This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 861111

Drones4Safety

Research & Innovation Action (RIA)

Inspection Drones for Ensuring Safety in Transport Infrastructures

Specification of the Multi-Drone Swarm System

D5.1

Due date of deliverable: 31.03.2021

Start date of project: June 1st, 2020

Type: Deliverable
WP number: WP5

Responsible institution: Aarhus University
Editor and editor's address: Rune Hylsberg Jacobsen, Aarhus University, Denmark

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 861111

Version 1.0
Release Date: March 31, 2021

Project funded by the European Commission within the Horizon 2020 Programme		
Dissemination Level		
PU	Public	☒
CO	Confidential, only for members of the consortium (including the Commission Services)	☐

Change Log

Rev.	Date	Who	Site	Change
0.1	01/06/2020	Annika Lindberg	SDU	Created initial version
0.2	25/03/2021	Rune Hylsberg Jacobsen	AU	Updated after internal WP5 review.
1.0	30/03/2021	Rune Hylsberg Jacobsen	AU	Final draft version. Updated with comments from review in the consortium. Ready for submission.

Contributing authors

This deliverable has received input from many project members of the Drones4Safety (D4S) consortium. The list of contributing authors follows below sorted alphabetically according to the first name.

Emad Samuel Malki Ebeid, University of Southern Denmark (SDU UAS)

Frederik Falk Nyboe, University of Southern Denmark (SDU UAS)

Lea Matlekovic, University of Southern Denmark (SDU IMADA)

Liping Shi, Aarhus University (AU)

Néstor J. Hernádez Marcano, Aarhus University (AU)

Nicolaj Haarhøj Malle, University of Southern Denmark (SDU UAS)

Rune Hylsberg Jacobsen, Aarhus University (AU)

Sam Münchow, Automotive & Rail Innovation Center GmbH (ARIC)

Contents

Acronyms

Acronym	Description
5G	Fifth-generation mobile
AC	Alternating Current
AI	Artificial Intelligence
AODV	Ad Hoc On-Demand Distance Vector
AP	Access Point
API	Application Programming Interface
BVLOS	Beyond Visual Line of Sight
C2	Command and Control
COTS	Commercial-Off-the-Shelf
CR	Coding Rate
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
D2C	Drone to Cloud
D2D	Drone to Drone
D2G	Drone to Ground
D4S	Drones4Safety
DB	Database
DC	Direct Current
DDS	Distributed Data Service
DNS	Domain Name Service
DP	Drone Pilot
DPA	Drone Pilot Assistant
EGNOS	European Geostationary Navigation Overlay Service
EMI	Electromagnetic Interference
ESC	Electronic Speed Control
ETSI	European Telecommunications Standards Institute
GALILEO	European Global Satellite Navigation System
GCS	Ground Control Station
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System

GPU	Graphics Processor Unit
GRE	Generic Routing Encapsulation
GSD	Ground Sampling Distance
HA	Home Agent
HTTP	HyperText Transfer Protocol
HWMP	Hybrid Wireless Mesh Protocol
IMU	Inertial Measurement Unit
IP	Internet Protocol
ISM	Industrial, Scientific, Medical (frequency band)
JSON	Javascript Object Notation
LoRa	Long Range radio
LoRaWAN	Lora Wide Area Network
LPWAN	Low Power Wireless Area network
LTE	Long Term Evolution
MAC	Medium Access Control
MAVLink	Micro Air Vehicle Link
MES	Mission Execution Supervisor
MLME	MAC Sublayer Management Entity
MPM	Mesh Peering Management
MPO	Mission Planning Operator
NOTAM	Notice to airmen
OBC	On-Board Computer
OLSR	Optimized Link State Routing Protocol
OMG	Object Management Group
OPL	Overhead Power Lines
PSR	Packet Success Rate
QoS	Quality of Service
REST	Representational State Transfer
RFL	Railway Feeder Lines
RGB	Red, Green, Blue
ROS	Robotic Operating System
RPC	Remote Procedure Call
RTPS	Real-Time Publisher-Subscriber protocol
SITL	Software-In-The-Loop
SME	Station Management Entity
SSH	Secure Shell
SysML	System Modeling Language
T&C	Telemetry & Control
TCP	Transmission Control Protocol
ToA	Time-on-Air
UAS	Unmanned Aircraft Systems
UDP	User Datagram Protocol
UE	User Equipment
UML	Unified Modeling Language
URLLC	Ultra-Low Latency and Reliable Communication
VLOS	Visual Line of Sight
VNC	Virtual Network Computing
XML	eXtensible Markup Language

1 Executive Summary

The deliverable describes the design of the multi-drone system and its communications within the Drones4Safety (D4S) infrastructure. The main inputs for this design specification are the requirements and use cases defined in WP2 of the project.

We report on a concept of operation analysis (Section 3) that describes the multi-drone system from an operational point of view identifying the key functions of the system. The description of these functions reflects on the system requirements and use cases and discusses how this impacts the multi-drone design. The autonomous missions are divided into three phases. The *preparation phase* involves key actors of the inspection mission planning and operation, defines the inspection objectives and plan the mission. In the *operation phase*, a swarm of drones carries out the inspection mission undergoing a sequence of charging procedures for longer durability. During this phase, mission progress is monitored from data provided through the Telemetry & Control (T&C) functions. The multi-drone system continuously upload the inspection results to the cloud services during the missions. In the *conclusions phase*, remaining data is gathered and the multi-drone system is either retrieved from the inspection site or remains "resting" until a consecutive mission is initiated.

The report defines the mission concept as a collection of "work tasks", where each task can be assigned to a drone. The decomposition of tasks is an essential step of a collaborative autonomous inspection mission as it allows individual drones to focus on the specific part of the inspection e.g., upper bridge deck, support pillar number 3, etc. During the mission, the multi-drone system interacts with a set of cloud services that support the mission. We introduce the cloud services that are distinct from today's online services used by drone pilots such as Optimal Task Allocation, Global Path Planning, No-fly Zone, Charging Spot, Cloud Storage, and Data Analytics Services. Furthermore, the scope for autonomous functions as cable detection, identification, and grasping, safe landing operation, energy harvesting & power supply, and positioning are defined.

The communication infrastructure is a key foundation for achieving autonomy and collaborative mission design. The D4S system design differentiates between three different types of communication: drone-to-ground, drone-to-drone, and drone-to-cloud infrastructure. This report outlines the result of the analysis of the communication infrastructure needs and defines the appropriate technologies to meet the different needs of the communication. For drone-to-ground communication, the range is a key performance requirement. We choose LoRa radio communication as it in practice provides few kilometers of range with data rates that can support T&C. A wireless mesh network is supporting drone-to-drone communication, which allows drones to share data and execute swarming protocols. To connect to cloud services and the Internet, the D4S communication infrastructure will be capable of using IPv6 for connectionless message exchange.

In Section 4, we specify the multi-drone system and provide a set of structural SysML models to describe the hardware and the software parts of the system. The hardware design revolves around mechanical parts, sensors, and computation systems needed for making an autonomous drone hardware design. The software system is divided into parts related to flight control and autonomy, inspection applications, recharging control, swarm control, as well as software part for simulations. The drone design will rely on open source software components to support flight control such as software from the PX4 community. The majority of software developed for the multi-drone system in the D4S project is based on the ROS firmware platform and the software execution environment. We introduce the platform and described the environment in which software is built on this platform.

Finally, the deliverable introduces the simulation environment used to assist the multi-drone system design. Simulations ease the steps needed to be taken from concept to actual design and support a fast feedback cycle during the development process. The report concludes by outlining the approach to testing and validation of the multi-drone system, which will be a major activity towards the end of WP5.

2 Introduction

The Drones4Safety (D4S) project aims to increase the safety of the European civil transport system by building a cooperative, autonomous, and continuously operating drone system that will be offered to railway and bridge operators to inspect their transportation infrastructure accurately, frequently, and autonomously. A key part of achieving this overarching objective is to provide a cooperative multi-drone system able to perform the inspections by acquiring data from observing the inspection targets with specific purpose sensors such as a high-resolution camera.

Deliverable D5.1 provides a specification of the cooperative multi-drone system design. It reports on the drone swarm system design and functional interfaces needed to support the main inspection use cases of WP2 in the D4S project. It furthermore addresses the connectivity and specifies the D4S communication infrastructure that allows the efficient exchange of information between the multi-drone system and cloud services including mission control.

Intentionally, deliverable D5.1 does not concern network security aspects of multi-drone swarm system as this will be the scope of D5.2 “Multi-drone system threat analysis and specification of the security system design”.

This report is organized as follows. The Introduction section (this section) outlines the scope of the deliverable and provides a linking to system requirements defined in WP2 of the D4S project. Section 3 provides a high-level description of the multi-drone system from an operational perspective inspired by the term concept of operations from systems engineering. The concept of operations describes the characteristics of a proposed system from the viewpoint of an actor who will use that system. Section 4 details the specification of key functional parts of the multi-drone system. As simulations is a central part of the software development process, we provide an introduction to the relevant simulation environments for the multi-drone software development in Section 5. Finally, we introduce the planned approach to validation of the multi-drone system in Section 6.

2.1 Linking to system requirements

The main input documents for this deliverable are the D2.4: “Use-case Document” version 3.5 and the D2.5: “Final System Requirements Document” version 1.0 from WP2 of the Drones4Safety (D4S) project. Deliverable D2.4 provided an analysis of different inspection sites of interest for the D4S project to find specific use-cases for bridge and railway inspection suitable to test and validate the D4S platform including the multi-drone swarm system. D2.5 presents high-level requirements and architecture for the D4S Drone System. Both deliverables contribute to the framing of the multi-drone system design.

To provide a basis for the specification of the multi-drone system we identify essential requirements that will be fully or in part allocated to the multi-drone system. The verbs MUST, SHALL, SHOULD, etc. have the meaning defined in D2.5, MUST or equivalent terms “REQUIRED” or “SHALL” means that the definition is an absolute requirement of the specification. SHOULD or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course. From D5.2 we identify key requirements governing the multi-drone system design (Table 1). Functional, architectural, design, and operational requirements have been considered. Performance requirements are excluded from this analysis.

Table 1: List of system requirements allocated to the multi-drone system.

Requirement tag	Short description	Requirement type
3.2.1.1 D4S_FUN_REQ_0010: BVLOS inspection	The D4S System SHALL ensure the supervised flight inspection of linear grid infrastructures in BVLOS.	Functional

3.2.1.6 D4S_FUN_REQ_0060: Multiple point of view	The D4S System SHALL be able to acquire targets from multiple points of view / angles, ensuring a proper overlap between each pair of images/videos.	Functional
3.2.1.7 D4S_FUN_REQ_0070: Long endurance	The D4S System SHALL be able to ensure long-endurance inspection campaigns of the target infrastructures without direct human intervention.	Functional
3.2.1.10 D4S_FUN_REQ_0100: Functional Safety	The D4S System SHALL respect all the UAS and railway safety regulations applicable to the inspected areas.	Functional
3.2.1.11 D4S_FUN_REQ_0110: Safe State	The D4S System SHALL foresee a safe state into which it automatically goes when a safety breach is detected.	Functional
3.2.1.12 D4S_FUN_REQ_0120: Positioning	The D4S System SHALL be able during a mission to measure and track in real-time the position of each element of the system itself with respect to a given coordination system (such as the longitude, latitude, and altitude with respect to a given reference).	Functional
3.2.1.13 D4S_FUN_REQ_0130: Recharging	The D4S System SHALL be able to detect when it is necessary to perform the recharge, also considering the distance to the nearest point of recharge.	Functional
3.2.1.14 D4S_FUN_REQ_0140: Autonomous recharging	The D4S System SHALL be able to autonomously harvest energy for self-recharge from both high-voltage cables (i.e. >100kV) and railway overhead power line cables (transmission lines/railways power lines4, between 2 and 50kV AC or between 220 and 3,000V DC).	Functional
3.2.1.15 D4S_FUN_REQ_0150: AC/DC	The D4S System SHALL be able to harvest energy either from AC or DC power lines.	Functional
3.2.1.16 D4S_FUN_REQ_0160: Collision avoidance	The D4S System MUST implement collision and object avoidance countermeasures during both flight and recharging.	Functional
3.2.1.17 D4S_FUN_REQ_0170: Communication	The D4S System SHALL implement a communication network to interconnect the different parts of the system itself.	Functional
3.2.1.18 D4S_FUN_REQ_0180: Processing	The D4S System SHALL expose processing capabilities either on-board, at ground-segment, and in the cloud.	Functional
3.2.1.20 D4S_FUN_REQ_0200: Mission Execution	The D4S System SHALL be able to autonomously execute a previously defined flight inspection mission, including take-off, en-route, recharging, and landing.	Functional
3.2.1.22 D4S_FUN_REQ_0220: Mission Replanning	The D4S System SHALL allow the dynamic reallocation of the resources during a mission.	Functional
3.2.1.25 D4S_FUN_REQ_0250: Online information sources ingestion	The D4S System SHOULD be able to integrate online information sources (like weather forecasts) for eventually and dynamically update flight plans.	Functional
3.2.1.26 D4S_FUN_REQ_0260: Drone Flight Planning	The D4S System SHALL provide a possibility to submit, change or remove a drone flight plan (or flight intent) and will respond by approving or rejecting.	Functional
3.2.1.27 D4S_FUN_REQ_0270: Geofencing	The D4S System SHALL receive, combine, store and provide geofencing and geographical limitations information to drones such as maps, map-layers, coordinates of the ground objects and obstacles (elevations), NOTAMs, according to their activation schedules.	Functional
3.2.1.28 D4S_FUN_REQ_0280: Unique Identification	The D4S System SHALL process the drone unique identifier (e.g. the Electronic Identification) received together with the tracking message (position and timestamp) and link them to the D4S system internal unique identifier assigned to a drone.	Functional

3.2.1.34 D4S_FUN_REQ_0340: Emergency	The D4S System SHALL be able to receive and send emergency information and alerts.	Functional
3.2.1.39 D4S_FUN_REQ_0390: Information about recharge possibility	The D4S System SHOULD receive information about the recharging possibilities, like if the planned recharging point is available or not (e.g. overhead line is in operation or out of service).	Functional
3.2.1.40 D4S_FUN_REQ_0400: EMI	The D4S System SHALL compensate for the effects of high electromagnetic interference on the drones and on the drones' sensors, such as compasses.	Functional
3.2.3.1 D4S_TNA_REQ_0550: Long range wireless communication	The D4S System SHALL use long-range wireless communication network techniques.	Technical/Architectural
3.2.3.2 D4S_TNA_REQ_0560: Effective wireless communication	The D4S System SHALL have wireless communications with acceptable goodput/throughput for conveying inspection and telemetry data to mission control.	Technical/Architectural
3.2.3.3 D4S_ TNA_REQ_0570: EGNOS/Galileo GNSS navigation	The D4S System navigation capabilities SHALL be based on EGNOS/Galileo GNSS.	Technical/Architectural
3.2.3.5 D4S_TNA_REQ_0590: Microelectronic interface between harvester and drone	The D4S System SHALL implement a microelectronic interface module between the light-weight harvester and the drone's energy and control systems	Technical/Architectural
3.2.4.1 D4S_DES_DES_0620: Cooperative Drone System	The D4S System SHALL implement a cooperative drone system.	Design
3.2.4.2 D4S_DES_DES_0630: Autonomous Drone System	The D4S System SHALL implement an autonomous drone system.	Design
3.2.4.3 D4S_DES_REQ_0640: Electric arcs protection	The D4S Drone shape has to be designed to protect the drone components from the electric arcs that are generated by the electric field as a result of a high voltage potential between the cables (e.g. 400 kV AC / 25 kV AC) and the drone voltage (e.g. 11.1 V DC / 22.2 V DC).	Design
3.2.4.4 D4S_DES_REQ_0650: Three-tiered design	The D4S System design is envisioned to be a three-tiered design consisting of cloud services-based back-end, the drone swarm, and a network facilitating communication between the former two.	Design
3.2.5.1 D4S_DES_OPE_0690: Continuously Operating Drone System	The D4S System SHALL implement a continuously operating drone system.	Operational

3 Concept of operation and functional requirements

This section analyses the multi-drone system from an operational point of view. The analysis is essential for the understanding of the functional composition of functions of the system as well as providing a basis for a multi-drone system architecture.

