•

D6.2 Integrated IoT-NGIN platform & laboratory testing results

WORKPACKAGE	WP6	PROGRAMME IDENTIFIER	H2020-ICT-2020-1
DOCUMENT	D6.2	GRANT AGREEMENT ID	957246
REVISION	V1.0	START DATE OF THE PROJECT	01/10/2020
DELIVERY DATE	30/11/2022	DURATION	3 YEARS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), NETCOMPANY-INTRASOFT SA (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Robert Bosch Espana Fabrica Aranjuez SA (BOSCHN), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), ENTERSOFT SA (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelixis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP6
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/11/2022
ACTUAL DATE OF DELIVERY	05/12/2022
DOCUMENT TITLE	Integrated IoT-NGIN platform & laboratory testing results
AUTHOR(S)	A. Gonos (OPT), A. Voulkidis (SYN), T. Velivassaki (SYN), K. Railis (SYN), Ch. Betzelos (SYN), A. Zalonis (INTRA), D. Skias (INTRA), F. Williams (EDD), F. Maier (EDD), T. Ensezgin (EDD), A. Corsi (ENG), F E. Mancinelli (EMOT), G. Tribolati (EMOT), J. Gorroñogoitia (ATOS), J. Klimt (RWTH), J. Costa Requena (CMC), M. Montagud (I2CAT), H. Rahich (SU), A. Su (SU), H. Kinnunen (ABB), Y. Kortesniemi (AALTO), N. Akmaikin (FVH)
REVIEWER(S)	T. Velivassaki (SYN), D. Skias (INTRA)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	Integration, deployment, cloud native, use case logic, architecture, validation, 5G enhancements' validation

Document History

Version	Date	Contributor(s)	Description
V0.1	12/7/2022	EDD	ToC and content
V0.2	31/8/2022	EDD	Updated ToC for EDD's test results Integrated text of CMC and RWTH into chapter 4
V0.3	15/09/2022	EDD	Text editing
V0.4	17/10/2022	CMC	Remaining comments addressed
V0.5	14/11/2022	OPT	Updates on application-specific implementation
V0.6	23/11/2022	INTRA, SYN, OPT, ENG, ATOS, EMOT	Contributions on CI/CD updates, Integration aspects, application-specific components, validation tests.
V0.7	24/11/2022	SU, EDD	Contributions on validation tests.
V0.8	28/11/2022	OPT, INTRA, SYN, ABB, ENG, EMOT	Updates in the integrated framework and instantiation in LLs.
V0.9	30/11/2022	OPT, INTRA, SYN, SU, I2CAT, FVH, ABB	Updates in configuration & deployment, LL implementation and validation; Document consolidation; Ready for peer review
V0.9.1	02/12/2022	INTRA	Peer review
V0.9.2	02/12/2022	SYN	Peer review
V0.99	05/12/2022	OPT, SYN	Integration of peer review comments; Document finalization
V1.0	05/12/2022	CAP, OPT	Quality check; Final version

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

AGLV	Automated Guided Land Vehicle
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CB	Context Broker
CI/CD	Continuous Integration & Continuous Delivery/Continuous Deployment
CPO	Charging Point Operator
D2D	Device-to-Device
DevSecOps	Development plus Security plus Operations
DIB	Decentralised Interledger Bridge
DL	Downlink
DLT	Distributed Ledger Technologies
DR	Demand Response
DS-TT	Device Side TSN Translator
DSO	Distribution System Operators
DT	Digital Twin
EG	Electric Grid
eMBB	enhanced Mobile Broadband
EV	Electric Vehicle
EVCS	Electric Vehicle Charging Station
GPS	Global Positioning System
GPSI	General Public Subscription Identifier
GPU	Graphics Processing Unit
gPTP	generalised Precision Time Protocol
GUI	Graphical User interface
HTTP	Hypertext Transfer Protocol
IDAC	IoT Device Access Control

IDI	IoT Device Indexing
IoT	Internet of Things
K8s	Kubernetes
LAN	Local Area Networks
LL	Living Lab
MAD	Malicious Attack Detector
ML	Machine Learning
MLaaS	Machine Learning as a Service
MQTT	Message Queuing Telemetry Transport
MTD	Moving Target Defences
NW-TT	Network TSN Translator
OBD	On-board Diagnostic Device
OC	Out-Of-Coverage
OPC UA	Open Platform Communications Unified Architecture
P2P	Peer-to-peer
PC	Personal Computer
PMU	Phasor Measurement Unit
PQA	Power Quality Analyser
PTPv2	Precision Time Protocol version 2
RAM	Random-Access Memory
RFID	Radio Frequency Identification
RL	Reinforcement Learning
RTSP	Real Time Streaming Protocol
SAST	Static Application Security Testing
SCR	Self-consumption ratio
SEAL	Service Enabler Architecture Layer for Verticals
SSI	Self-Sovereign Identities
SSR	Self-sufficiency ratio
TCP	Transmission Control Protocol
TSN	Time-Sensitive Networking

UC	Use Case
UDP	User datagram protocol
UE	User Equipment
UoP	Urban Open Platform
UPF	User Plane Function
URL	Uniform Resource Locator
URLLC	Ultra Reliable Low Latency Communications
vCPU	Virtual Central Processing Unit
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
WG	Working Group
WP	Work Package
WS	WebSocket
XAI	explainable AI

Executive Summary

The main purpose of this document is threefold. Firstly, this document intends to provide the integration status of the IoT-NGIN framework. This corresponds both to the development, deployment and integration status of the project's technical activities at the time of writing this deliverable, but also to next steps of the integration plan for each IoT-NGIN component for the following months. In addition, we have elaborated on integration considerations following a holistic approach and also taking into account the Living Lab integration and deployment requirements.

Secondly, this document provides a thorough analysis of the application logic enhancements that were introduced in each LL Use Case. Moreover, it describes the logical connection of these modules within each LL UC specific IoT-NGIN architecture.

Lastly, this document also encapsulates an extensive set of tests that have been conducted so far and focusing on the Federated Communication Layer of the IoT-NGIN architecture that consists of 5G networking software and tools. This includes the description of the implementation and the testing conducted for evaluating the performance of 5G as the communication network for the IoT-NGIN Use Cases but also the evaluation of the IoT-NGIN 5G network related components on specific use cases.

1 Introduction

The IoT-NGIN framework is built upon the IoT-NGIN meta-architecture that has been defined in the context of D1.2 [1] and updated in D1.3 [2]. The suggested meta-architecture dictates the necessity to build the corresponding IoT-NGIN framework that will materialise IoT-NGIN architecture instantiations that would address and comply to the project's Living Lab needs. This requires the development of a large number of components and sub-components (modules) covering the complete vertical of the IoT-NGIN functional stack. Then, these components have to be tested, both for terms of functionality, communication and performance and then incorporated into the integrated IoT-NGIN platform. Taking into consideration also the added complexity due to the enhancements introduced for the application logic of the IoT-NGIN Living Lab Use Cases, one can easily argue that this is not a trivial task, but a rather challenging joint effort.

D6.1 [3] elaborates on the IoT-NGIN platform architecture, derived by the IoT-NGIN meta-architecture and providing a detailed component-level view of the IoT-NGIN platform. The integration and testing infrastructure, as well as the integration and testing methodology of the project have been presented. This includes the integration guidelines and the DevSecOps practices and tools adopted by the project.

The aim of this deliverable, entitled "*Integrated IoT-NGIN platform & laboratory testing results*", is twofold. On one hand, it presents the integration status and the associated evaluation and performance tests that will lead to the finalised IoT-NGIN platform which will be released at M30. On the other hand, it presents in detail the advancements on the design and implementation of the application logic enhancements that will complement the project's technical developments in satisfying the UCs' requirements.

Therefore, we proceed with the necessary integration activities for building the integrated IoT-NGIN framework. More specifically, we elaborate on the IoT-NGIN CI/CD enhancements that have been introduced in order to provide a fully automated and at the same time Cloud Native compliant methodology that would be followed by the IoT-NGIN developers to accelerate the development and the integration process. A relevant guide presenting the integration steps that developers should follow in order to utilise the configured IoT-NGIN CI/CD pipeline is also included. Furthermore, the integration status of the IoT-NGIN developed technical solutions along with high-level interactions among IoT-NGIN's components.

In addition, the IoT-NGIN platform has to incorporate, including designing, developing and integrating, additional application enhancements in order to support the specific objectives that are addressed by each Living Lab. The respective application enhancements accompanied with the associated UCs' architecture instantiation figures are presented in detail in the present document.

Finally, within this deliverable an extensive set of tests that have been conducted so far and focusing on the Federated Communication Layer of the IoT-NGIN architecture that consists of 5G networking software and tools and includes the description of the implementation and the tests conducted for evaluating the performance of the 5G as the communication network for the IoT-NGIN Use Cases but also the evaluation of the IoT-NGIN 5G network related components on specific use cases are described thoroughly.

In the upcoming D6.3, entitled "*Interoperable IoT-NGIN meta-architecture & laboratory evaluation*" due in M33, we will describe the finalised integrated IoT-NGIN framework and present extensive integration and performance related tests of the IoT-NGIN implemented technical solutions.

1.1 Intended Audience

The intended audience of this document is primarily the consortium of the IoT-NGIN project, as well as the partners that were introduced through the project's Open Calls (#1 and #2). More specifically, this deliverable presents to the technical partners of the consortium, the integration and testing status of the IoT-NGIN platform and the integration steps that need to be adopted in order to enable the configured CI/CD pipeline. In addition, this document would facilitate the Living Lab owners to better comprehend the IoT-NGIN architecture instantiations that will be deployed in their premises and intend to support the particular UCs narrative.

