TAU SAIL-IL RoboBoat 2023 Technical Design Review

SAIL-IL Team, Tel Aviv University, Tel Aviv, Israel

***Abstract*— This report describes the entire strategy, system engineering, design consideration and project life cycle process include design, integration, verification, and validation of the 2023 Tel Aviv University SAIL-IL team, as part of the RoboBoat annual contest.**

I. INTRODUCTION

SAIL-IL is Tel Aviv University's Engineering Department RoboBoat team, competing third year in a row at the RoboBoat competition and the only team from the state of Israel. The RoboBoat program aims to create an environment that allows students to practice research and development in the challenging and evolving world of autonomous vessels. This year the mission was to continue last year work and develop the program further with new tasks.

The objective was to develop a fully autonomous ASV (Autonomous Surface Vehicle) to operate autonomously under various missions including obstacle avoiding route, obstacle detection and classification, platform control and localization. The development process was based on agile development due to the tight constraints of the project and uncertainties. It required an ongoing planning and execution method and integration between multidisciplinary teams and various sub-systems.

II. COMPETITION STRATEGY

This year we focused on improving and further developing of the system based on conclusions from last year work. The beginning of the development process focused on system engineering process, understanding the requirements from the operational concept as well as internal requirements related to last year conclusions and allocating them to the various sub-systems as well as to the related teams. The team focused on the ability to add more capabilities to the current system to be adaptable to new missions and requirements. We began by prioritizing the autonomous state machine along with the navigation systems abilities. We see these items as the core of the ASV and as such, we devoted the most attention to these parameters. We implemented new, complex, and more accurate navigation methods. The water cannon was improved significantly with a new servo-based operating system allowing the ASV to aim the cannon without needing to reposition itself entirely. The ball cannon was designed an implemented from scratch, with simplicity and effectiveness in mind. The docking missing was initially implemented by training our neural network but later switched to an image-processing algorithm for accurate detection of various docking stations. The "Ocean Cleanup" task in which the ASV must detect an active underwater pinger and collect racquetballs from the ocean floor will not be attempted, this was decided based upon the abilities and timeframes of the tasks at hand for each team.

All tasks are based on recognizing and classifying objects and have similar manoeuvrability requirements. The entire software architecture is based on ROS (Robot Operating System) framework[1]. Add to this that the architecture is loosely coupled between the various sub-systems allow parallel and scalable development (for example obstacle avoidance using Computer Vision and LIDAR in case one was not sufficient) as well as sub-system pairs integrations. Another advantage of ROS was noticed when the shore station was connected to the ASV via wireless network thus eliminating the need to write Server-Client infrastructure from scratch. To lower risks and mitigate the development all off-the-shelf components integrated in the system are

[1] See reference 1

compatible with ROS and has built-in SDK's. This allowed the teams to focus on algorithm and system software development.

A. Navigate the Panama Canal

To demonstrate basic autonomous capabilities, the navigation task is mandatory and must be completed first. To accomplish this task the ASV should identify both gates and navigate through them, keeping linear course. Preparing for this task allowed us to develop and improve all the basic capabilities such as object detection, localization, control, and path planning. In addition, it allowed us to test the stability and reliability of the trimaran hull.

B. Magellan's Route

The obstacle channel requires the ASV to identify the next gate and pass through it without hitting obstacle and gate buoys. This task requires updating the path continually while keeping the ASV in a curvy lane. Using stereo lab camera with 120˚ FOV, the buoys are detected and classified. Once the nearest set of buoys detected, the ASV will navigate to the mean point between them. Practically, the course is divided into segments which makes the path planning easier, as only one point is targeted and less obstacles should be identified at once, allowing for better error correction.

C. Beaching & Inspecting Turtle Nests

The docking task requires the ASV to recognize docking stations based upon their colors and dock at the desired station while counting and reporting the number of "turtles" at the "nest" to the on-shore control station. The image-processing algorithm uses the data from the ZED-Camera to determine the layout of the stations.

D. Northern Passage Challenge

The ASV must pass through a set of buoys, identify the turn-around marker and maneuver around the marker and back through the buoys in the opposite direction through which it entered the task. The ASV uses the LiDAR and CV to identify the task and the state-machine sets the correct algorithm for the desired maneuvers.

E. Feed the Fish – Ball Cannon

This task requires identifying a target with three buckets and firing three balls, one into each of the buckets. This demonstrates the ASV's ability to sense and interact with its environment showing precise control, aiming and hull stability as the ASV should relatively stay in place. The ball cannon was designed, implemented, and tested in various conditions. The cannon has three barrels each loaded with a single racquetball and released by a solenoid-based propulsion system.

F. Ponce de Leon – Water Cannon

This task requires identifying a target and filling a tank until the ball reaches the marked line. It demonstrates precise control, aiming and hull stability as the ASV should relatively stay in place. The water cannon from the 2022 team was improved with a servo-based control system to aim the pitch and angle of the cannon. An improved detection and aiming algorithm was implemented.

