velocidad de los motores que afectan a la estabilidad del angulo phi dando como resultado una gráfica del angulo pasado el tiempo de simulacion en funcion del valor de rpm aplicado a los motores

Codigo Fuente 9 - Función 'barrePHI'

```
function [x , dTH]=barrePHI(Motor)
tf = 60;                                % Tiempo Final de la Simulación
(s)
h = 0.01;                                  % Tiempo de Muestreo
Niter = round(tf/h);                          % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorPHI=0; ErrorAntPHI=0;dErrorPHI=0; EPHI=0; dEPHI=0;PHI=0;

for n=1:Niter

    t = n*h;          %Realiza la iteración desde t=0 hasta el tiempo
final

        T100(n,5)=Motor;
        T100(n,6)=-Motor;
[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;
dTH= rad2deg(Estado(10)) - PHI;
dTH=dTH/h;
PHI  = rad2deg(Estado(10)) ;
end
x=PHI;
end
```

```
clc
clear all
close all

 M=0:10:3800;

valorF=0;
der=0;
for n=1:length(M)
    [valorF(n) der(n)]=barrePHI(M(n));

end
plot(M,valorF);
figure
plot(M,der);
```

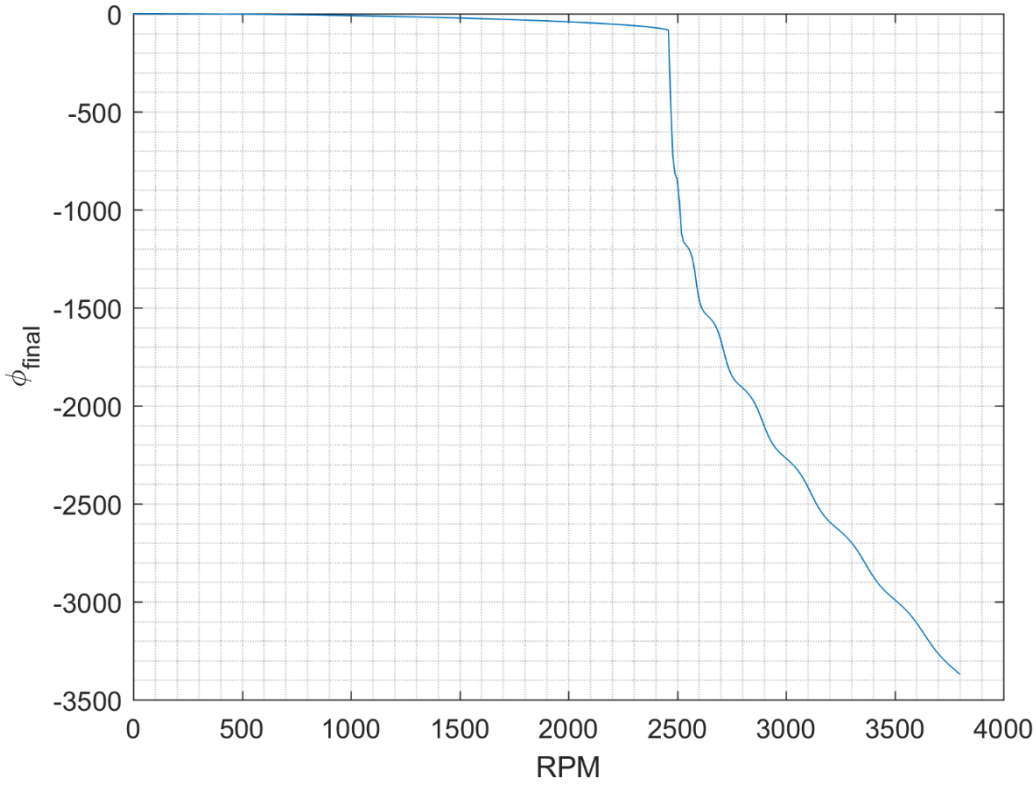


Ilustración 58 - Posicion angular final para phi

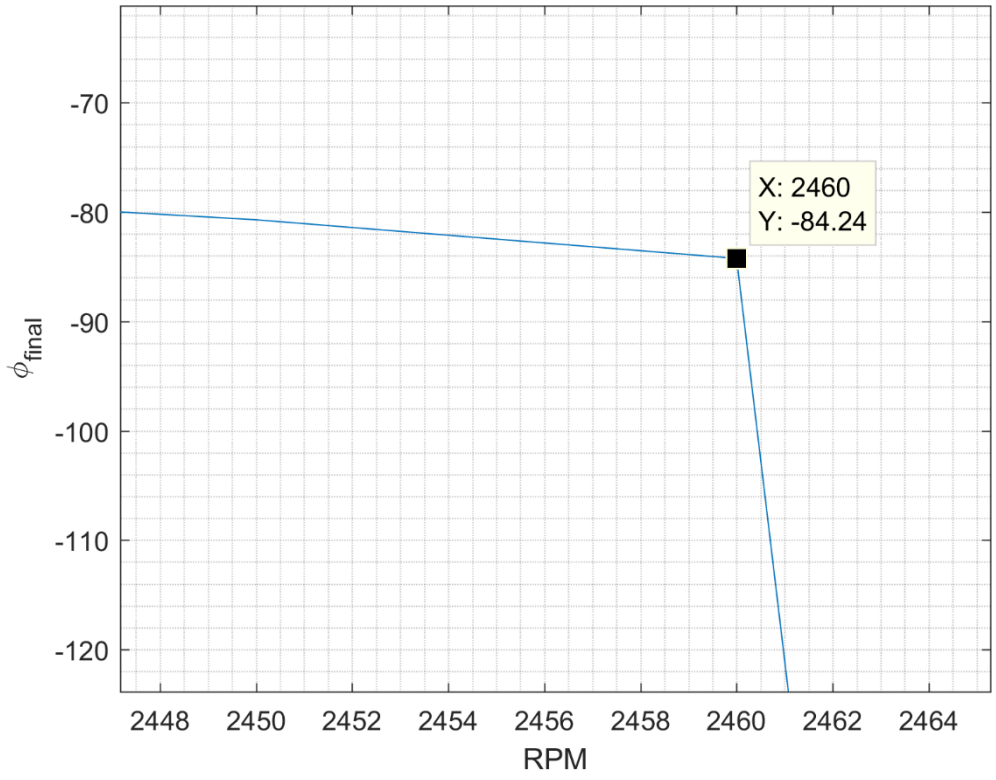


Ilustración 59 - Posicion angular final para phi (zoom)

Se encuentra el valor de velocidad aplicado en motores a partir del cual el vehiculo vence y empieza a rotar.

La función de entrenamiento en este caso es la que se muestra en el siguiente código:

Codigo Fuente 10- Función 'evFL_PHI'

```
function cost =evFL_PHI(X,C)

tf = 60;                                % Tiempo Final de la Simulación
(s)
h = 0.01;                                 % Tiempo de Muestreo
Niter = round(tf/h);                      % Numero de muestras

T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorPHI=0; ErrorAntPHI=0;dErrorPHI=0; EPHI=0; dEPHI=0;PHI=0;
%X= [24.6872   30.3741    0.0670    2.9110    1.4568    2.4627    3.1806
6.2492   16.0643];

LDPHI=CrearFuzzyPHIPARAM(X);

ConsignaPHI=C;
for n=2:Niter
    t = n*h; %Realiza la iteración desde t=0 hasta el tiempo final

while (ConsignaPHI>180) , ConsignaPHI=ConsignaPHI-360; end

 if ErrorPHI > 90, ErrorPHI=90; end
 if ErrorPHI < -90, ErrorPHI=-90; end
 if dErrorPHI > 6, dErrorPHI=6; end
 if dErrorPHI < -6, dErrorPHI=-6; end
       Mot=  LDout( ErrorPHI,  dErrorPHI,LDPHI);

        T100(n,5)=T100(n-1,5)+Mot(1) ;
        T100(n,6)=T100(n-1,6)-Mot(2) ;

       umbralMot=2480;
         if      (T100(n,5)>      umbralMot),      T100(n,5)=umbralMot;
T100(n,6)=-umbralMot; end
         if       (T100(n,5)<-umbralMot),       T100(n,5)=-umbralMot;
T100(n,6)=+umbralMot; end


[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;

PHI  = [PHI rad2deg(Estado(10))] ;
  dEPHI=[dEPHI dErrorPHI];
    ErrorAntPHI=ErrorPHI;
    ErrorPHI=ConsignaPHI-PHI(end);
    dErrorPHI=ErrorPHI-ErrorAntPHI;
     dErrorPHI=dErrorPHI/h;
EPHI=[EPHI ErrorPHI];

end
```

```
mediaError=mean(EPHI.^2);
mediaMotores=mean(diff(T100(:,5)).^2);
ts=tstab(EPHI,(1:Niter-1)*h,2);
PKS =findpeaks(EPHI);

cost= [ts  mediaError   mediaMotores/100 length(PKS ) ];


pause(0.01)
end
```

La función `CrearFuzzyPHIPARAM` genera dos entradas que corresponden al error y a la derivada del error del ángulo theta con 5 funciones de pertenencia, las de los extremos de tipo trapezoidal y las funciones centrales de tipo triangular; y dos salidas con 9 funciones de pertenencia.

Codigo Fuente 11 - Función 'CrearFuzzyPHIPARAM'

```
function a= CrearFuzzyPHIPARAM(X)

   P2=X(2);
   P3=X(1);
   Q2=X(4);
   Q3=X(3);
   M2=X(9);
   M3=X(8);
   M4=X(7);
   M5=X(6);
   M6=X(5);
a=newfis('ControlPHI');
RangoE=90.1;
a.input(1).name= 'Error'; a.input(1).range=[-RangoE RangoE];
a.input(1).mf(1).name='NN';            a.input(1).mf(1).type='trapmf';
a.input(1).mf(1).params=[-RangoE -RangoE -P2 -P3];
a.input(1).mf(2).name='N';              a.input(1).mf(2).type='trimf';
a.input(1).mf(2).params=[-P2 -P3 0];
a.input(1).mf(3).name='C';              a.input(1).mf(3).type='trimf';
a.input(1).mf(3).params=[-P3 0 P3];
a.input(1).mf(4).name='P';              a.input(1).mf(4).type='trimf';
a.input(1).mf(4).params=[0 P3 P2];
a.input(1).mf(5).name='PP';            a.input(1).mf(5).type='trapmf';
a.input(1).mf(5).params=[P3 P2 RangoE RangoE];

Q1=6.1;


a.input(2).name= 'dError'; a.input(2).range=[-Q1 Q1];
a.input(2).mf(1).name='NN';            a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-Q1 -Q1 -Q2 -Q3];
a.input(2).mf(2).name='N';              a.input(2).mf(2).type='trimf';
a.input(2).mf(2).params=[-Q2 -Q3 0];
```

```
a.input(2).mf(3).name='C';              a.input(2).mf(3).type='trimf';
a.input(2).mf(3).params=[-Q3 0 Q3];
a.input(2).mf(4).name='P';              a.input(2).mf(4).type='trimf';
a.input(2).mf(4).params=[0 Q3 Q2];
a.input(2).mf(5).name='PP';            a.input(2).mf(5).type='trapmf';
a.input(2).mf(5).params=[Q3 Q2 Q1 Q1];

M1=21.1;



a.output(1).name= 'MOTOR1'; a.output(1).range=[-M1 M1];
a.output(1).mf(1).name='NNNN    ';    a.output(1).mf(1).type='trapmf';
a.output(1).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(1).mf(2).name='NNN     ';     a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[-M2 -M3 -M4];
a.output(1).mf(3).name='NN      ';     a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[-M3 -M4 -M5];
a.output(1).mf(4).name='N       ';     a.output(1).mf(4).type='trimf';
a.output(1).mf(4).params=[-M4 -M5 -M6];
a.output(1).mf(5).name='C       ';     a.output(1).mf(5).type='trimf';
a.output(1).mf(5).params=[-M6  0 M6];
a.output(1).mf(6).name='P       ';     a.output(1).mf(6).type='trimf';
a.output(1).mf(6).params=[M6 M5 M4];
a.output(1).mf(7).name='PP      ';     a.output(1).mf(7).type='trimf';
a.output(1).mf(7).params=[M5 M4 M3];
a.output(1).mf(8).name='PPP     ';     a.output(1).mf(8).type='trimf';
a.output(1).mf(8).params=[M4 M3 M2];
a.output(1).mf(9).name='PPPP';        a.output(1).mf(9).type='trapmf';
a.output(1).mf(9).params=[M3 M2 M1 M1];

a.output(2).name= 'MOTOR2'; a.output(2).range=[-M1 M1];
a.output(2).mf(1).name='NNNN    ';    a.output(2).mf(1).type='trapmf';
a.output(2).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(2).mf(2).name='NNN     ';     a.output(2).mf(2).type='trimf';
a.output(2).mf(2).params=[-M2 -M3 -M4];
a.output(2).mf(3).name='NN      ';     a.output(2).mf(3).type='trimf';
a.output(2).mf(3).params=[-M3 -M4 -M5];
a.output(2).mf(4).name='N       ';     a.output(2).mf(4).type='trimf';
a.output(2).mf(4).params=[-M4 -M5 -M6];
a.output(2).mf(5).name='C       ';     a.output(2).mf(5).type='trimf';
a.output(2).mf(5).params=[-M6  0 M6];
a.output(2).mf(6).name='P       ';     a.output(2).mf(6).type='trimf';
a.output(2).mf(6).params=[M6 M5 M4];
a.output(2).mf(7).name='PP      ';     a.output(2).mf(7).type='trimf';
a.output(2).mf(7).params=[M5 M4 M3];
a.output(2).mf(8).name='PPP     ';     a.output(2).mf(8).type='trimf';
a.output(2).mf(8).params=[M4 M3 M2];
a.output(2).mf(9).name='PPPP';        a.output(2).mf(9).type='trapmf';
a.output(2).mf(9).params=[M3 M2 M1 M1];



a.rule(1).antecedent=    [1    1]    ;    a.rule(1).consequent=[9    9];
a.rule(1).weight=1; a.rule(1).connection=1;
a.rule(2).antecedent=    [1    2]    ;    a.rule(2).consequent=[9    9];
a.rule(2).weight=1; a.rule(2).connection=1;
a.rule(3).antecedent=    [1    3]    ;    a.rule(3).consequent=[9    9];
a.rule(3).weight=1; a.rule(3).connection=1;
```

```
a.rule(4).antecedent=    [1    4]    ;    a.rule(4).consequent=[8    8];
a.rule(4).weight=1; a.rule(4).connection=1;
a.rule(5).antecedent=    [1    5]    ;    a.rule(5).consequent=[7    7];
a.rule(5).weight=1; a.rule(5).connection=1;

a.rule(6).antecedent=    [2    1]    ;    a.rule(6).consequent=[7    7];
a.rule(6).weight=1; a.rule(6).connection=1;
a.rule(7).antecedent=    [2    2]    ;    a.rule(7).consequent=[6    6];
a.rule(7).weight=1; a.rule(7).connection=1;
a.rule(8).antecedent=    [2    3]    ;    a.rule(8).consequent=[6    6];
a.rule(8).weight=1; a.rule(8).connection=1;
a.rule(9).antecedent=    [2    4]    ;    a.rule(9).consequent=[5    5];
a.rule(9).weight=1; a.rule(9).connection=1;
a.rule(10).antecedent=   [2    5]    ;   a.rule(10).consequent=[5    5];
a.rule(10).weight=1; a.rule(10).connection=1;

a.rule(11).antecedent=   [3    1]    ;   a.rule(11).consequent=[6    6];
a.rule(11).weight=1; a.rule(11).connection=1;
a.rule(12).antecedent=   [3    2]    ;   a.rule(12).consequent=[5    5];
a.rule(12).weight=1; a.rule(12).connection=1;
a.rule(13).antecedent=   [3    3]    ;   a.rule(13).consequent=[5    5];
a.rule(13).weight=1; a.rule(13).connection=1;
a.rule(14).antecedent=   [3    4]    ;   a.rule(14).consequent=[5    5];
a.rule(14).weight=1; a.rule(14).connection=1;
a.rule(15).antecedent=   [3    5]    ;   a.rule(15).consequent=[4    4];
a.rule(15).weight=1; a.rule(15).connection=1;

a.rule(16).antecedent=   [4    1]    ;   a.rule(16).consequent=[5    5];
a.rule(16).weight=1; a.rule(16).connection=1;
a.rule(17).antecedent=   [4    2]    ;   a.rule(17).consequent=[5    5];
a.rule(17).weight=1; a.rule(17).connection=1;
a.rule(18).antecedent=   [4    3]    ;   a.rule(18).consequent=[4    4];
a.rule(18).weight=1; a.rule(18).connection=1;
a.rule(19).antecedent=   [4    4]    ;   a.rule(19).consequent=[4    4];
a.rule(19).weight=1; a.rule(19).connection=1;
a.rule(20).antecedent=   [4    5]    ;   a.rule(20).consequent=[3    3];
a.rule(20).weight=1; a.rule(20).connection=1;

a.rule(21).antecedent=   [5    1]    ;   a.rule(21).consequent=[3    3];
a.rule(21).weight=1; a.rule(21).connection=1;
a.rule(22).antecedent=   [5    2]    ;   a.rule(22).consequent=[2    2];
a.rule(22).weight=1; a.rule(22).connection=1;
a.rule(23).antecedent=   [5    3]    ;   a.rule(23).consequent=[1    1];
a.rule(23).weight=1; a.rule(23).connection=1;
a.rule(24).antecedent=   [5    4]    ;   a.rule(24).consequent=[1    1];
a.rule(24).weight=1; a.rule(24).connection=1;
a.rule(25).antecedent=   [5    5]    ;   a.rule(25).consequent=[1    1];
a.rule(25).weight=1; a.rule(25).connection=1;



a.defuzzMethod='centroid';

 end
```

Las reglas que se definen en el código se muestran en la siguiente tabla:

Tabla 23 - Tabla de reglas en controlador de angulo phi

	E --	E -	E C	E +	E ++
dE --	++++	++	+	0	--
dE -	++++	+	0	0	---
dE C	++++	+	0	-	----
dE +	+++	0	0	-	----
dE ++	++	0	-	--	----

De los resultados obtenidos los parámetros que se consideran óptimos son:

$$Solucion = (10.1 \quad 30.2 \quad 0.22 \quad 3.33 \quad 1.78 \quad 2.69 \quad 2.92 \quad 4.16 \quad 15.66)$$

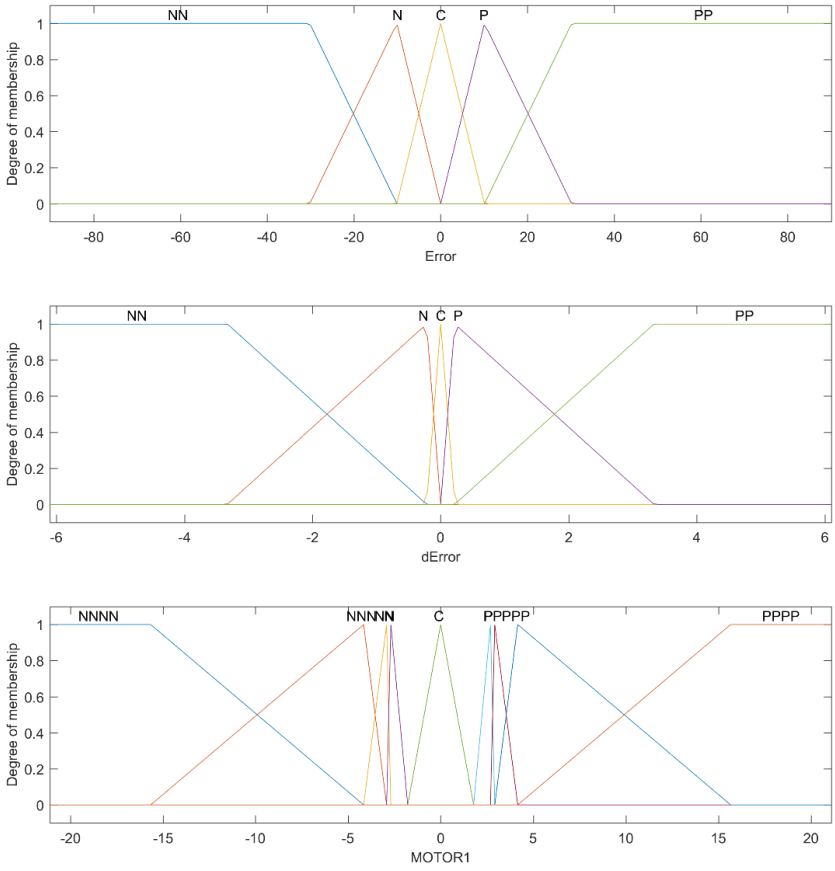


Ilustración 60 - Funciones de pertenecia optimizadas obtenidas para phi

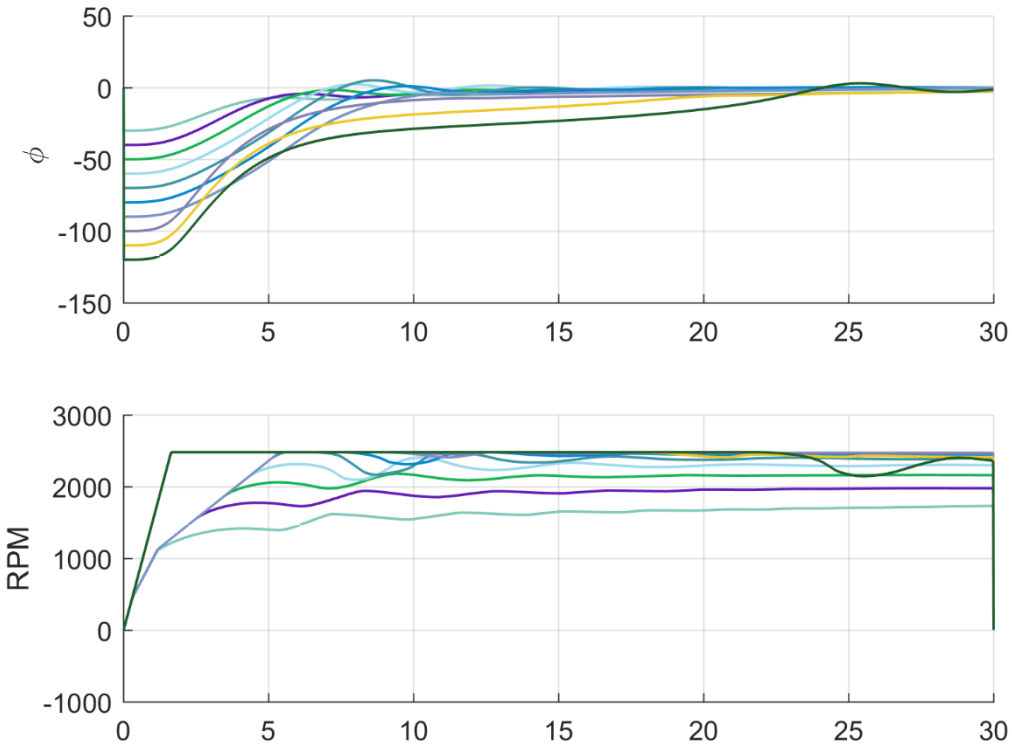


Ilustración 61 - Respuesta del sistema para angulo phi

Optimización para controlador estabilización de ángulo THETA

Este tipo de control difiere de los anteriores en que no se trata de un tipo incremental. Cuando el error es nulo las salidas también son nulas.

