Article

Design and Construction of an ROV for Underwater Exploration

Oscar Adrian Aguirre-Castro [1], Everardo Inzunza-González [1,*], Enrique Efrén García-Guerrero [1], Esteban Tlelo-Cuautle [2], Oscar Roberto López-Bonilla [1], Jesús Everardo Olguín-Tiznado [1] and José Ricardo Cárdenas-Valdez [3]

1. UABC, Faculty of Engineering, Architecture and Design, Autonomous University of Baja California, Ensenada 22860, Baja California, Mexico; oscar.aguirre@uabc.edu.mx (O.A.A.-C.); eegarcia@uabc.edu.mx (E.E.G.-G.); olopez@uabc.edu.mx (O.R.L.-B.); jeol79@uabc.edu.mx (J.E.O.-T.)
2. INAOE, Department of Electronics, Puebla 72840, Mexico; etlelo@inaoep.mx
3. ITT, Department of Electrical and Electronic Engineering, Tijuana Institute of Technology, Tijuana 22500, Baja California, Mexico; jose.cardenas@tectijuana.edu.mx

* Correspondence: einzunza@uabc.edu.mx; Tel.: +52-646-1750744

Received: 24 October 2019; Accepted: 2 December 2019; Published: 6 December 2019

Abstract: The design of a remotely operated vehicle (ROV) with a size of 18.41 cm $\times$ 29.50 cm $\times$ 33.50 cm, and a weight of 15.64 kg, is introduced herein. The main goal is to capture underwater video by remote control communication in real time via Ethernet protocol. The ROV moves under the six brushless motors governed through a smart PID controller (Proportional + Integral + Derivative) and by using pulse-wide modulation with short pulses of 1 µs to improve the stability of the position in relation to the translational, ascent or descent, and rotational movements on three axes to capture images of 800 $\times$ 640 pixels on a video graphic array standard. The motion control, 3D position, temperature sensing, and video capture are performed at the same time, exploiting the four cores of the Raspberry Pi 3, using the threading library for parallel computing. In such a way, experimental results show that the video capture stage can process up to 42 frames per second on a Raspberry Pi 3. The remote control of the ROV is executed under a graphical user interface developed in Python, which is suitable for different operating systems, such as GNU/Linux, Windows, Android, and OS X. The proposed ROV can reach up to 100 m underwater, thus solving the issue of divers who can only reach 30 m depth. In addition, the proposed ROV can be useful in underwater applications such as surveillance, operations, maintenance, and measurement.

Keywords: ROV; underwater exploration; video capture; subsea inspection and intervention; marine robotics; underwater technology; real time; vision; complementary filter; Raspberry Pi

1. Introduction

Recently, several works on remotely operated vehicles (ROVs) have been reported for applications in ocean research [1–4]. In particular, ROVs have been used for underwater intervention, repair, and maintenance operations in offshore industries, including oil and gas industries, marine structures, marine sciences, naval defense, marine renewable energy, and scientific purposes [5–9]. Within submarine applications, recognition tasks stand out; for example, in [10–12] ROVs are employed for tracking mines and are programmed to carry out high-risk tasks, executing algorithms related to prediction, diagnosis, and classification. Other research efforts have been focused for underwater surveillance [13,14] and they are configured to manage continuous tasks with specific objectives. Some other research attempts have focused on the support in an optical fiber cable laying system on the seabed [15–17]. In [18,19], they are key to oceanographic research related to obtaining highly dynamic tidal data. In marine archeology, they are used for the exploration of underwater pools of toxic brine [20] and observation of underwater structures [21]. Regarding underwater communication, the authors of [22] proposed the use of a secure communication framework concealing the information under certain characteristics of pulses used by mammals. In addition, ROVs can be used to capture underwater images, of which nowadays there are open research lines [23–26]. However, despite the significant advances that have been achieved in the different areas of application and development of ROVs, it is a fertile field for research given the wide range of specialties that come into play in achieving optimal functionality. The precise sensing of underwater applications is a current topic in the research community. Efforts related to positioning a system based on GPS have been made; the characterization of the whole system in this case was done by means of a statistical study, considering different numbers of beacons [27]. Additionally, a novel programmable event-driven acoustic detector featuring audio pattern recognition for underwater deployment has been developed in [28]. Even a robust algorithm related to an autonomous underwater vehicle (AUV) was proposed to improve the supervised missions [29]. Currently, a real-time proposal of underwater acoustical imaging systems has been developed in [30], with the contribution of high-resolution underwater 3D acoustical imaging. Results based on implementation and experimental analysis of underwater stereo fusion and an algorithm for real-time 3D dense reconstruction with camera tracking have been reported [31]. In [32], the design and fabrication of an underwater remotely operated vehicle with single thruster configuration is reported. In [33], the design and implementation of an aquatic robot type ROV is described. The authors are focused on the design and implementation of an aquatic robot suitable for underwater exploration in order to improve its stability performance and achieving an efficient exploration system.

Nevertheless, aspects such as the ROV's autonomy, efficiency in energy consumption, processing and information storage, hydrodynamic structure, position control, and limitations in communication are some of the topics on which current research could contribute [34–37]. Another factor that can affect the ROV design and implementation is the high cost involved, but it is important to consider an adequate balance between the cost and the functionality of the system in relation to its application [38–41]. On this issue, one can take advantage of embedded systems, which are very useful for solving real-world problems for different fields of application. For example, the System on a Chip (SoC) Raspberry Pi has been used to solve different problems of practical applications, such as the accounting of underwater fish [42] as special case in a fish farm, or to perform complex tasks such as parallel computing [43].

In this paper, we propose the use of an SoC Raspberry Pi 3 as an onboard computer, since it can be programmed by using open source software, it is cheap, it has multiprocessing capabilities, easy scheduling tasks, and the programming language Python can be used. In addition, it is proposed to add a complementary filter to a smart PID (Proportional + Integral + Derivative) controller reported in [44]. Therefore, experimental results show that the use of the complementary filter to the output signals of inertial measurement unit (IMU) helps to improve the performance of the smart PID controller used for the trajectory control, allowing to capture better quality of underwater images. Furthermore, experimental results show that the proposed ROV can process up to 42 frames per second for underwater video capture, due to its capability of performing parallel processing taking advantage of the threading library.

The rest of this paper is organized as follows: Section 2 describes the development, hardware organization, complementary filter, smart PID controller, and mechanical design of the proposed ROV. Section 3 describes the main algorithms performed by the computer onboard the ROV: (i) remote communication; (ii) motor control and temperature measurement; (iii) 3D position sensing; and (iv) video capture in real-time. Section 4 presents the experimental results, such as ROV performance, some images captured in real-world underwater environments, and a comparison versus commercial ROVs. Finally, Section 5 summarizes the conclusions of the paper.

2. Design and Construction of the ROV

2.1. Electronic Design

Figure 1 shows the block diagram of the proposed hardware that consists of two parts: the remotely controlled vehicle and the remote control. These two subsystems are communicated through an Ethernet cable with Kevlar reinforcement, so that the user can control the ROV from a personal computer, laptop or SoC, all with Virtual Network Computing (VNC) connectivity. The ROV hardware involves a SoC Raspberry Pi 3 that is responsible for executing parallel tasks, such as: (i) acquisition of video by means of a digital camera; (ii) measurement and recording of the different variables associated with the sensors of the system; (iii) motors control in coordination with an Arduino Nano microcontroller; (iv) battery voltage monitoring for energy consumption and internal temperature; and (v) communication management with the remote control. The used digital camera is a Vemont full HD 1080 p of 12 Megapixels, which is built-in with an IP68 waterproof case, *allowing to dive underwater*. Besides, two power sources are integrated into the system, one with a power bank of 5 V with 10,000 mAh for the Raspberry Pi 3, an Arduino Nano microcontroller, and digital sensors, and the other one is a bank of six batteries of 11.1 V with 19,800 mAh to power six brushless motors. With these battery banks, the ROV has an autonomy of up to 2 h. The remote control shown in Figure 1 consists of an Ethernet network hub (Ethernet switch/router) interconnected by an Ethernet network cable to a computer or SoC, from which the user controls the ROV through a touchscreen type graphical user interface (GUI).

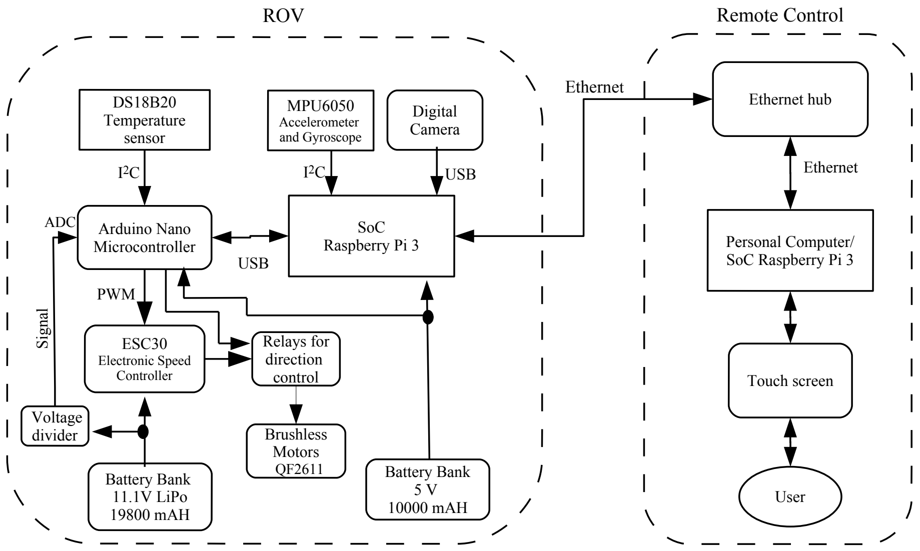

ROV
Remote Control
DS18B20 Temperature sensor
MPU6050 Accelerometer and Gyroscope
Digital Camera
I²C
I²C
USB
ADC
Arduino Nano Microcontroller
USB
SoC Raspberry Pi 3
PWM
Signal
ESC30 Electronic Speed Controller
Relays for direction control
Voltage divider
Brushless Motors QF2611
Battery Bank 11.1V LiPo 19800 mAH
Battery Bank 5 V 10000 mAH
Ethernet
Ethernet hub
Ethernet
Personal Computer/ SoC Raspberry Pi 3
Touch screen
User

Figure 1. Block diagram of the proposed ROV.

The brushless motors are model QF-2611 manufactured by XCSOURCE with a turning speed of 4500 KV three-phase voltage, are connected in a star configuration. The brushless motors are controlled by an electronic speed controller (ESC), which generates a three-phase output signal, having as an input a pulse width modulation (PWM) signal generated by an Arduino Nano microcontroller. Based on the experiments, it was found that 1 µs is the most efficient time to provide the best ROV stability underwater for video capture. The ESC30A delivers up to 30 A.

The speed controllers are unidirectional, and the changes in their rotation directions are shown in Figure 2. The circuit includes a pair of relays per motor, so that two phases reach the motor's terminals, generating the change in the direction of rotation of the ROV according to the operation requirements.

