IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Vehicle-in-the-loop framework for testing long-term autonomy in a heterogeneous marine robot swarm

Anja Babić*[1], Goran Vasiljević[2] and Nikola Mišković[1]

***Abstract*— A heterogeneous swarm of marine robots was developed with the goal of autonomous long-term monitoring of environmental phenomena in the highly relevant ecosystem of Venice, Italy. As logistics are a continuing challenge in the field of marine robotics, especially when dealing with a large number of agents to be collected and redeployed per experimental run, an approach is needed that provides the benefits of simulation while also reflecting the complexity of the real world. This paper focuses on the development of a vehicle-in-the-loop test environment in which a surface station simulates and transmits the data of any number of simulated agents, while a real marine platform operates based on the received information. Several experimental runs of a specific use-case test scenario using the developed framework and carried out in the field are described and their results are examined.**

I. Introduction

The Lagoon of Venice, Italy is a very particular ecosystem, highly relevant in a scientific, environmental, cultural, and socio-economic context and a critical subject of study. It is also an area undergoing significant shifts, severely impacted by both global climate change-related effects as well as site-specific phenomena [1] [2]. With such environmental factors as abundant spring rains and intense summer heatwaves, along with eutrophication and the effect of intense human activity on various types of sediment present in the lagoon, notable phenomena include hypoxic and anoxic events in the form of rapid localised drops in the concentration of oxygen in the water [3].

The Horizon 2020 Future and Emerging Technologies project subCULTron was conceptualised as a novel answer to the demand for long-term environmental monitoring in the waters of Venice. The goal of the project was superseding the need for boats and personnel periodically deploying and collecting probes at sea and at a variety of inaccessible locations by developing an autonomous heterogeneous swarm of marine robots. In effect, creating a topologically reconfigurable underwater acoustic sensor network with surface access points [4]. This is achieved using a three-layer structure envisioned as an artificial ecosystem: five Autonomous Surface Vehicles (ASV) called aPads (artificial lily pads) comprise the top layer, a small swarm of highly mobile exploration-focused aFish (artificial fish) represent the middle layer, and more than 100 underwater sensor nodes called aMussels (artificial mussels) act as the bottom layer.

Energy management in robotic swarms is generally addressed by implementing energy-efficient behaviours and providing some method of recharging to prolong swarm autonomy while not adversely impacting the mission in progress [5]. In [6] and [7] the authors aim to achieve arbitrary operating times and continuous energy (re)supply by modifying the operating environment of the mobile ground robot swarms. More applicable to the subCULTron swarm, several approaches have been proposed which focus on energy exchange between members of the swarm, with specialised agents serving as the starting points of energy supply chains [8] or navigating within the swarm while offering use of their multiple charging stations, monitoring other agents' location and battery levels, and responding to charging requests [9]. Among the several possible interactions between robotic agents in the subCULTron swarm, as shown in Fig. 1, hardware and software has been developed to enable agents to conserve and share energy [10]. Actively prolonging the operational time of the swarm is done by having an over-actuated autonomous surface platform dock up to four floating sensor nodes at a time and replenish their batteries using wireless inductive charging.

Working in the marine environment poses unique challenges. In [11], the authors aim to achieve an energy-efficient underwater swarm by having swarm members adjust their behaviour depending on their energy levels and needs. Similarly to the subCULTron swarm, acoustic communication is used for relaying information from agents on the surface to those that are submerged, as well as for underwater localisation. The projects described in [12] also aim to address the challenges of using large-scale swarms of aquatic drones in

Fig. 1. subCULTron heterogeneous marine robot swarm concept illustration highlighting interactions between three agent types.

*This work is supported by the European Union, by funding the EU H2020 FET-Proactive project 'subCULTron', no. 640967 and by the Croatian Science Foundation by funding the project IP-2016-06-2082 CroMarX.

[1]Faculty of Electrical Engineering and Computing, LABUST - Laboratory for Underwater Systems and Technologies, University of Zagreb, Unska 3, Zagreb, Croatia. anja.babic@fer.hr, nikola.miskovic@fer.hr

[2]Faculty of Electrical Engineering and Computing, LARICS - Laboratory for Robotics and Intelligent Control Systems, University of Zagreb, Unska 3, Zagreb, Croatia. goran.vasiljevic@fer.hr

Copyright ©2020 IEEE

real-world scenarios, primarily focusing on communication aspects and, very relevantly, ad-hoc networking between heterogeneous nodes - ones with long-range communication access, and ones with only local communication capabilities.