Codigo Fuente 12 - Funcion 'evFL_PSI'

```
function cost =evFL_PSI(X)


tf = 5;                                  % Tiempo Final de la Simulación
(s)
h = 0.01;                                   % Tiempo de Muestreo
Niter = round(tf/h);                           % Numero de muestras

T100 = zeros(Niter+1,6);
T100(1,:)=[0,0,0,0,0,0];
Estado=zeros(12,1);
ErrorPSI=0; ErrorAntPSI=0;dErrorPSI=0; EPSI=0; dEPSI=0;PSI=0;

LDPSI=CrearFuzzyPSIPARAM(X);
ConsignaPSI=C;
for n=2:Niter
    t = n*h;          %Realiza la iteración desde t=0 hasta el tiempo
final

 if ErrorPSI > 20, ErrorPSI=20; end
 if ErrorPSI < -20, ErrorPSI=-20; end
 if dErrorPSI > 5, dErrorPSI=5; end
 if dErrorPSI < -5, dErrorPSI=-5; end

       Mot=  LDout( ErrorPSI,  dErrorPSI,LDPSI);

       T100(n,2)=-4000*Mot(2) ;
       T100(n,1)=4000*Mot(1) ;

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;


PSI  = [PSI rad2deg(Estado(12))] ;
dEPSI=[dEPSI dErrorPSI];
    ErrorAntPSI=ErrorPSI;
    ErrorPSI=ConsignaPSI-PSI(end);
    dErrorPSI=ErrorPSI-ErrorAntPSI;
    dErrorPSI=dErrorPSI/h;
EPSI=[EPSI ErrorPSI];

end

mediaError=mean(EPSI.^2);
mediaMotores=mean(diff(T100(:,1)).^2);
ts=tstab(EPSI,(1:Niter-1)*h,2);
PKS =findpeaks(EPSI);

%if length(PKS)<2 , disp(X); end
cost= [ts  mediaError   mediaMotores/100 length(PKS ) ];
```

```
 color=rand(1,3);
hold on
subplot(211)
 hold on;

plot((1:Niter)*h,EPSI,'Color',color,'Linewidth',1)    ; grid on;
  hold on;

ylabel(' \psi')

subplot(212)
 hold on;

 plot((0:Niter)*h,T100(:,1),'Color',color,'Linewidth',1); grid on;
%  plot((0:Niter-1)*h,dEPSI,'Color',color,'Linewidth',1); grid on;
 ylabel('RPM')
 hold on;

end
```

La función `CrearFuzzyPSIPARAM` genera dos entradas que corresponden al error y a la derivada del error del ángulo PSI con 5 funciones de pertenencia, las de los extremos de tipo trapezoidal y las funciones centrales de tipo triangular; y dos salidas con 5 funciones de pertenencia.

Código Fuente 13 - Función 'CrearFuzzyPSIPARAM'

```
function a= CrearFuzzyPSIPARAM(X)

   P2=X(2);
   P3=X(1);
   Q2=X(4);
   Q3=X(3);
   M2=X(6);
   M3=X(5);

a=newfis('ControlPHI');
P1=20.1;
a.input(1).name= 'ErrorPSI'; a.input(1).range=[-P1 P1];
a.input(1).mf(1).name='NN';           a.input(1).mf(1).type='trapmf';
a.input(1).mf(1).params=[-P1 -P1 -P2 -P3];
a.input(1).mf(2).name='N';             a.input(1).mf(2).type='trimf';
a.input(1).mf(2).params=[-P2 -P3 0];
a.input(1).mf(3).name='C';             a.input(1).mf(3).type='trimf';
a.input(1).mf(3).params=[-P3 0 P3];
a.input(1).mf(4).name='P';             a.input(1).mf(4).type='trimf';
a.input(1).mf(4).params=[0 P3 P2];
a.input(1).mf(5).name='PP';           a.input(1).mf(5).type='trapmf';
a.input(1).mf(5).params=[P3 P2 P1 P1];
Q1=5.1;
```

```
a.input(2).name= 'dErrorPSI'; a.input(2).range=[-Q1 Q1];
a.input(2).mf(1).name='NN';           a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-Q1 -Q1 -Q2 -Q3];
a.input(2).mf(2).name='N';             a.input(2).mf(2).type='trimf';
a.input(2).mf(2).params=[-Q2 -Q3 0];
a.input(2).mf(3).name='C';             a.input(2).mf(3).type='trimf';
a.input(2).mf(3).params=[-Q3 0 Q3];
a.input(2).mf(4).name='P';             a.input(2).mf(4).type='trimf';
a.input(2).mf(4).params=[0 Q3 Q2];
a.input(2).mf(5).name='PP';           a.input(2).mf(5).type='trapmf';
a.input(2).mf(5).params=[Q3 Q2 Q1 Q1];

M1=1;



a.output(1).name= 'Motor1'; a.output(1).range=[-1 1];
a.output(1).mf(1).name='BB';         a.output(1).mf(1).type='trapmf';
a.output(1).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(1).mf(2).name='B';           a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[-M2 -M3 0];
a.output(1).mf(3).name='Cero';        a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[-M3 0 M3];
a.output(1).mf(4).name='S';           a.output(1).mf(4).type='trimf';
a.output(1).mf(4).params=[0 M3 M2];
a.output(1).mf(5).name='SS';         a.output(1).mf(5).type='trapmf';
a.output(1).mf(5).params=[M3 M2 M1 M1];

a.output(2).name= 'Motor2'; a.output(2).range=[-1 1];
a.output(2).mf(1).name='BB';         a.output(2).mf(1).type='trapmf';
a.output(2).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(2).mf(2).name='B';           a.output(2).mf(2).type='trimf';
a.output(2).mf(2).params=[-M2 -M3 0];
a.output(2).mf(3).name='Cero';        a.output(2).mf(3).type='trimf';
a.output(2).mf(3).params=[-M3 0 M3];
a.output(2).mf(4).name='S';           a.output(2).mf(4).type='trimf';
a.output(2).mf(4).params=[0 M3 M2];
a.output(2).mf(5).name='SS';         a.output(2).mf(5).type='trapmf';
a.output(2).mf(5).params=[M3 M2 M1 M1];

a.rule(1).antecedent=    [1    1]    ;    a.rule(1).consequent=[5    5];
a.rule(1).weight=1; a.rule(1).connection=1;
a.rule(2).antecedent=    [1    2]    ;    a.rule(2).consequent=[5    5];
a.rule(2).weight=1; a.rule(2).connection=1;
a.rule(3).antecedent=    [1    3]    ;    a.rule(3).consequent=[5    5];
a.rule(3).weight=1; a.rule(3).connection=1;
a.rule(4).antecedent=    [1    4]    ;    a.rule(4).consequent=[4    4];
a.rule(4).weight=1; a.rule(4).connection=1;
a.rule(5).antecedent=    [1    5]    ;    a.rule(5).consequent=[3    3];
a.rule(5).weight=1; a.rule(5).connection=1;
a.rule(6).antecedent=    [2    1]    ;    a.rule(6).consequent=[5    5];
a.rule(6).weight=1; a.rule(6).connection=1;
a.rule(7).antecedent=    [2    2]    ;    a.rule(7).consequent=[5    5];
a.rule(7).weight=1; a.rule(7).connection=1;
a.rule(8).antecedent=    [2    3]    ;    a.rule(8).consequent=[4    4];
a.rule(8).weight=1; a.rule(8).connection=1;
a.rule(9).antecedent=    [2    4]    ;    a.rule(9).consequent=[3    3];
a.rule(9).weight=1; a.rule(9).connection=1;
a.rule(10).antecedent=   [2    5]    ;   a.rule(10).consequent=[2    2];
a.rule(10).weight=1; a.rule(10).connection=1;
```

```
a.rule(11).antecedent=   [3   1]   ;   a.rule(11).consequent=[5   5];
a.rule(11).weight=1; a.rule(11).connection=1;
a.rule(12).antecedent=   [3   2]   ;   a.rule(12).consequent=[4   4];
a.rule(12).weight=1; a.rule(12).connection=1;
a.rule(13).antecedent=   [3   3]   ;   a.rule(13).consequent=[3   3];
a.rule(13).weight=1; a.rule(13).connection=1;
a.rule(14).antecedent=   [3   4]   ;   a.rule(14).consequent=[2   2];
a.rule(14).weight=1; a.rule(14).connection=1;
a.rule(15).antecedent=   [3   5]   ;   a.rule(15).consequent=[1   1];
a.rule(15).weight=1; a.rule(15).connection=1;
a.rule(16).antecedent=   [4   1]   ;   a.rule(16).consequent=[4   4];
a.rule(16).weight=1; a.rule(16).connection=1;
a.rule(17).antecedent=   [4   2]   ;   a.rule(17).consequent=[3   3];
a.rule(17).weight=1; a.rule(17).connection=1;
a.rule(18).antecedent=   [4   3]   ;   a.rule(18).consequent=[2   2];
a.rule(18).weight=1; a.rule(18).connection=1;
a.rule(19).antecedent=   [4   4]   ;   a.rule(19).consequent=[1   1];
a.rule(19).weight=1; a.rule(19).connection=1;
a.rule(20).antecedent=   [4   5]   ;   a.rule(20).consequent=[1   1];
a.rule(20).weight=1; a.rule(20).connection=1;
a.rule(21).antecedent=   [5   1]   ;   a.rule(21).consequent=[3   3];
a.rule(21).weight=1; a.rule(21).connection=1;
a.rule(22).antecedent=   [5   2]   ;   a.rule(22).consequent=[2   2];
a.rule(22).weight=1; a.rule(22).connection=1;
a.rule(23).antecedent=   [5   3]   ;   a.rule(23).consequent=[1   1];
a.rule(23).weight=1; a.rule(23).connection=1;
a.rule(24).antecedent=   [5   4]   ;   a.rule(24).consequent=[1   1];
a.rule(24).weight=1; a.rule(24).connection=1;
a.rule(25).antecedent=   [5   5]   ;   a.rule(25).consequent=[1   1];
a.rule(25).weight=1; a.rule(25).connection=1;

a.defuzzMethod='centroid';

end
```

Las reglas que se definen en el código se muestran en la siguiente tabla:

Tabla 24 - Tabla de reglas para control de angulo psi

	E --	E -	E C	E +	E ++
dE --	++	++	++	+	0
dE -	++	++	+	0	-
dE C	++	+	0	-	--
dE +	+	0	-	--	--
dE ++	0	-	--	--	--

De los resultados obtenidos los parámetros que se consideran óptimos son:

$$\text{Solución} = (9.7945 \quad 11.4078 \quad 3.6607 \quad 4.7958 \quad 0.2429 \quad 0.9213)$$

En este caso no es necesario definir tantas funciones de pertenencia como en los casos anteriores.

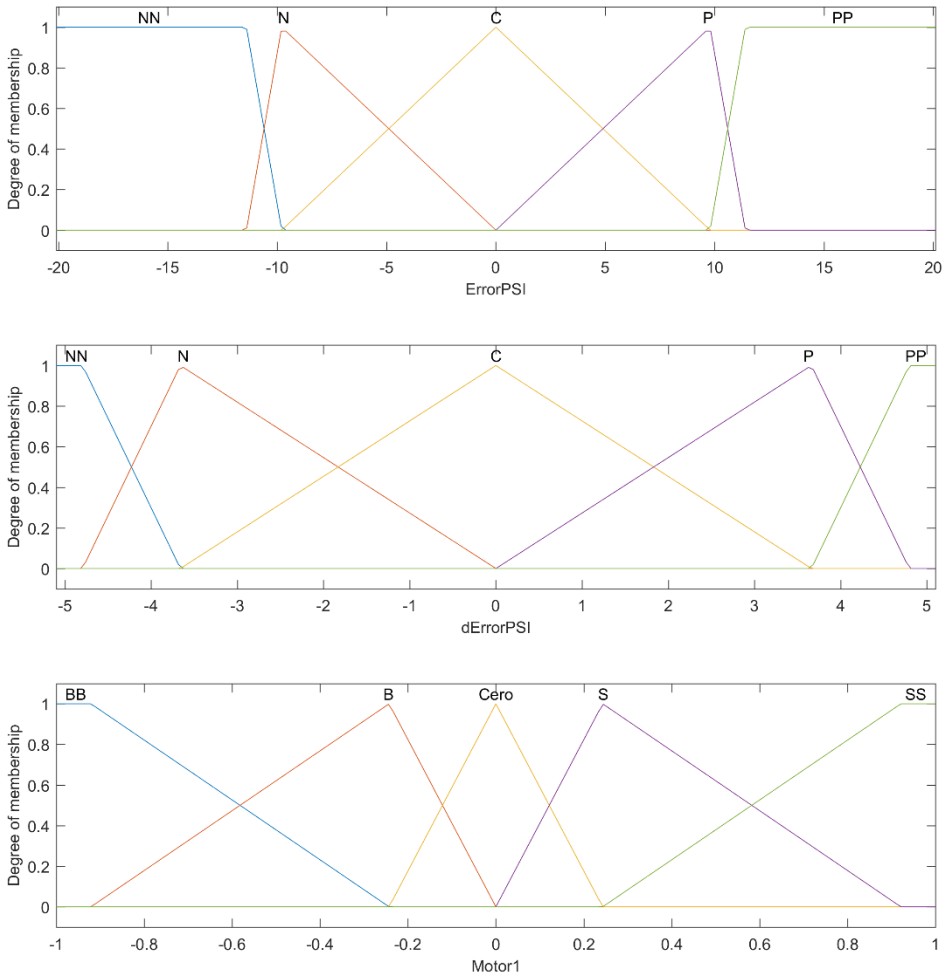


Ilustración 62 - Funciones de pertenencia para control de angulo psi

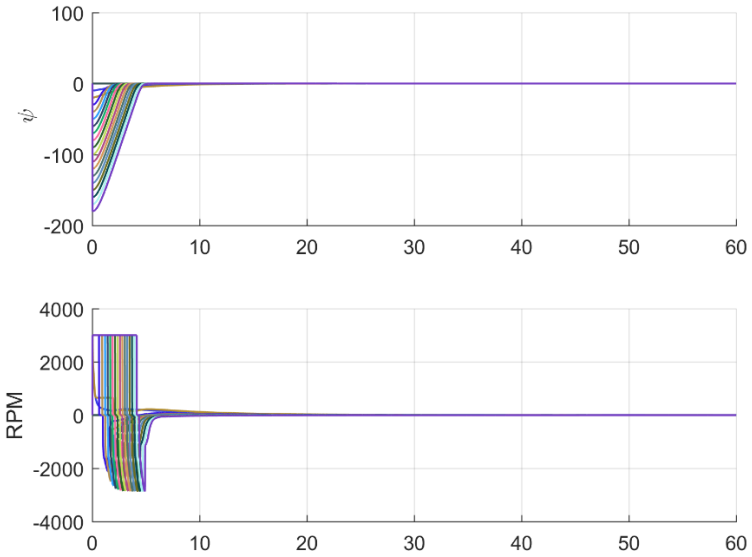


Ilustración 63 - Respuesta del sistema de control para angulo psi

Ensayo de trayectoria

Se realiza un ensayo en la que el vehiculo sigue una trayectoria definida por unos puntos de control que debe seguir a la mayor velocidad de avance posible. Para ello se define una función de entrenamiento para optimizar mediante algoritmos genéticos

Código Fuente 14 - Función 'evFLorientacion'

```
function cost=evFLorientacion(X,C)
% C=60;
tf = 60;                                % Tiempo Final de la Simulación
(s)
h = 0.01;                                  % Tiempo de Muestreo
Niter = round(tf/h);                          % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[2500,2500,0,0,0,0];
Estado=zeros(12,1);
ErrorPSI=0; ErrorAntPSI=0;dErrorPSI=0; EPSI=0; dEPSI=0;angPSI=0;
LDPSI=CrearFuzzyPSItrayPARAM(X(1),X(2),X(3),X(4),X(5),X(6),X(7),X(8)
);
lim1=X(7);
lim2=X(8);
angulosPSI=0;
Velocidad=0;
%X
 %ConsignaPSI=15;
 ConsignaPSI=C;
for n=1:Niter

    t = n*h;          %Realiza la iteración desde t=0 hasta el tiempo
final

       Mot=LDout( ErrorPSI,  dErrorPSI,LDPSI);
```

```
        T100(n,2)=4000*Mot(2) ;
        T100(n,1)=4000*Mot(1) ;


[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;
dErrorPSI=ErrorPSI-ErrorAntPSI;
dErrorPSI=dErrorPSI/h;
PSI  =rad2deg(Estado(12));    %Pasamos a grados
    ErrorAntPSI=ErrorPSI;
    ErrorPSI=ConsignaPSI-PSI;
   PSIth=5;
   if dErrorPSI> lim2, dErrorPSI= lim2*.99;end
   if dErrorPSI<-lim2, dErrorPSI=-lim2*.99;end

   if ErrorPSI> lim1, ErrorPSI= lim1*.99;end
   if ErrorPSI<-lim1, ErrorPSI= -lim1*.99;end

angulosPSI=[angulosPSI PSI];
EPSI=[EPSI ErrorPSI];
Velocidad=[Velocidad Estado(1)];
end

figure(1)
subplot(211)
   plot((0:Niter)*h,angulosPSI)   ;
     hold on;
   subplot(212)
   plot((0:Niter)*h,T100(:,2))   ;
   xlabel(num2str(X))
    hold on;
  pause(0.000001);
mediaError=mean(EPSI.^2);
ts=tstab(EPSI,(0:Niter-1)*h,0.5);
mediaMotor1=mean(diff(T100(:,2)).^2);
mediaMotor2=mean(diff(T100(:,1)).^2);
mediaVelocidad=5-mean(Velocidad);
cost= [ts  mediaError   mediaMotor1 mediaMotor2 mediaVelocidad];
 end
```

La función `CrearFuzzyTHETAPARAM` que se encarga de generar la lógica difusa de un modo parámetrico genera dos entradas que corresponden al error y a la derivada del error del ángulo theta con 5 funciones de pertenencia, las de los extremos de tipo trapezoidal y las funciones centrales de tipo triangular; y dos salidas con 5 funciones de pertenencia. A diferencia de los anteriores, en este caso se añaden unos límites superiores e inferiores para las funciones de entrada.

Codigo Fuente 15 - Funcion 'CrearFuzzyPSItrayPARAM'

```
 function a= CrearFuzzyPSItrayPARAM(P3,P2,Q3,Q2,M3,M2,lim1,lim2)


a=newfis('ControlPSI');

P1=lim1;
a.input(1).name= 'ErrorPSI'; a.input(1).range=[-P1 P1];
a.input(1).mf(1).name='NN';            a.input(1).mf(1).type='trapmf';
a.input(1).mf(1).params=[-P1 -P1 -P2 -P3];
a.input(1).mf(2).name='N';              a.input(1).mf(2).type='trimf';
a.input(1).mf(2).params=[-P2 -P3 0];
a.input(1).mf(3).name='C';              a.input(1).mf(3).type='trimf';
a.input(1).mf(3).params=[-P3 0 P3];
a.input(1).mf(4).name='P';              a.input(1).mf(4).type='trimf';
a.input(1).mf(4).params=[0 P3 P2];
a.input(1).mf(5).name='PP';            a.input(1).mf(5).type='trapmf';
a.input(1).mf(5).params=[P3 P2 P1 P1];

Q1=lim2;


a.input(2).name= 'dErrorPSI'; a.input(2).range=[-Q1 Q1];
a.input(2).mf(1).name='NN';            a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-Q1 -Q1 -Q2 -Q3];
a.input(2).mf(2).name='N';              a.input(2).mf(2).type='trimf';
a.input(2).mf(2).params=[-Q2 -Q3 0];
a.input(2).mf(3).name='C';              a.input(2).mf(3).type='trimf';
a.input(2).mf(3).params=[-Q3 0 Q3];
a.input(2).mf(4).name='P';              a.input(2).mf(4).type='trimf';
a.input(2).mf(4).params=[0 Q3 Q2];
a.input(2).mf(5).name='PP';            a.input(2).mf(5).type='trapmf';
a.input(2).mf(5).params=[Q3 Q2 Q1 Q1];

M1=1;


a.output(1).name= 'Motor1'; a.output(1).range=[-1 1];
a.output(1).mf(1).name='BB';          a.output(1).mf(1).type='trapmf';
a.output(1).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(1).mf(2).name='B';            a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[-M2 -M3 0];
a.output(1).mf(3).name='Cero';         a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[-M3 0 M3];
a.output(1).mf(4).name='S';            a.output(1).mf(4).type='trimf';
a.output(1).mf(4).params=[0 M3 M2];
a.output(1).mf(5).name='SS';          a.output(1).mf(5).type='trapmf';
a.output(1).mf(5).params=[M3 M2 M1 M1];

a.output(2).name= 'Motor2'; a.output(2).range=[-1 1];
a.output(2).mf(1).name='BB';          a.output(2).mf(1).type='trapmf';
a.output(2).mf(1).params=[-M1 -M1 -M2 -M3];
a.output(2).mf(2).name='B';            a.output(2).mf(2).type='trimf';
a.output(2).mf(2).params=[-M2 -M3 0];
a.output(2).mf(3).name='Cero';         a.output(2).mf(3).type='trimf';
a.output(2).mf(3).params=[-M3 0 M3];
```

```
a.output(2).mf(4).name='S';            a.output(2).mf(4).type='trimf';
a.output(2).mf(4).params=[0 M3 M2];
a.output(2).mf(5).name='SS';          a.output(2).mf(5).type='trapmf';
a.output(2).mf(5).params=[M3 M2 M1 M1];

PINTAR='n';
    if PINTAR=='S' ,
        pintaFuzzy(a.input(1));
        pintaFuzzy(a.input(2));
        pintaFuzzy(a.output(1));
    end


a.rule(1).antecedent=    [1    1]    ;    a.rule(1).consequent=[5    3];
a.rule(1).weight=1; a.rule(1).connection=1;
a.rule(2).antecedent=    [1    2]    ;    a.rule(2).consequent=[5    3];
a.rule(2).weight=1; a.rule(2).connection=1;
a.rule(3).antecedent=    [1    3]    ;    a.rule(3).consequent=[5    3];
a.rule(3).weight=1; a.rule(3).connection=1;
a.rule(4).antecedent=    [1    4]    ;    a.rule(4).consequent=[5    4];
a.rule(4).weight=1; a.rule(4).connection=1;
a.rule(5).antecedent=    [1    5]    ;    a.rule(5).consequent=[4    4];
a.rule(5).weight=1; a.rule(5).connection=1;

a.rule(6).antecedent=    [2    1]    ;    a.rule(6).consequent=[5    3];
a.rule(6).weight=1; a.rule(6).connection=1;
a.rule(7).antecedent=    [2    2]    ;    a.rule(7).consequent=[4    3];
a.rule(7).weight=1; a.rule(7).connection=1;
a.rule(8).antecedent=    [2    3]    ;    a.rule(8).consequent=[4    3];
a.rule(8).weight=1; a.rule(8).connection=1;
a.rule(9).antecedent=    [2    4]    ;    a.rule(9).consequent=[4    4];
a.rule(9).weight=1; a.rule(9).connection=1;
a.rule(10).antecedent=   [2    5]    ;   a.rule(10).consequent=[3    5];
a.rule(10).weight=1; a.rule(10).connection=1;

a.rule(11).antecedent=   [3    1]    ;   a.rule(11).consequent=[4    3];
a.rule(11).weight=1; a.rule(11).connection=1;
a.rule(12).antecedent=   [3    2]    ;   a.rule(12).consequent=[4    4];
a.rule(12).weight=1; a.rule(12).connection=1;
a.rule(13).antecedent=   [3    3]    ;   a.rule(13).consequent=[5    5];
a.rule(13).weight=1; a.rule(13).connection=1;
a.rule(14).antecedent=   [3    4]    ;   a.rule(14).consequent=[4    4];
a.rule(14).weight=1; a.rule(14).connection=1;
a.rule(15).antecedent=   [3    5]    ;   a.rule(15).consequent=[3    4];
a.rule(15).weight=1; a.rule(15).connection=1;

a.rule(16).antecedent=   [4    1]    ;   a.rule(16).consequent=[3    5];
a.rule(16).weight=1; a.rule(16).connection=1;
a.rule(17).antecedent=   [4    2]    ;   a.rule(17).consequent=[4    4];
a.rule(17).weight=1; a.rule(17).connection=1;
a.rule(18).antecedent=   [4    3]    ;   a.rule(18).consequent=[3    4];
a.rule(18).weight=1; a.rule(18).connection=1;
a.rule(19).antecedent=   [4    4]    ;   a.rule(19).consequent=[4    3];
a.rule(19).weight=1; a.rule(19).connection=1;
a.rule(20).antecedent=   [4    5]    ;   a.rule(20).consequent=[5    3];
a.rule(20).weight=1; a.rule(20).connection=1;

a.rule(21).antecedent=   [5    1]    ;   a.rule(21).consequent=[4    4];
a.rule(21).weight=1; a.rule(21).connection=1;
```

```
a.rule(22).antecedent=   [5   2]   ;   a.rule(22).consequent=[4   5];
a.rule(22).weight=1; a.rule(22).connection=1;
a.rule(23).antecedent=   [5   3]   ;   a.rule(23).consequent=[3   5];
a.rule(23).weight=1; a.rule(23).connection=1;
a.rule(24).antecedent=   [5   4]   ;   a.rule(24).consequent=[3   5];
a.rule(24).weight=1; a.rule(24).connection=1;
a.rule(25).antecedent=   [5   5]   ;   a.rule(25).consequent=[3   5];
a.rule(25).weight=1; a.rule(25).connection=1;

a.defuzzMethod='centroid';

 end
```