3.1 Overall autonomous mission control

A mission can be divided into three phases: The *preparation* phase, the *operation* phase, and the *conclusion* phase. Since inspection missions are reoccurring events, the three phases will be repeated cyclically. The Actors important for these phases were introduced in D2.4 [1].

3.1.1 Preparation phase

The Mission Planning Operator (MPO) is the key actor of the planning phase. For the visual line of sight (VLOS) inspection operations, the Drone Pilot (DP) and the Drone Pilot Assistant (DPA) are key actors. He/she will in the beyond visual line of sight operation (BVLOS) be accompanied by the Mission Execution Supervisor (MES). The mission planning is supported by a set of cloud services such as map information, weather forecast, and route planning that are available as cloud service support (cf. Section 3.10).

An inspection mission defines the objective and the associated data to describe the mission type, the starting and ending points of the mission, a geofence, among others. It is a key outcome of the mission planning phase. A mission can be divided into a set of tasks that can be allocated to the drone swarm in the operation (cf. Section 4.2.4). Before the inspection mission begins the drone swarm is transported to the inspection area and means to supervise the inspection missions. This includes the establishment of a physical Ground Control Station (GCS). Due to current regulation in the drone domain, the GCS is necessary for the DP or the MES to take over control of the drone swarm, e.g. in case of unforeseen events that could jeopardize safety.

3.1.2 Operation phase

The swarm of autonomous drones carries out the inspection mission by executing the tasks of the mission. First, they navigate to the starting point of the mission accompanied by a *global path plan* provided by a cloud service. A task allocation function decides on the allocation of individual mission tasks to drones. The task allocation function can be either centralized, e.g. controlled from the GCS, or fully decentralized to support the continuous swarming operation. For instance, a drone in the swarm may be given the task to inspect the upper deck of a bridge whereas two other drones would be allocated the tasks of inspecting the north and south sides of the bridge, respectively. The drone swarm performs the inspection mission in a continuous way by receiving a new task when an assigned task has been completed. This continues until there are no more tasks defined for the mission.

The *inspection path plan* is planned/calculated by the drone swarm. The inspection path plan provides a finer granularity of waypoints for navigating the inspection, considers inspection camera viewpoints and obstacles in the local environment. During the inspection, the drone will follow the inspection path and record images according to the specification of the inspection task. In this process, robust and accurate navigation is critical for the quality and the safety of the inspection and cannot alone rely on GNSS as GNSS signal interference and obstructions may occur due to the influence of the structure under inspection.

An inspection mission is constrained by the energy (charge) of the drones. Each drone will need to go through several charging cycles during an inspection mission. When operating autonomously, the drone swarm handles charging by following a *charging protocol* that essentially creates high-priority charging tasks to be scheduled by the task scheduler function. The location of charging points may be preloaded to individual drones (static) or fetched from a cloud service (dynamic). The latter case addresses the situation where the charging point has a limit on the number of drones that it can serve at the same time or if the charging point is mobile. The key

output of the inspection phased is the collection of annotated images that are stored and continuously shared with a cloud service storage. Annotation results from the continuous inference provide by AI algorithms running on the inspection drones.

To be able to continuously monitor the progress of an inspection mission, a Telemetry & Control (T&C) function is implemented in the drones. Each drone will report is telemetry data such as position, velocity, a current task assigned, charging level, etc. to the GCS. The GCS provides this information to the mission control on the continuous progress of the inspection mission. The communication to support telemetry needs potentially need to support long-range e.g., several kilometers. However, the amount of data needed for telemetry is small compared to inspection images.

3.1.3 Conclusion phase

When all tasks of the mission are completed, the drone swarm is retrieved and removed from the inspection area or drones may "rest" until a new mission is assigned. All relevant information such as geotagged and annotated inspection images are uploaded to the Cloud Storage Service. Mission information can be retrieved and visualized after the mission ended.

In the conclusion phase, the MPE actor examines the data from the mission and validates that the mission has objectives that have been met. The MPE takes necessary actions to "clean-up" after the mission such as collecting drones and GCS installation and reporting to the owner of the mission.

3.2 Inspection

The business logic of an inspection operation can be modeled as a set SysML/UML activity diagrams. The activity described provides the logic of the software application controlling the inspection mission. A mission is defined by a Mission Specification created by the MPO actor. The mission has a unique identifier and is allocated to a drone swarm. It is possible to associate a geofence polyhedron to a mission that will restrict drones to stay inside drones to only perform inspection inside this polyhedron. The polyhedron is defined by a set of GNSS location coordinates. The mission is associated with one or more tasks that are ordered in a list. Tasks are described by Task Specifications and have unique identifiers. Furthermore, tasks are described by an inspection type e.g., bridge (upper) deck inspection, catenary cable inspection, etc. A task has a start and end location defined by GNSS coordinates. Tasks are associated with a data acquisition specification that describes which sensors to use e.g., RGB camera, frequency of data acquisition, speed of the drone during data acquisition, etc.

3.2.1 Mission initiation model

Before an inspection mission can begin, preparatory steps need to be taken. It assumes that the drones have been transported to an area that is close to the inspection site. It is further assumed that drones are charged.

Figure 1 shows an activity diagram describing the inspection initiation process. When drones are turned on the *Connect to GCS* using Drone-to-Ground (D2G) communication. Subsequently, drones connect to form a wireless mesh network by using Drone-to-Drone (D2D) connections i.e., enabling connectivity for swarming functions. This allows drones to form a swarm and invoke swarming functions. Following this, a mission is assigned to the drone swarm. The mission specification is retrieved from the cloud service by using Drone-to-Cloud (D2C) communication. Alternatively, the mission can be downloaded to the drones before the initiation of the inspection. A Validate step is performed to ensure that all initial settings are in place and that the multi-drone system is ready to begin the inspection mission.

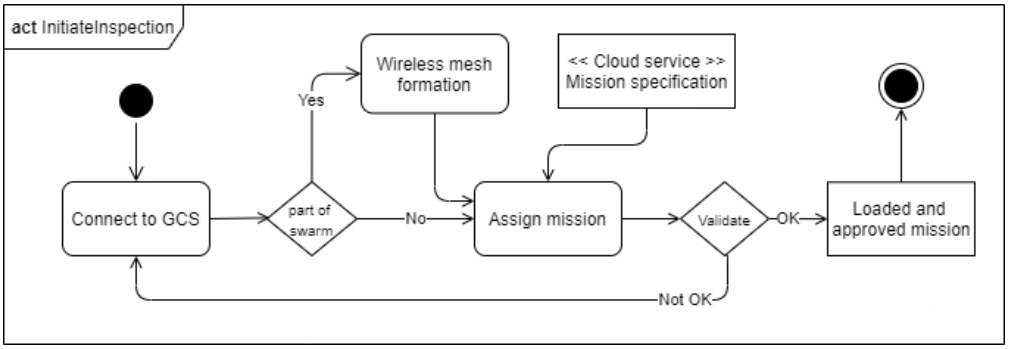

Figure 1: Activity diagram modeling the preparation of the multi-drone system for an inspection mission.

3.2.2 Task inspection model

Figure 2 shows an activity diagram for a task inspection for a single drone of a D4S drone swarm. The process is initiated by the mission operator actor and presumes that the inspection mission has been prepared through the initiate task inspection.

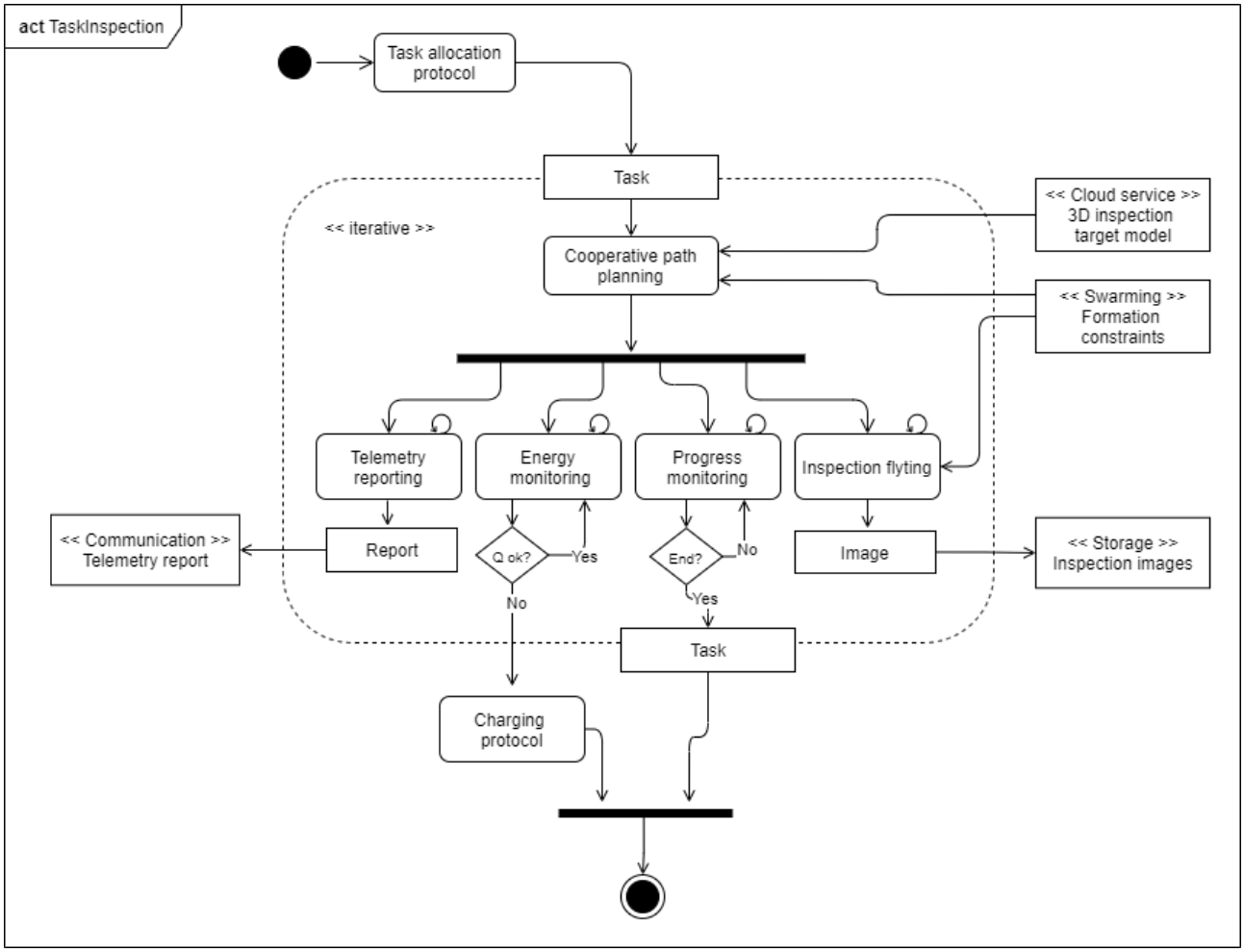

Figure 2: Activity diagram for a task inspection of a drone.

The first action after initialization is for the drone to participate in the *task allocation* activity. Details of the

protocol are subject to further research and are beyond the scope of this deliverable. The output of the process is an allocation of a set of prioritized inspection tasks as symbolized by the Task object. An inspection task could in natural language be formulated as “inspect the upper bridge deck from location A to location B using the high-resolution RGB camera with an image frequency of one image per 10 seconds” or similar. Tasks are performed sequentially according to priority in an iterative way. Note that tasks in this context also include traveling to the starting location of the inspection.

Subsequently, a *cooperative path planning* process is run. The cooperative aspect of the path plan arises from the fact that the path plan of a single drone in the multi-drone system depends on the path plan of the other drones in the system. This ensures that drones can travel together along the same route without colliding and/or they can fly as a swarm in a predefined formation. For the latter part, a specification of the *formation constraints* is used as an input in the cooperative path planning. Part of the cooperative path planning embeds a waypoint-guided path planning that describes the inspection path a drone should undertake to position itself in the right viewpoints for when taking inspection images. This process is leveraged by the use of 3D models of the inspection target (i.e, the digital twin) or a part of the inspection target. For instance, a segment with a supporting pillar and the underneath deck of a bridge may be used by the drone to calculate an inspection path. By using a 3D model as input for the inspection, a good inspection coverage of the construction can be ensured. An example of a bridge model from the use case definitions is shown in Figure 3. The bridge is located near Villalvernia in Italy and described in D2.4. In the example, the part that needs inspection is cropped and captured by a bounding box (middle part of Figure 3) and converted into a voxelized map (right part of Figure 3). The smoothness is represented by the difference in color with blue color representing the more smooth facets.

Figure 3: Example of 3D model of the Villalvernia Bridge, Italy. Left: point cloud information. Center: cropped and cleaned section for inspection. Right: Transformed voxelization of the bridge used for input in the inspection path planning algorithm.

When a cooperative path plan for the drone is calculated, the inspection may begin. The process forks in four critical activities that run in parallel: *telemetry reporting*, *energy monitoring*, *progress monitoring*, and *inspection flying* each running continuously (cf. Figure 2). The telemetry reporting activity ensures that the progress and *telemetry reports* are sent to the GCS. The progress monitoring supervises the progress of the current inspection task (or a prioritized list of tasks). When a task has run-to complete it breaks out of the loop and proceeds to the end-node through the join-node. The battery charge (Q) status of the drone is continuously monitored by the *energy monitoring* activity. If the charge falls below a defined threshold, the drone breaks out of the loop and proceeds with the *charging protocol*. Specific details of the charging protocol are beyond the scope of this specification. The charging protocol will return the drone recharged and ends the current task inspection. The drone can retake inspection work from the start node and go through the task allocation activity again. This decoupling of task allocation and charging is advantageous because the charging time can vary significantly and may be hard to predict. Finally, the *inspection flying* activity is responsible for the drone visiting the waypoints defined in the cooperative path plan and acquiring inspection images according to the specification provided in the task specification. The acquired images are transferred to a storage system from where they can be fetched and sent to the cloud services asynchronously relative to the task inspection.

3.3 Drone to cloud interactions

The exchange of information between the drone swarm system and the cloud services is done over the Internet. A common *D4S data object model* (i.e., the set of relevant data objects cf. Section 4.2.4), ensures the interoperability between applications running in the drone swarm system on air, the GCS, and the cloud infrastructure. The basic interactions between the drone system and the cloud are characterized by being asynchronous with loosely coupled entities such as clients and servers of a client-server programming model. This communication paradigm fits well with the D4S Drones System design and accordingly with standard global Internet connectivity and web services-based client-server interaction models. A mission global path plan, which is determined by a service in the cloud, is communicated to the drones in form of locations to be inspected with tasks that define the inspection itself. Drones report their status back to the cloud by sending their current location, velocity, charging level, etc., along with other relevant telemetry data. During the mission, drones send the telemetry data and inspection images (Result data) to the cloud. Data is stored in the cloud and can be retrieved after the mission. Drones must be monitored according to such telemetry data and be able to receive commands as part of their operations, preventions of hazardous maneuvers, and/or maintenance.

3.4 Communication

Following the D4S system architecture, we divide the communication aspect into three scopes. These three different scopes for communication can also be identified from the analysis of the inspection use cases [4]. First, the system is required to support communication between a drone and the ground infrastructure to support control and command as well as continuous telemetry reporting. Second, the drone swarm needs to form a communication network that allows drones to coordinate and share local information for different tasks coordination activities, e.g., joint path planning, task allocations, etc. Third, the multi-drone system should be able to connect to the cloud services from mission support for image data offloading, and remote telemetry reporting/telecommand. Table 2 provides an overview of the key performance characteristics for the different scopes of communication.

