



Dyna

ISSN: 0012-7353

dyna@unalmed.edu.co

Universidad Nacional de Colombia

Colombia

Aristizábal, Luis M.; Rúa, Santiago; Gaviria, Carlos E.; Osorio, Sandra P.; Zuluaga, Carlos A.; Posada, Norha L.; Vásquez, Rafael E.

Design of an open source-based control platform for an underwater remotely operated vehicle

Dyna, vol. 83, núm. 195, febrero, 2016, pp. 198-205

Universidad Nacional de Colombia

Medellín, Colombia

Available in: <http://www.redalyc.org/articulo.oa?id=49644128025>

- How to cite
- Complete issue
- More information about this article
- Journal's homepage in redalyc.org

redalyc.org

Scientific Information System

Network of Scientific Journals from Latin America, the Caribbean, Spain and Portugal

Non-profit academic project, developed under the open access initiative

Design of an open source-based control platform for an underwater remotely operated vehicle

Luis M. Aristizábal ^a, Santiago Rúa ^b, Carlos E. Gaviria ^c, Sandra P. Osorio ^d,
Carlos A. Zuluaga ^e, Norha L. Posada ^f & Rafael E. Vásquez ^g

Escuela de Ingenierías, Universidad Pontificia Bolivariana, Medellín Colombia.

^a luismi911@gmail.com, ^b santiago.ruape@upb.edu.co, ^c cgaviriaceles@gmail.com, ^d sandrapatricia.osoriog@gmail.com, ^e carlos.zuluaga@upb.edu.co,
^f norha.posada@upb.edu.co, ^g rafael.vasquez@upb.edu.co

Received: March 25th, de 2015. Received in revised form: August 31th, 2015. Accepted: September 9th, 2015

Abstract

This paper reports on the design of an open source-based control platform for the underwater remotely operated vehicle (ROV) Visor3. The vehicle's original closed source-based control platform is first described. Due to the limitations of the previous infrastructure, modularity and flexibility are identified as the main guidelines for the proposed design. This new design includes hardware, firmware, software, and control architectures. Open-source hardware and software platforms are used for the development of the new system's architecture, with support from the literature and the extensive experience acquired with the development of robotic exploration systems. This modular approach results in several frameworks that facilitate the functional expansion of the whole solution, the simplification of fault diagnosis and repair processes, and the reduction of development time, to mention a few.

Keywords: open-source hardware; ROV control platforms; underwater exploration.

Diseño de una plataforma de control basada en fuente abierta para un vehículo subacuático operado remotamente

Resumen

Este artículo presenta el diseño de una plataforma de control basada en fuente abierta para el vehículo subacuático operado remotamente (ROV) Visor3. Primero se describe la plataforma de control original del vehículo con arquitecturas cerradas de hardware y software. La modularidad y la flexibilidad se establecen como guías para el diseño propuesto, dadas las limitaciones de la infraestructura previa. El nuevo diseño incluye las arquitecturas de hardware, software, firmware y control. Se usan plataformas abiertas de hardware y software para el desarrollo de la nueva arquitectura del sistema, con soporte en la literatura y la extensa experiencia adquirida en el desarrollo de sistemas robóticos de exploración. Esta aproximación modular arroja varias plataformas que facilitan, entre otros: la expansión funcional de la solución completa, la simplificación de los procesos de diagnóstico y reparación de fallos, y la reducción del tiempo de desarrollo.

Palabras clave: hardware de fuente abierta; plataformas de control de ROVs; exploración subacuática.

1. Introduction

Several underwater operations in aquaculture, port security, archeology, marine biology, offshore industry, to name but a few, are performed nowadays using underwater remotely operated vehicles (ROVs) [1]. The NORSEK U-102 standard [2] classifies such vehicles as follows:

- Class I – Pure observation

- Class II – Observation with payload option
- Class III – Work class vehicles
- Class IV – Seabed-working vehicles
- Class V – Prototype or development vehicles

A typical ROV system is comprised of an underwater vehicle, connected by a tether cable to a surface station where different tasks are specified through a mission control software interface.

Correa et al. [3] described the architecture for the conceptual design of underwater exploration vehicles; they state that several mechanical and hardware components are required for the appropriate operation of the system. The hardware subsystem constitutes the ROV's nucleus since it concentrates information provided by propulsion elements, a vision system, instrumentation, a power supply, a control system, among others [4].

The brain of the ROV is the on-board processor since it executes navigation, guidance, and control algorithms, and manages information exchange with the surface station using fiber optic communications, etc. [5]. Requirements such as robustness, reliability, processing speed, memory space, and flexibility are commonly specified during the control platform's design process [6].

Deep-water ROV operations demand navigation systems with high performance computation requirements since they use, for instance, Kalman filter-based algorithms to estimate the position, orientation, and velocity of the vehicle from measurements provided by different sensors (IMU, magnetometer, depth meter, DVL, USBL, among others) [7-12]. Such computation requirements are commonly met by using high-cost embedded processors with real-time operating systems such as the ones presented in [13,14].

Nowadays, several underwater vehicles and robotic platforms have been developed with low cost components [15-19], for instance, control platforms that use Arduino®-based hardware [20-25]. These open-hardware platforms can be used in ROVs that are intended for inspection tasks which are performed at low depths, and that do not require high performance real-time computations. Typically, Arduino®-based hardware is used to perform low-level tasks, such as data acquisition and communication with sensors and actuators, while the implementation of control algorithms and surface communication systems is done with open-source single-board computers such as Beaglebone® or Raspberry PI® [26]. Arduino platforms simplify the process of developing firmware by using a high level programming language, which implies a trade-off when trying to optimize code.

