European Journal of Technique

journal homepage: https://dergipark.org.tr/en/pub/ejt

Vol.14, No.1, 2024

Research Article

A Communication System for Dynamic Leader Selection in Distributed UAV Swarm Architecture

Abdulmelik Bekmez[1] Kadir Aram[2*],

[1]Fatih Sultan Mehmet Vakıf University, Computer Engineering Department, 34445, Istanbul, Turkey. . (e-mail: abdulmelik.bekmez@gmail.com).
[2 *]Fatih Sultan Mehmet Vakıf University, Computer Engineering Department, 34445, Istanbul, Turkey (e-mail: karam@fsm.edu.tr).

ARTICLE INFO

Received: Jan., 18. 2024
Revised: Feb., 13, 2024
Accepted: Feb, 15, 2024

Keywords:
Formation Control
Leader-Follower
Robotics
Distributed
Communication

Corresponding author: *Kadir Aram*

ISSN: *2536-5010* | e-ISSN: *2536-5134*

DOI: https://doi.org/10.36222/ejt.1434751

ABSTRACT

Distributed swarm robot systems are made up of several robots that communicate with one another and often work together to complete a task or reach a predetermined objective. These systems frequently consist of many platforms, like unmanned aerial aircraft, mobile robots, or other types of vehicles.

This paper offers a comprehensive exploration of the design, modeling, and real-world hardware and software implementation of a distributed swarm system. The decision was made to employ standard Pixhawk hardware for the swarm agents. Pixhawk, a freely available hardware and software platform for autonomous flight control, is commonly utilized in autonomous cars, multirotor vehicles, drones, and various robotic applications. Operating autonomously from the ground control station, swarm agents dynamically identify leaders during operation and execute leader tracking navigation to model swarm behavior. Ensuring generality and dynamism in all protocols and communication was a primary focus during the research phase. To maintain this dynamism, each protocol and communication process is implemented in distinct threads on the computer, and synchronization is achieved through synchronization primitives, shared memory, and interthread communication.

1. INTRODUCTION

Robot swarms are important for automation systems in many areas, such as search and rescue operations, environmental monitoring, environmental cleaning, area surveillance, agricultural activities, and transportation of heavy loads [1-5] . Unmanned aerial vehicle (UAV) swarms provide a collaborative structure to perform complex operations with a single independent UAV.
Swarm robot systems are generally controlled through two main methods: centralized and distributed. In centralized systems, control originates from a single central point. On the other hand, in decentralized systems, each agent operates autonomously, making independent decisions and taking actions without relying on a central control point.
Various methods have been employed in the design of distributed UAV swarm control. These include a hybrid-flocking control algorithm, amalgamating vector field guidance, augmented Cucker-Smale model, and potential field techniques to attain path following, collective flocking behavior, and collision avoidance [6]. Another approach involves a mixed game theory utilizing a hierarchical learning algorithm for large-scale multi-agent systems, employing cooperative game, Stackelberg game, and mean field game for efficient coupling between leaders and followers [7]. Additionally, a distributed method, relying on monocular vision information, integrates a control model, target detection through a modified YOLOV3-tiny method, and orientation and distance estimation using geometric approaches [8]. Zhu and Deng proposed a distributed swarm control framework with limited interaction. In this framework, UAVs select limited interactive neighbors, combining interaction force and obstacle avoidance to ensure safety and effective guidance [9].
Various strategies are employed in the formation control of robot swarms, including leader-follower [10], virtual structure [11], and behavioral-based approaches [12].
The leader-follower strategy involves designating one agent as the leader, with the remaining agents following its movements. This study adopts the leader-follower strategy due to its capacity to minimize the number of connections. Numerous studies have explored variations of this strategy, employing different methodologies.
Zhang et al. introduced three strategies for formation reconfiguration, focusing on leader disengagement, follower detachment, and adding new members to minimize the need for frequent connection changes [13]. Restrepo and Loria proposed

two controllers for formation-tracking control of velocity-controlled unicycles in a leader-follower configuration, addressing scenarios with known and unknown leader velocities [14]. Lee et al. conducted a study on swarm control algorithms for unmanned surface vehicles (USVs), validating the effectiveness of a leader-follower swarm control method through sea area tests [15]. Pauli and Fichter proposed a leader-follower formation control algorithm for UAV swarms, ensuring precise lateral and vertical separation during turns and adeptly handling communication limitations among agents [16].