Table 2: Scope of communication between parts in the system architecture.

	Scope of communication		
	Drone to Ground (D2G)	Drone to Drone (D2D)	Drone to Cloud (D2C)
Range	up to 2 km	100-200 m	global connectivity
Throughput	10-100 kbps	0.1-10 Mbps	>10 Mbps
Latency budget	<50 ms for Command and Control (C2)	Determined by ROS QoS profiles	Determined by network latency. Limited by ROS performance
Resiliency	Preferrable dual-channel system.	Low communication layer retransmissions	High communication layer acknowledgments (TCP/IP)
Network types	Point-to-point links	Mesh network	Global IP networking
Applications	1) Telemetry & Control (T&C) 2) Communication relay	Swarm coordination	High-level mission control and reporting- Data exchange

The Telemetry and control (T&C) function allows continuous telemetry reporting from the drones. Also, it provides a way for an operator or supervisor to take over control of the flight of an autonomous operation. Besides, the T&C function is essential during test and development in the D4S project as it provides a way to make functional testing in a controlled way. The essential characteristics of such T&C communication channels include long communication ranges, low latency, and high reliability.

The latency budget specifies the maximum acceptable delay from the time the data is written until the data is inserted in the receiver's application. Such a latency budget must be established for each of the previous communication scopes and where the acceptable threshold for these budgets is defined by the corresponding running applications or technology/network physical limitations.

To enable swarming functions, continuous coordination between drones is needed. The connectivity model is primarily base on one-to-many communication (*1:n*) or group communication (*m:n*) in a highly dynamic setting. The multi-drone system will provide a basis for wireless mesh communication that will ensure a good network coverage with robust communication links for this D2D communication.

To connect the drone swarm will be able to communicate with cloud servers by using IP to cope with the transport of information over heterogeneous networks. Internet communication is a well-tested technology with widespread deployment. It is possible to access the Internet in many places throughout Europe including rural areas.

3.5 Swarming

The swarming solution shall efficiently segment tasks to multiple drones and provide a path plan for multiple drones to follow task-specific high-level guidance in an environment with obstacles. Besides, swarming shall enable multiple drones to fly in configurable formations, which e.g., allows for implicit coordination of the inspection operation. For instance, three drones of a swarm may fly in a formation where two drones examine the two sides of a bridge deck and the third drone the upper part of the deck. The swarming solution reads task data and on-board sensor data as input, then generates trajectory data (waypoints) for the flight controllers of multiple drones. The solution shall also provide member discovery, data sharing, and synchronization between drones. In this regard, the swarming system provides a common information-sharing context where shared information of the drone swarm is handled.

The multi-drone system implements a membership management protocol to ensure drones dynamically joining or leaving the swarm. The swarm membership management mechanism ensures that drones can at any time decide to join the swarm or leave if already joined provided that the drones are within the same subnetwork[1]. Note that the drones of the swarm may not all be mutually in the radio range of each other as the mesh networking may ensure drones to be connected via multi-hop communications. From a network layer point of view, the drone shall implement one or more multicast channels to support the communication within the swarm. Furthermore, it is a prerequisite that each drone continuously keeps track of the members of the swarm i.e., run the membership management protocol. A critical aspect of the swarm membership management mechanism is the authentication and authorization of join attempts. These security aspects will be addressed further in deliverable D5.2.

Cooperative task allocation is required for the multi-drone system to optimize the efficiency of the task operation among multiple drones. The cooperative task allocation should segment the task based on the number and the characteristics of the drones. It monitors the progress of the task during the operation and automatically

[1] This is not stricktly necessary as multicast routing could be implemented, which will allow drones to be part of different subnetworks. However, it simplifies the network configuration.

adjusts the task allocation plan for drones to maximize the use of drones. Cooperative task allocation provides high-level command, the task allocation plan, for the task-specific path planner of the multi-drone system.

3.6 Cable detection, identification, and grasping

Some tasks, such as charging with energy harvesting during inspection operations, rely on information about the position of nearby electrified cables and related infrastructures such as pylons, insulators, and dampeners relative to the drone. To detect and track these objects the drone will be equipped with an optimized sensor package. Sensors in this package are selected based on criteria such as weight, size, power consumption, and price – and of course their ability to detect the relevant objects. During operation, the sensor package will generate data that can be used to estimate positions of nearby infrastructure such as cables and pylons.

The diameters of electrified cables in power infrastructures in Europe vary depending on their purpose and geographical location. To accommodate for this, the proposed solution must be able to detect cables with diameters between 10 mm to 40 mm from various distances. During powerline detection, the drone will be positioned next to or underneath the cables. Close to the cables, an accurate estimate will be required to facilitate precise navigation and avoid undesired physical contact, while longer distances allow for a less accurate estimate. At long ranges (>5 m from cable), 80% detection accuracy is tolerable while at short range (<5 m) detection accuracy should be at least 90%.

Once a cable is detected, identifying different cable parameters (voltage, energized/non-energized, grasping points) is important to determine an optimal landing and recharging cable candidate. A learning algorithm will be trained to make a decision based on information such as knowledge of the power distribution network, results of cable pose estimation, and onboard vision sensors. If the level of *confidence* of the system is higher than the safety threshold, then the drone will proceed with the *cable grasping*. Otherwise, it will go for a safe *landing* operation cf. Section 3.9; assuming that the drone is low on energy since it was persuing a charging task.

Initiating the cable grasping procedure within 2 m below the cable, the drone must verify the wind conditions and the cable pose estimate accuracy. The system must then be able to carry out the cable grasp operation for wind speeds up to 10 m/s and wind gust speeds up to 15 m/s. The drone must be able to predict the cable movements while applying combined trajectory planning and tracking until reaching the cable. When the cable is within reach of the gripper, the drone grasps immediately and enters into the charging mode. The success rate of this operation must be >70% under the stated circumstances. In case of failing to reach the cable, the drone comes back to the latest point of detecting the cable.

3.7 Energy harvesting and power

The energy harvester design is a special development of the D4S project (WP3) that brings an energy harvester system (recharging system) for drones at overhead power lines (OPL) as well as railway feeder lines (RFL) during an autonomous inspection flight of bridges and railways. The main design concept of the energy harvester has been specified in deliverable D3.1 [37]. The deliverable describes the boundary conditions and requirements for the development of the harvester system. The harvester will significantly extend the duration of inspection mission operations by allowing drones to shift between flying and recharging. by contacting the conductors and convert the voltage to the level needed for charging the battery of the drone

The design of the energy harvester subsystem aims to provide a good tradeoff between efficiency in energy conversion and the weight of the harvester, which critically impacts the flight time of individual drones. The goal is to harvest energy fields from AC or DC lines. It should be noted that the recharging solution is different when the drone needs to recharge from an AC (inductive coupling) or a DC source (contacting). At OPL the drone approaches the line and attaches itself to the conductor. According to the inductive principle, the drone

harvests energy by bringing a coil around the conductor. The electromagnetic stray field around the current-carrying conductor induces a voltage in the coil, which is converted by the electronics for charging the battery. From a technical point of view, high-voltage OPLs provide easy access for drones. The drone can approach the power line from underneath and grasp itself to the line resulting in a stable mechanical attachment to the line. This procedure requires the drone to be able to detect the position to the cable, navigate to the cable, and grasp the cable. For the harvesting from DC lines e.g., RFLs, the drone has to connect to the ground potential and the high-voltage line to enable the recharging process. It is planned that the drone will first connect to a ground contact via a cable connection and then connect to the high voltage. The electronics convert the high voltage into the necessary low voltage to charge the battery.

3.8 Positioning

To get an accurate and robust position, every single drone in the swarm will use a multi sensor-based positioning algorithm. The basis for localization is the GNSS-based position, which is determined by a multi-constellation GNSS receiver that receives and processes GPS, GLONASS, and GALILEO signals. The GNSS-based position is stabilized with 3D-IMU-data and visual odometry. The visual odometry will be calculated from a lidar point cloud (or depth camera point cloud) and the sensor data is processed in a Kalman filter.

For better accuracy, the drones will use an extension system for the GNSS position. Within the D4S project, we will evaluate if a satellite-based system like EGNOS or an internet-based system like Skylark will fit best for the multi-drone application.

Additional research will be applied to further increase the positioning accuracy. Two approaches will be pursued to explore whether they can be beneficially applied to the swarm's localization performance:

- *Use Map (digital twin) for localization:* The concept is to use the digital twin also for localization. Particularly, well recognizable, significant parts can be used as anchor points, which can be clearly identified in the Lidar point cloud. Furthermore, if the exact geodetic position of such an anchor point would be known, it could be used as a ground truth reference, too.

- *Use Swarm itself for triangulation:* The drones continuously exchange their determined positions amongst each other. If a drone is in an area of weak satellite reception, i.e., GNSS is not working properly, its position can be determined by triangulation with two other drones. Furthermore, by using this procedure, each drone could validate its calculated position, too.

3.9 Safe landing operations

Safety is of utmost importance for the multi-drone system operation. Drones must be able to conduct emergency procedures that will bring safety to a stop and land on the ground without jeopardizing the safety of activities in the ground e.g., human activities.

The multi-drone system will be working in environments where people and animals are present. The local environment may also be represented by steep slopes and hills or might be crowded with man-made objects. Since the drones will have an inevitable chance that they malfunction or otherwise are required to land, it is important that their landing is non-threatening. It is, therefore, necessary for the drones to have a well-defined procedure for emergencies that could occur initially or during flight. The multi-drone system should implement a protocol for the detection of *safe landing zones*. The protocol should rely on perception based on local sensors such as a depth camera.

A safe landing zone is generally defined as 1) an area without the location of man-made objects and 2) a flat region. The avoidance of man-made objects is included to disallow the drone from landing on buildings, roads, or similar potential hazardous places. It makes sense to include this criterion as the risk of humans being nearby

man-made objects is substantially bigger than at non-man-made objects. This will reduce the risk of damaging any humans in an emergency where the drone would require the use of safe landing zones. In many areas where the reconnaissance protocol will be used for bridge or railway inspections, there will likely be a decent number of locations that conform to the two requirements, as many bridges and railway segments are located distant from cities. Flat regions are included because the drone needs a flat region to land in a controlled manner.

The general approach in the literature is to use stand-alone images to detect safe landing zones for the drones [5][6][7]. Furthermore, the images are captured from directly above the area of interest with an angle of the camera view perpendicular to the ground, which may not be realistic in a real drone flight mission.

Another safety aspect concerns the situation where a motor failure or a failure to the low-level flight control of a drone potentially leading to a drone crash. In such a case, when a drone will lose its maneuverability, a mechanism to mitigate the size of the impact of the drone crashing to the ground or impacting humans, animals, or manmade objects. A possible mitigation mechanism includes implementing an emergency parachute to be deployed automatically from the drone to minimize the impact of the crash.

3.10 Cloud service inspection support

To support the autonomous mission, several cloud services are essential. Today, drone operators already use cloud services such as local weather forecasts and restricted fly-zones. For the D4S use cases, additional cloud services result from an effort in WP4 and WP6. Cloud services are supported with a web interface where the operator can plan, execute and monitor the mission. All data collected through the mission is stored in the cloud and could be retrieved, visualized, or analyzed any time after the mission. Nevertheless, the cloud services needed for the autonomous drone operation are shown in Table 3.

Table 3 Cloud Services for the D4S project.

Service name	Description
Optimal Task Allocation Service	When the inspection targets are set, the service uses algorithms to determine the optimal or suboptimal order of allocating tasks to the drone swarm.
Global Path Planning Service	Global path planner takes into consideration information provided by other services and calculates mission routes for the drones in the swarm. Calculated routes are based on the MPO's entries and visualized on the web interface. Route data is sent to the drones in form of real-world locations.
No-fly Zone Service	The service provides areas where autonomous drone flights cannot be executed. No-fly zones are determined based on static data provided by the authorities. Additionally, official no-fly zones are supplemented with locations of residential areas, highways, and critical infrastructure (power plants, transformation stations, etc.) where it is not safe to execute autonomous flights. Determined areas are used for global path planning.
Weather Forecast Service	Weather forecast service determines and dynamically provides weather data (wind, snow, rain, etc.) to the path planner for a specific flight area. Based on that data, it is decided if the flight should be suspended, rerouted, or continued as planned.
Charging Spot Service	Charging spot service determines areas where drone charging is safe and possible. Spots are taken into consideration while allocating tasks and planning routes.

Cloud Storage Services	Cloud storage consists of databases where data received from the drones is stored. Stored data consists of drones' mission positions, velocities, telemetry, and images collected during the inspection. Data can be visualized anytime using a web interface and can be used for further analysis.
Data Analytics Services	Data analytics services analyze data stored in cloud storage. It includes image analysis and 3D image reconstruction, which give valuable results important for mission success.

4 Multi-drone system design

Based on the above analysis and functional descriptions of the multi-drone system, a more detailed design specification follows below. As a means to specify the system, we provide model descriptions for selected parts of the system using the SysML/UML graphical modeling language.

Our analysis starts by defining the structure of the multi-drone system (Figure 4), which is our *system-of-interest* for this deliverable. Essentially, the multi-drone system can be defined from a hardware and a software part. As the software part have several different constituents we represent it as an abstract block on this level in the block definition diagram (bdd).

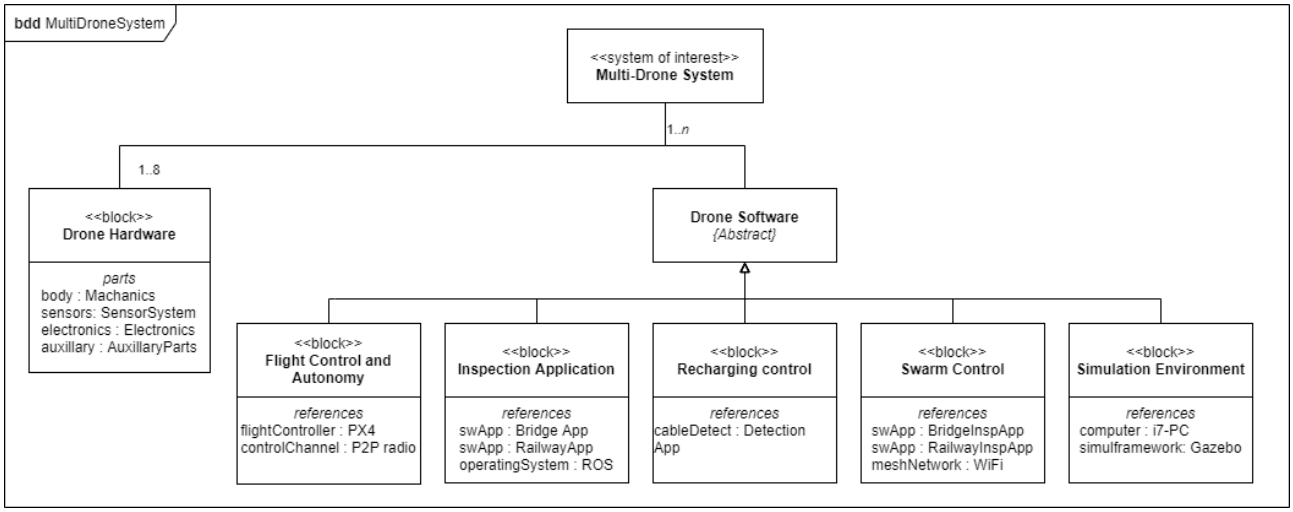

Figure 4: High-level system model of the Multi-Drone System.

The following subsections specified the individual blocks from a functional point of view.

