Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0408

Arquitectura para Teleoperación Inalámbrica con Realimentación Visual de ROVs basados en ArduSub

Diego Centelles[†], Antonio Soriano[‡] Raúl Marín[†] and Pedro J. Sanz[†]
[†] Departamento de Ingeniería y Ciencia de los Computadores. Universitat Jaume I
Av. Sos Baynat, s/n, 12071 Castellón de la Plana (España)
[‡] Departament d'Informàtica. Universitat de València. Av. de la Universitat, s/n. 46100 Burjassot (España)
centelld@uji.es, antonio.soriano-asensi@uv.es, rmarin@uji.es, sanzp@uji.es

Resumen

En este trabajo se presenta la adaptación de la arquitectura de un BlueROV para habilitar la teleoperación inalámbrica sobre un canal que, por naturaleza, dispondrá de un ancho de banda muy limitado. Para ello se propone la aplicación de un protocolo cross-layer fuera de la pila TCP/IP. El sistema de teleoperación integra el algoritmo de compresión de imagen progresivo DEBT (Depth Embedded Block Tree) para habilitar la realimentación visual y utiliza el simulador UWSim como extensión de realidad virtual del HRI (Human-Robot-Interface).

Palabras clave: Robótica Submarina, Comunicación Inlámbrica, Sonar, Radio Frecuencia, Protocolos Red, Compresión de imagen, Control Supervisado

1 Introducción

La realización de actividades en entornos submarinos a cierta profundidad suele conllevar un riesgo y unos costes considerables. Cada vez es más frecuente el uso de robots para la realización de tareas relacionadas con la arqueología y ciencias marinas, así como en el mantenimiento de plataformas petrolíferas o en la indústria del gas, porque permite reducir el riesgo y los costes. Las tareas desempeñadas por robots en entornos subacuáticos son cada vez más complejas. La realización de algunas de ellas requiere del trabajo cooperativo de varios robots. Es necesario que todos los robots que colaboran en una misma tarea tengan la capacidad de comunicarse, sin necesidad de cables que puedan restringir aún más su capacidad de operación. Es por este motivo que en los últimos años ha habido un interés creciente en el desarrollo de comunicaciones inalámbricas en entornos submarinos. Este tipo de comunicaciones es especialmente importante en tareas donde se necesitan equipos de robots y sensores realizando tareas de forma cooperativa.

La solución más habitual en comunicación inalámbrica submarina se basa en señales acústicas, que permiten conectar dispositivos separados a varios kilómetros de distancia. El inconveniente de esta comunicación es su sensibilidad al ruido acústico, lo que restringe el rango de frecuencias acústicas a unos pocos kHz. El bajo ancho de banda y el elevado retardo (≈ 0.6 ms/m) dificulta la implementación de enlaces full-duplex con esta tecnología. Además, la comunicación acústica es sensible a los problemas de camino-múltiple en las proximidades de objetos sólidos. A pesar de todo ello se ha podido demostrar la transmisión de video en un entorno submarino empleando un canal acústico[6].

En los casos en que no es viable el uso de señales acústicas también es posible la comunicación mediante VLC (*Visible Light Communication*)[2]. La técnica VLC proporciona un mayor ancho de banda que las señales acústicas, pero su alcance es limitado, del órden de metros, debido a la atenuación de la luz visible en el agua. Es una buena solución en aguas claras pero su rendimiento se ve muy afectado en aguas turbias o en función de las condiciones de iluminación. Otra alternativa es la comunicación basada en radiofrecuencia (RF) [1]. La comunicación RF no se ve afectada por la turbidez del agua ni por las variaciones de iluminación. Sin embargo, la elevada conductividad del agua de mar limita el alcance de la comunicación submarina a unos pocos metros. Así, en aquellas situaciones en que no se requieran grandes distancias cabe la posibilidad de emplear la comunicación basada en RF o en VLC, que proporciona mejores tasas de transferencia que la comunicación basada en señales acústicas.

El presente trabajo se enmarca dentro de los experimentos del proyecto Nacional Coodinado MERBOTS (`http://www.irs.uji.es/merbots/`), en el cual se pretende avanzar en el contexto de la robótica de intervención submarina, como herramienta que facilite el trabajo de estudio arqueológico a medias y altas profundidades. En este proyecto se pretende, como demostración de las técnicas desarrolladas, experimentar la recuperación de objetos del fondo del mar con el uso de dos robots, uno de intervención y otro de apoyo visual, pudiendo tener canales inalámbricos (Sónar/RF/VLC) en

