Remote Piloting Development of an ROV

Jefferson Osowsky
Department of Innovation
OceanPact - Serviços Marírimos S.A.
Rio de Janeiro, RJ, Brazil
Email: jefferson.osowsky@oceanpact.com

Tin Muskardin
Department of Innovation
OceanPact - Serviços Marírimos S.A.
Rio de Janeiro, RJ, Brazil
Email: tin.muskardin@oceanpact.com

Leonardo M. Barreira
Department of Innovation
OceanPact - Serviços Marírimos S.A.
Rio de Janeiro, RJ, Brazil
Email: leonardo.barreira@oceanpact.com

***Abstract*—This work presents a project that consists of turning a commercial Remote Operated Vehicle (ROV) into a Remotely Piloted ROV. It was taken two approaches. First we developed a serial/ethernet converter to transfer data from the hand controller to the ROV control console and vice-versa. In this approach, an ESP32 microcontroller was used to receive serial data in the 9-bit framing protocol and transmit them over a network connection. The reverse operation was carried out on the other side by another ESP32. The second approach was based on replacing the ROV remote control with an Single Board Computer (SBC), whose function was to emulate the signals sent to the ROV by the remote control. It will be presented the development and the achieved results for these two methods.**

I. Introduction

In this project, we intend to upgrade the video and control systems of the Outland 2000 Remotely Operated Vehicle[1] (ROV) [1], [2], turning it into a remotely piloted ROV as illustrated in Figure 1. Thereby, instead of connecting by wire the control console to the hand controller we have inserted between them a subsystem composed of serial/ethernet converters and video encoder and decoder. Through this subsystem we will have built a full ROV piloting solution via remote connection by means of satellite or wireless network links. This is also known as 'Onshore Piloting System'.

Searching the scientific literature, we did not find many works concerned with solving problems similar to this one. Indeed, only a few projects were found in the literature, namely [3], [4], [5] and [6]. On the other hand, companies like TechnipFMC, Oceaneering, and Subsea7 have already developed and tested their own remotely piloted ROVs since 2022, 2016, and 2023, respectively.

This work is divided as follows: First, in Section II the authors present the development of this project, turning the original ROV system into a teleoperated ROV solution. Then, in Section III the results achieved so far are presented. Finally, in Section IV we present some concluding remarks about this project.

II. Development

In this work, we approach the remote piloting problem by dividing it into a video transmission problem and a data exchange problem through an RS-485 serial interface. The former will not be addressed in this paper because we consider that its solution is already well defined with the use of video streaming encoders and decoders. The latter has been solved by means of the development of a RS-485 Serial/Ethernet converter. Note in Figure 1 that the control console and hand controller when connected by wire, exchange data through standard RS-485 serial communication. In a remote piloting system, this connection must necessarily go through a satellite or wireless communication network. Thus, first we need to translate from RS-485 serial data into ethernet data. Next, ethernet data are transmitted to the other Ethernet/Serial converter via switches/routers. Finally, these data are converted back to Serial data.

However, an issue that arises in this project is that the data framing protocol used in the RS-485 multidrop communication between the control console and the hand controller is 9-bit data framing instead of the usual 8-bit, the differences between them are shown in Figure 2. This uncommon type of serial communication protocol is used to identify if a sequence of 8-bits transmitted to a device is an address message (setting the parity bit to mark) or a data message (setting the parity bit to space). This way, it is possible to create a half-duplex serial master/slave network where two or more devices, each identified by a unique address, can exchange data with each other.

