DOI 10.5162/iCCC2024/P14

An Open Architecture for Networked Mobile Sensor Platforms

Rolando Cortés Martínez [1], Oscar Archila [1] and Johannes Schiffer [1,2]
[1] Control Systems and Network Control Technology Group, Brandenburg University of Technology Cottbus-Senftenberg (BTU C-S), 03046 Cottbus, Germany
[2] Fraunhofer Research Institution for Energy Infrastructures and Geothermal Systems (IEG), 03046 Cottbus, Germany
cortesma@b-tu.de

Summary: To unfold their full potential, many modern and future industrial applications, require high-performing, agile, and smart sensing systems. For that purpose, networked mobile sensor platforms (NMSPs) comprising multiple sensing agents represent a promising framework. We present a low-cost open architecture for NMSPs (OA4NMSPs) based on unmanned aerial vehicles (UAVs) which enables embedded complex algorithms to run in real-time and communicate with other UAVs or systems, thereby carrying out collaborative work. The architecture is divided into three layers: (i) Low-level control, (ii) Process management, and (iii) Communication. The main technologies employed comprise the flight controller unit (FCU) for the low-level control, the onboard computer (OBC) with ROS 2 framework for the process manager, and the MAVROS package for the administration of the communication among the devices. Core details about the implementation of the OA4NMSPs are provided. The performance is demonstrated in an experimental scenario with two UAVs.

Keywords: Mobile sensor network, ROS, UAVs, communication topology, and control layers

Introduction

The scope of applications related to sensors with the capability of changing their position in time, i.e., mobile sensor platforms (MSPs), varies from a range of domains such as Industry 4.0, hazard prevention systems, search and rescue, security and surveillance, precision agriculture, and infrastructure inspection, among others. The MSPs substantially improve the performance and versatility of the sensor measurements since they can react in accordance with the environment inherent to the application. Thus, covering bigger areas of sensing and adapting their position for uncertain scenarios are the major advantages of the MSPs. Moreover, by networking a set of distributed heterogeneous MSPs (NMSPs), the sensor information is used for cooperative work for more advanced scenarios. Different surveys with examples and classifications of applications for MSPs and NMSPs can be found in [1], [2], and [3].

The architecture of UAVs for NMSP applications is defined by the hardware components, the most relevant being the embedded processor unit, which computes both the UAV individual control and the overall distributed group behavior. The computational load is normally separated into a flight controller unit (FCU), and an onboard computer (OBC), respectively. Some examples can be found in [4], [5], and [1]. While the FCU can be found as a reliable available commercial solution [6], the OBC is open for application based design. In [5], the Dronekit software is used for outdoor flocking of drones. In other works, the open-source ROS framework is used instead, thanks to its real-time characteristics. In [4], ROS is combined with C++ and Python to implement UAVs collision avoidance, while in [7], it is used for hand gesture recognition. Additional works use the last version, ROS 2, for UAV applications [8]. For multi-agent systems control, ROS is used in [9]. They provide details for the use of their developed software, however, the information for experimental implementation is limited.

The current solutions disregard the potential network capabilities of the NMPs, or there is no detailed information about the implementation, or it is limited in terms of software possibilities and flexibility. Consequently, an open architecture for NMSP (OA4NMSP) with detailed information for implementation is in demand. In this work, several key contributions for an OA4NMSP are presented, including a detailed description of the implementation using up-to-date open-source software tools. A second key point is the flexibility of the design, allowing for the seamless integration of heterogeneous sensors and UAVs, and efficient administration of communication within the NMSPs. The proposed solution provides transparent sharing of the UAVs individual status for cooperative tasks and local sensor information. Finally, experimental results are also provided to demonstrate the efficacy of the proposed approach.

The remainder of this paper is organized as follows. In Section , some preliminaries on the employed hardware are given. In Section , there is a description of our proposed architecture. In Section , we present experimental data. Finally, we derive concluding remarks in Section .