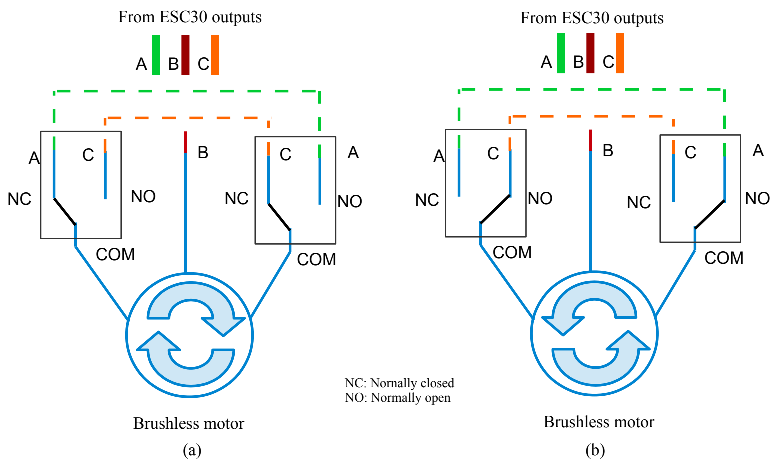


Figure 2. Electric circuit to invert the rotation direction of the motors: (**a**) turning the motor clockwise; and (**b**) turning the motor counterclockwise.

Regarding Figure 1, the ROV incorporates the digital sensor DS18B20 for measuring the temperature inside its capsule, which is communicated through the I^2C protocol directly to the Arduino Nano microcontroller.

Due to the fact that at several meters underwater there is no line of sight between the remote control and the ROV, it is necessary to measure the precise position and orientation. That is why, knowing the 3D angular position of the ROV serves to automatically control its movement. This is done by using an MPU6050 sensor, which has six degrees of freedom. This means that there are three accelerometers and three gyroscopes inside the MPU6050. Figure 3 shows the measurements of one of the signals achieved directly at the output of the MPU6050. It can be appreciated that in both conditions of movement and static there exists the presence of small variations (noise vibrations), i.e., the forces driving the ROV are also perceptible in the sensor, which affect the stability of the ROV's movements. To reduce these variations, a complementary filter [45–47] is used to filter undesired effects and therefore the Raspberry Pi 3 is used to process the MPU6050 data. In the complementary filter, the cutoff-frequency of the low-pass filter is designed considering the accelerometer measurements, while for the high-pass filter, its design depends on the signal measured by the gyroscope. The schematic of the complementary filter is shown in Figure 4.

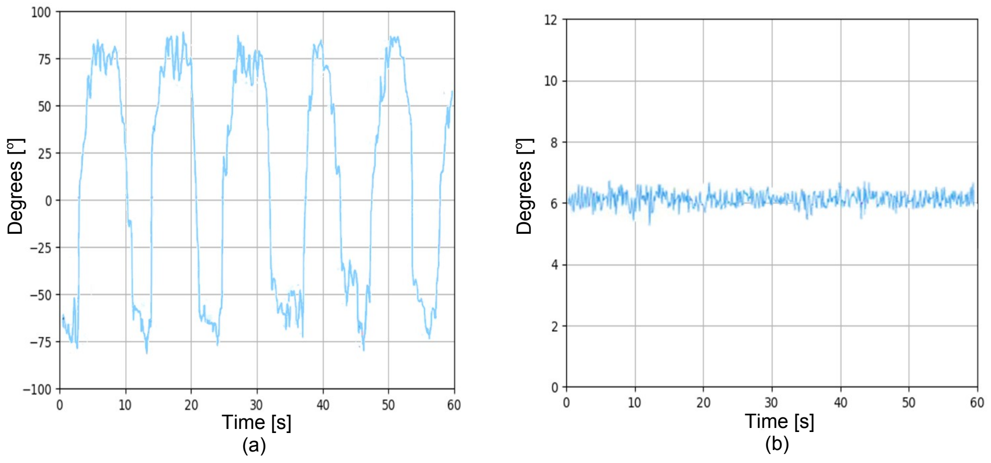


Figure 3. Output signal of the MPU6050 sensor without conditioning: (**a**) ROV in motion with direction in *y*; and (**b**) ROV without movements (parked).

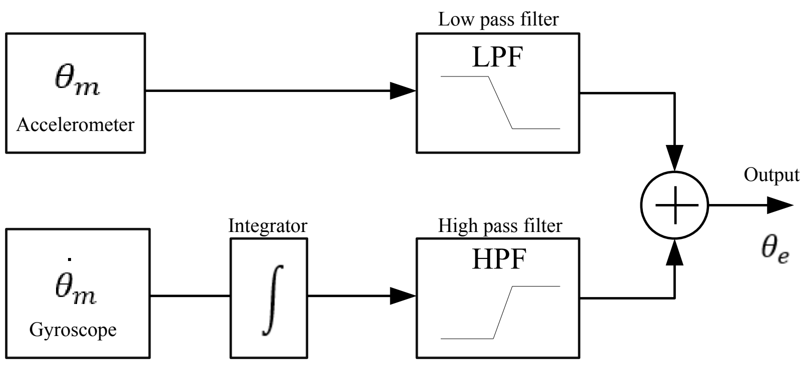


Figure 4. Scheme of the complementary filter [45–47] implemented on the ROV.

By mixing data from the accelerometer and gyroscope, the next transfer function for the complementary filter is obtained [45–47],

$$\theta_e(z) = \frac{\omega_{c1}}{(1-z^{-1})/dt + \omega_{c1}}\theta_m(z) + \frac{dt}{1-z^{-1}} \cdot \frac{(1-z^{-1})/dt}{(1-z^{-1})/dt + \omega_{c2}}\dot{\theta}_m(z), \tag{1}$$

where θ_m is the angular data measured with the accelerometer in degrees [°], $\dot{\theta}_m$ is the angular velocity data measured with the gyroscope [°/s], θ_e is the output of the complementary filter in degrees [°], $\omega_{c1} = 5$ Hz is the cutoff frequency of the low pass filter, $\omega_{c2} = 2$ Hz is the cutoff frequency of the high pass filter, and dt is the differential time [s]. By simplifying and rearranging the transfer function, the following is obtained:

$$\theta_e(z) \cdot [(1-z^{-1})/dt + \omega_c] = \omega_c \cdot \theta_m(z) + \dot{\theta_m}(z), \tag{2}$$

by applying the inverse Z transform,

$$\theta_e(k) \cdot (1/dt + \omega_c) = \theta_e(k-1)/dt + \omega_c \cdot \theta_m(k) + \dot{\theta_m}(k), \tag{3}$$

therefore,

$$\theta_e(k) = (1-\alpha)\cdot[\theta_e(k-1)+\dot{\theta_m}(k)\cdot dt]+\alpha\cdot\theta_m(k), \tag{4}$$

where $\alpha = 0.02$ and k is the sample time.

By implementing the complementary filter, greater ROV handling stability is obtained, as shown in Figure 5. It can be appreciated that, when the complementary filter is used, the ROV gains better stability, because the noise is significantly reduced. Hence, the above contributes in a quality improvement of captured images.

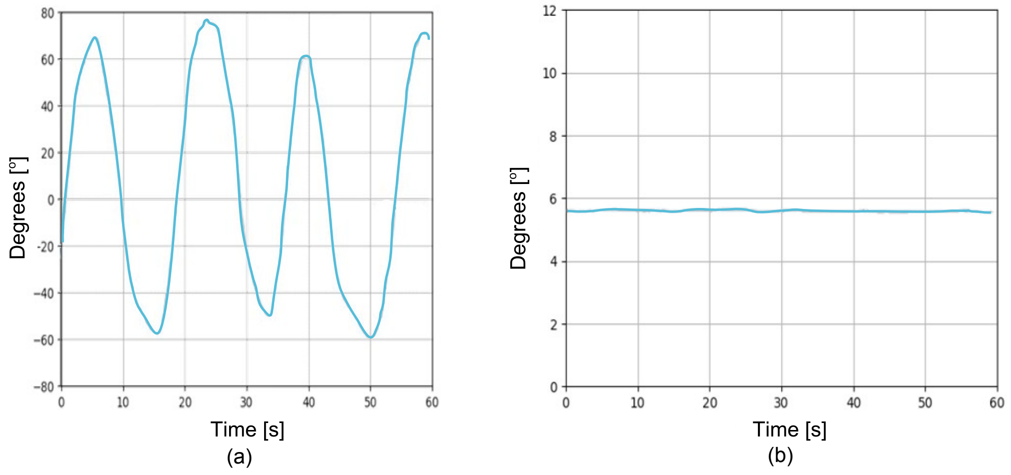


Figure 5. Motion sensor signal by using complementary filter: (**a**) ROV in motion with direction in y; and (**b**) ROV parked.

Control and Actuation Subsystems

It is well known that ROVs are subject to parametric changes (weight, buoyancy, added mass, payload, etc.) or external disturbances such as ocean currents; thus, the control subsystem used in the proposed ROV is inspired by the smart PID controller reported by Hernández-Alvarado et al. [44], which has the advantage of online tuning the gains of the PID controller as it is based on an artificial neural network (ANN). However, in this paper, it is proposed to add a complementary filter on the MPU6050 sensor output to improve PID controller performance and ROV stability, with the purpose of improving the quality of underwater images.

In the domain of discrete time, the digital PID algorithm can be expressed as [44]:

$$\tau(k) = \tau(k-1)+K_p(e(k)-e(k-1))+K_ie(k)+K_d(e(k)-2e(k-1)+e(k-2)), \tag{5}$$

where $\tau(k)$ is the original control signal, $e(k) = \eta_d(k)-\eta_f(k)$ represents the position tracking error, $\eta_d(k)$ denotes the desired trajectory, $\eta(k)$ is the real trajectory without processing, $\eta_f(k)$ are the filtered data of real trajectory, K_p is the proportional gain, K_i the integral gain, K_d is the derivative gain, and k is the sample time. Figure 6, depicts the block diagram of self-tuning PID controller based on ANN [44]. It can be observed that the use of the complementary filter described in Figure 4 reduces noise levels at the output of the MPU 6050 sensor. The stage of the actuation subsystem, as depicted in Figure 1, is integrated with an Arduino Nano microcontroller, the power stage with six ESC30A speed controllers, twelve relays for controlling the rotation direction of thrusters (Figure 2), and six QF2611 thrusters/motors with propellers.

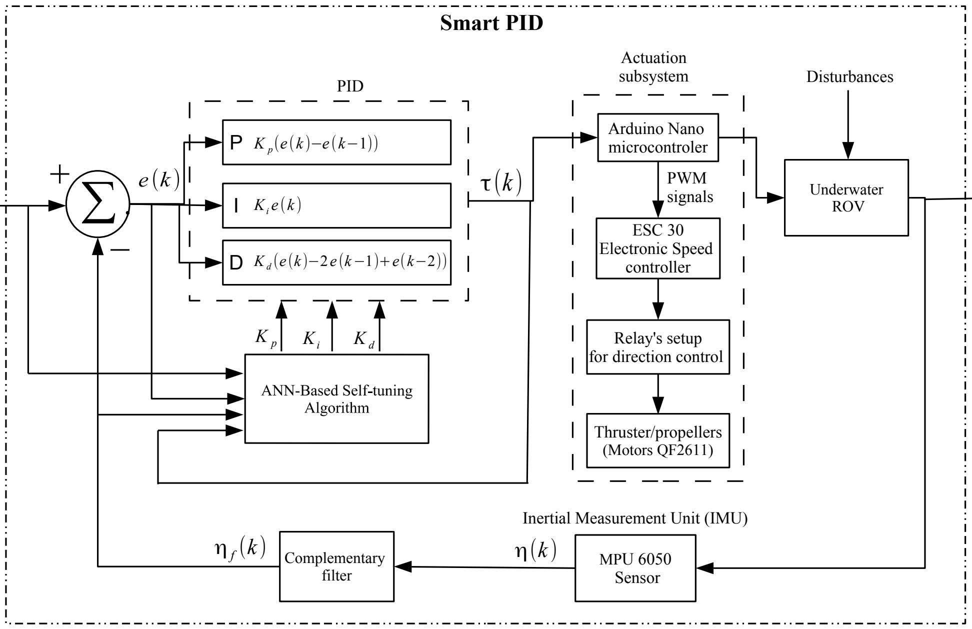


Figure 6. Block diagram of smart PID controller with actuation subsystem and complementary filter.