The use of low-cost open-hardware robotic platforms can be useful to motivate learning activities in people with no previous high-tech education; hence, they can be used for teaching tasks in programs related to science, technology, engineering, mathematics (STEM), ocean engineering, and marine sciences [27,28]. An interesting example is the MIT Sea Grant college program, developed with help from the Office of Naval Research (ONR) in the United States, and within that the Sea Perch ROV [29]. This ROV is comprised of low-cost equipment in order to easily teach students how to build an underwater vehicle [30].

This work addresses the design of an open-source control platform for the underwater remotely operated vehicle Visor3 [13], which has been used as a test platform for the development of robotic technology for the underwater exploration of Colombian seas. In this work, expansion capability and modularity of hardware, firmware, and software are established as general guidelines for the design process, since they are fundamental for open-source solutions.



Figure 1. Underwater remotely operated vehicle Visor3. Source: [13]

Section two of the paper describes the ROV Visor3 and details its original control platform and associated problems. The third section describes the approach followed to develop the new open-hardware architecture. Section four addresses software architecture development for each stage of the vehicle; including firmware, vehicle software, and surface station software. The fifth section describes the vehicle's control architecture and its relation with the hardware architecture elements. Finally, some conclusions are provided.

2. Remotely operated vehicle Visor3

Visor3, Fig. 1, is a Class I ROV that was designed to acquire visual information for the surveillance and maintenance of ships hulls and underwater structures in port facilities, and oceanographic research tasks [13]. The original hardware architecture of this ROV system was divided into two main parts: the surface control station that is comprised of an industrial joystick as command input, and a computer used as a Human-Machine Interface (HMI), connected to the vehicle using a fiber optic link; and the vehicle which is comprised of an IP camera, fiber optic communications devices, an embedded processor, multiple sensors (inertial measurement unit, thermometer, compass, etc.), thrusters, drivers, and DC power units [31], Fig. 2.

2.1. Original control platform

Visor3's original control platform was conceived, within a general framework for the control of unmanned vehicles [32]. This infrastructure relies on a central PC/104 form factor embedded processor that would allow one to use a real-time operating system (RTOS) in order to guarantee execution times for critical tasks, commonly present in systems such as in some unmanned vehicles with fast dynamics.

As is shown in Fig. 2, a CAN network was used to connect sensors, actuators, and other devices inside the vehicle. The network was designed as a modular tower board

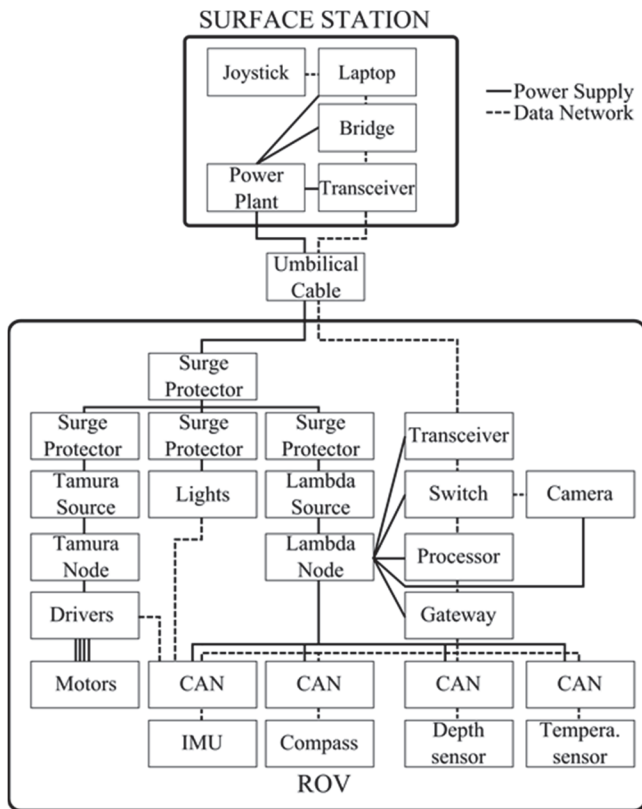


Figure 2. Hardware infrastructure in Visor3.
Source: The authors

platform, with six independent boards used to integrate sensors and actuator signals. Such general control architecture resulted in a complex bulky system with a lack of reliability; a negative characteristic for an underwater exploration system intended to be used in maintenance activities and oceanographic research.

After several deployments and operations on both laboratory and field environments with Visor3, the failure rate for both hardware and software increased. Additionally, due to the complexity of the control architecture, diagnostics and repair processes were delayed. Furthermore, the availability of technical information for several devices, e.g. the embedded processor, and the proprietary nature of software and hardware, restricted the implementation of improvements in order to increase the system's reliability.

3. New hardware architecture

The new open source-based hardware architecture has been designed by taking Visor3's original requirements and the experience acquired with its previous implementation into consideration. Additional considerations were taken into account, for instance, the use of low cost, highly available components, and preferably with a wide support community.

