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 Drone Inspection as a Service platform (M6)

D6.1

Due date of deliverable: <30.11.2020>

Start date of project: June 1st, 2020

Type: Deliverable
WP number: WP6

Responsible institution: SDU
Editor and editor's address: Lea Matlekovic, SDU

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

Version 0.1
Release Date: June 30, 2020

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
1	20/10/2020	Lea Matlekovic	SDU	Created initial version
2	25/10/202	Lea Matlekovic	SDU	Added sections
3	30/10/2020	Golizheh Mehrooz	SDU	Update acronym
4	2/11/2020	Lea Matlekovic	SDU	Added images
5	2/11/2020	Golizheh Mehrooz	SDU	Add 3D constructure
6	2/11/2020	Naeem Ayoub	SDU	Faults detection system
7	9/11/2020	Lea Matlekovic	SDU	Corrections based on partners' feedback
8	11/11/2020	Sam Münchow	ARIC	Positioning service
9	22/11/2020	Lea Matlekovic	SDU	Acronym update, contents table update

Executive Summary

The following deliverable is focused on design and specification of Drone inspection as a Service platform and is part of the WP6 – Mission control and navigation. It describes the software architecture and services for mission control and navigation.

The document is structured in 4 sections. First section describes the D4S project and how the platform development contributes to the project's main scope. It describes an overall design of the platform. Following sections describe various parts of the platform in details. Second section describes a part of the platform deployed on the cloud server. Cloud software architecture is divided on backend and frontend and described in subsections. Third chapter describes architecture of the software deployed on the drones. Fourth, last chapter describes the network communication architecture. Specification of drone swarm software and communication will be further developed and tested in WP4 and WP5.

Contents

References

“Developers bring their ideas to life with Docker”, https://www.docker.com/why-docker

“What is Kubernetes?”, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

“Jenkins User Documentation”, https://www.jenkins.io/doc/

“GitLab CI/CD”, https://docs.gitlab.com/ee/ci/

“PostgreSQL: The World's Most Advanced Open Source Relational Database”, https://www.postgresql.org/

“About PostGIS”, https://postgis.net/

“Map Features”, https://wiki.openstreetmap.org/wiki/Map_Features

“QGroundControl”, http://qgroundcontrol.com/

“Open source autopilot designed for scale”, https://px4.io/

“MAVROS -- MAVLink extendable communication node”, http://wiki.ros.org/mavros

“About ROS” https://www.ros.org/about-ros/

Acronyms

Acronym	Description
ROS	Robot Operating System
IDE	Integrated Development Environment
CI	Continuous Integration
CD	Continuous Delivery
D4S	Drones for Safety
DIaaS	Drone Inspection as a Service
CPU	Central Processing Unit
GPU	Graphics Processing Unit
SfM	Structure from Motion
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
RTPS	Real-Time Publish Subscribe
VPN	Virtual Private Network
IP	Internet Protocol
DL	Deep Learning
ML	Machine Learning
SBD	Single Board Device
AAN	Autoencoder Adversarial Network
CUDA	Compute Unified Device Architecture
DDS	Data Distribution Service

1 Introduction

The main scope of the Drones4Safety (D4S) project is to develop a system of autonomous, self-charging, and collaborative drones that, inspecting an extensive portion of transportation infrastructures in a continuous operation, can increase the safety of the European civil transport network. The project outcomes, in forms of software services and hardware drone system, will offer to railway and bridge operators the chance to inspect their transportation infrastructure accurately, frequently, and autonomously.

The main purpose of this document is to provide a description and specification of software services needed for autonomous inspection. All the software enabling the infrastructure inspection will be developed as a Drone Inspection as a Service platform. The platform includes cloud services for mission control, monitoring, and navigation. It also includes web interface, drone operation system and communication between the platform, web interface and drones.

The platform architecture is designed to support modifications and additions, so that old features can be easily modified, and new features can be easily added. Modularity is particularly important since we can expect to extend the platform with different infrastructure inspection capabilities in the future.