Figure 7 shows a simulation of smart PID controller's response to unit step input during a time lapse of 50 ms. It can be observed that, when an abrupt change is produced in the input, its stabilization response time is less than 10 ms.

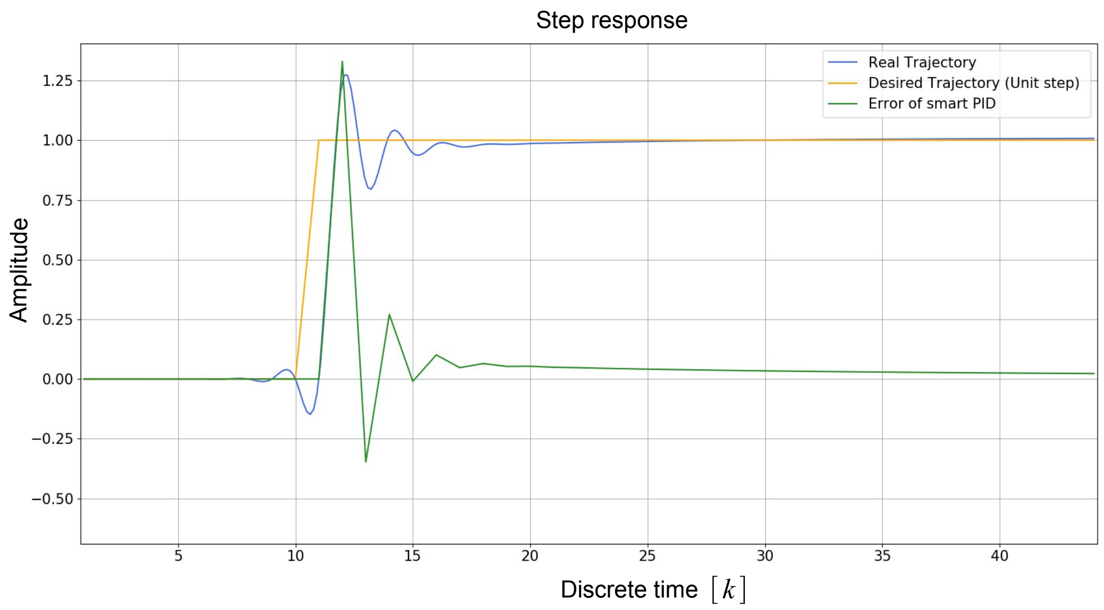


Figure 7. Simulation of smart PID controller's response to unit step input.

2.2. Mechanical Design

The physical characteristics of materials that were used in designing the ROV to operate in underwater conditions can be seen in [48]. In this work, polyvinyl chloride (PVC pipe with steel bars inside) is used as a structural element since it complies with mechanical characteristics for the ROV to function properly, and determines the maximum depth that can be reached. The depth is calculated by Equation (6); it requires knowing the fracture point of the material provided by the manufacturer, which in this case corresponds to a pressure of 1724 kPa. Under these conditions, the maximum depth for a PVC pipe with schedule 40 is $h = 125.65$ m.

$$P = \rho \cdot g \cdot h, \tag{6}$$

where P is pressure, $\rho = 1400\ \text{kg/m}^3$ is seawater density, g is gravity, and h is depth in meters. The 3D structural design of the proposed ROV is shown in Figure 8. The design takes into account aspects of hydrostatic and buoyancy depending on the application environment, either a marine environment (saltwater) or under controlled conditions (fresh water). The structure presented considers a weight balance configuration to facilitate the implementation in the topology of the motors, and it was developed in the 3D figure simulator software known as SolidWorks. Its physical dimensions are 18.41 cm $\times$ 29.50 cm $\times$ 33.50 cm, with an estimated volume of $V = 18.19 \times 10^{-3}\ \text{m}^3$ and an estimated weight of $W = 15.64$ kg. Based on these data and considering a saltwater density of $\rho = 1400\ \text{kg/m}^3$, the buoyancy corresponds to 249.6 N, which can be obtained by Equation (7), where E is the total thrust, ρ is the density of the fluid, g is the gravity, and V is the volume. Therefore, the proposed topology for the six motors in the ROV considers a minimum thrust force of 9.82 kg.

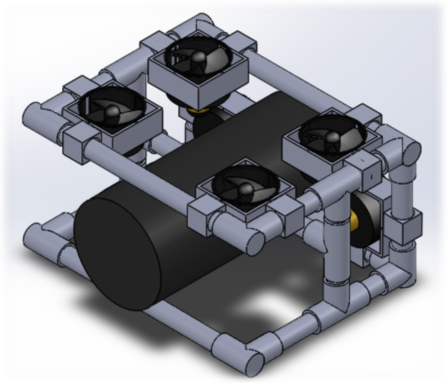


Figure 8. 3D design view of the proposed ROV.

$$E = -\rho \cdot g \cdot V, \tag{7}$$

Figure 9 shows the locations of the six brushless motors: four are located on the top for the ascent and descent movements and two on the front for translational and rotational movements.

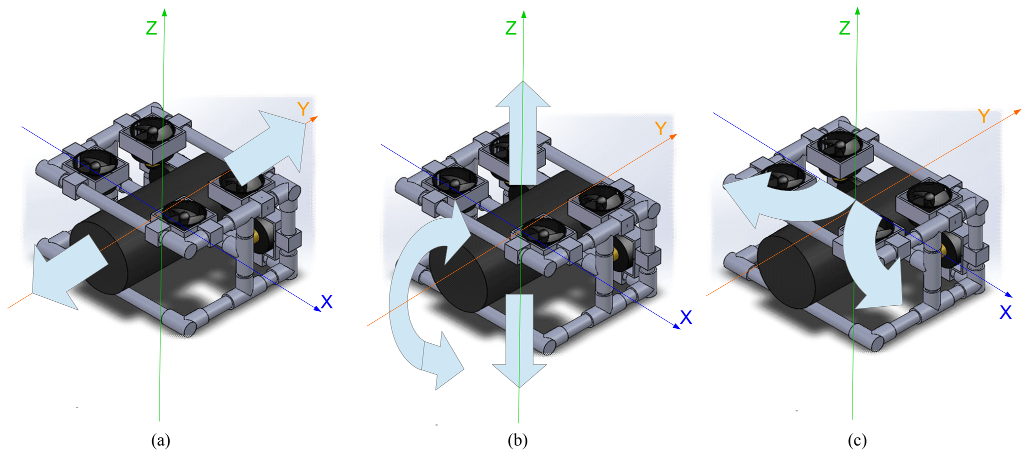


Figure 9. Movements of the ROV: (**a**) translational; (**b**) ascent and descent; and (**c**) rotation.

According to Figure 8, Table 1 shows the materials used in the construction of the mechanical structure of the ROV. The designed mechanical structure allows the placement of the six brushless motors, as can be seen in Figure 9. Table 2 shows the components with their estimated weights that give the total weight of 15.64 kg. To achieve mechanical stability within the water and operate with the buoyancy estimated by Equation (7), eight steel bars are incorporated into the structure. The final design of the ROV structure allows placing the camera in different positions, i.e. it can be placed in front, under, or on the sides of the ROV.

Table 1. Materials for the construction of the ROV structure.

Quantity (Pieces)	Description	Size [cm]
4	PVC pipe	$\phi 1.27 \times 25$
4	Corner connector	$\phi 1.27$
2	PVC pipe	$\phi 1.27 \times 11.5$
5	T-connector	$\phi 1.27$
4	PVC pipe	$\phi 1.27 \times 5$
4	L-connector	$\phi 1.27$
1	PVC pipe	$\phi 10.16 \times 25$

Table 2. Weights of the ROV's materials.

Item	Quantity	Weight [kg]	Total Weight [kg]
PVC structure	1	6.940	6.940
Battery 11.1 V	6	0.200	1.200
Battery 5.0 V	1	0.200	0.200
Raspberry Pi	1	0.100	0.100
Arduino Nano	1	0.001	0.010
Relays	6	0.150	0.900
ESC30A	6	0.050	0.300
Brusless motor	6	0.225	1.350
Steel bars	8	0.580	4.640
		Total weight	15.64

3. ROV Algorithms

Figure 10 shows the main algorithms executed by the SoC Raspberry Pi 3 that is used as the ROV's onboard computer, which are divided into four parts: (i) main control of the whole structure and communication (**ROV_Parallel_computing**); (ii) motors control and temperature measurement **(Motors_Control_and_Temperature)**; (iii) 3D position sensing **(Acquire_3D_Position)**; and (iv) video capture **(Video_capture)**. For the programming and execution of these algorithms, Python software version 2.7.9 was used.

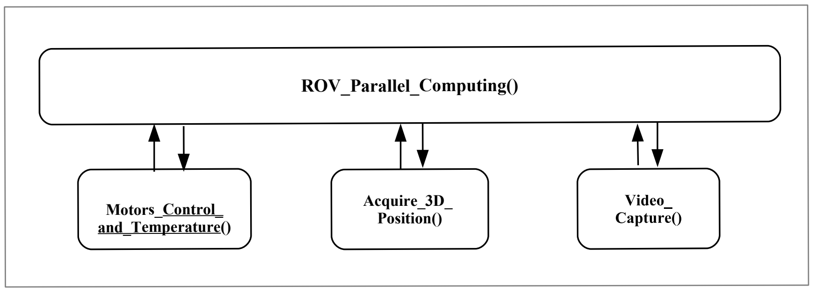


Figure 10. Algorithms executed on the Raspberry Pi 3 performing parallel computing.

Furthermore, the Raspberry Pi 3 SoC contains four cores inside its microprocessor with a 1 GHz clock, so each algorithm is mapped to each core for execution. Figure 11 depicts an outline of the multicore organization. The communication between the ROV and the remote control is via Ethernet and can be controlled remotely with different computer hardware or SoC, allowing the use of different operating systems in the computer connected to the Ethernet network hub, as described in Figure 1. The Raspberry Pi 3 works with the GNU/Linux operating system Raspbian distribution, and the remote computer/SoC works using the VNC protocol, which is compatible with Windows, Android, OS X, or GNU/Linux.

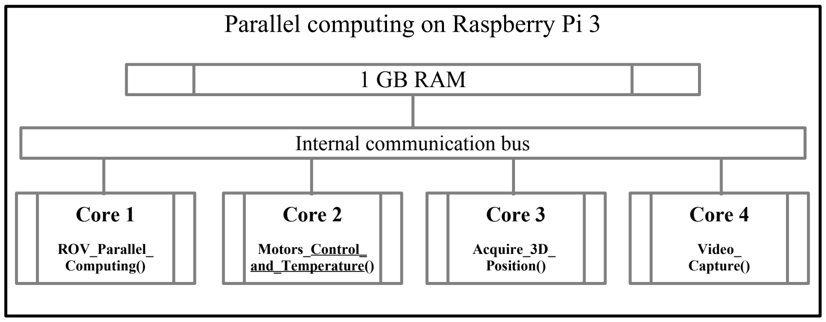

Parallel computing on Raspberry Pi 3
1 GB RAM
Internal communication bus
Core 1
ROV_Parallel_ Computing()
Core 2
Motors_Control_ and_Temperature()
Core 3
Acquire_3D_ Position()
Core 4
Video_ Capture()

Figure 11. Block Diagram of multicore SoC Raspberry Pi 3.