Since most Universal Asynchronous Receiver-Transmitter (UART) chipsets in commercial microprocessors and microcontrollers are 8-bit mode only, we had to implement our own serial UART 9-bit mode to Ethernet converter by software. The processing hardware chosen for this development was the ESP32-DevKitC module from Espressif with the following main features: Xtensa® dual-core 32-bit LX6 CPU, frequency up to 240MHz; 448 KB of ROM, 520 KB of SRAM, and 4 MB of Flash memory; WiFi 802.11b/g/n; and up to 26 General Purpose Input/Output (GPIOs). The source code of this project has been implemented in C programming language and we have chosen the ESP-IDF from Espressif as development framework. The result of this work is shown in Figure 3. On the right side is the Serial/Ethernet converter that is connected to the hand controller. On the other hand, the left side is the Serial/Ethernet converter that is connected to the control console. Below, in Figure 4, it is shown an example of an RS-485 serial output signal from the hand controller which is sent to the ROV control console.

[1]https://www.outlandtech.com/rov-home/rov2000

OCEANS 2024 - Halifax | 979-8-3315-4008-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/OCEANS55160.2024.10753704

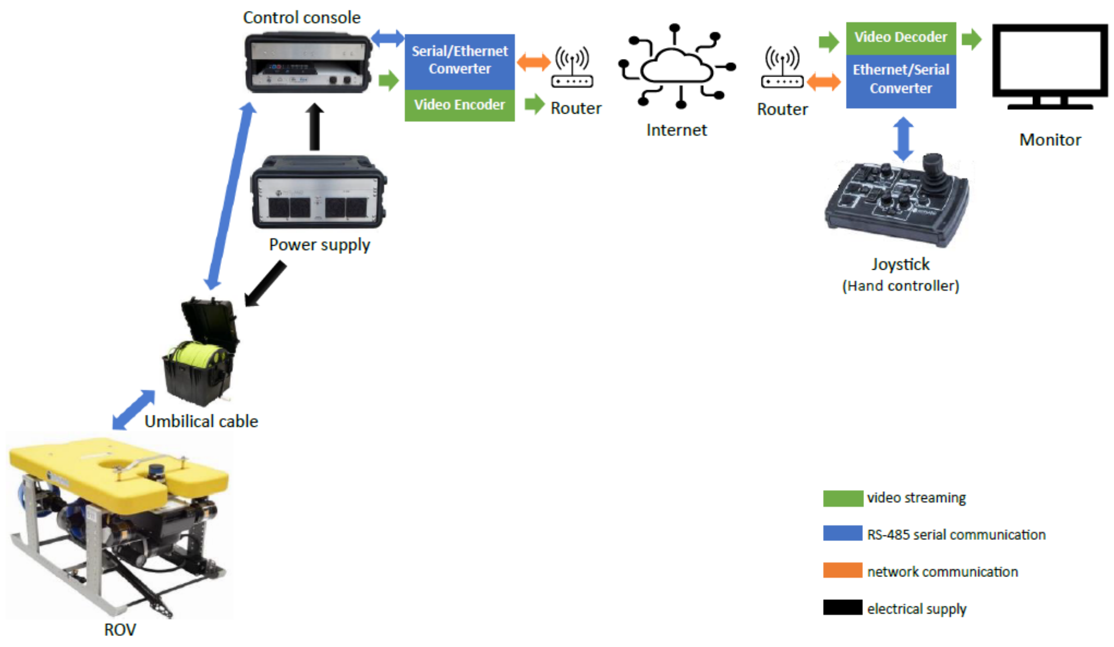


Fig. 1. System architecture for IP-based remote control.

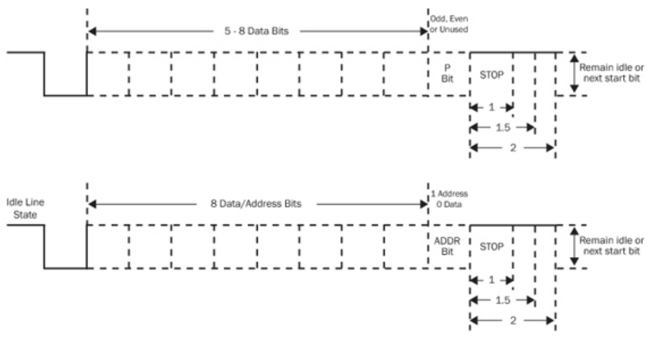


Fig. 2. Comparation of 8-bit and 9-bit data framing protocol.

