Technical Design of Caleuche WAM-V Platform for the 2022 RobotX Competition

Paula Martínez, Francisco Eterovich, Javiera Fuentes, Vicente Sufán, Gabriel Olguín, Tomás Contreras, Diego Galleguillos and Cristian Nova.

***Abstract*—The Pontificia Universidad Católica de Chile participated for the first time in the Maritime RobotX Challenge 2022, a competition that seeks to promote the development of autonomous vehicles in marine environments.**

This new team named Caleuche took on the task of automating a Wave Adaptive Modular Vessel (WAM-V), equipping the vehicle with a mechanical, electrical and intelligent software system, along with the incorporation of a drone that allows it to face the competition challenges. This is how the first version of the Caleuche team was ready to compete in RobotX Maritime Challenge 2022.

Being part of this competition has allowed unprecedented technological development in our university, gaining experience and knowledge which will allow us to continue to improve for future versions of the competition.

I. Introduction

Every year the demand for unmanned surface vehicles (USV's) increases in scientific and defense areas, especially for marine environments. Due to this, the need arose to promote the development of these vehicles and equip them to accomplish multiple goals in dynamic environments.

To promote the development of these technologies, the Robotx Maritime Challenge was founded, which takes place every two years and brings together student teams from around the world, who carry out the task of automating a Wave Adaptive Modular Vessel (WAM-V).

A. Background Overview

For the first time, Chile participated in some version of the Maritime RobotX Challenge being its representative, the Caleuche team. This document presents the work that was developed from August 2021 to the competition in November 2022, documenting its implementation systems in the areas of mechanics, electricity, and software, along with algorithm designs and the incorporation of a drone.

B. Software and Hardware Overview

The software was developed using the Robotic Operating System (ROS) framework and external ROS packages. In the robot, the software runs on multiple processing units that communicate through the ROS multi-machine package since they are on the same local network.

The processing hardware consists of four Single Board Computers (SBCs), three Nvidia Jetson Nano, and one Raspberry Pi (RPi).

Available sensors include one OS0 Ouster Light Detection and Ranging (LIDAR), an OpenCV AI Kit with depth perception PoE camera (OAK-D), and a Microstrain Global Navigation Satellite System (GNSS) with an integrated Inertial Navigation System (INS).

TABLE 1: Hardware Units

Units	**Nº**	**type**
Nvidia Jetson Nano 4GB	3	Processing Unit
Raspberry Pi 3B+ 2GB	1	Processing Unit
Oak D POE (AF)	1	Vision Sensor
LORD 3DMGQ7-GNSS/INS	1	Navigation Sensor
Uniform OS0 LIDAR	1	Perception Sensor

II. Design Strategy

A. Simulation Environment

To start with the challenges, it was decided to use the official simulation environment of the competition, Virtual RobotX[1] (VRX), developed in Gazebo within the ROS environment. The simulator made it possible to replicate most of the conditions that made up the competition tasks: objects, physical environment properties, and even personify simulated sensors so that they met the same properties as the real ones, such as LIDAR, camera, IMU, and GPS.

All of these features made VRX the ideal environment to learn, develop, and test software without having to rely on hardware or physical terrain, speeding up our workflow.

B. Task Selection

Since time was a scarce resource, the team decided to prioritize five tasks to be developed. The first was task 0 since it is a requirement to pass this test in order to compete in the others. The second task was HeartBeat since it allows the WAM-V to report its status to both the team and the judges. The other three tasks chosen were: Follow The Path (task 3), Scan The Code (task 5), and Search & Rescue (task 9) with the Drone team.

The decision was made taking into consideration the capabilities and knowledge of the team, added to the resources and time available.

C. Control

For control of the autonomous surface vessel (ASV), first, it is necessary to eliminate part of the nonlinearities of the system. To achieve this, force measurements were taken for each of the motors to know the force exerted by each of them at a certain level of power. With these results, a linearizer was developed for each of the motors, which has a function for transforming the signal coming from the controller into a power signal corresponding to the characteristics of each motor. Once this is done, it is possible to apply control strategies for the vehicle movement, which are explained in section III.

D. Perception

At first, it was planned to build a 3D object perception system using sensor co-registration between the OAK-D camera and LIDAR. After some experiments, it was decided that the camera, given its field of view, resolution, and precision, did not make a useful contribution to the system, so it was developed using LIDAR only.