The hardware architecture is based on a hierarchical structure, with three defined levels. A higher, or surface level, that has all the components that are located outside the vehicle, i.e., in the surface control station, and includes the

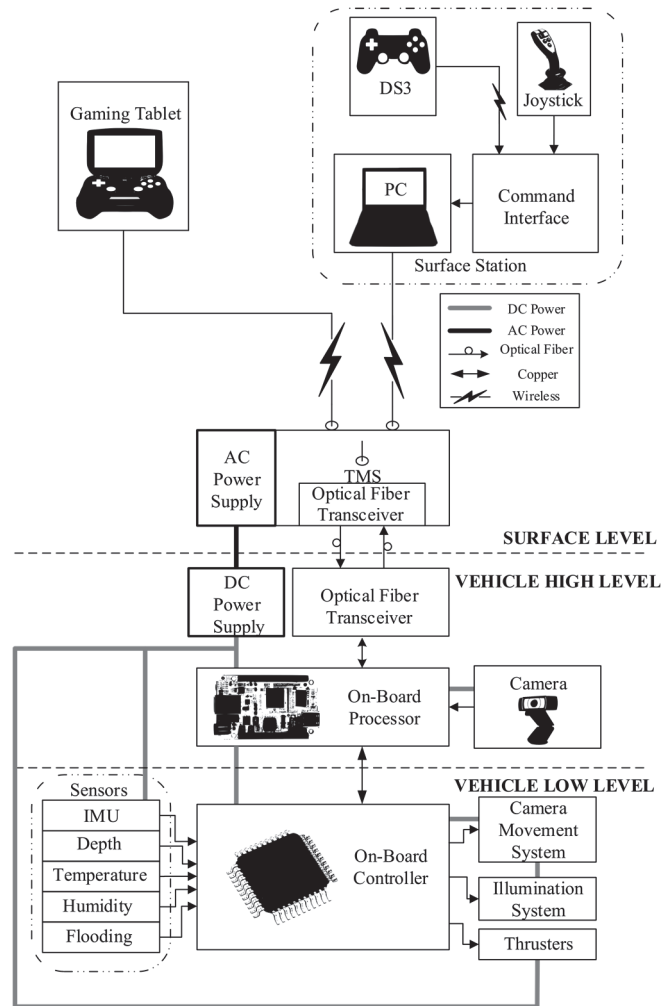


Figure 3. Proposed hardware architecture.
Source: The authors.

tether, power supply, and optical fiber communication devices. Middle and lower levels are located inside the ROV, so they can be identified as a vehicle's high and low-level respectively, as seen in Fig. 3.

3.1. Surface level

From a functional perspective, the surface level is comprised of two elements: a Tether Management System (TMS) and an operator interface. TMS is in charge of ensuring the ROV's power demand, and communication with the operator's interface, which includes a set of tools used by the pilot to command and drive the ROV.

Communication between the TMS and the operator interface is made through a wireless high speed WiFi link, thus allowing one to use a variety of open-source software enabled devices to control the ROV, e.g., personal computers (PC) equipped with joysticks and buttons, gaming tablets based on Android Operating Systems (OS) [33,34], and so on.

For this ROV, two operator devices are used. The first one is the surface station, and is comprised of the following devices: a computer for processing data coming from the

vehicle, including video streaming and visualization; a command interface, i.e., a device that integrates control elements, such as joysticks, buttons, indicators and a Global Positioning System (GPS), that gathers information and sends it to the computer; and an auxiliary wireless gamepad (DualShock3® or DS3) that can be connected through the command interface, and offers a portable alternative for the main control elements. The second device is wholly based on a commercial gaming tablet computer running Android OS, in which a Human-Machine Interface (HMI) can be implemented, allowing one to integrate video stream visualization, touch screen capabilities, analog joysticks, digital buttons and directional pads in a small, handheld device. The latter option was chosen mainly because Android OS based systems have been increasingly used in automation and remote control of vehicles [33], but it is not within the scope of this work.

3.2. Vehicle level

Regarding the on-board hardware system, it was devised as a two-layer structure, with each layer being a centralized subsystem controlled by a processing unit. The upper layer's central unit is an on-board open source embedded computer that is responsible for data reception from the surface station, video acquisition from a high-definition USB web camera, processing and streaming acquired images over a TCP/IP network, along with the vehicle status data, and, in short, every high level processing task that must be executed on-board. The lower layer's processing unit is an Arduino-compatible, 32-bit Micro Controller Unit (MCU) running at 84 MHz bus clock which has direct access to the ROV's devices, such as motor drivers and thrusters, lights, camera movement, and sensors, which include fault detection sensors (internal temperature and flooding are measured within the vehicle) and navigation sensors (Inertial Measurement Unit IMU and a depth meter). A communication link has been established between both layers to integrate all the ROV devices; this link allows one to treat the hardware system as a single unit. The integration of physical devices is made through a custom-made circuit board, designed to be compatible with additional components that allow the ROV extend its functionality.

Additional efforts are being made in order to maintain the architecture's flexibility to consider future growth and expansion; specifically on the ROV instrumentation, e.g., installing underwater positioning devices: an Ultra-Short Base Line (USBL) for positioning, a Doppler Velocity Log (DVL) for speed measurements, among others. The designed data acquisition system is equipped with appropriate hardware for reading additional sensor measurements, including standard connections used for instrumentation; for example, 4-20 mA signals or RS-232 serial communication. Moreover, an Arduino shield socket has been included in the system for the fast integration of Arduino shield-type expansion cards.

4. Software architecture

To define the appropriate software architecture for an ROV, some requirements must be taken into account.

Standards matching and portability are important aspects that need to be considered; in addition, the complexity of such applications requires modularity, reusability, and ease of integration between every element [35]. Hence, modularity is the main aspect that defines the guidelines for the final design of this software architecture.

A modular architecture confers a significant advantage over other approaches, as it supports massive reuse of existing code through different solutions; this allows one to reduce the development time of new applications, as has been demonstrated by several software platforms used for robotic devices and unmanned vehicles [36-39]. Although these software implementations are usually aimed at high level processing platforms, the designed architecture covers both high and low-level platforms making use of object-oriented programming, which has been recently enabled and optimized for microcontroller systems [40].