The code fragment shown in Algorithm 1 describes the main algorithms executed on the SoC Raspberry Pi 3. On Line 1–5, the sys, Tkinter, cv2, threading, and serial libraries are included. The sys library is included in the ROV application; commands from the GNU/Linux operating system can be executed. The Tkinter Library is used for the development of graphical user interface. The cv2 library contains subroutines for image processing, such as acquisition, display, etc. The threading library is used to execute the algorithms shown in Figure 10 in parallel. The serial library is used to manage the serial communication between Raspberry Pi and the Arduino Nano microcontroller. On Line 6, the main function **ROV_Parallel_Computing()** is declared. On lines 7–25, the function **Motors_Control_and_Temperature()** is declared, which is used to control the motors for the movement of the ROV by using the smart PID controller, as depicted in Figure 6, and to measure the internal temperature by using the DS18B20 sensor. On lines 26–30, the function **Acquire_3D_Position()** is declared, which is used to check the 3D position of the ROV by means of the MPU 6050 sensor. On Lines 31–45, the **Video_Capture()** function is declared, which is used to capture video in real time through the camera. On Lines 46–52, the code is used to setup the serial communication between Raspberry Pi and the Arduino Nano microcontroller. On Lines 53–55, the program sends a message in the case of a connection error and to exit the ROV application. Finally, Lines 57–64 execute the ROV's algorithms by using parallelism with threads. The function **Motors_Control_and_Temperature()** controls the movements required by the ROV via the smart PID controller, such as those shown in Figures 6 and 9, which are carried out by an embedded algorithm developed for the Raspberry Pi 3, and it communicates with the Arduino Nano microcontroller to generate the PWM signals that will control the motors's speed, which is composed of four algorithms: (i) calibration of the speed controllers; (ii) adjustment of the direction of rotation through relays; (iii) speed adjustment through a signal using pulse width modulation (PWM) with steps of 1 μs; and (iv) motor control via the smart PID controller. This function also measures the capsule's interior temperature where the ROV electronics are located. Regarding the function **Acquire_3D_Position()**, the acquisition of the ROV position is performed by the SoC Raspberry Pi, providing information on the acceleration about inclination angles and angular velocity data. It also provides six-axis digital information, uses communication with I^2C protocol, and operates at 3.3 V. The code fragment presents the three algorithms of execution: (i) declaration of the communication port; (ii) acquisition of data from the acceleration and rotation; and (iii) angle filtering.

Since the algorithms are programmed using open software tools, the whole system allows the addition of more sensors and functionalities to the ROV, according to the operational needs, maintenance, supervision, and sense of physical-chemical variables underwater.

Algorithm 1 Executing the ROV's algorithms by performing parallel computing.

```
import sys                  # System parameters and functions
from Tkinter import   # Library for GUI
import cv2                  # Library for image processing
import threading         # Library for parallel computing by threats
import serial               # Library for serial communication
 def ROV_Parallel_Computing ( )
        def Motors_Control_and_Temperature( )  # Function for motor control and temperature sensing.
             switch( PushButton ( ) ) # Configuration of the motions
                  case 1:
                       def Config_UP ( )
                  case 2:
                       def Config_DOWN ( )
                  case 3:
                       def Config_LEFT ( )
                  case 4:
                       def Config_RIGHT ( )
                  case 5:
                       def Config_STOP ( )
                  case 6:
                       def Config_CALIBRATE ( )
               if(Serial.read == TEMP) # Read temperature
                    Read Temperature
                    Print Temperature.
                    Store Temperature.
               endif
        def Acquire3D_Position( )      # Read 3D position
             def Read_Acl( )            # Read the accelerometer data
             def Read_Gyr( )            # Read the gyroscope data
             def filter_Comp( )    # Filtering accelerometer and gyro.
             print( 'positon' )
        def Video_Capture( ):
             print ("Camera")
             cap=cv2.VideoCapture(0)           # Video capture.
             while (1)
                  ret, FPS = cap.read().
                  cv2.imshow('ROV CAM',frame)
                  CPUTEMP = Read_CPU_temp.
                  print (FPS, CPUTEMP)
                  key= cv2.waitKey(1)
                  if key == 27:
                            break
             endwhile
             cap.release.
             cv2.destroyallWindows.
        ROV_Parallel_Computing.mainloop()
 try:               # Setup communication with Arduino Nano Microcontroller.
                       ard = serial.Serial()
                       ard.port = '/dev/ttyUSB0'.
                       ard.baudrate = 9600.
                       ard.timeout = 3.
                       ard.open().
                       print('Wait a moment....').
except:.
                       print('Connection Error')
                       sys.exit().
# Parallel algorithms execution.
# Assign a thread for parallel execution.
CAMERA=threading.Thread(target = Video_Capture )
CAMERA.start(). .
POSITION=threading.Thread(target = Acquire_3D_Position )
POSITION.start().
MOTORS=threading.Thread(target = Motors_Control_and_Temperature )
MOTORS.start(). .
APP=threading.Thread(target = ROV_Parallel_Computing)
APP.start()
```

4. Experimental Results

The experimental results using the Raspberry Pi 3 as the ROV's onboard computer are summarized herein, as well as the hardware resources used by each algorithm and other processes running simultaneously. The captured images in a controlled aquatic environment, as well as obtained images in the sea environment are also shown. In addition, the results obtained in the implementation of the smart PID controller and a comparison of the main features versus two commercial ROV are presented.

4.1. ROV Performance

4.1.1. Motors Test

The Arduino Nano microcontroller and Ardu-Pilot development boards were used for the brushless motors speed and operation tests to identify which one is more efficient for the generation of the PWM signals required by the ESC30A. We found that the change made by the Ardu-Pilot flight controller was quite sudden for the brushless motors used in this paper. Due to this limitation, a program was implemented on the Arduino Nano microcontroller to generate higher resolution PWM signals for the speed changes. Based on the experimentation, it was found that 1 μs is the most efficient time and that it provides the best ROV stability under water. Figure 12 shows an example of PWM signals generated by the Arduino Nano microcontroller for the control of brushless motors. Figure 12a depicts the PWM signal of average speed, Figure 12b shows the PWM signal of maximum speed, Figure 12c depicts the PWM signal of minimum speed and the corresponding output signal of ESC30A controller, and Figure 12d depicts the PWM signal of maximum speed and output signal of ESC30A controller.

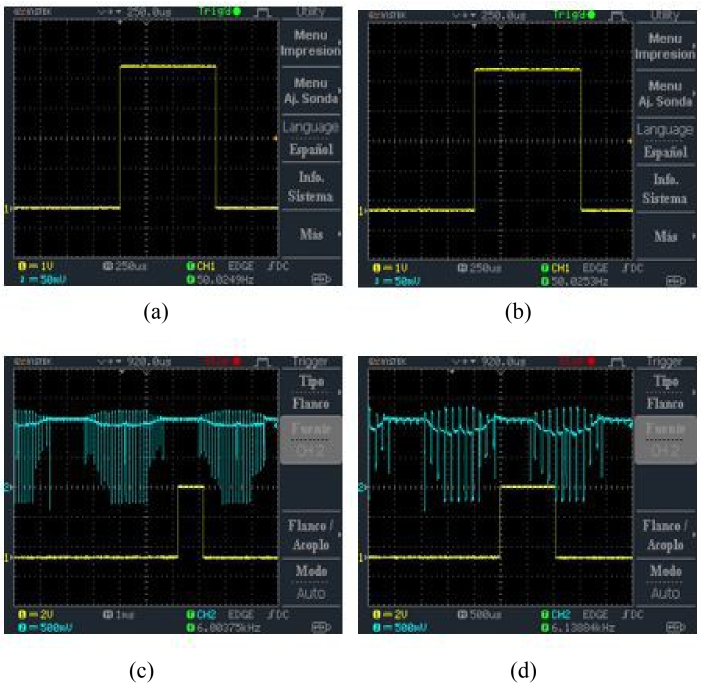


Figure 12. PWM signals generated by the Arduino Nano microcontroller: (**a**) PWM signal of average speed; (**b**) PWM signal of maximum speed; (**c**) PWM signal of minimum speed and output signal of ESC30A controller; and (**d**) PWM signal of maximum speed and output signal of ESC30A controller.

4.1.2. Temperature in SoC Raspberry Pi 3 and ROV Capsule

Figure 1 shows that the ROV incorporates the DS18B20 digital sensor for measuring the temperature inside the capsule, which is communicated through the I^2C protocol directly to the Arduino Nano microcontroller. An experimental behavior analysis of the internal temperature in the ROV capsule was required due to the heating produced by the ESC30 when operating the motors. For these measurements, an embedded algorithm on the Arduino Nano microcontroller was employed, obtaining temperature data from the DS18B20 sensor. Additionally, an algorithm was implemented in the SoC Raspberry Pi 3 to measure the temperature of its chipset, given that the maximum operating temperature must not exceed 65 °C. Figure 13a shows the temperature behavior from the Raspberry Pi 3 chipset. Figure 13b shows the temperature inside the ROV capsule. The figure shows that, when the motors are turned on at their maximum speed, the temperature of the Raspberry Pi3 chipset increases, reaching 54 °C in less than 1 h, while the temperature of the ROV capsule rises up to 37 °C. On the other hand, when the motors are turned off, the chipset and capsule temperature decrease over time.

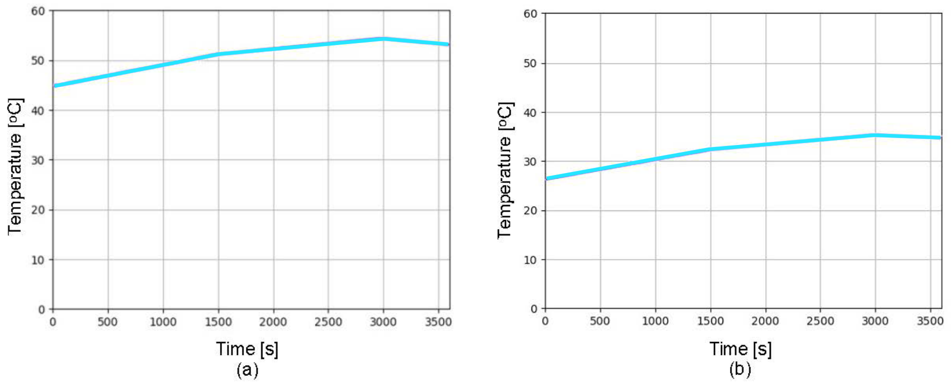


Figure 13. Temperature measurement of every second over a period of one hour: (**a**) temperature of the chipset Raspberry Pi 3; and (**b**) temperature in the ROV capsule.

