RoboBoat 2023: Technical Design Report SimLE SeaSentinel Team

Igor Rusiecki, Norbert Szulc, Klaudia Głowacka, Cezary Wieczorkowski, Tomasz Ujazdowski, Igor Baranowski, Franciszek Górski, Piotr Stroiński, Jakub Wilk, Patryk Sobolewski. Maciej Zawadzki, Maciej Cieślak, Marcin Waryś, Francisco Osuna Chudzio, Wiktoria Piech, Karol Rzepiński, Jan Sontowski, Piotr Wykowski, Michał Ganczarenko, Andrzej Pilguj, Michał Biczkowski
Gdańsk University of Technology, Poland, Gdańsk

***Abstract*—This report discusses the strategy of the SimLE SeaSentinel team for RoboBoat 2023 and the design of our Autonomous Surface Vehicle (ASV) named ASV Perkoz. It's a modular, mostly 3D-printed vessel equipped with a robotic arm. It was designed as an easily transportable platform capable of autonomous navigation and performing tasks during the RoboBoat 2023 competition.**

***Index Terms*—component, formatting, style, robot operating system, behavioural trees**

ACRONYMS AND ABBREVIATIONS

ASV Autonomous Surface Vehicle
VCU Vehicle Control Unit, a.k.a. Flight Control Unit
FOV Field of View
OCS Operator Control Station
OBP Onboard Processing
GS Ground Segment
DOF Degrees of Freedom
SBC Single Board Computer
ROS Robot Operating System

I. COMPETITION GOALS

A. General Strategy

As a first-year team, we decided to reduce the scope of our project by attempting only six out of eight tasks during the RoboBoat 2023 competition. Due to the limited number of team members in the early phases of the development and the fact that the competition takes place in March instead of June, which greatly reduced the time in which we had to design our Autonomous Surface Vehicle, we decided to limit the number of iterations in our design spiral to an absolute minimum. Although such an approach came with a high risk of failure, it allowed us to meet deadlines and thus participate in the competition.

In order to set our priorities and formulate requirements for ASV Perkoz we decided to use the MoSCoW method [1]. We identified requirements related to Tasks 1, 2, 4, and 8 as "must-have", requirements related to Tasks 3 and 7 as "should have", some basic Task 6 related requirements as "could have" and finally we identified Task 5 and advanced Task 6 related requirements as "won't have".

Gdańsk University of Technology, Ministry of Education and Science

From the beginning of our work, we have put a lot of emphasis on modularity. This concept, despite its controversial use on larger vessels, can be applied with great success on smaller vessels like the ones built with RoboBoat competitions in mind. Capabilities resulting from our ASV's modular design are crucial for our logistics. We plan to bring our ASV together with us in our flight luggage, instead of sending it as a parcel. Such a decision gave us additional three weeks of time for preparation and testing.

B. Course Strategy

Firstly, the ASV will attempt the mandatory task called Navigate the Panama Canal. For every task, the ASV's motion control will be based on computer vision and the decision-making process will be supported by a behavioral tree.

After completing the first task, the ASV will detect the pair of green and red buoys, which will indicate the end of Task 1 and the beginning of Task 2 – Magellan's Route / Count the Manatees & Jellyfish. Object avoidance will be based on image acquisition and processing, and the ASV will be suggested to move to the nearest "clear" segment. Furthermore, distances to objects will be calculated based on the data provided by three Oak-D stereo cameras. Total Field of View (FOV) will be 180 degrees. To minimize the risk of the ASV getting lost, we will save its GPS coordinates before and after every task. In case of being unable to detect any desired object, ASV will return to the last saved position.

For Task 3 – Beaching & Inspecting Turtle Nests – we are training our model to detect nests with different numbers of dots (1-6) in three colors (green, blue and red). The ASV will detect the object based on the given color, then recognize and report the number of "eggs". To dock, we will use path planning and object avoidance algorithms.

While preparing our strategy for Task 4 – Northern Passage Challenge – and after analyzing videos from previous editions of RoboBoat we have come to a conclusion that the main deciding factor for the success is not speed but the ability to fluently and without any disturbance turn around the blue buoy. This is why we didn't identify the speed of our vehicle to be a crucial parameter although our thrusters can generate up to 44 N of thrust each [7].