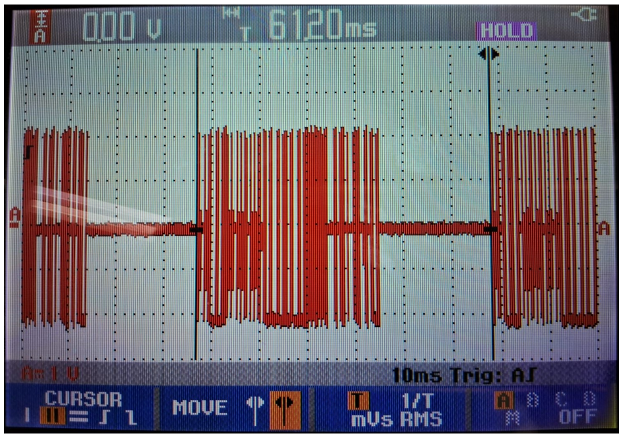


Fig. 4. An example of an RS-485 serial output signal from the hand controller to the control console.

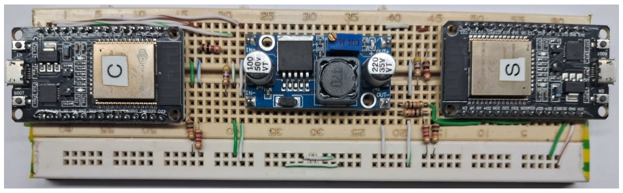


Fig. 3. Serial/Ethernet converter connected to the ROV hand controller (right side), and Serial/Ethernet converter connected to the ROV control console (left side).

It is worth noting in this Figure that the commands that control the ROV are repeated periodically at an interval of 61.20 milliseconds. We figured out that if this interval is greater than 100 milliseconds then the ROV stop working.

As will be presented in Section III, the results achieved by the approach were good enough when the system was connected to a Local Area Network (LAN), allowing us to pilot remotely the ROV, but not good enough when it was connected to a Wide Area Network (WAN) duo to some network communication failures, loss of data packets, timing issues, and network latency.

In contrast with the above development, we decided to tackle this problem by mean of another approach. So far, our system could be considered only as a gateway equipment that received serial data from the hand controller (client side), turned it into an network data packet, sent this packet to the server side via a network connection, turned it into serial data again to sent it to the ROV control console.

In this new approach, we introduce a Single Board Computer (SBC), namely a Rapberry Pi 4, which is connected to the control console directly, and operates as the ROV hand controller, sending periodically commands to the ROV. A block diagram of the ROV side is presented in Figure 5.