1.1 Platform design

Drone Inspection as a Service (DIaaS) is a cloud-based platform for mission control and autonomous navigation of drone systems that allows users to plan inspection missions with a given drone system, execute inspection missions by deploying a drone system, monitor the state and location of a drone system, control a drone system remotely and safely, and deliver inspection images, sensory data, and on-board analysis results to analysis systems (D4S_DES_REQ_0620, D4S_DES_REQ_0630). Figure 1 shows the graphical representation of Drone Inspection as a Service platform.

The platform is designed in three layers consisting of cloud services, the drone swarm, and a network facilitating communication between the former two (D4S_DES_REQ_0650). The cloud services are based on a microservices approach (D4S_DES_REQ_0660). The drone operating system interacts with the cloud services based on a modular distributed message-passing approach. The network handles routing, Quality-of-Service etc.

The design further is divided into a frontend and a backend. The frontend is web-based, relies on the cloud services, and allows the user (i.e., the remote operator) to interact with the drone swarm, to visualize its status, and to schedule missions. The backend relies also on the cloud and is accessed by both the frontend and by the drone swarm, in the latter case mediated through the network.

For the cloud, services are containerized in a Kubernetes cluster and will include fault detection, inspection mission control, swarm fleet management, and data analysis. Microservices approach gives modularity to the architecture, so additional services can always be easily integrated. Integration to the platform in automatized and the future focus can be at additional features development.

The drone modules interacting with the cloud through the network are developed as nodes in the ROS/ROS2 used as the drone operating system.

Described platform design assures long-endurance, continuous and autonomous inspection campaigns of the target infrastructure without direct human intervention (D4S_FUN_REQ_0070).

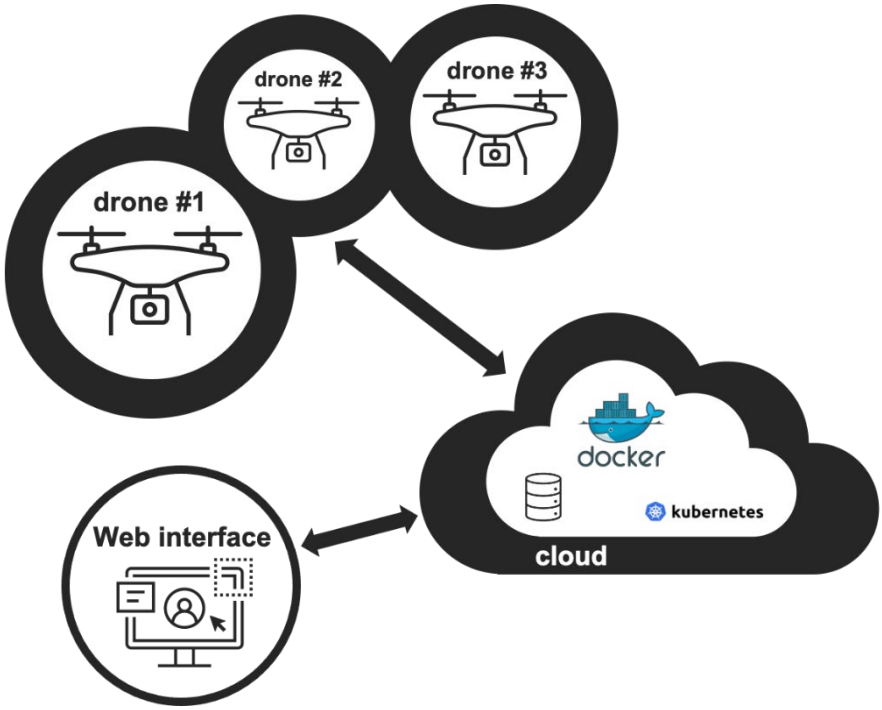

Figure 1 *Drone Inspection as a Service platform*

2 Cloud Software Architecture

Cloud computing is a paradigm shift in the way computing resources are used and applications are delivered. These resources include services, storage, and the network infrastructure along with software applications. Cloud computing refers to providing these resources over the internet to the public or an organization. The cloud is one of our servers and all the services will be deployed there. Some other cloud solutions will be revised in the future as the traffic on the platform increases. GitLab is installed on our premises and will be used for development of the Drone Inspection as a Service platform. GitLab is a complete DevOps platform and provides toolchain for software development and operations. After the software is integrated to the GitLab it will be automatically deployed to the cloud. Jenkins tool will be investigated for continuous deployment. Benefit of the platform deployment in the cloud is that it can be reached from anywhere by users and developers. Developers contribute through the integrated IDE, locally installed on their computers. The IDE choice is left to the developers.

