Droneport: From Concept To Simulation

Ondřej Severa
osevera@ntis.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

Zdeněk Bouček
zboucek@kky.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

Petr Neduchal
neduchal@kky.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

Lukáš Bláha
frere@ntis.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

Tomáš Myslivec
tmyslive@ntis.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

Miroslav Flídr
flidr@kky.zcu.cz
Faculty of Applied Sciences
New Technologies for the Information Society
University of West Bohemia
Plzeň, Czech Republic

ABSTRACT

The paper describes the simulation environment for the autonomous battery management system for drones called Droneport. First, the problem of battery management is briefly described, and specific experimental and commercial solutions are listed. Then the concept of the proposed Droneport system is described. The next section is fully dedicated to the description of the individual components of the simulation: simulation of the drone, Droneport, the outer environment, and the traffic controller. The function of the individual components is illustrated by the example of two drones performing a predefined mission.

CCS CONCEPTS

• **Computer systems organization → Robotics**; • **Applied computing** → Transportation; • **Computing methodologies → Real-time simulation**.

KEYWORDS

Aerial robotics, Drones, Battery management, Robot simulation, Traffic control

ACM Reference Format:
Ondřej Severa, Zdeněk Bouček, Petr Neduchal, Lukáš Bláha, Tomáš Myslivec, and Miroslav Flídr. 2021. Droneport: From Concept To Simulation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In many drone applications, it is advantageous and sometimes necessary to operate several drones at once for long periods of time [5, 8]. This can be challenging as the number of drones grows, but also risky from the standpoint of safety and costly with respect to the number of human operators.

After robust control, the most important element for ensuring the drone stays airborne is the power-supply. In most applications, the drone cannot be physically connected to the ground for constant power replenishment due to, for example, the consequent limitation of operational area and the increase in weight of the whole system due to the weight of the distribution system. Therefore, the drone must carry its own power source. However, with a power source that has great capacity, the overall weight of the system increases, which leads to an increase in power consumption. The vast majority of drones are powered by electricity from lithium polymer (LiPo) batteries. These batteries typically allow a maximum flight time in the lower tens of minutes.

One option to keep the drone operational is to recharge the battery, which can be done automatically without human involvement [5, 11]. However, that solution requires the drone to remain on the ground for long periods of time and results in blocking the charging station for other drones. Hence it is suitable for single drone systems where the drone does not need to be in the air most of the time but is flying at given intervals. Another solution is to leave the battery replacement to a human operator, who can also operate the charging station. However, this is not the most suitable as drones are mostly employed to increase safety while reducing costs.

A more convenient option is to replace the battery with an automatic station. There are several experimental [6, 10] but nowadays also commercial solutions [1]. Experimental systems are tightly coupled to a particular type of drone or battery. Commercial *drone-in-box* systems vary greatly in parameters and support for different drones (most commonly the DJI Mavic 2). They focus on safety, autonomy, and ease of user accessibility. A few systems even support battery swapping. Our proposed system called Droneport (DP) was introduced in [2]. It can operate various drones with Vertical Take-Off and Landing (VTOL) ability. It is capable of swapping batteries of different sizes and shapes. Unlike the above-mentioned commercial systems, DP will be open-source with published source code for the traffic controller, MAVLink API description, etc.

In addition, DP will adaptively rally drones for battery swapping according to their status and mission progress. Thus, it does not need to control the drone's mission directly but only monitor progress, initiate a battery replacement event based on the current state, and afterward return the drone to its original mission.

The article is structured as follows. The next section briefly introduces the DP system and all its components. The next section describes the DP with respect to simulating the entire system in a virtual environment. Finally, the future development of the system is outlined.

2 CONCEPT

This section will briefly describe the concept of the DP system, which was introduced in more detail in [2]. The system is developed in cooperation between the University of West Bohemia and the company SmartMotion s.r.o. and it consists of both software and hardware components.

The backbone component of the entire system is the Droneport which is a landing port equipped with a smart battery charger and a manipulator that removes the drone's battery, places it in the charger, and then inserts another charged battery into the drone. The DP is also equipped with an ArUco marker, which is used as a landing marker. Another HW component is the computer that communicates with the drones, tasks the exchange platform, and runs Droneport Traffic Control (DPTC). The drones themselves are required to communicate using the MAVLink protocol [4] and be equipped with a down-facing camera to guarantee a precise landing on the device.