To use a leader-follower strategy in a distributed system, the leader robot must also be dynamically selected by the swarm agents. One approach utilizes behavior-based control and a repulsive force method to navigate maze-like environments, automatically designating a leader when robots are stranded [17]. Another strategy concentrates on choosing a minimal set of leaders, employing the sub modularity ratio and defining metrics based on consensus tracking criteria [18].

The study encompasses several key aspects, including the dynamic determination of a single leader by swarm agents, the communication protocols and techniques employed in this process, the implementation of these protocols, and the subsequent testing of the architecture. Upon reviewing the literature, it became evident that existing research on distributed swarm systems is predominantly theoretical and algorithmic. This study aims to bridge this gap by adapting algorithms for the communication of a distributed swarm system and the navigation of agents to real hardware. The second section delves into the hardware structure, communication protocol, and autopilot software. Following this, the communication architecture, dynamic leader selection process, and leader tracking strategy are elucidated. The third section showcases the interface and simulation environment of the study. The final section engages in a comprehensive discussion of the results.

2. MATERIALS AND METHODS

This study involves multiple stages, particularly in implementing the leader-follower strategy. In this strategy, the robots are required to autonomously designate a leader and then organize themselves into a formation around this leader, following them to the specified location. The consecutive steps for these processes are detailed below.

2.1. Materials

The study is designed to be implemented on real hardware, and for this purpose, Pixhawk has been chosen as the hardware for autonomous flight control. Pixhawk, an open-source platform for autonomous flight control hardware and software, includes a control board that seamlessly integrates with diverse sensors and actuators, providing extensive capabilities for flight control and automation [19]. Widely employed in various robotic applications like multirotor vehicles, drones, and autonomous vehicles, Pixhawk serves as a standard control platform. Several autopilot software, including Ardupilot and PX4, have been developed based on this standard.

Ardupilot and PX4 are the most popular open-source autopilot systems. They are designed to operate a wide range of autonomous vehicles, including submarines, rotary-wing platforms, and fixed-wing aircraft [20].

PX4 is an embedded robotics middleware and programming environment with a multithreaded, publish-subscribe design pattern. It offers a software interface for microcontroller applications. The PX4 autopilot platform can operate independently on Pixhawk standard hardware or be coupled with a companion computer for tasks demanding additional processing power or an external GPU [21]. The PX4 autopilot software was used because it is an open-source platform, uses an open-source hardware standard, and is well integrated with widely used software such as ROS2.

ROS is an open-source software environment tailored for both commercial and research-based robotic applications [22]. It's preferred for its versatility, enabling development in various languages, fostering communication between processes through a publisher-subscriber architecture, supporting project-specific message types, allowing packaging and sharing of applications, and providing extensive package support for common robotic challenges. ROS features tools like RVIZ for data visualization [23]. In ROS, processes are termed nodes, and they communicate via messages, with publisher nodes sending messages to subscriber nodes through topics [24].

ROS2 was created to address the limitations of ROS, which include a centralized network structure, a lack of network security mechanisms, and dependence on third-party software for integration into microprocessors [25]. ROS2 adopts a distributed network system, employing the network layer's Data Distribution Service (DDS). Additionally, for communication between drones and between drones and ground stations, the Zenoh protocol was implemented.

Zenoh, a Pub/Sub-Query protocol, intricately unifies computations, data in motion, and data at rest. It was purposefully designed to cater to the demands of the shift from micro-controllers to the cloud, offering a seamless integration of diverse network topologies and technologies. This enables Zenoh to deliver messages with heightened speed and efficiency, fulfilling the need for high-density bandwidth while keeping latency to a minimum. As a result, it has evolved into an essential component of the Cloud-to-Edge Continuum [26].

Zenoh facilitates communication between two processing units on the same computer and extends its functionality to units on different computers within the same local network. This functionality resembles the DDS network software found in ROS2, sharing the dynamic discovery feature with DDS. Notably, Zenoh exhibits greater efficiency than DDS in wireless networks[27]. In the upcoming swarm system design, where swarm agents lack specific IP addresses and need to communicate via RF communication, the Zenoh protocol will be employed. This choice is driven by Zenoh's capability to operate in systems without IP addresses. Consequently, swarm agents will communicate through the serial interface provided by telemetry modules, utilizing the Zenoh protocol.

Gazebo is a 3D simulation platform that enables the development and testing of robots to be operated in indoor and outdoor environments [28]. It is frequently used in robotics research projects, competitions and commercial applications. PX4 autopilot software supports the simulation of rotary-wing platforms, fixed-wing, and VTOL robots with the Software In The Loop technique in development processes using the Gazebo environment. The Gazebo Garden version was used during the study.