Finally, software developers, engineers and system integrators might also find useful the present document as we present a detailed large-scale integration process that adheres on state-of-the-art Cloud Native approaches and covers the complete integration and testing spectrum, including practical examples and technical approaches. So, this document might be of interest to third party developers both for understanding the configuration and deployment of the IoT-NGIN framework, but also for taking a tour to our generic automated integration and deployment process which could be applied in other software projects as well.

1.2 Relations to other activities

Integration and testing activities act as the backbone of the system development process and thus of the integrated IoT-NGIN framework. This document is strongly connected with all the technical WPs (2-6) and also related to WP7 that consists of the Living Lab IoT-NGIN platform deployment and the subsequent validation work that need to be performed, as dictated by the IoT-NGIN specific Use Cases. Finally, it is also related with WP8 and WP9 in the context of impact creation and Open Calls, respectively.

1.3 Document overview

The present document provides insights on the integration and testing activities of the IoT-NGIN framework. More specifically, the document is organised as follows.

Section 1 provides an introduction on the objectives that are tackled by this deliverable and the relevant activities.

Section 2 presents the integration status of the IoT-NGIN components, the advancements with respect to the project's CI/CD pipeline and an IoT-NGIN developer's guide describing the required integration steps that need to be taken to deploy and integrate the developed technical solutions. In addition, the IoT-NGIN integrated framework is presented focusing in particular on the IoT-Edge-Cloud computing integration and on the integration of the ML services/modes to the MLaaS component. Finally, it presents the configuration and deployment of the IoT-NGIN platform elaborating on Cloud Native technologies and tools that will be adopted in order to facilitate this process. (e.g., Helm Charts).

Section 3 describes the IoT-NGIN application logic enhancements design and development status and also the relevant architecture instantiation figures that support each of the Living Lab's Use Cases.

Section 4 provides information on the implementation and performance testing of 5G as the communications network for IoT-NGIN use cases.

Section 5 presents information on the implementation and the evaluation of the IoT-NGIN components that comprise the Federated Communications layer of the IoT-NGIN framework

Section 6 draws conclusions for this report.

In addition, Annex 1 provides additional parameters for RWTH's EdgePMU, which calculates phasors by utilizing the edge cloud.

Annex 2 presents detailed information on the development and integration of each technical solution implemented in the context of the project in a tabulated format.

Finally, Annex 3, provides indicative HELM-charts of IoT-NGIN components showcasing the utilization of HELM in view of the holistic deployment approach that will be adopted by the IoT-NGIN.

2 Integration of IoT-NGIN Components

The integration of the IoT-NGIN framework is based on the principle of adopting the cloud-native computing paradigm, which provides the benefits of Increased efficiency, reduced costs and ensured availability and scalability. The Cloud Native Computing Foundation (CNCF) provides the *Cloud Native Definition v1.0* [4]:

"*Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.*

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."

Aligned to this definition, IoT-NGIN builds a framework that is based on containerised microservices, which are built, tested, deployed and operated in the cloud, focusing on the automation of these processes to the extent possible.

The IoT-NGIN components are developed as stateless microservices which provide functionality almost independent from other microservices and run as containers or unikernels in the IoT-NGIN Kubernetes (K8s) cluster, running on OneLab facilities [3]. The deployment of such containers relies on Helm charts [5], which provide the ability to provide, share, and use the containers built for Kubernetes, alleviating from the burden to manually author a set of configuration (yaml) files for the Kubernetes *Deployments* [6], *Secrets* [7], *ConfigMaps* [8], etc. In addition, HELM charts offer additional flexibility on deploying the various IoT-NGIN platform instantiations on the project's Living Labs' infrastructures.

On the other hand, IoT-NGIN adopts a DevSecOps approach, based on GitLab CI/CD pipeline, as described in D6.1 [3]. Moreover, this pipeline is used to safely deploy the IoT-NGIN components to the IoT-NGIN Kubernetes cluster. IoT-NGIN has already delivered stable versions of individual components in the project's GitLab group [9].

In the following, thorough technical guidance is provided to developers on deploying components' source code already uploaded on GitLab into IoT-NGIN's Kubernetes cluster via the GitLab Agent.

Moreover, the updated IoT-NGIN architecture is provided in subsection 2.2, as a guide for the integration of the individual components, discussing the status of the integration, as well as the approach of IoT-NGIN towards the adoption of edge computing and delivery of ML services, which involve interaction of cross-WP components, while addressing IoT, edge and cloud integration, Digital Twins and MLOps.

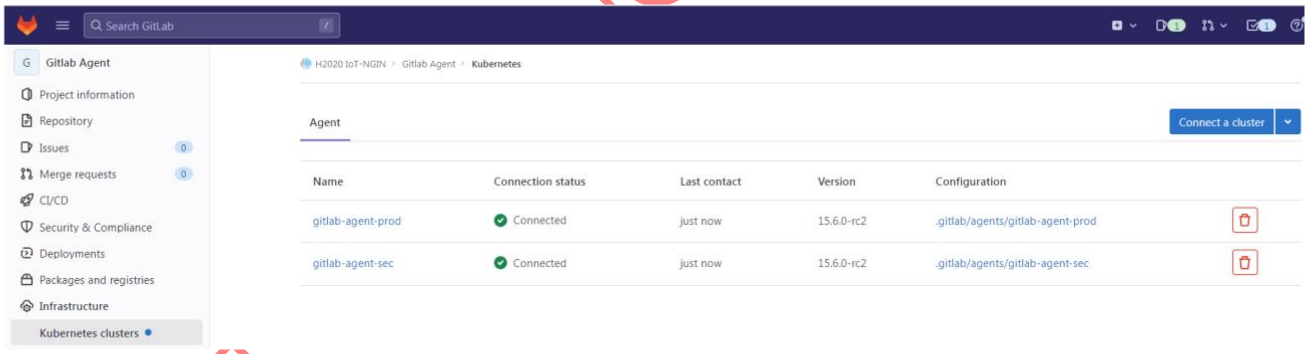
2.1 Continuous Integration – Continuous Deployment

The IoT-NGIN project has adopted DevSecOps practices which support the project's development and integration activities. DevSecOps practices are realised through the CI/CD pipeline that has been configured for the IoT-NGIN project. At the same time the various technologies adopted to support the CI/CD pipeline adhere to the Cloud Native paradigm. IoT-NGIN CI/CD is realised through GitLab which is an Open source DevSecOps platform. In the context of this document, we elaborate on the functional enhancements that we have incorporated into the already setup CI/CD workflow. In addition, here we provide technical information that pertains to the utilization of the IoT-NGIN CI/CD pipeline by the IoT-NGIN developers.

2.1.1 GitLab Agent

IoT-NGIN, adhering to the Cloud Native approach, provides automated delivery procedures realizing the CD part of the CI/CD pipeline through GitOps. GitOps is a deployment methodology centralized around a Git repository (GitLab) that advocates using versioned files in source control repositories to define and manage the underlying infrastructure. This agent based GitOps methodology for continuous deployment (CD) refers to running an active component within the infrastructure which handles the components' deployments and is achieved with the GitLab agent. An agent is installed and deployed in the Kubernetes cluster and is used as a communication bridge between the source code in GitLab and the cluster. The agent is authorised to access all GitLab sub-groups and projects of the H2020 IoT-NGIN group.

IoT-NGIN project's cloud infrastructure consists of two clusters, the main one and the auxiliary one which incorporates a dedicated GPU that facilitates ML pipelines (through Kubeflow [10]) facilitating the development and integration activities of WP3. Thus, one GitLab Agent is deployed in each IoT-NGIN cluster, as illustrated in the Figure 1 below. The *GitLab-agent-prod* is deployed within the IoT-NGIN main cluster, while *the GitLab-agent-sec* is deployed within the auxiliary (secondary cluster).



Name	Connection status	Last contact	Version	Configuration
gitlab-agent-prod	Connected	just now	15.6.0-rc2	.gitlab/agents/gitlab-agent-prod
gitlab-agent-sec	Connected	just now	15.6.0-rc2	.gitlab/agents/gitlab-agent-sec

Figure 1 – IoT-NGIN Gitlab agents' configuration.

2.1.2 Integration Steps

The IoT-NGIN project, utilizing GitLab CI/CD, allows the developer to define and deploy jobs in order to build Docker images and publish them to the project's container registry in docker hub [11]. The basic steps to enable GitLab CI/CD on an IoT-NGIN GitLab project and a sample pipeline template are described below. In addition, the required configuration files (.yaml files) that pertain to the docker and Kubernetes are also provided.

In order to demonstrate the utilization of the CI/CD pipeline that has been configured for the project's needs, we apply and configure the relevant files that would enable the CI/CD pipeline for an implemented IoT-NGIN component, which, in this case, is the IoT Device

Indexing (IDI) component. Its source code is available in the IoT-NGIN project's GitLab under the "*enhancing_iot_tactile_contextual_sensing_actuating*" sub-group.

IDI consists of a set of sub-components (modules). The following example demonstrates the steps that developers should follow in order to set up the CI/CD pipeline for the "*Historic Data Registry*" module.

2.1.2.1 Dockerfile

First, we should create a Dockerfile file in the root of the repository.

```
FROM node:19
WORKDIR /app
ADD . /app
RUN npm install
EXPOSE 3000
CMD npm start
```

2.1.2.2 Kubernetes Manifest

We should also create the Kubernetes.*yaml* configuration files in the config/*k8s* directory. For instance, here we have configured a deployment, a service, a secret, a persistent volume claim and an Ingress.