Preliminaries on the employed hardware and software

FCU Pixhawk 6C

The FCU is a computer attached to the UAV that is in charge of executing an inner control loop that stabilizes the UAV. The state variables to be stabilized depend on the different modes of operation. For example, the UAV can be sent to a reference position, or the angles must reach a reference angle to maintain a fixed translational velocity indirectly. There are many commercially available FCUs that, in most cases, are used to interpret the commands sent by a remote control and compute the required motor control signals accordingly [6]. For the FCU firmware side, there are two developments that can be used equivalently: Ardupilot and PX4. Both approaches support autonomous flights, among other characteristics. However, the Ardupilot presents a more robust control since it includes internal fault detection and handling features [6].

Onboard computer software and middleware

With regards to the process management software (i.e., the middleware) for controlling a UAV with an OBC, the most common tools are the DroneKit and ROS. In this work, we explore both options since, on the one hand, in this way, we exploit and demonstrate the flexibility of the OA4NMSP, and on the other hand, every one of them can be better fitted into the potentially different devices available in the network.

ROS 2 is a middleware especially suited for robotic applications since it allows concurrent task execution in real-time. ROS programming is based on the concept of *nodes*, *topics*, *publishers*, and *subscribers*. Every process can be executed as a single *node* with its own loop rate. To share information between nodes, the sending *node* writes the information (with a *publisher*) into a *topic*. Any other *node* can read the information on the *topic* by creating a *subscriber* to it. On the other hand, DroneKit is a Python library with a set of vehicle control-directed methods and functions. It is relatively easy to command a UAV to perform tasks like the arming of motors, taking off, sending the vehicle to a desired GPS position, and landing, among others.

Communication

The lower level of communication between the UAVs in the NMSP and the ground station uses a WLAN connection though Wi-Fi technology. A higher level of communication is related to the application data level. At this level, ROS 2 provides the information interchange using the Mavlink protocol. Mavlink standardizes the message structure for sending and receiving commands and information between remote control vehicles, remote controllers, and the ground control station (GCS) in general [10]. Every topic or node of each device connected is visible to any other device in the same WLAN. This is done by a ROS package called MAVROS, which is composed of a set of ROS nodes specifically designed to administrate the Mavlink protocol.

Open architecture for network mobile sensors

The open architecture OA4NMSP is divided into three layers: (i) Low-level control, (ii) Process management, and (iii) Communication.

The low-level control layer comprises the FCU, responsible for controlling the UAV's mechanical dynamics. The FCU takes the information from the inner navigation and attitude sensors and computes the control actions for the UAV motors. Depending on the operation mode, the behavior of the controller can be different. The most common important operation modes for the current work are *loiter*, and *guided*. In *loiter* mode the drone maintains its current position, allowing it to remain stationary in the air without any intervention from the pilot. The guided mode allows for navigation to single points commanded by a standard GCS or an OBC via Mavlink commands, i.e., an automatic control of the position.

Next, the process management layer runs complex computing processes on an OBC. The embedded code inside this hardware fundamentally requires to have scalability, multi-language code flexibility, and high reliability. Only then can additional sensors, such as cameras and lidars, be easily attached, and the OA4NMSP can be easily and modularly scaled to applications with several UAVs.

Finally, the communication layer maintains the connection between the devices and uses the capabilities of interconnection between machines from ROS 2, as exposed in section . The communication topology of the network is part of the functions defined in the GCS (CT-GCS), see Fig. 1. Every node uav_i, $i = 1, ..., n$, has a corresponding *i*-th set of sensor data published in the topic s_i. In principle, MAVROS makes s_i visible to any other uav_j. However, for efficient use of the communication bandwidth, not all of the topics are subscribed by the uav_j nodes. The available interconnections are defined in an adjacency matrix $A = \{a_{ij}\}$, with entries $a_{ij} = 1$ if node uav_j subscribes to topic s_i , and $a_{ij} = 0$ otherwise. The CT-GCS, publishes A to a topic shared to all uav_i nodes at the start of the mission. The flexibility to freely select A opens the possibility to further apply the results from multi-agent control theory [11]. It is to be noted that in this approach, the UAVs can share not only their spatial position and velocity $\{x, y, z, \dot{x}, \dot{y}, \dot{z}\}$, but also information about their embedded sensors, for example, Lidar or cameras. The sensor data can be centralized and handled to have a complete overview of the status of the object of study. The analysis of the information can then yield a new reallocation of the NMSP. The diagram in Fig. 1 gives a conceptual representation of the general architecture. A detailed diagram showing an example with two NMSPs