In order to accomplish energy sharing tasks as efficiently as possible, the aPads have access to a pool of heuristics for approaching decision-making problems. Evaluating the performance of different approaches and tuning the many parameters present in the system to improve its longevity is a challenging task and requires being able to repeat experiments several times. Logistically, experimental validation of the environmental monitoring potential of the robot swarm and its energy sharing capabilities poses a considerable challenge due to both the number of robotic agents involved and the goal of investigating a truly long-term approach. Hence, a desire to create an experimental framework in which simulated agents interact with real ones, in the spirit of the hardware-in-the-loop (HIL) approach (or in this case vehicle or vessel-in-the-loop (VIL) [13] [14]), which is the focus of this paper. While there have been some attempts at standardisation, existing simulators focusing on underwater environments and aspects relevant to marine vehicle development are not widely available and have some VIL capabilities [15] [16], with the more common approach being partial HIL, such as featuring the main electronics boards of the vehicle and one or more real sensors, with the rest simulated [17]. The simulator that is part of the system proposed in this paper does not feature a dedicated visualisation component, instead being compatible with existing command and control frameworks. It also aims to simulate underwater agents in their entirety, with seamless switching between simulated and real agents. Thus, the main contribution of this paper is the development and experimental validation of a vehicle-in-the-loop system for long-term autonomy of a marine robot swarm. The concept of the real autonomous marine vessel aPad picking up, charging, redeploying and releasing virtual agents aMussels with the aim of maintaining their long-term autonomy is presented. The experimental validation of the proposed concept was conducted in the realistic environment of the Adriatic sea.

Section II contains a brief description of the robotic agents involved in the vehicle-in-the-loop simulation, while Section III outlines the software implementation of both simulated and real agents. Section IV describes the design and realisation of the simulated agent, focusing primarily on modelling battery levels. Section V describes the proof-of-concept experimental scenario implemented using the developed system and gives some commentary on the results of several experimental runs in the field. Finally, Section VI gives some conclusions and plans for future work.

II. Robotic agent description

A detailed description of the functionalities and agent interactions featuring the aMussel and aPad robots (shown in Fig. 2), as well as the subCULTron system as a whole, is given in [18]. A short overview is given here to provide necessary context.

Fig. 2. The aPad platform and four aMussel underwater sensor nodes.

The role of the aPad agents in the swarm is that of an energy and information sharing hub, as well as a bridge between the swarm and a human observer. They also serve as anchors in the process of acoustic underwater localisation.

The platform has four thrusters in an X-shaped configuration which make it overactuated and omnidirectional. It can return to a home position and be charged while deployed thanks to a waterproof charging jack on its hull. For surface communication purposes, the aPads are equipped with a mesh-capable wireless router which provides both WiFi access points for aMussels and surface stations and a mesh network for the aPads themselves. For underwater communication purposes, the aPads use miniaturised acoustic modems called nanomodems, while for localisation and positioning, each platform has an IMU and a GPS module. The aPad on-board computer is an Intel NUC mini PC. Each aPad is also equipped with a Kinect sensor used for visual aMussel detection and autonomous docking.

The aMussel-type agents act as underwater sensor nodes, working together to form a sensor network for the purposes of distributed environmental monitoring and long-term data collection. They are severely limited in movement capability as they have no means of propulsion, only a buoyancy system which allows them to sink to the seabed, float to the surface, or hover at a certain depth. They house numerous sensor modules (the oxygen concentration, pressure, temperature, and turbidity sensors being of particular relevance to studying the anoxia phenomenon), with room for expansion. The aMussels use acoustic communication with dedicated timeslots for each agent when underwater, and have Bluetooth, GSM, and WiFi capabilities when on the surface. Aside from general communication, by using WiFi the aMussels can efficiently backup their data logs onto aPad computers while charging.

In order to make long-term autonomy possible, the aMussel was developed with low energy consumption in mind. Its main electronic board - called the MU (Measurement Unit) board - contains a Cypress PSoC4 microcontroller running FreeRTOS and Embedded C software and is capable of deep hibernation, minimising energy consumption. The rest of its modules were developed and connected to the central board in a way that makes it possible to disable the power supply of each individual module and sensor as needed. The modules can be woken up by the main board, or by an acoustic signal received by the acoustic modem on their top cap.

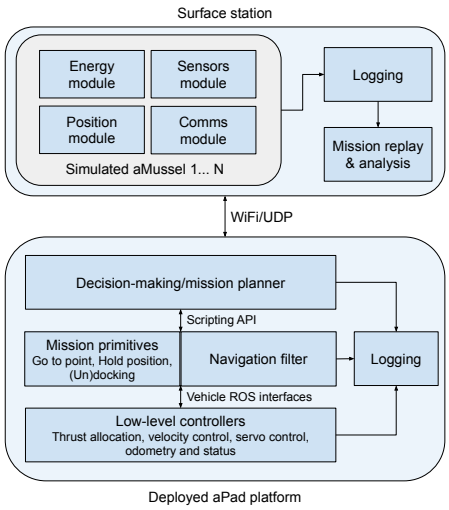


Fig. 3. System software and communication structure - simulated aMussel agents and real aPad platforms.

These configurable sleep cycles enable a considerable degree of adjustment - primarily reduction - of the robot's power consumption. To offset the lack of computational power of the MU, each aMussel was also equipped with a Raspberry Pi unit with a custom-made adapter board, powered on in short intervals when extra processing power is needed.

The aMussels are equipped with three inductive charging coils which enable battery recharging during mission execution. Each aPad has four mechanical docking stations housing inductive transmitter coils and guaranteeing good coil alignment, which allows it to charge up to four aMussels at a time, and a visual servoing system which makes it capable of autonomously picking up aMussels as they float and redeploy them where needed [10]. Energy exchange and deployment/relocation are among the key cooperative behaviours of the aMussel and aPad agents. The aMussel itself houses two lithium polymer batteries and two independent battery chargers. Both batteries are charged simultaneously when in the charging dock, but in standard operation the primary battery is in active use, while the secondary battery remains in standby and serves as backup.