G. Explore the Coral Reef

This task requires the ASV to autonomously navigate back to the starting point of the mission while avoiding all obstacles on the way. The state-machine recognizes the completion of all the tasks and activates the return-to-home algorithm.

III. DESIGN CREATIVITY

In the struggling 2023 market, we were unable to raise funds as done in previous years. With a very limited budget we were able to acquire few new components in order to improve the ASV from the 2022 model. Many of the improvements were completed in-house. We performed a cost benefit analysis on each design component to best utilize our very limited budget.

A. Hull & Propulsion Design

This year's model uses the same hull design of the 2022 model which is based on the trimaran design with two azimuth stern thrusters at 90˚

and one bow thruster embedded into the center hull. This allows the "crab-like" sailing which is useful for avoiding obstacles and maneuvering at low speeds. As most of the center hull is placed underwater, most of the weight is centralized thus allows less pitching. Furthermore, stability is gained also from submerging the side hulls and creating a predictable increase in righting moment. To meet this requirement, the vessel has a symmetrical design.

The hulls are made from foamed polystyrene, which was milled with a CNC machine. Foamed polystyrene was chosen due to its lightweight and ability to bear strikes. After producing the desired shape of the hull, it was coated with fiberglass, carbon fiber, and epoxy to keep it sealed and stiff. The deck of the ASV is made from layers of birch wood. To ensure the electronic systems on the ASV are cooled properly, the electronic box placed on deck is cooled by using a heat sink which are placed under the wooden deck and create and ideal convection.

B. Computer Vision

1. AI Interface Algorithm

For object detection we decided to use the real-time object detection algorithm model YOLOv4-tiny[2], which offers a good balance between speed and accuracy. The model was trained and deployed via darknet framework, which enables integration of the model with the rest of the system through ROS[3].

2. AI Training

To create our custom trained model, we assembled a dataset of the required objects as detailed in the competition rules. The dataset was composed of pictures from a variety of sources, including previous competitions and pictures we staged ourselves during test runs in the field. The dataset was then expanded by creating augmentation and different variations to simulate variance in angle, exposure, saturation, brightness, noise, etc. The dataset was split into three different sets: training set (70%), validation set (20%) and testing set (10%).

3. Deployment

After the model was trained, it was then integrated into the system such that it would process live images from the camera and augment them with bounding boxes. This data stream is fused with the camera's pointcloud to create a marker array detailing type, relative location, and certainty of detected objects.[4]

C. Range Estimation

As part of last year's conclusions, we planned to fuse the data from the stereo camera and the LiDAR system. The stereo camera is used to classify obstacles in the close surrounding and the LiDAR is used for both detecting obstacles from further range and improving the accuracy of the object range estimation. Throughout the work, we understood that we can get even more information by separating the two sensors' data collecting and processing. Hence, we decided to use the LiDAR to generate Laser Scan data for the navigation process and SLAM[5] (Hector)[6].

The LiDAR used is the InnovizOne LiDAR which is a solid-state sensor with 115˚x25˚ FOV and up to 250m Detection Range.

D. Localization

The MTI 680 INS by XSENS provides IMU and GNSS capabilities. The device is first configured to the team's needs. Using a provided software, parameters such as data type, data rates, signal to collect and more, are set. The raw data is processed using XSENS's ROS driver and the product is published to the system for further interpretation by other units.

Navigation unit utilizes GNSS and IMU data to determine the boat's location.

Control unit utilizes IMU data to allow precise motors control as determined by the navigation unit.

[2] See reference 2
[3] See appendix B.1
[4] See appendix B.2
[5] See reference 3
[6] See appendix B.3

Sensors unit utilizes IMU data to create a SLAM map.

E. Path Planning

First, the sensors team creates the SLAM map of the environment, using the G-mapping which is an open-source algorithm for building maps of indoor environments using a mobile robot equipped with a range sensor, such as a laser scanner. We take the SLAM map and send goal points to the start of each mission starting from Mission #1, where the ASV will navigate to autonomously using the "move_base" method, which is a navigation stack in the Robot Operating System (ROS) that provides a robust and flexible way to move a mobile robot from one place to another. The "move_base" package consists of several ROS nodes that work together to plan and execute robot motion. After we reach the goal points, the relevant mission code will begin executing, and by relying on the information received from the sensors in real time, [camera and the LiDAR], and by sending the needed velocities (of the goal points inside the mission) to the control team, we are able to execute the mission as desired.

F. Control

Nvidia Jetson AGX Orin is used as a Controller that receives angular and linear velocity according to data from different sensors (such as INS, camera, etc.). Linear velocity is extracted from the INS data by integration of the accelerations in different axes. In addition, the INS provides measurements of the changes in angles in two axes, which enables the calculation of the angular velocity. For the thrust configuration, BlueRobotics datasheet was used to establish the momentum generated by the engines in respect to Pulse Width Modulation (PWM) and voltage that are provided through Arduino Nano board.