The UAV used for this research is based on a Holybro X500 quadrotor equipped with a Pixhawk flight controller. The appearance of the robot in the simulation environment is given in the Figure 1.

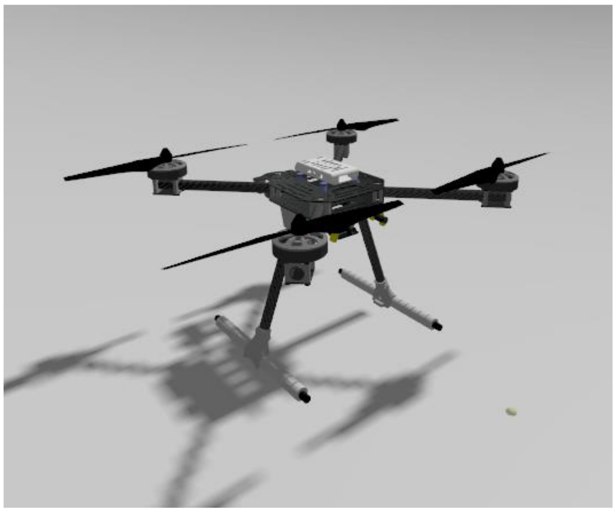

Figure 1. X500 in Gazebo Simulation

2.2. Communication Architecture

A crucial aspect of a distributed swarm architecture is the communication mechanism among swarm agents and other system elements. To establish a distributed framework, each agent must support peer-to-peer communication. Defining the communication protocols and message formats for these interactions is imperative. For agent-to-agent communication, the "swarm/<agent_name>" topical format is utilized when one agent sends a message to another. For instance, if agent1 intends to convey a message to agent2, it sends a P2P message to the "swarm/agent2" topical. This P2P message comprises the sender's name, the message type, and the P2PType data, encapsulating message-specific information. A P2P message includes 4 types of messages. These are:

- Heartbeat
- HeartbeatAck
- Pooling
- Selection

HeartBeat Message: A dynamically generated swarm leader sends the Heartbeat message to the swarm members. This message indicates that the swarm is connected and allows the swarm leader to assign tasks to the swarm members.

HeartbeatAck Message: The HeartbeatAck message indicates the message the swarm member sends to the leader. The swarm member sends the swarm leader position and orientation data about itself in this message. This message also determines the status of the swarm member's connection to the leader.

Pooling Message: Pooling message, a type employed for dynamic swarm leader selection, comprises four subtypes. These are:

- StartPool
- StartPoolAck
- SyncPool
- SyncPoolAck

StartPool Message One agent sends a StartPool message to another agent to initiate the pooling protocol to create a swarm.

StartPoolAck Message The "StartPoolAck" message serves as the response to an incoming "StartPool" message. Depending on the agent's present state, it dispatches a "Success" message if it is available and an "AlreadyInPool" message if it is already part of a pool. The "AlreadyInPool" message contains details about the current pool state.

SyncPool Message The "SyncPool" message is sent to agents during the pool creation process to synchronize the instantaneous pool with other potential swarm agents at a specific frequency. This message contains the pool data maintained by the agent.

SyncPoolAck Message SyncPoolAck message is sent by the agent receiving the SyncPool message to the sending agent as a reply. If the pools of the agents are the same, the 'Same' message is sent; if they are different, a 'Different' message is sent. The 'Different' message also holds the names of the elements in the pool. It also contains the status of whether the agent's pool is locked or not.

Selection Message: The Selection message dynamically specifies the type of message that each agent shares its selection with the other agents in the pool with a specific frequency in the selection part, which is the last part of the swarm leader selection process.

2.3. Dynamic Leader Selection

In order to have a distributed architecture, a swarm system should not be utterly dependent on any central authority. However, swarm systems with a fully distributed architecture are subject to communication constraints. A simple calculation of the number of connections is shown in equation 1.

$$c = \frac{(n-1)\times n}{2} \qquad (1)$$

Here: c is connection count; n is agent count.

For example, if a swarm system with ten agents wants to create a communication topology between each agent, it must create 45 peer-to-peer channels. For these reasons, the swarm system is designed to determine the swarm's leader and maintain this connection dynamically.

The process of dynamically selecting a leader consists of the following sub-parts.