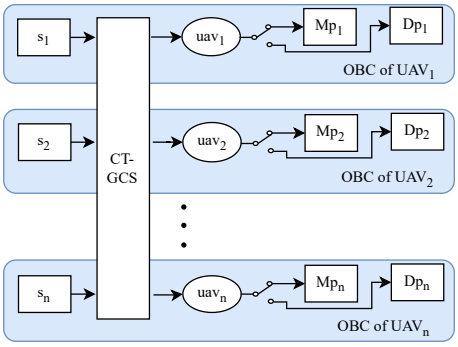

Fig. 1: General diagram of ROS nodes and topics for the NMSP. Every node uav_i receives the information from a selected group of topics s_i. Every node can publish the reference position to the topics Mp_i (MAVROS approach) or is sent to Dronekit functions Dp_i, depending on the heterogeneous nature of the OBC

is shown in Fig. 2. We show only the most relevant nodes related to the OA4NMSPs concept. Three devices are represented, the OBC of UAV1 is an example of the use of a MAVROS-based process manager. The OBC of UAV2 is an example of the use of a DroneKit-MAVROS-based process manager, and the GCS exemplifies how, through the */adj_mat* topic published by the node */com_top* the two OBCs nodes interact with each other. When the nodes */uav* and */uav2* subscribe to the topic */adj_mat*, they obtain the CT-GCS as the A matrix, that in this case, allows the */uav* to subscribe only to its corresponding topics, while the node */uav2* can subscribe to the */mavros/global_position/global* topic from the UAV1. A second function of the GCS is to send

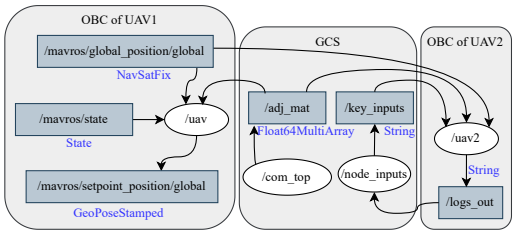

Fig. 2: ROS 2 nodes detailed architecture example for two UAVs. The topics are represented in rectangles, the nodes in ovals, and the types of messages transmitted for every topic are in blue.

control commands to specific or in-group NMSPs like arming, take off, landing, etc. At the same time, the GCS provides information from the selected UAVs.

Experimental setup

A use case for implementing the OA4NMPS is conducted. First, we describe the components to build on the architecture and how they are physically interconnected. Next, we describe the sample mission, and finally, we show the results.

Testbed components

The experimental setup uses two different-sized UAVs, two remote controls, a portable computer as a GCS, and a Wi-Fi router (see Fig 3). Employing the remote control, the pilot can switch the UAV operation mode from *guided* (the software defines the control actions) to *loiter* (the user defines the control actions) at any time. Every OBC and the GCS are provided with Wi-Fi modules to make this possible. The UAV's

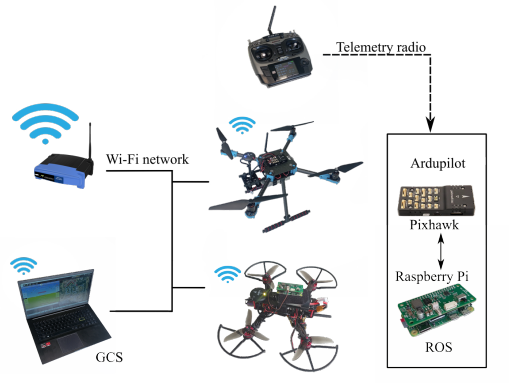

Fig. 3: Every UAV is connected via its OBC Wi-Fi module. Every FCU communicates only with its corresponding OBC.