G. Water Cannon

A cannon was designed based on reverse engineering of an RC jet-ski waterjet system. The following design[7] enables to deliver large amounts of water. The cannon has two main stages, called First Stage and Second Stage, where propellers are set in a water leak proof enclosure and driven by a shaft connected to a brushless motor. This brushless motor can rotate at very high RPM (Max. 40,560RPM) at voltage of 12V. It is located on the bow, above the camera and LiDAR on a variable degree adapter.

To obtain the task, the target is identified via the computer vison system described above. By conducting several experiments, we were able to determine the angle of the cannon[8], the right R.P.M and the distance from the target to fill the tank in the most efficient way. The cannon shoots in specific time intervals and readjusts the boat's location and the cannon's R.P.M according to the target's updated location. This strategy was chosen to avoid the use of complex computer vision and delicate angle changes.

H. Autonomy

The autonomous function of the ASV is governed by a state machine[9], managing the flow and transition between competition tasks, and monitoring the various onboard systems to detect and handle hazards.

Provided that all start-up checks have passed, and we have successfully reached our desired starting point (e.g., 6 ft from the mandatory navigation channel), the ASV transitions to autonomous mode. In this mode, the state reflects the current task, with the basic flow of: (a) locate task starting point (for example, gate buoys). (b) perform task; More complex tasks, such as the Water Blast, are divided into subtasks, each represented by its own state. (c) Move to next task according to our competition strategy; location is predefined using the provided course configuration. In addition to the regular flow, there are flows dedicated to hazard handling and recovery, for example in the case of a malfunctioning

[7] See appendix B.6
[8] See appendix B.7
[9] See appendix B.8

sensor. Finally, there is a general "return to home" state.

For implementing the state machine, we looked for a framework that natively supports integration with ROS, since the state machine should communicate with the other ASV components over a distributed ROS network. As the state machine is a new addition to the ASV, we did not have a previous implementation to build on, and it was important for us to focus our efforts on developing the state machine itself rather than on drivers, APIs, or possible compatibility issues. Additionally, we preferred tools that allow for testing and simulation, in both isolated and real-world environments, with relative simplicity.

Following initial difficulties with Gazebo, we decided to use Stateflow and Simulink from MathWorks, who also kindly offered us guidance through the first steps of the project. Stateflow supports many features for state machine design, including global variables, code and graphical functions, timeout-based state transitions and more, all in a graphical interface. This enabled us to focus on the underlying logic and design a more robust state machine, that maintains autonomous control for as long as possible. Moreover, the MathWorks ecosystem includes a complete ROS toolbox that can generate C++ code for a ROS node running our state machine. Besides significantly reducing coding time, running on C++ is of critical importance, since our state machine is intended to be run on an Nvidia Jetson Xavier NX and is required to operate in real-time.

I. Communication

For the physical network, we used a communication switchboard, which all the ASV computers are connected via Ethernet cables (802.3) and the shore station's computer is connected via a closed WIFI 2.4Ghz (802.11) network[10].

For software implementation, the Rosmaster package[11] was used. With Rosmaster, one computer is configured as the master in all the computers that are connected to the network, thus all ROS nodes and topics are shared between all on deck and shore computers[12]. Since most of the modules on the ASV already use ROS as their platform to publish information, the connection and integration were immediate.

A communication package was coded that allows recording sensors' data, present real-time data of system health in shore station and perform emergency shut down.

J. Simulation

The ASV simulator is based on the "gazebo" simulator, it is a dedicated model for robotic systems that includes a high level of detail including physical behaviors such as friction and buoyancy. In the simulator, we mounted engines and sensors on the boat model as they exist on the ASV and performed calibration as it exists in reality. We built the different routes by modules of floats and surfaces. The simulator interpolates data from the simulative sensors according to the simulated environment in relation to the boat, the boat performs the entire navigation and control process as designed, while at the same time publishes back to the simulator the commands of the engine that drives the boat in the simulator. In order to carry out this connectivity, we created a "Publish" function for all the sensors and implanted a similar function in the control code that encrypts the motors' command after converting to a format [as well as units] which are suitable for the simulator.

IV. EXPERIMENTAL RESULTS

The very first experiments we conducted involved reinstating the 2022 model from last year's competition. We examined the different systems during basic lab experimentation and

[10] See appendix B.9
[11] See reference 7
[12] See appendix B.10

determined the necessary improvements and adjustments.

After completing the laboratory experimentations, we completed the first in-water experiment and test run based upon the 2022 model.

Multiple in-water tests were done in the university swimming pool as well as the TLV Lake while each time testing more advanced implementations of the various systems as well as the ASV as whole. The goals of the experiments were set by the progress of each team individually as well as the continuously updated competition guidelines.

The majority of our experimentations were conducted in the laboratory, where we learnt that it is easier and more effective to tackle issues that arise.

The final stage of experimentations were completed using the new simulation software developed for this purpose.