III. Software architecture

The aPad vehicle and aMussel simulator software architecture is based on the Robot Operating System (ROS) [19] middleware and its structure of nodes, services, and messages, realised using Python and C++. The overall structure can be seen in Fig. 3.

The simulation run on the surface station includes a ROS node containing a class representing a selected predefined number of aMussel agents with unique agent ID numbers. In its several modules it implements communication protocols, tracks agent positions and battery status, and provides simulated or dummy sensor data, depending on experimental scenario demands. The modules are all connected to the singular comms module and use the same simulation clock, but are otherwise unaffected by each other.

The comms module subscribes to and parses messages received from aPads, and publishes aMusel sensor, position, and battery data at a fixed rate, using the same data structures and serialisation defined and used for real agents. For the purposes of experiments described in this paper, the simulated aMussels communicate with the aPads via WiFi (as if they are on the surface), though an interface for acoustic communication queuing exists, with a separate physical nanomodem unit attached to the surface computer via serial port. It is possible to intermix both real and simulated aMussels in one experiment, as long as care is taken to ensure no agent ID overlap.

The position module either generates a random uniform distribution of aMussels within a preset polygon representing the experimental area, or loads starting aMussel configurations from a file saved during a previous run, ensuring repeatability. A water current vector field map can be overlaid on the experimental area, causing agents to drift over time by translating their position each time step depending on the current vector. Depending on aPad status messages, the position simulation can switch between the free-floating mode and a charging mode in which aPad position is forwarded to the aMussel position topic to represent the aMussel in question being docked.

The models used in the simulated battery module are described in detail in Subsection IV-A. Starting battery states can be randomly generated or loaded from a configuration file. The simulator charges and discharges all simulated aMussel batteries every fixed time step depending on charging status data received from the aPad via the comms module (i.e. responding to an alert once a certain aMussel has been "docked" by the aPad reaching its position and closing the designated docking mechanism, or "released" and thus no longer being charged).

Mission primitives implemented and used in the long-term monitoring mission include Go To Point for moving the aPad to a certain location, Hold Position for dynamic positioning, and Docking/Undocking for collecting and redeploying aMussels. They are realised in the form of services that can be called manually or from within a mission script, accepting a set of parameters such as vehicle speed, victory radius, or docking mechanism selection. Controller parameters for all low-level controllers (e.g. yaw rate and surge speed for the docking procedure) can also be adjusted if needed.

As part of the overall system development, an Application Programming Interface (API) was developed to simplify high-level mission design and implementation. The aPad API is realised in the Python programming language and enables the creation (and automation thereof) of both extremely simple and linear task sequences as well as complex missions and behaviours, while removing the need for compilation before runtime. The API also contains definitions for all aPad- and aMussel-specific data structures. An example of a simple sequential aPad mission consisting of going to the last received position of the aMussel with the designated agent ID 24, then, once in position, initiating docking the mussel to the second docking mechanism, is given in Listing 1.

Listing 1

EXAMPLE OF SCRIPTING A SIMPLE APAD MISSION.

```
lat = api.get_location(24).lat
lon = api.get_location(24).lon
api.go_to([lat, lon])
while (api.goto_status(id_self) == 1):
        print(''GoTo in progress.")
        time.sleep(1)
print(''Goal reached. Docking start.)
api.docking(True, 2, 24)
```

The aPad mission planner is realised using this API. It contains a selection of strategies for collection and deployment which use a universal solution encoding format in order to make it easy to swap between approaches and add new ones if desired. The mission planner also contains aPad distance and energy cost calculation modules with adjustable scale factors, making it possible to produce a variety of behaviours and observe their effects on aPad task sequencing.

Data logging is realised in a redundant way: it takes place both locally on each aPad and on the surface station. The mission replay and analysis interface is realised in MATLAB based on the gathered *rosbag* format data logs (an example of the mission replay screen is given in Section V). The Neptus C4I Framework [20] can be used for aPad mission supervision, replay, and control, with aMussel data being relayed by connected aPads and displayed on the map overlay.

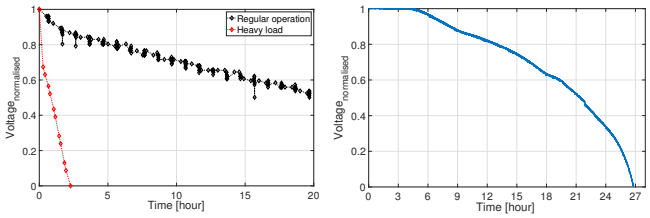
IV. SIMULATION DESIGN

A. Battery modelling and simulation

During the fairly extensive subCULTron field tests, it was empirically determined that an aMussel could reliably turn on and use all of its modules at least once in battery voltage ranges from a fully charged 4200 mV to a minimum of 3600 mV. For safety reasons, an aMussel should never be allowed to drop below this lower voltage limit, as this could mean it loses the ability to activate its buoyancy system motors and surface for recovery.