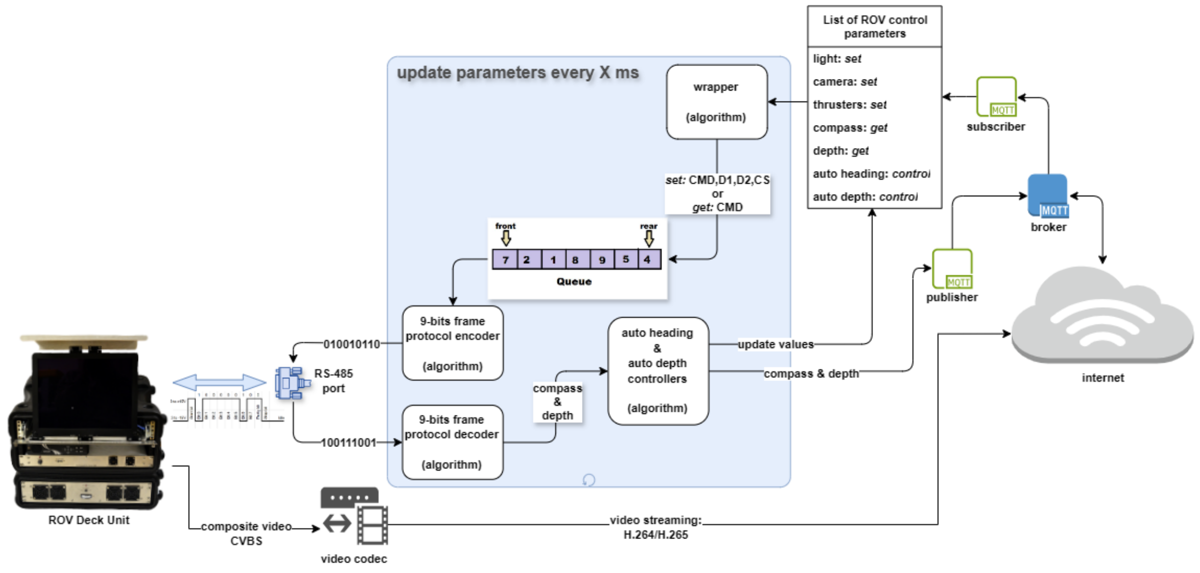


Fig. 5. Block diagram of the sub-system that is connected to the control console (deck unit). This connection is made by means of a Raspberry Pi 4 SBC.

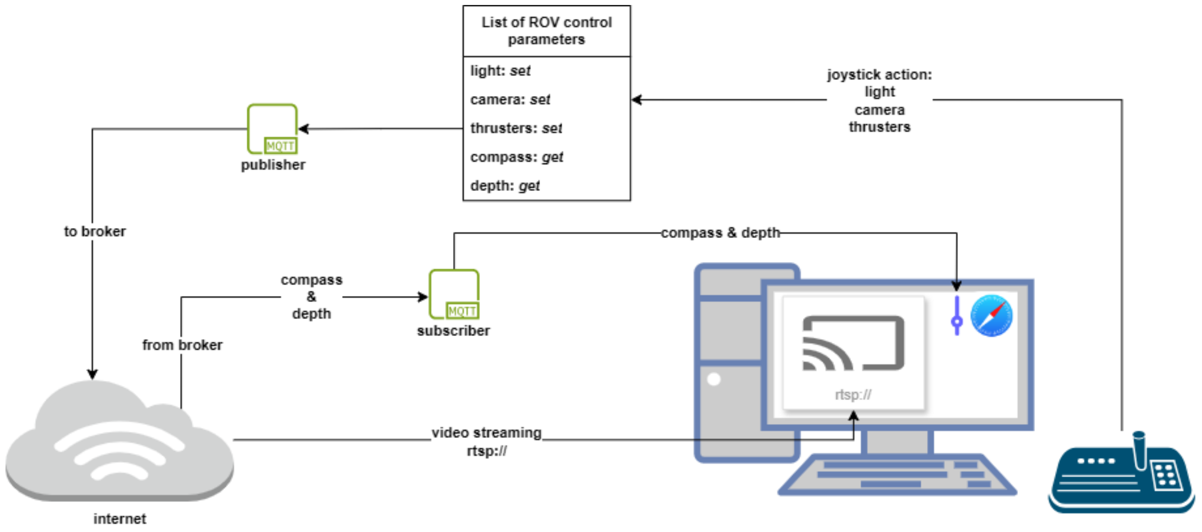


Fig. 6. Block diagram of the sub-system that is connected to the joystick. This side of the system will change the ROV states.

As in the development of the serial/ethernet converter using the ESP32 microcontroller, described previously, our efforts were concentrated on serial communication with the 9-bit protocol between the control console and the SBC. Since the Raspberry Pi Operating System (OS) is a Linux OS, its UART does not support the 9-bit protocol, so it was necessary to implement a code in C programming language that could emulate it. The implementation of this code was possible due to the possibility of changing the serial port configuration parameters dynamically. Thus, we could transmit/receive an address message by dynamically setting the parity bit to Mark or transmit/receive a data message by dynamically setting the parity bit to Space.