4.1 Drone hardware subsystem

The physical specifications of the drone are contained as the Drone Hardware subsystem (Figure 1Figure 5). This spans several conceptual blocks. The drone Mechanics comprises the structural layout of the drone, the propulsion system, and the Electromagnetic Interference (EMI) shielding for enabling operation in harsh environments. The Sensor System covers sensors utilized for flight control, both low level and high level. The drone Electronics specify the major electronic equipment the drone is carrying for enabling computation and communication. Finally, the Auxiliary Parts cover the additional payload of the drone enabling energy harvesting, inspection, and localization. This part depends on the specifics of the inspection mission.

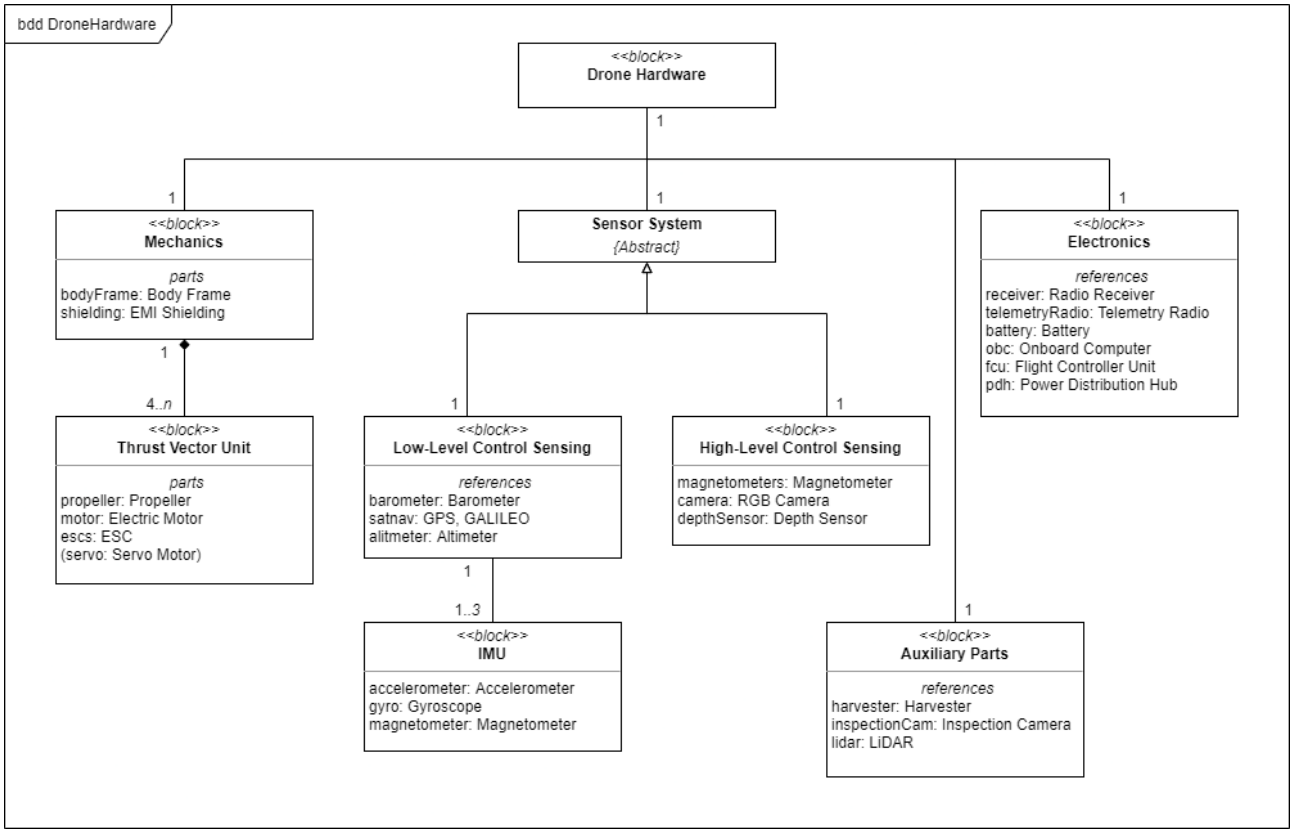

Figure 5: Inspection drone hardware block diagram.

4.1.1 Drone mechanics

The mechanics of the drone hardware subsystem will be designed for optimal wind disturbance rejection. This influences the mechanical layout of the propulsion system and the morphology. The propulsion system will consist of several thrust vector units. These will be positioned on the drone for optimized wind disturbance rejection, which is necessary when flying near infrastructure potentially damaging the drone if hit. The mechanical parts of the drone are summarized in Table 4.

Table 4: List of mechanical parts of the drone hardware subsystems.

Mechanical part	Description
Body Frame	The body frame is the structural part of the drone. This will be designed according to the intended drone morphology.
EMI Shielding	The drone will be protected by EMI shielding from the electrical charge exchanged when touching an overhead power line.
Thrust Vector Unit	A thrust vector unit comprises a propeller, an ESC, an electric motor, and possibly a servo motor. The drone will carry ≥ 4 thrust vector units.
Propeller	The propellers will be scaled according to the final requirements for payload capacity and wind disturbance rejection.

ESC	The Electronic Speed controllers (ESCs) controls the rotational speed of the electric motors.
Electric Motor	The electric motors drive the propellers.

4.1.2 Drone sensor system

The sensors equipped on the drone are divided into sensors necessary for stability and navigation (low-level control sensing) and sensors required for detection of the power line from which the drone charges (high-level control sensing). The sensors applied for low-level control sensing are summarized in Table 5.

Table 5: List of low-level control sensors of the drone hardware subsystem.

Low-level control sensor	Description
IMU	Inertial measurement unit. Comprises of a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer. Delivers positional information for the low-level flight controller. The drone will carry ≥ 1 IMU.
Accelerometer	The accelerometer in an IMU measures the translational acceleration in each direction. It also supplies information about the direction of gravity.
Gyroscope	The gyroscope in an IMU measures the rotational velocity around each axis.
Magnetometer	The magnetometer in an IMU measures the direction of the magnetic north.
Barometer	A barometer measures the air pressure intended for deriving the altitude of the drone. The drone will possibly carry a barometer.
GNSS Module	The GNSS module (i.e. Global Navigation Satellite System using GPS and GALILEO signals) supplies global position information. The drone will carry ≥ 1 GNSS module.

The sensors deployed for high-level control of the drone during cable grasping operations are not yet specified, as the choice of these constitutes a design choice and a research objective. However, the sensors will be chosen by a requirement for the combined weight (< 250 g), a requirement for the combined price (< $1,000), a requirement for combined power consumption (< 15 W), and a requirement for combined sensor fusion output rate (> 100 Hz for distance < 2 m). Some relevant sensor technologies are summarized in Table 6. Furthermore, it is intended to apply Kalman filter sensor fusion and hardware acceleration.

Table 6: List of high-level control sensors of the drone hardware subsystem.

High-level control sensor	Description
Magnetometers	Using magnetometers to measure the magnetic field emitted by the power lines, the position of the power line can be derived geometrically.
RGB Camera	An RGB camera can be applied for the detection of the power lines.
Depth Sensors	Several possible depth sensors exist, that can supply three-dimensional point cloud representations of the environment.

4.1.3 Drone electronics

The additional electronics of the drone include the necessary electronics for flight and communication as well as computation boards for processing. Table 7 lists the general electronic parts equipped on the drone.

Table 7: List of drone electronics systems.

Electronic part	Description
Radio Receiver	The radio receiver receives signals for manual flight.
Telemetry Radio Module	The telemetry radio module covers radio communication with the ground control station.
Battery	The battery powers the drone.
Power Distribution Hub	Circuitry for distributing the power to the drone's electronic equipment.

The intended computation boards equipped on the drone is summarized in Table 8

Table 8; Computation systems of the drone hardware.

Computation part	Description
Onboard Computer Unit	General onboard computer. Used for all computation and applications not comprising the low-level flight controller.
Flight Controller Unit	Computation board running the low-level flight controller.

Raspberry Pi4 B [17]:

Intel NUC7i3DNBE [18]:

NVIDIA Jetson Xavier NX [19]:

DJI Manifold 2 [41]:

Figure 6: Example of OBC hardware platforms.

It should be noted that there is a fast-paced development of computer systems (and sensors) in the marketplace for drone components. Consequently, the drone hardware platform needs to be flexible to cope with different computation devices. Examples of possible target platforms are shown in Figure 6. The different platform represents different trade-offs between processing capability, weight and power consumption. Some platforms like the NVIDIA Jetson Xavier provides GPU support applicable to the acceleration of machine learning interference applications. This final choice of platform depends on the algorithm development of the WP4 in the D4S project.

4.1.4 Auxiliary parts

Auxiliary parts cover the drone's additional payload necessary for the inspection, harvesting, and navigation applications. The auxiliary drone parts are summarized in Table 9.

Table 9: List of possible auxiliary parts of the drone hardware subsystem.

Axiliary part	Description
Energy Harvester	Equipment for harvesting energy from overhead power lines.
Inspection Sensors	Sensors used specifically for the inspection application.
Environmental Sensors	These sensors can sense the local environment e.g., LIDAR sensor, stereo/depth-sensing camera, optical flow camera, etc.

For a multi-rotor system, there is a wide range of choices for inspection cameras. From a technical point of view, it mainly depends on the payload-carrying capacity of the drone and its electrical performance. A good candidate for an inspection camera requires a minimum resolution of 20 megapixels to get a good Ground Sampling Distance (GSD). The inspection camera will primarily be used to take still pictures to keep the need for data storage reasonable. Other inspection sensors as thermal cameras and radiometric sensors for UV irradiance measurements. The exact equipment for the drone with inspection depends on the target for and the type of inspection as well as the cost limits for the drone design.

4.2 Drone software subsystem

This section introduces the basic software execution environment for the on-board computer (OBC). It describes the inspection application (high-level control) from a software point of view. It describes the swarming functions (extensions for multi-drone operation. It introduces data objects essential for interoperability with the cloud services.

4.2.1 Software platform support

Figure 7 shows the software architecture stack running on the high-level controller platform of the drone system executed on the OBC hardware module.

The software architecture is built and executed on the OBC hardware. It is based on the Linux operating system patched with specific drivers needed to support peripheral sensors and actuators of the drone. To ease portability and to provide a good foundation for multi-site software development the architecture supports containerization technology using Docker [16]. The docker container hosts a ROS/ROS2 execution environment for user space applications builds on the ROS middleware APIs. To support these applications a relevant Python version, as well as C/C++ libraries, are provided as part of the software architecture. Additional assisting programs that do not require ROS are supported through standard Linux APIs.

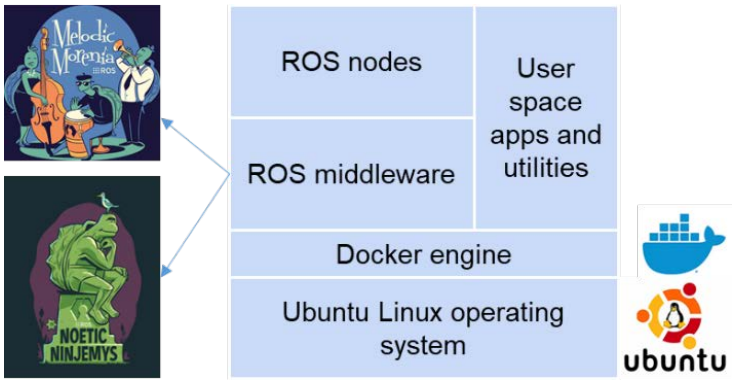

Figure 7: Software architecture stack of the drone system.

The Robot Operating System (ROS) creates the foundation for the autonomous inspection application. ROS also comes with basic link-layer support for the communication between ROS nodes from the MAVROS [8] and MAVlink [9] support packets. Together, these software components constitute the basic ROS communication infrastructure. The second version of ROS i.e., ROS2, implements the message-oriented middleware support for communication through the Distributed Data Service (DDS) standard provided by the Object Management Group (OMG). ROS is an open-source software framework used for managing robot systems [10]. ROS can be considered a meta-operating system for robots, because it provides services similar to an operating system such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message passing, and even package management. The main feature of ROS is the reuse of code in development, as ROS is a distributed framework of processes that interact with each other through the publisher/subscriber communication pattern. This pattern makes it possible for the different nodes to publish messages with specific tags called topics, which can subsequently be received by subscribers that have subscribed to a particular topic. When a publishing node publishes a message, a subscriber node will request a connection over an agreed connection protocol to receive the message. For the nodes to establish this connection, a master node also provides lookup information about the nodes. The limitation of the message pattern is that it is mainly for group communication, and as such, it has no reply concept implemented. Still, ROS overcomes this by introducing services, which operate with a request/reply message pattern. This is mainly used for one-to-one communication between nodes.

The way ROS is structured enables individual development of the different components and decouples it greatly, thereby making it possible to focus on one part of the system without affecting another part. Such components are usually wrapped in a ROS package, which can be compared to a standard program library. ROS packages can also launch a node, which publishes/subscribes to standard topics. Another benefit of ROS is that it is multilingual, which means that processes or nodes can communicate regardless of programming language assuming it is supported by ROS. For the reasons stated above, it scales remarkably well, when the number of nodes or robots increases as the processes are independent in the system.

With the distributed nature of ROS, it will be simple to integrate several different components into a multi-drone system, while also making it easier to focus on more crucial components without changing the other components too much. This makes it an ideal framework to adopt within the project, as the project also has a lot of different components that need to be integrated into one fully-functional system distributed among identical drones. It is also open-source, which means there are many communities developing ROS packages available for use.

MAVROS is a ROS package that enables MAVlink messages into the domain of ROS. It acts as a bridge between ROS and MAVlink protocol. With this node set up, it will now be possible to communicate with the PX4 autopilot and in extension, the drone it is installed on. MAVlink is a lightweight communication protocol

for communication with drones and the components of a drone [9][11]. MAVlink uses a hybrid publish-subscribe and point-to-point pattern, where data streams are sent/published as topics while configurations are point-to-point with retransmission. The key features of MAVlink are that it is designed to be efficient and reliable, with only 8 or 14 bytes overhead per packet depending on the version of MAVlink that is used, so it is also very suited for applications with limited bandwidth and has methods which can detect packet loss, packet corruption, and packet authentication. It also enables offboard and onboard communications while scaling up to 255 concurrent systems in the network, which can be the vehicles, ground stations, and so on.

A major development is currently being undertaken in the robotics software community to develop ROS2 as the preferred software framework for future robotics applications. We, therefore, estimate ROS2 to be the final base for the multi-drone system of the D4S project. However, no all-needed software packages might be upgraded during the project and ROS nodes from ROS and ROS2 should be able to coexist in the multi-drone system.

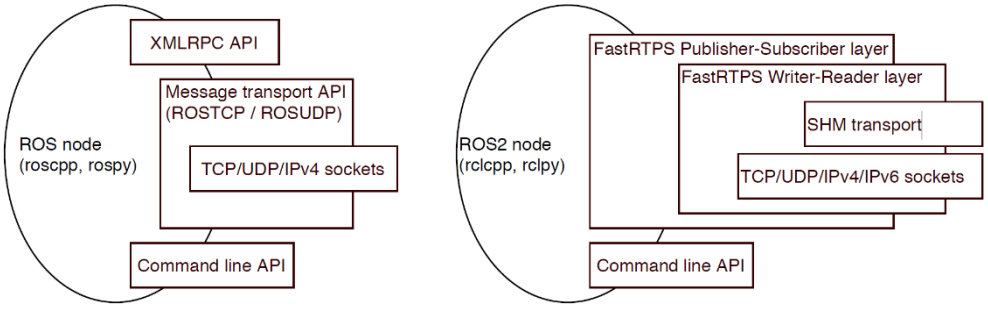

Figure 8: ROS vs. ROS2 APIs.