Fig. 4 (a) shows a realistic example of 20 hours of an aMussel operating overnight recorded in situ during a field experiment in Venice. The aMussel in this scenario sleeps and wakes up in regular intervals for several minutes in order to gather measurements, do some minor data processing, and await its designated acoustic timeslot, upon which it transmits an acoustic data packet - note the clearly visible voltage drops indicating brief hourly periods of increased activity interrupting low-power sleep mode. As contrast, the figure also shows an example of operation with unrealistically high power consumption highlighting two of the aMussel's biggest power sinks, where an aMussel's Pi board was kept on, and its buoyancy motors ran every 10 minutes, leading to full discharge within 3 hours. For reference and comparison, battery data of an aPad running its on-board computer and Kinect camera, activating all of its thrusters for 3 seconds every 30 seconds is shown in Fig. 4 (b).

In order to not only gather data for simulation design, but also test the long-term operating potential of the final



(a) (b)

Fig. 4. aMussel battery discharge data in two different modes of operation (a). aPad battery discharge scaled to operating range $[16, 10.5]V$ (b).

TABLE I

TIMES TO FULL AMUSSEL BATTERY DISCHARGE AND CHARGE.

Discharging, MU turned on but idle	962 min
Discharging, MU in sleep, waking up once an hour	14057 min
Charging, primary battery, MU turned on but idle	575 min
Charging, primary battery, waking up once an hour	398 min
Charging, secondary battery	485 min

developed aMussel system in a more structured manner, aMussel robots were left to discharge and then charged using aPad charging mechanisms at room temperature on dry land while engaging in a variety of operating behaviours. Their battery voltage was measured, after which the data segment representing the previously determined permitted operational area of $[3600, 4200]$ was scaled to an interval of $[0, 1]$.

The cases of particular interest for examining the discharging behaviour included a single aMussel battery discharging with the MU board turned on but idle (approximately 16 hours needed for full discharge) and the aMussel battery discharging with MU board in sleep, but waking up once every hour for about 10 seconds in order to record measurements from all sensors (9.8 days needed for full discharge).

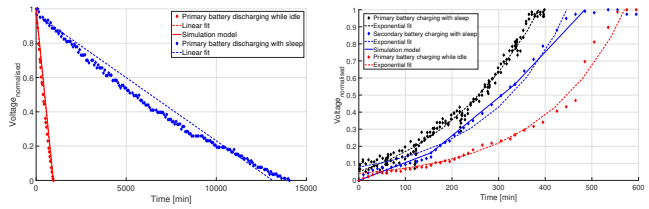
While discharging behaviours will be the same for both primary and secondary batteries under the same load, the charging will not due to the differing number of connected charging coils (two out of three charge the primary battery), so an additional charging use-case scenario was recorded. For the charging behaviours, of interest was the aMussel primary battery charging with the MU board turned on but idle (approximately 9.5 hours needed for full charge), the aMussel primary battery charging with sleep, waking up briefly in regular intervals to record sensor data (6.5 hours), and the aMussel secondary battery charging with the same sleep behaviour, with the MU powered by the primary battery (8 hours). For ease of comparison, recorded charging and discharging times are shown in Table I.

The gathered data confirmed the long-term potential and viability of several aMussel use-cases. A sleeping aMussel waking up once an hour for a few seconds has enough time to retrieve measurements from all sensors, and the data it is monitoring consists of very slow-changing variables so sparse sampling is appropriate. In case of an emergency an aMussel that is asleep can be woken up by an acoustic ping and receive whatever instruction is necessary. Furthermore, while an aMussel is charging on the surface it is not in active

use and can be asleep, speeding up the charging process. While taking into account the batteries' ageing and ability to hold charge [21], switching to and actively using the secondary battery should effectively double an aMussel's total deployment time.

For the experiments described in this paper, the decision was made to implement the "worst case scenario" of the aMussel only actively using power from one battery and keeping the other as an emergency backup, while charging assumes the need to charge the slower-charging secondary battery. Concerning the discharge model used, it is assumed there will be no sleep periods allowed during operation, thus guaranteeing an active aMussel is truly performing work in every time step.

Least squares fitting was applied to the recorded data in order to get a simple discrete simulation model-suitable representation of the chosen behaviours. A linear fit function was used for the discharging and an exponential represented using a piecewise-linear function for the charging. The equations were then implemented in the battery module of the aMussel simulator ROS node working at a rate of 1 Hz. Note that only voltage information was used in the simulation, as run-time indication was considered more important than any accurate state-of-charge modelling [22], and of primary concern was being able to make relative simulated aMussel battery state and charge/discharge time comparisons during high-level decision making. A time scaling factor was also introduced into the simulation, allowing for simulations where one unit of real time represents minutes or even hours in simulated time. Representative discharge and charge cycle datasets, as well as the simulation models of two specific behaviours that were chosen for the proof-of-concept experimental scenario, are shown in Fig. 5.



(a) (b)

Fig. 5. aMussel battery discharge data and implemented simulation model (a). aMussel battery charge data and implemented simulation model (b).