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 433MHz telemetry radio set for communication with a GCS, a 2.4GHz radio receiver R9DS, a GPS module Neo-M8N, a Lipo battery, and an FCU. In both cases, the FCU is the Pixhawk 6C with the firmware Ardupilot version 4.3.7, and it is connected by the *telem 2* serial port to the OBC. Table 1 summarizes the characteristics of both UAVs.

Results

A sample mission that can fit in the scope of agriculture inspection is set up for two UAVs with potentially complementary sensor capabilities. This corresponds to a sample scenario where every UAV has a different type of camera for recognition/detection on the surface to be studied. Four main spots in the area should be inspected by the two UAVs at the same time. This scenario can be implemented with a leader-follower scheme. The leader (X500 V2 UAV) flies through the four waypoints, making a pause in every one of them, while the UAV follower (QAV250) receives the position of the leader and takes it as a reference, maintaining an offset of 4 m in the

Tab. 1: Set up for every UAV

	QAV250	X500 V2
OBC hardware	Raspberry Pi Zero w2 with Pi Connet Lite module. Single-core Processor up to 1 GHz, 512 MB RAM	LattePanda 3-Delta and 12V inverter. 4-core Processor 11th gen. up to 2.9 GHz, 8GB RAM.
OBC firmware	Dronekit and ROS 2	MAVROS and ROS 2
Battery	Lipo 1300 mAh	Lipo 3500 mAh
Size	250 mm	500 mm

latitude coordinate. As a result of the test carried out, Fig. 4 presents the trajectories followed by the NMSP in the horizontal plane. In this way, the leader has surveyed the requested waypoints successfully while the follower maintains the desired offset of 4 m in the longitude coordinate (x-axis in the graph).

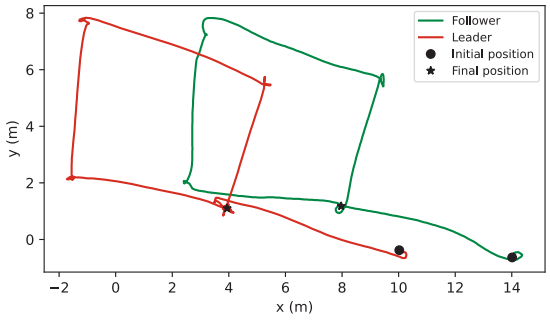

Fig. 4: Experimental result for leader-follower scheme using the OA4NMSP

Conclusions

The proposed OA4NMSP exhibits the desired characteristics in terms of scalability, flexibility, and computational capacity, all these, for performing different types of mobile sensor missions. The options for different capabilities in terms of computational capacity are also exposed, ranging from one of the smallest boards in the market, like the Raspberry Pi Zero, up to the more advanced Lattepanda board.

Our solution offers reliability and compatibility with products in the market thanks to the use of up-to-date components and flexible development firmware. The results can also be scalable to several UAVs in swarm applications. For that purpose, it is necessary to explore the different communication topologies and expand the communication range, incorporating cutting-edge technologies like 5G.

References

[1] C. Zecha, J. Link, and W. Claupein, "Mobile sensor platforms: Categorisation and research applications in precision farming," *Journal of Sensors and Sensor Systems*, vol. 2, no. 1, pp. 51–72, 2013.

[2] 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.

[3] 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.

[4] 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*, pp. 773–780, Springer, 2018.

[5] 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.

[6] 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.

[7] Y. Yu, X. Wang, Z. Zhong, and Y. Zhang, "Ros-based uav control using hand gesture recognition," in *2017 29th Chinese Control And Decision Conference (CCDC)*, pp. 6795–6799, IEEE, 2017.

[8] W. Meng, Y. Hu, J. Lin, F. Lin, and R. Teo, "Ros+ unity: An efficient high-fidelity 3d multi-uav navigation and control simulator in gps-denied environments," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pp. 002562–002567, IEEE, 2015.

[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*, p. 4245, 2018.

[10] 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. 87658–87680, 2019.

[11] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.