To process the point clouds generated by the LIDAR, it was decided to use Machine Learning (ML) techniques, over Deep Learning (DL), due to the limited processing capacity of the SBCs. LIDAR turned out to be the precise sensor to detect objects in marine conditions. Due to water reflecting light away from the sensor instead of uniformly diffusing it, the only points detected by the LIDAR belonged to objects above the water surface and outliers, which was ideal for applying clustering algorithms and ML classifiers.

E. Electrical

The electrical engineering team was tasked with powering up sensors, SBCs, antennas, networking equipment, motor systems, and emergency stop system, as well as developing the communication system between the SBCs used by the software team to control the motors. The design goal was to make the systems previously mentioned as simple and robust as possible. To achieve this, all the connections were designed to be as close to plug and play as possible while maintaining waterproof capabilities with standard waterproof connectors.

To further simplify the assembly process, a large PCB was designed and attached to one of the Pelican cases to manage all connections from the SBC that control the system to the electronics in each motor. This system made it easy to test, debug and solve errors during testing.

F. Software

The ROS framework was chosen for its versatility as a communication system, for its compatibility with the Virtual RobotX simulator, but mainly for its wide repertoire of code tools and graphical interfaces, such as control packages and the ROS visualizer (Rviz).

Most of the developers on the Caleuche team lacked experience with tools like ROS and Gazebo and had never worked on such large-scale robotics projects, so it was necessary to take the “learning and then apply” ideology. The first objective was to get to know VRX tools: Rviz, Gazebo and the Unified Robot Description Format (URDF). With this knowledge, it was possible to adapt the simulation environment to start facing challenges in specific conditions.

This “learn and then apply” ideology was maintained in each of the fields in which software needed to be built: control, perception, vision, and communication. It may have caused development delays, but it increased the team's ability to build robust solutions.

G. Drone

The Drone relies on MAVLink serial protocol from ArduPilot framework and Python algorithms to achieve autonomous flight and complete tasks. It is equipped with one Raspberry Pi connected to a Pixhawk Flight Controller, along with an RPi Camera and Time of Flight LiDAR (TFLIDAR) sensors. To communicate with the drone from the ground there are four up/downlink signals, a 2.4GHz Frsky ACCST radio frequency, for manual operation, manual arming and selection of flight mode. A 915MHz MAVLink telemetry radio frequency, for flight parameters monitoring using Mission Planner.tA 915MHz LoRa connection between the Raspberry Pi and the ground station for online tunneling between both components, which allows for the modification of flight variables and commands previous or during flight. Finally, there is an SSH tunnel via Wi-Fi to access the Linux terminal of the RPi aboard the drone.

III. Vehicle Design

A. Power and Propulsion System

1) Power Distribution

The electrical power for the motors came directly from their respective batteries that are directly connected to them, except the hardware installed inside the motors that was powered by batteries installed in the Pelican cases that supplied 5 Volts to the Spirit 1.0 motors and 9 Volts to the Torqeedo Travelmotors. This design was chosen so that it was easier to later implement the emergency stop system. The two Jetson Nano boards are also powered by 5 Volts and are in the same Pelican case as their batteries and the two batteries for the hardware electronics in the motors that were previously mentioned.

Finally, in order to power up the sensors there was a second Pelican case containing a 24V Battery with three Voltage dividers (5V, 12V and 24V) that delivered power to the cameras, GNSS and LIDAR. A power inverter provided 220V AC power to the power over ethernet injector for the long range antenna radio.

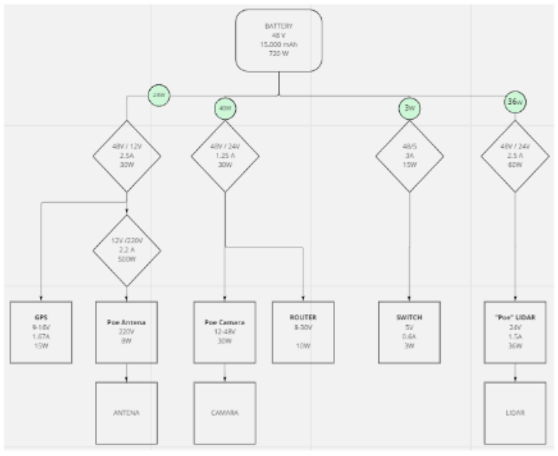

Figure 1: Sensor power supply diagram

2) Emergency stop system

For the emergency stop system, power comes directly from the batteries used to power up the electronics in the motors, where this signal goes through a circuit with four emergency stop buttons and finally through two relays so that it is possible to enable a digital stop from one of the Jetson Nano SBCs. The two relays were soldered to a custom-made PCB that also has the connections that go directly to the motors in order to simplify the assembly process when running tests.