The final selected simulation models used in the proof-of-concept scenario are, for discharging:

$$u_k = u_{k-1} - 0.00104271340 \cdot \Delta t \tag{1}$$

and for charging:

$$u_k = \begin{cases} u_{k-1} + 0.001045\theta \cdot \Delta t & u_{k-1} \leq 0.16197 \\ u_{k-1} + 0.00223\theta \cdot \Delta t & 0.16197 < u_{k-1} \leq 0.5634 \\ u_{k-1} + 0.002911\theta \cdot \Delta t & 0.5634 < u_{k-1} \end{cases} \tag{2}$$

Where u is voltage, k represents the simulation step and θ is the simulated time scaling factor. In this case with a 1 Hz sampling time, in real time the step is $\Delta t = 1s = \frac{1}{60} min$. Final simulated voltage is clamped to $[0, 1]$.

B. Performance indices

Due to the nature of the end goal of the swarm being a long-term monitoring mission, aMussel uptime, defined as the percentage of total mission time during which the aMussel was actively performing useful work and capable of collecting relevant data, was chosen as the main performance criterion and benchmark. Another indicator of long-term monitoring performance is the number of active aMussels over time, as loss of active aMussels in the experimental area represents severe loss of monitoring capability and thus of valuable data. The other goal to be achieved is minimising aPad energy used for powering its thrusters for movement.

Time spent charging is not part of the total aMussel uptime, as due to the position of their sensors aMussels need to be fully submerged in order to collect measurements. Of course, an aMussel is also not considered active and contributing to its uptime while its battery is depleted below the lower operational threshold. Each aMussel's activity state is logged every time step for the entire duration of the experiment as a simple binary value - it is either active or inactive. Uptime for each individual aMussel is then calculated as:

$$T_{up,i}[\%] = 100 \cdot \frac{\sum_{k=1}^{N_{step}} a_{i,k} \Delta t}{t_{max}} = 100 \cdot \frac{\sum_{k=1}^{N_{step}} a_{i,k}}{N_{step}} \tag{3}$$

where $a_{i,k} \in \{0, 1\}$ represents activity of aMussel i during time step k, Δt is the time step, t_{max} is total mission duration, and N_{step} is the total number of time steps recorded. Also calculated are the average, minimum, and maximum of uptime for all aMussels present in the experiment. The number of currently active aMussels during each time step is also recorded and considered, as in a case without any charging taking place and for a sufficiently short mission time, aMussels could show "good" (if perhaps front-loaded) uptime as there is no loss of activity due to periods of charging, but this would ultimately leave the area completely unsupervised.

The aPad movement cost used by the decision-making algorithms is a simple calculation of Euclidean distance between the current aPad position and the GoTo mission primitive goal setpoint, meaning the spatial distribution of aMussels (i.e. the scatter of their collective cluster) greatly affects movement costs during a mission. While this useful abstraction can be expected to roughly correspond to aPad battery depletion and a smaller distance to travel will imply less energy expenditure, real world effects and disturbances such as wind and current speeds affect the vehicle. VIL tests enable the study of these effects, among others. Charging-based cost is a representation of charging demand from the aMussel side, i.e. an abstracted battery percentage difference from a fully charged state.

V. VEHICLE-IN-THE-LOOP EXPERIMENT

A very important aspect of this work is the experimental validation of the proposed methodology. This section demonstrates how the VIL system is used in a realistic environment to manipulate virtual agents in order to achieve their long-term autonomy.

A. Proof-of-concept experimental scenario

In [23], the general outline and early simulated implementations of the aPad decision-making approach are described, in particular the use of k-means clustering to divide the aMussels among several available aPads, essentially switching between a Vehicle Routing Problem (VRP) -analogous to a Travelling Salesman Problem (TSP) -analogous problem. One aPad platform at a time is used in the experiment presented here, under the assumption that the aPads have reached a consensus on which aMussels belong under whose jurisdiction.

As mentioned earlier, an aMussel draws the most power when it is using its buoyancy system motors, leading to the assumption that surfacing and going down needs to be kept to a minimum during operation. Time spent on the surface is undesirable, as it not only means the agent is inactive, but also means a risk of aMussels drifting away from desired positions. Thus, once an aMussel is docked, it will remain charging until it is fully charged - no partial charging sessions are allowed. aMussels that are charging candidates enter the charging selection pool, the upper threshold for which was set to a battery level of 80%. After charging, the aMussels must be returned to their erstwhile positions due to the assumption that they were purposefully placed on points selected in relation to known anoxia field spreads or other anomalous points of interest and study. Relocation procedures are a separate and important part of exploration paradigms also investigated within the subCULTron project.

Either 24 (the actual ratio of aMussel to aPad-type agents in the final subCULTron swarm) or 12 (in the pool experiments due to covering a smaller area) aMussel positions were randomly generated within the two areas chosen for initial testing: an outdoor pool and a bay next to it that opens up into the Adriatic sea. These generated aMussel positions are shown in Fig. 6. No current vector was applied, but a random drift of $\mathcal{U}(-0.5, 0.5)$ centimetres every second was introduced to simulated aMussel positions to account for them moving and drifting while sinking to or rising from the seabed, or localisation noise. Starting aMussel battery levels were generated from an interval of $[0.6, 1]$. All experiments described were run using the same aMussel configurations for their respective total agent number.