- Selection pool initialization phase
- Expanding the selection pool
- Phase to lock the selection pool
- Election phase
- Leader selection

Selection pool initialization: While in this state, the agent broadcasts a self-introduction message via the “swarm/advertize“ topical. Through this topic, the agent signals its readiness for potential connections and swarming, accompanied by sending its name to facilitate establishing a connection. Simultaneously, all available agents monitor this topic, including the agent who transmitted this message. Another agent receiving this message sends a “StartPool” message to the sending agent. Then, it waits for the "StartPoolAck" message to arrive. The agent receiving the "StartPool" message, if it is not already in a pool, sends a Success message to start the pool and gives feedback to the other agent. However, if it is already in a pool and this pool is still accepting members, it sends an "AlreadyInPool" message and adds the agent that sent the message to the group. If the recipient of the "StartPool" message is not in any of these situations, it ignores the incoming message and does not send any response.

Expanding the selection pool: After the pool is initialized for dynamic leader election, the agents perform two types of operations.

- Send a pooling request to agents sending 'Advertize' messages

- Ensuring pool synchronization between agents in the pool

Agents that are expanding the pool also keep the last time it was updated. This time, data is updated if a member is added or removed from the pool, preventing the pool from being created and closed immediately. If a new member is added to the pool, the time data is updated, and as a result of synchronization messages, other agents also update their pools and the time data they keep.

Lock Selection Pool: Once the selection pool stabilizes for two seconds, it becomes locked for all agents. Agents that have successfully locked the pool continue transmitting SyncPool messages to others in the pool at a specific frequency. If there are agents whose pools are still unlocked, the pool locking status is indicated with the SyncPoolAck message. This ensures that if, for any reason, there are agents whose pools remain unlocked, the synchronization of pool locking among all agents is guaranteed. In the case where received pools differ, both agents merge their pools. Following the pool merging process, other agents update their pools through synchronous messages sent to one another. Similar to expanding the selection pool, the last update time is retained in the case of locking. This time, data is updated during an asynchronous state, and if there is no update for two seconds, the pool selection process is initiated.

Election Phase: The voting process can be tailored to the specific swarm architecture or problem at hand. In the devised architecture, each agent autonomously casts a vote for itself as the potential swarm leader, with each vote assigned a weight and randomized during implementation. Agents within the pool exchange their votes at a designated frequency. Upon receiving a vote message, the recipient agent scrutinizes the incoming vote. If it is still locked within the pool, the agent promptly updates its status to "Selection" and initiates the voting process. If the incoming vote weight surpasses its own, the agent updates its vote with the incoming one and the corresponding time variable.

Conversely, if the incoming vote carries less weight, the agent maintains its current vote. This mechanism ensures unanimity and synchronization among the agents. If, after 2 seconds, the voting pool converges on a consistent vote, the leader selection process within the pool is triggered, similar to other scenarios.

Leader Selection: Once a unanimous vote is secured in the election pool, the leader initiates a heartbeat message transmission to its swarm at a specified frequency. This transmission serves the dual purpose of confirming the leader's continued activity and assigning tasks to the swarm members. Concurrently, the leader awaits the reception of "HeartBeatAck" messages, indicating the ongoing activity of swarm members and facilitating the exchange of necessary data, such as Pose messages. If a "HeartBeatAck" message isn't received from a swarm member within a defined timeframe, the leader expels that agent from the swarm and discontinues the transmission of heartbeat messages to them. Conversely, if a swarm member fails to receive a heartbeat message from the leader within a specified period, it refrains from sending a "HeartbeatAck" message. Even if a heartbeat message is received subsequently, the swarm member departs from the swarm, opting to join another or create a new one. The flow chart of the leader selection process is shown in Figure 2.

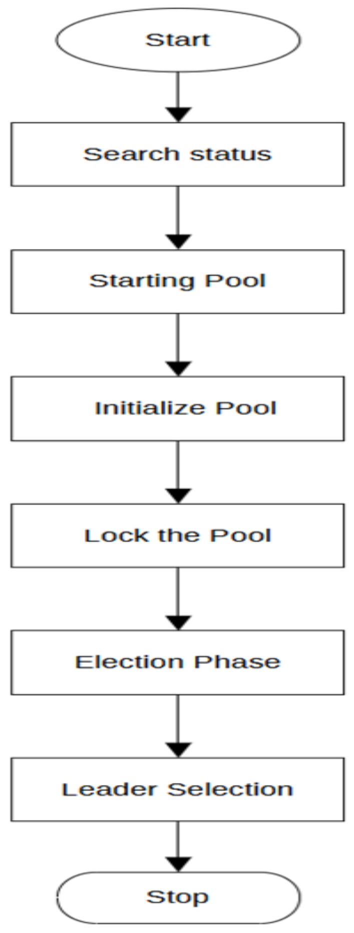

Figure 2. Leader Selection Flow Cart

2.4. *Formation Process*

The strategy involves the leader robot moving to a predefined target point while the other robots follow it. Continuous communication is essential between the leader and the followers. The leader robot continuously broadcasts its orientation and position, which is then transmitted to the follower robots. The followers, utilizing the leader's position, maintain a predetermined formation by following the leader's movements. The leader's current location serves as the goal for the follower robots, and they align their orientation with that of the leader. Throughout the tracking process, a constant distance between the leader and the followers is maintained.