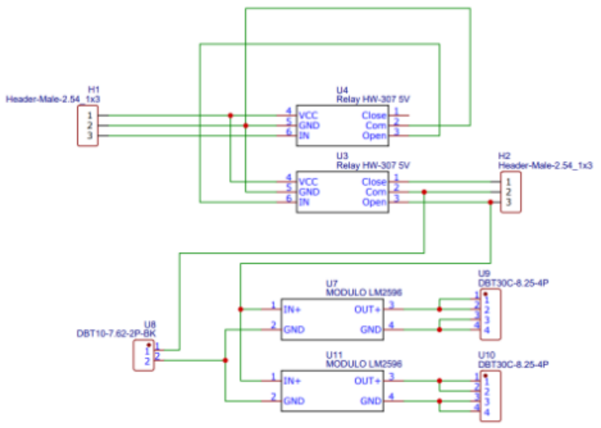

Figure 2: Emergency stop relay diagram

3) Propulsion System

The Caleuche's propulsion system began with two thrusters at the stern in a forward direction, providing the WAM-V with a differential motion configuration. From the beginning, it was clear to the team that two additional thrusters were needed to achieve full movement of the WAM-V and not lose maneuverability if one or two other motors were lost. The current configuration has two thrusters at the stern and two other thrusters amidships the WAM-V, each oriented at a fixed 30 degree point. The make of the thrusters used were Epropulsion Spirit 1.0L 1kW and two Torqeedo Travel 1103CL 1.1kW electric motors.

It was expected to achieve holonomic motion control with the WAM-V, this required a mounting system that would allow a 30 degree tilt with respect to the central axis of the pontoons. The system had to meet several requirements: maintain the angle despite the oscillations of the propulsion, be easy to install and disassemble, perform well in marine environments and adjust to the conditions of the available motors. To carry out the final design, several preliminary CAD models were made and compared with the conditions of the real motor. Finally, the metal base of the upper part of the propulsion and the presses that are attached to the pods were used.

The design consisted of two parts, the first was a threaded bar fixed to the propulsion presses, which is attached to these through stainless steel sheets with perforations. The other was a profile folded at 90 degree with a plate welded at 30 degree to transmit this inclination to the motor, as shown in Figure 3. b). The first prototype for the folded part was 3D printed. After roughing it out and obtaining the exact dimensions, the system could be installed on one of the motors and tested with different power levels to corroborate the results of the simulations in Fusion 360.

Once the prototype was approved, a final version made of steel could be manufactured. This part was obtained by folding a 20 X 20 mm profile at 90 degree, which was then cut at the desired angle. A 10 mm steel plate was welded to it, divided into a central part and 2 hooks to fix it to the motor.

The parts met expected objectives in different tests and contexts, as well as conforming to all the requirements set by the team.

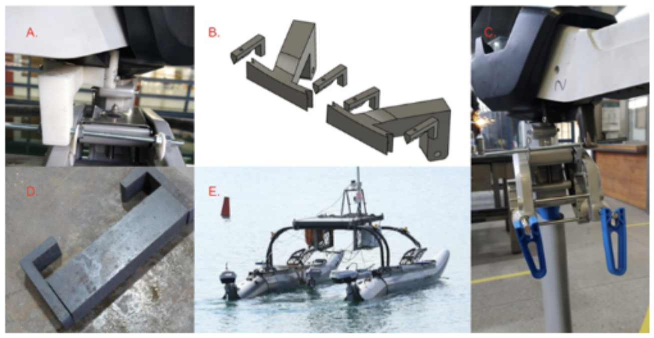

Figure 3: A. Testing of the initial prototype. B. CAD model of the frames. C. Installation of final frame. D. Part cut with EDM method. E. Real test of the implemented tilting system.

Regarding the propeller protections for the motors, they had to be designed to withstand corrosive environments and with high chlorine content, since it would be used in the sea, lakes and swimming pools; besides having mechanical resistance to ensure that the structure does not break or deform, as this could mean catastrophic damage to the motor. Finally, the mounting of this protection had to ensure that the integrity of the motor was maintained, i.e., it could not require disassembly of the system in order to be mounted.

Based on the morphology of the propulsion motor and various references of blade guards used by other teams participating in the competition, such as the one from the National University of Singapore (Bumblebee, 2022) shown in Figure 2. a); the Caleuche's Mechanics team determined that the protection should be anchored to the propulsion system shaft at the top and bottom of the motor by means of a type of clamps, as shown in Figure 2; since in this way it is possible to secure the position of the protection without modifying the morphology of the propulsion system.