The general software architecture, as shown in Fig. 4, is composed of three layers. The bottom layer is firmware-based, and is in charge of low-level control of most devices in the ROV. The middle layer runs on the embedded on-board processor, and acts as middleware between the surface station and the low level controllers; it is in charge of communicating both layers and integrating a video signal. Finally, the upper level implements the user interface, including command devices, and every program and routine that cannot be executed on-board due to processing power limitations.

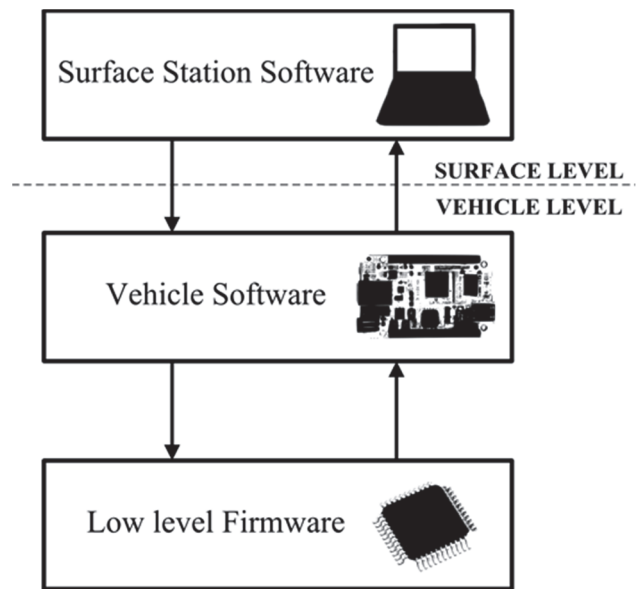


Figure 4. Overall Software Architecture.
Source: The authors.

4.1. Low level firmware

Following the established guidelines of modularity, and making use of an available open-source Object-Oriented Programming environment for microcontrollers, a firmware class structure was defined using C++ language, see Fig. 5. It is based on modules that can be added or removed without

significant modifications to the program. For this implementation, modules are treated as devices, mainly because of their specific functions related to the vehicle.

Every device depends on a group of core modules that cover basic functions, such as coordination and communication, between devices and with upper level elements of the architecture.

Device manager. This is the main core module of the firmware, designed as a general abstract base class that serves as a template for other devices. Every new device is a derivation of the device manager, allowing one to take advantage of polymorphism, and greatly simplifying the process of incorporating devices to the main program's workflow by providing a single method for executing every device's routine, regardless of the number of devices. Nevertheless, the maximum number of devices is limited, due to physical factors, e.g., memory or processing capacity, by the target implementation device.

Shared memory. In this particular application, communication between devices is a requirement due to functionality and safety of the vehicle; for instance, the fault detection module that depends on information from sensors, or the motor controller that needs information about energy consumption levels. For this reason, a shared memory module serves this purpose, allowing devices to share data seamlessly.

Communications module. This module handles communications with upper layers of the architecture, collecting data from devices, encapsulating them, and sending messages periodically according to the message's priority. The communication module is also in charge of receiving, parsing, and storing incoming data from higher level devices, making it available for the recipient device.

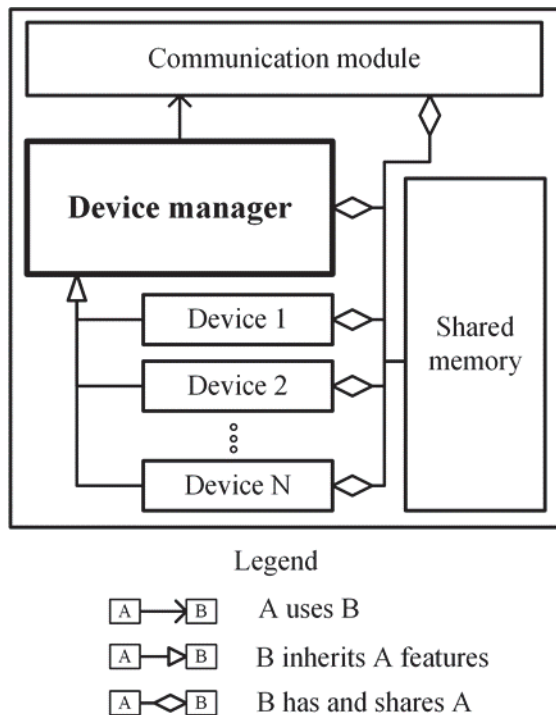


Figure 5. Low-level firmware general class structure.
Source: The authors

In the program flow, each device's routine is executed sequentially, each of their instances have been declared in the same order. Timer-based loops are defined for ensuring the time execution requirements of each device, e.g., a 10-millisecond loop must be used for acquiring IMU samples, and an additional 1000-millisecond loop is used for reading slower sensors, such as temperature, pressure, and flooding.

4.2. Vehicle's software

The vehicle's software is implemented in a Beaglebone embedded computer. This processor runs an Ångström operating system which is a Linux distribution used in embedded devices with built-in components. From the point of view of functionality, the vehicle's onboard processor is responsible for executing two main processes: the first one is focused on running a video streaming server that deals with camera operation, receives frames, compresses and sends them to the surface station; and the second process handles communication between the surface station and the vehicle, i.e., it allows data to be exchanged between the surface level and the vehicle level.

For communication with sensors, according to the proposed hardware architecture, the BeagleBone uses a serial link with the microcontroller that handles low-level processes. Then, the processor is responsible for sending all data acquired from sensors to the surface station, via UDP protocol. Both communication tasks are performed by a routine developed on a programming environment based on JavaScript called Node.js, which allows one to create highly scalable network programs such as web servers (streaming video, TCP / UDP, etc.). This framework enables the development of modular software to import and export modules developed by third parties, implementing several middleware utilities to be used in web applications, such as express frameworks [41].