It is important to specify the namespace as a work package and the exact image name as dictated by the IoT-NGIN architecture. It is also recommended to deploy to k8s cluster only Semantic Versioning (SemVer) tags.

2.1.2.2.1 Deployment

As all IoT-NGIN docker images will be located in the project's DockerHub registry, we have configured a *kubernetes.io/dockerconfigjson* type Secret in the Kubernetes cluster, reflected in all namespaces, with name *public-regcred*. This ***public-regcred*** secret must be specified as ***imagePullSecret*** in every deployment that uses these private images.

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: device-indexing-registry
    labels:
      app: device-indexing-registry
spec:
  selector:
    matchLabels:
      app: device-indexing-registry
  replicas: 1
  template:
    metadata:
      labels:
        app: device-indexing-registry
    spec:
      imagePullSecrets:
      - name: public-regcred
      containers:
```

```
- name: device-indexing-registry
  image: iotngin/device_indexing_registry:v0.1.3
  env:
    - name: MONGO_HOST
      value: device-indexing-registry-db
    - name: MONGO_INITDB_ROOT_USERNAME
      valueFrom:
        secretKeyRef:
          name: device-indexing-registry-db-secret
          key: MONGO_INITDB_ROOT_USERNAME
    - name: MONGO_INITDB_ROOT_PASSWORD
      valueFrom:
        secretKeyRef:
          name: device-indexing-registry-db-secret
          key: MONGO_INITDB_ROOT_PASSWORD
    - name: MONGO_INITDB_DATABASE
      valueFrom:
        secretKeyRef:
          name: device-indexing-registry-db-secret
          key: MONGO_INITDB_DATABASE
    - name: MONGO_PORT
      value: "27017"
    - name: FLASK_APP
      value: ./main.py
  ports:
   - containerPort: 5000
```

2.1.2.2.2 Service

```
apiVersion: v1
kind: Service
metadata:
  name: device-indexing-registry
  labels:
    app: device-indexing-registry
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 5000
    protocol: TCP
  selector:
    app: device-indexing-registry
```

2.1.2.2.3 Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: device-indexing-registry-db-secret
  labels:
    app: device-indexing-registry-db-secret
type: Opaque
stringData:
  MONGO_INITDB_ROOT_USERNAME: mongoadmin
  MONGO_INITDB_ROOT_PASSWORD: secret
  MONGO_INITDB_DATABASE: device-indexing-registry
```

2.1.2.2.4 PersistentVolumeClaim

```
apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: device-indexing-registry-pv-claim

spec:

  accessModes:

    - ReadWriteOnce

  resources:

    requests:

      storage: 5Gi
```

2.1.2.2.5 Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: device-indexing-registry-ingress
spec:
  ingressClassName: nginx
  tls:
  - hosts:
      - device-indexing-registry.apps.kf.iot-ngin.onelab.eu
  rules:
  - host: device-indexing-registry.apps.kf.iot-ngin.onelab.eu
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: device-indexing-registry
            port:
              number: 80
```

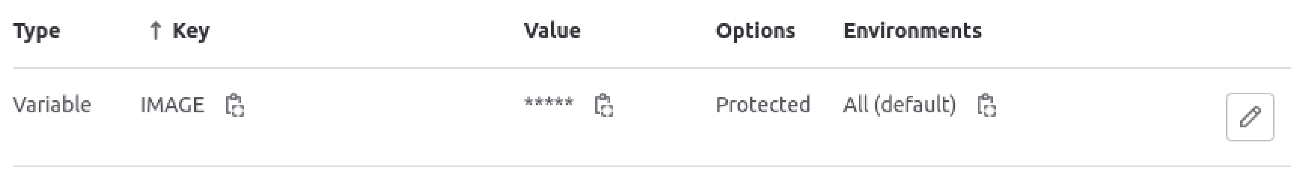
2.1.2.3 GitLab CI/CD Pipeline

In order to enable the IoT-NGIN GitLab CI/CD pipeline, we need to create the ***.gitlab-ci.yml*** file at the root of the repository.
We have configured the Docker-in-Docker with TLS enabled in Kubernetes to use Docker in our CI/CD jobs, so we must specify the appropriate Docker variables and the *docker:20.10.16-dind* service.

Since IoT-NGIN materialises the DevSecOps development practice and thus security initiatives are integrated in the CI/CD pipeline, we have enabled the GitLab Static Application Security Testing (SAST) by navigating to **Security & Compliance > Configuration**, to analyse our source code for known vulnerabilities. With the *SCAN_KUBERNETES_MANIFESTS* variable, we enable the Kubesec analyzer in SAST.

In addition to the predefined Group Variables (*DOCKER_LOGIN*, *DOCKER_PASSWORD*), we have also specified the *IMAGE* repository variable as shown in Figure 2, by navigating to *Settings* > *CI/CD* > *Variables* in the repository.

With the *kubectl config use-context h2020 IoT-NGIN/gitlab-agent-prod:gitlab-agent-prod* or *h2020 IoT-NGIN/gitlab-agent-sec:gitlab-agent-sec*, we select the GitLab agent's Kubernetes context to run the Kubernetes API commands for the deploy stage.



Type	↑ Key	Value	Options	Environments
Variable	IMAGE	*****	Protected	All (default)

Figure 2 – GitLab CI/CD Repository Variables.

```
image: docker:19.03.13

include:
  - template: Security/SAST.gitlab-ci.yml

variables:
  DOCKER_TLS_CERTDIR: "/certs"
  SCAN_KUBERNETES_MANIFESTS: "true"

services:
  - docker:19.03.13-dind

stages:
  - sast
  - test
  - build
  - deploy

test:
  stage: test
  image: python:3.8.10-slim
  script:
    - echo "Write your tests here"
  when: always

lint-test:
  stage: test
  script:
    - echo "Linting code... This will take about 5 seconds."
    - sleep 5
    - echo "No lint issues found."

build-main:
  stage: build
  before_script:
    - echo "$DOCKER_PASSWORD" | docker login --username $DOCKER_LOGIN --password-stdin
  script:
    - docker build -t $IMAGE:latest .
    - docker image push $IMAGE:latest
  only:
    - main

build-tags:
  stage: build
  before_script:
    - echo "$DOCKER_PASSWORD" | docker login --username $DOCKER_LOGIN --password-stdin
  script:
    - docker build -t $IMAGE:$CI_COMMIT_TAG .
    - docker image push $IMAGE:$CI_COMMIT_TAG
  only:
    - tags

deploy-prod:
  stage: deploy
  image:
    name: bitnami/kubectl:1.22.15
    entrypoint: ['']
  script:
    # - sed -i "s+\b$IMAGE:[^ ]*+$IMAGE:$CI_COMMIT_TAG+" config/k8s/combo.yaml
    - kubectl config get-contexts
    - kubectl config use-context h2020-iot-ngin/gitlab-agent:gitlab-agent-prod
    - kubectl set image deployment/device-indexing-registry device-indexing-registry=$IMAGE:$CI_COMMIT_TAG -n iot-
ngin-wp4
  only:
    - tags