4.1.3. Battery Banks Performance

The performance of the battery banks under ROV operation is shown in Figure 14. The behavior of the 5 V battery bank supplying power to the digital components is shown in Figure 14a, where it is observed that it has a work time of 500 min (8.33 h) continuous, executing the algorithms that control the ROV. Meanwhile, the battery bank of 11.1 V, which energizes the ESC30A for the control of the brushless motors has a time duration of 120 min (2 h) at medium speed, as shown in Figure 14b. However, this time may increase or decrease according to the navigation of the ROV.

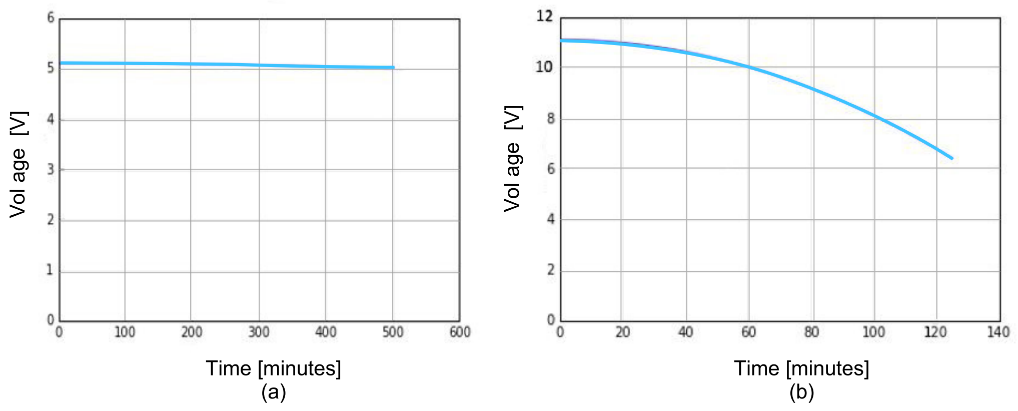


Figure 14. Behavior of the 5 V and 11.1 V battery banks during the operation of the ROV: (**a**) 5 V with 10,000 mAh battery bank for digital devices; and (**b**) 11.1 V with 19,800 mAh battery bank for brushless motors.

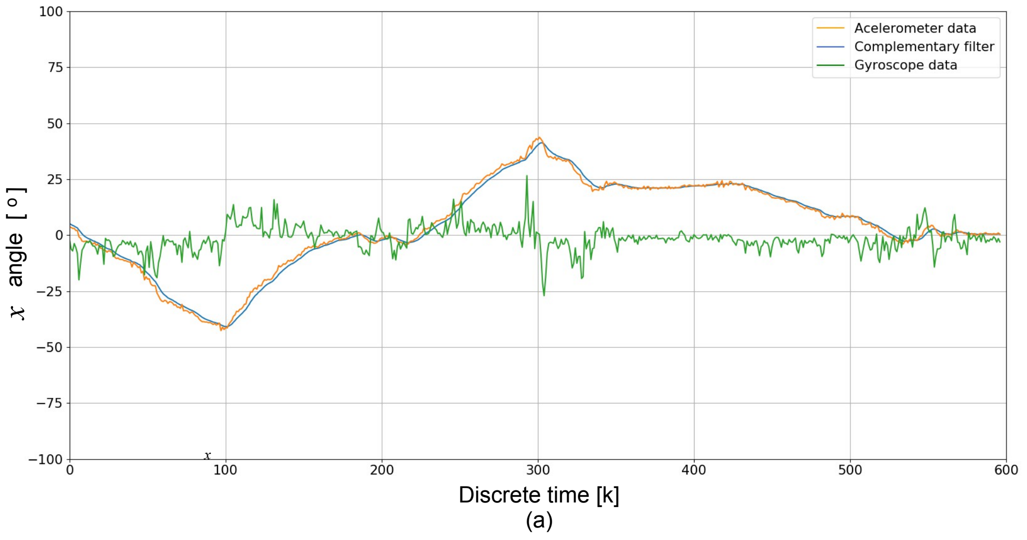
4.1.4. Stability Performance

The MPU6050 sensor is responsible for colorredkeeping track of the orientation of the ROV and is very useful to stabilize it. The complementary filter gives us a smoothed signal from both signals (accelerometer and gyroscope). In the short term, we used the data from the gyroscope, because they were very precise and not susceptible to external forces. In the long term, we used the data from the accelerometer, as they did not drift. In this experiment, the used sensitivity for the accelerometer was $16,384$ LSB/g and for the gyroscope was 131°/s. Figure 15 depicts the stability tests on the output of the complementary filter. Figure 15a shows movements on x-axis, while Figure 15b shows movements on y-axis. It can be observed that no noisy accelerometer and gyroscope data were detected, i.e., they did not drift away. In addition, the complementary filter helped improve the stability of the ROV and the quality of the captured images was improved. Finally, the complementary filter is easy and light to implement on embedded systems such as Raspberry Pi.

Figure 16 depicts the smart PID controller's response. Figure 16a,b shows the angle in x-direction and y-direction, respectively. It can be observed that the real trajectory followed the desired trajectory. In the control of both trajectories, it is appreciated that the error was close to zero, and the signals were smoothed as a result of the implementation of the complementary filter. Therefore, this helps to improve the stability of the ROV and consequently the quality of the images captured underwater.

4.2. Hardware Resources

Figure 17 shows the used resources by all the algorithms and other processes running on Raspberry Pi 3 SoC as the onboard computer of the ROV. It can be observed that the process labeled PID 3286, in the left column, which corresponds to the main algorithm labeled as ROV_Parallel_Computing in Figure 10, used 74.8% of the CPU resources, and 13.9% of RAM memory. The PID 3239 that corresponds to Video_Capture algorithm used 38.7% of the CPU resources and 11.0% of RAM memory. The PID 2645 corresponding to the VNC protocol used 31.1 % of the CPU resources and 9.9 % of RAM memory. The PID 3263 corresponding to the Acquired_3D_Position algorithm that receives the data from the position sensor MPU 6050 and graphs the 3D position of the ROV used 15.6% of the CPU resources and 2.2% of RAM memory. The PID 1203, which manages the motor control algorithm, labeled as Motors_Control in Figure 10, used 10.3% of the CPU resources and 3.7% of RAM memory. Finally, the PID 3223 corresponding to the Measure_Temperature algorithm, associated to the sensor DS18B20, used 8.3% of the CPU resources and 3.7% of RAM memory.



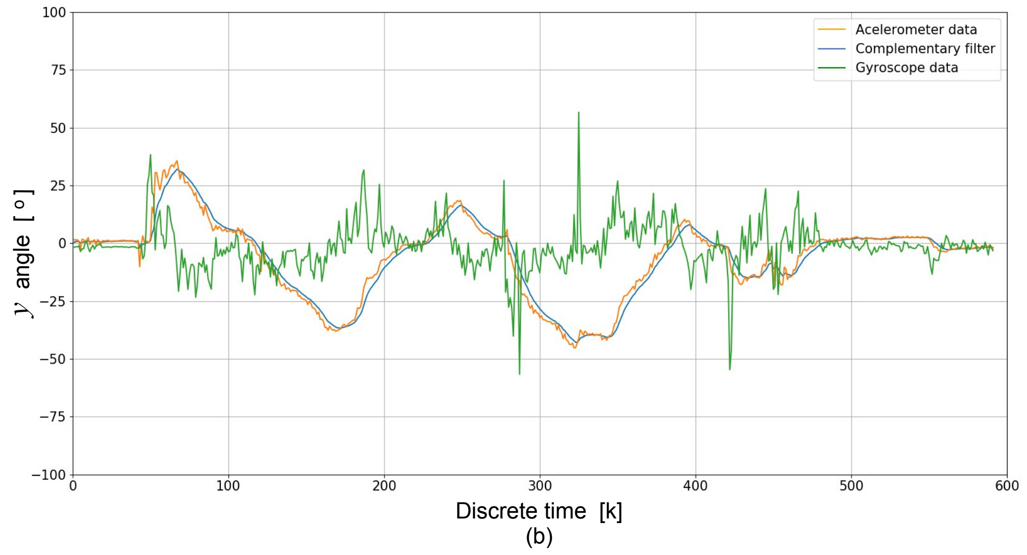
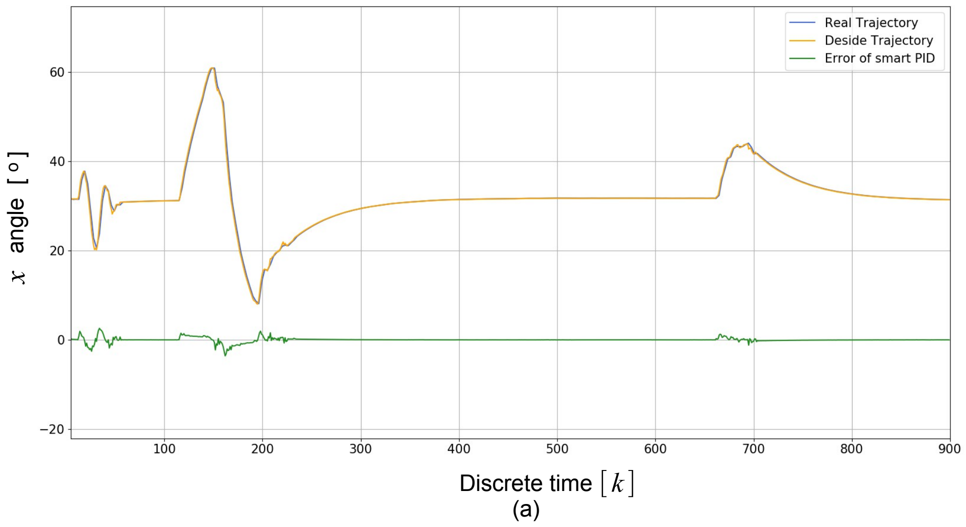


Figure 15. Stability tests of the ROV on the output of the complementary filter: (**a**) movements on x-axis; and (**b**) movements on y-axis.

Figure 18 shows the experimental results of the images captured by the camera given in frames per second (FPS), which are processed by the Raspberry Pi 3. It can be appreciated that the maximum speed of video capture is 42 FPS.