A timescale factor of $\theta = 30$ was used in the experiments, meaning a mission length of 40 minutes represents approximately 20 simulated hours. The node representing and simulating all active aMussels was run on a laptop on the shore connected to the aPad WiFi access point. The energy sharing scenario loop is shown in Fig. 7 and begins after the aPads have been deployed, reached the vicinity of the experimental area, and achieved a clustering

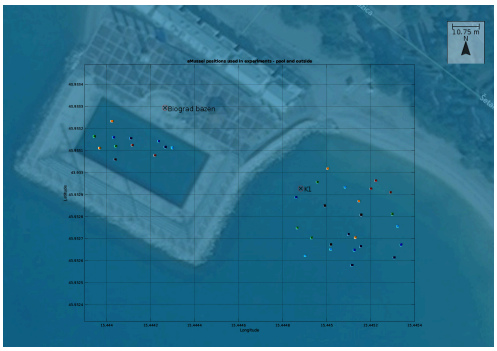


Fig. 6. The two experimental areas: initial generated aMussel positions used in all experiments, overlaid on map with satellite image.

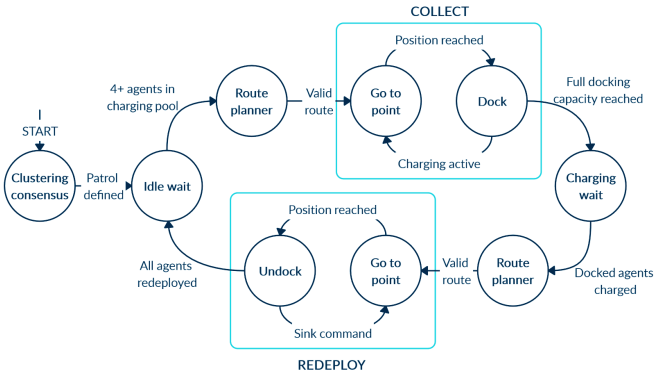


Fig. 7. Energy exchange scenario loop.

consensus, meaning all active aPads have their designated patrol areas defined by IDs of aMussels they are responsible for maintaining. The loop itself consists of the following:

- **Idle wait** - The aPad waits, holding its current position using the dynamic positioning mission primitive, until at least four aMussels are present in the charging pool.
- **Route planner (collection)** - The aPad plans a route to collect four aMussels chosen from the candidates in the charging pool using one of several heuristics.
- **Collect** - Four instances of the Go To aMussel position - Dock routine are carried out, until four aMussels have been successfully docked and the aPad is at full docking/charging capacity. Each time the aPad moves to the position of the next aMussel on its collection schedule, it sends it an acoustic message instructing it to surface for collection. This is done so the aMussels spend a minimal amount of time floating on the water surface and also reduces inaccuracies in aMussel position information (as their last localisation is assumed to have happened underwater). For safety reasons, real aMussels will never surface without an aPad present in acoustic range giving permission.
- **Charging wait** - The aPad holds position at the location of the last aMussel it picked up, until all four docked and inactive agents report battery levels above 99%.
- **Route planner (deployment)** - The aPad plans a redeployment route using the Greedy redeploy heuristic,

bearing in mind all aMussels must be returned to their original positions.

- **Redeploy** - The aPad alternates between going to the position of the next aMussel to deploy, and undocking it. Once an aMussel is released, it detects a falling edge on its charging state, and reacts to this by using its buoyancy motors to sink to the bottom of the water and resume collecting measurements - for a simulated agent, this means transitioning to a discharging battery behaviour and becoming active. The aPad also sends a WiFi data packet with a specific payload to let the floating aMussel know it can sink without interference.

The scenario can be interrupted by an override from a human operator, or it can be set to end once a certain predefined mission time has been achieved. Another termination condition available is aPad batteries reaching a certain threshold.

B. Collection and redeployment strategies

The heuristics contrasted in the scenario include collection using the so-called Greedy and Cautious methods, and redeployment using the Greedy Redeploy method. Greedy means the aPad moves to collect the closest available aMussel in each planning step. This heuristic aims to reduce aPad movement in a very simple way - there is no guarantee of global movement minimum being achieved, but the planning time required for it is very low. Cautious planning does not consider aPad movement at all, instead focusing on collecting the aMussel with the currently lowest battery state in each step, i.e. the aMussel that could be said to be most in need of charging, aiming to prevent complete depletion and potential loss of any agent. During Greedy Redeploy, once all aMussels have been charged, the aPad moves to redeploy them in order of current proximity to their initial positions.

C. Proof-of-concept experiment results

The VIL framework was tested in several experiments in Biograd na Moru, Croatia. Two examples of experiments in progress are shown in Fig. 8.

Table II shows a comparison of two experiments done in the pool with 12 mussels and one done outside with 24, as well as a "baseline" case with no aMussels being charged during the experiment. Each experiment lasted 40 minutes and included four full sets of collections and redeployments. The values being compared are average aMussel uptime, maximum and minimum uptime achieved by any of the

(a) (b)

Fig. 8. aPad autonomously carrying out decision-making experiment in pool, (a). Experiment outside pool, in nearby bay (b).