# deploy-sec:
#   stage: deploy
#   image:
#     name: bitnami/kubectl:1.22.15
#     entrypoint: ['']
#   script:
#     # - sed -i "s+\b$IMAGE:[^ ]*+$IMAGE:$CI_COMMIT_TAG+" config/k8s/combo.yaml
#     - kubectl config get-contexts
#     - kubectl config use-context h2020-iot-ngin/gitlab-agent:gitlab-agent-sec
#     - kubectl set image deployment/device-indexing-registry device-indexing-registry=$IMAGE:$CI_COMMIT_TAG -n iot-
ngin-wp4
#   only:
#     - tags
```

We have configured three stages in the CI/CD pipeline:

- Build - Push
- Test (with SAST)
- Deploy

When source code is **pushed to** the **main** branch by the developer, the first two stages are triggered as shown in Figure 3. The *build-main* job also pushes the $IMAGE *(iotngin/device_indexing_registry)* to DockerHub with a tag *latest*.

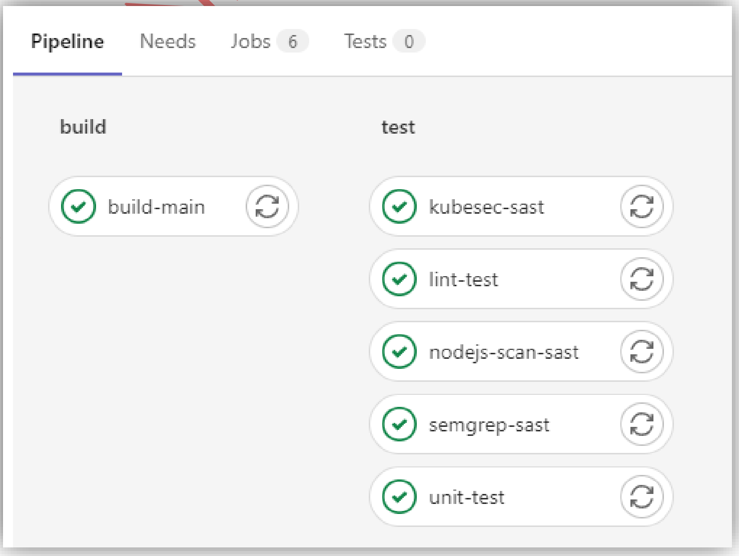


Figure 3 – CI/CD Pipeline when push to main.

When **a tag is pushed**, all stages are triggered, without the SAST operations, as shown in Figure 4. The *build-tags* job also pushes the $IMAGE (iotngin/device_indexing_registry*)* to DockerHub with tags: **$CI_COMMIT_TAG,** which is the *commit* tag name.

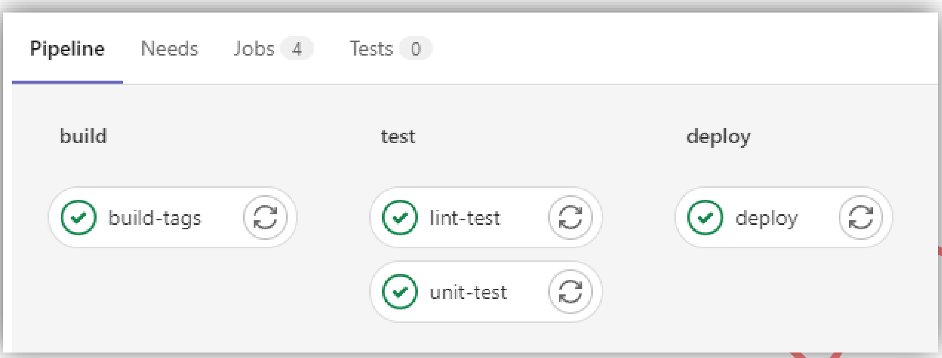


Figure 4 – CI/CD Pipeline when push a tag.

Finally, once the CI/CD pipeline execution is completed successfully, then the component is deployed in the cluster. If any step of the pipeline is not successful, then manual corrections should be applied by the developer in order to correct the issue. Once everything is resolved, the CI/CD pipeline should terminate normally by deploying the component to the cluster.

2.2 The IoT-NGIN integrated framework

The IoT-NGIN components are being developed in the context of providing functionality in the framework of the functional groups (elements) of the IoT-NGIN meta-architecture [1]. The development of this functionality is materialised in the component-level architecture, presented in D6.1, in which each component represents an IoT-NGIN micro-service.

The IoT-NGIN framework architecture has been updated, following the developments and progress of the work within the project so far, which have led to more comprehensive definition of the scope of each component, fine-graining the interactions among them and the project boundaries. The updated architecture is presented in Figure 5, in which the colours denote different functional groups, as described in D6.1.

The main updates compared to the first version concern all of the five functional layers that comprise the IoT-NGIN architecture, namely the Federated Communications layer, the Micro-services and Virtual Network Functions (VNFs), the Federated Data Sovereignty, the Federation of Big Data Analytics & ML and the Human-Centred Augmented Reality Tactile IoT. More specifically, the updates relate to the following.

The components providing 5G enhancements have been made more specific in the architecture diagram and now indicate comprehensively the enhancements and 5G capabilities exposed by the relevant APIs, as well as the way they manage 5G resources. Details can be found in D2.2 [12].

The "IoT Device Discovery" component has been presented in all four variants it is developed in IoT-NGIN, including the Computer Vision, Visual Light Positioning (VLP), Ultra-Wide Band (UWB) and Code Scanning (QR/barcode) variants, as detailed in D4.3 [13].

The "Deep Learning, Reinforcement Learning & Transfer Learning framework" has been split into "Reinforcement Learning framework" and "Online Learning framework", which both connect to the MLaaS framework. The Privacy-Preserving Federated Learning (PPFL) API has been added as an API to consume federated learning services across the different FL frameworks considered in the project. The updates on the project's ML tools are documented in D3.3. [14].

The term "Semantic Twin" has been adopted to align with the definition and scope of the component developed, instead of the term "Meta-Level Digital Twin". The latest updates can be found in D5.4 [15].

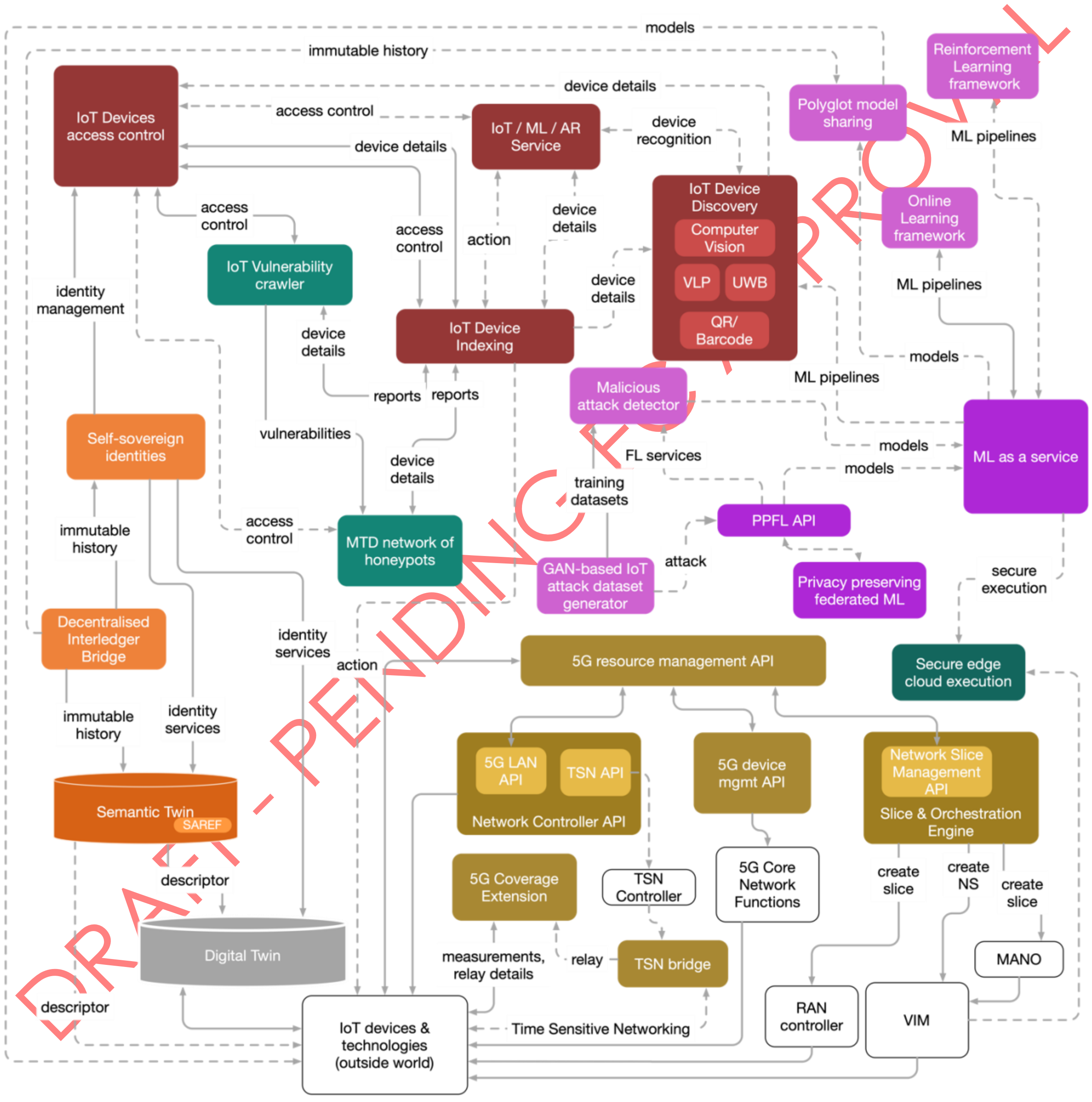


Figure 5 – The IoT-NGIN logical architecture.

The "IoT Device Indexing" and "IoT Device Access Control" are treated as parts of a generic Digital Twin solution, providing data communication and persistence. The IoT-NGIN platform could also connect to third-party Digital Twin solutions, described in the "Semantic Twin".

Moreover, the ML model storage, serving and sharing have been concretised as operations of the MLaaS platform, while model translation and sharing is provided by the "Polyglot Model Sharing" component.

The functionality and inter-operation of the IoT Vulnerability Crawler and the MTD Network of Honeypots have been concretely defined and fine-grained.

Moreover, Figure 5 indicates the updated interactions among the individual components, as well as the status of integration among them. Integration includes a number of steps, starting from the specification of interfaces, the development of the respective endpoints and potential logic, their testing and, finally, their automated deployment. Hence, dashed lines in the figure indicate that integration activities have been at least at the stage of development/testing, but are still in progress, while continuous lines indicate that integration among the interacting components has been performed at a mature level with interfaces working or even with automated configuration and deployment completed.

2.2.1 IoT-Edge-Cloud computing integration

The cloud computing revolution has democratised the development of use of technology, offering both business and end-users the illusion of unlimited resources availability and always-on experience of mobile and web services. Cloud computing has enabled the emergence of wide range of applications, supported by the SaaS, PaaS or IaaS model, delivering complex functionality to the end user, without necessarily requiring powerful end devices.