Las reglas que se definen en el código se muestran en la siguiente tabla:

Tabla 25 - Tablas de reglas para control de trayectoria para cada uno de los motores

	E --	E -	E C	E +	E ++
dE --	++	++	+	0	+
dE -	++	+	+	+	+
dE C	++	+	++	0	0
dE +	++	+	+	+	0
dE ++	+	0	0	++	0

	E --	E -	E C	E +	E ++
dE --	0	0	0	++	+
dE -	0	0	+	+	++
dE C	0	0	++	+	++
dE +	+	+	+	0	++
dE ++	+	++	+	0	++

Los valores optimos de parámetros que se encontraron tras ejecutar el algoritmo genético son:

$$Solucion = (9.0618,\ 10.3225,\ 3.2477,\ 3.7212,\ 0.1462,\ 0.9574,\ 18.8627,\ 4.3980)$$

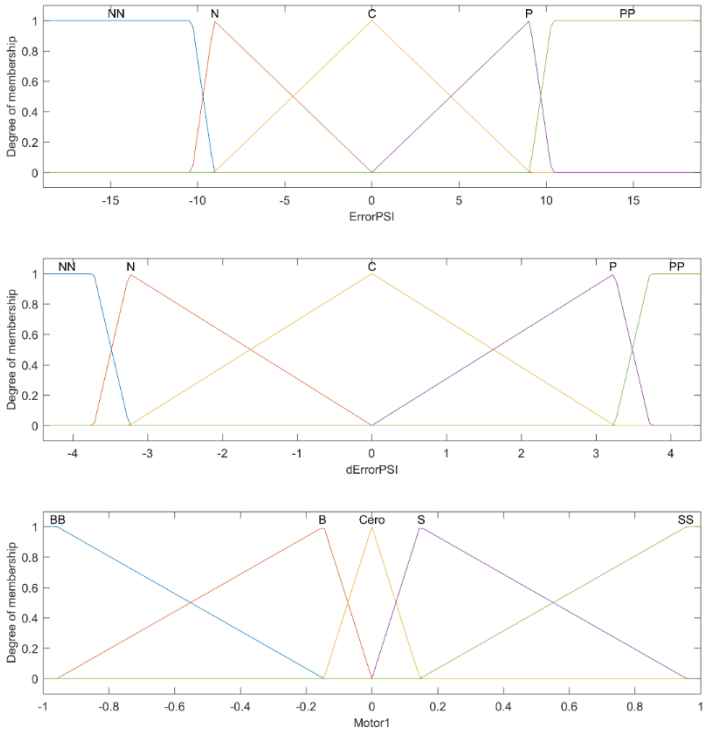


Ilustración 64 - Funciones de pertenencia para control de trayectoria

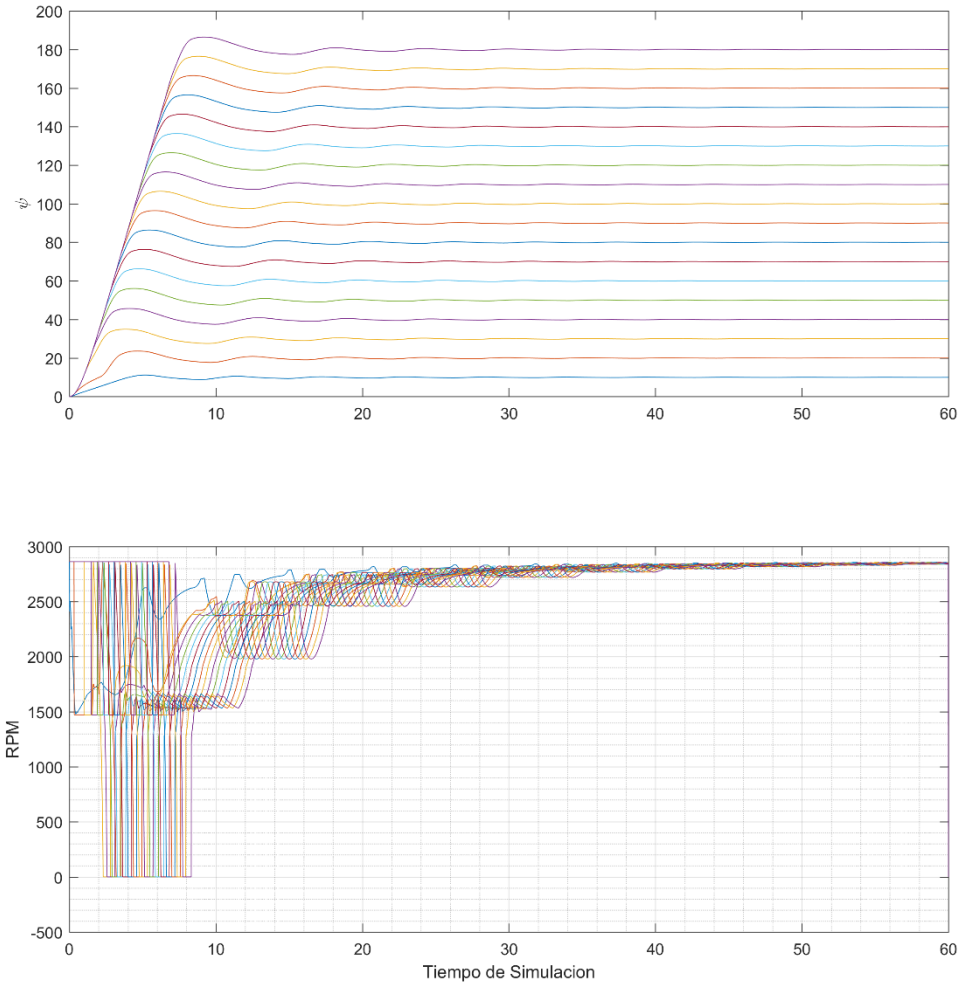


Ilustración 65 - Respuesta del sistema en control de trayectoria

Se programa un ensayo en el que el vehiculo se configura con los parámetros obtenidos para que siga una ruta marcada por unos puntos

Codigo Fuente 16- Script de trayectoria definida por puntos de control

```
clear all
close all
clc

tf = 30000;                                          % Tiempo Final de la
Simulación (s)
h = 0.1;                                        % Tiempo de Muestreo
Niter = round(tf/h);                               % Numero de muestras
T100 = zeros(Niter+1,6);
T100(1,:)=[2500,2500,0,0,0,0];

Estado=zeros(12,1);
X=0;Y=0;Z=0;PSI=0;
ErrorPHI=0; ErrorAntPHI=0;dErrorPHI=0; EPHI=0; dEPHI=0;angPHI=0;
ErrorTHETA=0;         ErrorAntTHETA=0;dErrorTHETA=0;         ETHETA=0;
dETHETA=0;angTHETA=0;
ErrorPSI=0; ErrorAntPSI=0;dErrorPSI=0; EPSI=0; dEPSI=0;angPSI=0;
ErrorZ=0; ErrorAntZ=0;dErrorZ=0; EZ=0; dEZ=0;valorZ=0;
CorreccionZ=0;flag=0;
  lim1=18.8627;
  lim2=4.3980;
  LDPSI=CrearFuzzyPSItrayPARAM( 9.0618 ,  10.3225  ,  3.2477 ,   3.7212
,   0.1462 ,    0.9574 ,   18.8627 ,    4.3980);


%PuntosX=[0   5   10   20    25    30 15 0];
%PuntosY=[0   10 5    15    10     5 30   5];

PuntosX=sort([0 fix( (rand(1,10)-0.5)*40)]);
PuntosY=sort([0 fix( (rand(1,10)-0.5)*40)]);

Estado(7)=PuntosX(1);
Estado(8)=PuntosY(1);
X=Estado(7);
Y=Estado(8);

Consigna=0;
P=1;
for n=1:Niter

    t = n*h;          %Realiza la iteración desde t=0 hasta el tiempo
final
    if    (       (abs(X(end)-PuntosX(P))<0.5)    &&       (abs(Y(end)-
PuntosY(P))<0.5)      )
        P=P+1           , disp(['Nuevo Punto --> ' num2str(PuntosX(P))
'   ,   '   num2str(PuntosY(P))])
    end
     if P==(length(PuntosX)+1) , break; end

   ConsignaPSI=calculaPSI(PuntosX(P)-X(end),PuntosY(P)- Y(end),PSI);

if dErrorPSI> lim2, dErrorPSI= lim2*.99;end
```

```
   if dErrorPSI<-lim2, dErrorPSI=-lim2*.99;end

   if ErrorPSI> lim1, ErrorPSI= lim1*.99;end
   if ErrorPSI<-lim1, ErrorPSI= -lim1*.99;end
       Mot=LDout( ErrorPSI,  dErrorPSI,LDPSI);
        T100(n,2)=4000*Mot(2) ;
        T100(n,1)=4000*Mot(1) ;

[VFuerzas,DerivEstado]=ROV2(Estado,T100,n);
Estado=Estado+h*DerivEstado;

EPSI=[EPSI ErrorPSI];
X=[X Estado(7)];   Y=[Y Estado(8)];Z=[Z Estado(9)];
Velocidad=Estado(1);
PHI  =rad2deg(Estado(10));    %Pasamos a grados
THETA=rad2deg(Estado(11));    %Pasamos a grados
PSI  =rad2deg(Estado(12));    %Pasamos a grados

    ErrorAntZ=ErrorZ;
    ErrorZ=Consigna-Z(end);
    ErrorAntPHI=ErrorPHI;
    ErrorPHI=Consigna-PHI;
    ErrorAntTHETA=ErrorTHETA;
    ErrorTHETA=Consigna-THETA;

    ErrorAntPSI=ErrorPSI;
    ErrorPSI=ConsignaPSI-PSI;

dErrorPSI=ErrorPSI-ErrorAntPSI;
dErrorPSI=dErrorPSI/h;

PSIth=5;
   if dErrorPSI> PSIth, dErrorPSI= PSIth;end
   if dErrorPSI<-PSIth, dErrorPSI=-PSIth;end

   if ErrorPSI> 20, ErrorPSI= 19.9;end
   if ErrorPSI<-20, ErrorPSI= -19.9;end

EZ=[EZ ErrorZ];dEZ=[dEZ dErrorZ];valorZ=[valorZ Z];
EPHI=[EPHI ErrorPHI];dEPHI=[dEPHI dErrorPHI];angPHI=[angPHI PHI];
ETHETA=[ETHETA                              ErrorTHETA];dETHETA=[dETHETA
dErrorTHETA];angTHETA=[angTHETA THETA];
EPSI=[EPSI ErrorPSI];dEPSI=[dEPSI dErrorPSI];angPSI=[angPSI PSI];
if mod(n,20)==0
    figure(1),  plot(X(end),Y(end),'.'),title([    num2str(fix(PSI))
'grados  ' num2str(fix(ConsignaPSI)) ' grados  ' num2str(t) ' segundos
' num2str(Velocidad) 'm/s ']),hold on,plot(PuntosX,PuntosY,'o'),axis
manual,axis([min(PuntosX)*1.1    max(PuntosX)*1.1    min(PuntosY)*1.1
max(PuntosY)*1.1]);
    xlabel(num2str([X(end) PuntosX(P) T100(n,1) T100(n,2)]));
    ylabel(num2str([Y(end) PuntosY(P)]));

    figure(2) ,
    subplot(2,1,1),hold off,plot(1,1), pintaFuzzy(LDPSI.input(1));
    plot(ErrorPSI*ones(1,2),[0  1],'r');  axis  manual,axis([-45  45  0
1]);

    subplot(2,1,2),hold off, plot(1,1), pintaFuzzy(LDPSI.input(2));
```

```
    plot(dErrorPSI*ones(1,2),[0 1],'r');  axis manual,axis([-45 45 0
1]);

    pause(0.00001);end

end


figure,                                                plot(EPHI),hold
on,plot(ETHETA,'g'),plot(EPSI,'r'),title('Error                    de
Angulos'),legend('\phi','\theta','\psi')
figure,                                                  plot(X),hold
on,plot(Y,'g'),plot(Z,'r'),title('XYZ'),legend('X','Y','Z')
figure, plot(X,Y), hold on, plot(PuntosX,PuntosY,'*')
figure,     plot(X,Y),     hold     on,     plot(PuntosX,PuntosY,'*'),
plot(X,T100(1:length(X),1),'g'),              hold                on,
plot(X,T100(1:length(X),2),'r')
```

En la imagen se muestra el resultado de la ejecución de la simulación del seguimiento de un trayecto definido por puntos.

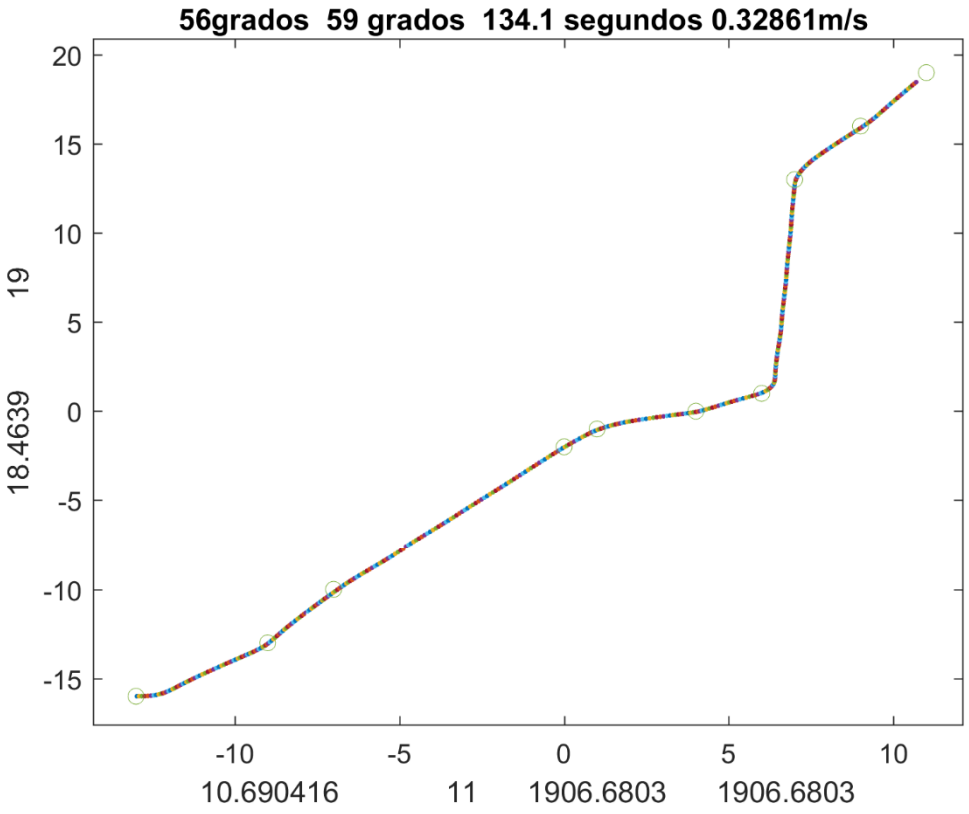


Ilustración 66 – Respuesta de trayectoria seguida por el vehículo

Comparativa entre Control Fuzzy y PID

Se realiza una comparativa del sistema de control de posición angular basado en lógica difusa (fuzzy) con un sistema de control basado en PID

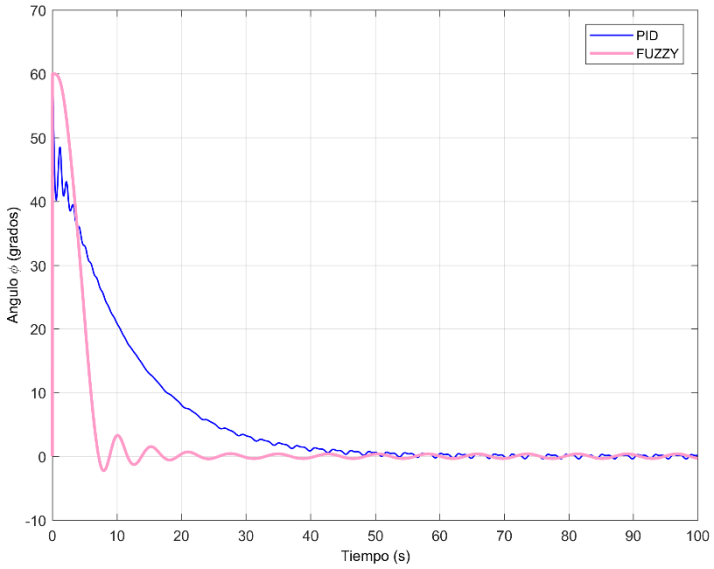


Ilustración 67 - Comparativa de Posición Angular PHI

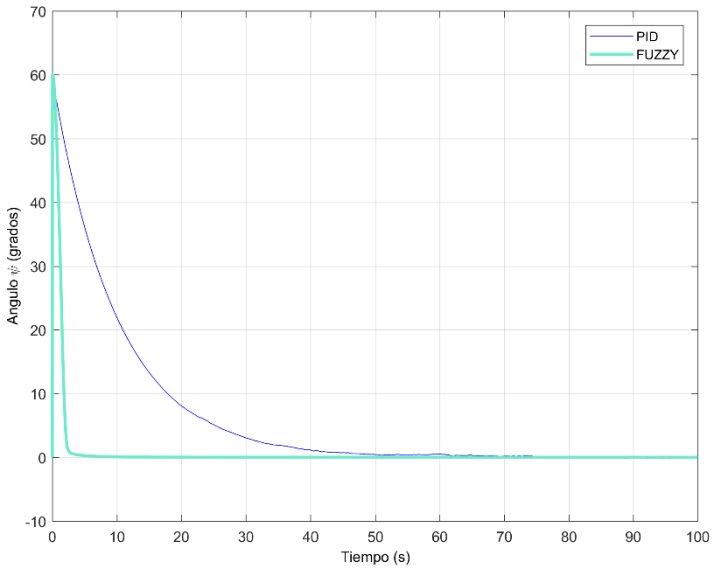


Ilustración 68 - Comparativa de Posición Angular PSI

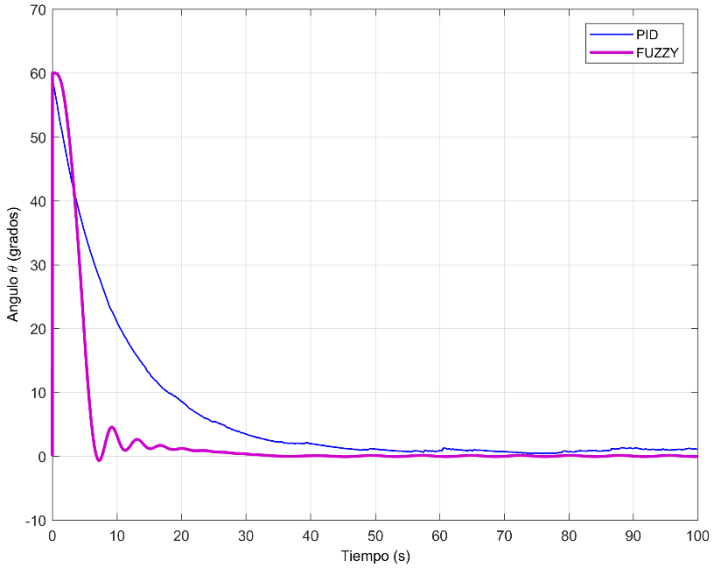


Ilustración 69 - - Comparativa de Posición Angular THETA

Ensayo de medida y transmisión de ángulos de navegación

Implementa un control de temporización, *Timer1*, que es el encargado de realizar consultas hacia la placa de Arduino.

Al pulsar en el botón CONECTAR se ejecuta el evento *Click* que activa el *Timer1* y abre la comunicación por puerto serie a través del objeto *pSerie*. El texto de botón cambia a "DESCONECTAR", para que la próxima vez que se pulse desactive el Timer1 y cierre la comunicación serie.