The system architecture is based on microservices. Microservices structures an application as a collection of loosely coupled services. Decomposing an application into smaller services improves modularity and makes the application easier to develop and test in parallel and independently by multiple teams. Microservices architectural style assures continuous integration and deployment. Continuous integration and deployment will be enabled for contributions to the platform. To be able to do that, continuous integration and deployment tools are investigated and carefully chosen. CI and CD are set of operating principles, and collection of practices that enable application development teams to deliver code changes more frequently and reliably. The implementation is known as the CI/CD pipeline. Since these procedures automatize deployment steps, developers can focus on meeting the project requirements, code quality and security. Continuous integration drives developers to implement minor changes and check in code to GitLab frequently. It is expected that, by implementing the continuous integration, the platform development will speed up since all the developers will

be able to see and work with the updated code. It leads to the better collaboration between the developers and better software quality. Continuous delivery automates the applications delivery to the selected infrastructure environment, in this case to the server. Since the platform is deployed only to one server in the beginning, foundations for continuous delivery are set up to assure easier platform expansion in the future.

Services will be integrated to the cloud in containers. Services are containerized using Docker tool and deployed using Kubernetes container management tool. Containers offer a logical packaging mechanism in which applications or services can be abstracted from the environment in which they run. It provides a consistent environment, including software dependencies, specific programming language runtimes, libraries, etc., no matter where the container is deployed. Containers virtualize CPU, GPU, memory, storage network resources. It provides isolated environments for running services and enables assignment of resources, e.g. CPU, GPU, with considering system and service requirements, e.g. reliability, real-time. Docker is popular, open-source container format that is supported on many cloud platforms and by Kubernetes system. Kubernetes groups containers that make up an application into logical units for easy management and discovery.

Each service in the system provide one functionality and act like an independent application. All the services communicate with each other. The main advantage of the microservices architecture is its scalability. Drone Inspection as a Service platform will be available in the future to infrastructure inspectors around the world. To ensure its architecture will be able to grow without re-writing the code, it is important to set a foundation in the beginning. When the load on the platform gets heavier, the platform will be prepared to handle it. In case of the increased traffic, the Kubernetes is responsible to make replicas of the service and distribute the traffic to all of them.

All the data will be stored in the database on the cloud. The services will interact with the database and will store and retrieve flight plans, tracking information, and inspection images when needed.

2.1 Backend

Every backend capability will be stored in the cloud in the previously mentioned architectural style. Figure 2 shows the graphical representation of the cloud and its architecture. Services are divided by their functionalities. Image processing services are responsible for highly computational image processing and machine learning. Safety services are responsible for exceptional situations when the flight is not possible, and the drone must land safely. Charging services are responsible for receiving information of the drones' battery level and to plan and command the drones when and where to charge. Mission planning and scheduling services are responsible to receive operator's input on mission goals, plan the mission suitable for current weather conditions and time schedule the mission considering the charging circumstances. Weather services are responsible to give the weather information around the flying area, communicate with the drone sensors to provide the right weather conditions to mission planning and scheduling services. Also, data storage and delivery services will take care of database management. Positioning services calculate drones' position in real-time based on the flight plan and support the GNSS position when the communication with the drone cannot be established. Communication between services is orchestrated in Kubernetes.

2.1.1 Image analysis services

Image analysis services will include infrastructure recognition software. Machine learning algorithms will be developed and trained to classify infrastructure from images (D4S_FUN_REQ_0230). All images received from the drone will automatically be categorized by type: bridge component or railway component.

Image analysis services will analyze images received from the drones, following requirements of the system. In order to develop precise inspection path plans, 3D model of the target infrastructure needs to be created (D4S_FUN_REQ_0030). Such 3D model can be created as a geometric mesh of polygons and can be constructed based on a set of images taken from different perspectives by an inspection drone using Structure from Motion (SfM) method. SfM is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals. However, the construction of such 3D models for a large-scale operation environment is computational and memory heavy and poorly fit with the constrained performance of the on-board computer of the drone. Therefore, a service on the cloud for generating the 3D mesh models will be designed and implemented based on WP4 results. After the 3D model of bridges, railways and railway overhead lines are created, detailed inspection path plan can be calculated by mission planning services.

