IMPLEMENTATION OF OPEN LOOP MOTORS CONTROL OF AN ROV

*Note: Sub-titles are not captured in Xplore and should not be used

1st Marisol Vázquez Tzompantzi
Secretaría de Investigación y *Posgrado*
Instituto Politécnico Nacional
México City, México
mvazquezt@ipn.mx
https://orcid.org/0000-0003-3892-7293

2nd Bruno Yael Silva Morales
Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional
Mexico City, Mexico
bsilvam1200@alumno.ipn.mx
https://orcid.org/0009-0007-3570-4904

3rd Suresh Kumar Gadi
Uniiversidad Autónoma de Coahuila
Instituto Politécnico Nacional
Coahuila, Mexico.
research@skgadi.com
https://orcid.org/0000-0001-7974-7825

4th Carlos Horus Acosta Espinoza
Unidad Profesional Interdisciplinaría en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional
Mexico City, Mexico
https://orcid.org/0009-0007-1767-1950

5th Marco Sekehen Gonzalez Tepoz
Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional
Mexico City, Mexico
https://orcid.org/0009-0006-2689-1851

6th Andy Alan Castro Valdez
Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional
México City, Mexico
https://orcid.org/0009-0001-4942-9161

Abstract— **This project focused on developing a remote-control station for an unmanned underwater vehicle (UUV). Two Digilent PMOD JSTK2 joystick modules and a CMOD S7 integrated with an FPGA XC7S25-1CSGA225C were utilized to implement an algorithm on the FPGA that discretized joystick signals, enabling control of four Blue Robotics T200 thrusters and four servomotors.**

The main objective of the project was to design and build an efficient and precise remote-control system capable of maneuvering the UUV in multiple dimensions. One joystick was used to control movement along the X and Y axes of the robot, while the second joystick managed movement along the Z axis and YAW rotation.

The system architecture relied on the SPI protocol for communication between the joysticks and FPGA. VHDL code developed in this project handled reading data from the PMODs, processed these signals, and converted them into discrete commands sent to the UUV. The system also included visual signals via LEDs to monitor the status of SPI communications and control signals. (*Abstract*)

Keywords—ROV, UUV, FPGA, PMOD JSTK2, CMOD S7, REMOTE-CONTROL.

I. INTRODUCTION (*HEADING 1*)

The remote control of remotely operated underwater vehicles (ROVs) presents significant challenges due to the need for precision, reliability, and low latency in control signals, as noted by Griffiths [1]. This is crucial for managing multiple degrees of freedom, such as movement along the X, Y, and Z axes, along with yaw, pitch, and roll rotations, under variable and often adverse underwater conditions, as described by Hawkins [2].

The underwater exploration of natural water bodies such as lakes, rivers, seas, cenotes, and underwater caves, as well as artificial water bodies including dams, reservoirs, and distribution tanks, demands advanced technological solutions. The inspection of submerged structures often requires specialized resources and poses risks to human divers due to factors such as hydrostatic pressure, lack of oxygen, extreme temperatures, and encounters with hazardous substances and wildlife. In this context, Unmanned Underwater Vehicles (UUVs) offer a safe and efficient alternative to carry out these challenging tasks as outlined by [3].

This article focuses on the design and implementation of an efficient and precise remote control system for the UUV using PMOD JSTK2 joystick modules and a CMOD S7 with FPGA XC7S25-1CSGA225C, supported by digital design fundamentals as described by Brown [4] and Chávez [5] for submarine communications. The goal is to discretize analog signals from the joysticks and process them in real-time to generate control commands transmitted with low latency to the FPGA of the UUV 'Don Francisco,' as depicted in Fig 1. The objective is to ensure reliable communication and appropriate response of the UUV to received commands, thereby optimizing its navigation and manipulation capabilities in challenging underwater environments.