On the other hand, it was determined that the protection should have two parts as shown in Figure 2.c): 1) the mounting to the propulsion system shaft, and 2) the protective casing for the blades; since in this way, it is possible to mount the protection without having to alter the motor. Consecutively, the design of Figure 2.c) was devised, which complied with the previously defined requirements, however, it had many parts joints which would imply longer assembly time. Finally, an iteration was carried out with the objective of minimizing the number of bolted joints between the parts of the assembly, resulting in a final design that meets the needs of the marine vehicle and is easy to handle for assembly and disassembly. Figure 2 shows the iterative process just mentioned.

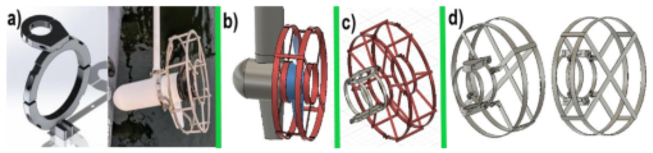

Figure 4: Iterative process of propeller protection design. a) Design reference; b) Preliminary design; c) Iteration of the preliminary design; d) Final design.

This design considers that its elements are made of stainless steel, in order to avoid corrosion and comply with the required mechanical resistance. As a result, two propeller guards were obtained for the WAM-V (one for each motor), which were validated individually and together with the catamaran during a pool test, as shown in Figure 3.

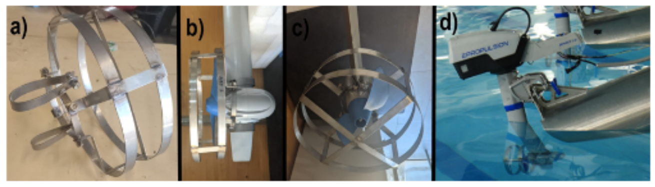

Figure 5: Propeller protection of the Caleuche equipment. a) Perspective view; b) Side view and c) Front mounted view; d) Pool view.

B. Communication System

1) Software

The communication system was built around a single Local Area Network (LAN), which allowed every network capable device to share the same address space, and to use ROS as a distributed system with its machine to machine package. The LAN had a mix of static IPs and dynamic IPs, with most static devices being those that needed to run ROS in machine to machine mode, as having static addresses simplifies this setup.

Aboard the ASV there was a five TP Link port gigabit unmanaged switch and a Mikrotik Hex Lite router running a DHCP server, required for those devices not running in link-local mode and without static addresses assigned. A secondary Mikrotik Hex Lite router acted as a bridge on the base, although this router could activate its DHCP server and exit bridge mode if the wireless link was lost.

A single long distance wireless link using a pair of Ubiquiti AirMax 5 GHz Rocket M5 radios at 19 dBi with omnidirectional antennas provided connectivity between the base station and the ASV, which given line of sight, could reach link speeds of up to 300 Mbps and over 10 km with latency no higher than 120ms. Although there was a single LAN, communication between devices inside the ASV was all over 1 Gbps links, while the wireless link had a maximum throughput of 100 Mbps, and as such, no data processing was done over on the base station, as sending multiple videos feeds or LIDAR point clouds would saturate this link. Only limited sensor visualization, teleoperation commands, and autonomous mode selection messages were sent to and from the base station. A backup NRF link provides teleoperation capabilities should the main wireless link fail.

2) Hardware

The telecommunication system had an antenna on top of the WAM-V and one antenna on the base, which were chosen because of their long range in open outdoor spaces, similar to the competition venue in Australia. The commands sent from the shore team will be received by this antenna and processed by one of the Jetson Nano SBCs on the ASV. Using an RS-485 module, the computer will send the necessary data to move the robot in the chosen direction. The two Torqeedo motors are equipped with an Arduino Due each, while each Spirit 1.0 has an ESP32. In both cases, the microcontrollers read the incoming information and simulate the signal that the magnetic encoder would produce in order to reach a certain speed in each motor. The Arduino was chosen because there was already material online where others had already hacked the motors, so it was easier to replicate. While for the other motors the choice required more freedom because we had to start from scratch, so the ESP32was chosen for its small size, low cost, built-in digital to analog converter and also because its Bluetooth capability made debugging problems on site much more manageable.

