A Distributed Sensor Network Architecture for UAVs with Heterogeneous Processing Hardware

Rolando Cortés-Martínez[1,*], Oscar Archila[1] and Johannes Schiffer[1,2]

***Abstract*— Many industrial applications require mobile sensors that can provide high coverage capacity and operational flexibility. Unmanned aerial vehicles (UAVs) represent a promising solution due to their precise maneuverability and ability to operate in targeted environments. The effectiveness of these mobile sensors is further enhanced when multiple UAVs collaborate to form a networked mobile sensor platform (NMSP), in which each UAV shares its local measurements with others. However, the heterogeneous nature of NMSPs in terms of hardware and software makes it difficult to communicate individual measurement data among UAVs in a unified and structured framework. This work provides a framework for sensor data exchange among UAVs. The design is organized into three layers: Low-level control, process management, and communication. The low-level control layer is centered on a flight control unit (FCU), while for the process management is implemented through customized modules under the robotic operating system 2 (ROS 2) middleware running on the OBCs. The communication layer focuses on the design of the sensors' networking and is developed within the ROS 2 framework. We present experimental results using three UAVs sharing measurements from their GPS sensors as well as an onboard light detection and ranging (LIDAR) sensor.**

I. Introduction

The scope of applications for sensors capable of changing their position over time, i.e., mobile sensor platforms (MSPs), spans various domains, including Industry 4.0, hazard prevention systems, search and rescue, security, surveillance, precision agriculture, and mapping, among others.

MSPs significantly enhance sensor performance and versatility by dynamically adjusting their position based on environmental conditions or according to the mission planning of the application. The advantages of MSPs are the adaptability and broader sensing coverage, which allow improved responsiveness in uncertain scenarios and increase time efficiency.

Furthermore, when multiple distributed heterogeneous MSPs interconnect to each other to form networked mobile sensor platforms (NMSPs), their sensor data can enable cooperative tasks in more advanced scenarios.

Comprehensive surveys provide a detailed classification of MSP and NMSP applications (e.g., [1]; [2], and [3]). In terms of the type of mobility, we focus on the use of unmanned aerial vehicles (UAVs) to integrate aerial NMSPs thanks to their dexterity and inexpensiveness [2]. We use the micro aerial vehicle (MAV) type, which corresponds to weights of less than 5 kg. Thus, UAVs are well-suited for navigation in complex or hazardous environments. However, these advantages also introduce significant technological challenges for implementation.

In terms of hardware, a common open architecture for UAV-based NMSPs consists of the use of a flight control unit (FCU) in combination with an onboard computer (OBC) to compensate the limitations of the FCU. The FCU is in charge of the low-level control providing a reliable and tested commecial solution [4], while the latter computes the overall distributed group behavior. Some examples using the pair FCU-OBC are found in [5] and [6].

DroneKit and the robotic operating system (ROS) are software development tools used for the OBC application core. DroneKit software is used in [6] for outdoor flocking of drones. In [5], they implement a decentralized model predictive control scheme and combine ROS with C++ and Python to implement collision avoidance algorithms for UAVs. Some works use the last version, ROS 2, which offers distributed network properties, real-time reliability, and the option to configure the communication quality of service [7]. [8] used ROS 1 for HIL simulation with computer vision. For multi-agent systems control, some works use ROS 1 with numerical simulations (e.g., [9]; [10]). However, ROS 1 has performance restrictions and does not offer real-time guarantees compared to ROS 2 (see e.g., [11]).In [12], ROS 2 is used for an indoor UAVs swarm scenario, while in [13] is used for a single UAV tested with different OBCs.

In summary, current solutions suffer from one of the following characteristics or combinations thereof: Disregard the potential network capabilities of the NMSPs, the details regarding the hardware and software implementation are not disclosed, and there are limitations in terms of hardware and software compatibility for heterogeneous systems.