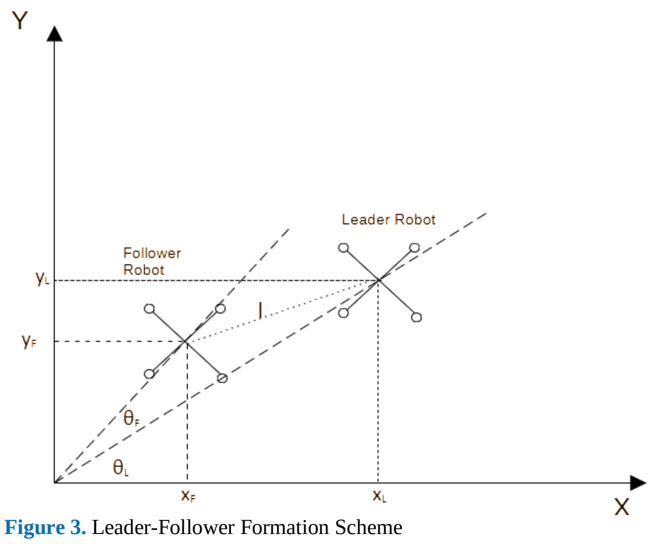

Figure 3. Leader-Follower Formation Scheme

In Figure:
X_L, Y_L : leader's position,
θ_L :is the leader's orientation
X_F, Y_F: follower robot's position
θ_F :follower robot's orientation.
l :the distance between the leader and the follower.
Equation 2 shows the distance between the leader robot and the follower robot.

$$dist\ (l) = \sqrt{(X_L - X_F)^2 + (Y_L - Y_F)^2} \qquad (2)$$

The PX4 autopilot software employs a position controller for guiding swarm agents from point A to point B. Using the Kalman Filter algorithm, a position estimation method, the position controller establishes a local coordinate plane, referencing the global position at the start time. Upon receiving a follow command from the leader through the interface, the leader communicates with each swarm agent, specifying the agent to be followed, the following direction, and the general heading angle for leader-follow navigation. Swarm members utilize peer-to-peer communication to subscribe to the Pose data of the designated agent. After receiving the agent's name, they follow the leader's command, creating a 10-meter vector in the opposite direction of the swarm's head rotation angle and a 5-meter direction vector based on the specified direction for the agent to stay. Position determination is accomplished by adding these vectors to the position of the agent being followed.

3. EXPERIMENTS

Swarm ground control station software has been developed for efficient swarm management, allowing dynamic connections and individual control of each agent. Agents introduce themselves to the ground control station via the swarm/GSAdvertize topical. The ground control station software consistently monitors this topic, sending connection requests to agents that identify themselves and initiate the connection. For the connection to be maintained, the agent must send a Heartbeat command with a specific frequency, indicating that the connection is active; otherwise, it disconnects from the ground control station software. To facilitate location tracking and control of swarm agents across the global map, a map interface was developed using OpenStreetMap data. This interface displays the locations of agents connected to the ground control station. The general map view of the ground control station is shown in Figure 4.

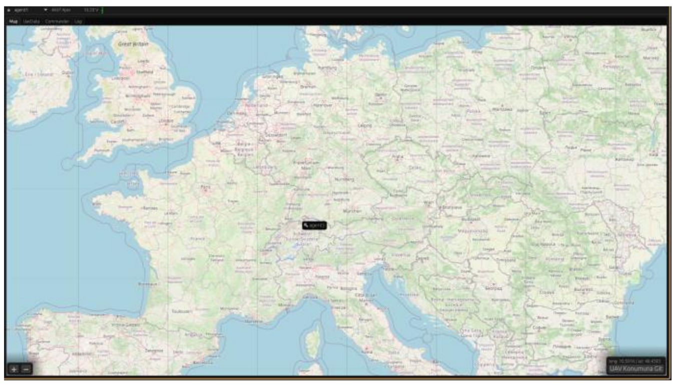

Figure 4. General Map View

A close-up map view of the ground control station is shown in Figure 5.