A. Object Detection

During both lab and field experiments photos of all obstacles were taken in different environment conditions to expand database. The system generates a consistent live data stream of 7 marker types, with stable speeds averaging at 24 FPS, and mean average precision of 86.29%. This is an improvement of previous year's CV team, which achieved identification of 4 obstacle types at mAP of 68%, at higher speeds of 30 FPS.

B. Path Planner

To verify the correctness of the algorithm simulations in RVIZ[13] and Gazebo were used. Using simulation allowed parallel development of the path planner, independent from the platform and control development. The characteristics of the platform and the missions' elements was injected to the simulator. This allowed easier debugging.

C. Floating Platform

As the hull was custom made an experiment to ensure buoyancy and stability was conducted. The hull was tilted and loaded with weights. We found that the hull is very much stable in all directions and can carry up to 30 kg.

D. Communication

As there are multiple computers on deck we used the rosmaster protocol. In the experiment we published multiple messages to ensure that all the computers are subscribed to the relevant topics and identify the master computer. we found that all CPUs can communicate and there is neglected latency in retrieving data. This allows better control and easy data transferring.

V. ACKNOWLEDGMENTS

We would like to thank Tel Aviv University's Faculty of Engineering and to Dean Prof. Noam Eliaz for allowing us to participate in this unique project, and for offering us the necessary support and resources.

Special thanks to Danny Berko and Simcha Leibovich, our academic directors, and Tel Aviv University faculty members, who believed in the project and helped make all our ideas come true.

Thanks to all our professional mentors for offering a lot of knowledge, experience, and support: Eitan Avisar, Roi Raich, and Jacob Fainguelernt.

A special thanks to MOVELLA for sponsoring our project with a XSENSE INU, as well as INNOVIZ Technologies and Lake TLV.

VI. RFERENCES

1. Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng, "ROS: an open-source Robot Operating System", ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

2. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv:2004.10934, 2020.

3. Robot Mapping course [online], 2013.

4. robot_localization Package Summary, ROS Wiki.

5. Huijuan Wang, Yuan Yu, Quanbo Yuan, "Application of Dijkstra algorithm in robot path-planning", 2011 Second International Conference on

[13] See appendix B.11

Mechanic Automation and Control Engineering, 2011.
6. dwa_local_planner Package Summary, ROS Wiki.
7. rosmaster Package Summary, ROS Wiki.
8. BlueRobotics T200 specs [online].

Appendix A - Component Specifications

Component	Vendor	Model/Type	Specifications	Custom/ Purchased	Cost [$]	Year of Purchase
ASV Hull Form/Platform	nanofiber	Custom	foamed polystyrene covered fiberglass, carbon fiber, and epoxy.	Custom	2,507	2022
Waterproof Connectors	MPH Electrical Engineering	EKPK 180 G 2538055 outdoor junction boxes		Purchased	111	2022
Propulsion	BlueRobotics	T200	T200 Specs	Purchased	210	2022
Power System	Fullymax	LIPO 14.8V 5000mAh		Purchased	144	2022
Power System	Fullymax	LIPO 14.8V 7500mAh		Purchased	212	2022
Power System	Fullymax	LIPO 18.5V 5750mAh		Purchased	206	2022
Power System	Fullymax	LIPO 7.4V 7500mAh		Purchased	123	2021
CPU	Nvidia	Jetson AGX Orin Developer Kit	Jetson Orin Datasheet	Sponsored		2021
Motor Controls	SeeedStudio	PCA9685	PCA specs	Purchased	20	2022
Teleoperation	Ubiquti	Bullet M5	Bullet M5 specs	Purchased	115	2021
GNSS/INS	Movella	XSENS MTi-680	Xsens MTi-680	Purchased	3000	2023
Camera	StereoLabs	ZED2	Camera specs	Purchased	500	2021
LiDAR	Innoviz	InnovizOne	LiDAR specs	Sponsored		2022
Water Cannon		Custom, self-production				2022, 2023
Ball Cannon		Custom, self-production				2023
Algorithms						
Vision		YOLOv4 and darknet				2022
Localization and Mapping		Custom based on Dijkstra, DMW ROS package and robot_localization package		Custom		2022
Autonomy	MathWorks	Simulink				2022
Communication		Custom based on rosmaster package				2022
Open-Source Software		ROS				2022

Appendix B

1.

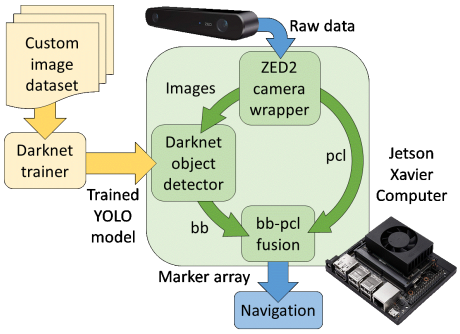

Figure 1 - CV Interface Architecture

2.