2.1.2 Weather services

Weather services will analyze the weather data from weather stations and determine if the flight area is safe for flying. Mission plan will adapt based on the weather information. (D4S_FUN_REQ_0250)

2.1.3 Safety services

Safety services will be monitoring the drones' safety and established connection with drones and between drones. It will report to the user interface the strength in connections and safety state. If the safety is jeopardized, the drones will be automatically put in a safety state (D4S_FUN_REQ_0110). Depending on the situation, the safety state can mean putting the drones into the charging mode, hovering mode, or landing them to the ground. The drones should be able determine a safe landing zone using a depth camera and simple computer vision techniques.

2.1.4 Charging services

Charging services receive near real time data about drones' battery level. They include time estimation until charging will be needed, identification of the charging spots, their availability and estimation of charging time until the drones are fully charged (D4S_FUN_REQ_0390). While identifying the charging spot, services will take care to provide the information on overhead cables type (high-voltage or transmission lines/railway overhead lines) (D4S_FUN_REQ_0140). The battery timing schedule is included in mission planning schedule meaning it will be automatically identified when the charging is needed and charging time will be included into the inspection mission time (D4S_FUN_REQ_0130). Also, services provide a plan for docking to the charging spot. Swarm charging strategies (such as all drones in a swarm charge at the same time or swarms will continue with fewer drones) will be investigated in WP5 and supported by this service.

2.1.5 Mission planning services

One of the services for mission planning is routing along the infrastructure. To reach the inspection target, drones will fly along corridors surrounding infrastructure. The service for routing along the power lines identifies power towers on the street map and saves their location into a tree structure as nodes. When starting and ending points are chosen on the map, the service identifies two nearest power towers and creates an optimal route between them by connecting the nodes from the tree structure.

There will be more inspection route planning services dependent on inspection needs. Quick bridge inspection service will provide the bridge fly-around route for determination of mayor faults and collection of images for recreation of the bridge's 3D model. The route will be planned in order for drones to capture images from multiple angles and points of view, so that 3D model recreation is facilitated (D4S_FUN_REQ_0060). Detailed bridge inspection service will provide inspection route based on the 3D model of the bridge created by image analysis services. Additional services will take care of route planning for the railway tracks inspection. A quick inspection route will be based on map and satellite images. If necessary, after the data from this quick inspection has been analyzed and 3D models have been created, a detailed inspection path plan will be

developed. For every inspection target, the route is created using GPS coordinates saved as nodes in a tree structure. The nodes are extracted from the street map and the map is updated based on near real-time satellite images (D4S_FUN_REQ_0240). More precise set of nodes are chosen based on infrastructure 3D models (D4S_FUN_REQ_0190). The services will take into a consideration safety and flying regulations while planning the route (D4S_FUN_REQ_0100). They should receive traffic data and trains' timetables in order to safely coordinate the mission. The drones will fly only in permitted areas and avoid no-fly zones such as airports, transformer stations, residential areas...

2.1.6 Mission scheduling services

Scheduling services will provide the optimal order of visiting the nodes and time frame for mission completion. For scheduling a mission, the service receives a list of GPS coordinates that need to be inspected and finds an optimal visiting order for the given number of inspection drones. The services will be further developed to take into a consideration the charging plan and all environmental circumstances. Geographical limitations such as objects and obstacles will be taken into the consideration and permitted/restricted flight area will be defined (D4S_FUN_REQ_0270). The flights will be planned according to flight authorization and procedures (D4S_FUN_REQ_0290). Polygons on the map will be created for different flight requirements and restrictions and will be included in the path planning algorithm. The services will be able to recalculate the mission plan based on new circumstances that may occur during the flight (D4S_FUN_REQ_0220).