Ilustración 70 - Aplicación de medida y transimision de ángulos

```
Private Sub bt_Conectar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bt_Conectar.Click
        If bt_Conectar.Text = "CONECTAR" Then
            bt_Conectar.Text = "DESCONECTAR"
            Timer1.Enabled = True
            pSerie.Open()
        ElseIf bt_Conectar.Text = "DESCONECTAR" Then
            bt_Conectar.Text = "CONECTAR"
            Timer1.Enabled = False
            pSerie.Close()
        End If
    End Sub
```

Cada vez que se desborda el Timer1 envía A, B y C hacia la placa Arduino, la cual está programada para que cuando reciba A devuelva el ángulo del eje X, para B devuelva el ángulo del eje Y, y para C devuelva el ángulo del eje Z.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick

        pSerie.Write("A")
        pSerie.Write("B")
        pSerie.Write("C")

    End Sub
```

El puerto serie está configurado con una velocidad 2500000bps en el puerto COM4.

La recepción serie se realiza a través de un evento llamado *pSerie.DataReceived.* Mientras existan datos para leer en el buffer del puerto serie, se leerán línea a línea los datos clasificándolos según la letra que preceda al ángulo correspondiente. Dentro del caso "C" se realizan unas operaciones (rotación y traslación de un LineShape) para poder representar la orientación a través de una línea a modo brújula.

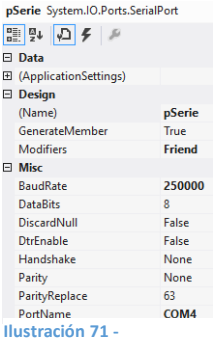


Ilustración 71 - Configuración puerto serie

En todos los casos representa el valor recibido en su correspondiente textbox.

```
Private Sub pSerie_DataReceived(sender As Object, e As
IO.Ports.SerialDataReceivedEventArgs) Handles pSerie.DataReceived
        While (pSerie.BytesToRead > 0)
            Dim cadena = pSerie.ReadLine()

            If cadena.Contains("A") Then
                tB_ax.Text = cadena.Substring(2)
            ElseIf cadena.Contains("B") Then
                tB_ay.Text = cadena.Substring(2)
            ElseIf cadena.Contains("C") Then
                tB_az.Text = cadena.Substring(2)
                Dim ang = Convert.ToDouble(tB_az.Text) / 100 * PI / 180
                x = 400 + (400 - 400) * Cos(ang) - (30 - 100) * Sin(ang)
                y = 100 + (400 - 400) * Sin(ang) + (30 - 100) * Cos(ang)
                LineShape1.X2 = Convert.ToInt32(x)
                LineShape1.Y2 = Convert.ToInt32(y)
            End If

        End While

    End Sub
```

En el código de Arduino se incluye la librería de LSM303, incluye un Low Pass Filter (con parámetro alpha) para el acelerómetro y el magnetómetro. En el código se han elaborado dos funciones que se repiten indefinidamente.

La función calculaAngulos(), realiza los cálculos, transformaciones y filtrados necesarios para obtener los ángulos de navegación (Yaw, Pitch, Roll).

La función ComSerie(), está diseñada para comprobar si hay datos en el buffer de entrada del puerto serie y contestar en base a las peticiones que se reciban.

Codigo Fuente 17 - Angulos en Ardunio

```
#include <LSM303.h>
LSM303 compass;
unsigned int  rx;
const float alpha = 0.15;
float fXa = 0;
float fYa = 0;
float fZa = 0;
float fXm = 0;
float fYm = 0;
float fZm = 0;

void setup()
{
Serial.begin(250000);
Wire.begin();
compass.init();
compass.enableDefault();
}
float pitch, pitch_print, roll, roll_print, Heading, Xa_off, Ya_off,
Za_off, Xa_cal, Ya_cal, Za_cal, Xm_off, Ym_off, Zm_off, Xm_cal,
Ym_cal, Zm_cal, fXm_comp, fYm_comp;
```

```
void loop()
{
calculaAngulos();
ComSerie();
}
void ComSerie()
{
  int recibido=0;
if(Serial.available())
    {
      recibido=Serial.read();
      delay(20);
        if(recibido==65)
        {
          Serial.println("A=" + String(pitch_print,2));
        }
        else if(recibido==66)
        {
          Serial.println("B=" + String(roll_print,2));

        }
        else if(recibido==67)
        {
          Serial.println("C=" + String(Heading,2));

        }

       recibido=0;
    }

}

void calculaAngulos()
{
  compass.read();

// Calibracion del acelerómetro
Xa_off = compass.a.x/16.0 - 7.837247;
Ya_off = compass.a.y/16.0 + 6.828846;
Za_off = compass.a.z/16.0 + 79.337329;

Xa_cal =  0.956712*Xa_off -0.026425*Ya_off +0.029607*Za_off;
Ya_cal =  -0.026425*Xa_off + 0.990613*Ya_off -0.003651*Za_off;
Za_cal = 0.029607*Xa_off -0.003651*Ya_off + 0.958057*Za_off;

// Calibracion del magnetómetro
Xm_off = compass.m.x*(100000.0/1100.0) - 39279.330639;
Ym_off = compass.m.y*(100000.0/1100.0) - 14391.779243;
Zm_off = compass.m.z*(100000.0/980.0 ) + 59077.331735

Xm_cal =  0.179781*Xm_off + -0.015747*Ym_off + 0.004337*Zm_off;
Ym_cal =  -0.015747*Xm_off + 0.191855*Ym_off + 0.001401*Zm_off;
Zm_cal =  0.004337*Xm_off + 0.001401*Ym_off + 0.180220*Zm_off;

// Low-Pass filter del acelerómetro
fXa = Xa_cal * alpha + (fXa * (1.0 - alpha));
fYa = Ya_cal * alpha + (fYa * (1.0 - alpha));
fZa = Za_cal * alpha + (fZa * (1.0 - alpha));

// Low-Pass filter del magnetómetro
```

```
fXm = Xm_cal * alpha + (fXm * (1.0 - alpha));
fYm = Ym_cal * alpha + (fYm * (1.0 - alpha));
fZm = Zm_cal * alpha + (fZm * (1.0 - alpha));

// Pitch y roll
roll  = atan2(fYa, sqrt(fXa*fXa + fZa*fZa));
pitch = atan2(fXa, sqrt(fYa*fYa + fZa*fZa));
roll_print = roll*180.0/M_PI;
pitch_print = pitch*180.0/M_PI;

// Mediciones del sensor magnético compensado por inclinación
fXm_comp = fXm*cos(pitch)+fZm*sin(pitch);
fYm_comp          =          fXm*sin(roll)*sin(pitch)+fYm*cos(roll)-
fZm*sin(roll)*cos(pitch);

// Arcotangente de y/x
Heading = (atan2(fYm_comp,fXm_comp)*180.0)/M_PI;
if (Heading < 0)
Heading += 360;
}
```

Ensayo de Control a través de Joystick

Esta aplicación se desarrolló como herramienta de apoyo al desarrollo de la Interfaz Gráfica para ROV llevado a cabo por el Departamento de Ingeniería Mecánica de la Universidad de Cartagena para monitorizar el comportamiento del joystick que estaban utilizando y poder incluir un control a través de este dispositivo en su interfaz.

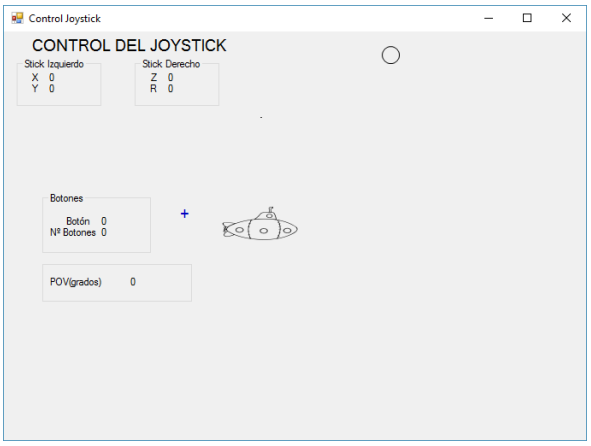


Ilustración 72- Aplicacion Control por Joystick

La función *joyGetPosEx* consulta un joystick para su posición y estado del botón. Pertenece a la librería *Winmm.lib* de la dll *Winmm.dll* de Microsoft. Los parámetros de la función son *uJoyID y pji*.

uJoyID

Identificador del joystick que se va a consultar. Los valores permitidos van desde JOYSTICKID1 hasta JOYSTICKID15

pji

Se trata de un puntero a una estructura JOYINFOEX que contiene información de posición y estado de los controles y botones del joystick. Se deben establecer los miembros *dwSize* y *dwFlags* o la función *joyGetPosEx* devolverá un fallo. La información devuelta por *joyGetPosEx* depende de los indicadores con los que se configure *dwFlags*.

La función devuelve JOYERR_NOERROR si todo es correcto, en caso contrario devolverá uno de los siguientes códigos de error

Tabla 26 - Codigos de error Joystick

MMSYSERR_NODRIVER	El driver del joystick no está presente
MMSYSERR_INVALPARAM	Se pasó un parámetro no valido
MMSYSERR_BADDEVICEID	El identificador de joystick especificado no es valido
JOYERR_UNPLUGGED	El joystick especificado no está conectado al sistema

La estructura *JOYINFOEX* contiene información ampliada sobre la posición y los controles del joystick. Los miembros son dwSize que indica el tamaño en bytes de esta estructura, y dwFlags que indica la información válida devuelta en esta estructura, se definen los siguientes flags:

Tabla 27 - Configuración de estructura JOYINFOEX

JOY_RETURNALL	Configuración de todos los bits JOY_RETURN excepto JOY_RETURNRAWDATA
JOY_RETURNBUTTONS	El miembro dwButtons contiene información válida sobre el estado de cada botón del joystick.
JOY_RETURNCENTERED	Centra la posición neutral del joystick en el valor medio de cada eje de movimiento.
JOY_RETURNPOV	El miembro dwPOV contiene información válida sobre el control cruceta, expresada en unidades discretas.
JOY_RETURNPOVCTS	El miembro dwPOV contiene información válida sobre el control de punto de vista expresado en unidades continuas en grados
JOY_RETURNR	El miembro dwRpos contiene datos válidos del pedal del timón. Esta información representa otro (cuarto) eje.
JOY_RETURNRAWDATA	Los datos almacenados en esta estructura son lecturas no calibradas del joystick
JOY_RETURNU	El miembro dwUpos contiene datos válidos para un quinto eje del joystick, si dicho eje está disponible, o devuelve cero en caso contrario
JOY_RETURNV	El miembro dwVpos contiene datos válidos para un sexto eje del joystick, si dicho eje está disponible, o devuelve cero en caso contrario
JOY_RETURNX	El miembro dwXpos contiene datos válidos para la coordenada x del joystick
JOY_RETURNY	El miembro dwYpos contiene datos válidos para la coordenada y del joystick
JOY_RETURNZ	El miembro dwZpos contiene datos válidos para la coordenada z del joystick

dwXpos

Coordenada X actual

dwYpos

Coordenada Y actual

dwZpos

Coordenada Z actual

dwRpos

Posición actual del timón o del cuarto eje del joystick

dwUpos

Posición actual del quinto eje del joystick

dwVpos

Posición actual del sexto eje del joystick

dwButtons

Estado actual de los 32 botones del joystick. El valor de este miembro se puede establecer en cualquier combinación de símbolos JOY_BUTTON n, donde n es un valor en el intervalo de 1 a 32 correspondiente al botón que se pulsa.

dwButtonNumber

Numero de botones que están pulsados actualmente

dwPOV

Posición actual del control de cruceta. Los valores para este miembro se encuentran en el rango de 0 a 35.900. Estos valores representan el ángulo, en grados, multiplicado por 100.

dwReserved1

Reservado, no utilizado

dwReserved2

Reservado, no utilizado

Se declara la siguiente función, que es la encargada de realizar la consulta para obtener toda la información del estado de los controles y los botones del joystick.

Codigo Fuente 18- Función joyGetPosEx en VB

```
Declare Function joyGetPosEx Lib "winmm.dll" (ByVal uJoyID As Integer, ByRef pji
As JOYINFOEX) As Integer


    <StructLayout(LayoutKind.Sequential)> _
    Public Structure JOYINFOEX
        Public dwSize As Integer
        Public dwFlags As Integer
        Public dwXpos As Integer
        Public dwYpos As Integer
        Public dwZpos As Integer
        Public dwRpos As Integer
        Public dwUpos As Integer
        Public dwVpos As Integer
        Public dwButtons As Integer
        Public dwButtonNumber As Integer
        Public dwPOV As Integer
        Public dwReserved1 As Integer
        Public dwReserved2 As Integer
    End Structure
```

Se declara bajo `Dim myjoyEX As JOYINFOEX`

Al inicio de la aplicación se inicializa dwSize a 64 y dwFlags a 0xFF para que nos muestre toda la información disponible, se inicializa un timer con una frecuencia de 5Hz

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        myjoyEX.dwSize = 64
        myjoyEX.dwFlags = &HFF ' Toda la Informacion
        Timer1.Interval = 50   ' Refresco a 5Hz
        Timer1.Start()         ' Inicia el timer
        iX = 0
        iY = 0
    End Sub
```

Cada vez que se produce el evento del Timer1 se hace una llamada a la función joyGetPosEx y los datos de la estructura se cargan en los Label que representan a cada uno de los botones y controles del joystick. Se generan dos cruces que cambian su propiedad Location de forma que represente el movimiento de los sticks.

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick

        ' Obtener la informacion de joystick
        Call joyGetPosEx(0, myjoyEX)

        With myjoyEX
            Label10.Text = .dwXpos.ToString
            Label11.Text = .dwYpos.ToString
            Label12.Text = .dwZpos.ToString
            Label13.Text = .dwRpos.ToString
            Label14.Text = .dwUpos.ToString
            Label15.Text = .dwVpos.ToString
            Label16.Text = .dwButtons.ToString("X")  'Representa en Hexadecimal
la variable
            Label17.Text = .dwButtonNumber.ToString  'Numero de botones pulsados
al mismo tiempo

            If (.dwPOV = 65535) Then
                Label18.Text = "Centro"
            Else
                Label18.Text = (.dwPOV / 100).ToString     ' Angulo de la cruceta
en grados

            End If

            Cruz1.Location = New Point(300 + 0.003 * (.dwXpos - 32676), 300 +
0.003 * (.dwYpos - 32676))
            Cruz2.Location = New Point(300 + 0.003 * (.dwZpos - 32676), 300 +
0.003 * (.dwRpos - 32676))

            If (.dwButtons = 1) Then
                B1.FillStyle = PowerPacks.FillStyle.Solid
                B1.FillColor = Color.Red
```

```
            Else
                B1.FillStyle = PowerPacks.FillStyle.Transparent

            End If

        End With
    End Sub
```

En el programa, se incluye un Timer2 que es el encargado de actualizar la animación de un submarino controlado a través de los controles del joystick.

```
Private Sub Timer2_Tick(sender As Object, e As EventArgs) Handles Timer2.Tick
        Dim vX As Integer
        Dim vY As Integer



        Call joyGetPosEx(0, myjoyEX)

        With myjoyEX

            vX = .dwXpos - 32677
            vY = .dwYpos - 32677
            iX = iX + vX * 0.0001
            iY = iY + vY * 0.0001
        End With

        ROV.Location = New Point(ROV.Location.X + vX * 0.00005 + iX,
ROV.Location.Y + vY * 0.00005 + iY)



    End Sub
```

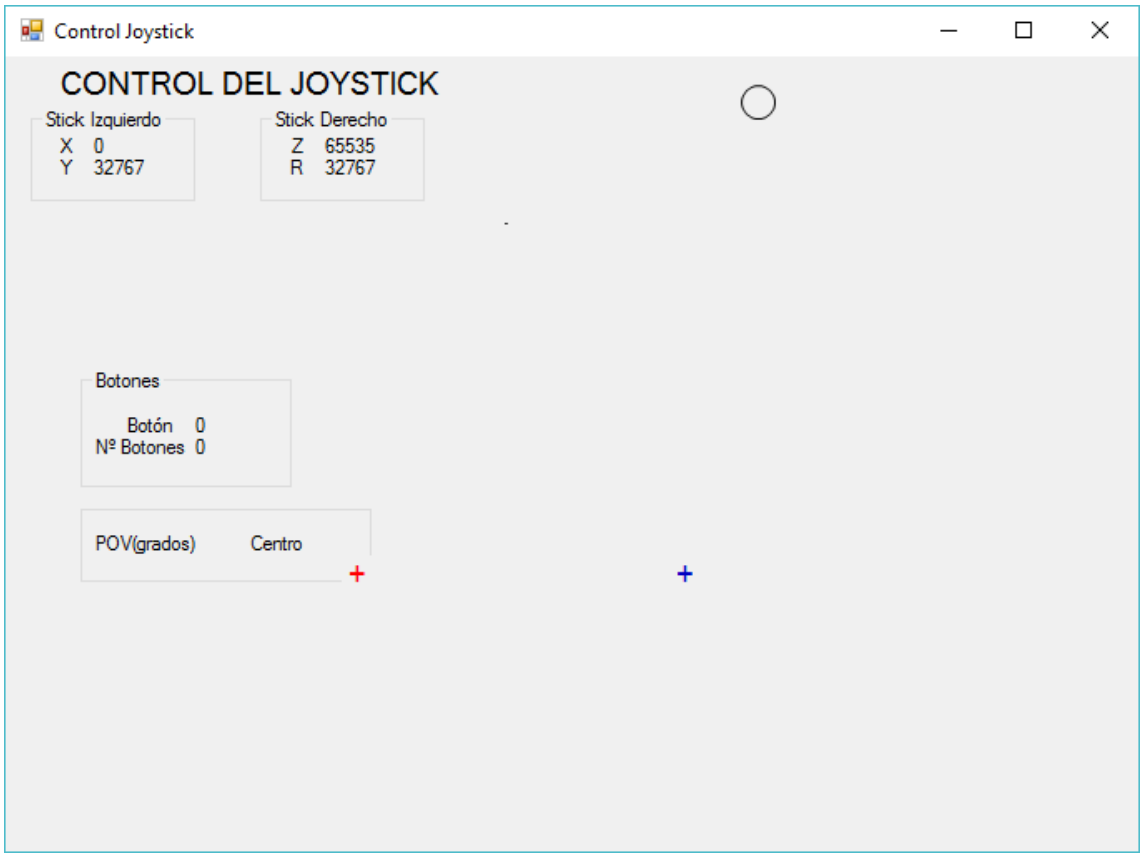


Ilustración 73 - Captura de aplicación de Control por Joystick

Prueba de control de motores mediante joystick o manual y monitorización de puerto serie

Esta aplicación realiza la conexión por puerto serie a Arduino y fue desarrollada en Visual Basic para depurar y verificar el funcionamiento de la comunicación y el control sobre los motores.

Permite manejar cada uno de los motores mediante barras de desplazamiento (Trackbars) o por joystick.

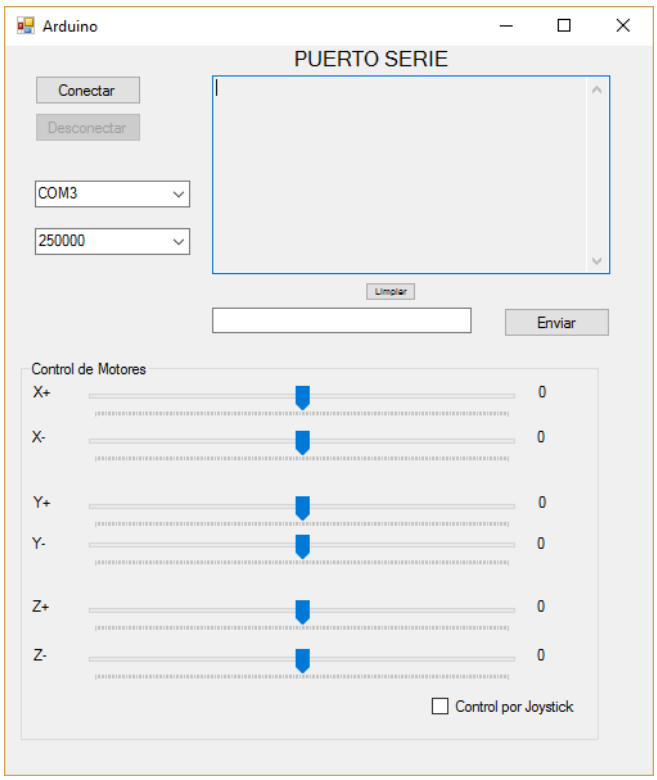


Ilustración 74 - Aplicación control de motores

Las barras de desplazamiento tienen límites de -100 y 100, por defecto se encuentran en el valor 0. Si las barras se desplazan hacia valores negativos, se traducirá en un sentido de giro del motor correspondiente, en cambio para valores positivos, el motor girara en sentido contrario al anterior.

Para iniciar el control y la comunicación, es necesario pulsar el botón de Conectar, una vez se conecta empezara a aparecer en el cuadro de texto la información que se está transmitiendo por el puerto serie. La aplicación también está diseñada para enviar "manualmente" una cadena de texto por el puerto serie, para ello se debe ingresar el texto a enviar el cuadro de texto pequeño y pulsar el botón de enviar.

Al inicio de la aplicación, se inicializan una serie de variables y controles.

```
Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        StrBufferIn = ""            'Buffer de entrada
        StrBufferOut = ""           'Buffer de salida
        cB_Port.SelectedIndex = 2   'COM3, por defecto
        cB_bps.SelectedIndex = 7    '250000 baudios, por defecto

        'Se asigna a cada Label el valor que marca el trackbar que le corresponde
        X1V.Text = X1.Value
        X2V.Text = X2.Value
```

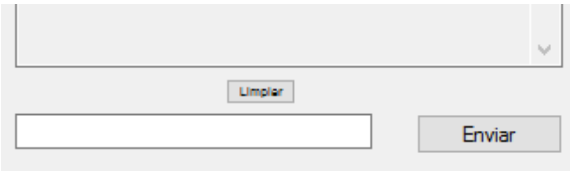
```
        Y1V.Text = Y1.Value
        Y2V.Text = Y2.Value
        Z1V.Text = Z1.Value
        Z2V.Text = Z2.Value
        CheckForIllegalCrossThreadCalls = False
    End Sub
```

Al pulsar en el botón CONECTAR se ejecuta el evento *Click* que activa el *timer T_Serie* y abre la comunicación por puerto serie a través del objeto *pSerie*.

```
Private Sub bT_CSERIE_Click(sender As Object, e As EventArgs) Handles
bT_CSERIE.Click
        pSerie.BaudRate = cB_bps.Text   'Asigna la velocidad al puerto serie
        pSerie.PortName = cB_Port.Text  'Asigna el puerto al puerto serie
        pSerie.Open()                   'Abre la conexión Serie
        bT_CSERIE.Enabled = False       'Desactiva el botón de conectar
        bT_DSERIE.Enabled = True        'Activa el botón de conectar
        T_Serie.Enabled = True          'Activa el timer T_Serie
    End Sub
```

El botón conectar queda deshabilitado y el botón desconectar se habilita. Las acciones que realiza el botón desconectar es la de cerrar la comunicación del puerto serie, habilitar el botón conectar de nuevo y deshabilitar el de desconectar.