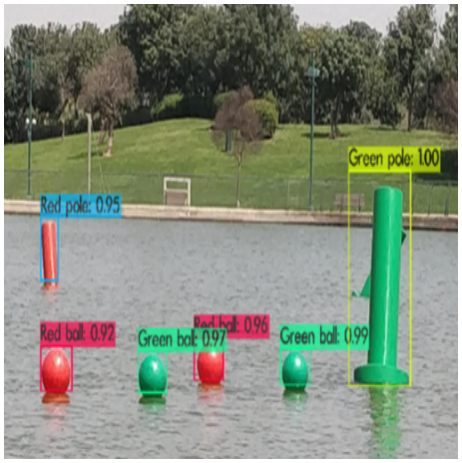

Figure 2 - Bounding Boxes / Real-Time Accuracy

3.

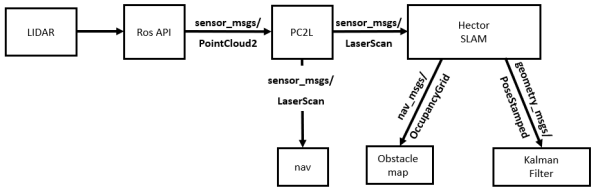

Figure 3 - LiDAR Interface Architecture

Pc2l – gets as input the sensor point cloud. converting it to 2D, filtering the relevant FOV (angle and height, mainly filtering out the water), and publishing laser scan message.
Hector SLAM – gets as input the laser scan and uses it to create an obstacle map (Occupancy Grid) of the environment and to estimate the boat's movement and its position on the map.

4.

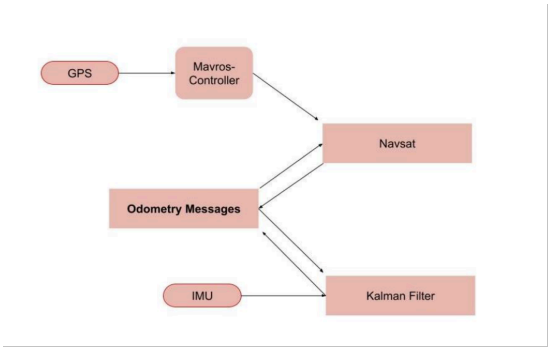

Figure 4 - Localization Interface Architecture

5.

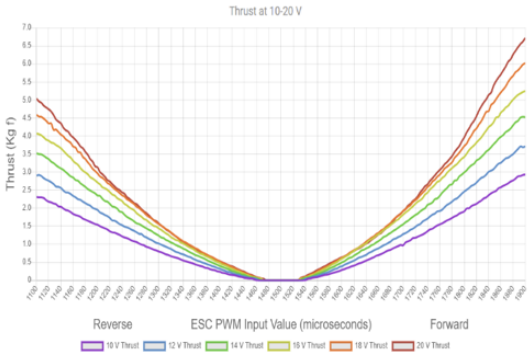

Figure 5 - Momentum vs. PWM[14]

6.

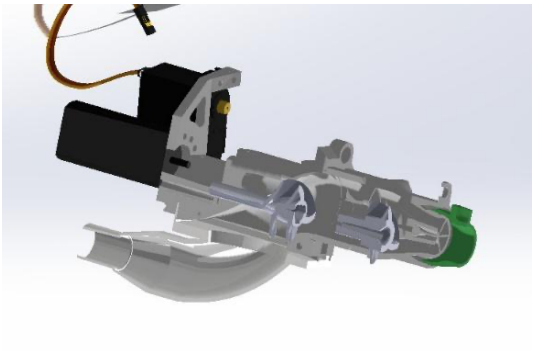

Figure 6 - Water Cannon Design

[14] See reference 8

7.

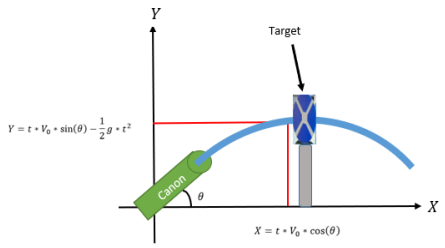

Figure 7 - Water Cannon Angle Calculation

8.

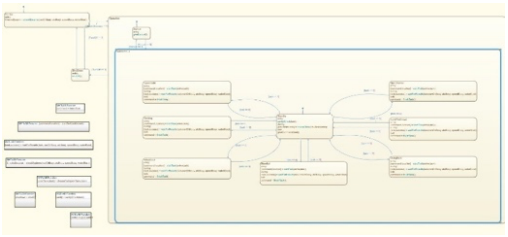

Figure 8 - State Machine

9.

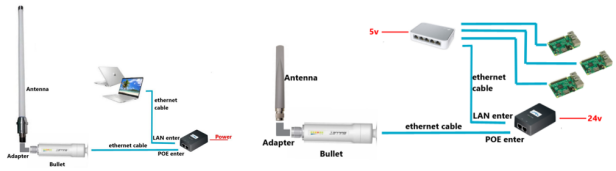

Figure 9 – Communication Interface Architecture

10.