Further Node.js comes with an API covering low-level networking, basic HTTP server functionality, file system operations, compression and many other common tasks. These libraries are used to adjust the communication protocol and send data frames to the surface station via Ethernet on UDP protocol [42].

4.3. Surface station's software

A similar approach to the one used in the firmware architecture was applied on the surface station, i.e., easy integration of specific functions in the form of modules, through a framework that handles essential tasks to ensure cohesion between all parts of the solution. The main difference between software and firmware is the target hardware platform; firmware runs on a device with limited processing resources compared to a personal computer, enabling the latter to be used with higher level developing environments.

To take advantage of the capabilities of the surface hardware platform, which in this case is a personal computer (PC) running Windows® OS, National Instruments™ LabVIEW® graphical dataflow programming environment was selected in order to allow one use parallel execution code through processor multithreading [43].

The proposed software structure uses parallel execution with three loops designed to run simultaneously; each loop takes care of a group of tasks, as shown in Fig. 6. The first loop, top down, is in charge of receiving and assembling incoming data from the vehicle and the command interface (see Fig. 3); then, it stores each message in a buffer for further reading and processing. The middle loop takes assembled frames as they become available and extracts, classifies, and stores data, making them available to other modules in the program. The bottom loop is responsible for collecting command data and transmitting them to the ROV; this is the designated space for adding functional modules to the program. This loop allows one to add and remove functional modules, as long as their time requirements meet the designated loop time characteristics; they are executed sequentially. For example, if the navigation algorithm's execution time is at least 100 ms, and the transmission loop time is defined to 50 ms, a delay will occur in the data exchange with the ROV. To solve this issue, additional loops with specific timings, different from the existing loops can be created, allowing one to add functional modules with longer or shorter execution times.

Caution measures must be taken when designing functional modules regarding execution time, due to limitations on the processing power of the host machine, so undesired delays can be avoided.

5. Open-source control platform

The navigation, guidance and control (NGC) system for an underwater vehicle can have different degrees of sophistication, depending on the type of operation that is to be performed, and the autonomy levels that need to be achieved [44, 45].

One of the important vehicle design parameters is the number of degrees of freedom needed to perform the planned operations, because they represent the number of independent

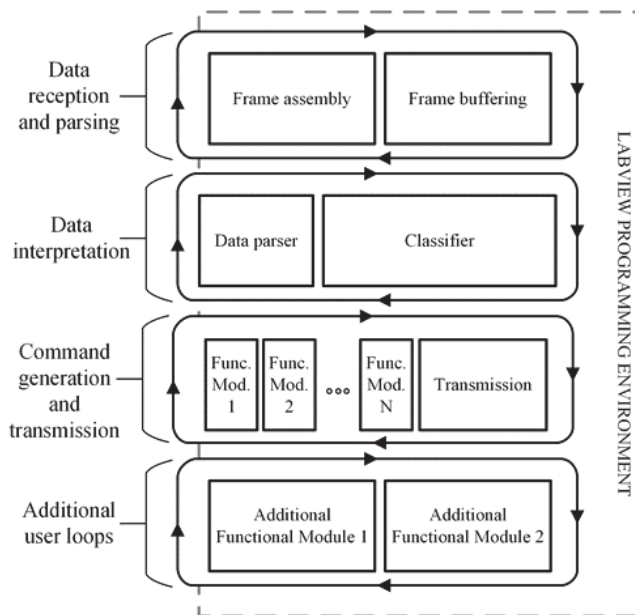


Figure 6. Surface Station software structure.
Source: The authors.

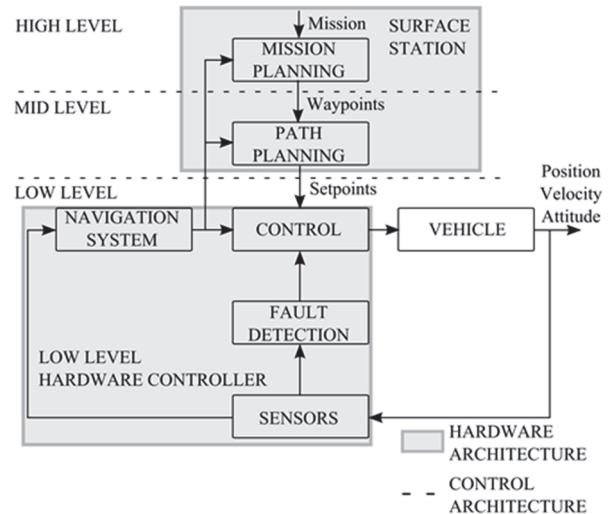


Figure 7. Control and hardware architecture of the Visor3 ROV.
Source: The authors.

movements that the vehicle can achieve in the three-dimensional space. Additionally, the tasks that are to be performed determine the instrumentation (sensors, actuators, complementary systems, among others) required to control the vehicle. Fig. 7 shows a three-level hierarchical NGC structure for an underwater vehicle; this kind of structure is useful to control and stabilize the vehicle. Additionally, each element of the control architecture is related to a specific group of hardware that executes the corresponding tasks.

The high and middle levels in the control structure are executed in the surface station. The objective is to allow the operator define the vehicle's mission using a high level language and translate it into commands to the low-level control system. Through the use of the PC, joystick and buttons, the operator is capable of moving the ROV in accordance with the independent movements that the vehicle can make. The joystick's three degrees-of-freedom are used to move in surge, sway, and yaw directions; heave motions are commanded by using 2 buttons, with increments and decrements of the speed in this direction (up/down). The graphical user interface (GUI) allows the operator to follow the mission's progress and the position of the vehicle.