Figure 8 illustrates the APIs for ROS communication following the two versions of the software framework. It shows the data format (i.e., ROS messages) as well as related transport protocols. ROS uses ROS messages for communication between ROS nodes based on a publish-and-subscribe mechanism. The mechanism is implemented via XMLRPC for naming and registration services, which is a Remote Procedure Call (RPC) protocol that uses XML to encode its calls and HTTP as a transport mechanism. However, the use of RPC introduces a risk of limited availability due to its blocking nature in contrast to the more loosely coupled publish-subscribe model that does not require simultaneous availability of subsystems. ROS nodes can be registered as publishers, subscribers, and service providers. The mechanism uses ROSTCP and ROSUDP for transportation, which is based on standard TCP/IP or UDP/IP sockets. There are three standard APIs for a ROS node, including XMLRPC API, ROS message transport protocol implementation (ROSTCP and ROSUDP), and command line API.

4.2.2 Application software for inspections

As shown in Figure 9, there are three types of networks related with the ROS, i.e., single master ROS network [12], multi-master ROS network [13], and ROS2 network [14]. In some applications, different ROS networks are deployed in combination [15]. A ROS master provides naming and registration services to the rest of the nodes in the ROS. The role of the master is to enable individual ROS nodes to locate one another. Once they are located by each other, they will communicate peer-to-peer.

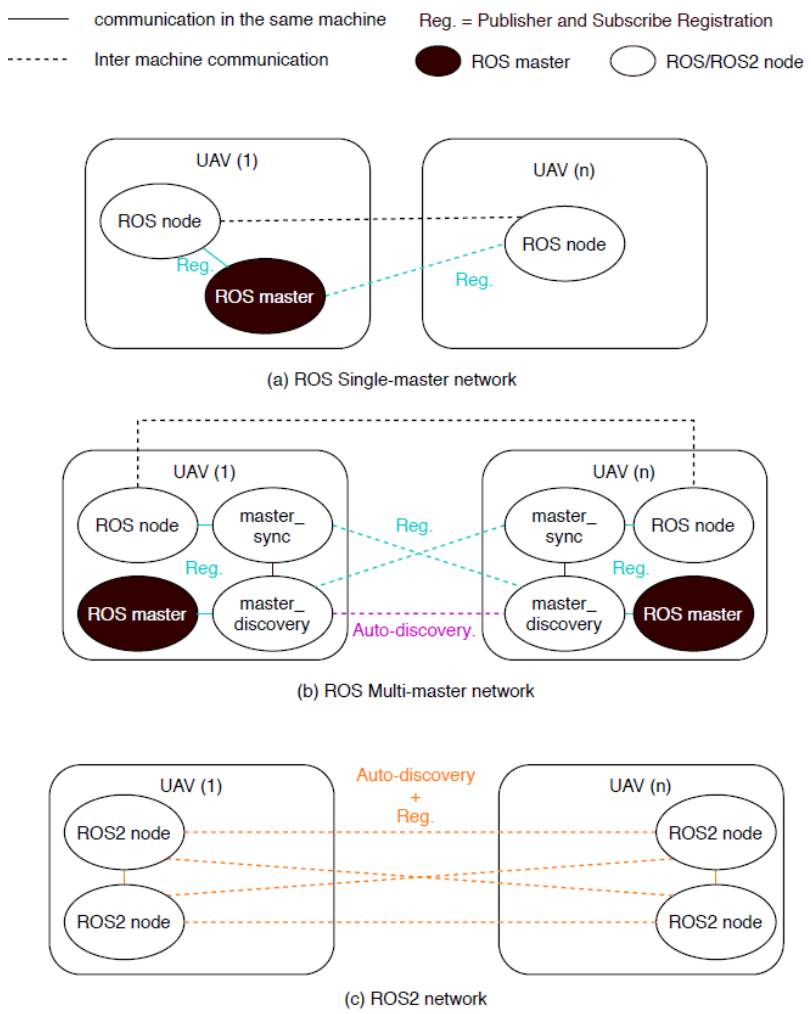

Figure 9: Illustration of the internal communication in the three types of ROS networks: (a) ROS Single Master, (b) ROS Multi-master, and (c) ROS2 network.

In a single master ROS network, only one ROS master is running on a host. ROS nodes for different functions are running on the localhost and remote hosts. Wireless or wired communication is required for communication to remote ROS nodes. A single master ROS network uses centralized connectivity management. Therefore, the complete line of communication is prone to failures if the master fails. The Multi-master ROS network provides an extension for a single master ROS network to spread the load of the centralized ROS master. It involves multiple ROS masters for connectivity management in the network, e.g., each host has its own ROS master (Figure 9). A ROS2 network is different from a ROS network. The ROS master no longer exists in a ROS2 network. A ROS2 network is implemented by using Data Distribution Service (DDS)/Real-Time Publisher-Subscriber protocol (RTPS) as its middleware, which provides discovery, message definition, message serialization, and publish-subscribe transportation. ROS2 provides a ROS type of interface on top of DDS, which hides most of the complexity of DDS for ROS users. Access to DDS-specific API is provided separately in the case of the need to integrate with the existing DDS system, e.g., eProsima FastRTPS, OpenSpliceDDS, and CycloneDDS [14]. FastRTPS is the default middleware implementation in ROS2.

4.2.3 Swarming software function

The swarming software function enables multi-drone collaboration for different tasks. It consists of task allocation, cooperative path planning, and multi-drone communication. According to different characteristics

of the inspection, two types of task allocations will be developed for the two different applications: railway and bridge inspection respectively. The railway task allocation requires railway line detection as input. The bridge task allocation requires 3D map data of the inspection target as input.

The multi-drone communication module supports D2D communication over the wireless mesh network (logical IP subnetwork) while providing an interface for ROS messages. ROS includes the discovery function, i.e., one drone finds the communication addresses of the other drones, and the synchronization function, i.e., share and update information between drones. The prerequisite for multi-drone communication is a WiFi network.

The cooperative path planning module implements three functions including a waypoint guided path plan, a collision-free path plan, and a formation fly path plan. The waypoint guided path plan generates the trajectory data for the flight controller to pass the provided high-level waypoints, e.g., inspection waypoints. The collision-free path plan generates the trajectory (data) for the flight controller to react to encounter obstacles. The planner requires depth estimation data of the obstacles, i.e., the distance to the obstacles. The formation fly path planning generates the trajectory data for multiple drones to keep fling according to a predefined formation.

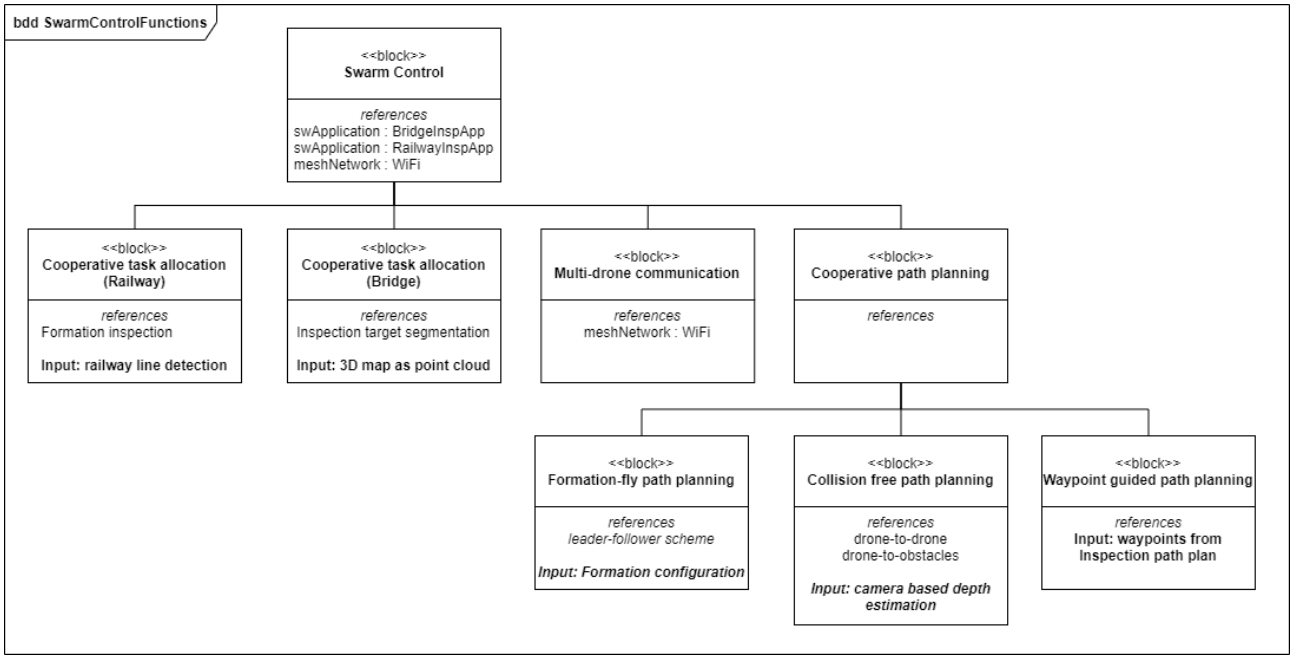

Figure 10: Block diagram for the swarm control functions of the Multi-drone system.

The swarming functions provide general path planning services for a team of drones. The algorithm adopts a hierarchical structure. At the bottom layer, each drone uses model-based optimization to find an efficient path while avoiding obstacles. The iterative best response method with the formation models is adopted at the second layer for multi-drone coordination and formation-fly. At the top layer, the waypoint-guided path planning for multiple drones is integrated to provide an abstract layer and to leave interfaces to task-specific path planers from other modules.

4.2.4 Data objects

To achieve data transfer, it is important to specify data objects and the type of information they are going to hold. Data needs to be transferred between services in the cloud, to the drone, and from the drone to the cloud.

Data objects specify Mission data (including inspection tasks), Telemetry data, and inspection Results data. Figure 11, Figure 12, and Figure 13 model these data objects, respectively. The data object models are extensible and the presentation below is to be considered as a first version of the common D4S data object model.

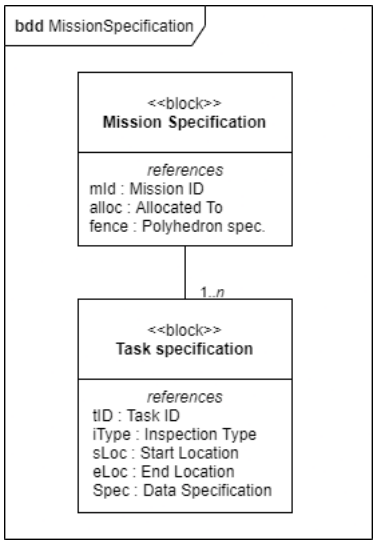

Figure 11: Model of the Mission Specification data object.

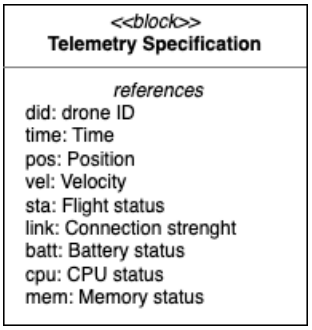

Figure 12: Model of a Telemetry Report data object

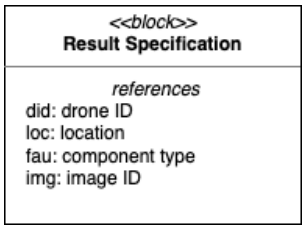

Figure 13: Model of an inspection result data object.

Central elements of the common D4S data object model are described in the following:

- *Mission specification:* A mission is defined by a Mission Specification created by the MPO actor. A mission has a unique identifier and is allocated to a drone swarm. It is possible to associate a geofence polyhedron with a mission that will restrict drones to stay inside the specified zone to only perform inspection inside this polyhedron. The polyhedron is defined by a set of GNSS location coordinates. The mission is associated with one or more tasks that are ordered in a list.
- *Task specification:* Tasks are described by Task Specifications and have unique identifiers. Furthermore, tasks are described by an inspection type e.g., bridge (upper) deck inspection, catenary cable inspection, etc. A task has a start and an end location defined by GNSS coordinates. Tasks are associated with data acquisition specification that describes which sensors to use e.g., RGB camera, frequency of data acquisition, speed of the drone during data acquisition, etc. Tasks are delegated to the specific drone and each task holds a drone identifier.
- *Telemetry Specification:* Drones send telemetry data to the cloud to visualize their location on the map and to monitor mission progression. Each drone will send its unique identifier, position in GNSS coordinates, velocity, current time, battery status, and flight status. The drones will also send information about connection quality to the cloud and with other drones. Telemetry data is described by objects in Telemetry Specification.
- *Inspection Result Specification:* When a fault is detected, drones will send data describing the fault, its location, current time, and image identifier. Image identifier will uniquely describe an image of detected fault, which will be sent to the cloud when the connection is strong.

Data objects are formatted using XML [38] or JSON [39] description languages. Both data formatting languages allow for human-readable and machine-readable data, which will ease the design and possible future troubleshooting procedures. The formats are independent of the programming languages of the application software in use and provide an efficient support for the exchange of data between distributed system components. Also, XML and JSON support schemas to ease data validation. Both formating languages are extensible and support future extensions of data models.

4.2.5 Software configuration management and build support

The ability to build a large complex software system as the one running in the multi-drone system needs to be under tight control. This section presents considerations for software configuration and build-management for the drone software subsystem.

The building of ROS packages and libraries uses the *colcon* build tool [44]. colcon replaces the former software *ament* build tools, which were designed as a support for *catkin* users to handle the migration from ROS version 1 (ROS-1) to ROS version 2 (ROS-2). ament is an evolution of catkin that address the growing need for side-by-side installable ROS-1 and ROS-2 packages, which has become a problem due to different targeted Python versions. To properly build the software in the Docker engine (cf. Figure 7), it is required to compile the ROS libraries as packages in *colcon*. colcon is a Python build system that utilizes CMake as the main backend, which ensures that the ROS middleware is properly linked to generating the target binaries according to the specific Docker container in which these are deployed. To do so, colcon is defined under a workspace directory that allows for the specific ROS packages to be deployed. In this way, the workspace is ready for the inclusion of new ROS packages. Once the colcon packages are progressively created, the whole development can be tested with the `colcon build` command.

To collaborate on the ROS development aspects of the project, a set of Git software repositories have been defined as a common ground for single workspaces. Two main repositories are maintained. First, the platform repository contains all necessary operating system, driver, system, and core Linux software to be able to deploy the Docker engine of a generic Linux distribution. Second, the OBC repository contains all the ROS developments within a common stable version of the latest available drone system functionalities that rely on ROS nodes. In this sense, the branching model of Git allows for collaboration across the project ROS developers of the different drone subsystems. A master branch will contain the mentioned stable version, with possible tags per feature, whereas a development branch contains the current work in progress of specific features. Hotfixes can be done as required. The coding style of the OBC repository should follow the ROS guidelines for functionality and API programming recommendations of the ROS community. Developed code should go through merge requests, where a configuration manager and other developers verify the functionality of the code and that its compilation does not break pre-existing code in the repository. In the long term, documentation and unit testing should be included in the repository to make guarantee the functionalities of the code and provide descriptions for APIs.

4.3 Communications subsystem

The autonomous drone inspection system offered by the D4S project offers a communication solution where connectivity and service operabilities are provided end-to-end. Thereby, we technically support the entire innovation value chain from drone manufacturer, system integrator, inspection operator, data infrastructure provides as well as third party information providers. To enable this cluster of innovation globally connectivity is a necessity. Our design choice is Internet technology based on IPv6 [42]. By using the new Internet standard, we circumvent the shortage of IPv4 addresses and provide a simpler and futureproof network architecture. Furthermore, IPv6 provides a set of useful network services such as neighbor discovery [43] and multicast listener discovery [44] that ease the implementation of the drone swarm network.

In the following, we will describe the overall network architecture. We argue that the subnetworks that are essential parts of the D4S network infrastructure can be classified into three categories:

- **Drone-to-ground communication** is the network technology to support the communication between individual drones and the ground infrastructure.
- **Drone-to-drone communication** support system of drones to form and enables swarming functions to be provided.
- **Drone-to-cloud communication** enables the exchange of information between information serves typically deployed in data centers with an ample amount of computing and storage resources.