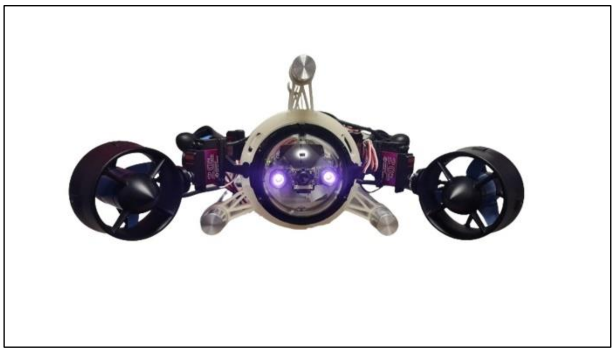


Fig. 1: UUV Don Francisco.

II. OBJETIVES

A. *Design and implement a signal discretization algorithm on the FPGA*

An algorithm was designed to convert the analog signals from the joysticks into discrete data usable by the FPGA. This algorithm interpreted voltage variations on the X, Y, Z axes, as well as yaw, pitch, and roll rotations, converting them into precise digital values.

B. *Integrating the PMOD JSTK2 modules with the CMOD S7 for reading joystick signals*

A physical and logical connection was established between the PMOD JSTK2 modules and the CMOD S7 FPGA to ensure accurate reading of joystick signals. This integration enabled continuous and precise capture of joystick data.

C. *Develop a reliable and efficient SPI communication interface.*

An SPI (Serial Peripheral Interface) interface was implemented in the FPGA for efficient and reliable communication between the joysticks and the FPGA. This interface handled the transmission and reception of data with low latency and high precision.

D. *Implement movement and rotation controls for the UUV.*

The discrete data from the joysticks were translated into control commands that managed the movement along the X, Y, Z axes, and the YAW rotation of the UUV. These controls enabled precise and smooth maneuverability of the vehicle.

E. *Provide a visual interface for monitoring the system status.*

A visual interface shown in Fig. 2 was created to allow the operator to monitor the system status in real time. This interface displays relevant information such as images acquired by the robot's camera, orientation, and internal temperature.

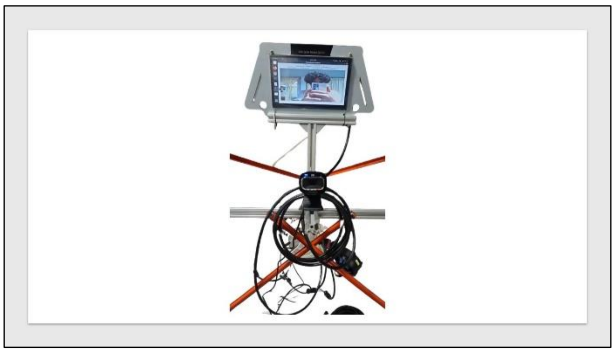


Fig. 2: HMI's Visual Interface

III. Problem definition

The remote control of Unmanned Underwater Vehicles (UUVs) poses significant challenges due to the need for precision, reliability, and low latency in control signals. This is crucial for managing multiple degrees of freedom, such as movement along the X, Y, and Z axes, along with yaw rotation, under variable and often adverse underwater conditions.

This article focuses on the design and implementation of an efficient and precise remote control system for the UUV using PMOD JSTK2 joystick modules and a CMOD S7 with FPGA XC7S25-1CSGA225C. The goal is to discretize analog signals from the joysticks and process them in real-time to generate control commands transmitted with low latency to the UUV's FPGA. The aim is to ensure reliable communication and appropriate response of the UUV to received commands, thereby optimizing its navigation and manipulation capabilities in challenging underwater environments.

Specifically, the challenge is to structure a Human-Machine Interface (HMI) to control a robot, implementing the mentioned components. To better understand the problem, the physical architecture of the UUV has been diagrammed, and the physical architecture of the HMI is proposed.

To analyze the issue, the information from authors Cross [6] and Hurst [7] has been used as a basis.

A. *UUV Physical Architecture.*

Fig. 3 shows how the UUV and its key components are physically composed, as well as the interactions between them.