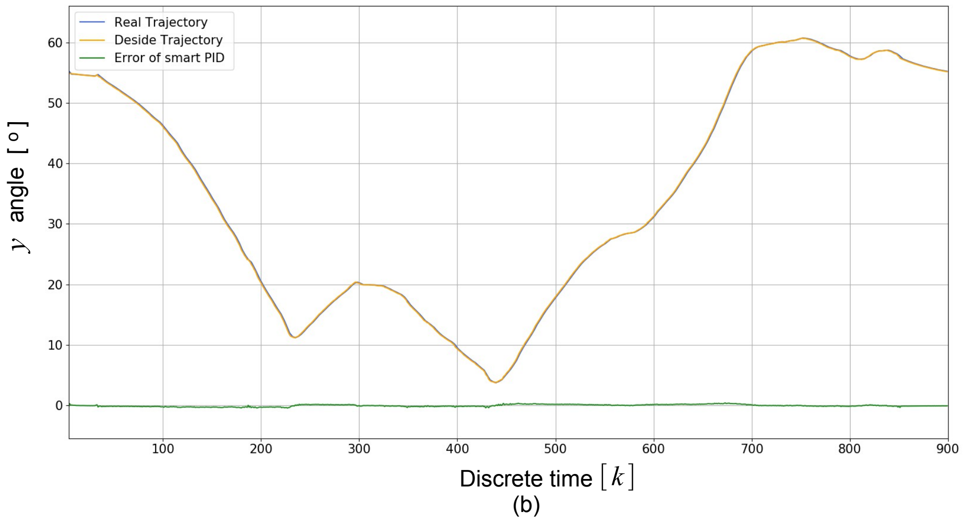
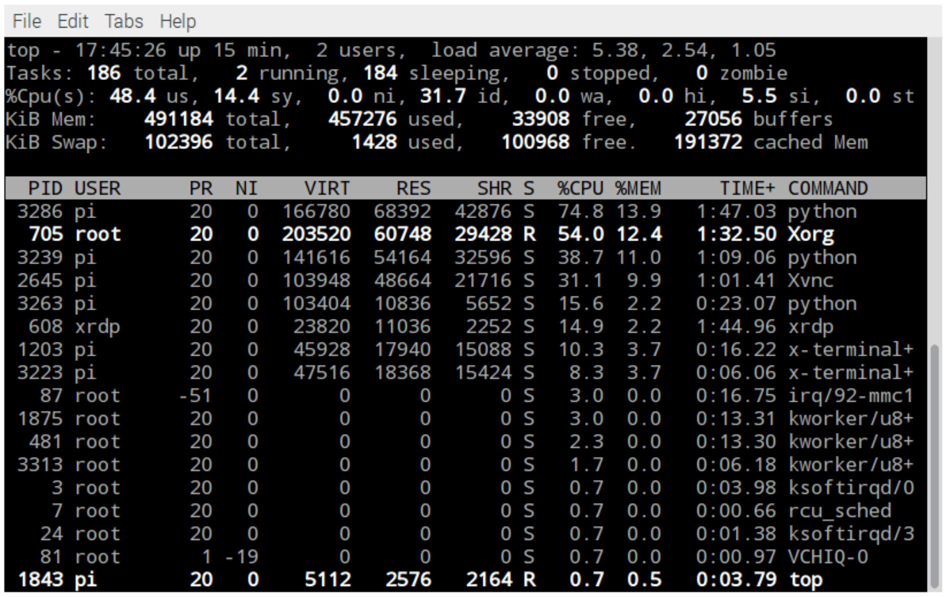


Figure 16. Smart PID controller's response: (**a**) *x*-direction; and (**b**) *y*-direction

4.3. Tests in Controlled Aquatic Environment

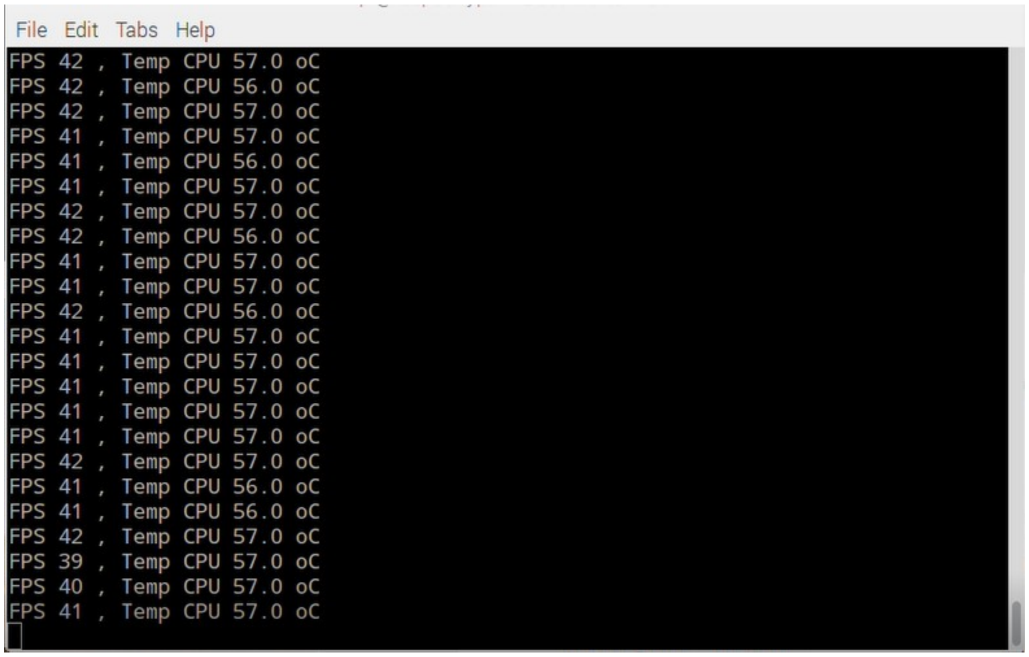
Figure 19 depicts the captured images by the ROV camera in a controlled aquatic environment. These tests also verified that the water does not leak inside the ROV capsule, and proved its buoyancy in the water. The top of Figure 19 shows the submerged ROV in the aquatic environment. The bottom of Figure 19 depicts an underwater image captured by the ROV. The remote control interface of the ROV can also be observed, which could work using a touch screen. In addition, a 3D graphic indicates the ROV position.



```
top - 17:45:26 up 15 min,  2 users,  load average: 5.38, 2.54, 1.05
Tasks: 186 total,   2 running, 184 sleeping,   0 stopped,   0 zombie
%Cpu(s): 48.4 us, 14.4 sy,  0.0 ni, 31.7 id,  0.0 wa,  0.0 hi,  5.5 si,  0.0 st
KiB Mem:   491184 total,   457276 used,    33908 free,    27056 buffers
KiB Swap:  102396 total,     1428 used,   100968 free.   191372 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3286 pi        20   0  166780  68392  42876 S  74.8 13.9   1:47.03 python
  705 root      20   0  203520  60748  29428 R  54.0 12.4   1:32.50 Xorg
 3239 pi        20   0  141616  54164  32596 S  38.7 11.0   1:09.06 python
 2645 pi        20   0  103948  48664  21716 S  31.1  9.9   1:01.41 Xvnc
 3263 pi        20   0  103404  10836   5652 S  15.6  2.2   0:23.07 python
  608 xrdp      20   0   23820  11036   2252 S  14.9  2.2   1:44.96 xrdp
 1203 pi        20   0   45928  17940  15088 S  10.3  3.7   0:16.22 x-terminal+
 3223 pi        20   0   47516  18368  15424 S   8.3  3.7   0:06.06 x-terminal+
   87 root     -51   0       0      0      0 S   3.0  0.0   0:16.75 irq/92-mmc1
 1875 root      20   0       0      0      0 S   3.0  0.0   0:13.31 kworker/u8+
  481 root      20   0       0      0      0 S   2.3  0.0   0:13.30 kworker/u8+
 3313 root      20   0       0      0      0 S   1.7  0.0   0:06.18 kworker/u8+
    3 root      20   0       0      0      0 S   0.7  0.0   0:03.98 ksoftirqd/0
    7 root      20   0       0      0      0 S   0.7  0.0   0:00.66 rcu_sched
   24 root      20   0       0      0      0 S   0.7  0.0   0:01.38 ksoftirqd/3
   81 root       1 -19       0      0      0 S   0.7  0.0   0:00.97 VCHIQ-0
 1843 pi        20   0    5112   2576   2164 R   0.7  0.5   0:03.79 top
```

Figure 17. Hardware resources in the Raspberry Pi 3.



```
FPS 42 , Temp CPU 57.0 oC
FPS 42 , Temp CPU 56.0 oC
FPS 42 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 56.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 42 , Temp CPU 57.0 oC
FPS 42 , Temp CPU 56.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 42 , Temp CPU 56.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
FPS 42 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 56.0 oC
FPS 41 , Temp CPU 56.0 oC
FPS 42 , Temp CPU 57.0 oC
FPS 39 , Temp CPU 57.0 oC
FPS 40 , Temp CPU 57.0 oC
FPS 41 , Temp CPU 57.0 oC
```

Figure 18. Experimental results of the number of captured frames per second (FPS), and temperature measurement provided by the Raspberry Pi 3.

4.4. Tests in Real-World Scenario

This section presents different images obtained by the ROV in a submarine environment. To carry out these tests, the camera was placed at the lower part of the ROV; this so that the images of the PVC pipe installed on the seabed could be captured. The images were taken at the seawater suction stage, located on the coast at 2.4 m depth. This seawater was then processed by several filters, used to feed various fish and mollusk culture ponds, at a biotechnology area for aquaculture. As an example, several image sequences are shown in groups of three, which allows the visualization of the potential of the proposed ROV for various applications and environments. As shown in Figure 15, the ROV captured video of up to 42 FPS, and from these sequences images can be extracted, as shown in Figures 20–25. In all these cases, one can observe three frames that were obtained in a real-world scenario, such as the subsea, where water turbidity is noticeable.

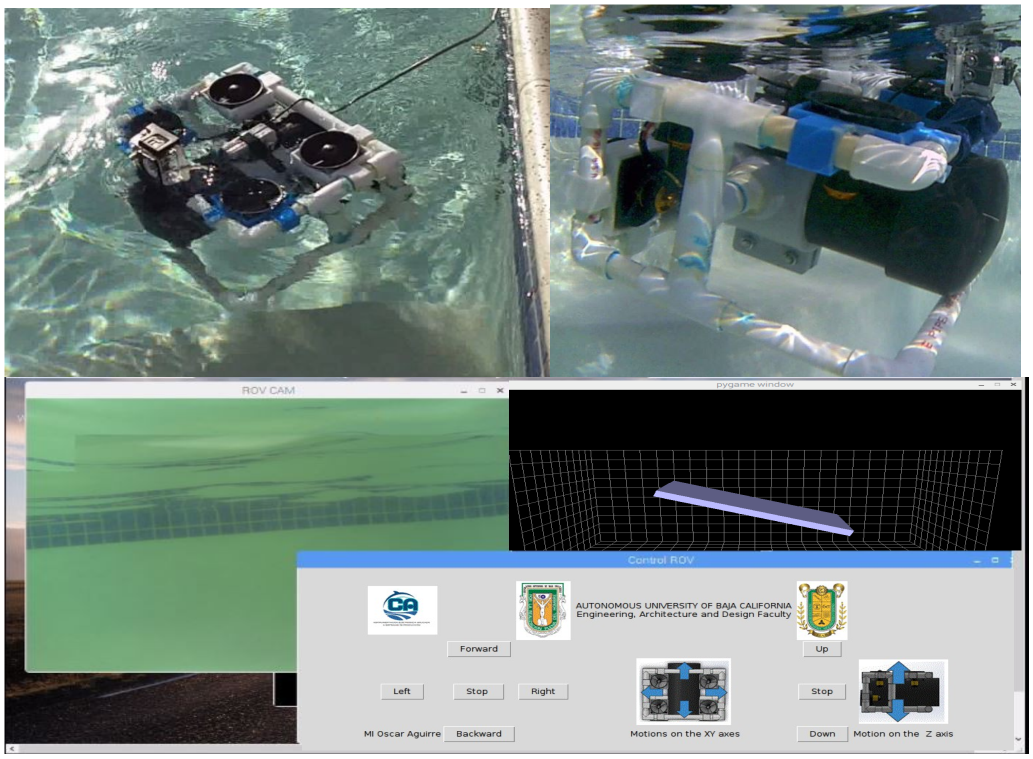


Figure 19. Images captured in a controlled aquatic environment.

Figure 20 depicts a sequence of a valve for suction in a subsea water intake. Different approaches can be observed and the external condition of the valve, a sediment film in its exterior, is particularly distinguished. This type of inspection is of great value, because, if the valve inlets were clogged, it could cause a greater deterioration to the suction pumps. Furthermore, due to excess of sediment, the water's flow through the suction valve might become blocked, which could cause greater damage to the suction pump and water shortage for the aquaculture ponds.