These subnetworks classes will be discussed following the introduction of the overarching network architecture.

4.3.1 Network architecture

The target network architecture assumes connectivity using IP version 6 (IPv6). Drones will in this regard be considered as IP nodes that offer IP host and IP forwarding capabilities. The use of IP at the network layer allows the system to operate over heterogeneous networks such as the global Internet. This choice of architecture is somewhat challenged in regards to:

1) LPWAN technologies such as LoRa that have limited support of IP for flexible network topologies, given their focus on physical and data link layers mostly;
2) Access to IPv6 Internet services may in some areas be limited; and
3) Perimeter protection of partner network sites enforces the use of firewalls that restricts incoming and outgoing traffic in different ways being more restrictive towards incoming traffic.

The targeted network architecture can cope with these challenges (Figure 14) by introducing a public data space i.e., the Drones4Energy data broker, and by relying on Domain Name Service (DNS) for mapping of unique global domain names to IPv6 addresses.

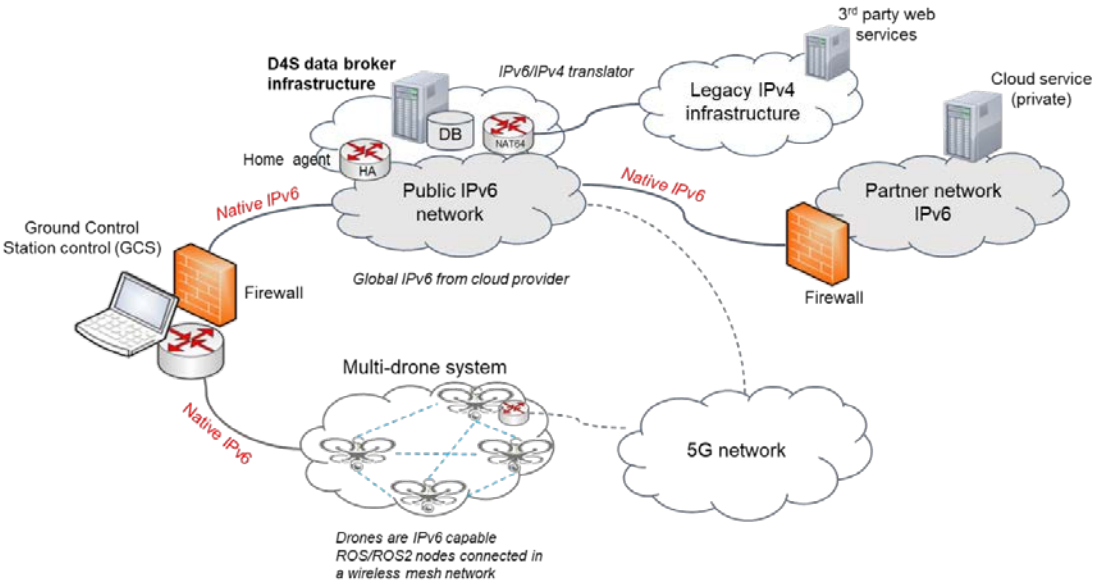

Figure 14: D4S network architecture.

The *D4S data broker infrastructure* is a network entity residing in a public cloud and is part of the Ground Infrastructure cf. D2.5. It offers a platform to exchange data between the drone system e.g., telemetry data and data from cloud services such as mission planning data in the form of scheduled telecommand. It is also the home network for the drone swarm. This means in practice that the IP addresses of the drones belong to this D4S data broker network and traffic sent to the drones will be routed to the data broker network in the global internet. A key component of the D4S data broker infrastructure is the database storage that holds mission-related data. By using this construction, we can circumvent firewalls as requests from cloud services that can be initiated from the partner network hosting the service. Likewise, drones and GCS can push their data such as telemetry data or inspection images to the database. This significantly simplifies the protocol setup in the network architecture.

From a networking point of view, the multi-drone system (i.e., the drone swarm) is a logical IPv6 subnet. When a GCS is present, this node becomes part of the subnet. The GCS can be regarded as a border router connecting the drone swarm to the global IPv6 network. Drones should also have the capability to be configured as border routers in the case where the underlying network technology to connect to the IPv6 Internet is different from the WiFi mesh e.g., using LoRa to connect between the remote GCS and the drones or providing Internet connectivity through a 5G (or LTE) radio access network. It is further proposed to provide a DNS service as part of the D4S data broker infrastructure so individual drones can be addressed by fully verbose qualified domain names. A name example for a drone could be `drone1.operator.d4s.eu` indicating this is drone number one belonging to the inspection operator named `operator` support by the service domain `d4s.eu`.

The Internet addressing is organized hierarchically with the assumption that the target node is located on the network segment defined through the IP address. Because a mobile node, such as D4S inspection drones, will often be located outside the home network, the rules for communicating with the node become more complicated. To maintain connectivity, the node must have a constant IP address which means that a roaming device cannot simply use an address assigned by the nearest network attachment point. Because the problem relates to Internet addressing, it cannot be solved strictly at the Network Access layer and requires an extension to the Internet layer IP protocol. For this purpose, the Mobile IPv6 extension is described in RFC 6775 [21]. The Mobile IPv6 standard solves the addressing problem by associating a care-of address of the mobile node. The node retains a permanent address for the home network. A specialized router known as the Home Agent (HA), located in the home network, maintains a table that binds the nodes' current location to their permanent addresses. When a given node enters a new network, it is provided with an IPv6 address from this network through a stateless or stateful autoconfiguration mechanism. It registers this current location and its new address with the HA. The HA then updates the mobility-binding table with the current location of the node. When an IPv6 packet addressed to the node arrives on the home network, the packet is encapsulated in another IPv6 packet destined to the foreign network where it is delivered to the node. To circumvent triangular routing for traffic that flows between the node and its corresponding node, a binding update can be sent to the corresponding node to inform about its care-of address. This process is known as route optimization. In case the corresponding node does not support route optimization traffic, it will be tunneled from the drone (reverse tunnel) back to the HA and then the routers normally through the IPv6 network [21].

Although the global launch of IPv6 took place in 2012, the majority of sites on today's Internet are only accessible by using IPv4. To be able to access services on the IPv4 network, an IPv4/IPv6 translation service is provided as part of the D4S data broker infrastructure. The translation services are offered as NAT64 gateway support with DNS64 for translating between A and AAAA DNS records. The NAT64 gateway is a translator between IPv4 and IPv6 protocols, for which (to properly function) it needs at least one IPv4 address and an IPv6 network segment comprising a 32-bit address space [22]. An IPv6 client embeds the IPv4 address using a "well-known prefix" when it wishes to communicate with using the host part of the IPv6 network segment, resulting in IPv4-embedded IPv6 addresses and sends packets to the resulting address. Note the

translation is not symmetric and for a stateless NAT64, the IPv4 host cannot initiate communication to nodes in the D4S network infrastructure. However, stateful NAT64 can support IPv4-initiated communications to a subset of the IPv6 hosts through statically configured bindings in the stateful NAT64 [22]. This enables the D4S infrastructure to provide services to IPv4-only third parties. The Linux implementation Tayga supports stateless NAT64. To provide a stateful NAT64 service the recommendation by the Tayga developer is to route TAYGA's IPv4 path through a stateful NAT44, which can be implemented by using IP tables [24].

As for another connectivity interface, the D4S project will closely monitor the rollout of 5G services in Europe. It is expected that 5G services will be offered in urban areas first and that the coverage in rural areas, where we find the majority of our inspection sites i.e., bridges and railway lines, will remain modest in the foreseeable future. Nevertheless, 5G provides two very interesting innovations for the D4S system. First, the promises of providing an ultra-reliable low latency communication (URLLC) service have the potential to offer a command and control (C2) channel for the drones. Second, the support for device-to-device communication in 5G enables a potential substitution of the WiFi mesh network. However, looking into the deployment plans of European mobile operators, these services seem not to be offered commercially in the first wave of 5G deployments. More likely we will be able to use 5G as a broadband access network allowing inspection images to be uploaded to the D4S data broker infrastructure efficiently. This can be achieved by adding a 5G user equipment (UE) modem to individual drones and subscribe to 5G data services from a mobile operator that has sufficient mobile network coverage at inspection sites. Taking into account the D4S network architecture a simplified protocol stack for a drone in the multi-drone system can be sketched (Figure 15).

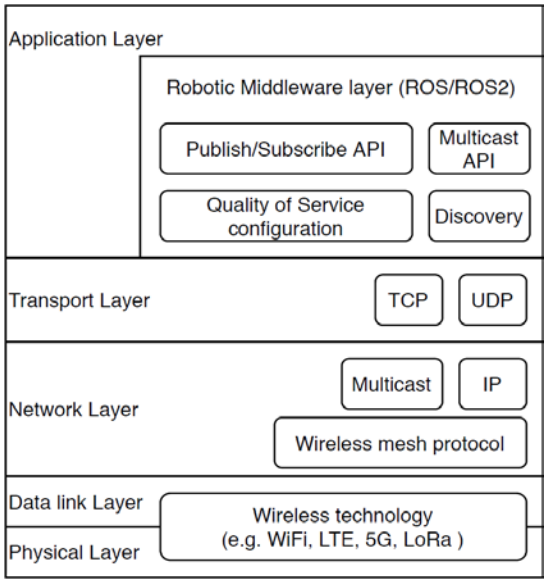

Figure 15: Multi-drone protocol stack.

The multi-drone communication system is formed with multiple layers with different protocols for achieving different functions. A protocol stack for the multi-drone system is presented in Figure 15 to specify potential standard protocols considering the proposed application. It shows a five-layer structure, application layer, transport layer, network layer, data link layer, and physical layer. For the drone system, most of the communication is handled by the robotic middleware layer, which is part of the application layer. It provides APIs for discovery, publish/subscribe mechanisms, and to configure the communication Quality of Service (QoS) and the multicast service. Considering the compatibility to robotic middleware, TCP and UDP are addressed in the transport layer for either reliable ordered connections or unreliable connections, supported by layers below. Also, IP and multicast functionalities are addressed in the network layer. Multicast is involved

to support node discovery mechanism in the network as well as general multicast data transmission. To maintain reliable and flexible network connectivity, a mesh network protocol is considered in the network layer. In the data link and physical layer, different wireless technologies, such as WiFi, LTE, 5G, and LoRa, are in the investigation scope.

The stack provides a data plane view only. Control protocols include protocols for dynamic routing, membership management, Internet error and control, neighborhood discovery, and address resolutions. The selection of these control protocols is part of the research in the D4S project and it is closely tied with the specific use cases of the project. Furthermore, the protocol stack in Figure 15 does not show the special case of using MAVLink communication over a point-to-point communication between a drone and the ground station. This is the special case of D2G communication described in Section 4.3.3.

4.3.2 Drone-to-drone communication

The D2D communication interface considers a wireless mesh setup with the WiFi standard IEEE 802.11s. This variant departs from the traditional WiFi mode based on a centralized Access Point (AP) to allow a wireless mesh network composed of various ad hoc nodes in a decentralized manner. This permits interconnection of drones in a swarm together. The maximum distance between drones in the multi-drone system is limited by the range supported by the WiFi standard amendments. We expect this range to extend up to approximately 200 meters in the rural areas where most of our inspection work will take place.

A mesh network is a communication network topology in which infrastructure nodes connect directly, dynamically, and often non-hierarchically to as many other nodes as possible for multi-hop packet forwarding. Various interconnected nodes allow establishing paths to efficiently route data across them and messages pass through intermediate nodes from any given source to a specific destination in multiple hops. The benefits of mesh networks include a rather rapid installation and low maintenance costs. Besides, mesh networks can add robustness and eliminate a single point of failure due to their decentralized network architecture. However, wireless mesh networks are prone to link failures due to interference, mobility and may fall short to meet data rate demands.

A few studies of wireless mesh networks for drones have been reported in the literature [25][26][31]. In [25], the authors evaluated a framework for an adaptive and mobile wireless mesh network using small drones. The network was based on IEEE 802.11s to provide a WiFi mesh network. The IEEE 802.11s amendment has been adopted in the IEEE 802.11 standard since 2012 [27]. It brings important methods for bridging the path selection to the Medium Access Control (MAC) layer and making PHY data easily available for routing optimization. The work demonstrated how each of the mesh nodes acted as a wireless access point and offered access to 802.11g network services. This allowed extended communication range between end-points through the mesh infrastructure, by using drones as air relay nodes in the mesh [25]. In [56], the authors provided an experimental study of two-hop communication using 802.11s mesh system. The study reports on network connectivity range up to 500 m with 12 Mbps. However, they also observed that the “mesh extension 802.11s is only moderately suited for networking UAVs” as the system only switched to two-hops for 3% of the transmitted packets [31]. This urges further research in 802.11 mesh networks for drones.

Mesh networks dynamically self-organize and self-configure. To achieve this operation, mesh networks run discovery and peering protocols to locate other nodes and to manage membership of the mesh network. This is handled by a Mesh Peering Management (MPM) mechanism. Essentially, a joining node has to discover the operative wireless mesh network by scanning all radio channels and waiting for beacons (passive mode) or by issuing beacon requests and awaiting beacon responses (active mode). Direct communication between neighbor nodes is allowed only when they are peer mesh nodes. After a mesh discovery, two neighbor mesh nodes may agree to establish a mesh peering to each other.

The IEEE 802.11s amendment adds an MPM protocol that facilitates the establishment and closure of the mesh peering (Section 6.3.73 of [27]). Generally speaking, MPM request messages are used by a Station Management Entity (SME) function to establish, confirm, or close a mesh peering with other peering nodes. These peering are managed by the mesh nodes through the MAC Sublayer Management Entity (MLME) function. The MPM confirm message reports the results of a request. The MPM indication message is used by the MLME to report any peering states with other nodes to the SME. Finally, MPM response messages are used to send responses to the MLME specified by the peer node MAC addresses.

Mesh networks relay messages using either a flooding technique or a routing mechanism. Messages are forwarded along a path from a sender to a receiver node. Nodes are making forwarding decisions based on the path information of the network. Basic forwarding information consists of the destination mesh node MAC address, a next-hop address, a precursor list, and the lifetime of forwarding information. A mesh path selection protocol may benefit from combining reactive and proactive elements that enable efficient path selection in a wide variety of mesh networks. Although agnostic to any specific routing protocol, the IEEE 802.11s mesh network standard promotes the Hybrid Wireless Mesh Protocol (HWMP) as the preferred routing protocol (Section 13.10 of [27]). HWMP is a hybrid protocol that combines the flexibility of an on-demand path selection process with proactive topology tree extensions. The reactive part is inspired by the Ad Hoc On-Demand Distance Vector (AODV) protocol [28] adapted for MAC address-based path selection and link metric awareness. Two modes of operation exist: 1) the on-demand mode allows mesh nodes to communicate using peer-to-peer paths and 2) the proactive tree building mode provides a tree building functionality added to the on-demand mode. Path information is maintained by the use of four protocol messages: PREQ (path request), PREP (path reply), PERR (path error), and RAAN (root announcement). If a source mesh node needs to find a path to a destination mesh node. It broadcasts a PREQ with the path target specified in a list of targets. When a mesh node receives a new PREQ, it creates or updates its path information to the originator mesh node and propagates the PREQ to its neighbor peer mesh nodes. After creating or updating a path to the originator mesh node, the target mesh node sends an individually addressed PREP back to the originator mesh node. If the mesh node that received a PREQ is the target mesh node, it sends an individually addressed PREP back to the originator mesh node after creating or updating a path to the originator. In the proactive mode, the root mesh node periodically propagates RANN messages in the network. The RANN messages are used to disseminate path metrics to the root mesh node and do not establish path information. Upon reception of a RANN, each mesh node that has created or refreshed a path to the root mesh node sends an individually addressed PREQ to the root mesh node via the mesh node from which it received the RANN. A few studies of the use of dynamic routing protocols for drone mesh networks have been reported [26][29]. In [26], the authors studied the performance of four available mesh routing protocol implementations (open80211s, BATMAN, BATMAN Advanced, and OLSR) in the context of swarming applications for drones. The paper evaluated the performance of IEEE 802.11s with emphasis on goodput and the transmission delay. In [29] the authors simulated the performance of HWMP and suggested an optimization that implemented the proactive tree-based routing scheme applied on ground infrastructure, while the reactive routing is initiated by the mobile mesh node. The study showed a significant reduction in routing overhead alongside improvements in transmission delay and Packet Success Rate (PSR). Other studies confirm that a pure reactive routing scheme may lead to significant degradation of network performance in terms of added delay and reduced goodput when attempted to be adapted to highly mobile network scenarios such as satellite constellation networks [30]. These protocols are briefly summarized in Table 10.