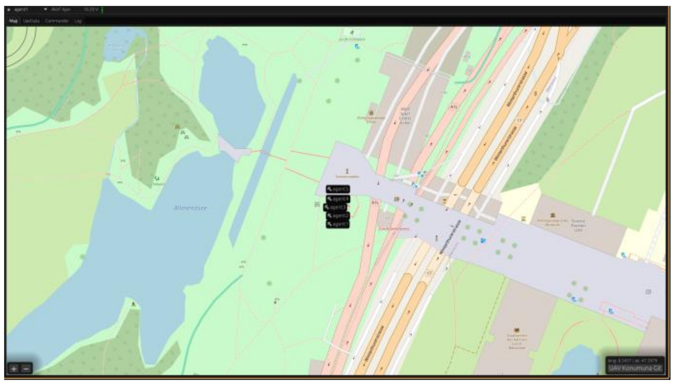

Figure 5. Close-up Map View

The interface enables the selection of the agent to be controlled, displaying instant telemetry data and the control panel of the selected agent. Additionally, individual swarm agents can be moved from one point to another using position controllers.
The user interface for position control of the swarm agents is shown in Figure 6.

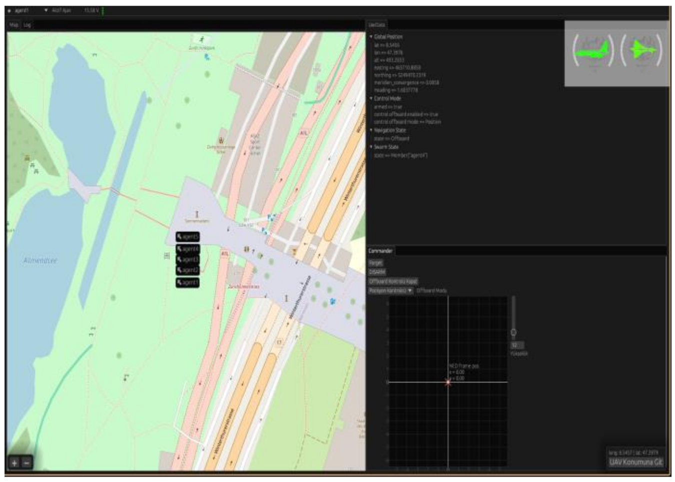

Figure 6. Position Controller User Interface

The position of the swarm robots in the Gazebo simulation environment is shown in Figure 7.

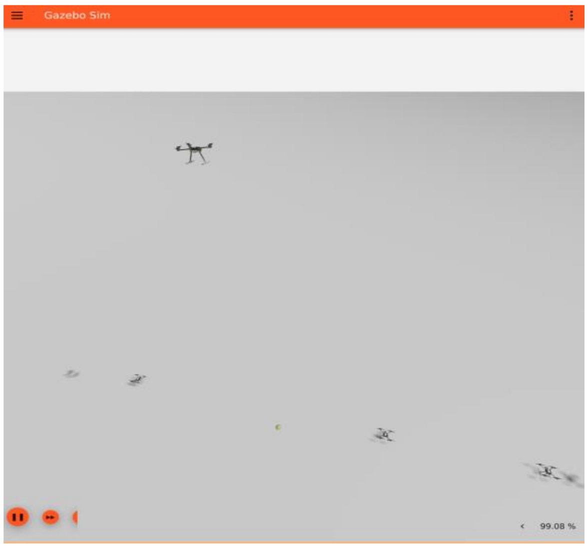

Figure 7. Swarm Robots in Gazebo Simulation

With the selection of the leader agent, the swarm system can be commanded to enter the leader-following state. Figure 8 shows the follower robots forming a v-formation behind the leader robot.

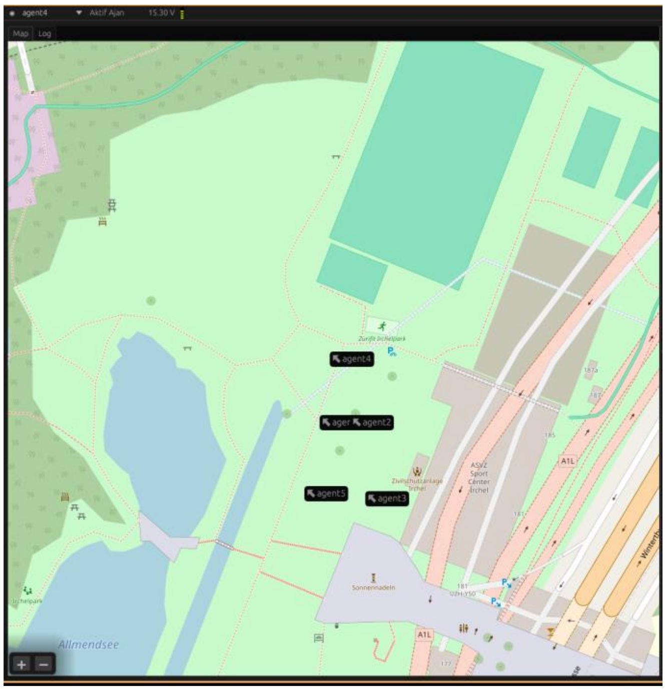

Figure 8. Swarm Robots in V-Formation

The view of the leader following the process in the Gazebo simulation environment is shown in Figure 9.