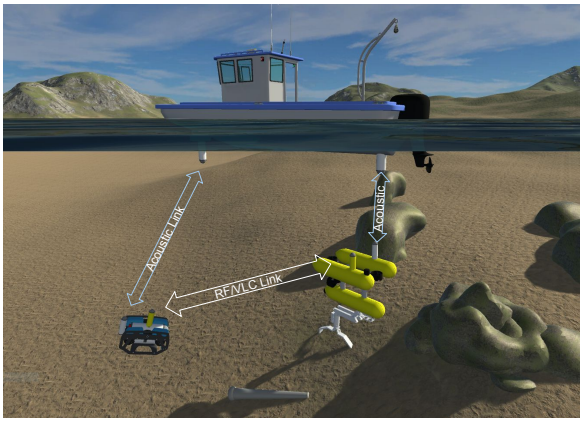


Figura 1: Visión conceptual del proyecto MERBOTS

función de la necesidad (ver Fig. 1).

Además de considerar el medio de comunicación, es importante el protocolo de transporte empleado. Las aplicaciones de telerobótica requieren de sistemas de comunicaciones en los que el retardo y su fluctuación (i.e. jitter) sean reducidos. Es por ese motivo que requieren de protocolos de transporte dedicados, como puede ser el *Bilateral Transport Protocol* (BTP) [8], o el *Trinomial* [3]. En el campo de la manipulación de drones es muy utilizado el protocolo *Micro Air Vehicle Communication Protocol* (MAVLink). Es un protocolo muy ligero especialmente dedicado para aplicaciones en las que el ancho de banda disponible es muy limitado. En este trabajo se presenta una arquitectura de teleoperación wireless de un ROV (*Remotely Operated Vehicle*) basado en el software ArduSub (`www.ardusub.com`), que es una adaptación para drones subacuáticos del proyecto ArduPilot (`ardupilot.org`).

En el presente trabajo se presenta la arquitectura de un sistema de teleoperación inalámbrica en entornos submarinos. En la sección 2 se describe el hardware y software utilizado para la realización de la experimentación. En la sección 3 se describen los diferentes componentes software y el protocolo que componen la arquitectura *wireless* desarrollada. Seguidamente, en la sección 4 se explican los experimentos realizados para optimizar el protocolo, y evaluar su rendimiento. Finalmente, en la sección 5 se resumen los principales resultados y conclusiones que se extraen de este trabajo.

2 La Plataforma de Experimentación

El equipo de experimentación ha consistido en un ROV basado en la versión 1 de BlueROV, los

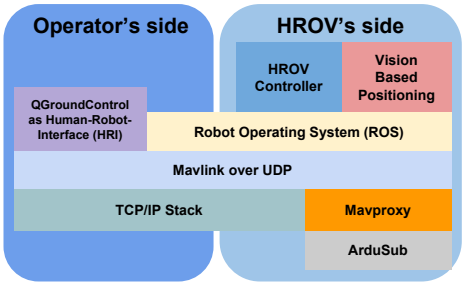


Figura 2: Arquitectura básica del sistema de teleoperación de un BlueROV a través de umbilical

módems de RF S100 de la empresa WFS (*Wireless For Subsea*) y el UWSim [5] como interfaz de VR (*Virtual Reality*) y herramienta para la simulación del protocolo de comunicación desarrollado sobre un canal de comunicación con errores.

El BlueROV utiliza una Pixhawk [4] como placa donde se ejecuta el software de ArduSub. En la arquitectura estándar (con umbilical), la teleoperación del BlueROV se realiza utilizando el protocolo MAVLink, ver Fig. 2. Los mensajes de MAVLink se encapsulan en datagramas que son enviados a una RaspberryPi 3B que se encuentra en el ROV. Una vez en el ordenador de abordo (Raspberry Pi), un programa llamado MAVProxy desencapsula los mensajes MAVLink de los paquetes UDP y los reenvía a través de un puerto serie donde se encuentra conectada la Pixhawk. La Pixhawk también envía al MAVProxy mensajes MAVLink que representan diferentes estados del robot o respuestas a comandos de alto nivel del operador, los cuales se encapsulan en datagramas que son enviados al operador a través de las restantes capas de la pila TCP/IP. Algunos de los mensajes de estado del robot pueden ser el nivel de batería, información de la IMU o la posición geodésica del ROV. Esta última está filtrada por un EKF (*Extended Kalman Filter*) que combina los datos de la IMU (*Intertial Measurement Unit*), la brújula, sensor de profundidad y las actualizaciones de posición GPS (*Global Positioning System*), las cuales pueden provenir tanto de mensajes MAVLink como de un GPS directamente conectado a la placa. Existe un paquete de ROS (*Robot Operating System*) llamado mavros que se comunica con el MAVProxy con el fin de ofrecer una interfaz de control del ROV a través de *topics* y servicios de ROS. Este nodo en ROS también puede actuar como un proxy de mensajes MAVLink. Por otro lado, se suele utilizar QGroundControl como GUI (*Graphical User Interface*), la cual envía y recibe mensajes MAVLink en datagramas hacia un proxy, que puede ser el el propio mavros o el MAVProxy. La Fig. 2 muestra una aproximación simple de la arquitectura de un