Software components include DPTC and a precision landing control system. The DPTC evaluates data about the drones and the battery charging system to make decisions about scheduling battery swaps. The precision landing control system processes the camera image to detect the ArUco marker and then estimates the position and orientation of the drone relative to the platform, with subsequent calculation of adequate control commands. The simulation of the DP system, which will be presented in more detail in the following section, is a component that is especially crucial for testing and debugging other software components, testing the physical parameters of the system, and guaranteeing smooth operation of the entire project.

3 SIMULATION

In the first part of this section, the whole simulation, including communication between individual building blocks, will be described. Then the description will be focused on each building block separately.

The whole project uses *REXYGEN* as a main real-time control system. It is based on the programming without hand-coding using the libraries of the functional blocks, which are predefined for certain uses, or there is an opportunity to use some special blocks which the user can code by himself. REXYGEN provides the whole package of software needed for automation. First, there is a multi-platform run-time core of the whole system – *RexCore* –, the algorithms are configured via *REXYGEN Studio* and compiled for real-time usage using *RexComp*. The bundle contains *HMI Designer*, which is used for the configuration of the web-based human-machine interface.

The next part of our prototype is the control system for our drones, the open-source flight software *PX4* is used for drones and other unmanned vehicles. The main benefit, in addition to open-source license, is an active community and lots of industrial applications. Thus, the PX4 is very reliable and also easy to use, even for very specific applications. PX4 uses MAVLink protocol for communication with other devices, so it is very easy to use, such as Raspberry Pi or UpBoard with REXYGEN installed. With this setup, we are able to command a drone with more advanced tasks like autonomous inspection or traffic control for swapping batteries. PX4 includes a Software-in-the-loop (SITL) simulator, which is one of the key components for the simulation.

For environment simulation purposes, we use *GAZEBO*, which is the entire solution for robot simulation. It consists of features like dynamic simulation, 3D graphics, sensors, and noises, custom plugins for specific applications, predefined robot models, TCP/IP transport, and also cloud simulations. We use it mainly for rapid prototyping our control algorithm and ensuring that everything works as expected. When you deal with unmanned aerial vehicles, there is a huge, dangerous potential, so simulations are necessary. Gazebo library also consists of plugins for PX4 systems, which are essential for our work.

3.1 Drone

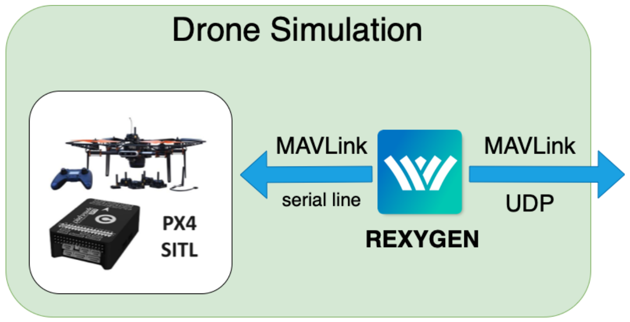


Figure 1: Drone simulation

The system was designed to work with drones that can communicate over the MAVLink interface. Usually, the flight controller is based on the PX4 or ArduPilot. The drone must be able to land very precisely. Thus standard GPS is not sufficient. The drone must be equipped with a small camera and software which is able to detect ArUco based markers to improve navigation accuracy during landing.

Our drone uses a combination of flight controller PX4 and companion computer (Aaegon UpBoard) with REXYGEN. The companion computer processes the image from the camera using OpenCV algorithms. There is a Python-based service with the MAVLink interface that periodically sends the position of ArUco markers seen in the scene. Such a message is processed in the companion computer using REXYGEN. Then the flight controller of the drone is commanded via Offboard mode for precise landing.

The simulation of the drone uses environment simulation in Gazebo (see section 3.3), SITL of the PX4 project, and one RexCore process as a simulation of the companion computer.

On top of the standard PX4 SITL, there is a small component that simulates the battery. This subsystem is injected between the traffic controller and PX4 simulation in the REXYGEN instance.

The drone simulator listens on predefined UDP port. The communication is routed via the MAVLink router, which is a part of the REXYGEN system. The messages are either routed directly to the PX4 instance via an additional UDP (in simulation mode) or serial line (in real device) connection. In the simulation, the messages regarding the battery status from PX4 are filtered, and new simulated messages are created in the REXYGEN subsystem. We can achieve battery discharge simulation which is not done properly in the PX4.

The remaining parts of the drone simulation use only PX4 SITL with no changes.