2.1.7 Positioning services

Positioning services calculate drone's position based on the previously planned mission and set parameters. Services also estimate uncertainties from the real position (D4S_FUN_REQ_0310). We should be able to visualize and update the drones' location on the map in near real time. Since the drones cannot report to the cloud each second, this calculation will be used to estimate drones' real position. Every time the drone reports to the cloud, the real location is updated. Services will also receive drones' real pose and velocity. If the difference between the drone's real position and calculated position is severe, the services will send and alert to the web interface (D4S_FUN_REQ_0380).

2.1.8 Data storage and delivery services

Data storage and delivery services will take care of virtual asset management. Services will receive telemetry data from drones' sensors and store it in suitable format. PostGIS will be used to store spatial data. PostGIS is spatial database extender for PostgreSQL object-relational database. Should the need for a more scalable data storage, processing, and delivery service arise, we could work towards integration with a lambada architecture consisting of e.g. Storm, Hadoop, and Druid. Also, when demanded, the services will retrieve flight plans, tracking information, and inspection images from the database and deliver to the user interface, to image analysis services or to any other service requesting. The services will take care of data formatting.

2.2 Frontend

Frontend will be based on the web user interface showing map and all the mission planning capabilities. Figure 2 shows the graphical representation of the web interface and its communication with the cloud. Web user interface is an ideal solution because it allows the operator to access the platform from anywhere in web browser, using any device connected to the internet. The interface will have role-based access to distinctive features. The roles will differ by user's ability to interact with the system. Only high-privileged roles will be able to control the system while low-privileged roles will have the ability only to monitor the system's state. (D4S_SEC_REQ_0710) The operator will be able to choose the type of the infrastructure to inspect, inspection type and select targets on the map. Also, it is possible to decide how many drones will participate in the mission. The operator will be able to command the drones to fly to the certain position or submit, change, and

remove the flight plan. The interface will show if the submitted plan is feasible (D4S_FUN_REQ_0260). If the mission with chosen targets is not feasible, the operator will receive an alert about possible conflicts (D4S_FUN_REQ_0360) and suggestion for conflict resolution (D4S_FUN_REQ_0370).

After setting up the mission requirements, the drones will automatically take-off, inspect the infrastructure, recharge when needed and land (D4S_FUN_REQ_0200). In case of the emergency, the safety service will provide all the information through the web user interface and the operator will be able to react if needed (D4S_FUN_REQ_0340). For example, an emergency could be when the strong wind is blowing and taking the drone out of the predetermined path. The system will detect the difference between the drone's real position and planed position and send an alert to the user interface (D4S_FUN_REQ_0380). The operator will then be able to decide the course of actions depending on the situation.

The interface serves also for mission visualisation, monitoring and supervision (D4S_FUN_REQ_0010). It will be able to show the drones' flight simulation, velocity, and position on the map in real-time (D4S_FUN_REQ_0300, D4S_FUN_REQ_0320). Also, it will receive a real-time information about air traffic and visualise on the map if there is another flying vehicle around (D4S_FUN_REQ_0330). The operator will be able to see and track the drones' position (latitude, longitude, and altitude) and distance from the given reference (D4S_FUN_REQ_0120). Each drone will be recognizable under the unique identifier and the operator will be able to see the planned mission for individual drone (D4S_FUN_REQ_0280). The interface will show the mission schedule with planed mission time and time to go until the end of the mission. The operator will see information based on the drones' sensors readings. The battery level of the drones will be visible. Through the interface, the operator will receive data and metadata on detected defects on the infrastructure in near real-time (D4S_FUN_REQ_0090). Also, it will be possible to monitor the weather on the inspection site.

After the mission, the operator will be able to see the flight history (flight plans, tracking information and other operational data) (D4S_FUN_REQ_0350), and browse the data acquired during the inspection through the web user interface (D4S_FUN_REQ_0210).