The present work extends the preliminary presentation given in [14] of the open architecture for NMSP (OA4NMSP). The contributions are as follows: 1) Description of the OA4NMSP software implementation utilizing the latest open-source ROS 2 framework and detailing its interaction and customization with existing open-source hardware and software standard protocols for the FCU. 2) Presentation of the variations of the OA4NMSP to simultaneously adapt to different UAV platforms with varying capabilities, where, despite hardware and software heterogeneity, these platforms integrate within the same network and communicate using a common set of variables. 3) Experimental validation of the proposed OA4NMSP through a UAV demonstration

[1]Control Systems and Network Control Technology Group, Brandenburg University of Technology Cottbus-Senftenberg (BTU C-S), 03046, Germany

[2]Fraunhofer IEG, Fraunhofer Research Institution for Energy Infrastructures and Geothermal Systems, 03046, Germany

*cortesma@b-tu.de

This work is partially supported by the Federal Ministry of Research, Technology and Space (BMFTR), under the scope of the project iCampus2 (16ME0420K).

scenario involving the integration of one LIDAR sensor and three NMSPs. Multiple tests were conducted to statistically evaluate performance in UAV flight trajectories.

In what follows and for the ROS framework of *nodes*, *topics*, *publishers*, and *subscribers*, we use the prefix "/" to refer indistinctly to ROS 2 *nodes* and *topics* (i.e., /this_node, /example_topic).

II. Open Architecture for Networked Mobile Sensors

The open architecture OA4NMSP is divided into three layers: (i) Low-level control, (ii) Process management, and (iii) Communication (see Fig. 1). Consider a MAS of $n > 1$ UAVs. The geographical position of the i-th UAV, $x_i \in \mathbb{R}^2$, is measured inside the low-level layer and sent to the OBC via the Micro air vehicle link (MAVLink) protocol [15]. The OBC can send x_i to the other UAVs and receive the position $x_j \in \mathbb{R}^2$ of the other $n-1$ UAVs, via the communication layer. We describe each block in more detail as follows.

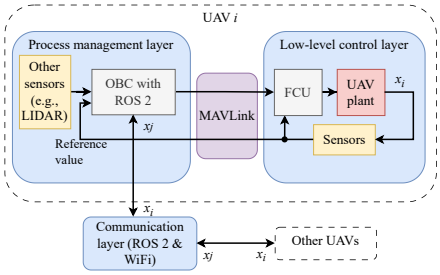

Fig. 1. OA4NMSP layers: Process management, low-level control, and communication.

A. Low-level control layer

The main hardware component of the low-level control layer is the FCU, which is responsible for controlling the flight dynamics of the UAV plant (see Fig. 1).

The FCU includes different controller modes of operation or flight modes [15]. The relevant modes for this work are the*loiter* mode and *guided* mode. The *loiter* mode is for manual flights, where the UAV moves in accordance to the inputs given by the human pilot via the remote control, remaining static in its current position when there are no inputs. In the *guided* mode, the FCU receives commands from the OBC or GCS, provided a previous configuration of FCU parameters [4]: SERIAL2_PROTOCOL=2, SERIAL2_BAUD = 115, and LOG_BACKEND_TYPE=3 (activates the telemetry port *Telem 2* at 115200 bps). We also activate the function return to launch (RTL) for low battery level (BATT_FS_LOW_ACT=2) regardless of the high-level control commands. All the parameters are changed by using MissionPlanner [15].

B. Process management layer

The functions of the process management layer are to administrate the flow of control commands between the control sources and the low-level layer and to carry out high-level control processes, which require either higher computational capacity or information outside the local domain of the low-level control (such as incoming information from external sensors or from other NMSPs). As part of the management of the control commands sources, we define the hierarchy of these sources during the different flight conditions according to the following rules.

- During the *guided* mode, the pilot can change to *loiter* mode at any moment.
- A set of emergency situations has a higher priority over *loiter* and *guided* modes, such as RTL, lack of communication, and flying out of fence barriers.
- The *guided* mode can only be started by a user command or action via the remote control.