```
Private Sub bT_DSERIE_Click(sender As Object, e As EventArgs) Handles
bT_DSERIE.Click
        pSerie.Close()              'Cierra la conexión Serie
        bT_CSERIE.Enabled = True    'Activa el botón conectar
        bT_DSERIE.Enabled = False   'Desactiva el botón desconectar
    End Sub
```



El pequeño cuadro de texto que antes se ha mencionado, se puede utilizar junto con el botón Enviar para realizar el envío de datos a través del puerto serie. Al pulsar el botón recoge el texto del textbox, lo carga en el buffer de salida del puerto serie y vacía el cuadro de texto.

```
'Evento que realiza el envió mediante Serial Port de los datos del textbox tB_enviar
    Private Sub bT_enviar_Click(sender As Object, e As EventArgs) Handles bT_enviar.Click

       StrBufferOut = tB_enviar.Text
        tB_enviar.Text = ""
        pSerie.Write(StrBufferOut)

    End Sub
```

El cuadro de texto maneja un evento tipo KeyDown que controla cuando se pulsa la tecla *Enter.* Cuando se pulsa, se automatiza un Click en el botón enviar.

```
Private Sub tB_enviar_KeyDown(sender As Object, e As KeyEventArgs) Handles tB_enviar.KeyDown
        If e.KeyCode = Keys.Enter Then

            bT_enviar.PerformClick()

        End If
    End Sub
```

Existe un checkbox que al ser modificado activa el control mediante joystick de los motores. Esta acción se realiza mediante la activación – desactivación de un timer (Timer1).

```
Private Sub cB_JoyControl_CheckedChanged(sender As Object, e As EventArgs) Handles cB_JoyControl.CheckedChanged
        If cB_JoyControl.Checked = True Then
            Timer1.Enabled = True
        Else
            Timer1.Enabled = False
        End If
    End Sub
```

El Timer1, captura los datos de los ejes del joystick y los adapta para modificar el valor de las barras de desplazamiento que controlan los motores.

```
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        ' Obtener la informacion de joystick
        Call joyGetPosEx(0, myjoyEX)

        With myjoyEX
            X1.Value = (.dwXpos.ToString - 32767) * 100 / 32767
            X2.Value = X1.Value
            Y1.Value = (.dwYpos.ToString - 32767) * 100 / 32767
            Y2.Value = Y1.Value
            Z1.Value = (.dwZpos.ToString - 32767) * 100 / 32767
            Z2.Value = Z1.Value

        End With

    End Sub
```

Cuando algún valor de las barras de desplazamiento es modificado, automáticamente es detectado por cada uno de los eventos que detectan este cambio y realiza un envio por el puerto serie hacia la placa Arduino

```
Private Sub X1_ValueChanged(sender As Object, e As EventArgs) Handles
X1.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
        enviado(0) = X1.Value + 100
        pSerie.Write("D")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub


    Private Sub X2_ValueChanged(sender As Object, e As EventArgs) Handles
X2.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
        enviado(0) = X2.Value + 100
        pSerie.Write("E")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub

    Private Sub Y1_ValueChanged(sender As Object, e As EventArgs) Handles
Y1.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
```

```
        enviado(0) = Y1.Value + 100
        pSerie.Write("F")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub

    Private Sub Y2_ValueChanged(sender As Object, e As EventArgs) Handles
Y2.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
        enviado(0) = Y2.Value + 100
        pSerie.Write("G")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub

    Private Sub Z1_ValueChanged(sender As Object, e As EventArgs) Handles
Z1.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
        enviado(0) = Z1.Value + 100
        pSerie.Write("H")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub

    Private Sub Z2_ValueChanged(sender As Object, e As EventArgs) Handles
Z2.ValueChanged
        Dim enviado As Byte() = New Byte(0) {}
        enviado(0) = Z2.Value + 100
        pSerie.Write("I")
        pSerie.Write(enviado, 0, enviado.Length)
    End Sub
```

En Arduino, se extiende la función Comunicacion() generada en pruebas anteriores según el siguiente código

```
        else if(recibido==68)
        {

          rx=Serial.read();
           rx=rx-100;
          rx=4*rx+1500 ;
          Serial.println(String(rx));
          servoX1.attach(3); servoX1.writeMicroseconds(rx);
```

```
}
 else if(recibido==69)
{
  rx=Serial.read();
  rx=rx-100;
  rx=4*rx+1500 ;
  Serial.println(String(rx));
  servoX2.attach(5); servoX1.writeMicroseconds(rx);

}
 else if(recibido==70)
{
  rx=Serial.read();
   rx=rx-100;
  rx=4*rx+1500 ;
  Serial.println(String(rx));
  servoY1.attach(6); servoY1.writeMicroseconds(rx);

}
 else if(recibido==71)
{
  rx=Serial.read();
  rx=rx-100;
  rx=4*rx+1500 ;
  Serial.println(String(rx));
  servoY2.attach(9); servoX1.writeMicroseconds(rx);
 }
 else if(recibido==72)
{

  rx=Serial.read();
   rx=rx-100;
  rx=4*rx+1500 ;
  Serial.println(String(rx));
  servoZ1.attach(10); servoZ1.writeMicroseconds(rx);
}
  else if(recibido==73)
{

  rx=Serial.read();
```

```
 rx=rx-100;
rx=4*rx+1500 ;
Serial.println(String(rx));
servoZ1.attach(11); servoZ2.writeMicroseconds(rx);
```

Se incluye la librería Servo y se declaran los objetos tipo Servo, uno por cada motor

```
#include <Servo.h>
```

```
Servo servoX1;
Servo servoX2;
Servo servoY1;
Servo servoY2;
Servo servoZ1;
Servo servoZ2;
```

Dentro de la función Setup se configuran en los pines de salida correspondientes y se inicializan a 1500 (valor de reposo del motor)

```
servoX1.attach(3); servoX1.writeMicroseconds(1500);
servoX2.attach(5); servoX2.writeMicroseconds(1500);
servoY1.attach(6); servoY1.writeMicroseconds(1500);
servoY2.attach(9); servoY2.writeMicroseconds(1500);
servoZ1.attach(10); servoZ1.writeMicroseconds(1500);
servoZ2.attach(11); servoZ2.writeMicroseconds(1500);
```

Diseño de una aplicación Simulador del Sistema de comunicación del UUV

Esta es otra aplicación diseñada también como herramienta de apoyo al desarrollo de la Interfaz Gráfica para ROV llevado a cabo por el Departamento de Ingeniería Mecánica de la Universidad de Cartagena para realizar la simulación del sistema de comunicación de UUA, envío y recepción de datos.

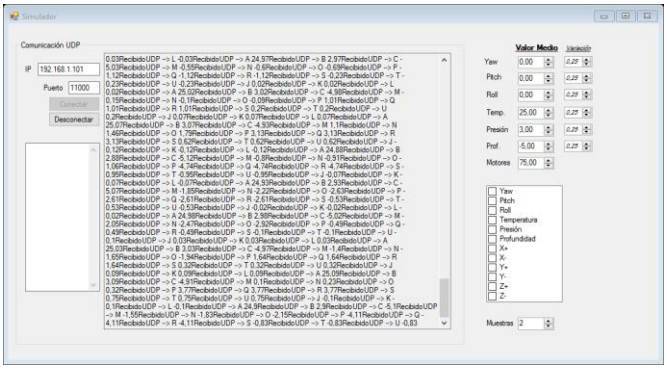
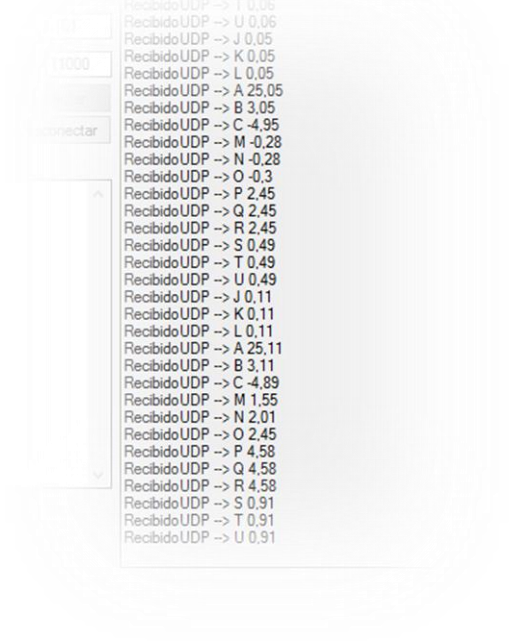


Ilustración 75 - Aplicación Simulador

El protocolo que se siguió en la comunicación fue la de añadir un carácter al valor transmitido, de tal forma que en recepción se pueda saber de qué sensor o a que actuador va referida tal información. La distribución de las variables es la siguiente:



- A Temperatura
- B Presión
- C Profundidad
- D Motor X1
- E Motor X2
- F Motor Y1
- G Motor Y2
- H Motor Z1
- I Motor Z2
- J Yaw
- K Pitch
- L Roll

La aplicación permite configurar un valor central para cada variable el cuál varía según un parámetro de máxima variación según una secuencia aleatoria. En la parte derecha superior puede verse dos selectores numéricos por cada una de las variables en los se puede configurar los parámetros valor medio y variación.

Al cargar la aplicación se obtiene la IP del equipo a través de una función implementada como obtenerIP()

```
Public Function obtenerIP() As String
        Dim t_IP As String
        t_IP                                                                    =
Dns.GetHostEntry(My.Computer.Name).AddressList.FirstOrDefault(Function(i)
i.AddressFamily = Sockets.AddressFamily.InterNetwork).ToString()
        Return t_IP
    End Function
```

Para realizar el inicio de la simulación y de la conexión hay que pulsar en el botón conectar. Al pulsar se inicia una conexión UDP mediante IP para enviar y recibir datos. Se activa un timer llamado Timer3 que es el encargado de realizar los envíos vía UDP y para la recepción UDP se declara un proceso paralelo mediante un thread denominado **ThreadReceive**.

```
Private Sub bT_Conecta_Click(sender As Object, e As EventArgs) Handles
bT_Conecta.Click

        udpClient.Connect(Net.IPAddress.Parse(tB_IP.Text), tB_Port.Text)
        Timer3.Enabled = True
        bT_desc.Enabled = True
        bT_Conecta.Enabled = False

        SocketN = tB_Port.Text
        rxudpClient = New System.Net.Sockets.UdpClient(SocketN)
        ThreadReceive = New System.Threading.Thread(AddressOf ReceiveMessages)
        ThreadReceive.Start()

    End Sub
```

ThreadReceive está enlazado a una función, ReceiveMessages(), que pone en escucha el puerto donde está configurada la conexión UDP. Cuando recibe datos, los representa en el cuadro de texto que se encuentra en la parte central de la aplicación.

```
Public Sub ReceiveMessages()

        Dim receiveBytes As [Byte]() = rxudpClient.Receive(RemoteIPEndPoint)
        tB_IP.Text = RemoteIPEndPoint.Address.ToString
        Dim BitDet As BitArray
        BitDet = New BitArray(receiveBytes)
```

```
        Dim strReturnData As String =
System.Text.Encoding.Unicode.GetString(receiveBytes)
        tB_rx.AppendText("RecibidoUDP --> " &
System.Text.Encoding.ASCII.GetChars(receiveBytes))
        NewInitialize()

    End Sub
```

Cada vez que se ejecuta el evento del **Timer3**, se realiza el envío de datos simulados según el protocolo comentado anteriormente. Al finalizar el envío reconfigura el timer para que realice un numero de envíos por segundo

```
Private Sub Timer3_Tick(sender As Object, e As EventArgs) Handles Timer3.Tick
        Dim txtSnd As Byte() = New Byte() {}

        txtSnd = System.Text.Encoding.ASCII.GetBytes("J " &
(NumericUpDown1.Value + Aleatorio(NumericUpDown14.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd = System.Text.Encoding.ASCII.GetBytes("K " &
(NumericUpDown2.Value + Aleatorio(NumericUpDown13.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd = System.Text.Encoding.ASCII.GetBytes("L " &
(NumericUpDown3.Value + Aleatorio(NumericUpDown12.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd = System.Text.Encoding.ASCII.GetBytes("A " &
(NumericUpDown4.Value + Aleatorio(NumericUpDown11.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd = System.Text.Encoding.ASCII.GetBytes("B " &
(NumericUpDown5.Value + Aleatorio(NumericUpDown10.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd = System.Text.Encoding.ASCII.GetBytes("C " &
(NumericUpDown6.Value + Aleatorio(NumericUpDown9.Value)).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)

        PosX = PosX + Aleatorio(4)
        PosY = PosY + Aleatorio(5)
        PosZ = PosZ + Aleatorio(6)

        txtSnd = System.Text.Encoding.ASCII.GetBytes("M " &
PosX.ToString("####0.##"))
```

```
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("N      "      &
PosY.ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("O      "      &
PosZ.ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("P      "      &
Aleatorio(10).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("Q      "      &
Aleatorio(10).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("R      "      &
Aleatorio(10).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("S      "      &
Aleatorio(2).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("T      "      &
Aleatorio(2).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        txtSnd      =      System.Text.Encoding.ASCII.GetBytes("U      "      &
Aleatorio(2).ToString("####0.##"))
        udpClient.Send(txtSnd, txtSnd.Length)
        Timer3.Interval = 1000 / NumericUpDown8.Value

    End Sub
```

Implementa una función para generar un numero aleatorio denominada **Aleatorio()**, la cual se configura dentro de un rango determinado. Es la encargada de generar variaciones en los datos que se transmiten.

```
Public Function Aleatorio(variacion) As Double
        Dim numAleatorio As New Random()
        Dim valorAleatorio As Integer = numAleatorio.Next(-variacion * 100 / 2,
variacion * 100 / 2)
        Return valorAleatorio / 100
    End Function
```

Diseño de una aplicación destinada a generar Sistemas de Lógica Difusa

Construir una base de reglas en un controlador basado en lógica difusa puede convertirse en una tarea tediosa. Por ello, se decidió implementar en C#, bajo Visual Studio 2015, una herramienta que fuese capaz de generar, visualizar y modificar de una manera más intuitiva las reglas de un sistema de lógica difusa que se generalizo a dos entradas cuyos pesos son configurables, y una única salida, dejando como grado de libertad el número de variables lingüísticas de cada entrada.

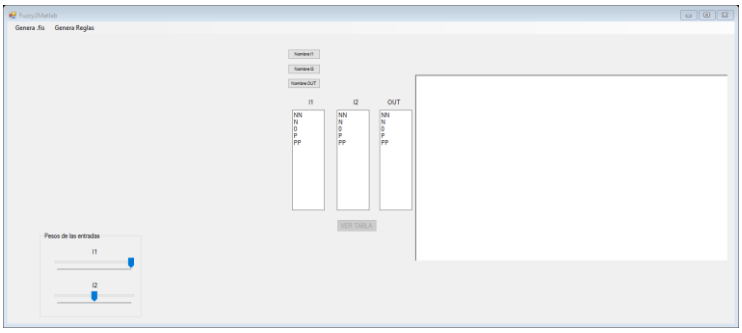


Ilustración 76 - Aplicación Fuzzy2Matlab

Otra funcionalidad que implementa es la de generar un archivo .fis que se pueda cargar directamente en Matlab para realizar los ensayos y simulaciones pertinentes.

Prestando atención a la parte central, existen tres cuadros de texto donde se definen las variables lingüísticas de cada entrada y de la salida del sistema. Una vez se han definido estas etiquetas, se pasa a configurar el peso y la influencia que van a tener la entradas sobre la salida. En la parte inferior izquierda se encuentran dos barras de desplazamiento que controlan los pesos, en la parte central el peso seria nulo, es decir, no existe influencia sobre la salida; hacia la izquierda tendría una influencia negativa y hacia la derecha positiva.

Una vez configurado todo lo anterior se puede proceder a generar las reglas. Si se pulsa en la barra de menú en Generar Reglas se ejecuta un algoritmo cuyo fin es la de presentar las reglas en el cuadro de texto situado en la parte derecha. Este algoritmo se ejecuta mediante una función que se denominó generadorReglas() que realiza los siguientes pasos:

1. Detectar pesos de las entradas
2. Detectar el número de funciones de pertenencia de cada variable
3. Detección del máximo de los pesos
4. Generar matriz de reglas en función de los pesos e influencias
5. Representar en cuadro de texto en formato de línea de comandos para Matlab

Codigo Fuente 19 - Generador de Reglas

```
private void generadorReglas()
        {
            richTextBox1.Text = "";
            int i, j, k = 0;
            int pesoE = trackBar1.Value, pesodE = trackBar2.Value;  //detección de los pesos de las entradas
            int Ne = Convert.ToInt32(textBox1.Lines.LongLength);                    //número de funciones de pertenencia
            int NdE = Convert.ToInt32(textBox2.Lines.LongLength);
            int Nout = Convert.ToInt32(textBox3.Lines.LongLength);
            Console.WriteLine("Ne = " + Ne + "\nNdE = " + NdE);
            int maximo = Math.Max(Math.Abs(pesoE), Math.Abs(pesodE)); //deteccion del máximo de los pesos

            int [,]  reglasG = new int[Ne,NdE];
            rules = new int[Ne * NdE];
            string[] lineas = new string[textBox3.Lines.LongLength];

            int ajuste1 = Ne / 2;
            int ajuste2 = NdE / 2;
            int ajuste3 = Nout / 2;

            richTextBox1.AppendText("a.output(" + 1 + ").name= " + "VAR1" + "; a.output(" + 1 + ").range=[" + (-1) + " " + 1 +
"];\n");
```

```
            richTextBox1.AppendText("a.output(1).mf("  +  (1)  +  ").name='"+(textBox3.Text.Split('\n')[0]).Replace('\r',' ')  +  "';
a.output(1).mf(" + (1) + ").type='trapmf'; a.output(1).mf(" + (1) + ").params=[-M1 -M1 -M2 -M3];\n");
            int M=0, r = 0;
            for (i=1;i<Nout-1;i++)
            {

                if(i<(Nout/2))
                {
                    richTextBox1.AppendText("a.output(1).mf(" + (i+1) + ").name='" + (textBox3.Text.Split('\n')[i]).Replace('\r', '
') + "'; a.output(1).mf(" + (i) + ").type='trimf'; a.output(1).mf(" + (i + 1) + ").params=[-M" + (i + 1) + " -M" + (i + 2) + " -M" +
(i + 3) + "];\n");
                }
                else if (i < (Nout / 2 - 1))
                {
                    richTextBox1.AppendText("a.output(1).mf(" + (i + 1) + ").name='" + (textBox3.Text.Split('\n')[i]).Replace('\r',
' ') + "'; a.output(1).mf(" + (i) + ").type='trimf'; a.output(1).mf(" + (i + 1) + ").params=[-M" + (0) + " M" + (i + 3) + " " + (i+2)
+ "];\n");
                }
                else if (i == (Math.Ceiling(Convert.ToDouble(Nout/2))) )
                {
                    richTextBox1.AppendText("a.output(1).mf(" + (i + 1) + ").name='" + (textBox3.Text.Split('\n')[i]).Replace('\r',
' ') + "'; a.output(1).mf(" + (i) + ").type='trimf'; a.output(1).mf(" + (i + 1) + ").params=[-M" + (i + 2) + "  " + (0) + " M" + (i
+ 2) + "];\n");
                    M = i + 2;
                }
```

```
                else if(i < (Nout / 2 +1))
                {
                    richTextBox1.AppendText("a.output(1).mf(" + (i + 1) + ").name='" + (textBox3.Text.Split('\n')[i]).Replace('\r',
' ') + "'; a.output(1).mf(" + (i) + ").type='trimf'; a.output(1).mf(" + (i + 1) + ").params=[-M" + (M-r) + " M" + (M -r-1) + " " +
(0) + "];\n");
                    r++;
                }
                else
                {
                    richTextBox1.AppendText("a.output(1).mf(" + (i + 1) + ").name='" + (textBox3.Text.Split('\n')[i]).Replace('\r',
' ') + "'; a.output(1).mf(" + (i) + ").type='trimf'; a.output(1).mf(" + (i + 1) + ").params=[M" + (M-r) + " M" + (M - r-1) + " M" +
(M - r - 2) + "];\n");
                    r++;
                }
            }

            richTextBox1.AppendText("a.output(1).mf(" + (Nout) + ").name='" + (textBox3.Text.Split('\n')[Nout-1]).Replace('\r', ' ')
+ "'; a.output(1).mf(" + (Nout) + ").type='trapmf'; a.output(1).mf(" + (Nout) + ").params=[M3 M2 M1 M1];\n");
            richTextBox1.AppendText("\n \n");
            lineas = textBox3.Lines;



            Console.WriteLine("Peso 1= " + (pesoE*100/maximo) + " \n" +"Peso 2= " + pesodE*100/maximo );
                    Console.WriteLine("---> CONJUNTO DE REGLAS <---");
            int[] ling1 = new int[Ne];
```

```
            int[] ling2 = new int[NdE];

            for (i = -Ne / 2 ; i <= Ne / 2 ; i++) ling1[i + Ne  / 2] = i;
            for (i = -NdE / 2; i <= NdE / 2; i++) ling2[i + NdE / 2] = i;

            for (j=0;j<NdE;j++)
            {
                for (i=0;i<Ne;i++)
                {
                    reglasG[i, j] = (ling1[i] * pesoE + ling2[j] * pesodE) / maximo;

                    if (reglasG[i, j] < (-ajuste3)) reglasG[i, j] = -ajuste3;
                    else if (reglasG[i, j] > (ajuste3)) reglasG[i, j] = ajuste3;
                    reglasG[i, j] = -reglasG[i, j];

                    Console.Write("\t " + reglasG[i , j ] );
                    rules[k] = reglasG[i, j]+ ajuste1;

                    richTextBox1.AppendText("a.rule(" + (k+1) + ").antecedent= [" + (j + 1) + " " + (i + 1) + "] ; a.rule(" + (k +
1) + ").consequent=[" + (reglasG[i, j]+ajuste3+1) + " " + (reglasG[i, j] + ajuste3+1) + "]; a.rule(" + (k + 1) + ").weight=1; a.rule("
+ (k + 1) + ").connection=1; \n");
                    k++;
                }
                Console.WriteLine(" ");
```

```
            }



        }
```

Ejecutar Python Script desde C#

La idea de esta prueba es la poder invocar un script realizado en Python. Posteriormente se implementará un script capaz de capturar video desde una Pi Camera y transmitir vía UDP.

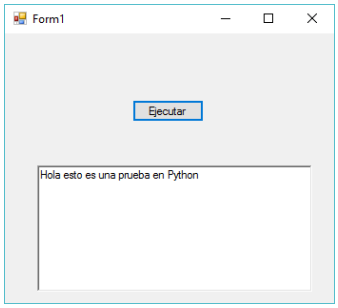


Ilustración 77 -
Aplicación Python Script desde C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EjecutarPythonScript
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            run_cmd();

        }
        private void run_cmd()
        {

            string fileName = @"C:\Python27\prueba.py";
```

```
            Process p = new Process();
            p.StartInfo    =    new    ProcessStartInfo(@"C:\Python27\python.exe",
fileName)
            {
                RedirectStandardOutput = true,
                UseShellExecute = false,
                CreateNoWindow = true
            };
            p.Start();

            string output = p.StandardOutput.ReadToEnd();
            p.WaitForExit();
            richTextBox1.AppendText(output + "\n");
        }

    }
}
```

Implementación del modelo en C#

En modelo realizado por Javier de la Red ha sido el referente en este proyecto para la realización de todos los ensayos de los controladores desarrollados en Matlab.