The centralization of intelligence at remote cloud resources has, however, been an obstacle to real-time applications and services or is critically addressed from the privacy perspective. The advances in IoT have allowed a multitude of diverse devices with varying computational and power resources to generate an overwhelming amount of data. While the analysis of those data can be a real gold mine, data transfer to the cloud and communication of results back to the end device can be too costly to support strict latency requirements of real-time services. Also, the data transfer raises significant security, privacy or legal concern, which have often been a major obstacle to public cloud adoption. Moreover, this remote task offloading comes with increased energy footprint, attributed to both communication costs, as well as increased consumption due to often inefficient management of available resources. These concerns can be placated by bringing computation closer to the data sources and offloading only higher computational and orchestration tasks to the cloud. On the other hand, IoT now avails of different devices or "things" which may support wider range of task execution on the end device or on a device that is close to it, e.g., in the network (LAN).

The next logical step of evolution has thus been edge and fog computing, which enable architectures of distributed computation among a set of nodes close to the end user devices and the cloud infrastructure, yielding decreased latency and increased privacy. Time-critical applications can be executed at the edge devices, while heavier, less time-sensitive computations in more powerful devices located between the edge and the cloud resources, such as an IoT gateway or fog node with LAN-connected processors or within the LAN hardware itself.

IoT-NGIN enables the integration of IoT, edge and cloud computing, through the Digital Twin of the Edge device, following AIOTI's decentralised data space example for AI capability in the Digital Twin, as shown in Figure 6. Based on this paradigm, "*a data space is a trustworthy* *decentralised environment for data sharing*" [16], in which decentralization is realised in two data space layers (processing and data layer) and three concepts (data exchanges, data interoperability and data operations) among organizations.

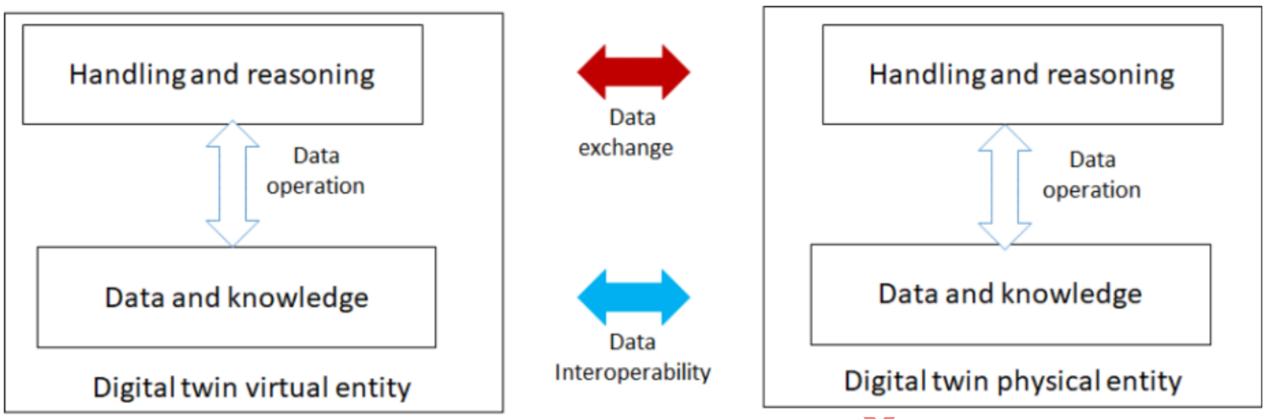


Figure 6 – AIOTI example for AI capability in a digital twin [16].

In IoT-NGIN, the AIOTI example for decentralised data spaces is leveraged through the Digital Twin (DT), which is realised through the IoT Device Indexing (IDI) component that heavily relies on the *FIWARE Orion Context Broker (CB)* [17], extended with the Historic Data Registry and the Inactivity Checker submodules [18] for the "Data and knowledge" part of the Digital Twin. These DT services are protected by IoT-NGIN's IoT Device Access Control (IDAC), which applies access and usage control on device data.

As depicted in Figure 7, this mechanism is used for bi-directional message exchange:

- Data communication from the IoT device to the digital twin and possibly the cloud
- Command communication from a remote service to the digital twin and then to the IoT device, possibly using data that reside at the cloud resources.

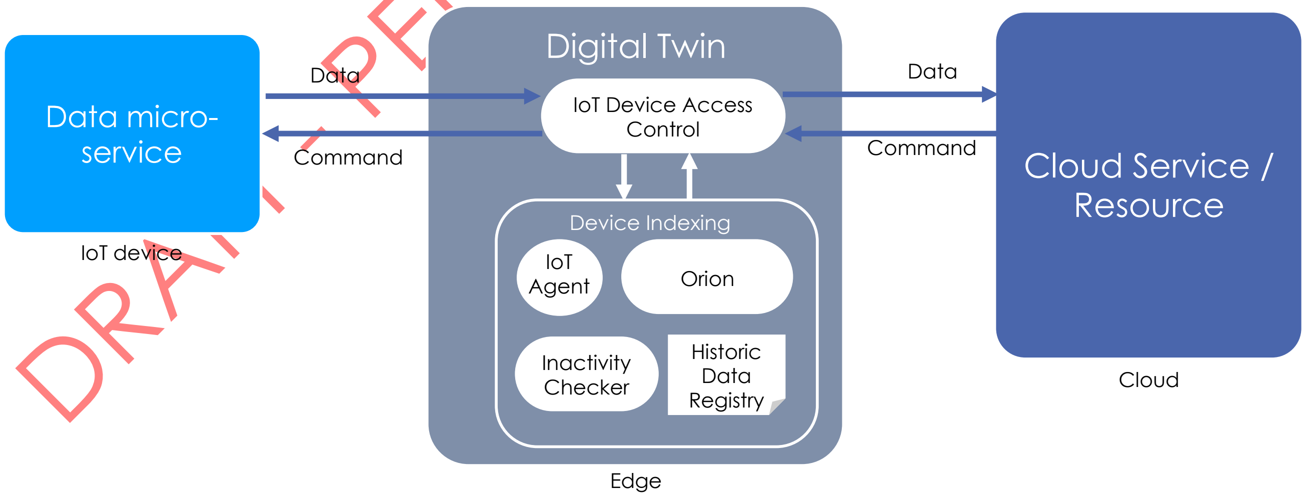


Figure 7 – IoT-edge-cloud integration.

The Data communication constitutes *Northbound Traffic* of the FIWARE Context Broker, with the IoT agent receiving data updates through REST or publish/subscribe services. The communication of commands is treated as Southbound Traffic within the CB.

Assuming that devices are properly registered on the Device Indexing module, data could be communicated, using the services depicted in Table 1. Details on the use of these services is provided in D4.3 [13]. Both communication routes can be protected by the Access Control module, imposing criteria-based control on the usage of the requested resources by updating the relevant headers while issuing requests to IDI services. The criteria for access control could vary across use cases and may be realised as one or more plugins of the Device Access Control module.

Table 1 – Digital Twin services for data and command communication.

Description	From	To	Service
Data communication	IoT Device	Digital Twin	IoT agent - REST */iot/d?k={apikey}&i={device_id}* IoT agent – MQTT */{apikey}/{device_id}/attrs*
Command communication	Digital Twin	IoT Device	Context Broker */v2/op/update*

2.2.2 Integration of ML services to MLaaS

AI is a strategic priority in EU's Digital Decade, while the arrival of 6G networks will enable further advances in AI/ML, such as exploiting data that is local to the 6G sensor by efficiently transporting the (AI/ML) algorithms [19]. Hence, the integration of ML services with the MLaaS platform is a significant part of the IoT-NGIN architecture.

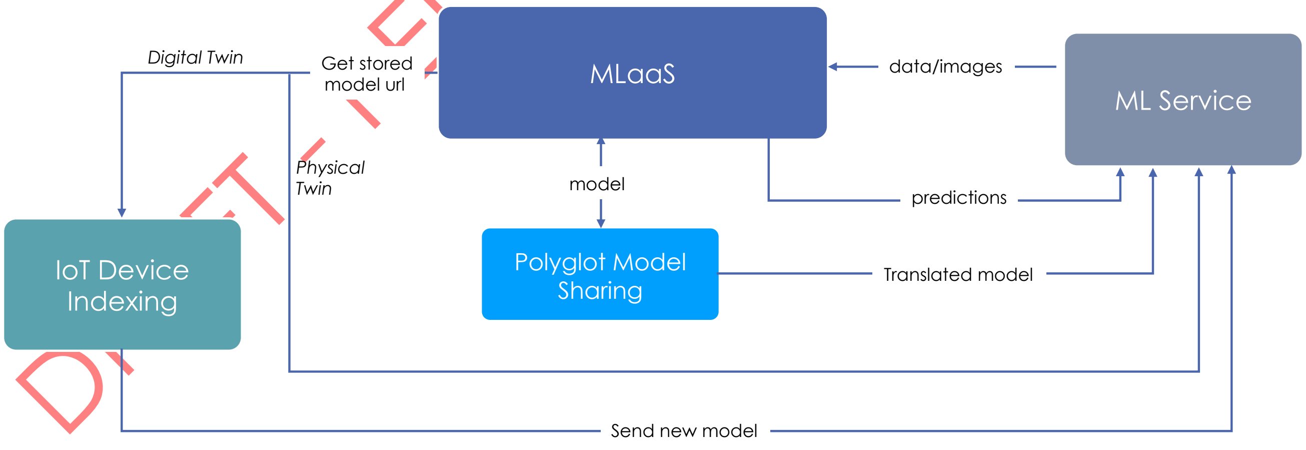


Figure 8 – MLaaS interaction with other ML services.

The full fledge of the ML operations supported by the MLaaS platform are detailed in D3.1 [20], D3.2 [21] and D3.3 [14]. Here, the exploitation of trained ML models through the platform is described. An ML service could use an already trained ML that is stored in MLaaS directly, i.e. by retrieving the model and deploying it locally on the user's premises, or have the model inference being provided as a service, i.e. sending input data to MLaaS and receiving the relevant predictions.

In case of trained ML models, the following options are available for the ML developers to use them in their ML services:

- Through the Digital Twin, retrieving the model and deploying it locally on the IoT device
- Directly retrieving and deploying the ML model locally
- Using the MLaaS model serving component, with the MLaaS performing serverless deployment of model inference for common ML frameworks Scikit-Learn, XGBoost, Tensorflow, PyTorch as well as pluggable custom model runtime.
- Retrieving already trained ML models, stored in MLaaS, but translated in a different ML framework than the one that they have been trained.

These use cases cover a wide range of diverse use cases, across which IoT-NGIN's ML operations can be exploited. The adoption of the cases described above is also explained and discussed in section 3, in the context of each IoT-NGIN Living Lab particular Use Cases.

2.2.3 Configuration & Deployment of the IoT-NGIN platform