The low-level control structure is executed in the ROV's hardware. The navigation system, with information provided by all sensors, allows one to estimate the position, velocity, and attitude of the vehicle with respect to an inertial frame located in the surface station. The control component contains the algorithms that stabilize the state of the vehicle, in order to follow the commands given by the operator. The Beaglebone embedded computer executes different algorithms in order to achieve the desired movements.

6. Conclusions

This paper addressed the design of an open source-based control platform for an underwater exploration ROV. Previous experiences obtained through the development and

deployment of the ROV Visor3 have been compiled and analyzed, and shortcomings of using closed-source elements have been identified.

A major part of the selected hardware and software elements are distinguished by their open-source features; the main benefits are the high availability of information and extended support communities, differentiating them from closed-source solutions with limited information and, in some cases, poor support.

A modular approach for developing software architectures for robotic platforms has been presented. The results show that several frameworks facilitate, among others: functional expansion, simplification of fault diagnosis and repair processes, and reduction of development time. The software platforms selected and used in the design have been comprehensive, available information and extensive support communities, factors that are key for a rapid and successful development. Furthermore, programming languages such as C++, JavaScript and LabVIEW® have proved to be adequate tools for developing modular software frameworks for robotic devices. Additionally, software development has been undertaken using Python, a high-level programming language that allows one to implement control algorithms that require complicated matrix computations.

Finally, it has been shown how each component of the hardware architecture has a role in the control platform. The use of a compact hardware architecture can help one to increase reliability, maintenance and durability, without compromising the controllability of the vehicle.

Even though, there are observation-class ROVs with high-technology equipment such as the VideoRay such ROVs are not manufactured with open-source philosophies, therefore, they are difficult to repair or improve with changing operation requirements. In comparison with the OpenROV, which is another open source underwater vehicle solution; Visor3 has better instrumentation and the appropriate robustness to accomplish observation tasks in harsher environments.

Acknowledgements

This work was developed with the funding from the Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación, Francisco José de Caldas; the Colombian petroleum company, ECOPETROL; the Universidad Pontificia Bolivariana – Sede Medellín, UPB; the Universidad Nacional de Colombia – Sede Medellín, UNALMED; through the Strategic Program for the Development of Robotic Technology for Offshore Exploration of the Colombian Seabed, project 1210-531-30550, contract 0265 – 2013.

References

- [1] Christ, R. and Wernli, R., The ROV Manual. A User guide for remotely operated vehicles, 2nd Ed. Elsevier, 2013. DOI: 10.1016/B978-0-08-098288-5.00033-6
- [2] NORSOK, NORSOK STANDARD U-102, Norwegian Technology Centre Std., 2nd Ed., [Online]. 2012 [Date of reference: January 20 of 2015]. Available at: <http://www.standard.no/petroleum>
- [3] Correa, J.C., Vásquez, R.E., Ramírez-Macias, J.A., Taborda, E.A., Zuluaga, C.A., Posada, N.L. and Londoño, J.M., Una arquitectura para el