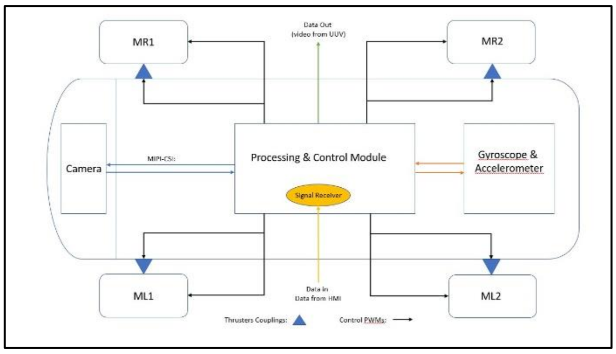


Fig. 3: Physical Architecture from UUV.

1) *Processing & Control Module:*

   a) *Responsible for processing data and controlling the various components of the UUV.*

   b) *Receives input data from the HMI.*

2) *Gyroscope & Accelerometer:*

   a) *It measures the orientation and acceleration of the UUV, the data is sent to the Processing and Control Module for analysis and use in vehicle control.*

3) *Camera:*

   a) *Captures video that is sent to the Processing and Control Module via the MIPI-CSI interface.*

   b) *The processed video is sent out of the UUV as data output.*

4) *Thrusters:*

   a) *ML1 and ML2 (Motor Left 1 and Motor Left 2): Engines on the left side of the UUV.*

   b) *MR1 y MR2 (Motor Right 1 y Motor Right 2): Engines on the right side of the UUV.*

   c) *They are mounted on the UUV's hull and receive PWM (Pulse Width Modulation) control signals from the Control Module to adjust their speed and direction.*

B. *HMI's Physical architecture.*

Fig. 4 shows the proposed physical architecture for the Human Machine Interface (HMI) of the UUV control system. Each component and its interaction are described below.

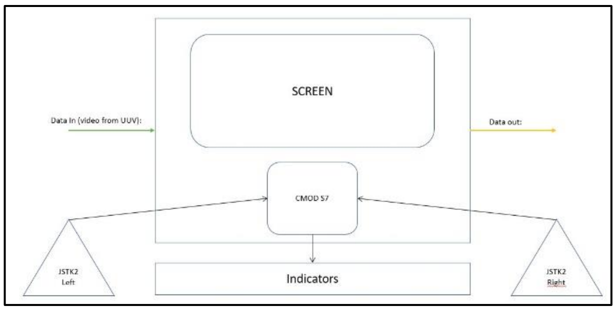


Fig. 4: Physical Architecture from HMI.

1) SCREEN (Monitor):

a) Displays the video received from the UUV and relevant information for the operator.

2) CMOD S7:

a) Acts as an electronic control unit in the HMI.

b) Receives signals from JSTK2 joysticks (left and right).

c) Procesa las señales de control y las envía al UUV.

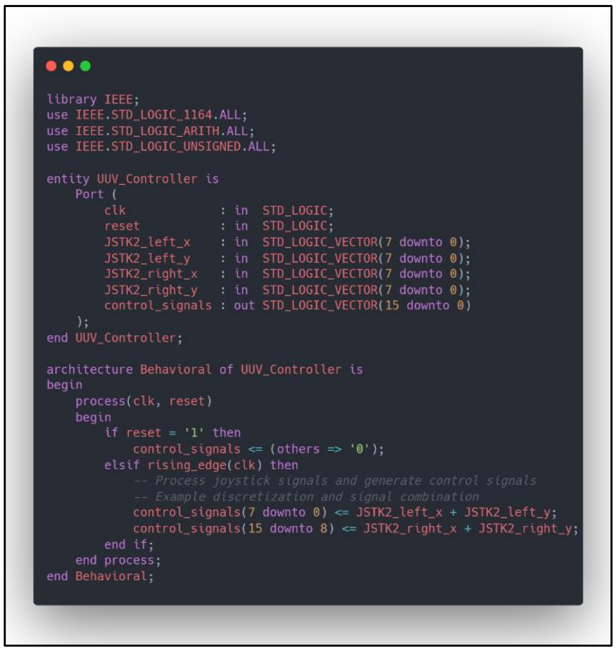
3) JSTK2 Left y JSTK2 Right (Joysticks Izquierdo y Derecho):

4) Indicators (LEDs):

They show information about the signals that are being sent to the UUV from the HMI.

C. Hardware dessign Implementation

Fig. 5 shows a VHDL code snippet exemplifying one of the hardware modules in the HMI design that sends instructions to the robot.