The core of our strategy for Task 7 – Ponce de Leon / Fountain of Youth – is the use of our custom robotic arm that will reach to the target and then start pumping water into it. In order to successfully execute Task 7 this way we needed to ensure that we will be able to maintain a stable position and minimize the heel when our robotic arm is working. For this reason we equipped our ASV with a four-thrusters propulsion system to enhance our capabilities regarding dynamic positioning and with a bulb keel that will improve mass displacement and thus provide enough stability.

II. Design Strategy

Since the beginning of this project, we have adhered to the principles of systems engineering. This has enabled us to divide the project into subsystems and components, which have been assigned to team members for implementation. Each top-level system has been defined with a specific role and function, according to single responsibility principle. We have chosen to split our work into the following systems:

- Mechanical – responsible for hull design, hydrodynamics, and task-specific modules.
- Propulsion – motor placement attachment and driving.
- Electrical – all the cabling, batteries, power, and integration of Vehicle Control Unit (VCU) (a. k. a. Flight Controller) with sensors and motors.
- Onboard Processing – application domain, high-level command over VCU, object detection, decision-making, task-specific algorithms.
- Ground Segment – everything that won't be on the ASV, including the Operator Control Station (OCS).

Their structure and components have been reflected within our Work Breakdown Structure**??**. We will skip some of the components to emphasize the creative aspects of the system.

A. Hull design

When designing our hull, we needed to find a compromise between the stability and simplicity of our construction. Stability was important as it decreased disturbance to the camera view, which improved our object detection, identification, and location capabilities [2]. It was also absolutely crucial for our strategy regarding Task 7 since our robotic arm can generate up to 12 Nm of torque which needed to be adequately countered. The latter was imposed on us by the limited time-frame, which forced us to make our hull as simple to manufacture as possible.

As a result, we decided to create a modular flat bottom hull made of XPS sheets. On top of it is a frame to which 3D-printed hull modules are attached. A PVC mast for cameras and antennas is located at the stern. At the bow, we located our robotic arm.

B. Propulsion

We decided to equip ASV Perkoz with a differential thrust propulsion system with the use of four Blue Robotics T200 Thrusters. Two at the bow and two at the stern. Each pair is angled in a way that diverts water streams to the sides of the hull rather than under it. Such a design allows movement in four Degrees of Freedom (DOF) by changing the amount of thrust generated by a given thruster. It also should enhance our ASV's capabilities regarding dynamic positioning, which will be benefitial when executing Task 7.

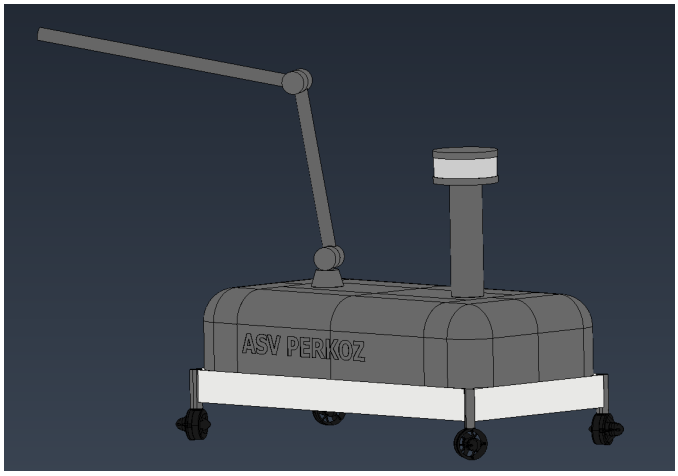


Fig. 1. Hull concept

In order to control them we are extending existing PX4 autopilot firmware [5]. By writing a custom controller, we are essentially porting a similar feature from ArduRover [6]. We hope to open-source it and contribute our work to the PX4 ecosystem. Unfortunately as it is still unstable, as a fallback we consider the aforementioned ArduRover. Because both VCU firmware alternatives are abstracted through Mavlink protocol, we can risk the novel propulsion approach.