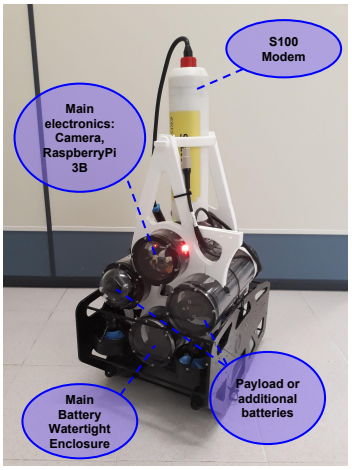


Figura 3: Integración del módem S100 en el BlueROV v1

BlueROV.

Para conseguir una comunicación sin umbilical se ha integrado un módem S100, de la empresa WFS (*Wireless For Subsea*) en el BlueROV (ver Fig. 3). Estos módems habilitan un canal half-duplex de comunicación por RF con una velocidad de transmisión de hasta 2,4 kbps. Estos dispositivos transmiten los bytes que el programador envía a través del puerto serie encapsulándolos en pequeños paquetes con un *overhead* determinado. También permiten elegir entre usar una codificación Manchester o normal (de esta última, no se específica el tipo). La primera protege mejor contra errores de transmisión aunque consume mucho más ancho de banda. También permiten configurar otros parámetros como la activación de reenvío de paquetes ante pérdidas, lo cual requiere el envío de paquetes de reconocimiento automáticos por parte del módem (ACK o *Acknowledgment*) aumentando, en consecuencia, el retardo de los datos.

Se ha utilizado un módulo de UWSim para simular la red (ver `https://github.com/dcentelles/underwater_simulation`) con el fin de poder experimentar el protocolo en diferentes condiciones del canal y cuando no se dispone de los módems reales. Aunque todavía está en una fase temprana de desarrollo, este módulo ya permite simular dispositivos de comunicación mediante un modelo estadístico simple de los mismos. Durante la ejecución de un escenario de UWSim se simulan tiempos de transmisión, propagación y errores en los paquetes. Los errores pueden ser debidos a la atenuación de la señal, una colisión, o por no haber espacio suficiente en el buffer de transmisión. Este módulo para UWSim realiza llamadas al planificador de eventos en tiempo real y a otras utilidades de la biblioteca de NS3 (*Network Simulator 3*) para realizar la simulación de la red. Realizar

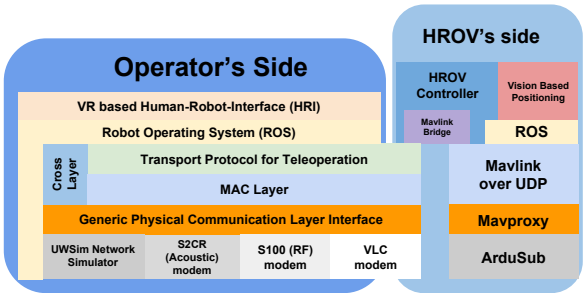


Figura 4: Arquitectura *Wireless*

simulaciones de la red ha sido fácilmente abordable gracias a una interfaz de abstracción de la capa física para el envío y recepción de paquetes. Esta capa de abstracción se ha implementado mediante IPC (*Inter-Process-Communication*) y se encuentra representada en la Fig. 4 como *Generic Physical Communication Layer Interface* o GPCLI).

3 Arquitectura Software del Sistema de Teleoperación Inalámbrica Submarina

Debido al ancho de banda tan limitado que ofrecen los módems S100 sería impossible retransmitir todos los mensajes de MAVLink a través del canal de RF. Por esta razón se ha implementado un protocolo minimalista que no depende de la pila TCP/IP para la comunicación *wireless*. Un programa en la Raspberry Pi del ROV, mostrado en la Fig. 4 como HROV Controller, interpreta los mensajes de este protocolo para luego intercomunicarse con la Pixhawk mediante MAVLink, a través del MAVProxy.