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity UUV_Controller is
    Port (
        clk             : in  STD_LOGIC;
        reset           : in  STD_LOGIC;
        JSTK2_left_x    : in  STD_LOGIC_VECTOR(7 downto 0);
        JSTK2_left_y    : in  STD_LOGIC_VECTOR(7 downto 0);
        JSTK2_right_x   : in  STD_LOGIC_VECTOR(7 downto 0);
        JSTK2_right_y   : in  STD_LOGIC_VECTOR(7 downto 0);
        control_signals : out STD_LOGIC_VECTOR(15 downto 0)
    );
end UUV_Controller;

architecture Behavioral of UUV_Controller is
begin
    process(clk, reset)
    begin
        if reset = '1' then
            control_signals <= (others => '0');
        elsif rising_edge(clk) then
            -- Process joystick signals and generate control signals
            -- Example discretization and signal combination
            control_signals(7 downto 0) <= JSTK2_left_x + JSTK2_left_y;
            control_signals(15 downto 8) <= JSTK2_right_x + JSTK2_right_y;
        end if;
    end process;
end Behavioral;
```

Fig. 5: VHDL Hardware Design

1) Entity Statement:

a) The UUV_Controller entity is defined with its input and output ports.

b) Inputs: clk (clock), reset (reset), JSTK2_left_x, JSTK2_left_y, JSTK2_right_x, JSTK2_right_y (joystick signals).

c) Output: control_signals (control signals for UUV).

2) Functional Architecture:

a) A process that runs on the ascending edge of the clock (rising_edge(clk)) or when the reset is activated (reset = '1') is used.

b) When the reset is activated, the control signals are reset to zero.

c) On the rising edge of the clock, the joystick signals are processed and discretized control signals are generated.

3) Signal Processing:

a) The signals are encoded and sent to the UUV's control electron unit.

IV. METODOLOGY

A functional analysis using IDEF0 has been conducted on the UUV to propose the HMI and establish the operation sequence of the robot, building upon a prior functional analysis of the UUV as indicated in the methodology by X. Yan [8].

These diagrams express the function to be analyzed within a block, the inputs, outputs, the mechanisms or tools, and the control or models that will be employed.

A. IDEF0's UUV

The diagram shown in Fig 6 states that the movement of the robot is the function of interest, the input is the control signal, and the outputs are the information that is sent to the user, the dimming of the lighting, and the new position of the robot.

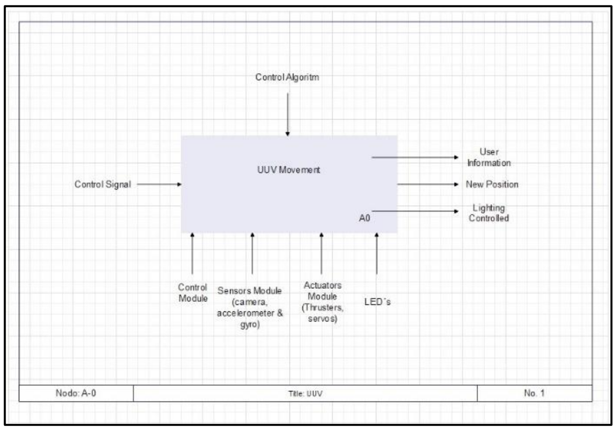