In the framework of the IoT-NGIN project we have incorporated development and deployment automations that adhere to the Cloud Native paradigm and are driven via the GitLab CI/CD pipelines. The steps that are required by the developers in order to deploy their implemented technical solutions into the project's cloud infrastructure are described in detail in section 2.1. The IoT-NGIN integration and testing infrastructure is running in OneLab infrastructure and is managed via the Kubernetes container orchestrator. The infrastructure consists, at present, of two different Kubernetes Clusters which serve different purposes. A primary cluster as the production cluster and a secondary cluster for KubeFlow and for all components that require a Graphics Processing Unit (GPU) workload. The clusters' master nodes have 8 vCPU cores, and 16 GB of RAM and slave nodes have 16 vCPU and 32GB of RAM each. The specific number of the slave nodes depends on the workload and on the deployed components' hardware requirements. Since the IoT-NGIN integration and Testing infrastructure is managed by Kubernetes, we are able to scale out (Horizontally) and add as many additional VM nodes as necessary. In addition to the aforementioned Kubernetes clusters, the project also utilises a 5G mobile network established in Eurolab (Ericsson's laboratory infrastructure). IoT-NGIN components that adhere to the WP2 development activities (e.g., M2M, MCM, 5G resource management etc.) are deployed and integrated there.

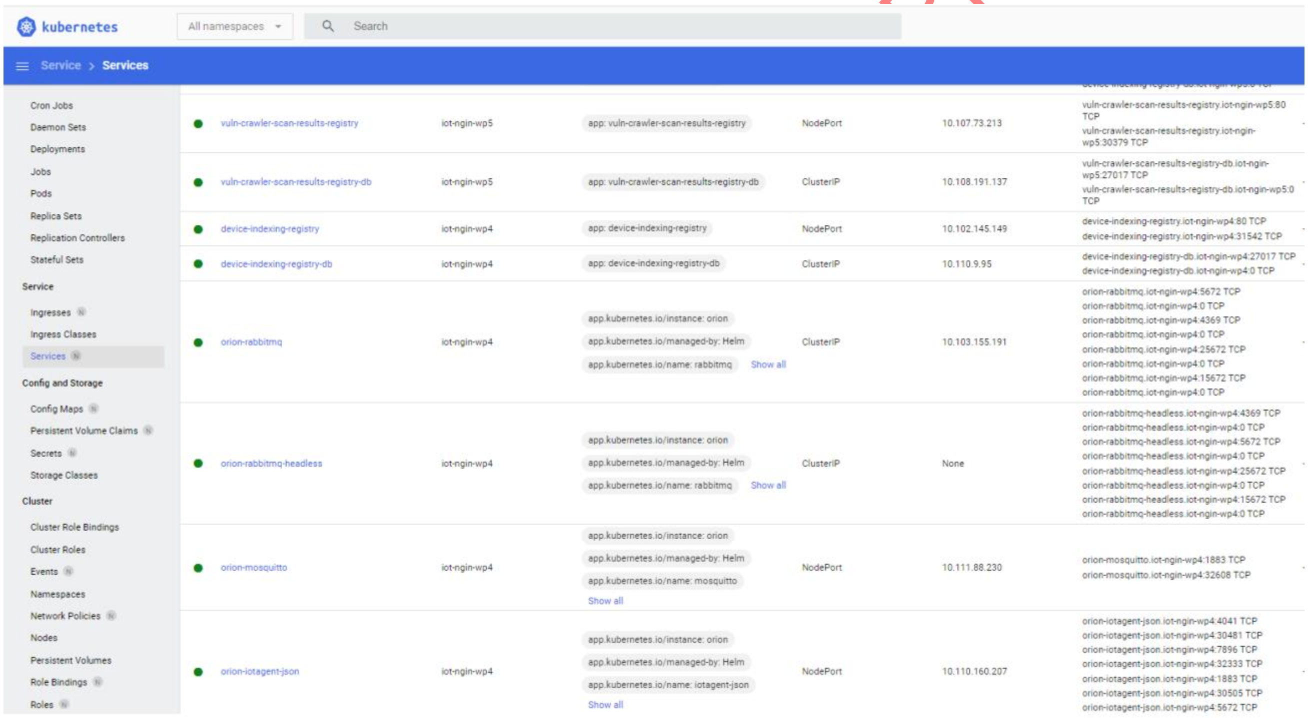
More specifically, the Kubernetes namespaces which correspond to each WP activities and are defined within each IoT-NGIN cluster are described in Table 2 in tabulated format.

Table 2 – IoT-NGIN WP, Namespace and cluster mapping.

Kubernetes Namespace	IoT-NGIN WP	IoT-NGIN cluster
N/A	WP2: Enhancing IoT Underlying Technology	Eurolab cluster

Default	WP3: Enhancing IoT Intelligence	OneLab auxiliary cluster (GPU enabled)
Iot-ngin-wp3	WP3: Enhancing IoT Intelligence	OneLab main cluster
Iot-ngin-wp4	WP4: Enhancing IoT Tactile & Contextual Sensing/Actuating	OneLab main cluster
Iot-ngin-wp5	WP5: Enhancing IoT Cybersecurity & Data Privacy	OneLab main cluster
Iot-ngin-wp6	WP6: IoT-NGIN Integration & Laboratory evaluation	OneLab main cluster

In Figure 9, some indicative IoT-NGIN components' services deployed in the Kubernetes cluster are described.



●	vuln-crawler-scan-results-registry	iot-ngin-wp5	app: vuln-crawler-scan-results-registry	NodePort	10.107.73.213	vuln-crawler-scan-results-registry.iot-ngin-wp5:80 TCP vuln-crawler-scan-results-registry.iot-ngin-wp5:30379 TCP
●	vuln-crawler-scan-results-registry-db	iot-ngin-wp5	app: vuln-crawler-scan-results-registry-db	ClusterIP	10.108.191.137	vuln-crawler-scan-results-registry-db.iot-ngin-wp5:27017 TCP vuln-crawler-scan-results-registry-db.iot-ngin-wp5:0 TCP
●	device-indexing-registry	iot-ngin-wp4	app: device-indexing-registry	NodePort	10.102.145.149	device-indexing-registry.iot-ngin-wp4:80 TCP device-indexing-registry.iot-ngin-wp4:31542 TCP
●	device-indexing-registry-db	iot-ngin-wp4	app: device-indexing-registry-db	ClusterIP	10.110.9.95	device-indexing-registry-db.iot-ngin-wp4:27017 TCP device-indexing-registry-db.iot-ngin-wp4:0 TCP
●	orion-rabbitmq	iot-ngin-wp4	app.kubernetes.io/instance: orion app.kubernetes.io/managed-by: Helm app.kubernetes.io/name: rabbitmq Show all	ClusterIP	10.103.155.191	orion-rabbitmq.iot-ngin-wp4:5672 TCP orion-rabbitmq.iot-ngin-wp4:0 TCP orion-rabbitmq.iot-ngin-wp4:4369 TCP orion-rabbitmq.iot-ngin-wp4:0 TCP orion-rabbitmq.iot-ngin-wp4:25672 TCP orion-rabbitmq.iot-ngin-wp4:0 TCP orion-rabbitmq.iot-ngin-wp4:15672 TCP orion-rabbitmq.iot-ngin-wp4:0 TCP
●	orion-rabbitmq-headless	iot-ngin-wp4	app.kubernetes.io/instance: orion app.kubernetes.io/managed-by: Helm app.kubernetes.io/name: rabbitmq Show all	ClusterIP	None	orion-rabbitmq-headless.iot-ngin-wp4:4369 TCP orion-rabbitmq-headless.iot-ngin-wp4:0 TCP orion-rabbitmq-headless.iot-ngin-wp4:5672 TCP orion-rabbitmq-headless.iot-ngin-wp4:0 TCP orion-rabbitmq-headless.iot-ngin-wp4:25672 TCP orion-rabbitmq-headless.iot-ngin-wp4:0 TCP orion-rabbitmq-headless.iot-ngin-wp4:15672 TCP orion-rabbitmq-headless.iot-ngin-wp4:0 TCP
●	orion-mosquitto	iot-ngin-wp4	app.kubernetes.io/instance: orion app.kubernetes.io/managed-by: Helm app.kubernetes.io/name: mosquitto Show all	NodePort	10.111.88.230	orion-mosquitto.iot-ngin-wp4:1883 TCP orion-mosquitto.iot-ngin-wp4:32608 TCP
●	orion-iotagent-json	iot-ngin-wp4	app.kubernetes.io/instance: orion app.kubernetes.io/managed-by: Helm app.kubernetes.io/name: iotagent-json Show all	NodePort	10.110.160.207	orion-iotagent-json.iot-ngin-wp4:4041 TCP orion-iotagent-json.iot-ngin-wp4:30481 TCP orion-iotagent-json.iot-ngin-wp4:7896 TCP orion-iotagent-json.iot-ngin-wp4:32333 TCP orion-iotagent-json.iot-ngin-wp4:1883 TCP orion-iotagent-json.iot-ngin-wp4:30505 TCP orion-iotagent-json.iot-ngin-wp4:5672 TCP

Figure 9 – Indicative IoT-NGIN Kubernetes services.

2.2.3.1 IoT-NGIN platform integration considerations

The integrated IoT-NGIN platform is realised based on the IoT-NGIN meta-architecture that was defined and described in D1.2 [1] and further updated within D6.1 [3]. Moreover, within WP6 (T6.2 activities), additional application-logic enhancements have been introduced (described in detail in section 3), and then designed and developed, in order to both support but also refine the IoT-NGIN Living Labs' particular UCs provided functionality and thus their operation. This indicates the necessity to configure the integrated IoT-NGIN platform in multiple variations/instantiations and subsequently be able deploy them in the respective Living Labs' cloud/edge infrastructures.

The abovementioned integration procedure is rather complex and undoubtedly is not a trivial task to undertake effectively and at the same time not further increase the complexity both for the integration and also the deployment phase of the IoT-NGIN platform in the Living Labs. In order to facilitate this process, we opted to utilise HELM charts [5]. Helm is a package manager for Kubernetes that makes it easy to take applications and services that are either highly repeatable or used in multiple scenarios and deploy them to a typical K8s cluster and HELM charts essentially describe how to manage a specific application or set of applications on Kubernetes. More specifically, HELM charts consist of metadata that describe the applications and the infrastructure's associated needs aiming to provide a simple, smooth and trouble-free deployment of the integrated IoT-NGIN framework variations.

Adopting HELM, IoT-NGIN aims to deliver smooth integration between the numerous developed components and also an easy way to update (upgrade) them when is required. In addition, HELM eases the installation of the various IoT-NGIN platform instantiations that are dictated by the defined meta-architecture and the additional application-logic enhancements that have been incorporated.