A second feature of this side of the remote piloting system is the architecture used to allow the exchange of information between the ROV control console and the remote station, where the joystick that controls the ROV is installed. Initially, we are using the Secure Shell (SSH) service for this message exchange, allowing us to control the ROV states in real time. However, to increase the system security, this message exchange will be performed via the Message Queuing Telemetry Transport (MQTT) protocol, as shown in Figure 5.

Finalizing the system development, Figure 6 presents the remote sub-system that is connected to the joystick and it is responsible for acting in the ROV states. A joystick action is transferred to the ROV via a network connection over SSH service or MQTT protocol.

Figure 7 illustrates an emulated control signal that the Raspberry Pi 4 sends to the ROV control console (red signal) and its respective response (blue signal).

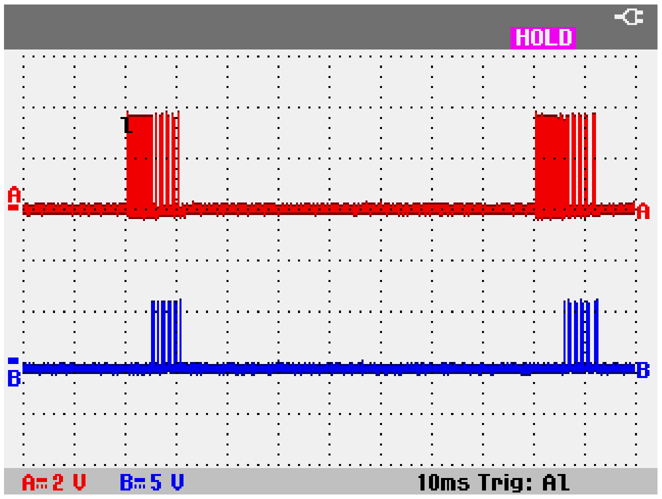


Fig. 7. An example of an RS-232 serial output signal from Raspberry Pi 4 to the control console.

III. Results

Figures 8-10 show the ROV being piloted remotely over a Wifi LAN connection by using the ESP32 Serial/Ethernet converter. In Figure 8, we can see an overview of the experiment. On the left side, there is the pilot and the client side Serial/Ethernet converter, in the center, there is the water tank with the ROV submerged, and on the right side, there is the server Serial/Ethernet converter and the control console.

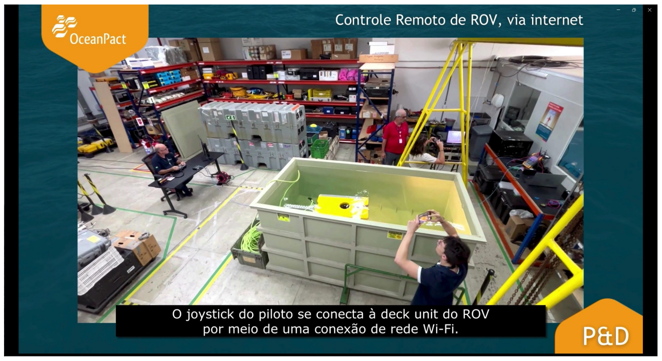

OceanPact
Controle Remoto de ROV, via internet
O joystick do piloto se conecta à deck unit do ROV
por meio de uma conexão de rede Wi-Fi.
P&D

Fig. 8. ESP32 controller: Overview of the experiment.

The server Serial/Ethernet converter and the ROV control console are shown in Figure 9. On the other hand, the pilot, the client Serial/Ethernet, the ROV joystick, and the ROV are shown in Figure 10.

It is worth noting that this experiment was carried out via a Wifi connection on a LAN. In this configuration latency and jitter were very low and the teleoperation task was easily executed. Subjectively there was no difference to the direct control version via cable connection. On the other hand, when we connected this system to a WAN via the Starlink network it was not possible to pilot the ROV. For this reason, a second approach was developed and tested.