The main hardware component of the process management layer is the OBC, which endows code scalability, multi-language code flexibility, and high reliability. Then, additional sensors, such as cameras and LIDARs, can be seamlessly integrated, and the OA4NMSP modularity can be scaled up for applications involving multiple UAVs.
The OBC is also responsible for sending high-level commands to perform actions such as takeoff, landing, and RTL, and for providing references for the position-based low-level control. Another important function of the OBC is to generate MAVLink messages for transmission and reception to/from the FCU. For that purpose, DroneKit and MAVROS (MAVLink extendable communication node for ROS) are two options that offer the necessary support. DroneKit is a Python package with a set of vehicle control-directed methods and functions. It is relatively easy to command a UAV to perform tasks such as arming motors, taking off, sending the vehicle to desired waypoints, and landing, among others [6]. DroneKit is easier to use and better suited to small embedded systems; however, it does not support communication between UAVs. Thus, we combined it with ROS to maintain compatibility. The other software, MAVROS, is a ROS package consisting of a set of ROS *nodes*, *topics*, and *services*, which have a correspondence with the main messages of the MAVLink protocol.

MAVROS connects the low-level hardware (FCU) to the OBC process management application code developed in this work, which runs around the dedicated ROS *node* /uav_i of the i-th NMSP. Table I shows the MAVROS *topics* that interact with our node /uav_i, and the ROS message types. In this way, MAVROS not only provides information from the FCU (e.g., GPS position, flight mode, battery level, etc.) to the /uav_i ROS *node*, but also translates the high-

TABLE I
MAVROS TOPICS RELATED TO NODE /UAV_i

S/P [a]	Topic name	Message type
S	[b]/state	State
S	[b]/global_position/global	NavSatFix
S	[b]/global_position/rel_alt	Float64
P	[b]/setpoint_position/global	GeoPoseStamped

[a] Subscriber/Publisher as seen from /uav_i.
[b] Name-space prefix (e.g., /mavros, /uav_1/state, etc).

level commands from the /uav_i ROS *node* to the FCU. A similar approach is presented in [16]; however, they used ROS 1 and applied the approach to a single UAV. In the DroneKit-based setup, the node /uav_i subscribes to a topic created by a python code that reads the DroneKit-based UAV position. Although MAVROS requires more computational cost than DroneKit, it is a more complete and flexible software package than DroneKit. In this work, we use ROS 2 *Humble* for *Ubuntu 22.04 LTS*.

C. Communication layer

For ROS-based applications, a five-layer stack model was presented by [7]: physical layer, data link layer, network layer, transport layer, and application layer. For each layer, different protocols can be chosen depending on the application.

The application layer of the communication uses the *topic* sharing capabilities between ROS 2-enabled OBCs. We define the set of ROS 2 application-related variables (*topics*) that are shared in the network by the DDS component via the WLAN. To enable direct communication among devices in the network, it is necessary to specify that all machines running ROS 2 have the same ROS domain. Thus, during the software installation and setup of every OBC, we set the corresponding ROS parameter "ROS_DOMAIN_ID" to be equal in all the OBCs.

We can define the communication topology of the network through selecting which *topics* are subscribed to by each OBC, this is aligned with the necessity of an efficient use of the communication bandwidth (see Fig. 2).

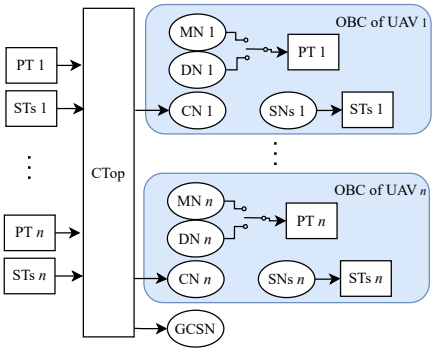

Fig. 2. Conceptual diagram of ROS *nodes* and *topics* for the n NMSPs and the GCS

The i-th UAV with its corresponding i-th OBC can have a MAVROS *node* (MN i) or Dronekit-based *node* (DN i) to publish the UAV's position in a *topic* related to the position (PT i). Whether we have an MN or DN depends on the OBC capabilities as mentioned above. Among the information that can be shared, we have the UAV's positions and velocities, as well as the embedded sensors' data values, for example, LIDAR or cameras. The sensor *nodes* (SNs i) publish their data values in the sensor-related *topics* (STs i). Both i-th PT and STs *topics* are available to the complete network and can be accessed by any of the control *nodes* (CN) inside the OBCs, or even by a GCS *node* (GCSN).