La misión del protocolo desarrollado es hacer llegar las órdenes del operador al robot y el estado del mismo al operador en forma de odometría e imagen. Sobre este protocolo se ha implementado un HRI combinando el UWSim y una interfaz de controles dedicada para la teleoperación. El sistema completo habilita al operador el control del robot tanto por velocidades como por comandos de posición usando coordenadas NED (North-East-Down). El control por velocidades se realiza mediante un joystick conectado al computador del operador. Para el control por posición, el operador mueve un marcador interactivo en UWSim representado por un BlueROV semi-transparente hacia la posición deseada. Una vez obtenida, el usuario envía una orden de posición a través de la interfaz de controles. La interfaz notifica al usuario cuando el robot ha recibido la orden y cuando la ha completado, siendo capaz de cancelar dicha orden (ver Fig. 5). Mediante órdenes de alto nivel, el oper-

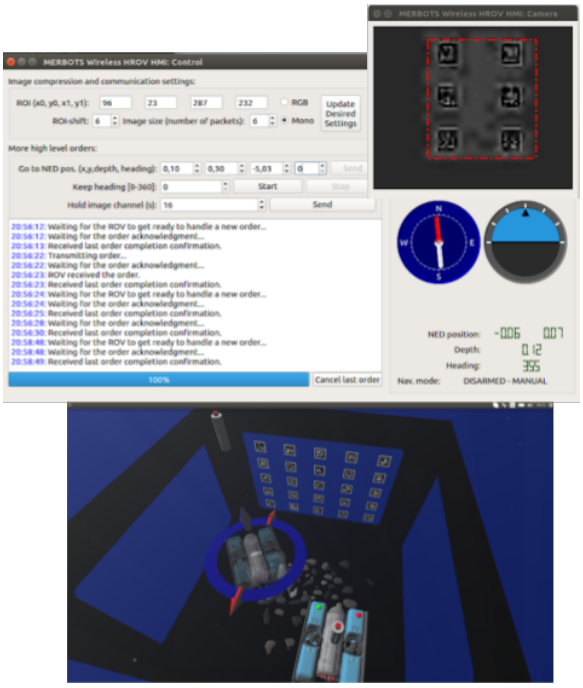


Figura 5: Interfaz de Realidad Virtual con UWSim, controles de operador y feedback visual con ROI (*Region Of In*).

ador actualiza los parámetros de compresión. Uno de estos parametros es el tamaño fijo de la imagen en términos de número de paquetes necesario para transmitir la imagen en su totalidad. Otro es una región de interés (ROI) en la imagen donde se desea mayor calidad sin aumentar el tamaño de la misma. Esta región de interés la selecciona el operador dibujando una forma rectangular sobre la imagen.

Para maximizar la tasa de envío y minimizar los retardos es necesario que el protocolo de transporte tenga conocimiento del estado del canal de comunicación y no delegar el control de acceso al medio a una capa inferior. Teniendo en cuenta la naturaleza half-duplex del canal inalámbrico, el envío de una orden al robot debe realizarse cuando haya certeza de que dicho paquete no va a almacenarse en un buffer de la capa inferior para esperar a que el canal esté libre para realizar la transmisión. Por ello, el protocolo desarrollado enviará una petición de envio de paquete directamente a la capa física cuando exista la seguridad de que el canal está libre y que, por lo tanto, no se producirá una colisión o un apilamiento del paquete. El protocolo propuesto se basa en un sistema maestro-esclavo, donde el operador es el maestro y el robot el esclavo. El operador (maestro) envía paquetes al ROV (esclavo) a una cadencia constante. Estos paquetes transportan el estado deseado del robot y órdenes de alto nivel. El tiempo de envío entre un paquete y el siguiente se denomina IPG

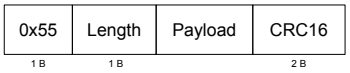


Figura 6: Formato de las tramas

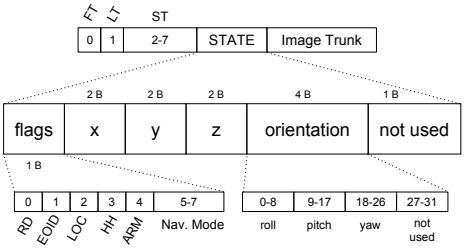


Figura 7: Mensaje que transmite el ROV

(Inter-Packet-Gap). Este tiempo ha de ser el suficiente para que el ROV transmita un paquete respuesta con su estado actual, sin colisionar con el siguiente paquete enviado por el operador. Es decir, el ROV solo lanzará la transmisión de un paquete inmediatamente después de la recepción de uno del operador. Se trata pues, de un TDMA (*Time Division Multiple Access*) controlado por el operador. La Fig. 6 muestra el formato de trama donde se encapsulan los paquetes del protocolo.

El protocolo integra el algoritmo de compresión progresivo DEBT (*Depth Embedded Block Tree*) [7], lo cual habilita al operador el poder especificar un tamaño fijo de la imagen comprimida. En cada paquete enviado por el ROV se transmite la odometría del mismo, unas *flags* de estado y de información del paquete, y una porción de la imagen codificada (ver Fig. 7). Por el otro lado, el operador envía paquetes con el estado deseado del robot. Estos paquetes contienen unas *flags* de control y una orden para el robot (ver Fig. 8a).