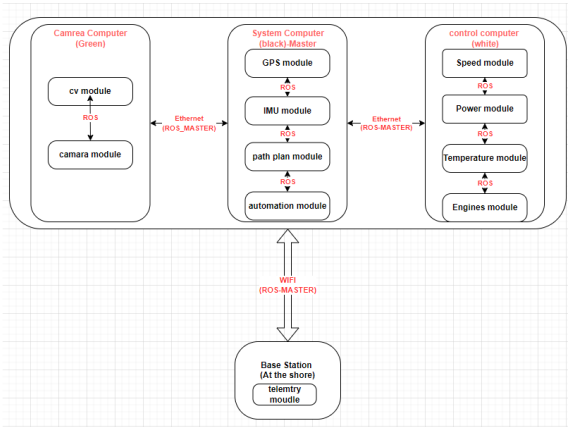

Figure 10 - System Deployment

11.

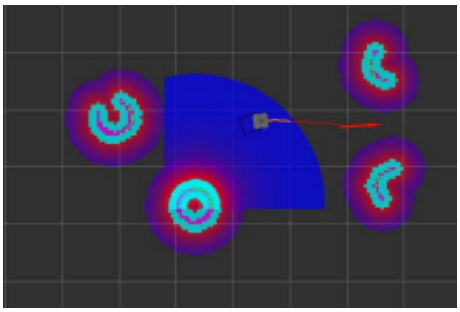

Figure 11 - RVIZ Simulation

12.

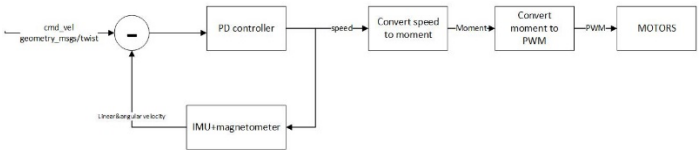

Figure 12 - Control Interface Architecture

13.

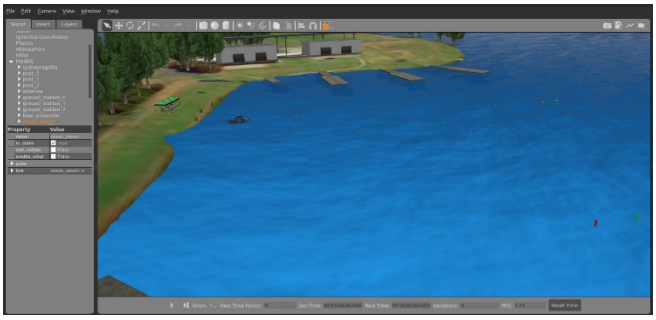

Figure 13 - Simulation "Gazebo"

14.

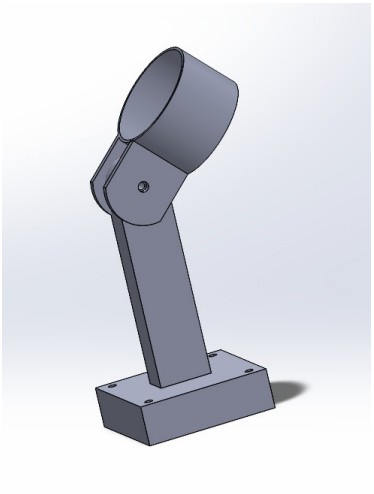

Figure 14 - Ball Canon Mount

15.