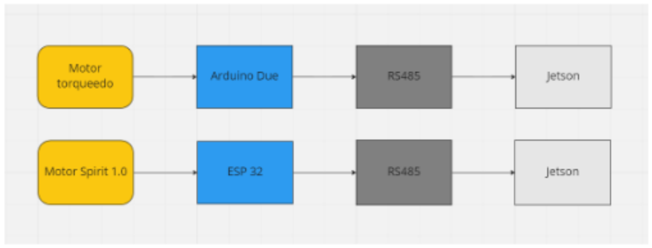

Figure 6: Hardware communication for each motor system.

C. Sensors System

In almost all the challenges, it was necessary to detect objects that are around the ASV. As the LIDAR is the main sensor responsible for detection, the entire mount was designed in order to allow the LIDAR to have the largest possible field of view.

The second sensor with the highest priority was the camera, which was positioned in the bow in order to classify the objects found in front of the ASV.

The GNSS/INS was placed just behind the LIDAR mount, its small size, and setup making it easy to install.

All the sensors are on a disassembled steel structure, with three levels, where the highest one houses the LIDAR, then the cameras, status semaphore light and antenna. The GNSS/INS rests on the main platform.

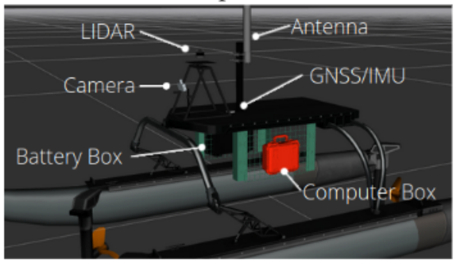

Figure 7: A visualization in Rviz of the ASV model, pointing out its main components.

D. Control System

Two independent PID controllers were developed: one for angular position (yaw) and one for linear velocity. For its implementation, the ROS PID controller package was used, which receives as input the reference value for each variable, as well as the current state of the ASV and delivers as output the actuation to the motors.

The reference values for the angular position (yaw) and linear velocity are obtained from algorithms developed for each of the tasks, where the frequency with which these values are updated depends on the particular task. On the other hand, the current state of the ASV, i.e., the angular position and linear velocity, are obtained from the fusion of GPS and IMU data in an extended Kalman filter.

In order to tune the PID controllers in the simulator and in the field tests, a graphical interface was implemented in ROS GUI Tools (RQT), which allows to dynamically configure the gain values of the controllers, as well as to visualize the time response of each of the controlled variables.

E. Software System

1) Localization

The Microstrain GNSS is the key piece of the location system; it has a ROS driver which allows it to quickly integrate it into the software architecture. GNSS was configured to obtain the position in latitude, longitude and height (LLH) coordinates and the orientation in the east, north, up (ENU) system. Since the robot moves in 2D space, it was necessary to project the spatial movement of the ASV onto a plane. A relative projection algorithm was developed using an approximation of the radius of the earth. The angle of the ASV was obtained directly from the information fusion between the GNSS and its IMU. With this mechanism, it was possible to obtain the pose of the ASV in real time, which was displayed in Rviz through a URDF model of the ASV.

2) Perception

The LIDAR-based perception algorithm is made up of three steps. The first is to condense the point cloud by projecting it from 3D to 2D, in order to reduce the computational consumption of post-processing. This step is performed using the *pointcloud-to-laserscan*[2] ROS package. The second step is to apply a hierarchical clustering based on Euclidean distance, with this we distinguish between the point clouds that make up the same object. Finally, a classification is applied on the characteristics of the resulting clusters, such as area and number of points.

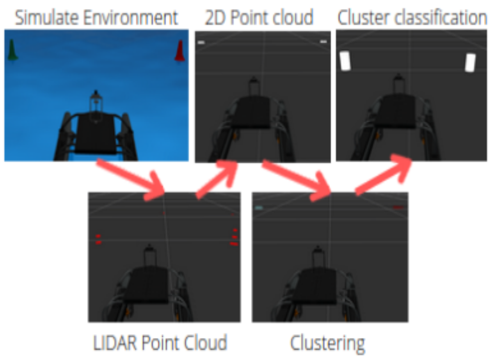

Figure 8: The complete *Perception* processing algorithm: 1) Point Clouds projection from 3D space to 2D space. 2) Grouping of agglomerations. 3) Cluster classification

3) Vision

Convolutional Neural Networks (CNN) were implemented to detect and classify images. The OAK-D camera was specially selected for this purpose; it is capable of computing CNN thanks to its built-in Tensor Processing Unit (TPU). It was easy to program and integrate to ROS software thanks to the depthai[3] Python library.