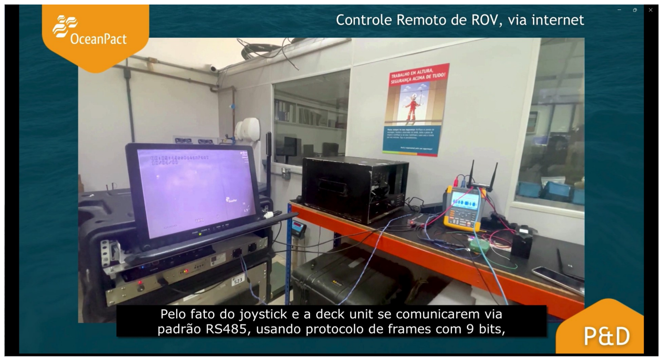

OceanPact
Controle Remoto de ROV, via internet
Pelo fato do joystick e a deck unit se comunicarem via
padrão RS485, usando protocolo de frames com 9 bits,
P&D

Fig. 9. ESP32 controller: ROV side view.

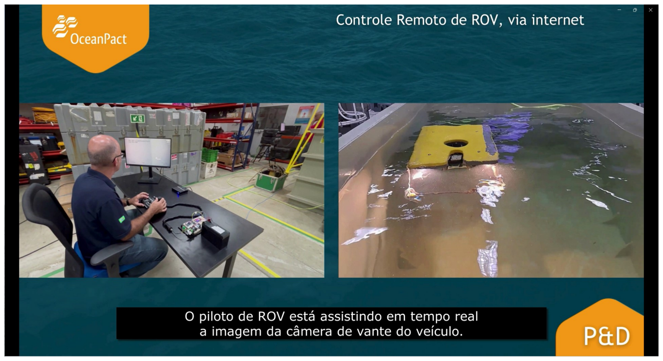

OceanPact
Controle Remoto de ROV, via internet
O piloto de ROV está assistindo em tempo real
a imagem da câmera de vante do veículo.
P&D

Fig. 10. ESP32 controller: Remote piloting view.

The second approach uses a Raspberry Pi 4 to emulate the ROV hand controller. It is connected directly to the ROV and sends commands to the control console periodically.

Figures 11-13 show the results of this second experiment. On the ground floor there is all the equipment on the ROV side, see Figure 11. On the upper floor, there are devices for the remote piloting of the ROV, see Figure 12.

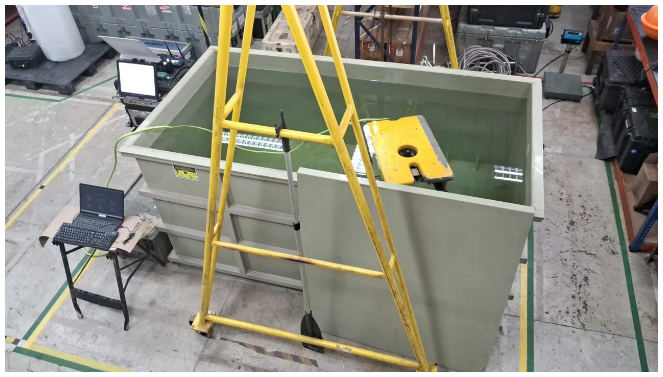


Fig. 11. Raspberry Pi: ROV side overview.

In Figure 13, we can see the Raspberry Pi 4 connected to a Serial converter (green enclosure) which is connected via RS-485 to the ROV control console. Furthermore, the Raspberry Pi 4 is also connected via cable to a Wifi router. This router sends and receives network packets to/from the remote station

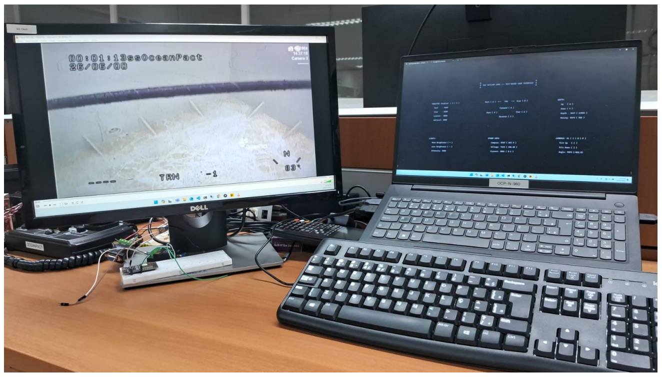


Fig. 12. Raspberry Pi: Remote piloting view.