More specifically, HELM removes the necessity of going through a Kubernetes Manifest's details to make the appropriate changes every time an update or new deployment is required. All the changes are handled via a "values.yaml" file. This approach allows the user of the HELM chart to be agnostic to the details of the underlying implementation and know only of the values they are required to set during preparation for the installation. Additionally, the user can deploy multiple instances of an application either on the same or different premises, each one being differently configured.

Therefore, in the case of IoT-NGIN, where the meta-architecture is quite complex and involves a number of different components and different partners require to manage their own instances, HELM proves a helpful tool to easily rollout new updates or keep track of different instances of the installations.

2.2.3.2 Integration phases and time-plan

The IoT-NGIN integration plan has been presented in D6.1 [3] and suggests that the IoT-NGIN implementation would be established in 3 phases that would result in 3 respective integration activities. Each IoT-NGIN integrated platform release takes place prior to the end of each phase and is tested and validated by the end of each phase. As indicated by the Figure 10, the main target here is to provide the completed IoT-NGIN integrated framework by M30 and proceed on deploying and testing the platform both in a Lab setting and on the various Living Labs. Thus, the final integrated version of the IoT-NGIN framework should be delivered through incremental upgrades of the IoT-NGIN developed components and their associated interfaces.

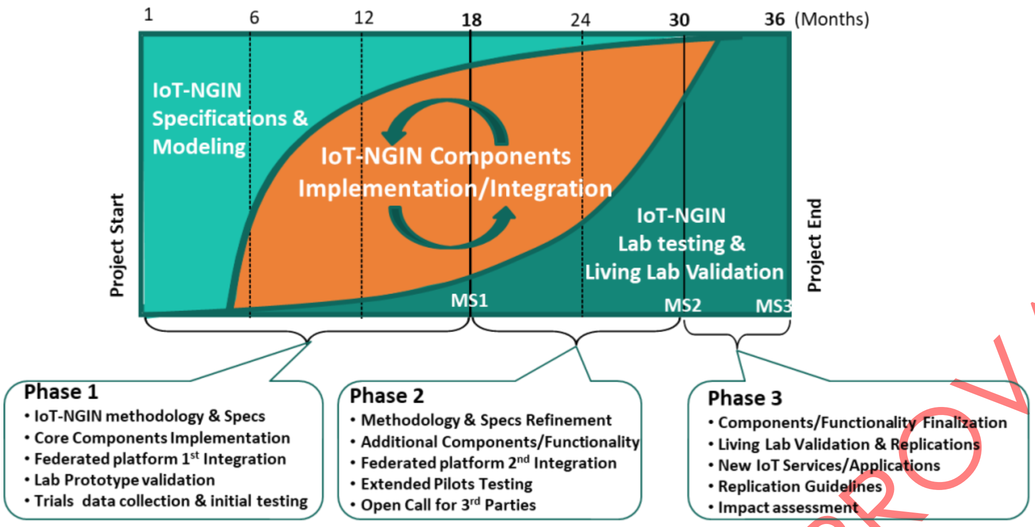

1
6
12
18
24
30
36 (Months)
Project Start
Project End
IoT-NGIN Specifications & Modeling
IoT-NGIN Components Implementation/Integration
IoT-NGIN Lab testing & Living Lab Validation
MS1
MS2
MS3
Phase 1
• IoT-NGIN methodology & Specs
• Core Components Implementation
• Federated platform 1st Integration
• Lab Prototype validation
• Trials data collection & initial testing
Phase 2
• Methodology & Specs Refinement
• Additional Components/Functionality
• Federated platform 2nd Integration
• Extended Pilots Testing
• Open Call for 3rd Parties
Phase 3
• Components/Functionality Finalization
• Living Lab Validation & Replications
• New IoT Services/Applications
• Replication Guidelines
• Impact assessment

Figure 10 – IoT-NGIN phased approach.

2.2.3.3 IoT-NGIN Integrated Platform (1st version)

One of the main objectives of this document is to describe the process followed in order to provide the 1st version of the IoT-NGIN integrated platform. Therefore, the basic prerequisite for this activity is to have the IoT-NGIN developed technical solutions' source code uploaded in the project's git repository (GitLab) following the project's development guidelines. In brief, these guidelines include the adoption of modular programming for delivering containerised applications, the provision of adequate documentation both for the component's functionality and within the source code, the definition (and description) of proper interfaces following standardised formats and finally the provision of the necessary configuration files.

Then, the IoT-NGIN developers would need to utilise the provided CI/CD pipeline and deploy their components in the integration and testing Kubernetes cluster. The required steps and indicative examples for setting up the CI/CD pipeline are provided in section 2.1.2. The associated integration activities can also be performed individually in a bilateral format by the relevant partners. This is more convenient for small scale integration activities that can be preliminary performed in a local level and then proceed on deploying the now integrated and tested components in the project's integration and testing infrastructure. The applied integration activities concern the interconnection activities between the components' implemented interfaces.

The components that comprise the IoT-NGIN framework are described in a tabulated format in Table 3.

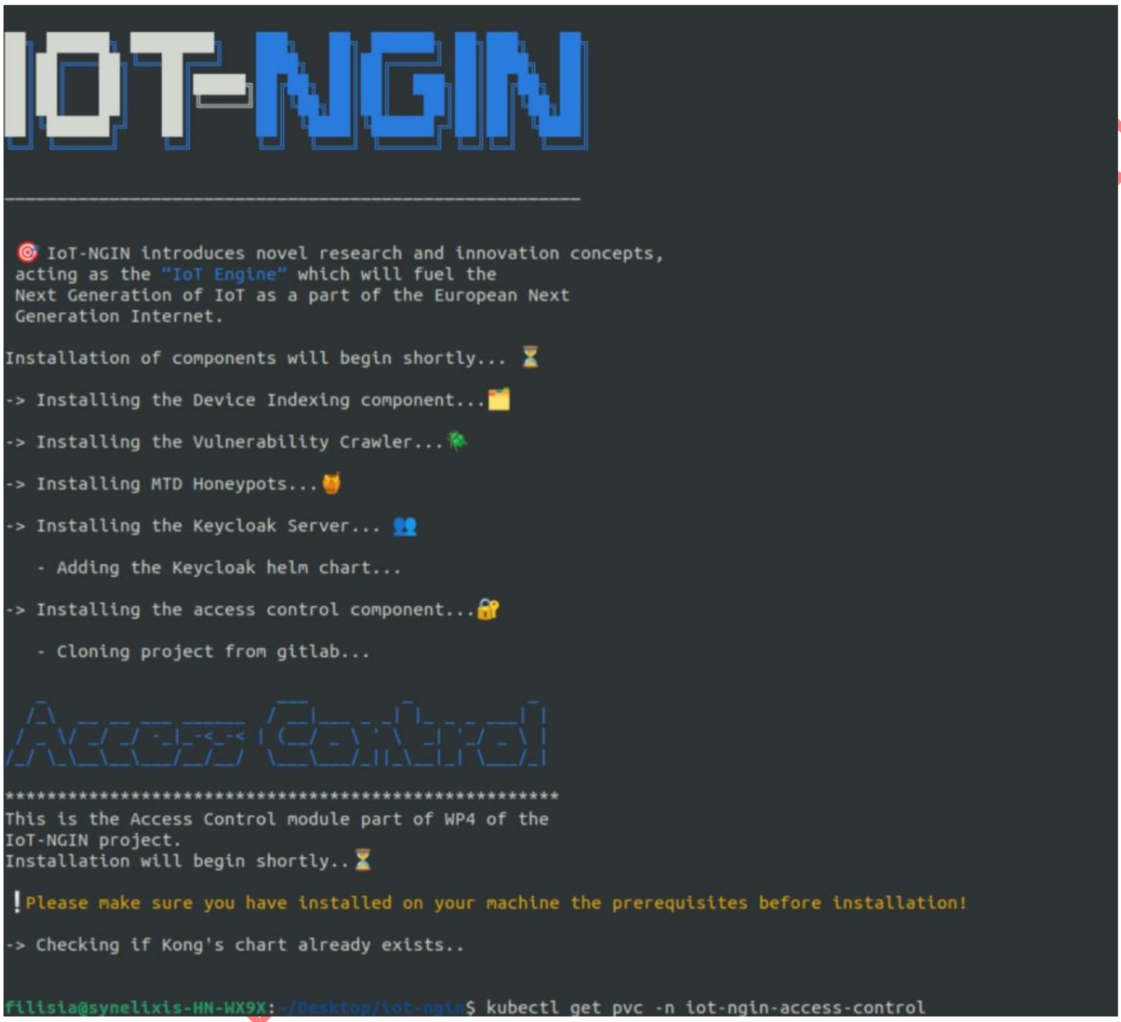
Table 3 – IoT-NGIN Framework components' list.

IoT-NGIN WP components	IoT-NGIN components
WP2: Enhancing IoT Underlying Technology	• Secure Edge Cloud execution framework • 5G Resource Management API • Slice & Orchestration Engine • 5G device management API • Network Controller API • TSN Bridge

	• Device-to-Device communication (5G Coverage Extension)
WP3: Enhancing IoT Intelligence	• MLaaS • Online Learning Framework • Reinforcement Learning Framework • Privacy Preserving Federated Learning • PPFL API • Polyglot Model Sharing
WP4: Enhancing IoT Tactile & Contextual Sensing/Actuating	• IoT Device Discovery (Computer-vision, face) • IoT Device Discovery (Computer-vision, object) • IoT Device Discovery (Visual Light Positioning - VLP) • IoT Device Discovery (Ultra-Wide Band - UWB) • IoT Device Discovery (Code scanning) • IoT Device Indexing • IoT Devices Access Control • AR tools
WP5: Enhancing IoT Cybersecurity & Data Privacy	• GAN-based Data Generator • IoT Vulnerability Crawler • Malicious Attack Detector (MAD) • Moving Target Defense (MTD) Honeypot Framework • Decentralised Interledger Bridge • Privacy Preserving Self-Sovereign Identities (SSI) • Semantic Twins • SAREF ontologies
WP6: IoT-NGIN Integration & Laboratory evaluation	• Quorum Network • Keycloak server • UCs' application logic enhancements • Installation scripts

In section 2.2, Figure 5 indicates the integration status of the IoT-NGIN developed components. Hence, dashed lines in the figure indicate that integration activities have been at least at the stage of development/testing, but are still in progress, while continuous lines indicate that integration among the interacting components has been performed at a mature level with interfaces working or even with automated configuration and deployment completed. IoT-NGIN related integration information is described in detail in Annex 2 "*IoT-NGIN Integration roadmap*".

Automated configuration and deployment are achieved by combining (sub-)component-level Helm charts with custom installation scripts. The IoT-NGIN framework comprises a set of components or microservices and each of them may be composed of a set of submodules. Helm charts for IoT-NGIN components and their submodules will be configured and deployed via custom installation bash scripts, which will be called by a single setup script for the complete IoT-NGIN framework.