C. Robotic arm as a water delivery system

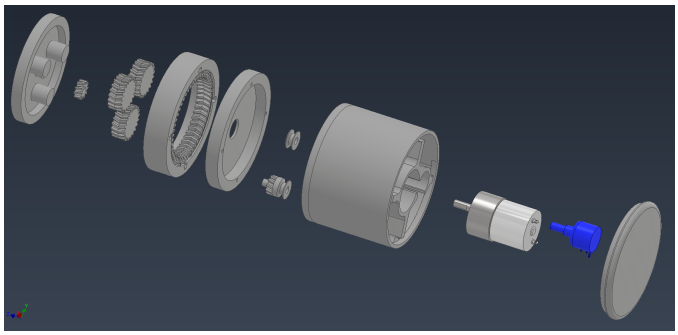


Fig. 2. Actuator Design

As part of our strategy for Task 7, we designed a robotic arm inspired by an open-source project [3]. The arm has four DOF, and it has a working range of up to 1.3 meters. All joints are 3D printed and driven by DC motors. Links are made of PVC pipes. Along the robotic arm runs a hose through which water, pumped by two diaphragm pumps, will be delivered to the target.

For driving, we will use inverse kinematics [4]. This method will be coupled with the camera feedback. As a fallback we

will hard-code the desired arm position and lock it while executing the task, disregarding camera feedback.

D. System Bus

We decided to interconnect subsystems on our vehicle using a CAN bus [9]. This technology is an industry standard that provides very robust communication. To ensure scalability of our system and to streamline development of new subsystems, we decided to utilize Cyphal communication protocol [10]. This protocol was built with autonomous vehicles in mind, it is simple and does not require much processing power from the hardware. Furthermore, it is decentralized, which means that all nodes are equal and failure of any one of them does not interrupt communication between the other ones. Cyphal also allows us to focus more on developing service-oriented interfaces between components by abstracting away lower level subjects needed for inter system communication. System bus connects to the Electrical power supply, Visual Feedback System, VCU, Single Board Computer (SBC) and robotic arm. We are hoping to integrate all newly developed systems into this environment.

E. Electrical power supply

Our ASV uses Li-Po batteries as a source of power and has a specially designed safety system that includes a BMS for safe charging/discharging and a custom power controller for power distribution, monitoring and emergency shutdown. Monitoring data is exposed over CAN bus. Remote emergency shutdown uses its own separate radio module.

F. Position and attitude determination

GPS RTK (Real-Time Kinematic) is a high-precision positioning technology that uses differential correction to improve the accuracy of GPS measurements by several orders of magnitude, allowing for centimeter-level accuracy in real-time. As we had readily available GPS RTK modules, we will rely on them for precise positioning. Additional data will be fed passively to ASV from OCS during it's autonomous operation, as it has GPS RTK base station.

To increase accuracy of heading determination we will attempt to use second GPS module to supplement VCUs compass readings. This second module will be places along ASVs axis, creating a heading vector.

G. Computer Vision

We are using a vision heavy approach, as we resigned from integrating LiDAR. We were unable to obtain an affordable device that would perform well in direct sunlight and on a pitching ASV. This pushed us to use multiple cameras totaling 180 degress of FOV. The object detection architecture we selected is YOLOv7 [11], the latest edition of the YOLO family [12]. It has been developed specifically for real-time object detection tasks and boasts state-of-the-art performance. In order to reduce computation resources, we opted to utilize a pre-trained tiny version of YOLOv7 with fewer trainable parameters. This is required due to constrained resources on our chosen SBC: Jetson Nano. It has been trained using publicly available data [13], primarily sourced from the Roboflow service, which is compatible with the YOLOv7 architecture. We hope to syntesize more training data through simulation in the near future.

We are utilizing stereo imaging for distance detection, due to the availability of multiple OAK-D cameras [14] at a cost-effective price compared to LiDAR. These cameras feature an AI coprocessor, as well as color and stereo vision capabilities. The color data is fed to the YOLO model running on the SBC, while the stereo data is processed on the cameras themselves. This allows the SBC to be unburdened of additional workload and have more resources available for autonomy. Both object detection and distance data are transmitted as Robot Operating System (ROS) topics.