Mientras no se produzca una orden de alto nivel por parte del usuario, los paquetes enviados por el operador (ver Fig. 8a) transportan una orden de control de bajo nivel con el estado actual de los controles del *joystick* (ver Fig. 8b). Algunas de las órdenes de alto nivel pueden ser la actualización de los parámetros de la imagen (ver Fig. 8c) o una orden de posición para el ROV (ver Fig. 8d).

A continuación se detalla cada uno de los campos que contiene el mensaje enviado por el ROV (ver Fig. 7):

- **FT** (*First Trunk*): Indica que el trozo de la imagen comprimida contenido en el mensaje es el primero de la imagen.
- **LT** (*Last trunk flag*): Indica que el trozo de la imagen comprimida contenido en el mensaje es el último de la imagen. En caso de que toda la imagen esté contenida en el men-

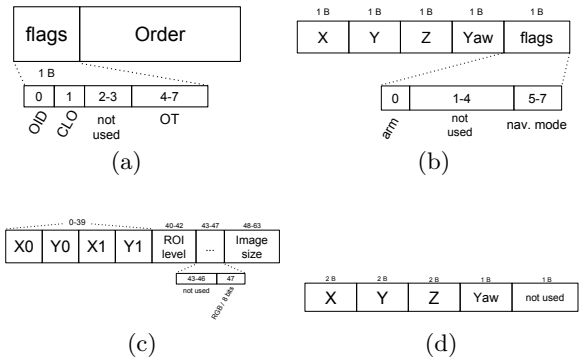


Figura 8: Formatos de los mensajes enviados por el operador. (a) Formato del mensaje. (b) Comando de velocidades. (c) Comando de compresión de imagen. (d) Comando Go To.

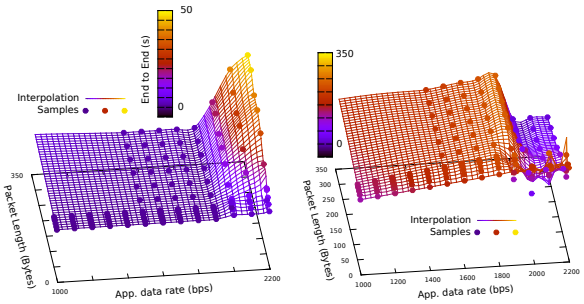
saje, se activa tanto FT como LT. Cuando se recibe este paquete, se ensamblan todos los trozos recibidos y se comprueba la integridad de la imagen mediante el código CRC (Comprobación de Redundancia Cíclica) contenido en los dos últimos bytes del ensamblado.

- **ST** (*State Type*): Este campo de 6 bits se reserva para posibilitar el envío de otros mensajes de estado. El estado del robot se representa en los campos que hay entre este y el campo IT.
- **IT** (*Image Trunk*): Campo que contiene un trozo de la imagen comprimida. El tamaño de este campo se calcula haciendo la diferencia entre el tamaño del payload de la trama donde se encapsula el mensaje (ver Fig. 6) y la suma de tamaños del resto de campos del mensaje.
- **RD** (*Ready*): Indica al operador si el robot está actualmente ocupado o no ejecutando una tarea de alto nivel.
- **EOID** (*Expected Order ID*): Codifica en un sólo bit el identificador o número de secuencia de la siguiente orden de alto nivel que espera recibir del operador. El ROV solo ejecutará una orden si su número de secuencia o OID (*Order ID*) es el mismo que el EOID. Este campo sirve también como ACK de la recepción de la última orden.
- **LOC** (*Last order cancelled*): Indica si se ha cancelado la última orden. Es decir, aquella con OID igual al complemento del EOID.
- **HH** (*Holding Heading*): Indica si el robot está manteniendo una orientación Yaw determinada. La activación/desactivación de este estado se realiza mediante una orden de alto nivel.
- **ARM** (*Armed*): Indica si el ROV está armado. Esto es, los motores estan activados para ejecutar las ordenes del usuario.
- ***Navigation Mode***: Indica el modo de navegación actual del ROV codificado en 3 bits. Los modos de navegación corresponden a los soportados por el software ArduSub: *Manual*, *Estabilize*, *Depth Hold*, *Hold Position*, y *Guided*.
- **Posición NED**: Se codifica la posición NED con dos bytes para cada eje.
- **Orientación**: Se dedican 9 bits para cada uno de los ángulos de navegación: roll, pitch y yaw.