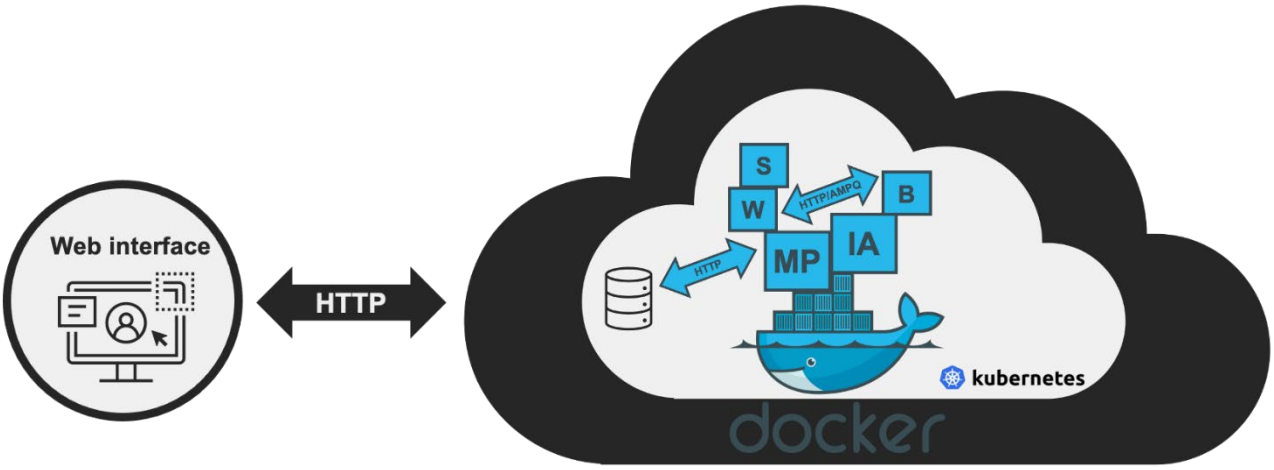

Figure 2 *Cloud software architecture and web interface (frontend) communication*

3 Drone swarm Software Architecture

A swarm of drones will be available for inspection. The operator will be able to choose how many drones should participate in the mission. This section describes software deployed on the drones. Further specification and tests are part of the WP5.

Drones will have PX4 autopilot software on board and the software will take care of drones' low-level control. PX4 is an open-source flight control software for drones and other unmanned vehicles. The software provides a flexible set of tools for drone developers to share technologies to create tailored solutions for drone applications. PX4 provides a standard to deliver drone hardware support and software stack, allowing an ecosystem to build and maintain hardware and software in a scalable way. Autopilot software controls the drone's actuators and provides the right inputs to the actuators in order to reach the desired destination. The desired destination is calculated in the cloud's mission planning services. The mission planning on the cloud is based on maps, satellite images and 3D models, but it still does not provide a completely safe flight plan.

In order to execute a safe flight, the drone must be aware of its surroundings in real-time. The local path planning software for mission plan adaptations on-board is a solution. It will use sensor data and images from the camera to make sure the flying path is safe. ROS middleware platform will be deployed on the on-board computer. Image processing for infrastructure faults detection will be deployed on-board. Figure 4 shows drone software architecture and its communication with the cloud.

3.1 Fault detection software

The main fault detection system is part of WP4. In addition to this, we also plan to investigate an on-board fault detection system with ROS nodes to perform different tasks. Based on supervised and unsupervised DL models, ROS nodes will perform following tasks:

- Running real-time on-board autonomous DL algorithm
- Detects the faults in different components
- Extract the image based on the faults and components
- Running unsupervised DL model for anomaly detection based on Variational Auto Encoders
- Running AANs for image feature extraction (D4S_TNA_REQ_0590)
- These features will be based on depth, viewpoints, camera position etc for 3D map generation.
- Detect the damages to electric traction overhead contact lines infrastructure (D4S_FUN_REQ_0040). These damages will be trained as positive classes (WP4-supervised DL method). Track deformation (depends on available dataset and can be performed with anomaly detection techniques). Obstacle avoidance (any know avoidable obstacles, so supervised DL will be used for known classes).
- Save the images in folder to root directory.

Then a communication algorithm (it can be a ROS node or built into the fault detection algorithm) between single board device (SBD) and cloud server will transfer images based detected components to the cloud database.

3.1.1 Supervised DL Framework

On-board fault detection software will be developed in order to automatically recognize faults on bridge and railway infrastructure (D4S_FUN_REQ_0020, D4S_FUN_REQ_0080). Deep learning methods will be used in the process (D4S_TNA_REQ_0570). Hardware choice for fault detection software deployment highly impacts the software's performance. The software will be developed and tested under the WP4.