H. Autonomous Navigation Software Architecture

The ASV's decision-making system is built with Behavioural Trees [8] and the ROS platform [15]. ROS provides libraries for multiple sensors and actuators, and allows for data acquisition and digital filtering. The system is divided into a higher-level control layer for strategy development and a lower-level layer for communicating with peripherals and VCU [17]. This node-based design facilitates testing and enhances code clarity, while making the system flexible and expandable. The use of ROS aligns with the project's goal of creating a modular and scalable ASV that can operate even with individual node failures. We are using rich ROS package ecosystem and only develop the core functionality. Use of PX4 as autopilot on VCU enables use of PX4-Avoidance package [16] that exposes autopilots low-level avoidance interface to ROS

I. Telemetry and remote shutdown

We are using two to three bands: 433 MHz, 2.4 and 5 GHz, fig. C-B. Standard PixHawk compatible frequency hopping radios will be used on ISM 433 MHz band [18]. This eases out legal requirements as we can use it both in the EU and and the US. This will be our primary radio link for Mavlink communication between OCS. We tried developing a novel approach with custom radios with narrow band modulations or LoRa technology, as it would decrease interference with other teams. Unfortunately deadline shift decided against it. Remote shutdown also is also placed on the same band. As required, uses its own radio with simpler modulation.

WiFi will be used for video feeds and as a secondary Mavlink radio link for PixHawk VCU. For OCS we have procured Federal Communications Commission certified, highly configurable router with a directional dual band antenna with 60 degrees FOV. It's characteristic will cover one course area without much of a problem. We will focus on 5 GHz band as it has shorter range and is still underutilized. This should allow us to share WiFi spectrum with other teams, as we can fine tune our radio presence. 2.4 GHz band will be avoided as its crowded and has much bigger range, it will be used only

as fallback. For ASV side simple 5 GHz WiFi dongle with dipole antenna is satisfactory.

J. Ground Segment

OCS will be composed of another SBC, this time Raspberry Pi 4. It will be connected to telemetry radio and local network bridged to ASV over WiFi. On this SBC will be setup telemetry logging and proxy. Also to this SBC is connected GPS RTK base station. This approach enables multiple operators with different ground control software to connect to a single vessel. Although QGroundControl already enables Mavlink proxy, a more robust setup can be achieved on SBC.

For manual control gamepad connected to QGroundControl should be sufficient.

III. Testing Strategy

With a long procurement time for parts, we have started with a simulated environment. We utilized the rich PX4 ecosystem and readily available docker containers [20]. This allowed more team members to actively develop and test Onboard Processing (OBP), without need to be locked to dependencies from year 2018. We want simulation to be a constant factor in the project cycle. Because of that we have leveraged a modern cloud solution: a local Kubernetes cluster, to create continuous integration (CI). This Kubernetes instance is running the same docker containers as our developers. This CI tests both VCU movement controller and OBP behavioral trees coupled with simulated view from stereo and color camera. One big advantage of running a local cluster is the ability to run containers on custom hardware, like our chosen SBC: Nvidia Jetson. We can run vision processing on real hardware to check processing power budget. Development of this CI has been cut short by competition date shift and relative complexity of Kubernetes setup.

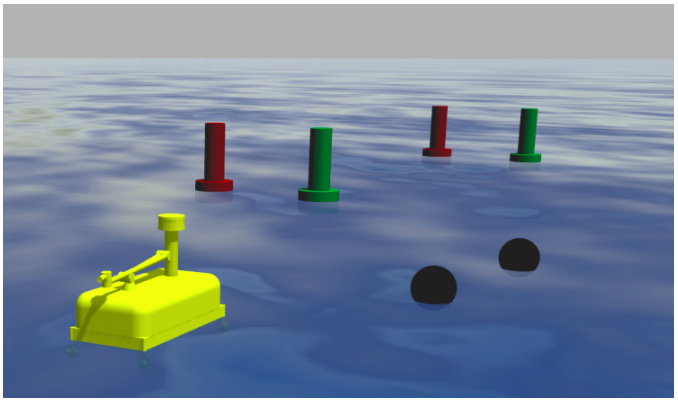


Fig. 3. Gazebo Simulation Environment

For hardware testing, we tried to import techniques from the CubeSat field. FlatSat [19] is an approach where subsystems are laid out and connected to communication and power buses on a test bench. This approach allows us to test every new subsystem with the rest of the systems in an environment that is very similar to its final deployment. Another benefit is that with everything laid out, it is much easier to troubleshoot or make adjustments rather than when everything is packed in the hull.