Por otro lado, el operador (master) envía paquetes con el formato mostrado en la Fig. 8a. A continuación se detallan los campos de este mensaje:

- **OID** (*Order ID*): Codifica en un bit el identificador o número de secuencia de la orden. El ROV sólo tiene en cuenta este campo si se trata de una orden de alto nivel.
- **CLO** (*Cancel Last Order*): Un bit para solicitar la cancelación de la última orden. El ROV realizará la cancelación si el OID de este paquete es igual al OID que espera el ROV, es decir, el EOID que envía en sus mensajes (ver Fig. 7).
- **OT** (*Order Type*): Codifica en 4 bits el tipo de orden que transporta el mensaje. Dependiendo del tipo de orden el ROV considerará la orden de alto nivel o de bajo nivel.

Para habilitar el control por posición es necesario que se esté ejecutando un sistema de localización autónomo en el propio ROV. En la Fig. 4 se representa un sistema de localización por visión sobre ROS. Este programa publica la posición del ROV relativa a un punto del escenario, la cual es transformada por el *HROV Controller* a coordenadas geodésicas que son enviadas a la Pixhawk a través de mensajes de MAVLink de tipo GPS_INPUT. Por lo tanto, es imprescindible que el sistema de coordenadas del escenario virtual (ver Fig.5) corresponda exactamente con la realidad. También es necesario realizar una buena calibración tanto de la brújula como de los acelerómetros de la Pixhawk, además de estar aislados lo mejor posible de cualquier campo magnético artificial. Con todo esto funcionando, el software ArduSub habilita el



(a) (b)

Figura 9: Estudio de los módems S100. (a) Retardo de extremo a extremo. (b) Tasa de envío.

modo de navegación *Guided*, el cual, una vez activado por el operador permite la navegación del robot mediante comandos de posición.

4 Resultados

En primer lugar se ha ajustado el IPG (*Inter-Packet-Gap*) de los paquetes enviados por el master u operador para que el protocolo funcione sin problemas con los módems de RF S100. Para ello se ha lanzado la transmission de 2000 paquetes de diferentes tamaños y a diferentes velocidades de envío, desde 1 hasta 2.3 kbps. El punto donde empieza a aumentar el retardo (ver Fig. 9a) y a disminuir la tasa de envío (ver Fig. 9b) corresponde con la máxima tasa de envío de los módems que, en nuestro caso ha sido sobre los 1.9 kbps. Haciendo una regresión lineal del retardo de extremo a extremo se ha obtenido el retardo intrínseco del módem. Esta constante coincide con la ordenada en el origen del retardo de extremo a extremo, la cual ha sido de unos 85 ms.

Teniendo en cuenta el tamaño de los paquetes que envían tanto el robot como el operador, el retardo intrínseco del dispositivo y velocidad de transmisión y de propagación se ha ajustado el IPG de forma experimental para evitar la colisión de los paquetes del operador con los del ROV.

En los siguientes dos apartados se muestran resultados de una misma teleoperación ejecutada con módems reales en un entorno óptimo y una repetición de la misma simulando el canal de comunicación con errores.

4.1 Simulación HIL con Módems de RF

En este apartado se muestran los resultados de una simulación HIL (*Hardware In The Loop*) de una teleoperación con los módems S100. En este experimento se ha realizado la teleoperación de un BlueROV simulado utilizando el simulador SITL (Software In The Loop) de ArduPilot (ver `http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html`). Durante el control todo el tráfico de paquetes se ha realizado a través de los módems de RF sumergidos medio metro de profundidad y situados uno del otro a una distancia de metro y medio en un tanque de agua dulce.

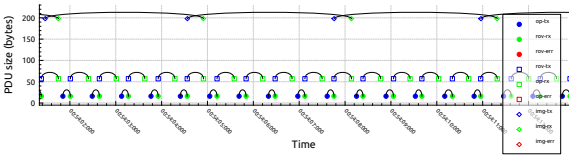
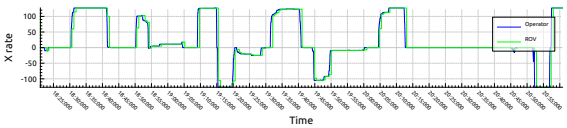
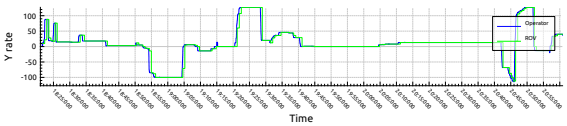


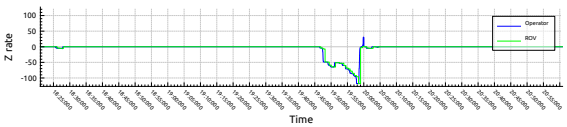
Figura 10: Lapso de tiempo de las comunicaciones a través de los S100.