3.2 Droneport simulation

The current version of the Droneport simulation is focused on communication interface definition, support of the drone landing algorithms development, designing of the high-level controller.

Real DP will contain a standalone battery charger (we assume SKY RC Q200) which is capable of charging four batteries at once. Thus the simulator is able to mock the SKY RC API with the simple simulation of the charging and discharging process of the Li-pol batteries. In Figure 2, the SW control block of the battery holder. Each holder can contain one battery. If the holder is connected to the charging station, it manages the connected battery. So when the robotic manipulator inserts the battery into the holder, the charging process starts automatically.

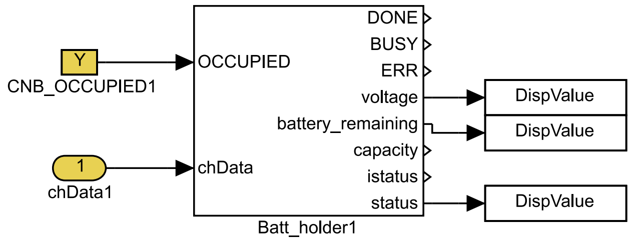


Figure 2: Battery holder subsystem interface

The DP simulator also includes the interface with the drone and traffic controller. DP can receive commands over the MAVLink network, and it also broadcast its state. Thus the 3D model in the environmental simulation can be adjusted accordingly in real-time.

3.3 Environment simulation

The next step in the simulation is to prepare a simulation environment capable of 3D visualization and physics simulation. There is multiple software that can be used for this task. For example, the AirSim [7] simulator developed by Microsoft or Flightmare [9] simulator can be mentioned. The third option that we used in this particular case is Gazebo [3], a well-known simulator in the robotic community. One of the reasons that we have chosen Gazebo is the PX4-SITL_gazebo[1] plugin suite which is supposed to enable PX4 drone simulation in Gazebo.

Thus, it is possible to simulate drones such as the Iris drone in Gazebo with the PX4 controller. Besides the drone, the goal was to develop a simulation of the Droneport device. In order to fulfill this goal, several steps had to be addressed. The development of the DP simulation model was the first one. Models in Gazebo are created using Simulation Description Format[2] (SDF), which is based on Extensible Markup Language (XML). In Figure 3, there is the most recent version of the Droneport simulation model in its closed state.

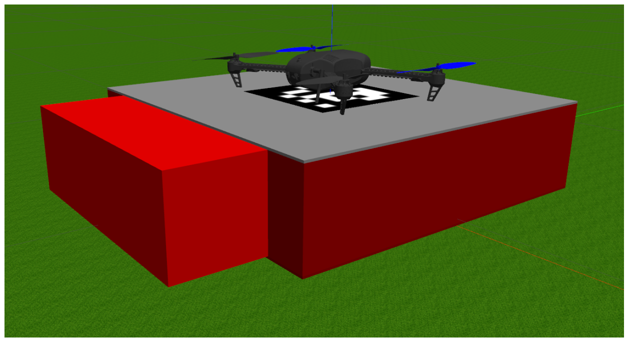


Figure 3: Droneport simulation model - closed state

In the closed state, it can be used as a landing port for the drone, but it cannot change batteries. On the other hand, this can be done when the Droneport is open. It is shown in Figure 4.

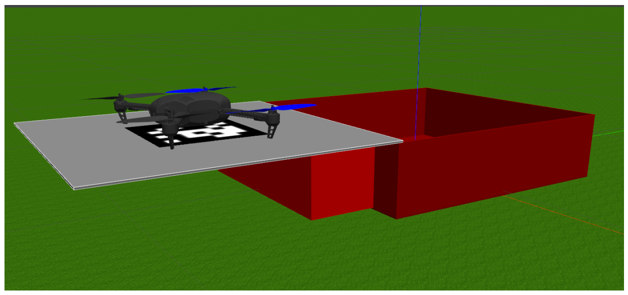


Figure 4: Droneport simulation model - open state

Besides the simulation model, it was necessary to develop a plugin for communication with other parts of the simulation, such as the PX4 controller, and for visualization of the battery charging process. The plugin is called gazebo_Droneport_plugin. It is based on MAVLink interface plugin from the PX4-SITL_gazebo plugin suite. The plugin contains MAVLink based communication over both the UTP and TCP protocols – the only one that should be selected for the simulation. In particular, it receives information about the state of the batteries plugged into the Droneport, from the battery simulator described above, in a message called BATTERY_STATUS.