TABLE II

AMUSSEL UPTIME, ACTIVITY, AND TOTAL APAD MOVEMENT COST

	Avg %	Max %	Min %	Cost [m]	Active
Cautious	74.37	87.38	59.88	307.1746	8.9243
Greedy	72.27	87.16	42.84	247.8026	8.9495
Greedy (bay)	71.98	90.31	31.18	186.4318	16.2368
No charging	67.36	79.56	56.64	0	8.0837

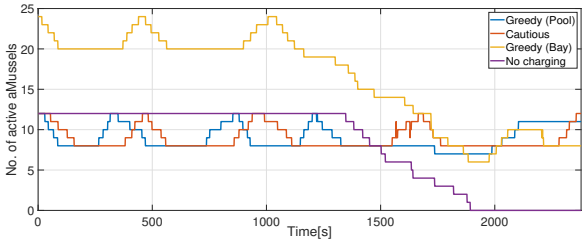


Fig. 9. Number of currently active aMussels during the experiments.

aMussels, cumulative movement costs of the aPad, and the average number of aMussels active during each time step.

The mission replay screen displays the preset aMussel locations, as well as aPad location and a trace of recent trajectory. aMussel markers change colour to indicate status: green for charged above charging pool inclusion threshold, red for charging pool candidates, and purple for currently undergoing charging. Cumulative aPad movement cost is displayed, as well as the currently active mission primitive. Battery states of all aMussels are plotted, as is uptime for each agent, updated every simulation step based on known total mission length and current aMussel activity. The number of active aMussels over time and aPad battery information are plotted and displayed separately. A video showing two examples of mission replays (one for each collection heuristic) is available as supplementary material at `http://ieeexplore.ieee.org`.

The Greedy method shows less aPad movement overall but more time "wasted" waiting for the lowest battery aMussels to recharge, with some fully charged aMussels needlessly inactive - Cautious has better balancing in that regard, as it makes decisions based on aMussel battery states. Various alternate redeployment solutions are possible to address this, though potentially at the cost of increased aPad movement.

Due to the fact the aMussels have simulated accelerated time while the aPad doesn't, the aPad seems to move very slowly, relatively (the fairly frequent 30-second bursts of movement corresponding to 15 minutes in simulated time). This can be interpreted as the aPad travelling over longer distances, however it will of course not be reflected in the actual aPad battery consumption and state. While the mission ended before this could fully come into effect, it can be seen that the aPad is becoming "overwhelmed" by the larger number of aMussels in the example taking place in the bay, leading to the number of active aMussels declining, and highlighting questions of aMussel-per-aPad maintenance capacity. The number of charging cycles a single aPad can feasibly provide is another useful variable to study, for which

the developed framework is very helpful, especially in a sense of "rapid prototyping" of behaviours.

VI. Conclusion

The developed VIL setup can be used to test a variety of specific use-cases of the subCULTron heterogeneous marine robot swarm in a logistically feasible way. The development process resulted in a twofold benefit: collected preliminary data confirmed the long-term operating potential of the aMussel agents themselves, while also providing several different stock behaviour variants for simulation models which will be useful for all future studies of the swarm's agent interactions.

The undertaken proof-of-concept experiments showed variations in aMussels covering two different experimental areas by actively contributing to data collection efforts and required environmental monitoring, while being supported by an aPad platform autonomously engaging in energy sharing using two different heuristic approaches to its task sequencing. The early experiments already provided valuable insights, such as the need for more complex selection and acceptance criteria during the decision-making process to ensure the correct behaviours are rewarded, leading to long-term sustainable and beneficial swarm behaviours.

The simulated aMussel structure can be run on the aPads themselves and used during missions with real sensor nodes, serving as a model for the aPads to estimate aMussel status and fill in any potential information gaps without needing to receive communication packets from the real agents who might be asleep, out of range, or subject to some other cause of interference and data loss. This way, aPads can pre-emptively make decisions and plan their missions, adjusting to incoming data in real time as they work.

The next step in the development of the system includes VIL experiments that go on for significantly longer in both real and simulated time. The effects of even small changes in parameters on the longevity of the swarm as a whole as well as the influence of prioritising different selection criteria will be studied extensively within the newly developed framework. The effects of sleep intervals spreading total aMussel uptime over a longer period and aMussel activity-over-time optimisations will also be studied.

References

[1] D. Tagliapietra, N. Aloui Bejaoui, D. Bellafiore, R. De Wit, C. Ferrarin, S. Gamito, P. Lasserre, P. Magni, M. Mistri, A. Pérez Ruzafa *et al.*, "The ecological implications of climate change on the Lagoon of Venice," *UNESCO Digital Library*, 2011.

[2] A. Ferrighi, *Flooding and environmental challenges for Venice and its lagoon: state of knowledge*. Cambridge University Press, 2005.

[3] E. Argese, G. Cogoni, L. Zaggia, R. Zonta, and R. Pini, "Study on redox state and grain size of sediments in a mud flat of the Venice Lagoon," *Environmental Geology and Water Sciences*, vol. 20, no. 1, pp. 35–42, jul 1992.