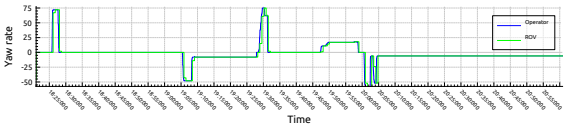
(a)



(b)



(c)



(d)

Figura 11: Control de velocidad con los modem S100. (a) Eje x. (b) Eje y. (c) Eje z. (d) Yaw.

La Fig. 10 muestra un lapso de tiempo de los flujos de paquetes y los eventos de captura y recepción de cada imagen. Como se puede observar, no existe solapamiento entre la transmisión de los paquetes del ROV y del operador. Debido a que no se han producido pérdidas de paquetes durante el experimento ha sido fácilmente posible enlazar gràficamente los eventos de transmisión con los eventos de recepción de paquetes y los eventos de captura de imagen con los de su recepción. La figura también muestra cómo son necesarios el envío de varios paquetes para completar la transmisión de una imagen.

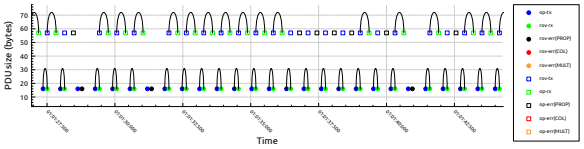


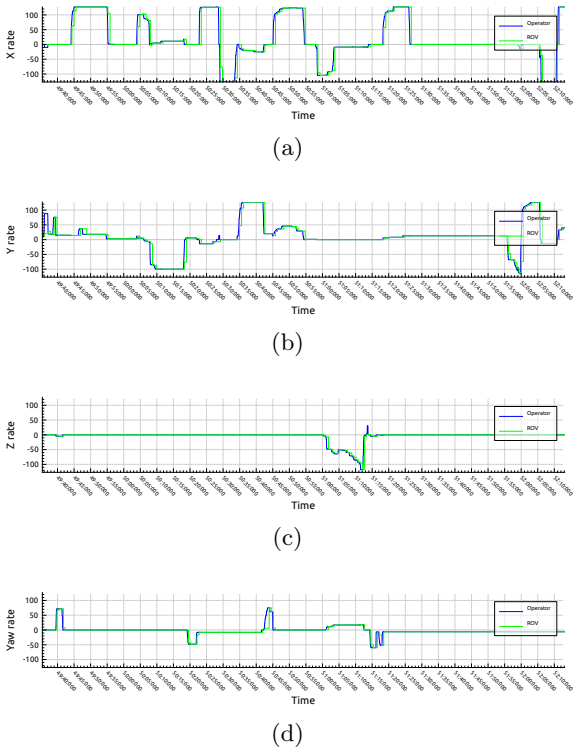
Figura 12: Lapso de tiempo durante comunicación simulada con una probabilidad de error por bit al 0.07%, un bit-rate de 1.9 kbps y un retardo intrínsico del dispositivo de 85 ms.

La Fig. 11 muestra el resultado del control del robot con comandos de bajo nivel. En ella se muestra, en azul, el estado de los controles de velocidad del *joystick* para cada uno de los ejes de desplazamiento x, y, z, y la rotación en *Yaw*. La línea verde de cada gráfico muestra la reacción del robot, la cual es una aproximación de la primera, aunque ligeramente desplazada debido al retardo de la comunicación. Se puede observar como es posible realizar un control en tiempo real, aunque el feedback visual no lo sea. Por este motivo es imprescindible visualizar el robot en un escenario virtual usando la información de odometría del ROV, la cual sí se recibe en tiempo real.

4.2 Simulación del Protocolo sobre un Canal con Errores

En los resultados mostrados en el apartado anterior no se han detectado errores por atenuación debido a la proximidad de los módems durante el experimento. Las limitadas dimensiones del tanque de agua disponible no nos han permitido experimentar a mayores distancias. Por lo tanto, para poner a prueba el protocolo, en este apartado se ha realizado una teleoperación SITL (*Software In The Loop*) simulando el canal de comunicación con pérdidas usando el módulo de comunicaciones de UWSim. De esta manera se pretende simular el canal de comunicación que se tendría, por ejemplo, en los límites del alcance de los módems.

Se ha realizado la misma teleoperación del apartado anterior de un ROV simulado, pero estableciendo una probabilidad de error por bit del 0.07%. Concretamente, se ha utilizado el modelo de error de NS3 *RateErrorModel* y el bit como unidad de error. La Fig. 12 muestra la captura de eventos de transmisión y recepción de paquetes en un lapso de tiempo durante la simulación. En ella se observa cómo son más frecuentes los errores en los paquetes enviados por el robot que en los enviados por el operador. Esto es debido a que los paquetes enviados por el robot tienen un tamaño bastante mayor. Debido a esto, resulta complicado recibir sin errores todos los paquetes necesarios para ensamblar la imagen. No obstante, la frecuencia de llegada de la odometría



(a) (b) (c) (d)

Figura 13: Control de velocidad simulando los módems con una probabilidad de error de bit fijada al 0.07%, un bit-rate de 1.9 kbps y un retardo intrínsico del dispositivo de 85 ms. (a) Eje x. (b) Eje y. (c) Eje z. (d) Yaw.

sigue siendo suficiente como para permitir al operador posicionar el robot en una escena de UWSim. Además, como se puede observar en los gáficos de la Fig. 13, el control del robot mediante el *joystick* sigue puediéndose realizar en tiempo real.