Acknowledgment

SimLE SeaSentinel Team would like to thank:
Wiktor Sieklicki, PhD for being our factulty advisor and mentor,
Jakub Zdroik, MSc for his help and advise regarding technical aspects of our ASV,
Henryk Lasota, PhD and Piotr Cywiński, MSc for their guidance during the early phases of our project,
Jerzy Temkowicz, PhD for,
Hackerspace Pomerania community for sharing components, tools and their knowledge,
ICETEK sp. z o.o. for sharing knowledge about cloud computing and giving access to kubernetes cluster
F.D.C. Willard for long and contributing discussions.

References

[1] A Guide to the Business Analysis Body of Knowledge, International Institute of Business Analysis, 2009, ISBN 978-0-9811292-1-1.
[2] L. Ren, H. Yin, W. Ge and Q. Meng, "Environment Influences on Uncertainty of Object Detection for Automated Driving Systems," 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019, pp. 1-5, doi: 10.1109/CISP-BMEI48845.2019.8965948.
[3] chilipeppr/robot-actuator-esp32-v8: This is the repo for the robot actuator v8 based on an ESP32 as the brains of each robot arm actuator. https://github.com/chilipeppr/robot-actuator-esp32-v8
[4] Inverse Kinematics – Modeling, Motion Planning, and Control of Manipulators and Mobile Robots https://opentextbooks.clemson.edu/wangrobotics/chapter/inverse-kinematics/
[5] Open Source Autopilot for Drones - PX4 Autopilot, Retrived from https://px4.io/
[6] ArduPilot - Versatile, Trusted, Open, Retrived from https://ardupilot.org/
[7] T200 Thruster: ROV thruster for marine robotics propulsion, Retrived from https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/
[8] BehaviorTree/BehaviorTree.CPP: Library in C++. Batteries included, Retrieved from http://github.com/BehaviorTree/BehaviorTree.CPP
[9] ISO - ISO 11898-1:2015 - Road vehicles — Controller area network (CAN), Retrieved from http://www.iso.org/standard/63648.html
[10] OpenCyphal Specification v1.0-beta – OpenCyphal Development Team, Retrieved from http://opencyphal.org/specification/Cyphal_Specification.pdf
[11] C.-Y. Wang, A. Bochowsky, M.L. Hong-Yuan "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors" arXiv:2004.10934
[12] Redmon, Joseph (2016). "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv:1506.02640
[13] RobotFlow dataset, Kriti Gupta, 2021, Retrieved from http://universe.roboflow.com/kriti-gupta/setup
[14] OAK-D — DepthAI Hardware Documentation 1.0.0 documentation, Retrieved from http://docs.luxonis.com/projects/hardware/en/latest/pages/BW1098OAK.html
[15] Stanford Artificial Intelligence Laboratory et al. (2018). Robotic Operating System. Retrieved from https://www.ros.org
[16] PX4 avoidance ROS node for obstacle detection and avoidance. Retrived from https://github.com/PX4/PX4-Avoidance
[17] MAVROS: MAVLink to ROS gateway with proxy for Ground Control Station Retrived from https://github.com/mavlink/mavros
[18] SiK Radio — PX4 User Guide Retrived from: https://docs.px4.io/main/en/telemetry/sik_radio.html