Image datasets were created and organized on the Roboflow[4] platform. YOLOV5-s/n[5] models were chosen as CNN for their excellent performance and short training times.

The combination of all these tools accelerated the process of developing and testing computer vision algorithms.

4) Task Planner

For each challenge, a sequence of states was developed to dictate the operation of the ASV. For this purpose the *smach-ros*[6] package was used, this is a Python library to develop, debug and visualize state machines. This turned out to be very useful for structuring software, including ROS services and actions.

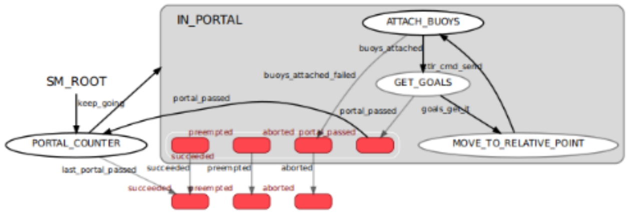

Figure 9: Task0 example of state machine system visualize with smach_view.

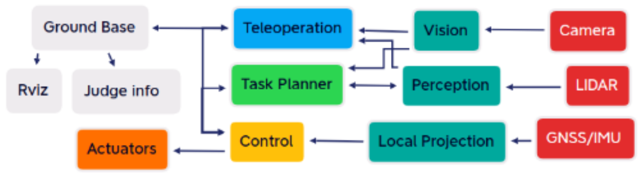

Figure 10: Software Architecture Summary

IV. Drone Design

A. Communication System

At the moment, there are four different types of communication channels between the Drone Ground Station (DGS) and the drone itself. First, there is a radiofrequency link for manual operation, under the ACCST communication protocol from FrSky technologies, which is a standard remote communication in drones that do not fly autonomously. It consists of a radio transmitter (FrSky Taranis X9D) and the corresponding receiver L9R in one drone, and the X8R in another. As mentioned, that is a standard 2.4 GHz link for manual operation of the drone, under the OpenTX software, which allows it to fly in the third person and enables simple commands to the flight controller units to switch between flight modes that are detailed in the control section.

The second communication system is a MAVLink 915MHz downlink that enables to get telemetry from sensors onboard the aircraft, which will be detailed in the corresponding section. This is a standard radio frequency link between DGS and the aircraft and is generally used to load missions and paths for autonomous flights, but it is not suitable, for the moment, to highly customized flight paths such as a precision landing algorithm, which is why a Raspberry Pi is used in addition to the Flight Controller and this radio link. This communication system is limited to basic functions typically described under open source programs such as Mission Planner or QGroundStation, so it is used at the moment only for downlink of the sensors parameters and attitude of the aircraft.

The third communication system is between the DGS and the Raspberry Pi which is an SSH tunnel that enables access to the Raspbian operating system of the RPi. This way the main computer can receive Python algorithms that will run the flight paths and custom scripts such as the precision landing. The internal communication between the Raspberry Pi and the Flight Controller is via General Port Input/Output (GPIO) serial communication. The connection of the SSH tunnel is made by a Wi-Fi dedicated network from a cell phone.

Finally, there is a LoRA connection between the drone and the DGS, using a 915 MHz up/downlink and sending light communication packages, with online dynamic parameters, such as the ASV GPS coordinates, dynamic waypoints and other significant information. This enables the dynamic landing of the drone over the ASV platform.

B. Sensors System

There are four main sensors in the drone, a Raspberry Pi camera, a TFLiDAR range finder, the IMU inside the Flight Controller and the GPS sensor. With a combination of these subsystems, the Drone is capable of flying autonomously and land where indicated.

The Raspberry Pi camera is a OV5647 HD 5MP CMOS camera, operating with a resolution of 2592*1944 px for imaging and 800*600 px for precision landing CV algorithms and a FOV of 59°.

The TFLiDAR rangefinder is the TFmini-S LiDAR is a unidirectional laser range finder based upon time-of-flight (ToF) technology, with 2° FOV and a maximum detection distance of 12 meters. It is used in the landing tasks, giving a more accurate estimation of the distance from the ground, compared to the GPS and IMU data.

The GPS+IMU sensors are combined from the Flight Controller unit, which is a Pixhawk 4 in one drone, and a Pixhawk 3 in the backup one. The standard GPS and IMU are used for both manual and autonomous flights using Mission Planner or QGroundControl software, and describe the vehicle's main attitude while flying with great accuracy.