[1]https://github.com/PX4/PX4-SITL_gazebo
[2]http://sdformat.org/spec

Simultaneously, it provides its own GPS coordinates using a MAVLink message called HIL_GPS back to the simulation network. It enables the possibility of working in the same GPS coordinates in both the Gazebo and the QGround Control software used for planning drones missions. Mission planning and traffic control will be described in detail in the next section of this paper.

3.4 Droneport Traffic Control

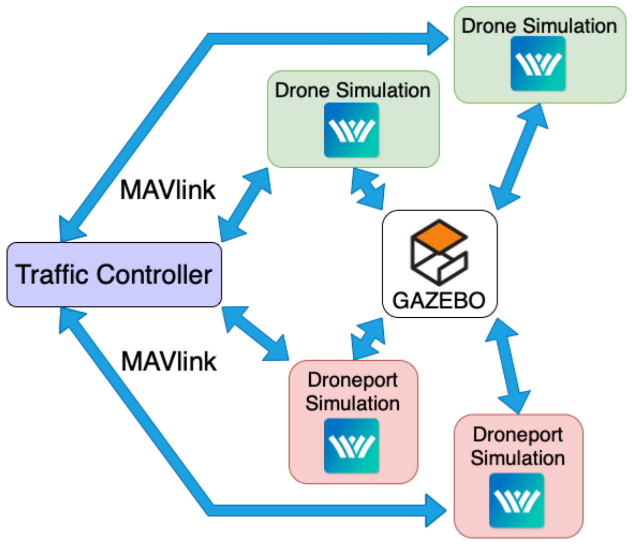


Figure 5: Simulation of two drones with traffic controller

If Droneport is the backbone of the whole system, then Droneport Traffic Control is the brain. In the current version, DPTC monitors the status of the drones and withdraws them to the Droneport as needed when the battery State-of-Charge (SoC) is low. DPTC was tested in a simulated scenario with two drones, where each drone is assigned to its own Droneport. Two different missions were designed for testing in the simulation, in which the park and adjacent buildings are monitored. Thanks to this simplification, there is no need to schedule battery swaps yet to avoid collisions. In standard mode, the drone flies a predefined cyclic mission. At SoC 10% the drone is withdrawn to DP to change the battery and then continues the mission from the last waypoint. A simplified state diagram of the drone is shown in Figure 6.

DPTC is implemented in Python and utilizes the pymavlink[3] library to communicate with drones and the platform. For testing, a simple console user interface was designed using the Urwid library[4].

The monitoring of the whole simulation is depicted in Figure 7, which shows QGroundControl (QGC) as a standard control center (that could be used on the client-side), the simulation of the physical devices in Gazebo, and the previously mentioned DPTC interface. In Figure 8, you can see the two drones, Droneports, the area under consideration, the specified missions, and other standard information provided by QGC.