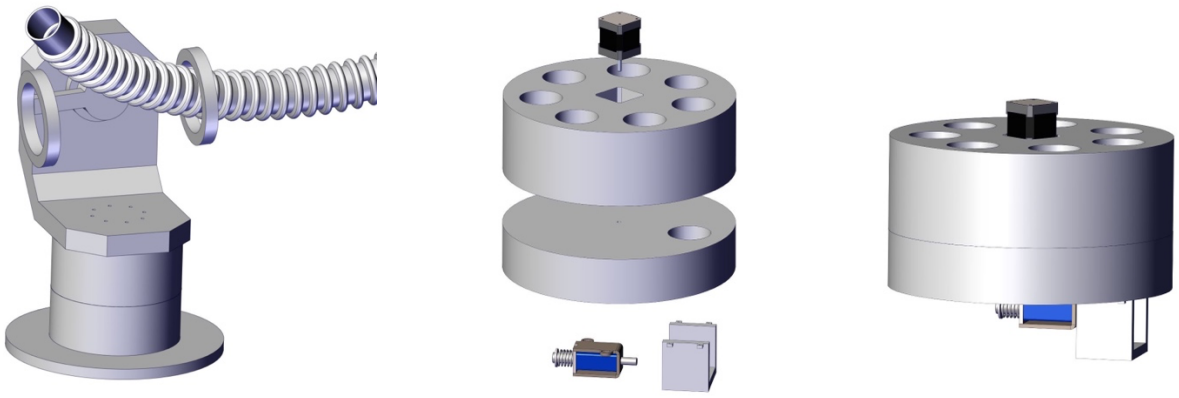

Figure 15 – Water Cannon Mount

Appendix C – System Architecture

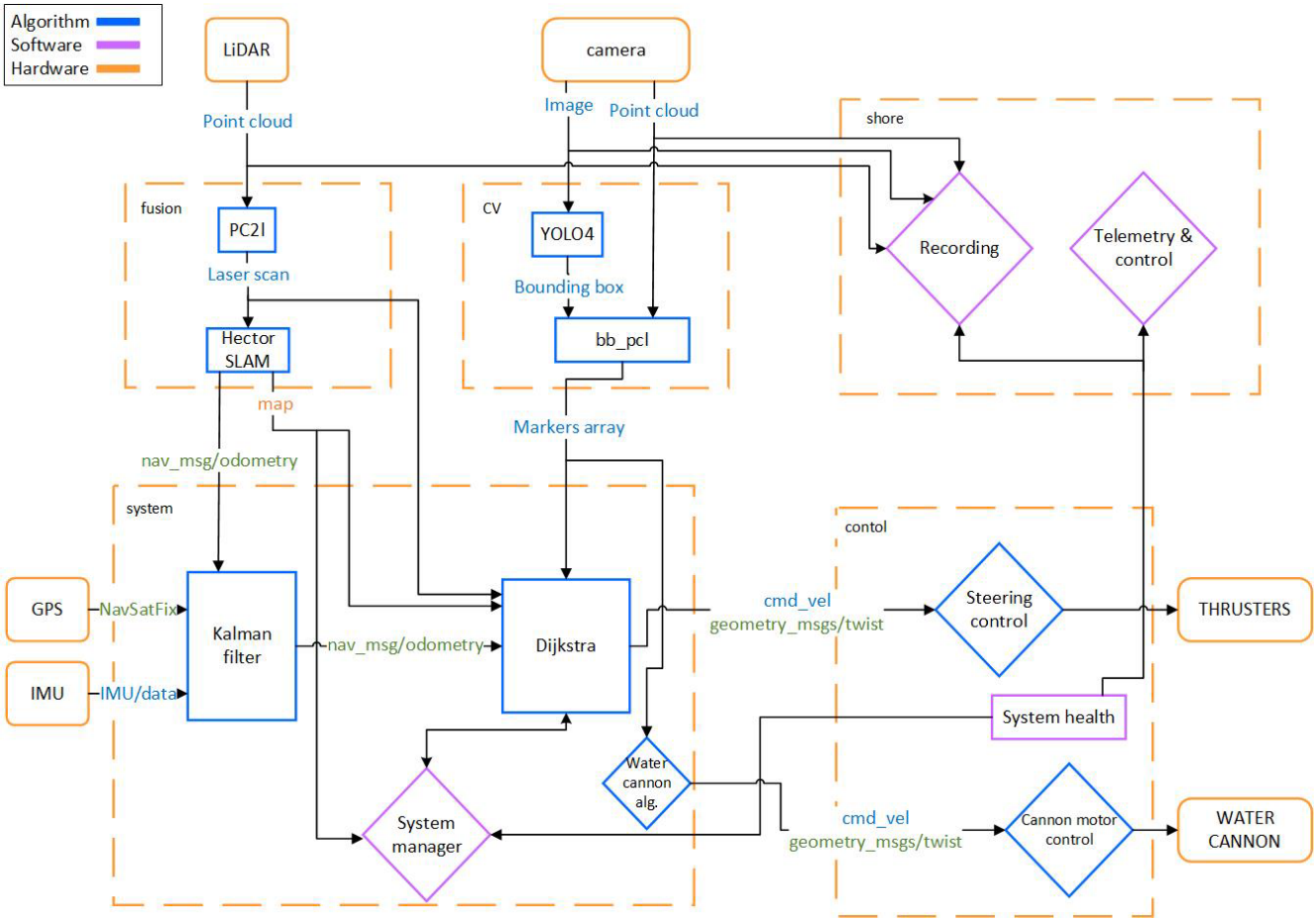