4. CONCLUSION

This study delves into the intricate realm of swarm systems with distributed architectures, acknowledging the inherent complexity compared to centralized architectures. The focal point is the analysis of formation control within a distributed architecture designed to operate on tangible hardware.

Utilizing PixHawk, an open-source autonomous flight control hardware and software platform, PX4 as open-source autopilot software, and employing ROS2 for implementation, the study takes a practical approach. Gazebo serves as the simulation environment for testing. Because swarm agents lack IP addresses in this system, communication relies on the Zenoh protocol. A standardized communication process is established for dynamic leader selection, and specific message formats are crafted for agents to execute these operations. Each robot is assigned a randomly determined vote weight during the leader election, and the leader is determined based on accumulated votes. The simulation demonstrates the leader's announcement and subsequent command for followership, adopting a V-formation for navigation. Simulation outputs affirm the successful realization of the study. Future endeavors involve extending the study to explore alternative leader selection methods and accommodating multiple leaders.

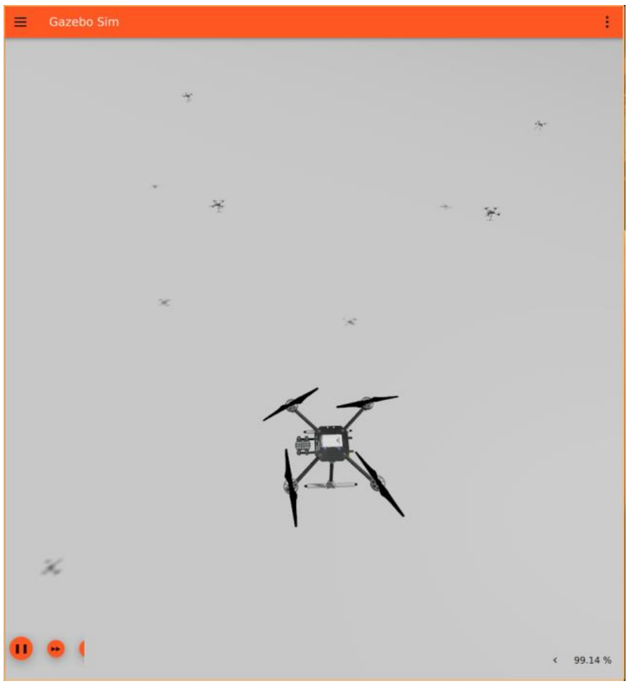

Figure 9. Leader Follower in Gazebo Simulation

REFERENCES

[1] W. Luo, S. S. Khatib, S. Nagavalli, N. Chakraborty, and K. Sycara, “Asynchronous distributed information leader selection in robotic swarms,” in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 606–611. doi: 10.1109/CoASE.2015.7294145.

[2] M. Carpentiero, L. Gugliermetti, M. Sabatini, and G. B. Palmerini, “A swarm of wheeled and aerial robots for environmental monitoring,” in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, 2017, pp. 90–95. doi: 10.1109/ICNSC.2017.8000073.

[3] M. Duarte *et al.*, “Application of swarm robotics systems to marine environmental monitoring,” in *OCEANS 2016 - Shanghai*, 2016, pp. 1–8. doi: 10.1109/OCEANSAP.2016.7485429.

[4] J. Asbach, S. Chowdhury, and K. Lewis, “Using an Intelligent UAV Swarm in Natural Disaster Environments,” in *International Design Engineering Technical Conferences and Computers and Information*

in Engineering Conference, 2018, p. V02AT03A013.

[5] J. Scherer *et al.*, "An autonomous multi-UAV system for search and rescue," in *Proceedings of the first workshop on micro aerial vehicle networks, systems, and applications for civilian use*, 2015, pp. 33–38.

[6] Y. Song *et al.*, "Distributed swarm system with hybrid-flocking control for small fixed-wing UAVs: Algorithms and flight experiments," *Expert Syst. Appl.*, vol. 229, p. 120457, 2023.