- diseño conceptual de vehículos para exploración subacuática Ingeniería y Ciencia, 11(21), pp. 73-97, 2015. DOI: 10.17230/ingciencia.11.21.4
- [4] Bai, Y. and Bai, Q., Subsea engineering handbook, in Subsea Engineering Handbook, Eds. Boston: Gulf Professional Publishing, 2010, pp. 3-25. DOI: 10.1016/B978-1-85617-689-7.10001-9
- [5] Kim, T. and Yuh, J., Development of a real-time control architecture for a semi-autonomous underwater vehicle for intervention missions, Control Engineering Practice, 12(12), pp. 1521-1530, 2004. DOI: 10.1016/j.conengprac.2003.12.015
- [6] Yoshida, H., Underwater Vehicles. In Tech, no. 29, ch. Fundamentals of Underwater Vehicle Hardware and Their Applications, 2009, pp. 557-582. DOI: 10.5772/6721
- [7] Caccia, M., Casalino, G., Cristi, R. and Veruggio, G., Acoustic motion estimation and control for an unmanned underwater vehicle in a structured environment. Control Engineering Practice, 6(5), pp. 661-670, 1998. DOI: 10.1016/S0967-0661(98)00057-4.
- [8] Caccia, M. and Veruggio, G., Guidance and control of a reconfigurable unmanned underwater vehicle. Control Engineering Practice, 8(1), pp. 21-37, 2000. DOI: 10.1016/S0967-0661(99)00125-2.
- [9] Blain, M., Lemieux, S. and Houde, R., Implementation of a ROV navigation system using acoustic/Doppler sensors and Kalman filtering. In Proceedings of the OCEANS 2003, pp. 1255-1260, 2003. DOI: 10.1109/OCEANS.2003.178033.
- [10] Loebis, D., Sutton, R., Chudley, J. and Naeem, W., Adaptive tuning of a Kalman filter via fuzzy logic for an intelligent AUV navigation system. Control Engineering Practice, 12(12), pp. 1531-1539, 2004. DOI: 10.1016/j.conengprac.2003.11.008.
- [11] Lee, P.M. and Jun, B.H., Pseudo long base line navigation algorithm for underwater vehicles with inertial sensors and two acoustic range measurements. Ocean Engineering, 34(3-4), pp. 416-425, 2007. DOI: 10.1016/j.oceaneng.2006.03.011.
- [12] Geng, Y. and Sousa, J., Hybrid derivative-free EKF for USBL/INS tightly-coupled integration in AUV. In Proceedings of the 2010 International Conference on Autonomous and Intelligent Systems (AIS), pp. 1-6, 2010. DOI: 10.1109/AIS.2010.5547035.
- [13] Gutiérrez, L.B., Zuluaga, C.A., Ramírez, J.A., Vásquez, R.E., Flórez, D.A., Taborda, E.A. and Valencia, R.A., Development of an underwater remotely operated vehicle (ROV) for surveillance and inspection of port facilities, in Proceedings of the ASME IMECE2010, 2010. DOI: 10.1115/IMECE2010-38217
- [14] Huang, H., Wan, L., Pang, Y. and Qin, Z., Control architecture of SY-II ship inspection remote operated vehicle, in Computer Science Service System (CSSS), 2012 International Conference on, pp. 2322-2325, 2012. DOI: 10.1109/CSSS.2012.576
- [15] Wang, B., Wu, C. and Ge, T., A low cost compact control system for the Hippo ROV, Applied Mechanics and Materials, 190-191, pp. 627-633, 2012. DOI: 10.4028/www.scientific.net/AMM.190-191.627
- [16] Wang, B., Wu, C. and Ge, T., Development of a remotely operated vehicle test-bed, Sensors & Transducers, [Online]. 153, pp. 45-52, 2013. Available at: http://www.sensorsportal.com/HTML/DIGEST/P_1218.htm
- [17] Bonarini, A., Matteucci, M., Migliavacca, M. and Rizzi, D., R2P: An open source hardware and software modular approach to robot prototyping, Robotics and Autonomous Systems, 62(7), pp. 1073-1084, 2014. DOI: 10.1016/j.robot.2013.08.009
- [18] Ahmed, Y., Yaakob, L. and Bong, S., Design of a new low cost ROV vehicle, Jurnal Teknologi 69(7), pp. 1-11, 2014. DOI: 10.11113/jt.v69.3262
- [19] Sinha, P., Stiehl, K.R., Huo, E.S., Oyebo, O.A., Dokov, R.P., Chin, S.J., Price, R.E. and Larson, R.W., Design of a modular, compact, multi-role remotely operated vehicle for sheltered water operations, in IEEE OCEANS 2007, pp. 1-7, 2007. DOI: 10.1109/OCEANS.2007.4449392
- [20] Busquets, J., Busquets, J.V., Tudela, D., Perez, F., Busquets-Carbonell, J., Barbera, A., Rodriguez, C., Garcia, A.J. and Gilabert, J., Low-cost AUV based on Arduino open source microcontroller board for oceanographic research applications in a collaborative long term deployment missions and suitable for combining with an USV as autonomous automatic recharging platform, in Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES, pp. 1-10, 2012. DOI: 10.1109/AUV.2012.6380720
- [21] Johansson, P. and Bernhard, J., Advanced control of a remotely operated underwater vehicle, Department of Electrical Engineering, Linköping