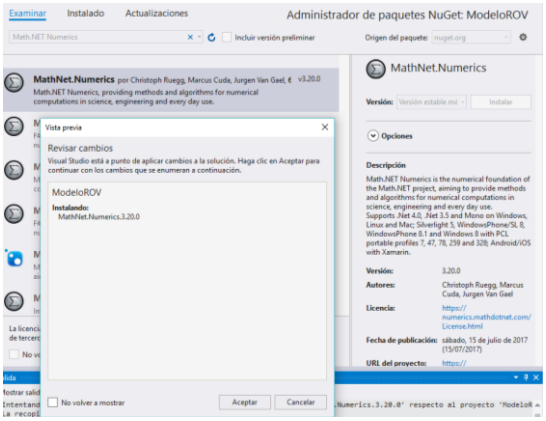


Ilustración 78 - MathNet.Numerics

Debido a que no se han podido realizar las pruebas pertinentes en el dispositivo real, se lleva a cabo la implementación de una clase en C# que es el lenguaje el cual se ejecutara en Raspberry Pi junto a los sistemas de lógica difusa y comunicaciones.

Se hace uso de una librería llamada *MathNet.Numerics* la cual nos facilita el manejo y operaciones con matrices y vectores.

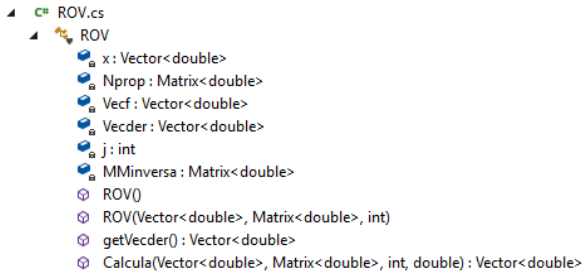


Ilustración 79 - Estructura del modelo creado

Codigo Fuente 20 - Modelo ROV en C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.LinearAlgebra.Double;
namespace ModeloROV2
{
    class ROV
```

```
    {
        //VARIABLES DE ENTRADA
        private Vector<double> x;
        private Matrix<double> Nprop;


        //VARIABLES DE SALIDA
        private Vector<double> Vecf = Vector<double>.Build.Dense(6);
        private Vector<double> Vecder = Vector<double>.Build.Dense(12);
        int j;


        Matrix<double> MMinversa = Matrix<double>.Build.DenseOfColumnArrays(
           new[] { 0.0779199300759296, 0, 0, 0, -0.00588630176125979, 0 },
           new[] { 0, 0.0779199300759296, 0, 0.00588630176125979, 0, 0 },
           new[] { 0, 0, 0.0779199300759296, 0, 0, 0 },
           new[] { 0, 0.00588630176125979, 0, 0.551840790118105, 0, 0 },
           new[] { -0.00588630176125979, 0, 0, 0, 0.551840790118105, 0 },
           new[] { 0, 0, 0, 0, 0, 0.5543 }
           );

        public ROV()
        {

        }

        public ROV(Vector<double> x, Matrix<double> nprop,  int j)
        {
            this.x = x;
            Nprop = nprop;
            this.j = j;
        }
        public Vector<double>  getVecder () { return Vecder; }

        public  Vector<double> Calcula(Vector<double> x, Matrix<double> nprop,
int j,double h)
        {
            this.x = x;
            Nprop = nprop;
            this.j = j;
            /*  Saturaciones del Eje */
```

```
            for (int i = 0; i < 6; i++)
            {
                if (Math.Abs(Nprop[j, i]) > 3800)
                {
                    Nprop[j, i] = Math.Sign(Nprop[j, i]) * 3800;
                }
            }
            //  Cálculo de las fuerzas de propulsión
            Vector<double> Np = Vector<double>.Build.Dense(6);
            Vector<double> T = Vector<double>.Build.Dense(6);
            for (int i = 0; i < 6; i++)
            {
                Np[i] = Nprop[j, i] / 60 * 2 * Math.PI;
                T[i] = 6.5e-5 * Np[i] * Math.Abs(Np[i]);

       /*  el efecto de los propulsores se considera unicamente cuadrático */
            }

            double u = x[0];
            double v = x[1];
            double w = x[2];
            double p = x[3];
            double q = x[4];
            double r = x[5];
            double phi = x[9];
            double theta = x[10];
            double psi = x[11];


            /*  Fuerzas y Momentos Hidrodinámicos */
            double X = -(-2.72) * Math.Sin(theta) + -35.0 * u * Math.Abs(u) + -
7 * u - 8.56269113149847 * (w * q - v * r - 0 * (Math.Pow(q, 2) + Math.Pow(r,
2)) + 0 * p * q + 0.016 * p * r) + T[0] + T[1];
            double Y = (-2.72) * Math.Cos(theta) * Math.Sin(phi) + -35.0 * v *
Math.Abs(v) + -7 * v - 8.56269113149847 * (u * r - w * p - 0 * (Math.Pow(p, 2)
+ Math.Pow(r, 2)) + 0 * p * q + 0.016 * q * r) + T[4] + T[5];
            double Z = (-2.72) * Math.Cos(theta) * Math.Cos(phi) + -35 * w *
Math.Abs(w) + -7 * w - 8.56269113149847 * (v * p - u * q - 0.016 * (Math.Pow(p,
2) + Math.Pow(q, 2)) + 0 * p * r + 0 * q * r) + T[2] + T[3];
            double K = -0.016 * 84 * Math.Cos(theta) * Math.Sin(phi) + -2 * p *
Math.Abs(p) + -1 * p + (1.0104e-1 - 9.1442e-2) * q * r + 8.56269113149847 * 0 *
```

```
(u * q - v * p) + 8.56269113149847 * 0.016 * (u * r - w * p) + 0.155 * (T[5] -
T[4]);
            double M = -0.016 * 84 * Math.Sin(theta) + -2 * q * Math.Abs(q) + -
1 * q + (9.1442e-2 - 1.0104e-1) * r * p + 8.56269113149847 * 0.016 * (v * r - w
* q) + 8.56269113149847 * 0 * (v * p - u * q) + 0.155 * (T[3] - T[2]);
            double N = -2 * r * Math.Abs(r) + -1 * r + (1.0104e-1 - 1.0104e-1)
* p * q + 8.56269113149847 * 0 * (w * q - v * r) + 8.56269113149847 * 0 * (w *
p - u * r) + 0.155 * (T[1] - T[0]);
            double udot = MMinversa[0, 0] * X + MMinversa[0, 1] * Y + MMinversa[0,
2] * Z + MMinversa[0, 3] * K + MMinversa[0, 4] * M + MMinversa[0, 5] * N;
            double vdot = MMinversa[1, 0] * X + MMinversa[1, 1] * Y + MMinversa[1,
2] * Z + MMinversa[1, 3] * K + MMinversa[1, 4] * M + MMinversa[1, 5] * N;

            K = -1.344 * Math.Cos(theta) * Math.Sin(phi) + -2 * p * Math.Abs(p)
+ -1 * p + (0.009598) * q * r + +137.003058103976e-003 * (u * r - w * p) -
137.003058103976e-003 * vdot + 0.155 * (T[5] - T[4]);
            M = -1.344 * Math.Sin(theta) + -2 * q * Math.Abs(q) + -1 * q + (0.00)
* r * p + 0.137 * (v * r - w * q) - 0.137 * udot + 0.155 * (T[3] - T[2]);

            Vecder[0] = MMinversa[0, 0] * X + MMinversa[0, 1] * Y + MMinversa[0,
2] * Z + MMinversa[0, 3] * K + MMinversa[0, 4] * M + MMinversa[0, 5] * N;
            Vecder[1] = MMinversa[1, 0] * X + MMinversa[1, 1] * Y + MMinversa[1,
2] * Z + MMinversa[1, 3] * K + MMinversa[1, 4] * M + MMinversa[1, 5] * N;
            Vecder[2] = MMinversa[2, 0] * X + MMinversa[2, 1] * Y + MMinversa[2,
2] * Z + MMinversa[2, 3] * K + MMinversa[2, 4] * M + MMinversa[2, 5] * N;
            Vecder[3] = MMinversa[3, 0] * X + MMinversa[3, 1] * Y + MMinversa[3,
2] * Z + MMinversa[3, 3] * K + MMinversa[3, 4] * M + MMinversa[3, 5] * N;
            Vecder[4] = MMinversa[4, 0] * X + MMinversa[4, 1] * Y + MMinversa[4,
2] * Z + MMinversa[4, 3] * K + MMinversa[4, 4] * M + MMinversa[4, 5] * N;
            Vecder[5] = MMinversa[5, 0] * X + MMinversa[5, 1] * Y + MMinversa[5,
2] * Z + MMinversa[5, 3] * K + MMinversa[5, 4] * M + MMinversa[5, 5] * N;
            Vecder[6] = Math.Cos(psi) * Math.Cos(theta) * u + (Math.Cos(psi) *
Math.Sin(theta) * Math.Sin(phi) - Math.Sin(psi) * Math.Cos(phi)) * v +
(Math.Sin(psi) * Math.Sin(phi) + Math.Cos(psi) * Math.Cos(phi) * Math.Sin(theta))
* w;
            Vecder[7] = Math.Sin(psi) * Math.Cos(theta) * u + (Math.Cos(phi) *
Math.Cos(psi) + Math.Sin(phi) * Math.Sin(theta) * Math.Sin(psi)) * v +
(Math.Cos(phi) * Math.Sin(theta) * Math.Sin(psi) - Math.Cos(psi) * Math.Sin(phi))
* w;
            Vecder[8] = -Math.Sin(theta) * u + Math.Cos(theta) * Math.Sin(phi)
* v + Math.Cos(phi) * Math.Cos(theta) * w;
```

```
            Vecder[9] = p + Math.Sin(phi) * Math.Tan(theta) * q + Math.Cos(phi)
* Math.Tan(theta) * r;
            Vecder[10] = Math.Cos(phi) * q - Math.Sin(phi) * r;
            Vecder[11] = Math.Sin(phi) / Math.Cos(theta) * q + Math.Cos(phi) /
Math.Cos(theta) * r;
            return (x+Vecder*h);   //Devuelve el vector nuevo Estado
        }

       }

    }
```

Ejemplo de test en el que se ejecuta el modelo con un control a lazo abierto con los propulsores parados durante 30 segundos con un paso de 0.01.

Codigo Fuente 21 - Programa de Test del modelo

```
using MathNet.Numerics.LinearAlgebra;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TestROV
{
    class Program
    {
        static void Main(string[] args)
        {

            ROV miRov = new ROV();


            double tf = 30;         //Tiempo Final de la Simulación(s)
            double h = 0.01;        // Paso de simulación

            int Niter = (int)System.Math.Ceiling(tf / h);
```

```
            Vector<double> x = Vector<double>.Build.Dense(12);
            Matrix<double> Nprop = Matrix<double>.Build.Dense(Niter+1, 6);



            Console.WriteLine("             Test del Modelo ROV en C#");
            Console.WriteLine("             -------------------------\n\n");

            Console.WriteLine("Vector ''x'' " + x.ToString());
            Console.WriteLine("Matriz ''Nprop'' " + Nprop.ToString());



            int j=0;
            int repite = 0;

            for (repite = 1; repite <= Niter; repite++)
            {
                j = repite;
                x = miRov.Calcula(x, Nprop, j, h);  //Realizamos el cálculo

            }
            Console.WriteLine("Vector ''x'' para t= " + (j*h).ToString() +"    "
+ x.ToString());
            Console.ReadKey();
        }
    }
}
```

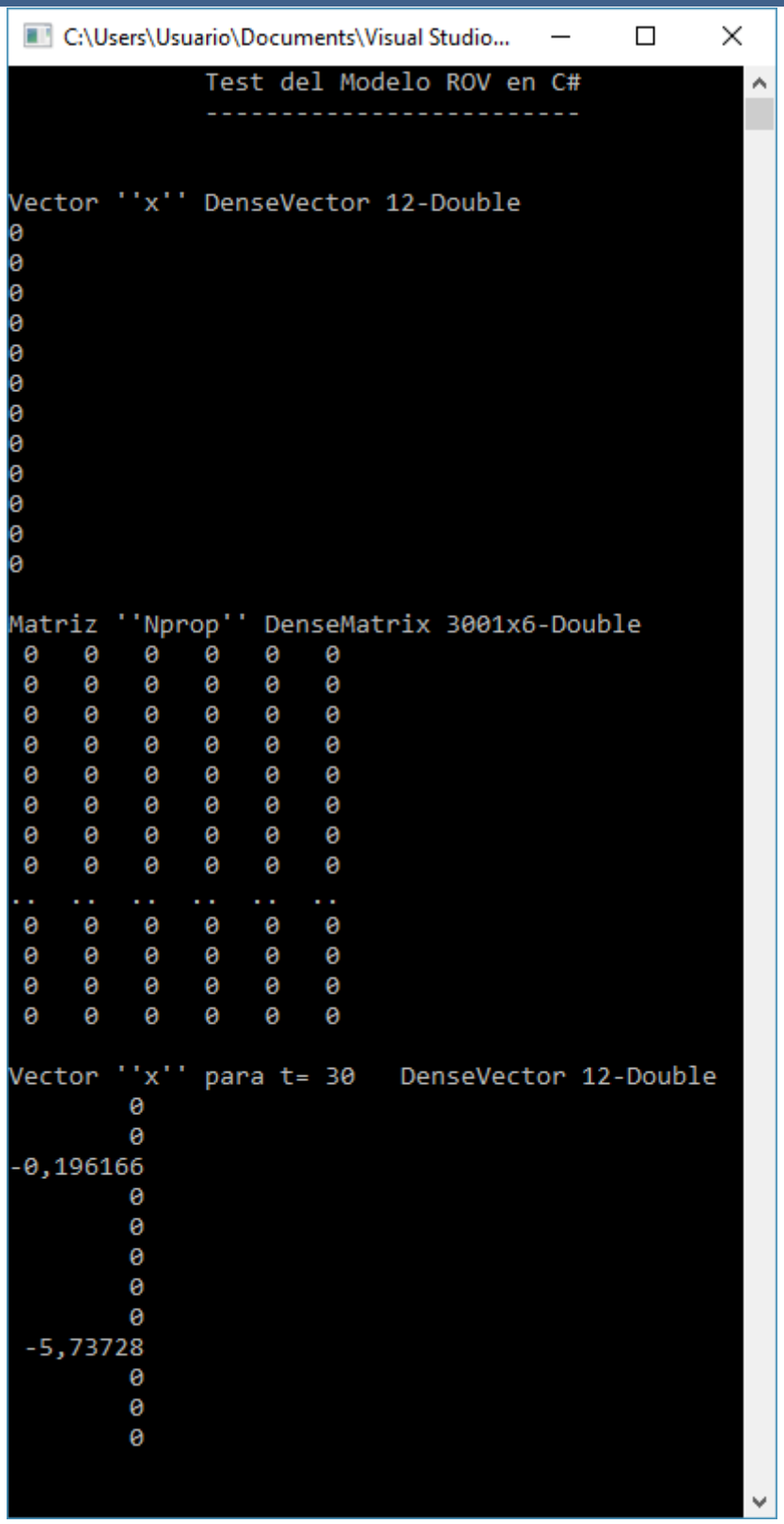
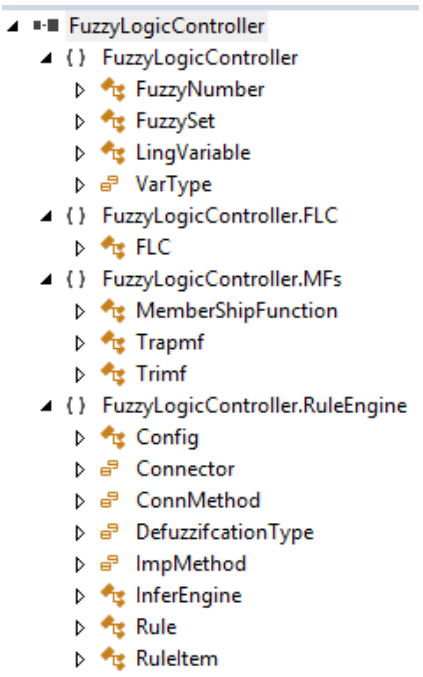


Ilustración 80 - Resultados de Modelo ROV en C#

El siguiente paso que se siguió fue el crear clases de cada uno de los controles basados en lógica difusa que se desarrollaron en Matlab. Para llevar a cabo esta tarea se hace uso de "FuzzyLogicController.dll", una dll desarrollada y publicada por Hesham Omran en CodeProject.



Se decidió implementar una clase por cada uno de los controladores difusos debido a la forma de implementar la base de reglas.

A continuación se muestran cada una de las clases implementadas:

- Control de estabilidad de profundidad
- Control de estabilidad ángulo PSI
- Control de estabilidad ángulo THETA
- Control de estabilidad ángulo PHI

Clase para Control de estabilidad de profundidad

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FuzzyLogicController;
using FuzzyLogicController.FLC;
using FuzzyLogicController.MFs;
using FuzzyLogicController.RuleEngine;
using System.Windows.Forms;

namespace ModeloROV2
{
   internal class ControladorZ
   {
```

```
double in1 = 0, in2 = 0;
private Config conf = new Config(ImpMethod.Prod, ConnMethod.Min);

private LingVariable e = new LingVariable("e", VarType.Input);
private LingVariable de = new LingVariable("de", VarType.Input);
private LingVariable MOTOR1 = new LingVariable("MOTOR1", VarType.Output);

private FLC c;
private FuzzySet set1;
private FuzzySet set2;
private List<FuzzySet> fuzset;

List<RuleItem> rule1in = new List<RuleItem>();
List<RuleItem> rule1out = new List<RuleItem>();
List<RuleItem> rule2in = new List<RuleItem>();
List<RuleItem> rule2out = new List<RuleItem>();
List<RuleItem> rule3in = new List<RuleItem>();
List<RuleItem> rule3out = new List<RuleItem>();
List<RuleItem> rule4in = new List<RuleItem>();
List<RuleItem> rule4out = new List<RuleItem>();
List<RuleItem> rule5in = new List<RuleItem>();
List<RuleItem> rule5out = new List<RuleItem>();
List<RuleItem> rule6in = new List<RuleItem>();
List<RuleItem> rule6out = new List<RuleItem>();
List<RuleItem> rule7in = new List<RuleItem>();
List<RuleItem> rule7out = new List<RuleItem>();
List<RuleItem> rule8in = new List<RuleItem>();
List<RuleItem> rule8out = new List<RuleItem>();
List<RuleItem> rule9in = new List<RuleItem>();
List<RuleItem> rule9out = new List<RuleItem>();
List<RuleItem> rule10in = new List<RuleItem>();
List<RuleItem> rule10out = new List<RuleItem>();
List<RuleItem> rule11in = new List<RuleItem>();
List<RuleItem> rule11out = new List<RuleItem>();
List<RuleItem> rule12in = new List<RuleItem>();
List<RuleItem> rule12out = new List<RuleItem>();
List<RuleItem> rule13in = new List<RuleItem>();
List<RuleItem> rule13out = new List<RuleItem>();
List<RuleItem> rule14in = new List<RuleItem>();
```

```
List<RuleItem> rule14out = new List<RuleItem>();
List<RuleItem> rule15in = new List<RuleItem>();
List<RuleItem> rule15out = new List<RuleItem>();
List<RuleItem> rule16in = new List<RuleItem>();
List<RuleItem> rule16out = new List<RuleItem>();
List<RuleItem> rule17in = new List<RuleItem>();
List<RuleItem> rule17out = new List<RuleItem>();
List<RuleItem> rule18in = new List<RuleItem>();
List<RuleItem> rule18out = new List<RuleItem>();
List<RuleItem> rule19in = new List<RuleItem>();
List<RuleItem> rule19out = new List<RuleItem>();
List<RuleItem> rule20in = new List<RuleItem>();
List<RuleItem> rule20out = new List<RuleItem>();
List<RuleItem> rule21in = new List<RuleItem>();
List<RuleItem> rule21out = new List<RuleItem>();
List<RuleItem> rule22in = new List<RuleItem>();
List<RuleItem> rule22out = new List<RuleItem>();
List<RuleItem> rule23in = new List<RuleItem>();
List<RuleItem> rule23out = new List<RuleItem>();
List<RuleItem> rule24in = new List<RuleItem>();
List<RuleItem> rule24out = new List<RuleItem>();
List<RuleItem> rule25in = new List<RuleItem>();
List<RuleItem> rule25out = new List<RuleItem>();
List<Rule> rules;
List<FuzzySet> impli;
InferEngine engine;
public ControladorZ()
{
   double P1 =30;
   double P2 =5.81;
   double P3 =4.54;
   double Q1 =5;
   double Q2 =1.82;
   double Q3 =1.09;

   double M1 =1;
   double M2 =0.33;
   double M3 =0.16;
   double M4 =0.12;
```

```
    double M5 =0.11;
    double M6 = 0.09;



    //Declaracion de variable lingüistica Error "e"
    e.setRange(-30, 30);
    e.addMF(new Trapmf("NN", -P1, - P1, - P2, - P3));
    e.addMF(new Trimf("N", -P2 ,- P3 ,0));
    e.addMF(new Trimf("Z", -P3, 0, P3));
    e.addMF(new Trimf("P", 0 ,P3 ,P2));
    e.addMF(new Trapmf("PP", P3, P2, P1, P1));

    //Declaracion de variable lingüistica dError "de"

    de.setRange(-10, 10);
    de.addMF(new Trapmf("NN", -Q1 ,- Q1, - Q2, - Q3));
    de.addMF(new Trimf("N", -Q2 ,- Q3, 0));
    de.addMF(new Trimf("Z", -Q3, 0, Q3));
    de.addMF(new Trimf("P", 0, Q3, Q2));
    de.addMF(new Trapmf("PP", Q3, Q2, Q1, Q1));

    //Declaracion de variable lingüistica MOTOR1 "MOTOR1"

    MOTOR1.setRange(-20, 20);
    MOTOR1.addMF(new Trapmf("NNNN", -M1, - M1, - M2, - M3));
    MOTOR1.addMF(new Trimf("NNN", -M2, - M3, - M4));
    MOTOR1.addMF(new Trimf("NN", -M3, - M4, - M5));
    MOTOR1.addMF(new Trimf("N", -M4, - M5, - M6));
    MOTOR1.addMF(new Trimf("Z", -M6,  0, M6));
    MOTOR1.addMF(new Trimf("P", M6, M5, M4));
    MOTOR1.addMF(new Trimf("PP", M5, M4, M3));
    MOTOR1.addMF(new Trimf("PPP", M4, M3, M2));
    MOTOR1.addMF(new Trapmf("PPPP", M3, M2, M1, M1));



}
public void setIn(double a, double b)
{
    in1 = a;
```

```
    in2 = b;
}
public void Fuzz()
{
    c = new FLC(conf);

    set1 = new FuzzySet(c.Fuzzification(in1, e), e.Name);
    set2 = new FuzzySet(c.Fuzzification(in2, de), de.Name);
    fuzset = new List<FuzzySet>();
    fuzset.Add(set1);
    fuzset.Add(set2);

}
public void reglas()
{
    rule1in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "NN") });
    rule1out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule2in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "N") });
    rule2out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
    rule3in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "Z") });
    rule3out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
    rule4in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "P") });
    rule4out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule5in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "PP") });
    rule5out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });

    rule6in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "NN") });
    rule6out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
    rule7in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "N") });
    rule7out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule8in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "Z") });
    rule8out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule9in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "P") });
    rule9out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule10in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "PP") });
    rule10out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });

    rule11in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "NN") });
    rule11out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
```

```
rule12in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "N") });
rule12out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule13in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "Z") });
rule13out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule14in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "P") });
rule14out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule15in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "PP") });
rule15out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });

rule16in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "NN") });
rule16out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule17in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "N") });
rule17out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule18in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "Z") });
rule18out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule19in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "P") });
rule19out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
rule20in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "PP") });
rule20out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });

rule21in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "NN") });
rule21out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule22in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "N") });
rule22out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule23in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "Z") });
rule23out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
rule24in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "P") });
rule24out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });
rule25in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "PP") });
rule25out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });

rules = new List<Rule>();
rules.Add(new Rule(rule1in, rule1out, Connector.And));
rules.Add(new Rule(rule2in, rule2out, Connector.And));
rules.Add(new Rule(rule3in, rule3out, Connector.And));
rules.Add(new Rule(rule4in, rule4out, Connector.And));
rules.Add(new Rule(rule5in, rule5out, Connector.And));
rules.Add(new Rule(rule6in, rule6out, Connector.And));
rules.Add(new Rule(rule7in, rule7out, Connector.And));
```

```
            rules.Add(new Rule(rule8in, rule8out, Connector.And));
            rules.Add(new Rule(rule9in, rule9out, Connector.And));
            rules.Add(new Rule(rule10in, rule11out, Connector.And));
            rules.Add(new Rule(rule12in, rule12out, Connector.And));
            rules.Add(new Rule(rule13in, rule13out, Connector.And));
            rules.Add(new Rule(rule14in, rule14out, Connector.And));
            rules.Add(new Rule(rule15in, rule15out, Connector.And));
            rules.Add(new Rule(rule16in, rule16out, Connector.And));
            rules.Add(new Rule(rule17in, rule17out, Connector.And));
            rules.Add(new Rule(rule18in, rule18out, Connector.And));
            rules.Add(new Rule(rule19in, rule19out, Connector.And));
            rules.Add(new Rule(rule20in, rule20out, Connector.And));
            rules.Add(new Rule(rule21in, rule21out, Connector.And));
            rules.Add(new Rule(rule22in, rule22out, Connector.And));
            rules.Add(new Rule(rule23in, rule23out, Connector.And));
            rules.Add(new Rule(rule24in, rule24out, Connector.And));
            rules.Add(new Rule(rule25in, rule25out, Connector.And));
        }
        public void inferencia()
        {
            engine = new InferEngine(conf, rules, fuzset);
            impli = engine.evaluateRules();
        }
        public String Deffuz()
        {
            double crisp1 = c.DeFuzzification(impli, MOTOR1);
            return crisp1.ToString();
        }
        public String salida(double In1, double In2)
        {
            setIn(In1, In2);
            Fuzz();
            reglas();
            inferencia();
            String fin = Deffuz();
            return fin;
        }
    }
}
```