[19] Reilly, Jack & Murphy, David & Doyle, Maeve & Walsh, Sarah & Akarapu, Sai Krishna Reddy & de Faoite, Daithí & Dunwoody, Rachel & Erkal, Jessica & Finneran, Gabriel & Mangan, Joseph & Marshall, Fergal & Salmon, Lána & Somers, Eoghan & Thompson, Joseph & Ulyanov, Alexey & Hanlon, Lorraine & McKeown, David & O'Connor, William & Wall, Ronan & McBreen, Sheila. (2022). EIRFLAT-1: A FlatSat platform for the development and testing of the 2U CubeSat EIRSAT-1. 10.5821/conference-9788419184405.113.
[20] Docker overview https://docs.docker.com/get-started/overview/
[21] Development containers https://containers.dev/

Appendix A – Component List

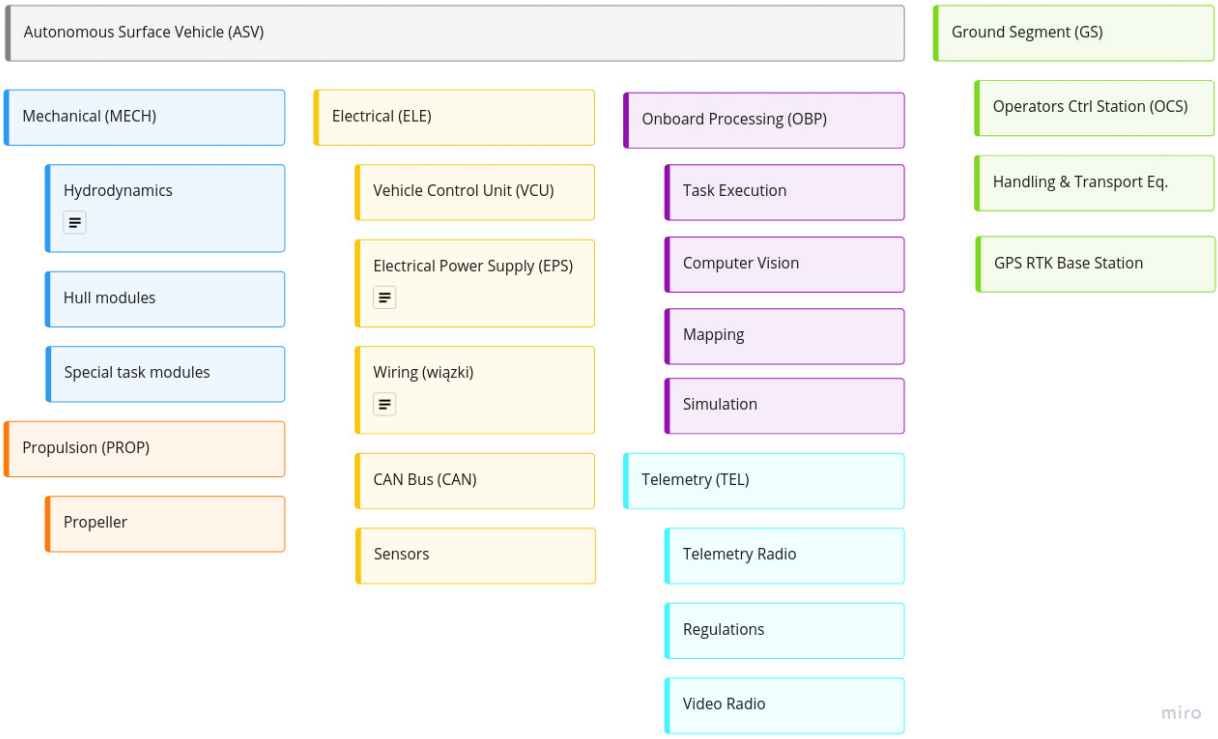
Component	Vendor	Model	Specs	Custom/ Purchased	Cost	Year of Purchase
ASV Hull	SRAD	Flat bottom hull	B=60cm, L=120cm, H=40cm	Custom	250$	2023
Waterproof connectors	Bulgin	PXP7010/02P/ST/1113	https://www.bulgin.com/products/pub/media/import/attachments/7000_power.pdf	Purchased	$24	2022
Propulsion	Blue robotics	T200	https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/	Purchased	$236	2022
Power system	Redox	Li-Po 4400mAh 30C 4s 14,8V	https://botland.com.pl/akumulatory-li-pol-4s-148v/8475-pakiet-li-pol-redox-4400mah-30c-4s-148v-5903754001277.html	Purchased	$58	2022
Motor controls	Blue robotics	Basic ESC	https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/	Purchased	$36	2022
Processing computer	Nvidia	Jetson Nano B01	https://developer.nvidia.com/embedded/jetson-modules	Purchased	$340	2023
Teleoperation	Holybro	SiK V3	http://www.holybro.com/product/transceiver-telemetry-radio-v3/	Purchased	$180	2023
Cameras	Luxonis	OAK-D	https://store.opencv.ai/products/oak-d	Purchased	$249	2022
Vehicle control unit	Holybro	Pix32 v6	https://shop.holybro.com/pix32-v6_p1338.html?	Purchased	$340	2022
GPS	SparkFun	GPS-RTK2	https://www.sparkfun.com/products/15136	Purchased	$275	2023
Vision		Yolo v7	https://github.com/WongKinYiu/yolov7	-	-	-
Autonomy	Open robotics	ROS	https://www.ros.org/			
Open source software	107-Systems	107-Arduino-MCP2515	https://github.com/107-systems/107-Arduino-MCP2515	-	-	-
Open source software	107-Systems	107-Arduino-Cyphal	https://github.com/107-systems/107-Arduino-Cyphal	-	-	-
Open source software	UAVCAN Consortium	Open cyphal	https://opencyphal.org/			
Open source software	Dronecode	Px4 autopilot	https://px4.io/			