Fig. 6: UUV's IDEF0.

To obtain these outputs, a control algorithm intervenes that, when reading the control signal, makes changes to the robot's actuators. This movement function is broken down into more functions that are shown in the diagram of Fig. 7, here are shown the different functions that are executed so that the robot moves when receiving the control signal from the HMI, once the signal is decoded the robot moves, this in turn modifies its linear and angular velocity parameters, in addition to the fact that the intensity of the lighting is modified.

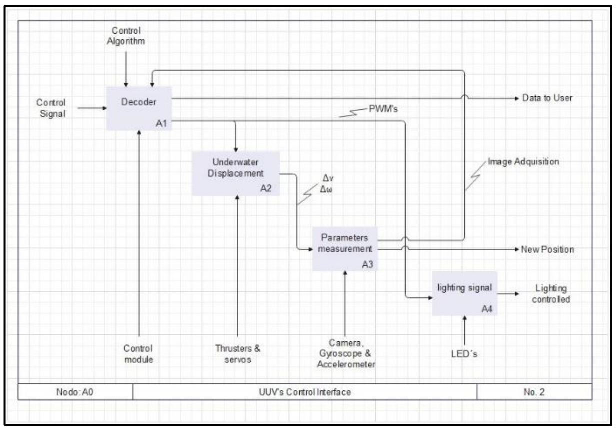


Fig. 7: UUV's control IDEF0.

The data of the change in position and the images captured by the robot are sent to the HMI so that the robot operator can interpret them.

The robot's actuators, sensors, LED lighting system and control modules are involved in this process.

B. *IDEF0's HMI*

Having clear the functional process of the UUV, the control strategy implemented in the HMI has been proposed, which is shown in the IDEF0 of Fig. 8.

The main node forms the control, whose inputs are made up of the joystick modules and their buttons, which after the process will result in a control signal.

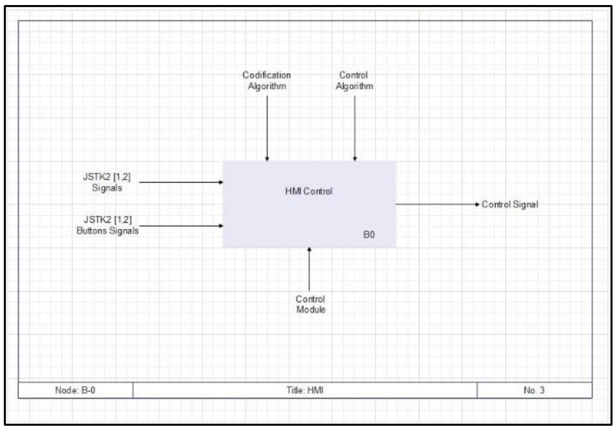


Fig. 8: HMI's IDEF0

This process involves control coding algorithms and the HMI control module itself, which is based on the CMOD S7.

In Fig. 9, the complete process of operation of the robot is shown, it is shown in which stages the signals of the joysticks and buttons are read, where the algorithms and the control module intervene.

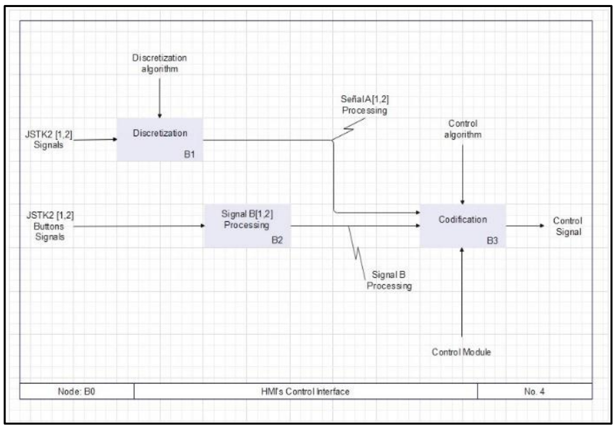
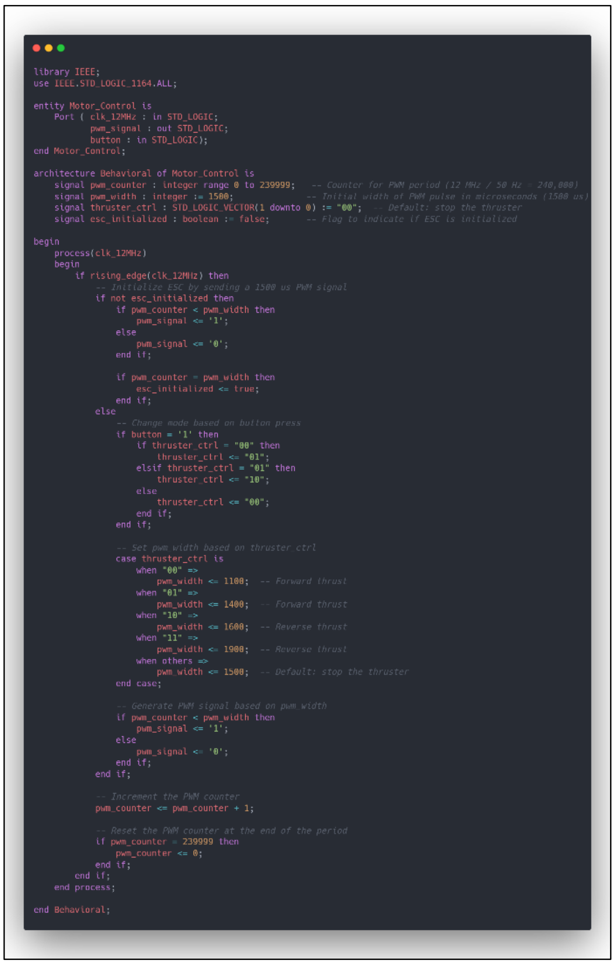


Fig. 9: HMI's Control IDEF0

These diagrams make it easier to understand how different systems work and thus the hardware design process.

C. *Thruster performance*

Another important factor in the development of UUV control is the performance of the thrusters, which are a key factor in the robot's battery life according to Claus [9]. For this, static tests were conducted with the thrusters and the UUV control FPGA. The code shown in Fig. 10 is the synthesized result of the testing algorithm for these tests, while Fig. 11 shows the schematic diagram generated by that hardware design.



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Motor_Control is
    Port ( clk_12MHz : in STD_LOGIC;
           pwm_signal : out STD_LOGIC;
           button : in STD_LOGIC);
end Motor_Control;