The physical layer of communication among the OBCs, including the GCS, uses a wireless local area network (WLAN) connection based on Wi-Fi technology. WLAN communication is well-suited for reliable high-speed communication requirements across a variety of applications [7]. To set up the WiFi network, we follow a standard configuration for host-defined fixed IP addresses. The user can access every OBC in the network from any computer connected to the WLAN using a secure shell protocol (SSH). Thus, it is possible to edit and monitor the scripts without directly connecting a screen monitor and keyboard to each OBC.

D. ROS 2 nodes implementation

The diagram in Fig. 2 is conceptual and does not reflect actual *node* and *topic* names of the final implementation. A more detailed diagram showing two NMSPs is depicted in Fig. 3, where three devices are represented, the OBC of UAV 1 is an example of the use of a MAVROS-based process management layer. The OBC of UAV 2 is an example of the use of a DroneKit-based process management layer, and the GCS exemplifies how other extra topics, such as the /key_inputs *topic* published by the *node* /node_inputs, the user can send commands online to the UAV 2. For example, the user can indicate to the *node* /uav_2 to subscribe to the /mavros/global_position/global *topic* from the node /uav_1. A second function of the GCS is to send control commands

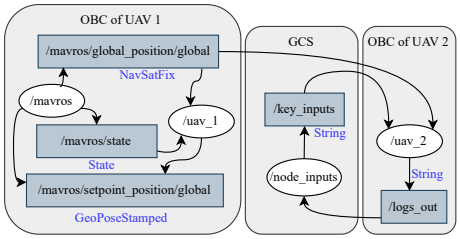

Fig. 3. ROS 2 nodes detailed architecture example for two UAVs. The *topics* are represented in rectangles, the *nodes* in ovals, and the types of messages transmitted for every *topic* are in blue. The "OBC of UAV 2" block can be repeated and interconnected as many times as required for $n \geq 2$ UAVs

to specific or in-group NMSPs, such as arming, takeoff, landing, etc. At the same time, the GCS can provide visual information to the user from the UAVs.

The code developed for the MAVROS-based and Dronekit-based UAVs are found in GitHub [17].

III. Experimental Setup

The experimental setup uses two small UAVs and one medium-sized UAV, three remote controls, a portable computer as a GCS, and a Wi-Fi portable router (see Fig. 4). Each remote control can control only its own paired UAV. The remote controls are used as a controller backup in case the human pilot is required for safety reasons, as stipulated by the European Aviation Safety Agency [18]. Using the remote control, the pilot can switch the UAV operation mode from *guided* to *loiter*, and vice versa, at any time, recall the manual *loiter* mode has higher priority over the automated

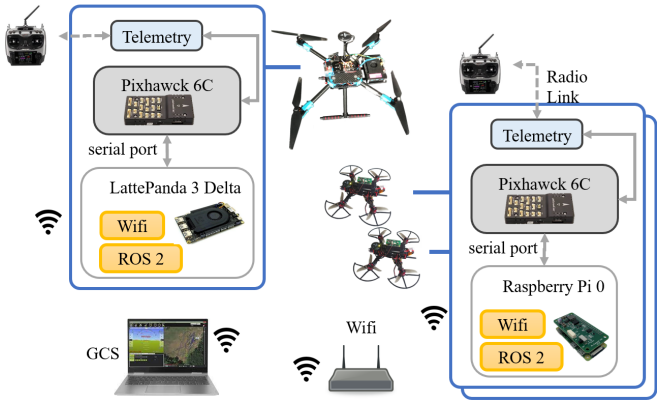

Fig. 4. Experimental setup hardware

guided mode. The communication between the UAV and remote control is through radio link communication devices, which offer a coverage distance range of approximately 1 km. Every UAV has an embedded pair FCU-OBC. The Wi-Fi communication take place among the OBCs and GCS by means of a Wi-Fi portable router in the UAV operation area. The Wi-Fi WLAN uses the 2.4 GHz network band technology and the protocol 802.11n, which offers ease of use and suitable transmission rates.