Table 10: Mesh networking control protocols.

Protocol	Type	Reactiveness	Metrics
HWMP (802.11s)	Hybrid	Hybrid	Range. PSR, Delay, Overhead
BATMAN(-adv)	Link State	Proactive	Goodput, Delay
OLSR	Link State	Proactive	Goodput, Delay
AODV	Distance Vector	Reactive	PSR, Goodput

4.3.3 Drone-to-ground communication

Commercial-off-the-shelf (COTS) T&C systems are available for controlling individual drones. These are typically based on wireless communication using proprietary protocols and carrier frequencies in unlicensed bands on 433 MHz or 868 MHz in Europe. While these systems are quite convenient for testing, they are a poor choice for a fully integrated D4S system design.

In the D4S project, we specify two methods for providing D2G communication.

- LoRa is a LPWAN technology for long-range with high reliability. The downside of LoRa is its relatively low data rates supported. LoRa is often connected with the term LoRaWAN, which provides a more complete communication solution including medium access control and network layer support, which are functions that are also provided by the D4S network architecture. Therefore, the use of the LoRaWAN overlay is not needed for D4S.
- WiFi (IEEE 802.11) is an alternative for providing D2G communication over short distances e.g. up to 200 meters in outdoor environments. By using WiFi, it is possible to configure the GCS to be an anchor node in the WiFi mesh network to connect to multiple drones of a swarm. The approach makes the GCS become a member of the WiFi mesh network discussed in Section 4.3.2. For a point-to-point connection between the GCS and individual drones, the GCS may be configured as an access point connecting drone in a start topology.

A Ground Control Station (GCS) is a system that provides the user on the ground the ability to monitor the real-time status of the drone during the flight operation. The system connects the PC on and ground and the drone during the flight. Telemetry messages are periodically transmitted between the drone and the GCS. For the considered cases in the D4S D2C system, a potential candidate technology is Long-Range Radio (LoRa), which is defined as a Low Power Wide Area Network (LPWAN), proprietary of Semtech. LoRa utilizes the unlicensed Industrial, Scientific, Medical (ISM) frequency bands ranging particularly at 433 MHz, 868 MHz, and 915 MHz, but also 2.4 GHz [34]. Such bands allow for rapid developments providing coverages of up to 10-15 km in unobstructed communication paths.

LoRa is the physical layer protocol of LoRaWAN [32]. LoRa is designed for long-range communication at the expense of low data rates. The long range is achieved by using the LoRa modulation scheme, which is based on Chirp Spread Spectrum (CSS). The principle of CSS is to spread the signal out by encoding it using a higher rate chip sequence. By spreading the signal out to a larger bandwidth, the signal becomes more robust to interference and jamming. The chirp makes it easy to eliminate frequency offsets: “because of the linearity of the chirp pules, frequency offsets between the receiver and the transmitter area equivalent to timing offsets” [33]. The key feature of the LoRa modulation is high robustness and resistance to multipath and Doppler fading as well as signal interference. The robustness and the long range are achieved by the use of variable spreading factors (SFs) where the LoRa signal is differently over the channel spectrum. The relation between the symbol rate R_s, the channel bandwidth BW, and the spreading factor SF are as follows:

Equation 1

$$R_s = \frac{BW}{2^{SF}}$$

The nominal bit rate R_b is given by:

Equation 2

$$R_b = R_s\, SF \frac{4}{4+CR} = BW \cdot \frac{SF}{2^{SF}} \cdot \frac{4}{(4+CR)}$$

Where the *CR* is the coding redundancy factor determining the relative amount of redundancy in Forward Error Correction (FEC) code symbols added to the communication. The *CR* ranges from 1 to 4. If the *CR* is increased, more redundant bits are added to be able to correct more errors.

The Time-on-air (ToA) metric determines the amount of time before a receiver obtains the LoRa signal. It offers a lower limit on the latency the communication channel can provide. The ToA is dominated by the time it takes the sender to modulate bit into the channel plus the time it takes the receiver to extract bits. The time delay due to signal propagation may be ignored. The ToA can be expressed as follows:

Equation 3

$$ToA = \frac{2^{SF}}{BW} \cdot N_{symbols}$$

Where $N_{symbols}$ is the number of symbols transmitted. The computation of the number of symbols differs depending on the parameters of the modulation. It is the sum of symbols needed for the protocol preamble, the LoRa protocol header, the data payload, and remaining bits for the CRC check value (see [34] for details).

Figure 16 shows the raw data rate and the ToA value for different spreading factors of LoRa using a CR of 4/5.

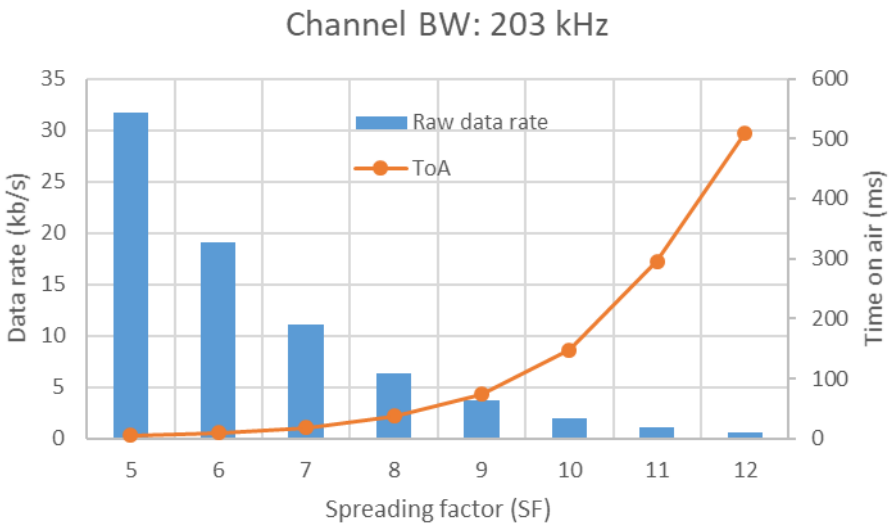

Figure 16: Data rate and Time on Air values for different LoRa spreading factors.

It can be read from the figure that a communication channel requiring a latency of less than 50 ms, which is

the case for the C2 channel the maximum spreading factor to be used is 8 for a bandwidth of 203 kHz. This represents a trade-off with the range of the LoRa channel as that longest range is achievable with the highest spreading factors. We expect that it is realistic to achieve a range of up to 2 km with a spreading factor of 8 or less in urban environments.

The LoRa protocol operates in the sub-1 GHz band (868 MHz). This is an advantage as the band undergoes less attenuation and multipath fading than bands, i.e., 2.4 GHz [11]. The 868 MHz band is an unlicensed frequency band, meaning that everyone can operate in it. To have co-existence with different wireless technologies, the European Telecommunications Standards Institute (ETSI) has set limits on the transmission power and duty cycle restrictions in the frequency bands in Europe.

When used for T&C with the GCS we specify MAVLink as the messaging protocol for the drones as it allows for data framing, packet sequencing, as well as the sender and component identification for data sent of the "wire". Two modes are supported: topic mode and point-to-point mode. In topic mode, the MAVLink implements a publish-subscribe system where receiving units choose specific topics to subscribe to. Typical examples for this communication mode are all autopilot data streams like position, attitude, etc. The point-to-point mode requires a target identification to be present. The protocol can support up to 255 robotic vehicles. MAVLink is supported on PX4 and QGroundControl. Moreover, the MAVROS software package provides a translation between RSO and MAVLink and thereby provides integration of MAVLink in the ROS software execution environment. The combination of LoRa, MAVLink, and MAVROS will be a foundation for the D2G communication for T&C in the D4S project.

When the GCS can connect to the Internet it can act as a gateway for the drone system to access cloud services. Such connection may be made by using publish mobile data service with 3G/4G or 5G services or through any other access technology carrying IP. Interface to the ROS communication can be made possible through the ROSbridge software package [3]. The ROSbridge provides an interface the ROS messing layer of the drone system. There is a variety of front ends that interface with ROSbridge, including a WebSocket. Alternatively, a GRE tunnel [35] of ROS messaged over IPv6 can be established between the GCS and the Data Broker network.

4.3.4 Drone-to-cloud communication

The system supports two distinct ways of interacting with cloud services. The first method of interaction is via web services supported by protocols like HTTP/REST or WebSockets. HTTP/REST is useful for communication that inherently needs a stateless information exchange e.g., telemetry reporting whereas WebSocket is a stateful protocol where communication happens over a reusable TCP connection. On the other hand, HTTP is inherently a stateless protocol. The other approach is to use the message-oriented middleware components in ROS. This approach makes use of a proxy function called ROSbridge. The ROSbridge protocol allows the sending of JSON formatted data commands to a ROS system [3]. It provides a WebSocket transport layer among others. Both methods allow cloud services to be implemented as standard web services that can interact with the drone swarm system using standard web transport protocols.

5 Simulation environment

Simulations of protocol and algorithms are an utterly important part of the development process for designing, integrating, and testing multi-drone systems. There are significant time savings involved by discovering problems in software early in the design phase before the software is tested on the intended hardware. Simulation environments such as Gazebo [40] allow the testing of software in a virtual world that implements drone systems with emulated sensors such as cameras. Gazebo offers the ability to accurately and efficiently simulate populations of drones in complex environments.

This section describes first the approach and the structure of a multi-drone simulation setup. Hereafter follows a brief introduction to the Gazebo simulation environment.

5.1 Structure of a multi-drone simulation

The simulation environment follows the Software-In-The-Loop (SITL) structure and is implemented by the simulation software, flight controller firmware, and application ROS package. Application ROS packages refer to different function implementations of the system, e.g. inspection, recharging, swarm control, and autonomous flight. By using the ROS middleware, ROS packages contain interfaces to the simulation software and the flight controller firmware. PX4 is selected for the flight controller firmware, which is the same firmware that is deployed on the OBC of the drone. The robot simulation platform Gazebo is used as the simulation software. It uses the models and configurations to define the simulation scenarios that emulate the behaviors of the drones for achieving different functions in different environments.

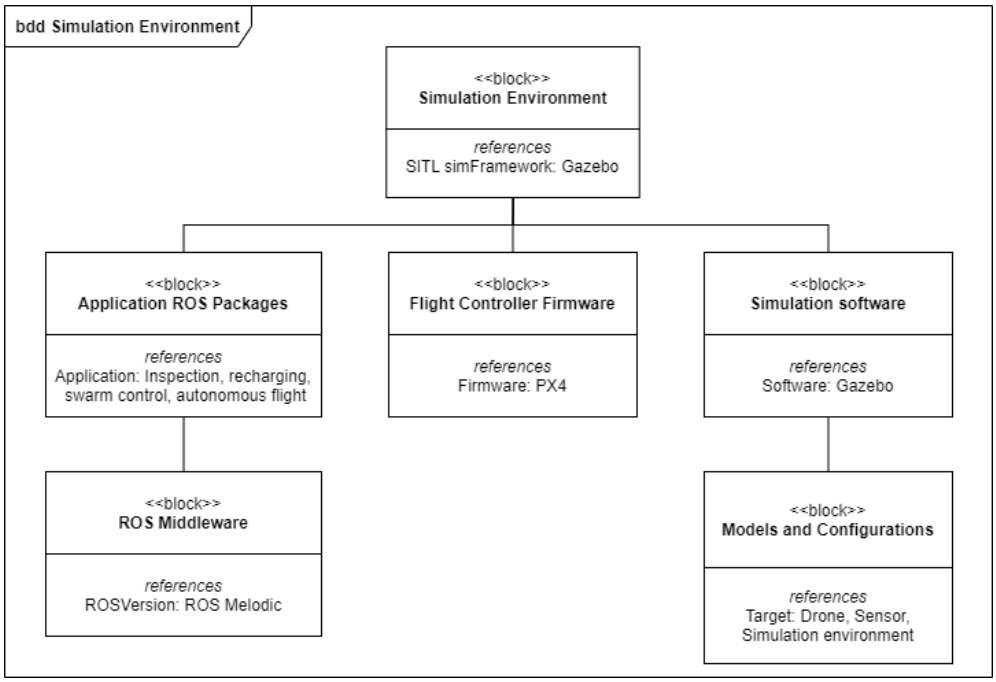

Figure 17: Structure of the drone simulator system architecture.

Docker technology is used for the fast deployment of the simulation environment, as shown in the attached image. The toolbox of SSH and VNC are used to provide a graphical interface for remote access of users. RViz is adopted to visualize the data from the drone. Each docker container represents a simulated drone. For testing multi-drone functions, multi-master ROS communication in a local network is established.

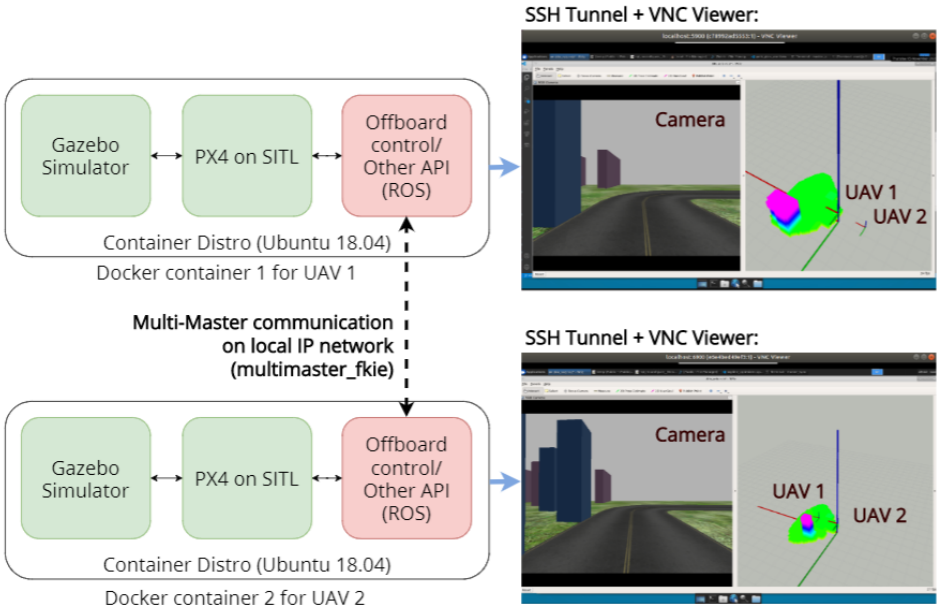

Figure 18: A Docker-based simulation platform for the multi-drone system.

The simulation environment is designed to be used for validating the data link and software functions with the system models and the simulated environment configurations. It provides a safe and close to the real environment to test different submodules of the system during the development. Models and configurations are defined for different test scenarios.

5.2 Gazebo environment

The Gazebo simulation environment provides virtual worlds for different simulation scenarios. This includes varied heights of terrain, trees, manmade objects of different kinds. Figure 19 shows an image from the simulation environment seen from the drone's initial perspective (Figure 19a) as well as a general overview of the entire simulation world (Figure 19b).