APPENDIX B
TESTING REPORT

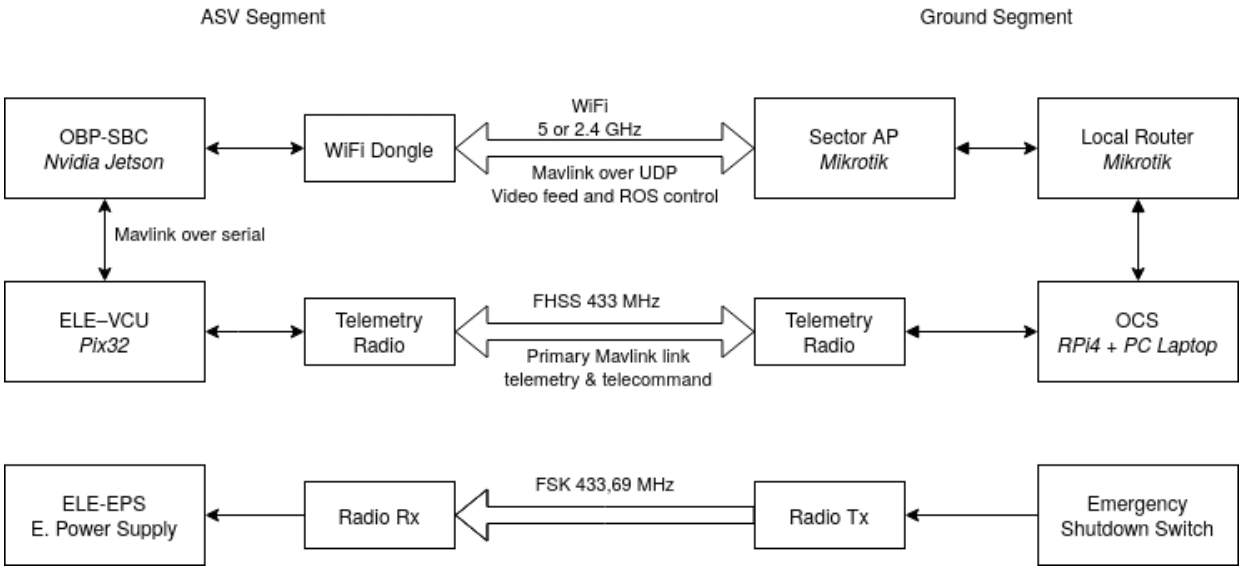
intentionally left blank

APPENDIX C
SYSTEM DETAILS

A. Product Tree



B. Telemetry System Overview



C. Electrical System Overview