5 Conclusiones

En el presente artículo hemos presentado el estado actual del sistema de comunicaciones inalámbrico para el control remoto de un robot submarino. En concreto, se han presentado resultados relacionados con el protocolo de comunicaciones, utilizando como base un módem de radiofrecuencia, en el contexto del proyecto MERBOTS.

Teniendo en cuenta los resultados de los experimentos concluimos que es posible controlar remotamente un vehículo submarino de forma inalámbrica. No obstante, debido a la baja frecuencia de recepción de las imágenes será necesario el uso de de un módulo de realidad virtual o la implementación de un control supervisado (mediante comandos de alto nivel) y no teleoperado. Por lo tanto, se requerirá la integración de un sistema de localización inalámbrico para el ROV y, debido a las limitaciones del canal de comuni-

cación, también será necesario el uso de un protocolo de red cross-layer dedicado fuera de la pila TCP/IP. El sistema presentado permite tanto la teleoperación en tiempo real como un control supervisado del robot visualizando el resultado de los comandos antes de ser enviados. Además, permite la realimentación visual con región de interés y a una cadencia constante gracias a la integración del algoritmo de compresión progresivo DEBT.

Como trabajo futuro se plantea mejorar la inteligencia de los sensores de visión, con el objetivo de no solamente controlar el sistema de compresión de imágenes, sino también extraer información semántica (e.g. reconocimiento de objetos), a ser transmitida a mayor frecuencia, con el objetivo de mejorar la interacción con el ususario supervisor de las tareas robóticas.

Agradecimientos

Los autores agradecen el soporte del Gobierno Español, a través de los proyectos DPI2014-57746-C3 (MERBOTS) y DPI2017-86372-C3-1-R (TWINBOTS), y la ayuda predoctoral BES-2015-073112, la Universidad Jaume I de Castellón con el proyecto P1-1B2015-68 (MASUMIA), y la Generalitat Valenciana (PROMETEO/2016/066).

English summary

Wireless Teleoperation Architecture for ArduSub based ROVs with Visual Feedback

Abstract

An upgrade of the BlueROV architechture is introduced in this work, so that it enables wireless teleoperations using a channel with extremelly narrow bandwith. A cross layer protocol, appart from the TCP/IP stack is proposed. An progressive image compression algorithm, named DEBT (Depth Embedded Block Tree), is integrated within the teleoperation system. Thus, providing visual feedback. UWSim-simulator was used as a human-robot interface (HRI) virtual reality extension.

Keywords: Underwater robotics, wireless communications, sonar, radio frequency, network protocols, image compression, supervised control

Referencias

[1] X. Che, I. Wells, G. Dickers, P. Kear, and X. Gong. Re-evaluation of rf electromagnetic communication in underwater sensor networks. *IEEE Communications Magazine*, 48(12):143–151, 2010.

[2] H. Kaushal and G. Kaddoum. Underwater optical wireless communication. *IEEE Access*, 4:1518–1547, 2016.

[3] P. X. Liu, M. Q. H. Meng, P. R. Liu, and S. X. Yang. An end-to-end transmission architecture for the remote control of robots over ip networks. *IEEE/ASME Transactions on Mechatronics*, 10(5):560–570, Oct 2005.

[4] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

[5] M. Prats, J. Pérez, J. Fernández, and P. Sanz. An open source tool for simulation and supervision of underwater intervention missions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2577–2582, 2012.

[6] J. Ribas, D. Dura, and M. Stojanovic. Underwater video transmission for supervisory control and inspection using acoustic ofdm. In *OCEANS 2010*, volume 1, pages 1–9, New York, NY, Sept. 2010.

[7] E. M. Rubino, D. Centelles, J. Sales, J. V. Martí, R. Marín, P. J. Sanz, and A. J. Álvares. Underwater radio frequency image sensor using progressive image compression and region of interest. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Aug 2017.

[8] R. Wirz, R. Marin, M. Ferre, J. Barrio, J. M. Claver, and J. Ortego. Bidirectional transport protocol for teleoperated robots. *IEEE Transactions on Industrial Electronics*, 56(9):3772–3781, Sept 2009.