architecture Behavioral of Motor_Control is
    signal pwm_counter : integer range 0 to 239999;   -- Counter for PWM period (12 MHz / 50 Hz = 240,000)
    signal pwm_width : integer := 1500;              -- Initial width of PWM pulse in microseconds (1500 us)
    signal thruster_ctrl : STD_LOGIC_VECTOR(1 downto 0) := "00";   -- Default: stop the thruster
    signal esc_initialized : boolean := false;       -- Flag to indicate if ESC is initialized

begin
    process(clk_12MHz)
    begin
        if rising_edge(clk_12MHz) then
            -- Initialize ESC by sending a 1500 us PWM signal
            if not esc_initialized then
                if pwm_counter < pwm_width then
                    pwm_signal <= '1';
                else
                    pwm_signal <= '0';
                end if;

                if pwm_counter = pwm_width then
                    esc_initialized <= true;
                end if;
            else
                -- Change mode based on button press
                if button = '1' then
                    if thruster_ctrl = "00" then
                        thruster_ctrl <= "01";
                    elsif thruster_ctrl = "01" then
                        thruster_ctrl <= "10";
                    else
                        thruster_ctrl <= "00";
                    end if;
                end if;

                -- Set pwm_width based on thruster_ctrl
                case thruster_ctrl is
                    when "00" =>
                        pwm_width <= 1100;  -- Forward thrust
                    when "01" =>
                        pwm_width <= 1400;  -- Forward thrust
                    when "10" =>
                        pwm_width <= 1600;  -- Reverse thrust
                    when "11" =>
                        pwm_width <= 1900;  -- Reverse thrust
                    when others =>
                        pwm_width <= 1500;  -- Default: stop the thruster
                end case;

                -- Generate PWM signal based on pwm_width
                if pwm_counter < pwm_width then
                    pwm_signal <= '1';
                else
                    pwm_signal <= '0';
                end if;
            end if;

            -- Increment the PWM counter
            pwm_counter <= pwm_counter + 1;

            -- Reset the PWM counter at the end of the period
            if pwm_counter = 239999 then
                pwm_counter <= 0;
            end if;
        end if;
    end process;

end Behavioral;
```