- universitet, Sweden, Tech. Rep., [Online]. 2012. Available at: <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-79364>
 - [22] Fittery, A., Mazumdar, A., Lozano, M. and Asada, H., Omni-Egg: A smooth, spheroidal, appendage free underwater robot capable of 5 dof motions, in IEEE OCEANS 2012, pp. 1-5, 2012. DOI: 10.1109/OCEANS.2012.6404986
 - [23] Hassan, O.I.H., Black-box identification and control for autonomous underwater vehicles, PhD. Dissertation, School of Engineering and Information Technology, The University of New South Wales, Australian Defence Force Academy, [Online]. 2013. Available at: <http://handle.unsw.edu.au/1959.4/52888>
 - [24] Portugal, D. and Rocha, R.P., Distributed multi-robot patrol: A scalable and fault-tolerant framework, Robotics and Autonomous Systems, 61(12), pp. 1572-1587, 2013. DOI: 10.1016/j.robot.2013.06.011
 - [25] Faugel, H. and Bobkov, V., Open source hard- and software: Using Arduino boards to keep old hardware running, Fusion Engineering and Design, 88(6-8), pp. 1276-1279, 2013. DOI: 10.1016/j.fusengdes.2012.12.005
 - [26] Grimmer, R., Mastering beagle bone robotics. Packt Publishing, 2014
 - [27] Quintián, H., Calvo, J.L. and Fontenla, O., Aplicación de un robot comercial de bajo coste en tareas de seguimiento de objetos. DYNA, 79(175), pp. 24-33, 2012. ISSN 2346-2183.
 - [28] Álvarez-Chavarría, J.S., Jiménez-Builes, J.A. and Ramírez-Patiño, J.F., Design cycle of a robot for learning and the development of creativity in engineering. DYNA, 78(170), pp. 51-58, 2011.
 - [29] VanCott, R., Wilbur, B.M., Chrysostomidis, C., Soroka, M. and Shroyer, K., STEM education through open hardware at MIT sea grant, Open Hardware Journal, [Online]. 1, pp. 9-15, 2011. Available at: <https://archive.org/details/OpenHardwareJournal>
 - [30] Schneider, D., Build your own robosub [Hands On], IEEE Spectrum, 48(9), pp. 24-26, 2011. DOI: 0.1109/MSPEC.2011.5995890
 - [31] Tirado, D., Asistencia a la investigación en la educación e implementación del hardware (joystick, IMU, profundímetro, interfaz con actuadores), para el proyecto ROV. Trabajo de grado, Facultad de Ingeniería Eléctrica/Electrónica, Universidad Pontificia Bolivariana, Medellín, Colombia, 2008.
 - [32] Franco, J., Frameworks para sistemas de control embebidos de tiempo real. Trabajo de grado, Especialización en Automática, Universidad Pontificia Bolivariana, Medellín, Colombia, 2007.
 - [33] Nadvornik, J. and Smutny, P., Remote control robot using android mobile device, in Control Conference (ICCC), 15th International Carpathian, pp. 373-378, 2014. DOI: 10.1109/CarpathianCC.2014.6843630
 - [34] Speers, A., Forooshani, P.M., Dicke, M. and Jenkin, M., Lightweight tablet devices for command and control of ROS-enabled robots, in Advanced Robotics (ICAR), 16th International Conference on, 2013. DOI: 10.1109/ICAR.2013.6766481
 - [35] Bertolotti, I.C. and Hu, T., Modular design of an open-source, networked embedded system, in Computer Standards & Interfaces, 37, pp. 41-52, 2015. DOI: 10.1016/j.csi.2014.05.004.
 - [36] Gerkey, B.P., Vaughan, R.T. and Howard, A., The player/stage project: Tools for multi-robot and distributed sensor systems, in: Proceedings of the 11th International Conference on Advanced Robotics, 2003, pp. 317-323.
 - [37] Bruyninckx, H., Open robot control software: The OROCOS project, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, 3, pp.2523-2528 2001. DOI: 10.1109/ROBOT.2001.933002
 - [38] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y., ROS: An open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
 - [39] Huang, A., Olson, E. and Moore, D., LCM: Lightweight communications and marshalling, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010, pp. 4057-4062. DOI: 10.1109/IROS.2010.5649358
 - [40] Becker, L.B., Gergeleit, M. and Nett, E., Approach for implementing object-oriented real-time models on top of embedded targets, in OMER-2 – Workshop on O-O Modeling of Embedded RT Systems, 2001.
 - [41] Tilkov, S. and Vinoski, S., NodeJS: Using JavaScript to build high-performance network programs, internet computing, IEEE, 14(6), pp. 80-83, 2010. DOI: 10.1109/MIC.2010.145
 - [42] Ojamaa, A. and Duuna, K., Assessing the security of NodeJS platform, 2012 International Conference for Internet Technology and Secured Transactions, IEEE, pp. 348-355, 2012.
 - [43] National Instruments™, Benefits of Programming Graphically in NI LabVIEW, [Online]. 2013. Available at: <http://www.ni.com/white-paper/14556/en/pdf>.
 - [44] Roberts, G.N., Trends in marine control systems, Annual Reviews in Control, 32(2), pp. 263-269, 2008. DOI: 10.1016/j.arcontrol.2008.08.002
 - [45] Chyba, M., Haberkorn, R.S., and Choi, S., Design and implementation of time efficient trajectories for autonomous underwater vehicles. Ocean Engineering, 35(1), pp. 63-76, 2008. DOI: 10.1016/j.oceaneng.2007.07.007
- L.M. Aristizábal**, IEO, received a BSc. in Electronic Engineering in 2013 from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. He is currently studying for a Master's degree in the field of Engineering with emphasis in Automation at the UPB. His research interests are hardware and software architectures, robotics, and automation systems.
ORCID: 0000-0003-0607-9613
- S. Rúa**, MSc., received a BSc. in Electronic Engineering in 2013, and an MSc. in Engineering with emphasis in Automation in 2015, both from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. He is currently a PhD student in Engineering at the UPB, in the area of control and navigation systems. His research interests are control of dynamic systems, navigation systems, machine learning, and embedded systems.
ORCID: 0000-0003-0067-8316
- C.E. Gaviria**, IEO, received a BSc. in Electronic Engineering in 2011 from the Universidad de Antioquia, Medellín, Colombia. He is currently an MSc. student in Engineering at the UPB, in the area of control and automation systems.
ORCID: 0000-0003-2038-581X
- S.P. Osorio**, IEO, received a BSc. in Electronic Engineering in 2014 from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. She is currently studying for a Master's degree in the field of Engineering with emphasis in Automation at the UPB.
ORCID: 0000-0003-1985-499X
- C.A. Zuluaga**, MSc., received a BSc. in Electronic Engineering in 1999, and an MSc. in Engineering with emphasis in Automation in 2006, both from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. He is currently professor in the Department of Electrical Engineering at the UPB. He currently coordinates the graduate programs in automation and participates in several research projects related to control engineering. For several years his research work has been related to autonomous underwater vehicle control (AUV) and remotely operated vehicles (ROV).
ORCID: 0000-0002-1773-767X
- N.L. Posada**, MSc., received a BSc. Eng in Instrumentation and Control Engineering in 2000 from the Politecnico Colombiano Jaime Isaza Cadavid, and an MSc. in Engineering with emphasis in Automation in 2010 from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. Norha is currently professor in the Department of Mechanical Engineering at the UPB, in the area of systems and control. Her research interests are instrumentation, process automation, and state observer design.
ORCID: 0000-0002-7607-0756
- R.E. Vásquez**, PhD., received a BSc. in Mechanical Engineering in 2002, and an MSc. in Engineering with emphasis in Automation in 2007, both from the Universidad Pontificia Bolivariana (UPB), Medellín, Colombia. He received his PhD. in Mechanical Engineering from the University of Florida, USA in 2011. Rafael is currently professor in the Department of Mechanical Engineering at the UPB, in the area of dynamics, systems and control. His research interests are theory of mechanisms; robotics; design, analysis, and control of dynamic systems; tensegrity systems; new technologies for energy harvesting; and engineering education. He is a member of the American Society of Mechanical Engineers (ASME) since 2005.
ORCID: 0000-0003-4871-8823