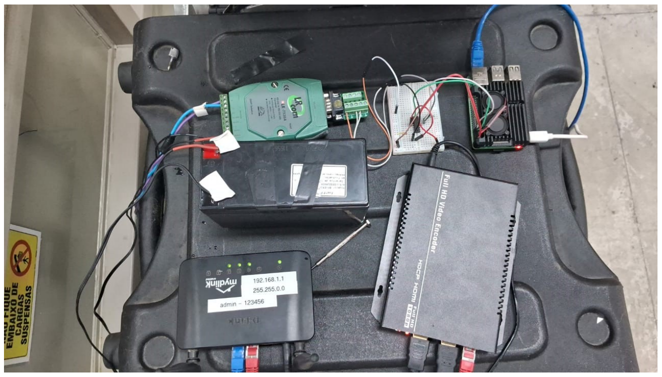
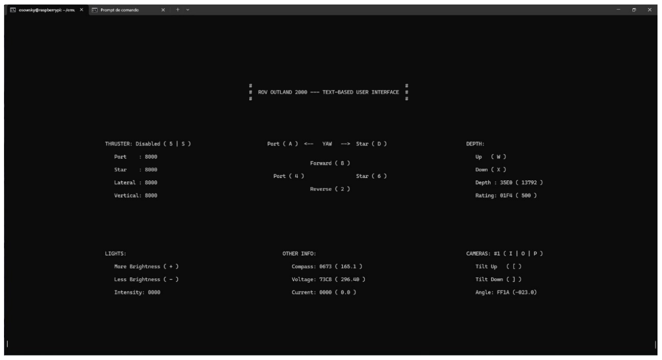


Fig. 13. Raspberry Pi: View of the electronic devices on the ROV side.



```
#                                               #
#  ROV OUTLAND 2000 --- TEXT-BASED USER INTERFACE  #
#                                               #

THRUSTER: Disabled ( 5 | S )        Port ( A ) <--   YAW   --> Star ( D )        DEPTH:
   Port     : 8000                          Forward ( 8 )                           Up   ( W )
   Star     : 8000                   Port ( 4 )          Star ( 6 )                 Down ( X )
   Lateral  : 8000                          Reverse ( 2 )                           Depth : 35E0 ( 13792 )
   Vertical: 8000                                                                   Rating: 01F4 ( 500 )

LIGHTS:                             OTHER INFO:                                  CAMERAS: #1 ( I | O | P )
   More Brightness ( + )               Compass: 0673 ( 165.1 )                      Tilt Up   ( [ )
   Less Brightness ( - )               Voltage: 73C8 ( 296.40 )                     Tilt Down ( ] )
   Intensity: 0000                     Current: 0000 ( 0.0 )                        Angle: FF1A (-023.0)
```

Fig. 14. Raspberry Pi: TUI on the remote control station side.

as illustrated in Figure 12. On the remote control station side there is a laptop that has a Text-based User Interface (TUI), see Figure 14, for sending commands to the ROV and a video stream from the ROV camera, as shown in Figure 15.

In this experiment, we can point out that the teleoperation performance was excellent, and the latency was in the order of milliseconds. All thrusters, lights, and cameras could be controlled via the laptop keyboard with greater accuracy than that obtained from the ROV joystick.