C. Control System

The solution for controlled flight is an off-the-shelf solution from the existing Ardupilot controllers implemented in the Pixhawk control board. There is a standard PID controller which is tunable and calibrated from the stock IMU from the drone. There are no further developments over the controllers, rather than the PID-tuning to match the criteria of the flights for the competition.

D. Hardware implementation

In addition to the sensors previously mentioned, the drone has 4 electric motors, 2213 for the main drone and 2212 for the backup one. The 2-blade propellers for the main drone are 9x4.5 inch and 10x4.5 inch for the backup one. All the motors are equipped with 30A Electronic Speed Controllers (ESC) and both drones are powered by a 4S 14.8V 6200 mAh battery. Additionally, there are a couple of telemetry sensors for the main voltage and current drop, an optional gimbal + FPV video transmitter (VTX) that could be used for a direct analog video link with high FPS. Everything is connected to the Pixhawk Flight Controller, including the Raspberry Pi 3b+ which is executing the main flight commands, and the 3 radiofrequency receivers mentioned in previous sections. The electronics are mounted in a F450 drone frame modified to float above water for both the main and backup drone.

V. Experimental Results

A. Simulation Testing

1) Mechanical simulation:

Computational simulations were essential in the designs devised by the Caleuche mechanics team. The team made use of Autodesk Inventor CAD software for its finite element (FEA) and computational fluid dynamics (CFD) simulations.

One example was for the design of the propeller protections of the motors in order to corroborate that the structure will not be deformed during use, a CFD simulation was performed considering an overestimated speed of the catamaran of 3 m/s in the water, which allowed determining that an average pressure of 50 kPa was exerted on the faces orthogonal to the direction of rotation of the motor. Based on this value, the behavior of the casing under this pressure was analyzed using an FEA simulation, which allowed determining that the design has a minimum safety factor of 3.87 for the simulation conditions, as shown in Figure 4.

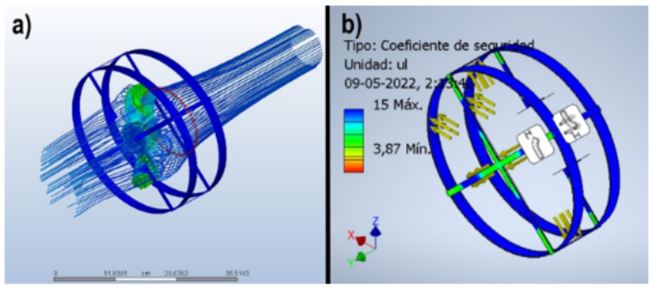

Figure 11: Simulations of the final design performed in Autodesk Inventor. a) CFD simulation; b) Stress analysis.

B. Lab Testing

Electrical test :

In terms of lab tests there are two worth mentioning that were vital to the electrical teams progress. These tests had the objective of discovering the range of values in which the Torqeedo and the Spirit 1.0 motors worked, in other words the values that the microcontrollers had to send in order to control the speed of each motor.

In the case of the Spirit 1.0 the test consisted of tapping into the signal read by the magnetic encoder when the throttle was used and then mapping the signal into functions on an ESP32. The results showed that these motors function differently to the Torqeedo motors, sending an analog signal instead of a digital message. There are 2 pins on the controller board that receives an analog signal from the ESP32 from 0 to 5V, where the sum of both voltages determines the speed of the motor, and depending on the order in which the DAC pins start to send the signal, the direction of the motor is adjusted.

For the Torqeedo motor it was a simpler task since these motors had been hacked by other people before and there was a lot of material online. Because of this we just had to adjust the ranges of values that had already been discovered by other people and tuned them to the systems that we developed and our model of the motor. For one of the motors the forward drive values were: 540-2214 and for the reverse direction : 2470-3900. For the other motor the values were 50 - 1645 and 1932 - 3330 respectively.

C. Field Testing

During the field tests, measurements were made of the force of the motors depending on the level of power sent as a signal, in order to use this information to control the vehicle. For this purpose, 1/8" steel cables were used, where one end of the cable was tied to the vehicle and the other end was connected to a dynamometer with a maximum weight of 300 kg. The results obtained for the Spirit 1.0 and Torqeedo motors are presented below.