Clase para Control de estabilidad ángulo PSI

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FuzzyLogicController;
using FuzzyLogicController.FLC;
using FuzzyLogicController.MFs;
using FuzzyLogicController.RuleEngine;
using System.Windows.Forms;

namespace ModeloROV2
{
   class ControladorPSI
   {
      double in1 = 0, in2 = 0;
      private Config conf = new Config(ImpMethod.Prod, ConnMethod.Min);

      private LingVariable e = new LingVariable("e", VarType.Input);
      private LingVariable de = new LingVariable("de", VarType.Input);
      private LingVariable MOTOR1 = new LingVariable("MOTOR1", VarType.Output);

      private FLC c;
      private FuzzySet set1;
      private FuzzySet set2;
      private List<FuzzySet> fuzset;

      List<RuleItem> rule1in = new List<RuleItem>();
      List<RuleItem> rule1out = new List<RuleItem>();
      List<RuleItem> rule2in = new List<RuleItem>();
      List<RuleItem> rule2out = new List<RuleItem>();
      List<RuleItem> rule3in = new List<RuleItem>();
      List<RuleItem> rule3out = new List<RuleItem>();
      List<RuleItem> rule4in = new List<RuleItem>();
      List<RuleItem> rule4out = new List<RuleItem>();
      List<RuleItem> rule5in = new List<RuleItem>();
      List<RuleItem> rule5out = new List<RuleItem>();
      List<RuleItem> rule6in = new List<RuleItem>();
```

```
List<RuleItem> rule6out = new List<RuleItem>();
List<RuleItem> rule7in = new List<RuleItem>();
List<RuleItem> rule7out = new List<RuleItem>();
List<RuleItem> rule8in = new List<RuleItem>();
List<RuleItem> rule8out = new List<RuleItem>();
List<RuleItem> rule9in = new List<RuleItem>();
List<RuleItem> rule9out = new List<RuleItem>();
List<RuleItem> rule10in = new List<RuleItem>();
List<RuleItem> rule10out = new List<RuleItem>();
List<RuleItem> rule11in = new List<RuleItem>();
List<RuleItem> rule11out = new List<RuleItem>();
List<RuleItem> rule12in = new List<RuleItem>();
List<RuleItem> rule12out = new List<RuleItem>();
List<RuleItem> rule13in = new List<RuleItem>();
List<RuleItem> rule13out = new List<RuleItem>();
List<RuleItem> rule14in = new List<RuleItem>();
List<RuleItem> rule14out = new List<RuleItem>();
List<RuleItem> rule15in = new List<RuleItem>();
List<RuleItem> rule15out = new List<RuleItem>();
List<RuleItem> rule16in = new List<RuleItem>();
List<RuleItem> rule16out = new List<RuleItem>();
List<RuleItem> rule17in = new List<RuleItem>();
List<RuleItem> rule17out = new List<RuleItem>();
List<RuleItem> rule18in = new List<RuleItem>();
List<RuleItem> rule18out = new List<RuleItem>();
List<RuleItem> rule19in = new List<RuleItem>();
List<RuleItem> rule19out = new List<RuleItem>();
List<RuleItem> rule20in = new List<RuleItem>();
List<RuleItem> rule20out = new List<RuleItem>();
List<RuleItem> rule21in = new List<RuleItem>();
List<RuleItem> rule21out = new List<RuleItem>();
List<RuleItem> rule22in = new List<RuleItem>();
List<RuleItem> rule22out = new List<RuleItem>();
List<RuleItem> rule23in = new List<RuleItem>();
List<RuleItem> rule23out = new List<RuleItem>();
List<RuleItem> rule24in = new List<RuleItem>();
List<RuleItem> rule24out = new List<RuleItem>();
List<RuleItem> rule25in = new List<RuleItem>();
List<RuleItem> rule25out = new List<RuleItem>();
```

```
List<Rule> rules;
List<FuzzySet> impli;
InferEngine engine;

public ControladorPSI()
{
   double P1 = 30;
   double P2 = 20;
   double P3 = 1;
   double Q1 = 6;
   double Q2 = 3;
   double Q3 = 1;

   double M1 = 20;
   double M2 = 15;
   double M3 = 4;
   double M4 = 3;
   double M5 = 2;
   double M6 = 1;


   //Declaracion de variable lingüistica Error "e"
   e.setRange(-31.1, 31.1);
   e.addMF(new Trapmf("NN", -P1, -P1, -P2, -P3));
   e.addMF(new Trimf("N", -P2, -P3, 0));
   e.addMF(new Trimf("Z", -P3, 0, P3));
   e.addMF(new Trimf("P", 0, P3, P2));
   e.addMF(new Trapmf("PP", P3, P2, P1, P1));

   //Declaracion de variable lingüistica dError "de"

   de.setRange(-6.1, 6.1);
   de.addMF(new Trapmf("NN", -Q1, -Q1, -Q2, -Q3));
   de.addMF(new Trimf("N", -Q2, -Q3, 0));
   de.addMF(new Trimf("Z", -Q3, 0, Q3));
   de.addMF(new Trimf("P", 0, Q3, Q2));
   de.addMF(new Trapmf("PP", Q3, Q2, Q1, Q1));

   //Declaracion de variable lingüistica MOTOR1 "MOTOR1"
```

```
    MOTOR1.setRange(-20.1, 20.1);
    MOTOR1.addMF(new Trapmf("NNNN", -M1, -M1, -M2, -M3));
    MOTOR1.addMF(new Trimf("NNN", -M2, -M3, -M4));
    MOTOR1.addMF(new Trimf("NN", -M3, -M4, -M5));
    MOTOR1.addMF(new Trimf("N", -M4, -M5, -M6));
    MOTOR1.addMF(new Trimf("Z", -M6, 0, M6));
    MOTOR1.addMF(new Trimf("P", M6, M5, M4));
    MOTOR1.addMF(new Trimf("PP", M5, M4, M3));
    MOTOR1.addMF(new Trimf("PPP", M4, M3, M2));
    MOTOR1.addMF(new Trapmf("PPPP", M3, M2, M1, M1));



}
public void setIn(double a, double b)
{
    in1 = a;
    in2 = b;
}
public void Fuzz()
{
    c = new FLC(conf);

    set1 = new FuzzySet(c.Fuzzification(in1, e), e.Name);
    set2 = new FuzzySet(c.Fuzzification(in2, de), de.Name);
    fuzset = new List<FuzzySet>();
    fuzset.Add(set1);
    fuzset.Add(set2);

}
public void reglas()
{
    rule1in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "NN") });
    rule1out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule2in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "N") });
    rule2out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
    rule3in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "Z") });
    rule3out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
    rule4in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "P") });
```

```
rule4out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule5in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "PP") });
rule5out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });

rule6in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "NN") });
rule6out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
rule7in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "N") });
rule7out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule8in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "Z") });
rule8out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule9in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "P") });
rule9out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule10in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "PP") });
rule10out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });

rule11in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "NN") });
rule11out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
rule12in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "N") });
rule12out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule13in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "Z") });
rule13out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule14in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "P") });
rule14out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule15in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "PP") });
rule15out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });

rule16in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "NN") });
rule16out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
rule17in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "N") });
rule17out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule18in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "Z") });
rule18out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule19in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "P") });
rule19out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
rule20in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "PP") });
rule20out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });

rule21in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "NN") });
rule21out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
```

```
    rule22in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "N") });
    rule22out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
    rule23in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "Z") });
    rule23out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
    rule24in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "P") });
    rule24out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });
    rule25in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "PP") });
    rule25out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });

    rules = new List<Rule>();
    rules.Add(new Rule(rule1in, rule1out, Connector.And));
    rules.Add(new Rule(rule2in, rule2out, Connector.And));
    rules.Add(new Rule(rule3in, rule3out, Connector.And));
    rules.Add(new Rule(rule4in, rule4out, Connector.And));
    rules.Add(new Rule(rule5in, rule5out, Connector.And));
    rules.Add(new Rule(rule6in, rule6out, Connector.And));
    rules.Add(new Rule(rule7in, rule7out, Connector.And));
    rules.Add(new Rule(rule8in, rule8out, Connector.And));
    rules.Add(new Rule(rule9in, rule9out, Connector.And));
    rules.Add(new Rule(rule10in, rule11out, Connector.And));
    rules.Add(new Rule(rule12in, rule12out, Connector.And));
    rules.Add(new Rule(rule13in, rule13out, Connector.And));
    rules.Add(new Rule(rule14in, rule14out, Connector.And));
    rules.Add(new Rule(rule15in, rule15out, Connector.And));
    rules.Add(new Rule(rule16in, rule16out, Connector.And));
    rules.Add(new Rule(rule17in, rule17out, Connector.And));
    rules.Add(new Rule(rule18in, rule18out, Connector.And));
    rules.Add(new Rule(rule19in, rule19out, Connector.And));
    rules.Add(new Rule(rule20in, rule20out, Connector.And));
    rules.Add(new Rule(rule21in, rule21out, Connector.And));
    rules.Add(new Rule(rule22in, rule22out, Connector.And));
    rules.Add(new Rule(rule23in, rule23out, Connector.And));
    rules.Add(new Rule(rule24in, rule24out, Connector.And));
    rules.Add(new Rule(rule25in, rule25out, Connector.And));

}
public void inferencia()
{
    engine = new InferEngine(conf, rules, fuzset);
```

```
            impli = engine.evaluateRules();
        }
        public String Deffuz()
        {
            double crisp1 = c.DeFuzzification(impli, MOTOR1);
            return crisp1.ToString();
        }
        public String salida(double In1, double In2)
        {

            setIn(In1, In2);
            Fuzz();
            reglas();
            inferencia();
            String fin = Deffuz();

            return fin;
        }
    }
}
```

Clase para Control de estabilidad ángulo THETA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FuzzyLogicController;
using FuzzyLogicController.FLC;
using FuzzyLogicController.MFs;
using FuzzyLogicController.RuleEngine;
using System.Windows.Forms;

namespace ModeloROV2
{
    class ControladorTHETA
    {
        double in1 = 0, in2 = 0;
        private Config conf = new Config(ImpMethod.Prod, ConnMethod.Min);
```

```
private LingVariable e = new LingVariable("e", VarType.Input);
private LingVariable de = new LingVariable("de", VarType.Input);
private LingVariable MOTOR1 = new LingVariable("MOTOR1", VarType.Output);

private FLC c;
private FuzzySet set1;
private FuzzySet set2;
private List<FuzzySet> fuzset;

List<RuleItem> rule1in = new List<RuleItem>();
List<RuleItem> rule1out = new List<RuleItem>();
List<RuleItem> rule2in = new List<RuleItem>();
List<RuleItem> rule2out = new List<RuleItem>();
List<RuleItem> rule3in = new List<RuleItem>();
List<RuleItem> rule3out = new List<RuleItem>();
List<RuleItem> rule4in = new List<RuleItem>();
List<RuleItem> rule4out = new List<RuleItem>();
List<RuleItem> rule5in = new List<RuleItem>();
List<RuleItem> rule5out = new List<RuleItem>();
List<RuleItem> rule6in = new List<RuleItem>();
List<RuleItem> rule6out = new List<RuleItem>();
List<RuleItem> rule7in = new List<RuleItem>();
List<RuleItem> rule7out = new List<RuleItem>();
List<RuleItem> rule8in = new List<RuleItem>();
List<RuleItem> rule8out = new List<RuleItem>();
List<RuleItem> rule9in = new List<RuleItem>();
List<RuleItem> rule9out = new List<RuleItem>();
List<RuleItem> rule10in = new List<RuleItem>();
List<RuleItem> rule10out = new List<RuleItem>();
List<RuleItem> rule11in = new List<RuleItem>();
List<RuleItem> rule11out = new List<RuleItem>();
List<RuleItem> rule12in = new List<RuleItem>();
List<RuleItem> rule12out = new List<RuleItem>();
List<RuleItem> rule13in = new List<RuleItem>();
List<RuleItem> rule13out = new List<RuleItem>();
List<RuleItem> rule14in = new List<RuleItem>();
List<RuleItem> rule14out = new List<RuleItem>();
List<RuleItem> rule15in = new List<RuleItem>();
```

```
List<RuleItem> rule15out = new List<RuleItem>();
List<RuleItem> rule16in = new List<RuleItem>();
List<RuleItem> rule16out = new List<RuleItem>();
List<RuleItem> rule17in = new List<RuleItem>();
List<RuleItem> rule17out = new List<RuleItem>();
List<RuleItem> rule18in = new List<RuleItem>();
List<RuleItem> rule18out = new List<RuleItem>();
List<RuleItem> rule19in = new List<RuleItem>();
List<RuleItem> rule19out = new List<RuleItem>();
List<RuleItem> rule20in = new List<RuleItem>();
List<RuleItem> rule20out = new List<RuleItem>();
List<RuleItem> rule21in = new List<RuleItem>();
List<RuleItem> rule21out = new List<RuleItem>();
List<RuleItem> rule22in = new List<RuleItem>();
List<RuleItem> rule22out = new List<RuleItem>();
List<RuleItem> rule23in = new List<RuleItem>();
List<RuleItem> rule23out = new List<RuleItem>();
List<RuleItem> rule24in = new List<RuleItem>();
List<RuleItem> rule24out = new List<RuleItem>();
List<RuleItem> rule25in = new List<RuleItem>();
List<RuleItem> rule25out = new List<RuleItem>();
List<Rule> rules;
List<FuzzySet> impli;
InferEngine engine;

public ControladorTHETA()
{
   double P1 = 180;
   double P2 = 30.3741;
   double P3 = 24.6872;
   double Q1 = 10;
   double Q2 = 2.9110;
   double Q3 = 0.0670;

   double M1 = 16.5;
   double M2 = 16.0643;
   double M3 = 6.2492;
   double M4 = 3.1806;
   double M5 = 2.4627;
```

```
    double M6 = 1.4568;



    //Declaracion de variable lingüistica Error "e"
    e.setRange(-P1, P1);
    e.addMF(new Trapmf("NN", -P1, -P1, -P2, -P3));
    e.addMF(new Trimf("N", -P2, -P3, 0));
    e.addMF(new Trimf("Z", -P3, 0, P3));
    e.addMF(new Trimf("P", 0, P3, P2));
    e.addMF(new Trapmf("PP", P3, P2, P1, P1));

    //Declaracion de variable lingüistica dError "de"

    de.setRange(-Q1, Q1);
    de.addMF(new Trapmf("NN", -Q1, -Q1, -Q2, -Q3));
    de.addMF(new Trimf("N", -Q2, -Q3, 0));
    de.addMF(new Trimf("Z", -Q3, 0, Q3));
    de.addMF(new Trimf("P", 0, Q3, Q2));
    de.addMF(new Trapmf("PP", Q3, Q2, Q1, Q1));

    //Declaracion de variable lingüistica MOTOR1 "MOTOR1"

    MOTOR1.setRange(-M1, M1);
    MOTOR1.addMF(new Trapmf("NNNN", -M1, -M1, -M2, -M3));
    MOTOR1.addMF(new Trimf("NNN", -M2, -M3, -M4));
    MOTOR1.addMF(new Trimf("NN", -M3, -M4, -M5));
    MOTOR1.addMF(new Trimf("N", -M4, -M5, -M6));
    MOTOR1.addMF(new Trimf("Z", -M6, 0, M6));
    MOTOR1.addMF(new Trimf("P", M6, M5, M4));
    MOTOR1.addMF(new Trimf("PP", M5, M4, M3));
    MOTOR1.addMF(new Trimf("PPP", M4, M3, M2));
    MOTOR1.addMF(new Trapmf("PPPP", M3, M2, M1, M1));



}
public void setIn(double a, double b)
{
    in1 = a;
    in2 = b;
```

```
}
public void Fuzz()
{
    c = new FLC(conf);

    set1 = new FuzzySet(c.Fuzzification(in1, e), e.Name);
    set2 = new FuzzySet(c.Fuzzification(in2, de), de.Name);
    fuzset = new List<FuzzySet>();
    fuzset.Add(set1);
    fuzset.Add(set2);

}
public void reglas()
{
    rule1in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "NN") });
    rule1out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule2in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "N") });
    rule2out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule3in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "Z") });
    rule3out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule4in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "P") });
    rule4out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
    rule5in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "PP") });
    rule5out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });

    rule6in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "NN") });
    rule6out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
    rule7in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "N") });
    rule7out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule8in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "Z") });
    rule8out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule9in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "P") });
    rule9out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule10in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "PP") });
    rule10out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });

    rule11in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "NN") });
    rule11out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule12in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "N") });
```

```
rule12out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule13in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "Z") });
rule13out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule14in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "P") });
rule14out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule15in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "PP") });
rule15out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });

rule16in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "NN") });
rule16out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule17in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "N") });
rule17out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
rule18in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "Z") });
rule18out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule19in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "P") });
rule19out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule20in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "PP") });
rule20out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });

rule21in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "NN") });
rule21out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
rule22in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "N") });
rule22out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });
rule23in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "Z") });
rule23out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });
rule24in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "P") });
rule24out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });
rule25in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "PP") });
rule25out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });

rules = new List<Rule>();
rules.Add(new Rule(rule1in, rule1out, Connector.And));
rules.Add(new Rule(rule2in, rule2out, Connector.And));
rules.Add(new Rule(rule3in, rule3out, Connector.And));
rules.Add(new Rule(rule4in, rule4out, Connector.And));
rules.Add(new Rule(rule5in, rule5out, Connector.And));
rules.Add(new Rule(rule6in, rule6out, Connector.And));
rules.Add(new Rule(rule7in, rule7out, Connector.And));
rules.Add(new Rule(rule8in, rule8out, Connector.And));
```

```
    rules.Add(new Rule(rule9in, rule9out, Connector.And));
    rules.Add(new Rule(rule10in, rule11out, Connector.And));
    rules.Add(new Rule(rule12in, rule12out, Connector.And));
    rules.Add(new Rule(rule13in, rule13out, Connector.And));
    rules.Add(new Rule(rule14in, rule14out, Connector.And));
    rules.Add(new Rule(rule15in, rule15out, Connector.And));
    rules.Add(new Rule(rule16in, rule16out, Connector.And));
    rules.Add(new Rule(rule17in, rule17out, Connector.And));
    rules.Add(new Rule(rule18in, rule18out, Connector.And));
    rules.Add(new Rule(rule19in, rule19out, Connector.And));
    rules.Add(new Rule(rule20in, rule20out, Connector.And));
    rules.Add(new Rule(rule21in, rule21out, Connector.And));
    rules.Add(new Rule(rule22in, rule22out, Connector.And));
    rules.Add(new Rule(rule23in, rule23out, Connector.And));
    rules.Add(new Rule(rule24in, rule24out, Connector.And));
    rules.Add(new Rule(rule25in, rule25out, Connector.And));

}
public void inferencia()
{
    engine = new InferEngine(conf, rules, fuzset);
    impli = engine.evaluateRules();
}
public String Deffuz()
{
    double crisp1 = c.DeFuzzification(impli, MOTOR1);
    return crisp1.ToString();
}
public String salida(double In1, double In2)
{




    setIn(In1, In2);
    Fuzz();
    reglas();
    inferencia();
    String fin = Deffuz();
```

```
            return fin;
        }
    }
}
```