The UAV's main components are taken from the commercial quadcopter kits QAV250 and X500 V2, both of which are from the Holybro brand. The hardware architecture in both cases comprises four motors with drivers, a power module, a 433 MHz telemetry radio set for communication with a GCS, a 2.4 GHz radio receiver R9DS, a GPS module Neo-M8N, a Lipo battery, and an FCU (see Fig. 5). In both cases, the FCU is the Pixhawk 6C with the

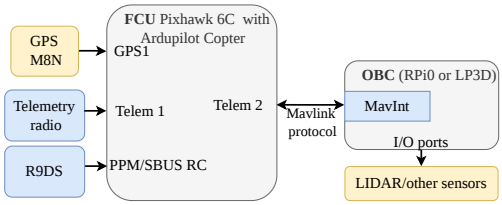

Fig. 5. Electronics diagram of the UAVs for both possible OBC selections: the RPi0 and the LP3D

firmware Ardupilot version 4.3.7, and it is connected by the *telem 2* serial port to the OBC by the MAVLink protocol through a MAVLink hardware connection interface (MavInt). Hardware heterogeneity is introduced at the OBC level: the selected boards are the Raspberry Pi Zero W2 (RPi0) and the LattePanda 3 Delta (LP3D). We use the RPi0 for the small UAVs and the LP3D for medium-sized UAV. The MavInt for the RPi0 is a hat (Pi Connect Lite module) that interfaces the physical connection to the RPi0's UART Rx/Tx serial port. The MavInt component in the case of the LP3D consists of a custom-made level changer between TTL and RS232 levels.

In the next section, an application scenario is proposed to showcase the OA4NMSP. With that in mind, the X500 V2 UAV is equipped with the LIDAR sensor T-mini Pro, from the YDLIDAR brand [19].

More details of the hardware characteristics, including OBCs, are presented in [14]. Table II summarizes the characteristics of the LIDAR sensor.

TABLE II
LIDAR T-MINI PRO CHARACTERISTICS

Item	Value	Units
Scan frequency	6 to 12	Hz
Range distance	0.02 to 12 max.	m
Field of view	0 to 360	Deg
Angle resolution	0.54	Deg

The LIDAR manufacturer provides ROS libraries for the use of the sensor [19]. We use the corresponding ROS 2 driver branch for the *humble* version. By executing the code in ydlidar_launch.py, the measurements of the LIDAR are published by the X500 UAV in the *topic /scan*, with the message type sensor_msgs/LaserScan, and it is subscribed by the QAV250 UAVs and GCS. The LIDAR's scan frequency used in this work is 5 Hz and is connected to the LP3D USB port. The LIDAR rotational axis is parallel to the UAV local vertical axis.

IV. APPLICATION SCENARIO

The flexibility of the OA4NMSP allows for different application scenarios. In particular, we explore an application in the scope of outdoor inspection. The UAVs' objectives in this type of mission can vary depending on the specific requirements. A common set of requirements consists of surveying an area in an experimental setup where multiple physical objects can be found in the surroundings of the area. To showcase how the proposed solution tackles this problem, a demonstration setup is described next.

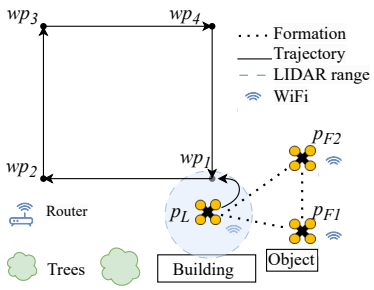

Fig. 6. Minimum setting scenario with three UAVs, one with LIDAR sensor

Three UAVs inspect four waypoints, $wp_i \in \mathbb{R}^2$, in an area of interest, as shown in Fig. 6. A leader-follower scheme is implemented to ensure a proper formation, where the leader is responsible for inspecting the environment first. In this scenario, the leader (X500 V2 UAV) flies through the four waypoints, making a pause at each one. The waypoints form a square polygon with a side length equal to 20m.