Building and running a DL faults detection algorithm depends on following main steps.

- Collection, pre analysis and labelling of dataset (For now, labelling is being performed manually but in future, we will build an autonomous algorithm for auto-labelling)
- Application of DL algorithm for training, testing and Evaluation in terms of accuracy (for this step, we are using Supercomputer GPU server (for instance Nvidia-Telsa v100)).
- Selection of suitable SBDs for running the inference in real-time on board of the UAV (on-board running the DL model: CUDA enabled SBD can run DL algorithm more efficiently up to 15-20 FPS)

We plan to base the object and fault detection on YOLOv4, which is expected to provide high accuracy (D4S_PER_REQ_0410) and comes with the possibility to optimize the network for inference on sufficiently computationally strong SBDs. The YOLOv4 model can detect up to 80 different classes per frame (D4S_PER_REQ_0420). Figure 3 shows a suggestion of a possible overall structure of our DL based fault detection model.

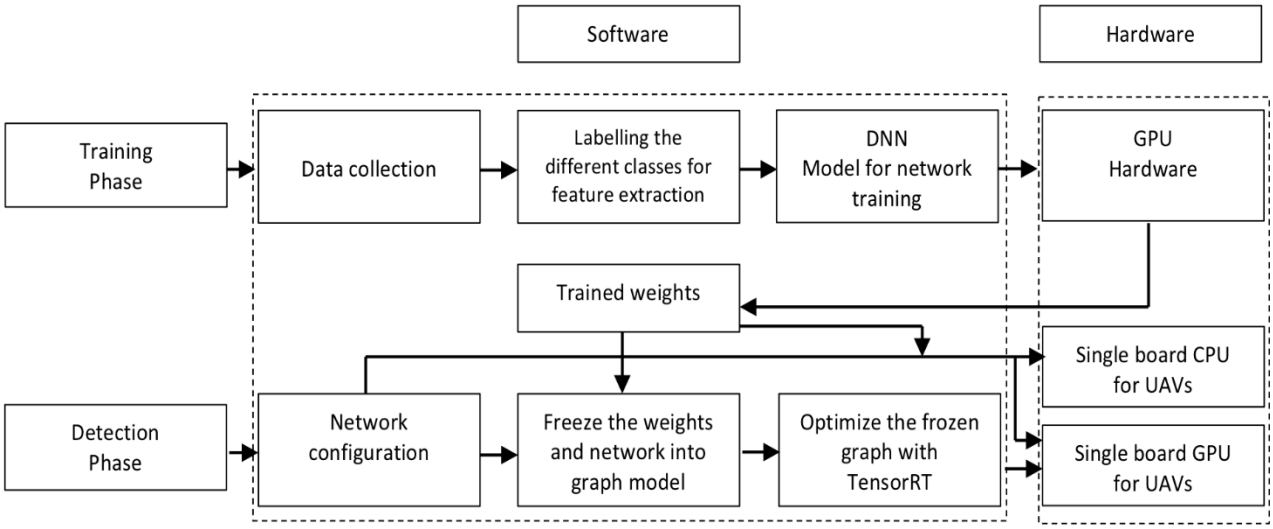

Figure 3 *Overall structure of DL based fault detection model*

3.1.2 Unsupervised ML Framework

During inspection for faults detection, drones will also perform anomaly detection by using variational autoencoders (D4S_TNA_REQ_0610) and/or Generative Adversarial Networks (D4S_TNA_REQ_600). Choosing one model from these unsupervised ML model is based on available dataset. GANs performs better as compared to VAEs but it requires a lot of data and tuning. We can use VAE-GAN a hybrid model which improve sample quality, diversity, stability, and representation learning e.g. Adversarial Autoencoder (AAE). AAE performs better in semi-supervised classification, disentangling style and content of images, unsupervised clustering, dimensionality reduction and data visualization. The software will be developed and tested under the WP4.