Clase para Control de estabilidad ángulo PHI

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FuzzyLogicController;
using FuzzyLogicController.FLC;
using FuzzyLogicController.MFs;
using FuzzyLogicController.RuleEngine;
using System.Windows.Forms;

namespace ModeloROV2
{
    class ControladorPHI
    {
        double in1 = 0, in2 = 0;
        private Config conf = new Config(ImpMethod.Prod, ConnMethod.Min);

        private LingVariable e = new LingVariable("e", VarType.Input);
        private LingVariable de = new LingVariable("de", VarType.Input);
        private LingVariable MOTOR1 = new LingVariable("MOTOR1", VarType.Output);

        private FLC c;
        private FuzzySet set1;
        private FuzzySet set2;
        private List<FuzzySet> fuzset;

        List<RuleItem> rule1in = new List<RuleItem>();
        List<RuleItem> rule1out = new List<RuleItem>();
        List<RuleItem> rule2in = new List<RuleItem>();
```

```
List<RuleItem> rule2out = new List<RuleItem>();
List<RuleItem> rule3in = new List<RuleItem>();
List<RuleItem> rule3out = new List<RuleItem>();
List<RuleItem> rule4in = new List<RuleItem>();
List<RuleItem> rule4out = new List<RuleItem>();
List<RuleItem> rule5in = new List<RuleItem>();
List<RuleItem> rule5out = new List<RuleItem>();
List<RuleItem> rule6in = new List<RuleItem>();
List<RuleItem> rule6out = new List<RuleItem>();
List<RuleItem> rule7in = new List<RuleItem>();
List<RuleItem> rule7out = new List<RuleItem>();
List<RuleItem> rule8in = new List<RuleItem>();
List<RuleItem> rule8out = new List<RuleItem>();
List<RuleItem> rule9in = new List<RuleItem>();
List<RuleItem> rule9out = new List<RuleItem>();
List<RuleItem> rule10in = new List<RuleItem>();
List<RuleItem> rule10out = new List<RuleItem>();
List<RuleItem> rule11in = new List<RuleItem>();
List<RuleItem> rule11out = new List<RuleItem>();
List<RuleItem> rule12in = new List<RuleItem>();
List<RuleItem> rule12out = new List<RuleItem>();
List<RuleItem> rule13in = new List<RuleItem>();
List<RuleItem> rule13out = new List<RuleItem>();
List<RuleItem> rule14in = new List<RuleItem>();
List<RuleItem> rule14out = new List<RuleItem>();
List<RuleItem> rule15in = new List<RuleItem>();
List<RuleItem> rule15out = new List<RuleItem>();
List<RuleItem> rule16in = new List<RuleItem>();
List<RuleItem> rule16out = new List<RuleItem>();
List<RuleItem> rule17in = new List<RuleItem>();
List<RuleItem> rule17out = new List<RuleItem>();
List<RuleItem> rule18in = new List<RuleItem>();
List<RuleItem> rule18out = new List<RuleItem>();
List<RuleItem> rule19in = new List<RuleItem>();
List<RuleItem> rule19out = new List<RuleItem>();
List<RuleItem> rule20in = new List<RuleItem>();
List<RuleItem> rule20out = new List<RuleItem>();
List<RuleItem> rule21in = new List<RuleItem>();
List<RuleItem> rule21out = new List<RuleItem>();
```

```
List<RuleItem> rule22in = new List<RuleItem>();
List<RuleItem> rule22out = new List<RuleItem>();
List<RuleItem> rule23in = new List<RuleItem>();
List<RuleItem> rule23out = new List<RuleItem>();
List<RuleItem> rule24in = new List<RuleItem>();
List<RuleItem> rule24out = new List<RuleItem>();
List<RuleItem> rule25in = new List<RuleItem>();
List<RuleItem> rule25out = new List<RuleItem>();
List<Rule> rules;
List<FuzzySet> impli;
InferEngine engine;

public ControladorTHETA()
{
   double P1 = 180;
   double P2 = 30.3741;
   double P3 = 24.6872;
   double Q1 = 10;
   double Q2 = 2.9110;
   double Q3 = 0.0670;

   double M1 = 16.5;
   double M2 = 16.0643;
   double M3 = 6.2492;
   double M4 = 3.1806;
   double M5 = 2.4627;
   double M6 = 1.4568;


   //Declaracion de variable lingüistica Error "e"
   e.setRange(-P1, P1);
   e.addMF(new Trapmf("NN", -P1, -P1, -P2, -P3));
   e.addMF(new Trimf("N", -P2, -P3, 0));
   e.addMF(new Trimf("Z", -P3, 0, P3));
   e.addMF(new Trimf("P", 0, P3, P2));
   e.addMF(new Trapmf("PP", P3, P2, P1, P1));

   //Declaracion de variable lingüistica dError "de"
```

```
    de.setRange(-Q1, Q1);
    de.addMF(new Trapmf("NN", -Q1, -Q1, -Q2, -Q3));
    de.addMF(new Trimf("N", -Q2, -Q3, 0));
    de.addMF(new Trimf("Z", -Q3, 0, Q3));
    de.addMF(new Trimf("P", 0, Q3, Q2));
    de.addMF(new Trapmf("PP", Q3, Q2, Q1, Q1));

    //Declaracion de variable lingüistica MOTOR1 "MOTOR1"

    MOTOR1.setRange(-M1, M1);
    MOTOR1.addMF(new Trapmf("NNNN", -M1, -M1, -M2, -M3));
    MOTOR1.addMF(new Trimf("NNN", -M2, -M3, -M4));
    MOTOR1.addMF(new Trimf("NN", -M3, -M4, -M5));
    MOTOR1.addMF(new Trimf("N", -M4, -M5, -M6));
    MOTOR1.addMF(new Trimf("Z", -M6, 0, M6));
    MOTOR1.addMF(new Trimf("P", M6, M5, M4));
    MOTOR1.addMF(new Trimf("PP", M5, M4, M3));
    MOTOR1.addMF(new Trimf("PPP", M4, M3, M2));
    MOTOR1.addMF(new Trapmf("PPPP", M3, M2, M1, M1));



}
public void setIn(double a, double b)
{
    in1 = a;
    in2 = b;
}
public void Fuzz()
{
    c = new FLC(conf);

    set1 = new FuzzySet(c.Fuzzification(in1, e), e.Name);
    set2 = new FuzzySet(c.Fuzzification(in2, de), de.Name);
    fuzset = new List<FuzzySet>();
    fuzset.Add(set1);
    fuzset.Add(set2);

}
public void reglas()
```

```
{
    rule1in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "NN") });
    rule1out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule2in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "N") });
    rule2out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule3in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "Z") });
    rule3out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPPP") });
    rule4in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "P") });
    rule4out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PPP") });
    rule5in.AddRange(new RuleItem[2] { new RuleItem("e", "NN"), new RuleItem("de", "PP") });
    rule5out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });

    rule6in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "NN") });
    rule6out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "PP") });
    rule7in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "N") });
    rule7out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule8in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "Z") });
    rule8out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule9in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "P") });
    rule9out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule10in.AddRange(new RuleItem[2] { new RuleItem("e", "N"), new RuleItem("de", "PP") });
    rule10out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });

    rule11in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "NN") });
    rule11out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "P") });
    rule12in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "N") });
    rule12out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule13in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "Z") });
    rule13out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule14in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "P") });
    rule14out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule15in.AddRange(new RuleItem[2] { new RuleItem("e", "Z"), new RuleItem("de", "PP") });
    rule15out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });

    rule16in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "NN") });
    rule16out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule17in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "N") });
    rule17out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "Z") });
    rule18in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "Z") });
```

```
rule18out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule19in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "P") });
rule19out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "N") });
rule20in.AddRange(new RuleItem[2] { new RuleItem("e", "P"), new RuleItem("de", "PP") });
rule20out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });

rule21in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "NN") });
rule21out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NN") });
rule22in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "N") });
rule22out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNN") });
rule23in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "Z") });
rule23out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });
rule24in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "P") });
rule24out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });
rule25in.AddRange(new RuleItem[2] { new RuleItem("e", "PP"), new RuleItem("de", "PP") });
rule25out.AddRange(new RuleItem[1] { new RuleItem("MOTOR1", "NNNN") });

rules = new List<Rule>();
rules.Add(new Rule(rule1in, rule1out, Connector.And));
rules.Add(new Rule(rule2in, rule2out, Connector.And));
rules.Add(new Rule(rule3in, rule3out, Connector.And));
rules.Add(new Rule(rule4in, rule4out, Connector.And));
rules.Add(new Rule(rule5in, rule5out, Connector.And));
rules.Add(new Rule(rule6in, rule6out, Connector.And));
rules.Add(new Rule(rule7in, rule7out, Connector.And));
rules.Add(new Rule(rule8in, rule8out, Connector.And));
rules.Add(new Rule(rule9in, rule9out, Connector.And));
rules.Add(new Rule(rule10in, rule11out, Connector.And));
rules.Add(new Rule(rule12in, rule12out, Connector.And));
rules.Add(new Rule(rule13in, rule13out, Connector.And));
rules.Add(new Rule(rule14in, rule14out, Connector.And));
rules.Add(new Rule(rule15in, rule15out, Connector.And));
rules.Add(new Rule(rule16in, rule16out, Connector.And));
rules.Add(new Rule(rule17in, rule17out, Connector.And));
rules.Add(new Rule(rule18in, rule18out, Connector.And));
rules.Add(new Rule(rule19in, rule19out, Connector.And));
rules.Add(new Rule(rule20in, rule20out, Connector.And));
rules.Add(new Rule(rule21in, rule21out, Connector.And));
rules.Add(new Rule(rule22in, rule22out, Connector.And));
```

```
            rules.Add(new Rule(rule23in, rule23out, Connector.And));
            rules.Add(new Rule(rule24in, rule24out, Connector.And));
            rules.Add(new Rule(rule25in, rule25out, Connector.And));


        }
        public void inferencia()
        {
            engine = new InferEngine(conf, rules, fuzset);
            impli = engine.evaluateRules();
        }
        public String Deffuz()
        {
            double crisp1 = c.DeFuzzification(impli, MOTOR1);
            return crisp1.ToString();
        }
        public String salida(double In1, double In2)
        {




            setIn(In1, In2);
            Fuzz();
            reglas();
            inferencia();
            String fin = Deffuz();

            return fin;
        }
    }
}
```

Aplicación de CUBO

Para finalizar este capítulo se implementa una aplicación en C# que integra las clases que definen los sistemas de control del vehículo, el modelo de CUBO y los sistemas de comunicación.

Se crea una clase que se encarga de controlar la comunicación UDP

Codigo Fuente 22 - Clase de Comunicación

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ModeloROV2
{
    class Comunicacion
    {
        public event EventHandler Recibido;

        String Texto;
        UdpClient udpClient;
        UdpClient rxudpClient;
        IPEndPoint RemoteIPEndPoint;

         public Comunicacion(String IPRemota, String PuertoRemoto, String
PuertoLocal)
        {
            udpClient = new UdpClient();
            udpClient.Connect(IPRemota, Convert.ToInt32(PuertoRemoto));
//Para enviar
            rxudpClient = new UdpClient(Convert.ToInt32(PuertoLocal));
            RemoteIPEndPoint = new IPEndPoint(IPAddress.Any, 0);
        }

        public void enviar(String txt)
        {
            byte[] txtSnd = new byte[0];
            txtSnd = System.Text.Encoding.ASCII.GetBytes(txt);
            udpClient.Send(txtSnd, txtSnd.Length);
        }

        public void enviarDatos(String X, String Y, String Z, String PSI,
String THETA, String PHI, String Vx, String Vy, String Vz)
        {
            byte[] txtSnd = new byte[0];
            txtSnd = System.Text.Encoding.ASCII.GetBytes("C"+Z);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("J"+PSI);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("K"+THETA);
```

```
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("L"+PHI);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("M" + X);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("N" + Y);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("O" + Z);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("P" + Vx);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("Q" + Vy);
            udpClient.Send(txtSnd, txtSnd.Length);

            txtSnd = System.Text.Encoding.ASCII.GetBytes("R" + Vz);
            udpClient.Send(txtSnd, txtSnd.Length);

        }

        public void ReceiveMessages()
        {
            Byte[] receiveBytes = rxudpClient.Receive(ref RemoteIPEndPoint);
            BitArray BitDet = new BitArray(receiveBytes);
            String strReturnData =
System.Text.Encoding.Unicode.GetString(receiveBytes);
            Texto +=("RecibidoUDP --> " +
System.Text.Encoding.ASCII.GetString(receiveBytes));
            NewInitialize();
            OnRx(new EventArgs());

        }

        private void NewInitialize()
        {
            Thread ThreadReceive = new
System.Threading.Thread(ReceiveMessages);
            ThreadReceive.Start();
        }


    }
}
```

Se implementa una simulación similar a las realizadas en los ensayos anteriores transmitiendo toda la información a la Interfaz Gráfica vía UDP y se envía video a través de RSTP con Raspberry Pi.

En la ilustración 81 se muestra una captura de una simulación en la que se posiciona CUBO en una profundidad de 3 metros, con un tiempo de simulación de 30 segundos y un paso de 0.01 segundos.

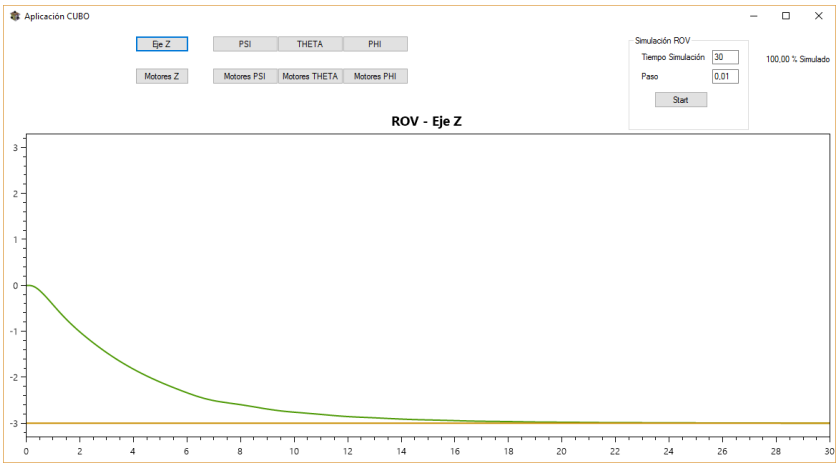


Ilustración 81 - Aplicación CUBO

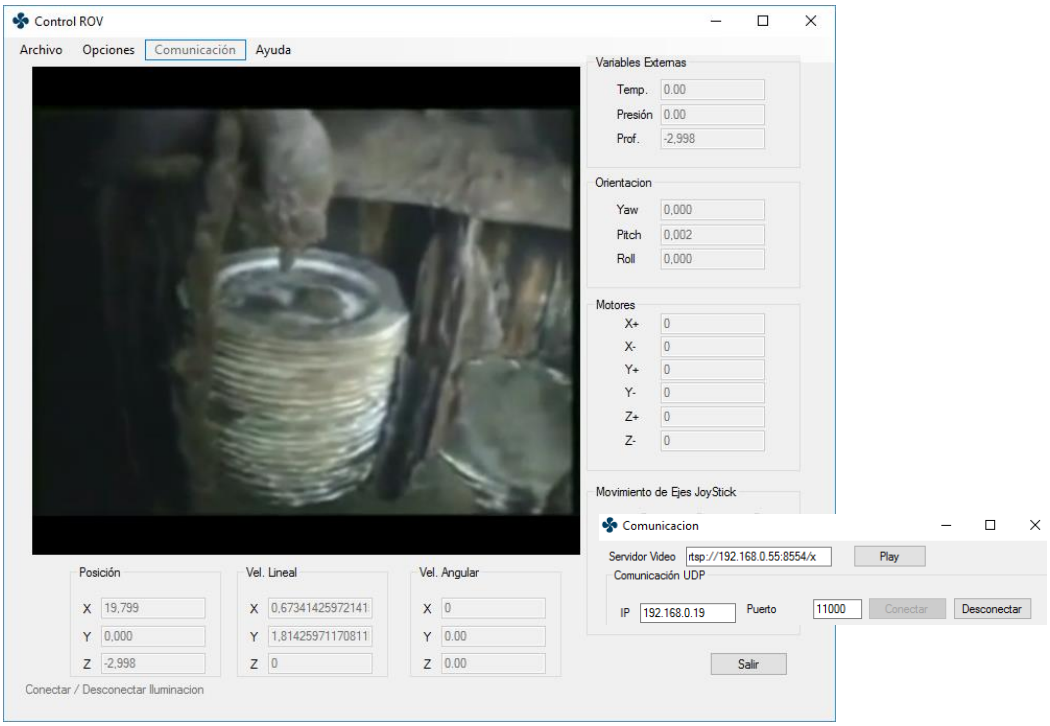


Ilustración 82 - Interfaz Gráfica recibiendo datos y video

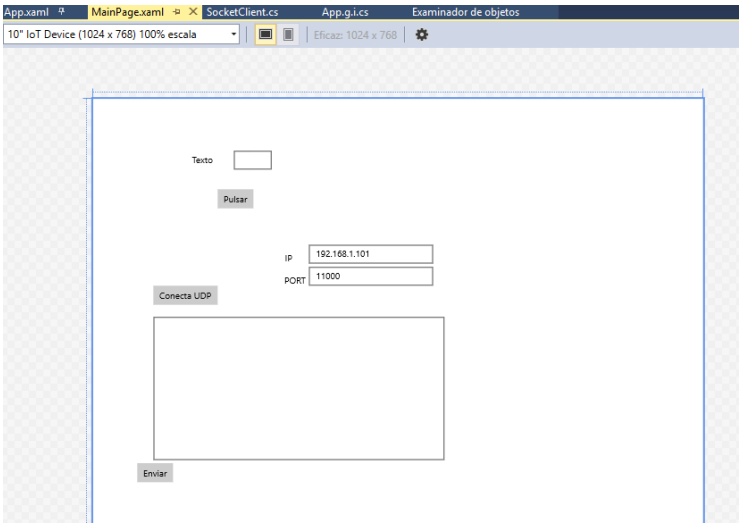
Capítulo 9
Conclusiones y trabajos futuros

En este proyecto la tarea principal que se ha llevado a cabo es el diseño de un sistema de control basado en lógica difusa del sistema de estabilización de profundidad de un vehículo submarino tripulado de forma remota. El diseño del sistema de control se ha realizado bajo entorno Matlab y la implementación se ha realizado en lenguaje C#.

El sistema de control implementado en C#, no ha podido ser verificado en el dispositivo real, por ello, se verificó a través de un modelo de CUBO que fue implementado en el mismo lenguaje de programación cuyos resultados obtenidos fueron satisfactorios.

El sistema de control es una parte de un software que se ha desarrollado el cual es el encargado de recoger información de los sensores a través de una comunicación serie a una placa de desarrollo Arduino. A su vez, este software se comunica mediante conexión UDP con la interfaz gráfica de la cual recoge las órdenes del joystick, y envía los datos de video de la Pi Camera.

A la hora de desarrollar el software implementado en la Raspberry Pi, se estudiaron varias opciones, entre ellas la de programar en sistema operativo Raspbian o con Windows 10 IOT. Se iniciaron paralelamente el desarrollo en los dos entornos aunque se optó por desarrollar bajo Raspbian. Como trabajo futuro cabría la opción de desarrollar este software bajo Windows 10 IOT



Como trabajo futuro principal, destacaría la de instalar todo el sistema electrónico en el interior del dispositivo real y realizar los ajustes y modificaciones pertinentes para su funcionamiento.

Otro trabajo futuro interesante en el que se ha trabajado en la realización de este proyecto pero no constaba como objetivo del mismo es la realización de placa de circuito impreso a modo de “shield” para conectar con Raspberry Pi el cual integre los sensores y un microcontrolador.

Bibliografía

Galisteo Streeksoff , H. (s.f.). *Propuesta de nuevos dispositivos, mejoras de circuitos y sistemas implementados en ROVs para trabajos de búsqueda y rescate.* Universidad Politécnica de Catalunya.

Aguirre Martínez, J., Munuera Saura, G., & Sánchez Durán, F. (2017). *Control de estabilidad de un vehiculo submarino operado de forma remota.* Departamento de Ingeniería Mecánica - Universidad Polotécnica de Cartagena: 13ºCongreso Iberoamericano de Ingeniería Mecánica.

Calvo, J. d. (2014). *Modelado, Simulación y Control de un Vehículo Submarino Manipulado de forma Remota (ROV) .* Universidad Politécnica de Cartagena.

Cordón, O., Herrera, F., Hoffman, F., & magdalena, L. (2001). *Genetic Fuzzy System - Evolutionary Tuning and learning of fuzzy knowledge bases.* World Scientific.

J. Garrigós Guerrero, J. Martínez Alajarín, A. Ros Vidal, & R. Ruiz Merino . (s.f.). *Optimización mediante algoritmos genéticos de un sistema de compresión de fonocardiogramas basado en PDA.* Departamento de Electrónica, Tecnología de Computadoras y Proyectos, Universidad Politécnica de Cartagena.

Kadmiry, B. (2002). *Fuzzy Control for an Unmanned Helicopter.* Department of Computer and Information Science - Linköpings Universitet.

Mahtani Mirchandani, A. (2012). *Diseño e implementación de una arquitectura de control de un ROV.* Escuela de Ingeniería de Informática. Universidad de Las Palmas de G.C.

Martínez, E. Y. (2016). *Diseño del sistema de control de un vehículo submarino manipulado de forma remota.* Universidad Politécnica de Cartagena.

Massimo Banzi, & Michael Shioh. (s.f.). *Introducción a Arduino.* Anaya.

Mathworks. (s.f.). *Fuzzy Logic Designer.* Obtenido de https://es.mathworks.com/help/fuzzy/fuzzylogicdesigner-app.html

Mathworks. (s.f.). *Fuzzy Logic Toolbox.* Obtenido de https://es.mathworks.com/help/fuzzy/index.html

Mathworks. (s.f.). *Genetic Algorithm.* Obtenido de https://es.mathworks.com/help/gads/genetic-algorithm.html

Mathworks. (s.f.). *Global Optimization Toolbox.* Obtenido de https://es.mathworks.com/help/gads/index.html

Mathworks. (s.f.). *Multiobjective Optimization.* Obtenido de https://es.mathworks.com/help/gads/multiobjective-optimization.html

Monk, S. (s.f.). *Raspberry Pi 200 Ejercicios prácticos.* Anaya.

Pacheco, M. A. (s.f.). Algoritmo Genéticos - Principios y aplicaciones. *Departamento de Ingeniería Electrica - Universidad Católica de Rio de Janeiro*.

Pi, R. (s.f.). *RaspiCam Documentation - Raspberry Pi.* Obtenido de https://www.raspberrypi.org/app/uploads/2013/07/RaspiCam-Documentation.pdf

Seck Tuoh Mora, J., Medina Marín, J., & Hernández Romero, N. (2016). *Introducción a los Algoritmos Genéticos.* Area Académica de Ingeniería-ICBI-UAEH.

Upton, E., & Halfacree, G. (s.f.). *Raspberry Pi Guia de Usuario.* Anaya.