The other two UAVs act as followers (model QAV250) and receive the leader's position as a reference position, maintaining a triangular formation w.r.t. the leader. The positions in x, y coordinates of the leader and the followers are $p_L = [x_L \quad y_L] \in \mathbb{R}^2$, $p_{F1} = [x_{F1} \quad y_{F1}] \in \mathbb{R}^2$, and $p_{F2} = [x_{F2} \quad y_{F2}] \in \mathbb{R}^2$, respectively. The references for the followers are $p_{Fr1} = p_L + [7 \quad -2] \times 10^{-5}$ deg, and $p_{Fr2} = p_L + [6 \quad 4] \times 10^{-5}$ deg (longitude-latitude geographical coordinates), where $p_{Fri} = [x_{Fri} \quad y_{Fri}] \in \mathbb{R}^2$, i=1,2. The leader is capable of detecting obstacles in the area by using the LIDAR sensor, and the followers receive the leader's position and information about the objects detected by the leader. The objects surrounding the scenario are trees and a small building. The advantage of this configuration is that both follower UAVs can benefit from the leader's scanning updates. In the scope of the present work, we show how the OA4NMSP allows the sharing of sensor information. The further processing of the data and how this data can lead to control actions on the followers' side is out of the scope of this work. The *nodes* and *topics* architecture corresponds to the diagram in Fig. 3 extended to the case of two followers /uav_2 and /uav_3 and the *topics* shown in Table I.

A. Results and discussion

Fig. 7 shows the trajectories followed by the three NMSPs. The trajectories lay in the horizontal x-y plane, with x and y axes pointing in the same direction as the geographical coordinates of longitude and latitude, respectively. The leader surveys the requested waypoints successfully while the followers maintain the desired formation. In the same figure, the surrounding objects are detected by the LIDAR at the bottom of the diagram, which corresponds to a small building structure and a soccer goal in the test area (see Fig. 6). The average of the six tests and their corresponding standard deviation, shown in Fig. 7, exhibits the high repeatability of the flights, only being considerably large during the transitory response at the beginning of the tracking. The angle of the square path corresponds to the orientation of the soccer field (where the tests were carried out) with respect to the North. In Fig. 8, we show the individual coordinates x and y of the UAVs during the duration of the trajectories (80 s). Again, the standard deviation exhibits higher values only at the beginning. The data source of information stems from the extended Kalman filter (EKF) provided by the FCU of each UAV. The main sensor contributing to the EKF is the GPS system, which typically has an error of approximately 1-2 m. As a consequence, the results in Fig. 7 and 8 do not reflect this error since the EKF signals are taken as the unique reference signals. This limitation should be taken into account in the design of future mission scenarios, where a minimum safety distance offset among UAVs as well as between the UAVs and the surrounding objects is necessary to avoid collisions. Moreover, the LIDAR measurements are severely affected since the GPS error accounts for almost 10% of the LIDAR's total detection range. A possible solution could involve fusing the LIDAR sensor with the EKF and incorporating a real-time kinematic (RTK) system

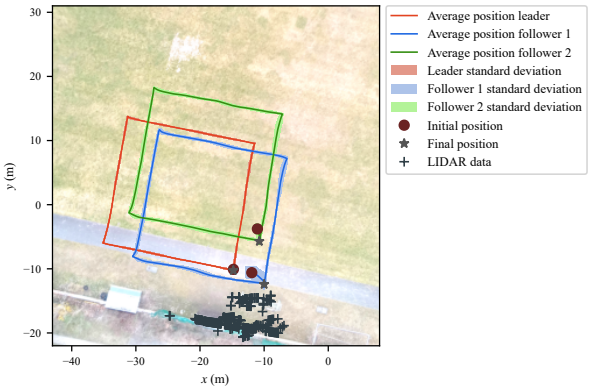

Fig. 7. Experimental results for the six trial leader-follower scheme using the OA4NMSP

to reduce GPS error.

The reference trajectory starts at $t = 10$ s and finishes at t =78 s. The time to change between waypoints is 17 s. The time that the leader remains static in each waypoint is 2 s. The altitude references for the UAVs are $z_{Fr1} = 2$ m, $z_{Fr2} = 3$ m, and $z_{Lr} = 8$ m. There is a delayed movement in the followers w.r.t. the leader's movement of around 2 s; see the start of the movements, for example, in signals x_F at t = 10 s and x_{F1}, x_{F2} at t=12 s in Fig. 8. This behavior is due to the body's inertia in the leader-follower scheme and to the fact that the control law implemented in the process management layer is limited to open-loop waypoint control. For simplicity, it is convenient to maintain open-loop behavior; however, for precise trajectory tracking, the process management layer should take the UAVs' velocity as another input.