[3]https://github.com/ArduPilot/pymavlink
[4]http://urwid.org/

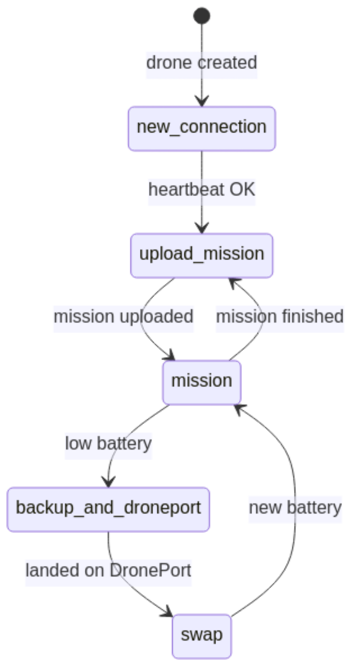


Figure 6: Simplified chart of drone states

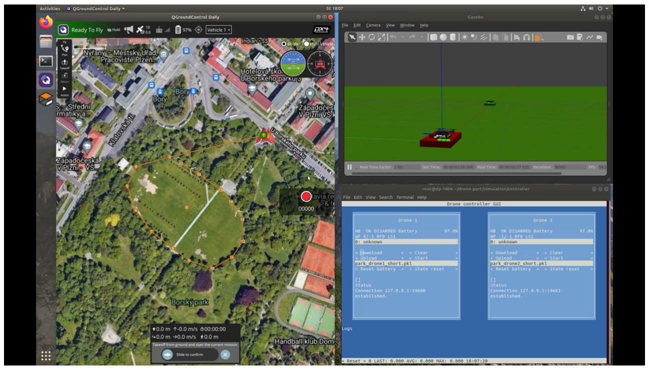


Figure 7: Running simulation

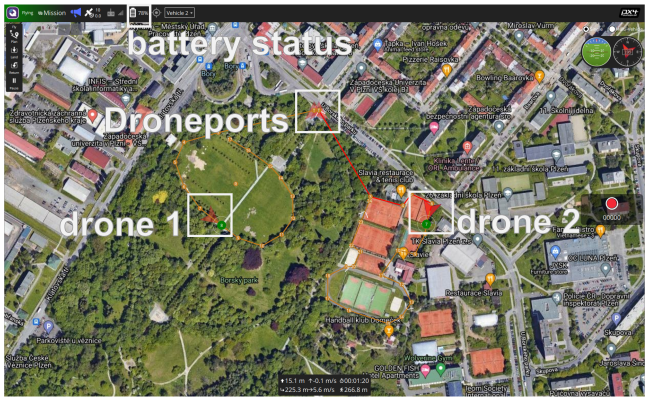
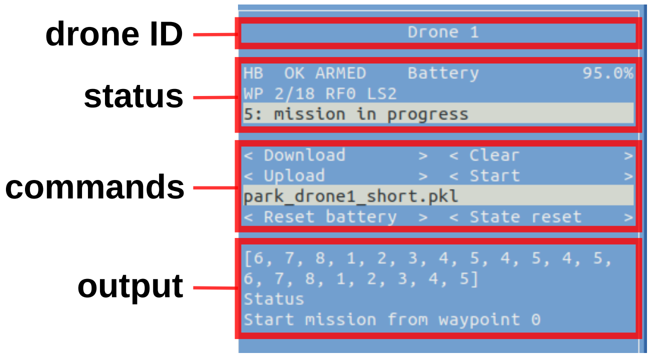


Figure 8: View from QGroundControl



```
Drone 1
HB  OK ARMED     Battery          95.0%
WP 2/18 RF0 LS2
5: mission in progress
< Download         >  < Clear          >
< Upload           >  < Start          >
park_drone1_short.pkl
< Reset battery    >  < State reset    >
[6, 7, 8, 1, 2, 3, 4, 5, 4, 5, 4, 5,
6, 7, 8, 1, 2, 3, 4, 5]
Status
Start mission from waypoint 0
```

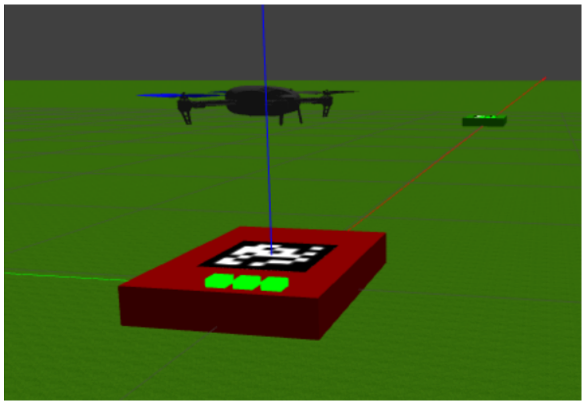
Figure 9: User Interface of Traffic Controller

The console user interface contains information and commands for both drones, and it is easily adaptable for more drones. Figure 9 shows the part that relates to the first drone. The header indicates the drone ID. Below that is the status, which consists of heartbeat information, whether the drone is armed or disarmed, battery SoC, mission progress, and finally, a number and description of the drone's status. Drone commands are also included. You can navigate between commands by using the arrow keys and selecting the command by pressing the Enter key. It is possible to specify the name of a mission file, which can then be uploaded to the drone but also saved from the drone to the file, delete the mission from the drone, and also start the mission from the first waypoint. There are also commands to reset the drone's battery, which will instantly change the SoC to 100%, and to reset the drone's state machine to wait for the first heartbeat. Finally, the output is shown with previous drone states and the confirmation of the last command, e.g., *start mission from waypoint 0*.