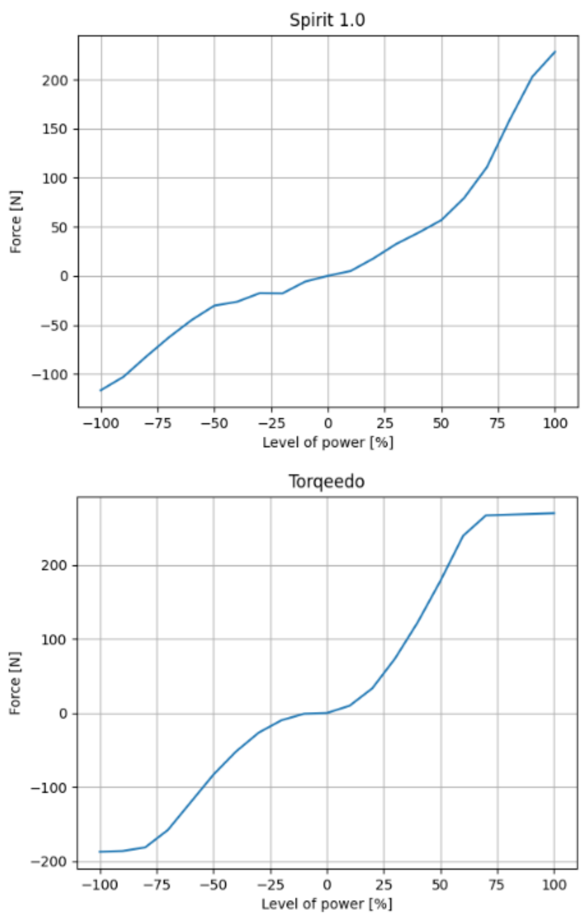

Figure 12: force measurements for Spirit 1.0 and Torqeedo motors

From the results, it was evident that the Torqeedo motors exert greater force at the same power level when compared to the Spirit 1.0 motors, where the Spirit 1.0 motors reach a maximum of 228.3 N at 100% and -116.6 N at -100%. The Torqeedo motors reach a maximum of 269.5 N at 100% and -187.2 N at -100%.

In addition, measurements were made of the speed values delivered by the GPS when it is still, in order to determine the noise level of the sensor, as well as to define if the data are reliable and robust to close the speed control loop. The results are presented below:

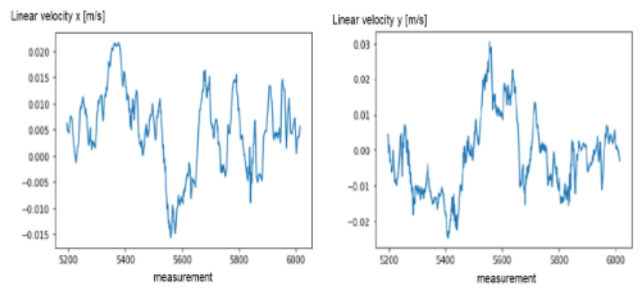

Figure 13: velocity measurements with the GPS still

From the results, it was noted that, throughout the measurements, the value of both speeds presented a low noise level, so it was determined that with these values it is possible to make a speed control loop.

VI. Conclusions

During this year we worked on designing, building and testing the autonomy of our WAM-V, developing and integrating mechanical, electrical and software components, along with navigation and control algorithms and the integration of a drone with the USV. This is how the first version of the Caleuche team's WAM-V is ready to compete in RobotX Maritime Challenge 2022.

Being part of this competition has allowed unprecedented technological development in our university, which has generated opportunities for research for both undergraduate and postgraduate students, with a broad generational reach.

During this period, the team has had the opportunity to learn and gain experience in the field, which will allow us to continue to improve our WAM-V for future versions of the competition.

VII. Acknowledgements

The team would like to thank our generous sponsors MultiAcero and Pontificia Universidad Católica de Chile.

In addition, we are grateful for all the help provided by the mechanical workshop of the UC Mechanical Engineering Department and UC Robotics Laboratory.

Finally, we would like to thank the Roboflow organization, who granted us privileges to use their platform.

VIII. References

[1] B. Bingham and C. Aguero, Toward Maritime Robotic Simulation in Gazebo, 2019, https://github.com/osrf/vrx.
[2] P. Bovbel and T. Foote, pointcloud_to_laserscan, 2015, https://wiki.ros.org/pointcloud_to_laserscan
[3] *DepthAI's Documentation — DepthAI documentation | Luxonis*. (s. f.). 2022, https://docs.luxonis.com/en/latest/
[4] *Roboflow: Give your software the power to see objects in images and video*. (s. f.). 2022, https://roboflow.com
[5] *GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*. (s. f.). GitHub. 2022, https://github.com/ultralytics/yolov5
[6] Bohren and I. I. Y. Saito, smach, 2018, http://wiki.ros.org/smach