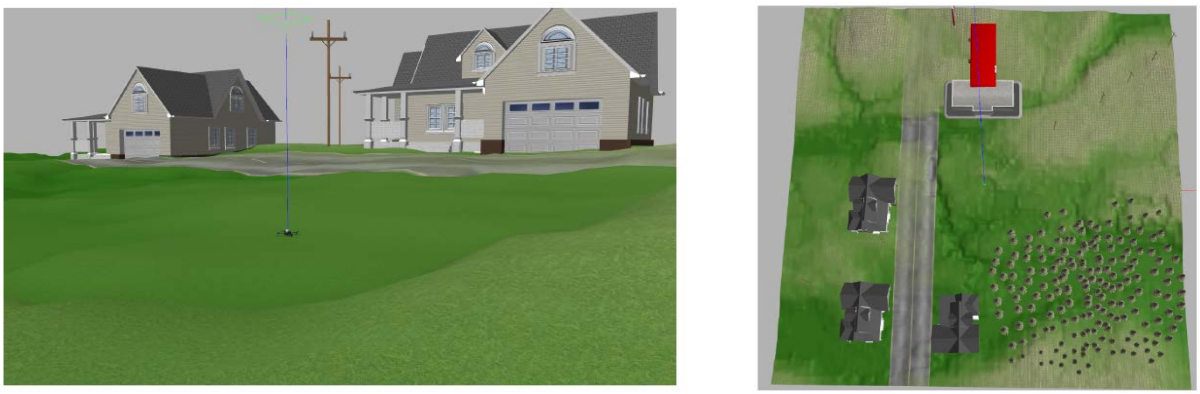

Figure 19: Gazebo views. Left: An example image from the simulation environment in Gazebo is seen from the drone's perspective. Right: Example of Gazebo world map.

A Gazebo "world map" is built using standard components of Gazebo, and consists of houses, trees, a gas

station, telephone poles, and a road. The result of the 3D mapping service on a realistic scene can be imported into the Gazebo simulation platform.

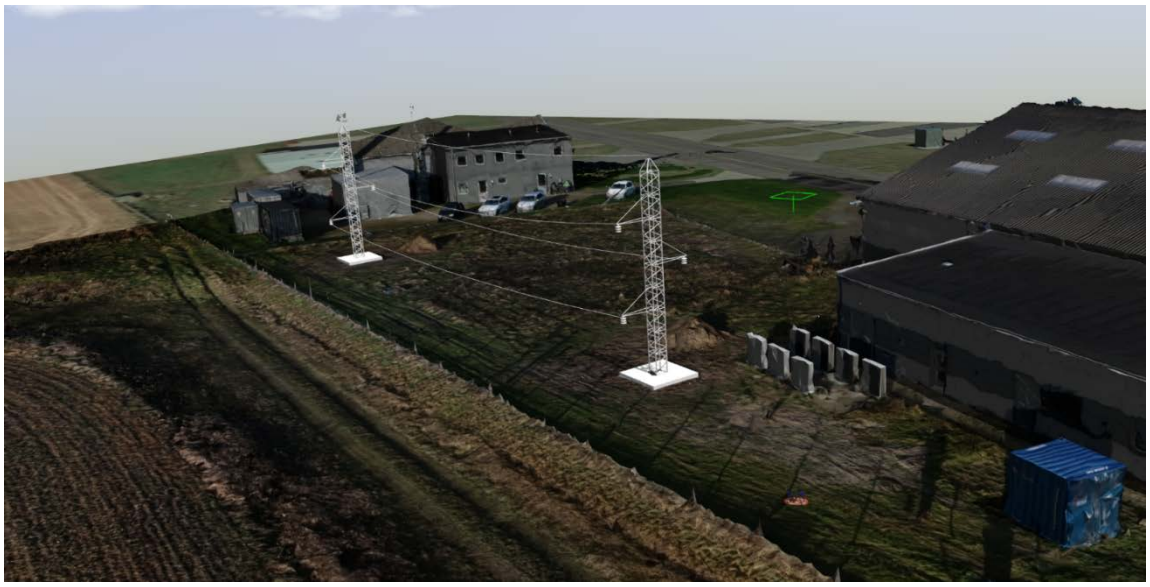

Figure 20: Example of the Gazebo simulation environment: Modeled by a test facility of UAS center located in Hans Christian Andersen Airport.

Figure 20 shows an example of a Gazebo simulation environment that is modeled based on the test facility for the drone. The test facility is established in Hans Christian Andersen airport near Odense in Denmark. It contains powerlines and a pair of power pylons and can for instance be used to test algorithms for navigating and detecting the cable as part of the procedure for autonomous recharging. The simulated environment can among other be used to test algorithms for cable detection and grasping.

6 Test and validation scenarios and environments

This section describes the approach to testing the multi-drone system. Besides the virtual test environment introduced in Section 5, the physical test environments planned for testing are introduced.

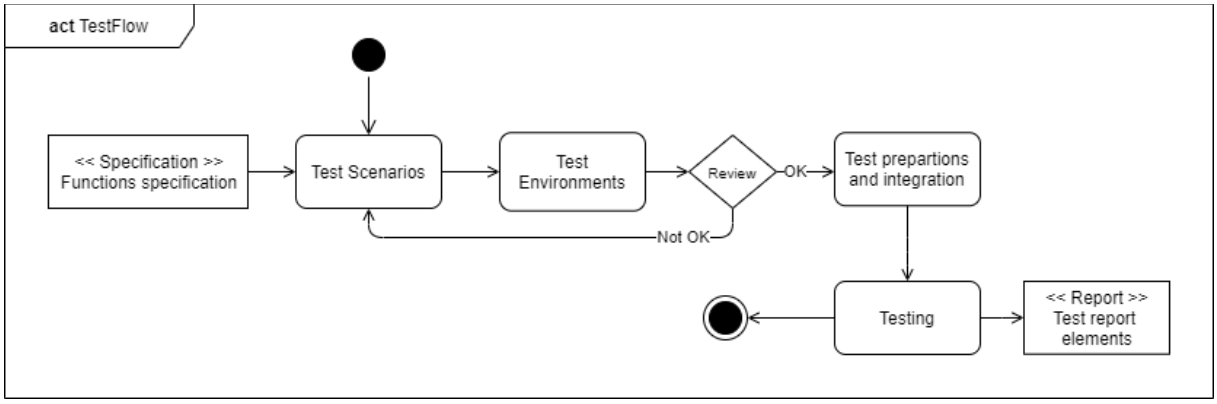

Figure 21: Illustration of the test procedure for the function testing in WP5.

Figure 21 shows the planned workflow for testing the multi-drone system (WP5). The function specification (this document) serves as input in the test design process. Test scenarios are generated based on function description and the relevant requirements (cf. Section 2.1) are mapped to these scenarios. Test scenarios are constructed to make a sufficiently good coverage of the functional requirements. Test scenarios will be related to one or more test environments including a simulation environment, indoor testing facility as well as outdoor facilities. Test scenarios and checks of the test environment are reviewed before proceeding with the preparations for the testing. This preparation includes the integration of software components with the relevant target drone platforms as well as the preparation of the GCS to support the testing. The testing produces results that document the outcome of the test, Based on these results an assessment of the success of the design and the following testing can be done. The entire test flow can subsequently be iterated yielding improved test scenarios, test environments, and better integrated multi-drone designs.

6.1 Test environments

WP5 of the D4S has access to several drone test facilities that are relevant for the multi-drone system design and can be used during the development phase. These environments are briefly introduced below:

- **SDU UAS drone test facility:** The test facility is located at Hans Christian Anderson airport near Odense in Denmark. It comprises a hangar (Figure 22) for indoor drone flying as well as an outdoor facility (Figure 23). Furthermore, it can be permitted to fly in a larger open area at the coastal shore and over the sea.
- **AU Deep Tech Hub:** An indoor drone cage is available at AU's facility at Skejby near Aarhus in Denmark (Figure 24, left).
- **Mollerup forest:** Mollerup forest has been chosen as a location for outdoor drone testing in the outskirts of Aarhus, Denmark. The forest is located in a rural area and drone flying with a license is permitted.

Figure 22: UAS Test center at Hans Christian Andersen Airport near Odense in Denmark. a) The hangar from outside and b) from inside.

Figure 23: Outdoor test facility at the Hans Christian Andersen Airport.

Figure 24: Drone test environment in Aarhus. Left: Indoor facility at Deep Tech Hub in Aarhus N, Denmark. Right: Outdoor flying in an open area near Mollerup forest, Aarhus N.

7 References

[1] D2.4: “Use-case Document” version 3.5, Drones4Safety, 2021

[2] D2.5: “Final System Requirements Document” version 1.0, Drones4Safety 2021

[3] ROSbridge_suite, URL: http://wiki.ros.org/rosbridge_suite

[4] L. Shi, N. J. Hernández Marcano and R. H. Jacobsen, “A Survey on Multi-unmanned Aerial Vehicle Communications for Autonomous Inspections,” 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 2019, pp. 580-587, doi: 10.1109/DSD.2019.00088

[5] Timothy Patterson, Sally McClean, Philip Morrow, and Gerard Parr. “Modelling safe landing zone detection options to assist in safety critical uav decision making”, Procedia Computer Science, 10:1146–1151, 2012. ANT 2012 and MobiWIS 2012.

[6] Maryam Asadzadeh Kaljahi, Palaiahnakote Shivakumara, Mohd Yamani, Idna Idris, Mohammad Hossein Anisi, Tong Lu, Michael Blumenstein, and Noorzaily Mohamed Noor. “An automatic zone detection system for safe landing of uavs”, Expert Systems with Applications, 122:319 – 333, 2019.

[7] Timothy Patterson, Sally McClean, Philip Morrow, Gerard Parr, and Chunbo Luo. “Timely autonomous identification of uav safe landing zones”, Image and Vision Computing, 32(9):568 – 578, 2014.

[8] MAVROS packet information @ ros.org. URL: https://wiki.ros.org/mavros

[9] MAVLink packet information @ ros.org. URL: http://wiki.ros.org/mavlink

[10] Morgan Quigley et al. “ROS: an open-source Robot Operating System.” In: vol. 3. Jan. 2009, pp. 1–6.

[11] A. Koubâa et al. “Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey.” In: IEEE Access 7 (2019), pp. 87658–87666.

[12] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger,905R. Wheeler, A. Ng, ROS: an open-source Robot Operating System, Technical Report, 2009. URL:http://stair.stanford.edu.

[13] A. Tiderko, F. Hoeller, T. Röhling, “The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems”, 2016. doi:10.1007/978-3-319-26054-9_24.91032

[14] Y. Maruyama, S. Kato, T. Azumi, “Exploring the performance of ROS2”, Proceedings of the 13th International Conference on Embedded Soft-708ware, EMSOFT 2016, Association for Computing Machinery, Inc, 2016.709doi:10.1145/2968478.2968502.

[15] E. Eros, M. Dahl, A. Hanna, A. Albo, P. Falkman, K. Bengtsson, “Integrated virtual commissioning of a ROS2-based collaborative and intelligent automation system”, IEEE International Conference on913Emerging Technologies and Factory Automation, ETFA, volume 2019-914September, Institute of Electrical and Electronics Engineers Inc., 2019,915pp. 407–413. doi:10.1109/ETFA.2019.8869444.

[16] Docker website, URL: https://www.docker.com/. Accessed 2/3/2021.

[17] Datasheet: Raspberry Pi 4 Computer Model B, URL: https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf. Accessed 2/3/2021.

[18] Intel NUC mini PC, URL: https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html, Accessed 2/3/2021.

[19] NVIDIA Jetson Xavier NX module and developer kit. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/. Accessed 2/3/2021.

[20] MAVlink, http://mavlink.io/en/

[21] C. Perkins, D. Johnson, and J. Arkko, Mobility Support in IPv6, Internet society, RFC 6275, July 2011

[22] M. Bagnulo, P. Matthews, and I. van Beijnum, “Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers”, Internet society, RFC 6146, April 2011.

[23] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum, “DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers”, Internet Society, RFC 6147, April 2011.

[24] TAYGA Simple, no-fuss NAT64 for Linux, web site, URL: http://www.litech.org/tayga/. Accessed 04-03-2021.

[25] S. Morgenthaler, T. Braun, Z. Zhao, T. Staub, M. Anwander, “UAVnet: A mobile wireless mesh network using unmanned aerial vehicles”, 2012 IEEE Globecom Workshops, 2012, pp. 1603–1608.

[26] J. Pojda, A. Wolff, M. Sbeiti, C. Wietfeld, “Performance analysis of mesh routing protocols for uav swarming applications”, 2011 8th International Symposium on Wireless Communication Systems, 2011, pp. 317–321.

[27] Iso/iec/ieee international standard - information technology-telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (mac) and physical layer (phy) specifications, ISO/IEC/IEEE 8802-11:2012(E) (Revision of ISO/IEC/IEEE 8802-11-2005 and Amendments) (2012) pp. 1–2798.

[28] C. Perkins, E. Belding-Royer, S. Das, RFC3561: “Ad Hoc On-Demand Distance Vector (AODV) Routing”, RFC 3561, 2003. URL: https://www.rfc-editor.org/rfc/rfc3561.txt.

[29] C. J. Katila, A. Di Gianni, C. Buratti, R. Verdone, “Routing protocols for video surveillance drones in ieee 802.11s wireless mesh networks”, 2017 European Conference on Networks and Communications (EuCNC), 2017, pp. 1–5.

[30] N. J. Hernández Marcano, J. G. F. Nørby, R. H. Jacobsen, “On Ad hoc On-Demand Distance Vector Routing in Low Earth Orbit Nanosatellite Constellations”, IEEE Vehicular Technology Conference (VTC) 2020, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 1–6.

[31] E. Yanmaz, S. Hayat, J. Scherer, C. Bettstetter, “Experimental performance analysis of two-hop aerial 802.11 networks”, 2014 IEEE Wireless Communications and Networking Conference (WCNC), 2014, pp. 3118–3123.

[32] LoRa Alliance. “LoRaWAN, what is it? A technical overview of LoRa and LoRaWAN”, Technical Marketing Workgroup 1.0, November 2015. URL: https://lora-alliance.org/resource_hub/what-is-lorawan/ (visited on 06/3/2021).

[33] Aloÿs Augustin et al. “A study of LoRa: Long Range & Low Power Networks for the Internet of Things”. In: Sensors 16.9 (2016), p. 1466.

[34] SemTech. “Long range, low power 2.4 GHz Wireless RF Transceiver with ranging capability”, URL: https://www.semtech.com/products/wireless-rf/24-ghz-transceivers/sx1280 (accessed 06/03/2021).

[35] C. Pignataro, R. Bonica, and S. Krishnan, “IPv6 Support for Generic Routing Encapsulation (GRE)”, Internet Society, RFC 7676, October 2015.

[36] D3.1: “Specification of Harvester System”, version 1.0, Drones4Safety, 2020

[37] “XML 1.0 Specification”. World Wide Web Consortium. Retrieved 12 March 2021.

[38] T. Bray et al., “The JavaScript Object Notation (JSON) Data Interchange Format”, Internet Society, RFC 8259, URL: https://tools.ietf.org/html/rfc8259. Accessed 12 March 2021.

[39] GAZEBO; “Robost Simulations Made Easy”, URL: http://gazebosim.org/. Accessed 13-03-2021.

[40] DJI Manifold 2 onboard computers. URL: https://www.dji.com/dk/manifold-2. Accessed 13-03-2021.

[41] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, Internet Society, RFC 8200, July 2017. URL: https://tools.ietf.org/html/rfc8200

[42] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, Internet Society, September 2007, URL: https://tools.ietf.org/html/rfc4861

[43] R. Vida, L. Costa, “Multicast Listener Discovery Version 2 (MLDv2) for IPv6”, Internet Society, RFC 3810, June 2004, URL: https://tools.ietf.org/html/rfc3810

[44] Readthedocs, colcon - collective construction, URL: https://colcon.readthedocs.io/en/released/. Accessed 25-03-2021.