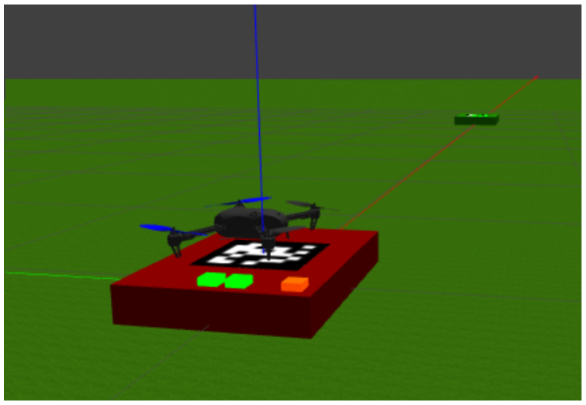
The SoC indication of DP's batteries is made directly in Gazebo (see Figure 10, where green corresponds to a fully charged battery and red to a fully discharged battery. The simulation also indicates in which slot the battery is located. Figure 10a shows the drone landing with a discharged battery, and it can be seen that the batteries in all slots of the DP are *green*, i.e., fully charged. In Figure 10b, the drone is taking off with the battery already replaced. One slot of Droneport with a charged battery has been freed, and a previously empty one has been fitted with the battery, which is now being recharged.

4 CONCLUSION

The paper described the Droneport system with an emphasis on simulation. First, the concept of the whole system was outlined. Next, the key components of the simulation were described. The drone was described with its upboard computer, extensions compared to the standard PX4 equipped machine, and the role of REXYGEN as the main real-time control system was presented. Then, a real Droneport with the considered charger and a simulation of the Droneport as an interface were briefly described. Next, the simulation environment created in Gazebo was described with all its main features, such as communication or battery status visualization. Finally, the current version of traffic control was described,



(a) Drone landing on the Droneport with discharged battery



(b) Drone taking off from Droneport with the fully charged battery

Figure 10: Indication of battery State-of-Charge on Droneport in simulation

and a simple example of a simulation with two drones and two Droneports was given.

The described simulation will allow easier testing and development of the real-world system. Future work on the Droneport system will include completing and describing the hardware components and extending the traffic control with a time-scheduler for optimal utilization of the entire system.

ACKNOWLEDGMENTS

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826610. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands.

REFERENCES

[1] Basil Alosious. 2021. 8 best DJI-compatible drone docking stations to consider for autonomy. https://flytnow.com/dji-compatible-docking-stations/
[2] Zdeněk Bouček, Petr Neduchal, and Miroslav Flídr. 2021. DronePort: Smart Drone Battery Management System. In *Interactive Collaborative Robotics*, Andrey Ronzhin, Gerhard Rigoll, and Roman Meshcheryakov (Eds.). Springer International Publishing, Cham, 14–26.

[3] Nathan Koenig and Andrew Howard. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3. IEEE, 2149–2154.

[4] Anis Koubâa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. 2019. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access* 7 (2019), 87658–87680. https://doi.org/10.1109/ACCESS.2019.2924410

[5] Danylo Malyuta, Christian Brommer, Daniel Hentzen, Thomas Stastny, Roland Siegwart, and Roland Brockers. 2020. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. *Journal of Field Robotics* 37, 1 (2020), 137–157. https://doi.org/10.1002/rob.21898

[6] Bernard Michini, Tuna Toksoz, Joshua Redding, Matthew Michini, Jonathan How, Matthew Vavrina, and John Vian. 2011. Automated Battery Swap and Recharge to Enable Persistent UAV Missions. In *Infotech@Aerospace 2011*. American Institute of Aeronautics and Astronautics, Reston, Virigina, 0–10. https://doi.org/10.2514/6.2011-1405

[7] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*, Marco Hutter and Roland Siegwart (Eds.). Springer International Publishing, Cham, 621–635.

[8] Hazim Shakhatreh, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. 2019. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* 7 (2019), 48572–48634. https://doi.org/10.1109/ACCESS.2019.2909530

[9] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. 2020. Flightmare: A Flexible Quadrotor Simulator. In *Conference on Robot Learning*.

[10] N. Kemal Ure, Girish Chowdhary, Tuna Toksoz, Jonathan P. How, Matthew A. Vavrina, and John Vian. 2015. An automated battery management system to enable persistent missions with multiple aerial vehicles. *IEEE/ASME Transactions on Mechatronics* 20, 1 (2015), 275–286. https://doi.org/10.1109/TMECH.2013.2294805

[11] Yafeng Wang, Qinbo Sun, Tristan Berger, and Weimin Qi. 2021. A Drive-through Recharging Strategy for a Quadrotor. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 420–425. https://doi.org/10.1109/ICRA48506.2021.9561482