3.2 Robot operating system (ROS/ROS2)

The Robot Operating System (ROS) is a flexible, open source framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. ROS runs on Linux and provides communication solution between distributed nodes via the anonymous publish/subscribe mechanism. It will be deployed on the drones and software for drones' navigation and localization should be developed. Also, the drones should be able to detect unexpected obstacles in real-time and avoid them (D4S_FUN_REQ_0160). ROS nodes will receive instructions from Mission planning services and will send position and orientation information back to the cloud. ROS2 solutions will be implemented as the software evolves. For communication between ROS on the drone and low-level autopilot PX4, mavros ROS package will be used. It enables communication using MAVLink protocol and can be used to communicate with any MAVLink enabled autopilot. For ROS2 communication with PX4 there is a PX4-FastRTPS bridge.

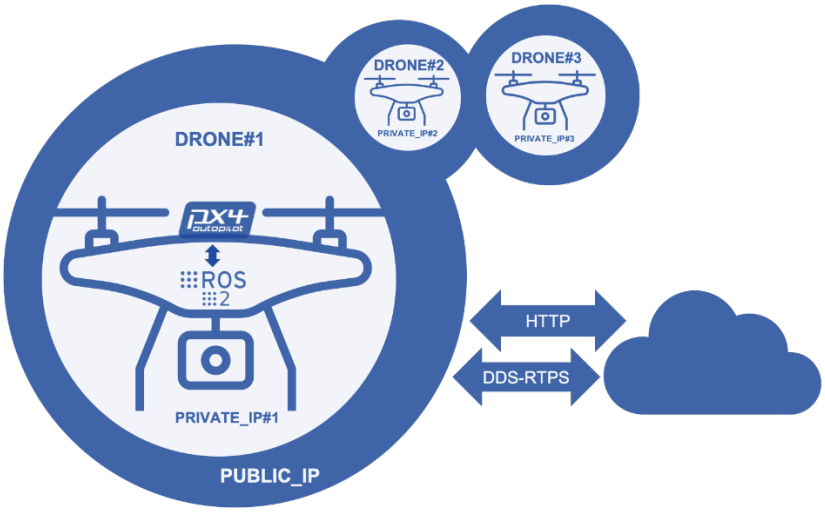

Figure 4 *Drone software architecture and cloud communication*

4 Communication Network Design

While designing a communication network there are more than a few challenges. To achieve autonomous flying, extraordinary software algorithms and control solutions have to be developed. Also, without stable communication, messages could not be transmitted. For that reason, it is crucial to develop a communication network ready to support challenges of autonomous inspection. On the cloud, services must have a reliable way of communication between them and with database. User interacts with the system through the web interface, so communication between the web interface and services is established. Services also need established connection with drones in order to send instructions and receive data (D4S_FUN_REQ_0170).

Depending on the nature of each service, they use synchronous or asynchronous messaging protocol for communication. HTTP is a synchronous protocol. The client sends a request and waits for a response from the

service. The key point here is that the protocol (HTTP/HTTPS) is synchronous and the client code can only continue its task when it receives the HTTP server response. It seems like a suitable protocol for communication between services and web interface. For asynchronous communication, the DDS-RTPS will be investigated. ROS2 is a second version of ROS and it is built on top of DDS-RTPS as its middleware. ROS2 is still evolving, but because of its high potential it will be used as the drone's operating system. DDS provides a publish-subscribe transport which is similar to ROS's publish-subscribe transport. DDS has a request-response style transport, which would be like ROS's service system. This would provide a communication with cloud services since DDS API and libraries are available for different programming languages.

Drones should be connected to the internet using 4G/5G. The connection will be based on IPv6 protocol and IPsec standard of protocols. In the future, IPv6 protocol will assure unique IP address for each drone. During the development and testing we will consider tunnelling the IPv6 connection through a Wireguard VPN. Wireguard VPN will serve as a temporary solution in order to identify drones. To establish the connection between services and ROS running on the drone, a suitable bridge server between services' message protocol and type and ROS message protocol and type should be implemented. In the case of ROS2 DDS will be used (D4S_DES_REQ_0670).

The drones should also be able to communicate between each other. Drone-to-drone communication will be established using wireless mesh networking communication standards and drone-to-ground communication using long-range wireless connection.

Further and more detailed network communication solutions will be specified, described, and tested in WP5.