Figures 21 and 22 depict two sequences of images from different sections of a PVC pipe used for seawater suction. Different approaches and the external conditions of the pipe are observed. In addition, different types of connections and the conditions in which they are found are highlighted here. This type of visual inspection, obtained remotely with the ROV, serves to detect any physical damage to the pipe or connections, which is important in preventive maintenance routines for this type of installation.

Figures 23 and 24 depict two sequences of images of a PVC pipe. Different approaches are shown; they specifically highlight the incrustations that have formed around them over time. This inspection is of great value as it make the conditions in which the pipe is found evident and the latent risk that this can generate throughout the system, such as a leak of some sort of substance.

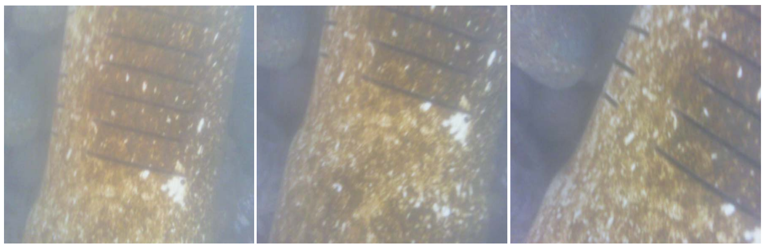


Figure 20. Inspection test images of a suction valve in subsea water intake.

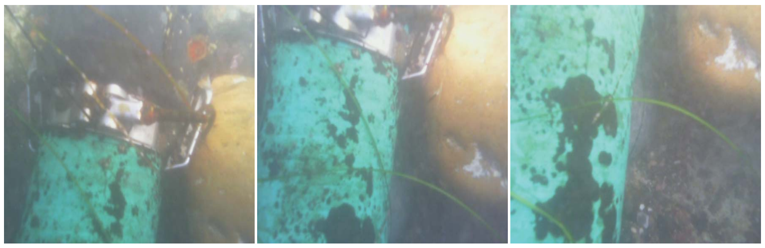


Figure 21. Underwater images of PVC pipe fasteners.

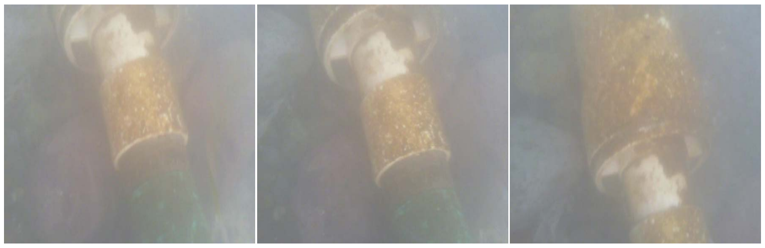


Figure 22. Underwater images of current status of connections between PVC pipes.

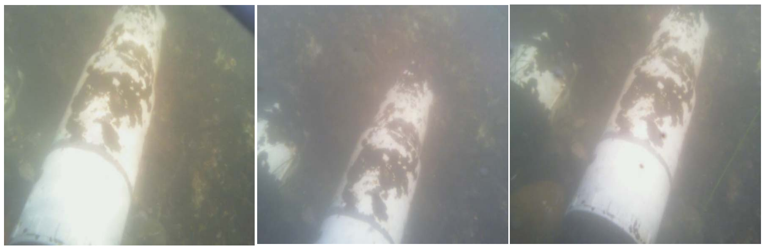


Figure 23. Images of connections between PVC pipes placed underwater.

Figure 25 shows a sequence of three images of a seabed region. Different approaches of its topography are observed. This type of inspection is of great value, for example for the study of fauna, flora, and for the search of microorganisms underwater.

4.5. Comparison with Other ROVs

Table 3 shows a features comparison of proposed ROV versus commercial ROVs. It can be observed that some features are the same. However, the proposed ROV has parallel computing capacity, which allows simultaneously executing several tasks or processes and capturing images of up to 42 FPS with a resolution of 800 $\times$ 640 pixels. The implementation of the complementary filter (see Figure 4) to the IMU signals (see Figure 15) helps to improve the performance of the smart PID controller (see Figure 6) used for the trajectory control (see Figure 16), allowing to capture better quality of underwater images (see Figures 20–25). The remote control of the ROV is executed under a graphical user interface coded on Python, which is suitable for different operating systems, such as GNU/Linux, Windows, Android, and OS X. The payload is the carrying capacity of a ROV, e.g., to add more sensors, measuring instruments, samples collected from the ocean, etc. The payload of the proposed ROV is 20% of its total weight, i.e., 3.128 kg. On the other hand, given the characteristics of the proposed mechanical and hardware design, the maintenance and replacement of mechanical and hardware components allows technological independence of ROV manufacturers.

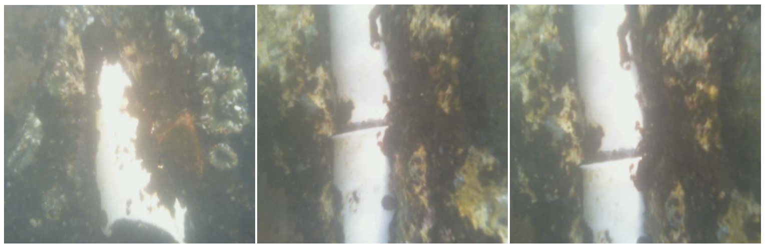


Figure 24. Images of incrustations on underwater PVC pipes.

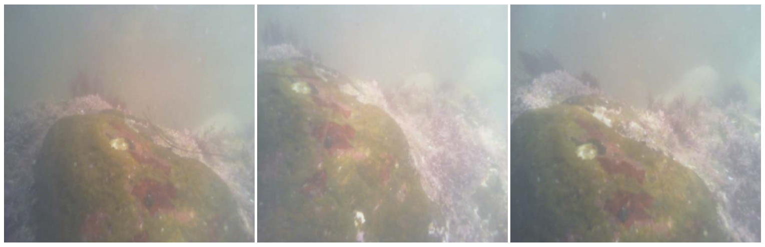


Figure 25. Images under seabed conditions.

Table 3. Comparison of characteristics of the proposed ROV vs. commercial ROV.

Features	Blue ROV (Standard)	OpenROV	Proposed ROV
Architecture	Open	Open	Open
Camera 1080 p	Yes	Yes	Yes
Autonomy time [h]	2–3	2–3	2–3
Communication	Ethernet	Ethernet	Ethernet
Internet connectivity	Yes	Yes	Yes
Maximum depth [m]	100	100	100
Processing type	Unknown	Unknown	Parallel
Frames per second [FPS]	30	30	42
Controller algorithm	PID	PID	Smart PID
Remote control	Joystick	Joystick for Android 5	Graphic user interface *
Payload [kg]	2.200	1.000	3.128
Dimensions (length×width×height) [cm]	45.7 × 33.8 × 25.4	8 × 20 × 40	18.4 × 29.5 × 33.5
Total weight [kg]	11.00	2.90	15.64
Total cost [USD]	2784.0	1695.0	600.0

* Multi-platform (Windows, Linux, Unix, Android).

5. Conclusions

The ROV development and implementation contained in a box of size 18.41 cm × 29.50 cm × 33.50 cm, with a weight of 15.64 kg, is introduced. The proposed ROV performs the translational, ascent, descent, and rotational movements on three axes to capture images of 800 $\times$ 640 pixels on video graphic array standard. The ROV design was done to reach up to 100 m underwater, which can solve the problem of divers who can only reach 30 m. The motion control, 3D position, temperature sensing, and video capture are performed in parallel by using the threading library, and they are processed by a main algorithm that was programmed to use the four cores of a SoC Raspberry Pi 3. The communication between the ROV and the remote control is handled by a graphical user interface coded on python, which is suitable for different operating systems, such as GNU/Linux, Windows, Android, and OS X. Furthermore, the ROV moves under the six brushless motors governed through a smart PID controller. From experimental results, the brushless motors were calibrated to work with a short pulse width of 1 µs to improve the ROV stability and underwater position. A complementary filter used to smooth noise vibrations from the MPU 6050 sensor improves ROV stability. Therefore, it helps to improve the captured video quality by processing up to 42 FPS on a Raspberry Pi 3. The autonomy of the proposed ROV is up to 2 or 3 h. The algorithms were programmed using open software tools, which allows adding more sensors and functionalities, according to the needs of operation, maintenance, supervision, and sensing of physical-chemical variables underwater. In addition, the flexibility of mechanical design and low-cost hardware increases potential applications, such as surveillance, fishing operations, growth control of fish, and study of marine flora and fauna, without demeriting the quality in the acquisition of the information, within a context of technological independence.

Author Contributions: Conceptualization, E.I.-G., E.E.G.-G., and E.T.-C.; Formal analysis, O.A.A.-C. and J.E.O.-T.; Funding acquisition, E.I.-G. and J.R.C.-V.; Investigation, O.A.A.-C., E.I.-G., E.E.G.-G., O.R.L.-B., E.T.-C., J.R.C.-V., and J.E.O.-T.; Methodology, O.A.A.-C. and E.I.-G.; Supervision, E.E.G.-G. and E.T.-C.; Writing—original draft, O.A.A.-C. and E.I.-G.; and Writing—review, and editing, E.I.-G., E.E.G.-G., J.R.C.-V., and E.T.-C.

Funding: This work was supported by the research project approved at the 18th Internal Call for Research Projects by UABC with number 485. Thanks are given to PRODEP (Professional Development Program for Professors) for supporting the academic groups to increase their degree of consolidation. O.A.A.-C. was supported for his postgraduate studies at Ph.D. level by CONACyT. The authors would also like to express their gratitude to the TECNM because of its financial support for the project TIJ-PYR-2019-6321.