Fig. 10: Hardware design from Thrusters Test

El diagrama representa la arquitectura de hardware de la FPGA para el funcionamiento de los thrusters.

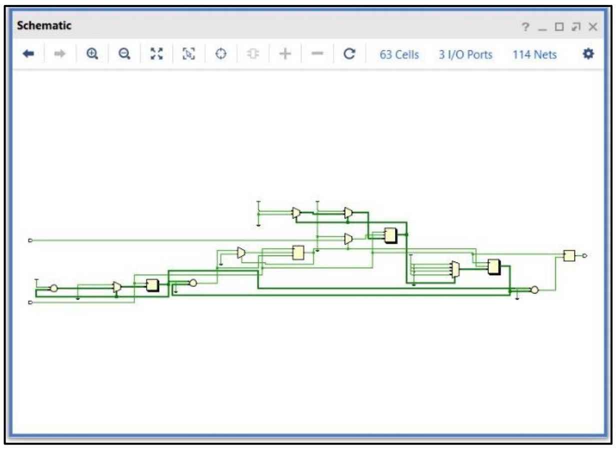


Fig. 11: Thursters Test Eschematic by RTL.

In TABLE 1 the data in multiples of 100 of the static tests of the thrusters are shown as well as their physical and electrical parameters when they are controlled.

TABLE I. THRUSTERS PARAMETROS

PWM (µs)	RPM	[A]	[V]	[W]	Thrust [Kg f]	Efficiency
1100	2976	17.03	12	204.4	-2.90	14.2
1200	2457	8.9	12	106.8	-1.95	18.2
1300	1815	3.3	12	39.6	-1.02	25.8
1400	1023	0.5	12	6.0	-0.32	52.9
1500	0	0	12	0.0	0.00	0.0
1600	1009	0.5	12	6.0	0.39	65.8
1700	1796	3.3	12	39.6	1.28	32.3
1800	2456	8.9	12	106.8	2.46	23.1
1900	2995	16.91	12	202.9	3.71	18.3

The board was built by measuring the different electrical parameters by varying the PWM in the Electronic Speed Controller that delivers power to the motor.

A MATLAB license was used to analyze the complete data table with a greater PWM range and with them obtain graphs of power, force and efficiency of the thrusters when implementing the control.

V. RESULTS & DISCUSSION

a) Power

The graph in Fig. 12 shows the relationship between the revolutions per minute (RPM) of the thrusters and the PWM signal from the ESC input.

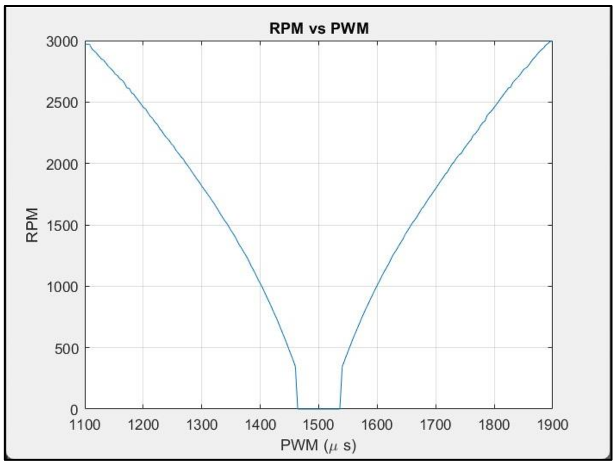


Fig. 12: RPM vs PWM's from Electronic Control Unit

A linear relationship between the PWM value and the RPM can be observed up to certain points, after which the RPM stabilizes.

b) RPM vs Thrust

The graph in Fig. 13 shows the power consumed by the thrusters as a function of the thrust generated, measured in kilogram-force (Kg f)

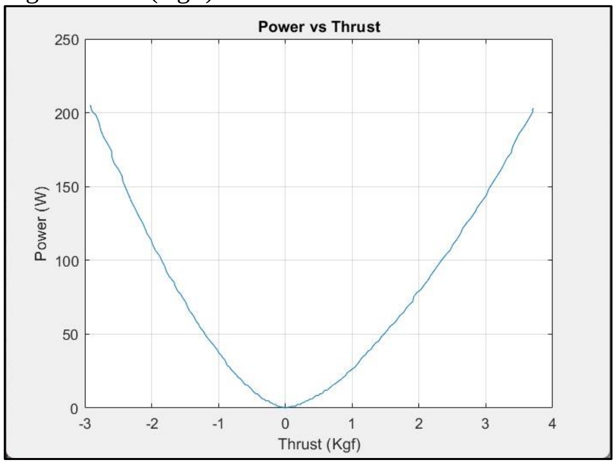


Fig. 13: Power vs Thrust.