IV. Concluding Remarks

In this paper we have presented two methods to turn a commercial ROV into a Remotely Piloted ROV. The first

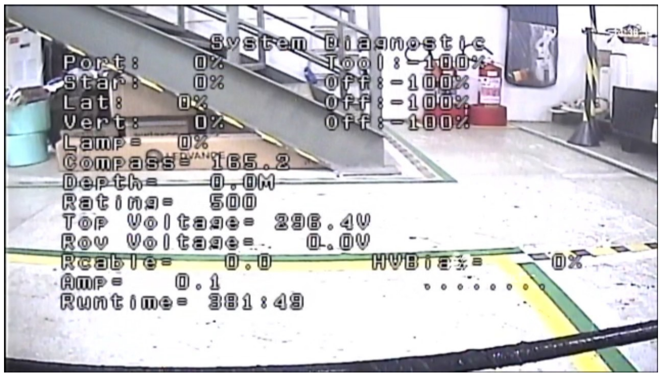

```
                  System Diagnostic
Port:      0%            Tool:-100%
Star:      0%            Off:-100%
Lat:      0%             Off:-100%
Vert:      0%            Off:-100%
Lamp=   0%
Compass=   165.2
Depth=     0.0M
Rating=   500
Top Voltage= 296.4V
Rov Voltage=   0.0V
Rcable=     0.0          HVBias=     0%
Amp=      0.1
Runtime=  381:49
```

Fig. 15. Raspberry Pi: Video stream from ROV camera.

one, which was based on a ESP32 Serial/Ethernet converter, achieved an excellent performance on a Wifi LAN connection. The second one, which was based on a Raspberry Pi 4 connected directly to the ROV control console to emulate its joystick, equally achieved a flawless performance on a Wifi LAN connection. Based on the results achieved using the Raspberry Pi 4, some steps are already being planned to allow the remote piloting of the ROV over a WAN network, which are: 1) implement the MQTT protocol for data exchange between the ROV and the remote control station; 2) reduce the video stream latency a jitter; 3) Implement a graphical user interface (GUI) connected to a generic joystick.

Acknowledgment

The authors would like to thank OceanPact Serviços Marítimos S.A. for funding the project development. Furthermore, the authors extend their gratitude to Outland Technology Inc. for providing technical information, which was instrumental in the project's development.

References

[1] *ROV-2000 System*, Doc. Number: 04-0024 Rev B1, Outland Technology, Inc, Louisiana: U.S.A, 2021.
[2] *User Manual OUTLAND ROV 1000, 2000 & 2500*, Doc. Number: 46-0002 Rev B1, Outland Technology, Inc, Louisiana: U.S.A, 2021.
[3] G. Bruzzone, R. Bono, M. Caccia, P. Coletta, and G. Veruggio, "Internet-based teleoperation of the Romeo ROV in the Arctic region", *IFAC Proceedings Volumes*, Vol. 36, no. 21, pp. 265–269, 2003.
[4] J. Gancet, D. Urbina, P. Letier, M. Ilzokvitz, P. Weiss, F. Gauch, G. Antonelli, G. Indiveri, G. Casalino, A. Birk, M. F. Pfingsthorn, S. Calinon, A. Tanwani, A. Turetta, C. Walen, and L. Guilpain, "Dexrov: Dexterous undersea inspection and maintenance in presence of communication latencies", *IFAC-PapersOnLine*, Vol. 48, no. 2, pp. 218-223, 2015.
[5] A. J. Dalpe, S. Suman, M. V. Jakuba, and A. Bowen, "Teleoperation of Remotely Operated Vehicles: Development, Challenges, and Future Directions", *OCEANS 2022*, VA, USA, 2022, pp. 1-7, doi: 10.1109/OCEANS47191.2022.9977341
[6] A. A. Dalhatu, A. M. Sa'ad, R. C. de Azevedo, and G. de Tomi, "Remotely Operated Vehicle Taxonomy and Emerging Methods of Inspection, Maintenance, and Repair Operations: An Overview and Outlook", *Journal of Offshore Mechanics and Arctic Engineering*, vol. 145, doi: 10.1115/1.4055476.