Figure 16 System architecture

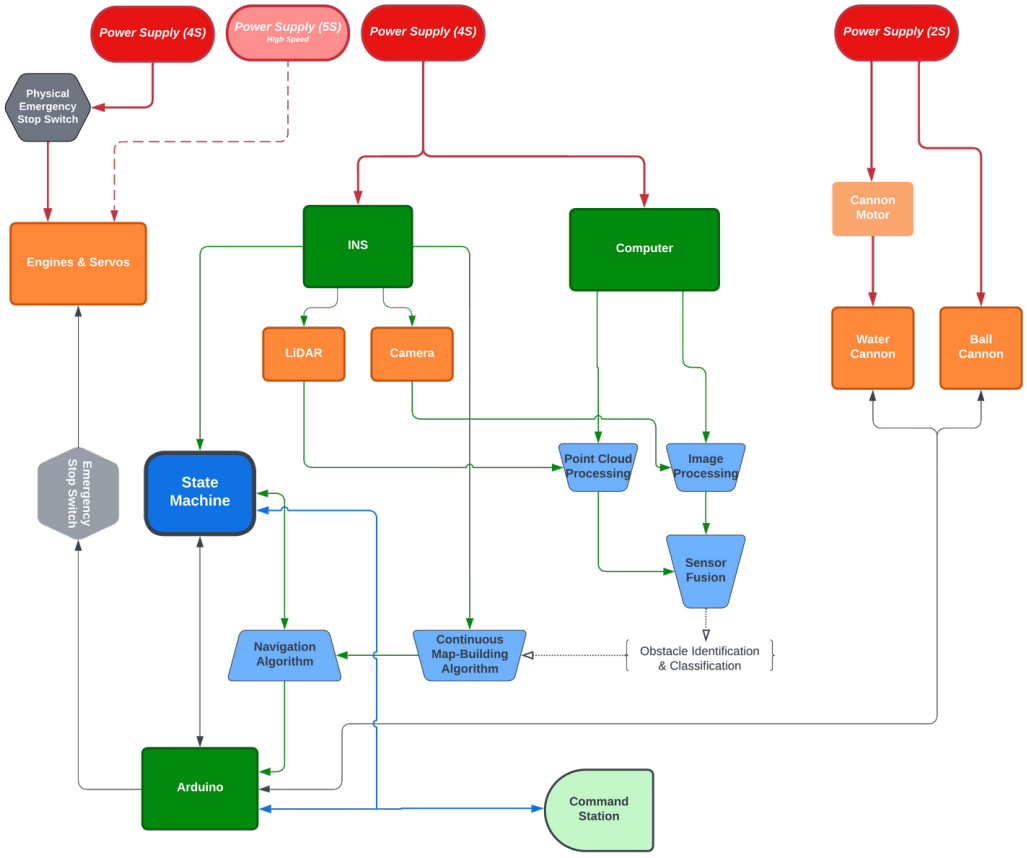

Figure 17 - System Block Diagram

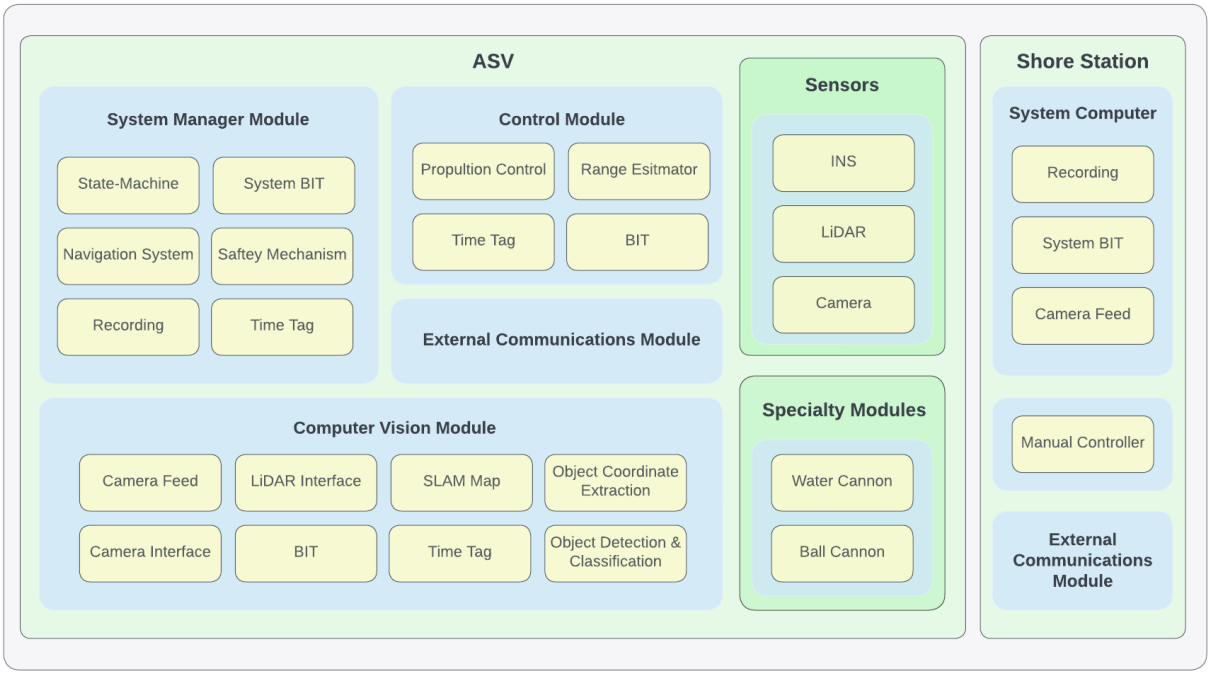

Figure 18 - System Modules