The relationship between power and thrust is critical to understanding how energy consumption varies with propellant load.

c) *Efficiency*

The graph in Fig. 14 shows the efficiency of the thrusters as a function of the PWM signal from input to the ESC. Efficiency is measured in grams per watt (g/W).

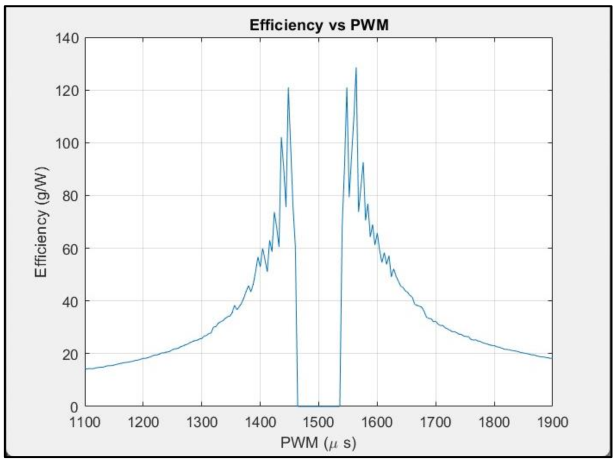


Fig. 14:Efficiency RPM vs PWM's from Electronic Control Unit

CONCLUSIONS

The implementation of FPGAs in the control interface shows to have a higher performance and response speed by providing the possibility of designing the System's own architecture, putting as an option the possibility of importing the design to an Application-Specific Integrated Circuit. (ASIC).

The linear behavior of RPMs vs PWMs is crucial to determine the operational limits of the thrusters and ensure that the system's capabilities are not exceeded, which could lead to premature wear or mechanical failures.

It is observed that at higher levels of thrust, the power required increases significantly. Planning the UUV maneuvers to operate in the most efficient ranges of this curve will allow optimal energy management, extending the battery life and, therefore, the vehicle's autonomy.

It is observed that the efficiency is maximum in the range of [1400-1500] microseconds of PWM values, indicating that the system should preferably operate in these ranges to minimize power consumption and maximize thrust. Operating outside of these ranges can result in inefficient energy use, reducing the range of the UUV.

ACKNOWLEDGMENT

Thanks to the IPN's Secretaría de Investigación y Posgrado for the funding and administrative support during the development of this project.

REFERENCES

[1] G. Griffiths, TECHNOLOGY AND APPLICATIONS OF AUTONOMOUS UNDERWATER VEHICLES, London: Taylor & Francis Group, 2003.

[2] J. Hawkins and A. Allcock, Oceanography and Marine Biology, CRC Press, 2019.

[3] G. Griffiths and Stevenson, "Open ocean operational experience with the Autosub-1 autonomous underwater vehicle," in *Proceedings 11th Unmanned Untethered Submersible Technology Symposium*, Durham, New Hampshire, 1999, pp. 1-12.

[4] S. Brown and Z. V, Fundamentals of digital logic with VHDL Design, Ciduad de México: McGraw-Hill, 2000.

[5] J. Chávez, Sistema de comunicaciones para submarinos autonomos empleando ultrasonido, Ciudad de México: M. S. Tesis, CINVESTAV, 2019.

[6] N. Cross, Engineering Design Metods Strategies for Product Design, 3ra ed, Wiley. Reino Unido: John Wiley & Sons, LTD., 2000.

[7] K. Hurst, Engineering Design Principies, Elsevier Science & Technology Books, 1999.

[8] X. Yan and R. Zante, A Mechatronic Design Process and Its Applications. Case Studies in Mechatronics - Applications and Educations, London, Dordrecht: Springer, 2010.

[9] B. Claus and R. Bachmayer, "Energy optimal depth control for long range underwater vehicles with applications to a hybrid underwather glider," in *Autonomous Robots*, 2016, p. 1307.