[7] S. Dey and H. Xu, "Intelligent Distributed Swarm Control for Large-Scale Multi-UAV Systems: A Hierarchical Learning Approach," *Electronics*, vol. 12, no. 1, p. 89, 2022.

[8] Y. Jia, M. Chen, Y. Gao, and H. Wang, "A Distributed Method to Form UAV Swarm based on Moncular Vision," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 41–48.

[9] B. Zhu and Y. Deng, "Distributed UAV swarm control framework with limited interaction for obstacle avoidance," *Aircr. Eng. Aerosp. Technol.*, no. ahead-of-print, 2022.

[10] R. Rafifandi, D. L. Asri, E. Ekawati, and E. M. Budi, "Leader--follower formation control of two quadrotor UAVs," *SN Appl. Sci.*, vol. 1, pp. 1–12, 2019.

[11] N. H. M. Li and H. H. T. Liu, "Formation UAV flight control using virtual structure and motion synchronization," in *2008 American Control Conference*, 2008, pp. 1782–1787.

[12] D. Xu, X. Zhang, Z. Zhu, C. Chen, P. Yang, and others, "Behavior-based formation control of swarm robots," *Math. Probl. Eng.*, vol. 2014, 2014.

[13] H. Zhang, G. Zhang, R. Yang, Z. Feng, and W. He, "Resilient Formation Reconfiguration for Leader--Follower Multi-UAVs," *Appl. Sci.*, vol. 13, no. 13, p. 7385, 2023.

[14] A. Lazri, E. Restrepo, and A. Lor\'\ia, "Robust leader-follower formation control of autonomous vehicles with unknown leader velocities," in *2023 European Control Conference (ECC)*, 2023, pp. 1–6.

[15] J.-H. Lee *et al.*, "Unmanned Surface Vehicle Using a Leader--Follower Swarm Control Algorithm," *Appl. Sci.*, vol. 13, no. 5, p. 3120, 2023.

[16] N. Pauli and W. Fichter, "Leader-Follower Formation Control with Longitudinal Separation along Lateral and Vertical Shifted Follower Paths," in *AIAA SCITECH 2023 Forum*, 2023, p. 484.

[17] L.-B. Wee and Y.-C. Paw, "Simultaneous Mapping Localization and Path Planning for UAV Swarm," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–6.

[18] H. Y. Liu, J. Chen, K. H. Huang, G. Q. Cheng, and R. Wang, "UAV swarm collaborative coverage control using GV division and planning algorithm," *Aeronaut. J.*, vol. 127, no. 1309, pp. 446–465, 2023.

[19] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *2011 ieee international conference on robotics and automation*, 2011, pp. 2992–2997.

[20] A. Allouch, O. Cheikhrouhou, A. Koubâa, M. Khalgui, and T. Abbes, "MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019, pp. 621–628. doi: 10.1109/IWCMC.2019.8766667.

[21] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240. doi: 10.1109/ICRA.2015.7140074.

[22] A. BEKMEZ and A. Kadir, "Three Dimensional Formation Control of Unmanned Aerial Vehicles in Obstacle Environments," *Balk. J. Electr. Comput. Eng.*, vol. 11, no. 4, pp. 387–394, 2023.

[23] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, p. 5.

[24] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System*. Packt Publishing Ltd, 2018.

[25] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, p. eabm6074, 2022.

[26] J. J. Lopez Escobar, R. P. Diaz-Redondo, and F. Gil-Castineira, "Unleashing the power of decentralized serverless IoT dataflow architecture for the Cloud-to-Edge Continuum: a performance comparison," *Ann. Telecommun.*, pp. 1–14, 2024.

[27] A. Corsaro *et al.*, "Zenoh: Unifying Communication, Storage and Computation from the Cloud to the Microcontroller," vol. DSD 2023, 2023.

[28] Z. Tüfekçi and G. Erdemir, "Experimental Comparison of Global Planners for Trajectory Planning of Mobile Robots in an Unknown Environment with Dynamic Obstacles," in *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2023, pp. 1–6.

BIOGRAPHIES

Abdulmelik Bekmez is currently studying in his bachelor's degree in the Department of Computer Engineering, Engineering Faculty, at the Fatih Sultan Mehmet Vakıf University. His current research areas include robotics and formation control.

Kadir Aram obtained his BSc and MSc degree in computer and control education Marmara University. He completed his PH.D. in Computer Science and Engineering at Istanbul Sabahattin Zaim University in 2023. He is research assistant in Computer Engineering department at Fatih Sultan Mehmet Vakıf University. His research interest are mobile robotics and natural language processing.