The background image shown in Fig. 7 corresponds to the trees, buildings, and objects in the surroundings of the test area. There are no LIDAR markers around the trees because they are defoliated seasonally. There is an object at the bottom-left corner in Fig. 7 which does not have LIDAR detected points. This situation arises during the UAV's takeoff, when the object is outside the detection range. Similarly, when the object is within the detection range during flight, it is not at the same altitude as the UAV, consequently, the object is not detected. However, the other objects at the bottom of the image are detected accurately since they are near the start point.

In Fig. 9, we can see the average of the error signals between the followers and the leader and the standard deviations for the six trial flights campaign, $\|x_{L,k} - x_{Fi,k}\|$, and $\|y_{L,k} - x_{Fi,k}\|$, $i = 1, 2$, $k = 1, ..., 6$.

The evolution for the error signals exhibited in Fig. 9 show a convergence around zero for both followers in some specific time intervals around $t_i = \{(0, 10), (25, 27), (40, 42), (57, 59), (74, 80)\}$. This occurs when the leader reaches every wp_i at zero speed. When the leader's speed is different from zero (nominal speed is set up to 1.5 m/s), there is a constant error of around 2.5 m and 0.5 m depending on the speed components in the x and y

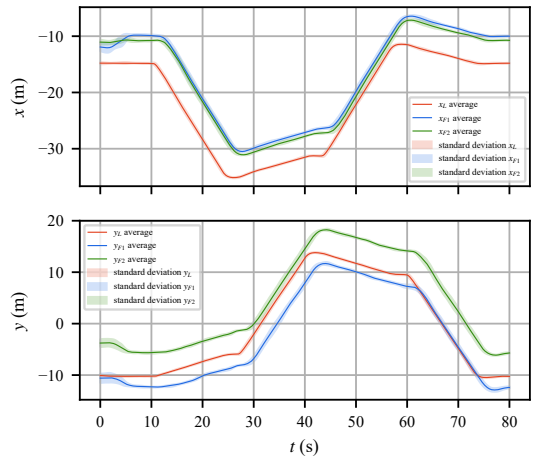

Fig. 8. x, and y coordinates for the leader and the followers (top and bottom, respectively)

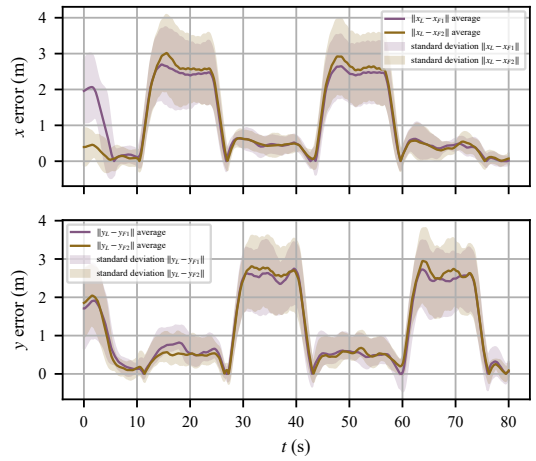

Fig. 9. Control errors for the leader and the follower in y (top) and x (bottom) axes

coordinates. It is to be noted that we are not implementing any compensation in the controller to solve the tracking problem for speeds different from zero, and thus, it is natural to obtain these error magnitudes.

The transmission of the ROS message types sensor_msgs/LaseScan and /NavSatFix, which correspond to the leader's geographical position and the LIDAR scans, occupies the main part of the communication bandwidth. Given sample times of 200 ms and 16 ms for both, these messages imply estimates of 216 kbps and 70 kbps, which are moderate for a standard Wi-Fi communication system.