[4] G. Tuna and V. C. Gungor, "A survey on deployment techniques, localization algorithms, and research challenges for underwater acoustic sensor networks," *International Journal of Communication Systems*, vol. 30, no. 17, p. e3350, nov 2017.

[5] G. Li, I. Svogor, and G. Beltrame, "Self-Adaptive pattern formation with battery-powered robot swarms," in *2017 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2017*. IEEE, jul 2017, pp. 253–260.

[6] F. Arvin, S. Watson, A. E. Turgut, J. Espinosa, T. Krajník, and B. Lennox, "Perpetual Robot Swarm: Long-Term Autonomy of Mobile Robots Using On-the-fly Inductive Charging," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 92, no. 3-4, pp. 395–412, dec 2018.

[7] T. Deyle and M. Reynolds, "Surface based wireless power transmission and bidirectional communication for autonomous robot swarms," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2008, pp. 1036–1041.

[8] H. Schioler and T. D. Ngo, "Trophallaxis in robotic swarms - beyond energy autonomy," in *2008 10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008*. IEEE, dec 2008, pp. 1526–1533.

[9] F. Arvin, K. Samsudin, and A. R. Ramli, "Swarm robots long term autonomy using moveable charger," in *Proceedings - 2009 International Conference on Future Computer and Communication, ICFCC 2009*. IEEE, apr 2009, pp. 127–130.

[10] A. Babić, F. Mandić, G. Vasiljević, and N. Mišković, "Autonomous docking and energy sharing between two types of robotic agents," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 406–411, jan 2018.

[11] A. Amory, T. Tosik, and E. Maehle, "A load balancing behavior for underwater robot swarms to increase mission time and fault tolerance," in *Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS*. IEEE, may 2014, pp. 1306–1313.

[12] A. L. Christensen, S. Oliveira, O. Postolache, M. J. De Oliveira, S. Sargento, P. Santana, L. Nunes, F. Velez, P. Sebastião, V. Costa, M. Duarte, J. Gomes, T. Rodrigues, and F. Silva, "Design of communication and control for swarms of aquatic surface drones," in *ICAART 2015 - 7th International Conference on Agents and Artificial Intelligence, Proceedings*, vol. 2, 2015, pp. 548–555.

[13] T. Bokc, M. Maurer, and G. Farber, "Validation of the vehicle in the loop (VIL); a milestone for the simulation of driver assistance systems," in *2007 IEEE Intelligent vehicles symposium*. IEEE, 2007, pp. 612–617.

[14] M. Brinkmann, M. Abdelaal, and A. Hahn, "Vessel-in-the-loop architecture for testing highly automated maritime systems," in *Proceedings of the 17th Conference on Computer and IT Applications in the Maritime Industries*, 2018.

[15] T. Tosik and E. Maehle, "MARS: A simulation environment for marine robotics," in *2014 Oceans - St. John's, OCEANS 2014*. IEEE, sep 2015, pp. 1–7.

[16] P. Ridao, E. Batlle, D. Ribas, and M. Carreras, "NEPTUNE: A HIL simulator for multiple UUVs," in *Ocean '04 - MTS/IEEE Techno-Ocean '04: Bridges across the Oceans - Conference Proceedings*, vol. 1. IEEE, 2004, pp. 524–531.

[17] V. K. Kaliappan, A. Budiyono, D. Min, K. Muljowidodo, and S. A. Nugroho, "Hardware-In-the-Loop simulation platform for the design, testing and validation of autonomous control system for unmanned underwater vehicle," *Indian Journal of Marine Sciences*, vol. 41, no. 6, pp. 575–580, 2012.

[18] I. Lončar, A. Babić, B. Arbanas, G. Vasiljević, T. Petrović, S. Bogdan, and N. Mišković, "A Heterogeneous Robotic Swarm for Long-Term Monitoring of Marine Environments," *Applied Sciences*, vol. 9, no. 7, p. 1388, apr 2019.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[20] J. Pinto, P. S. Dias, R. Gonçalves, E. Marques, G. Gonçalves, J. B. Sousa, and F. L. Pereira, "NEPTUS A Framework to Support the Mission Life Cycle," in *7th IFAC Conference on Manoeuvring and Control of Marine Craft*, 2006.

[21] S. Barcellona, M. Brenna, F. Foiadelli, M. Longo, and L. Piegari, "Analysis of ageing effect on li-polymer batteries," *The Scientific World Journal*, vol. 2015, 2015.

[22] V. Pop, H. J. Bergveld, J. H. Op Het Veld, P. P. Regtien, D. Danilov, and P. H. Notten, "Modeling battery behavior for accurate state-of-charge indication," *Journal of the Electrochemical Society*, vol. 153, no. 11, p. A2013, nov 2006.

[23] A. Babić, N. Mišković, and Z. Vukić, "Heuristics pool for hyper-heuristic selection during task allocation in a heterogeneous swarm of marine robots," *IFAC-PapersOnLine*, vol. 51, no. 29, pp. 412–417, jan 2018.