Acknowledgments: The authors are thankful to the reviewers for their comments and suggestions to improve the quality of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lin, Y.H.; Chen, S.Y.; Tsou, C.H. Development of an Image Processing Module for Autonomous Underwater Vehicles through Integration of Visual Recognition with Stereoscopic Image Reconstruction. *J. Mar. Sci. Eng.* **2019**, *7*. [CrossRef]
2. Choi, J.K.; Yokobiki, T.; Kawaguchi, K. ROV-Based Automated Cable-Laying System: Application to DONET2 Installation. *IEEE J. Ocean. Eng.* **2018**, *43*, 665–676. [CrossRef]
3. Anderlini, E.; Parker, G.G.; Thomas, G. Control of a ROV carrying an object. *Ocean. Eng.* **2018**, *165*, 307–318. [CrossRef]
4. Capocci, R.; Omerdic, E.; Dooly, G.; Toal, D. Fault-Tolerant Control for ROVs Using Control Reallocation and Power Isolation. *J. Mar. Sci. Eng.* **2018**, *6*. [CrossRef]
5. Liu, X.; Qi, F.; Ye, W.; Cheng, K.; Guo, J.; Zheng, R. Analysis and Modeling Methodologies for Heat Exchanges of Deep-Sea In Situ Spectroscopy Detection System Based on ROV. *Sensors* **2018**, *18*. [CrossRef]
6. Sivčev, S.; Rossi, M.; Coleman, J.; Omerdić, E.; Dooly, G.; Toal, D. Collision Detection for Underwater ROV Manipulator Systems. *Sensors* **2018**, *18*. [CrossRef]

7. Capocci, R.; Dooly, G.; Omerdić, E.; Coleman, J.; Newe, T.; Toal, D. Inspection-class remotely operated vehicles. *J. Mar. Sci. Eng.* **2017**, *5*. [CrossRef]
8. Khojasteh, D.; Kamali, R. Design and dynamic study of a ROV with application to oil and gas industries of Persian Gulf. *Ocean Eng.* **2017**, *136*, 18–30. [CrossRef]
9. Choi, H.T.; Choi, J.; Lee, Y.; Moon, Y.S.; Kim, D.H. New concepts for smart ROV to increase efficiency and productivity. In Proceedings of the 2015 IEEE Underwater Technology, UT 2015, Chennai, India, 23–25 February 2015. [CrossRef]
10. Yao, H. Research on Unmanned Underwater Vehicle Threat Assessment. *IEEE Access* **2019**, *7*, 11387–11396. [CrossRef]
11. Yan, J.; Gao, J.; Yang, X.; Luo, X.; Guan, X. Tracking control of a remotely operated underwater vehicle with time delay and actuator saturation. *Ocean Eng.* **2019**, *184*, 299–310. [CrossRef]
12. Berg, H.; Hjelmervik, K.T. Classification of anti-submarine warfare sonar targets using a deep neural network. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; pp. 1–5. [CrossRef]
13. Ferri, G.; Bates, J.; Stinco, P.; Tesei, A.; Lepage, K. Autonomous underwater surveillance networks: A task allocation framework to manage cooperation. In Proceedings of the 2018 OCEANS—MTS/IEEE Kobe Techno-Oceans (OTO), Port Island, Kobe, 28–31 May 2018. [CrossRef]
14. Ferri, G.; Munaf, A.; Lepage, K.D. An Autonomous Underwater Vehicle Data-Driven Control Strategy for Target Tracking. *IEEE J. Ocean. Eng.* **2018**, *43*, 323–343. [CrossRef]
15. Choi, J.K.; Yokobiki, T.; Kawaguchi, K. Peer-Reviewed Technical Communication. *IEEE J. Ocean. Eng.* **2018**, *43*, 665–676. [CrossRef]
16. Wang, R.; Chen, G. Design and Experimental Research of Underwater Maintenance Vehicle for Seabed Pipelines. In Proceedings of the 2018 3rd International Conference on Robotics and Automation Engineering (ICRAE), Guangzhou, China, 17–19 November 2018; pp. 146–149. [CrossRef]
17. Duraibabu, D.B.; Poeggel, S.; Omerdic, E.; Capocci, R.; Lewis, E.; Newe, T.; Leen, G.; Toal, D.; Dooly, G. An Optical Fibre Depth (Pressure) Sensor for Remote Operated Vehicles in Underwater Applications. *Sensors* **2017**, *17*, 406. [CrossRef] [PubMed]
18. Cardigos, S.; Mendes, R. Using LAUVs in highly dynamic environments: Influence of the tidal estuarine outflow in the thermocline structure. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018. [CrossRef]
19. Brito, M.P.; Griffiths, G. Updating Autonomous Underwater Vehicle Risk Based on the Effectiveness of Failure Prevention and Correction. *J. Atmos. Ocean. Technol.* **2018**, *35*, 797–808. [CrossRef]
20. Sawyer, D.E.; Mason, R.A.; Cook, A.E.; Portnov, A. Submarine Landslides Induce Massive Waves in Subsea Brine Pools. *Sci. Rep.* **2019**, 1–9. [CrossRef] [PubMed]
21. Ohata, S.; Ishii, K.; Sakai, H.; Tanaka, T.; Ura, T.A. Development of an autonomous underwater vehicle for observation of underwater structures. In Proceedings of the OCEANS 2005 MTS/IEEE, Washington, DC, USA, 17–23 September 2005. [CrossRef]
22. Jiang, J.-J.; Wang, X.Q.; Duan, F.-J.; Fu, X.; Yan, H.; Hua, B. Bio-Inspired Steganography for Secure Underwater Acoustic Communications. *IEEE Commun. Mag.* **2018**, *56*, 156–162. [CrossRef]
23. Tang, C.; von Lukas, U.F.; Vahl, M.; Wang, S.; Wang, Y.; Tan, M. Efficient underwater image and video enhancement based on Retinex. *Signal Image Video Process.* **2019**, *13*, 1011–1018. [CrossRef]
24. Sanchez-Ferreira, C.; Coelho, L.S.; Ayala, H.V.H.; Farias, M.C.Q.; Llanos, C.H. Bio-inspired optimization algorithms for real underwater image restoration. *Signal-Process.-Image Commun.* **2019**, *77*, 49–65. [CrossRef]
25. Qiao, X.; Bao, J.; Zeng, L.; Zou, J.; Li, D. An automatic active contour method for sea cucumber segmentation in natural underwater environments. *Comput. Electron. Agric.* **2017**, *135*, 134–142. [CrossRef]
26. Ghani, A.S.A.; Isa, N.A.M. Automatic system for improving underwater image contrast and color through recursive adaptive histogram modification. *Comput. Electron. Agric.* **2017**, *141*, 181–195. [CrossRef]

27. Joaquin, A.; Ana, J.; Álvarez, F.J.; Daniel, R.; Carlos, D.M.; Jesus, U. Characterization of an Underwater Positioning System Based on GPS Surface Nodes and Encoded Acoustic Signals. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 1773–1784. [CrossRef]
28. Philipp, M.; Michele, M.; Luca, B. Self-Sustaining Acoustic Sensor With Programmable Pattern Recognition for Underwater Monitoring. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 1–10. [CrossRef]
29. Li, Q.; Ben, Y.; Naqvi, S.M.; Neasham, J.A.; Chambers, J.A. Robust Student's t-Based Cooperative Navigation for Autonomous Underwater Vehicles. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 1762–1777. [CrossRef]
30. Chi, C.; Li, Z. High-Resolution Real-Time Underwater 3-D Acoustical Imaging Through Designing Ultralarge Ultrasparse Ultra-Wideband 2-D Arrays. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 2647—2657. [CrossRef]
31. Rossi, M.; Trslić, P.; Sivčev, S.; Riordan, J.; Toal, D.; Dooly, G. Real-Time Underwater StereoFusion. *Sensors* **2018**, *18*. [CrossRef] [PubMed]
32. Anwar, I.; Mohsin, M.O.; Iqbal, S.; Abideen, Z.U.; Rehman, A.U.; Ahmed, N. Design and fabrication of an underwater remotely operated vehicle (Single thruster configuration). In Proceedings of the 2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 12–16 January 2016; pp. 547–553. [CrossRef]
33. Mesa, M.A.; Forero, D.Y.; Hernandez, R.D.; Sandoval, M.C. Design and Construction of an AUV Robot Type ROV. *J. Eng. Appl. Sci.* **2018**, *13*, 7980–7989. [CrossRef]
34. García-Valdovinos, L.G.; Fonseca-Navarro, F.; Aizpuru-Zinkunegi, J.; Salgado-Jiménez, T.; Gómez-Espinosa, A.; Cruz-Ledesma, J.A. Neuro-Sliding Control for Underwater ROV's Subject to Unknown Disturbances. *Sensors* **2019**, *19*. [CrossRef]
35. Cely, J.S.; Saltaren, R.; Portilla, G.; Yakrangi, O.; Rodriguez-Barroso, A. Experimental and Computational Methodology for the Determination of Hydrodynamic Coefficients Based on Free Decay Test: Application to Conception and Control of Underwater Robots. *Sensors* **2019**, *19*. [CrossRef]
36. Martorell-torres, E.; Martorell-torres, E.; Gabriel, G.F.; Base, S. Xiroi ASV: A Modular Autonomous Surface Vehicle to Link Communications. *IFAC Papers Online* **2018**, *51*, 147–152. [CrossRef]
37. Xu, R.; Tang, G.; Xie, D.; Huang, D. Underactuated tracking control of underwater vehicles using control moment gyros. *Int. J. Adv. Robot. Syst.* **2018**, 1–8. [CrossRef]
38. Teague, J.; Allen, M.J.; Scott, T.B. The potential of low-cost ROV for use in deep-sea mineral, ore prospecting and monitoring. *Ocean Eng.* **2018**, *147*, 333–339. [CrossRef]
39. Bustos, C.; Sanchez, J.; García, L.G.; Orozco, J.P. CFD Modeling of the Hydrodynamics of the CIDESI Underwater Glider. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; pp. 1–6. [CrossRef]
40. Zhang, G.; Zeng, Q.; Zhu, Z.; Dai, X.; Zhu, C.; Dai, X. Research on underwater safety inspection and operational robot motion control. In Proceedings of the 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), Nanjing, China, 18–20 May 2018; pp. 322–327. [CrossRef]
41. Chen, X. Effects of leading-edge tubercles on hydrodynamic characteristics at different Reynolds numbers. In Proceedings of the 2018 OCEANS—MTS/IEEE Kobe Techno-Oceans (OTO), Port Island, Kobe, 28–31 May 2018; pp. 1–5. [CrossRef]
42. Hernández-Ontiveros, J.M.; Inzunza-González, E.; García-Guerrero, E.E.; López-Bonilla, O.R.; Infante-Prieto, S.O.; Cárdenas-Valdez, J.R.; Tlelo-Cuautle, E. Development and implementation of a fish counter by using an embedded system. *Comput. Electron. Agric.* **2018**, *145*, 53–62. [CrossRef]
43. Flores-Vergara, A.; Inzunza-González, E.; García-Guerrero, E.E.; López-Bonilla, O.R.; Rodríguez-Orozco, E.; Hernández-Ontiveros, J.M.; Cárdenas-Valdez, J.R.; Tlelo-Cuautle, E. Implementing a Chaotic Cryptosystem by Performing Parallel Computing on Embedded Systems with Multiprocessors. *Entropy* **2019**, *21*, 268. [CrossRef]
44. Hernández-Alvarado, R.; García-Valdovinos, L.G.; Salgado-Jiménez, T.; Gómez-Espinosa, A.; Fonseca-Navarro, F. Neural Network-Based Self-Tuning PID Control for Underwater Vehicles. *Sensors* **2016**, *16*. [CrossRef] [PubMed]
45. Brzozowski, B.; Rochala, Z.; Wojtowicz, K.; Wieczorek, P. Measurement data fusion with cascaded Kalman and complementary filter in the flight parameter indicator for hang-glider and paraglider. *Measurement* **2018**, *123*, 94–101. [CrossRef]

46. Yang, Q.; Sun, L. A fuzzy complementary Kalman filter based on visual and IMU data for UAV landing. *Optik* **2018**, *173*, 279–291. [CrossRef]
47. Pititeeraphab, Y.; Jusing, T.; Chotikunnan, P.; Thongpance, N.; Lekdee, W.; Teerasoradech, A. The Effect of Average Filter for Complementary Filter and Kalman Filter Based on Measurement Angle. In Proceedings of the 2016 9th Biomedical Engineering International Conference (BMEiCON), Laung Prabang, Laos, 7–9 December 2016. [CrossRef]
48. Wang, W.H.; Engelaar, R.C.; Chen, X.Q.; Chase, J.G. The State-of-Art of Underwater Vehicles—Theories and Applications. In *Mobile Robots—State of the Art in Land, Sea, Air, and Collaborative Missions*; IntechOpen: London, UK, 2012; Volume 2000. [CrossRef]