V. Conclusions

The proposed OA4NMSP exhibits the desired characteristics in terms of scalability and software flexibility inherent in ROS 2. Improved computational capacity is also achieved by using a companion OBC, making the proposed solution well-suited for a range of mobile sensor missions. The options for different computational capacities are also exposed, ranging from one of the smallest boards in the market, such as the Raspberry Pi Zero, to the more advanced Lattepanda 3 delta board. This is made possible by the integration of MAVROS and DroneKit in a unified manner.

Although we are able to share the sensor topics through the network, the data is not specifically processed, and no consequent control action in response is computed. Collision avoidance algorithms and sensor fusion of a network of sensors, like smoke sensors, are further directions to pursue.

References

[1] M. Erdelj, M. Król, and E. Natalizio, "Wireless sensor networks and multi-UAV systems for natural disaster management," *Computer Networks*, vol. 124, pp. 72–86, 2017.

[2] H. Hildmann and E. Kovacs, "Using unmanned aerial vehicles (UAVs) as mobile sensing platforms (MSPs) for disaster response, civil security and public safety," *Drones*, vol. 3, no. 3, p. 59, 2019.

[3] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[4] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source UAV flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, pp. 11–20, 2018.

[5] R. G. Braga, R. C. Da Silva, A. C. Ramos, and F. Mora-Camino, "Collision avoidance based on reynolds rules: A case study using quadrotors," in *Information Technology-New Generations: 14th International Conference on Information Technology*. Springer, 2018, pp. 773–780.

[6] Q. Yuan, J. Zhan, and X. Li, "Outdoor flocking of quadcopter drones with decentralized model predictive control," *ISA transactions*, vol. 71, pp. 84–92, 2017.

[7] L. Shi, N. J. H. Marcano, and R. H. Jacobsen, "A review on communication protocols for autonomous unmanned aerial vehicles for inspection application," *Microprocessors and Microsystems*, vol. 86, p. 104340, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014193312100497X

[8] E. Moréac, E. M. Abdali, F. Berry, D. Heller, and J.-P. Diguet, "Hardware-in-the-loop simulation with dynamic partial FPGA reconfiguration applied to computer vision in ROS-based UAV," in *2020 International Workshop on Rapid System Prototyping (RSP)*. IEEE, 2020, pp. 1–7.

[9] A. P. Lamping, J. N. Ouwerkerk, and K. Cohen, "Multi-UAV control and supervision with ROS," in *2018 aviation technology, integration, and operations conference*, 2018, p. 4245.

[10] S. Khaliq, S. Ahsan, and M. D. Nisar, "Multi-platform hardware in the loop (HiL) simulation for decentralized swarm communication using ROS and GAZEBO," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2021, pp. 310–315.

[11] ROSWiki, "ROS documentation," https://wiki.ros.org/, accessed May 2024, 2024.

[12] L. Pichierri, A. Testa, and G. Notarstefano, "Crazychoir: Flying swarms of crazyflie quadrotors in ROS 2," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4713–4720, 2023.

[13] F. F. Nyboe, N. H. Malle, and E. Ebeid, "MPSoC4Drones: An open framework for ROS2, PX4, and FPGA integration," pp. 1246–1255, 2022.

[14] R. Cortés-Martínez, O. Archila, and J. Schiffer, "An open architecture for networked mobile sensor platforms," in *iCCC2024 - iCampµs Cottbus Conference*, 2024, pp. 161–164.

[15] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro air vehicle link (MAVlink) in a nutshell: A survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.

[16] P. Arias-Perez, J. Fernández-Conde, D. Martin Gomez, J. M. Cañas, and P. Campoy, "A middleware infrastructure for programming vision-based applications in UAVs," *Drones*, vol. 6, no. 11, p. 369, 2022.

[17] fabianarchila and Ro1and0, "fabianarchila/-oa4msp_code: Oa4nmsp," Mar. 2026.

[18] EASA, "Specific category," https:www.easa.europa.eu/domains/civil-drones-rpas/specific-category-civil-drones, accessed: June 2023, 2023.

[19] L. Shenzhen EAI Technology Co., "Ydlidar t-mini pro datasheet," https://www.ydlidar.com/search?searchText=T-mini%20Pro, accessed June 2024, 2024.