```
IOT-NGIN
------------------------------------------------------------

🎯 IoT-NGIN introduces novel research and innovation concepts,
acting as the "IoT Engine" which will fuel the
Next Generation of IoT as a part of the European Next
Generation Internet.

Installation of components will begin shortly... ⏳

-> Installing the Device Indexing component...🗂️

-> Installing the Vulnerability Crawler...🕷️

-> Installing MTD Honeypots...🍯

-> Installing the Keycloak Server... 👥

   - Adding the Keycloak helm chart...

-> Installing the access control component...🔐

   - Cloning project from gitlab...

Access Control

*****************************************************
This is the Access Control module part of WP4 of the
IoT-NGIN project.
Installation will begin shortly..⏳

❕Please make sure you have installed on your machine the prerequisites before installation!

-> Checking if Kong's chart already exists..

filisia@synelixis-HN-WX9X:~/Desktop/iot-ngin$ kubectl get pvc -n iot-ngin-access-control
```

Figure 11 – Installation of IoT-NGIN cybersecurity tools through a single script.

At the current stage, IoT-NGIN cybersecurity tools have been already packaged under single "Master" Helm Charts (chart.yaml) which include the charts of the individual IoT-NGIN components and their dependencies as sub-charts. Annex 3 presents these charts for a subset of the IoT-NGIN components. In addition, values.yaml files that adhere to each IoT-NGIN namespace have been also created. The latter includes configuration settings that enable the IoT-NGIN components to be deployed in each defined namespace. One-click installation is possible through a single script, as depicted in Figure 11.

Once the integration of all IoT-NGIN developed components will be finalised, in the following months of the project's lifespan and in view of the finalised and integrated IoT-NGIN platform that will be delivered on M30, we plan to consolidate the individual HELM charts under a single setup script, which will enable automatic selective configuration and deployment of all developed technical solutions. The namespaces defined in the IoT-NGIN Kubernetes cluster are associated to each WP that the developed components belong to (e.g. iot-ngin-wp6 etc.).

A key attribute of the HELM Chart organization is the fact that by including all the sub-charts as entries to a values.yaml file, and setting the enabled variable accordingly, we are able to select the components of the "Master" Helm chart that are going to be deployed. Therefore, as we proceed towards the final IoT-NGIN integrated framework, we would be able to configure the appropriate IoT-NGIN infrastructure accordingly and deploy it in the required Living Lab. Thus, despite the initial overhead needed to author the necessary HELM charts, following this approach would significantly ease the deployment and subsequently the adoption of the IoT-NGIN platform by the project's Living Labs.

An indicative example of the chart.yaml is presented below.

```
apiVersion: The chart API version (required)
name: The name of the chart (required)
version: A SemVer 2 version (required)
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)
description: A single-sentence description of this project (optional)
type: The type of the chart (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this projects home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
dependencies: # A list of the chart requirements (optional)
  - name: The name of the chart (nginx)
    version: The version of the chart ("1.2.3")
    repository: (optional) The repository URL ("https://example.com/charts") or
alias ("@repo-name")
    condition: (optional) A yaml path that resolves to a boolean, used for
enabling/disabling charts (e.g. subchart1.enabled )
    tags: # (optional)
      - Tags can be used to group charts for enabling/disabling together
    import-values: # (optional)
      - ImportValues holds the mapping of source values to parent key to be
imported. Each item can be a string or pair of child/parent sublist items.
    alias: (optional) Alias to be used for the chart. Useful when you have to add
the same chart multiple times
maintainers: # (optional)
  - name: The maintainers name (required for each maintainer)
    email: The maintainers email (optional for each maintainer)
    url: A URL for the maintainer (optional for each maintainer)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). Needn't be
SemVer. Quotes recommended.
deprecated: Whether this chart is deprecated (optional, boolean)
annotations:
  example: A list of annotations keyed by name (optional).
```

In addition, an initial IoT-NGIN setup script (shell script) has been considered to be authored in order to facilitate further the installation process of the IoT-NGIN platform. Here, the plan is to provide separate installation workflows associated to the specific IoT-NGIN platform instantiation required by each Living Lab. Within this setup script the user among other is able to define e.g. the IPv4 address of the installation machine, the docker registry password, namespaces, the Kubernetes client type etc.

3 IoT-NGIN implementation for Living Lab Use Cases

In this section, the planned IoT-NGIN implementation across the Living Lab use cases is described. Based on the living lab preparation for IoT-NGIN validation reported in D7.2 [22], in particular the technology alignment with the project's developments and the sequence diagrams provided for each use case, the IoT-NGIN architecture instantiations are presented for each use case. These architecture instantiations also incorporate the application logic enhancements that are tailored to each LL specific UC needs. These business specific components which fall into the scope of WP6 are further elaborated, where relevant.

3.1 Human-Centred Twin Smart Cities Living Lab

The Human-Centred Twin Smart Cities Living Lab focuses on Jätkäsaari, which is a long and narrow peninsula in southern Helsinki. For vehicle traffic, Jätkäsaari has only two connections to the mainland: the root of the peninsula itself and a bridge near the outermost tip. This limited connectivity leads to regular traffic congestions as not only is the peninsula expected to eventually reach some 17.000 residents and 6.000 jobs, but it also already hosts a ferry harbour, where large ferries from Tallinn, Estonia arrive every 3 hours throughout the day. To help the passengers and locals move around, Jätkäsaari is serviced by three tramlines, but the numerous trucks and other vehicles arriving on the Ferries still create significant congestion.

All three use cases UC1-3 in the Smart Cities Living Lab, therefore, aim at reducing traffic congestion in Jätkäsaari by focusing on different types of traffic: UC1 focuses on vehicle traffic, UC2 focuses on pedestrians both on the streets of Jätkäsaari and in the ferry terminal, and UC3 focuses on the co-commuting of the ferry passengers.

3.1.1 UC1 - Traffic Flow Prediction & Parking prediction

This use case focuses on the vehicles in Jätkäsaari and neighboring areas. Of particular interest are the trucks arriving to and from the ferries as they create more noise and pollution than regular vehicles, but simply focusing on them alone will not be sufficient to resolve the congestion issues, so other vehicles, including the vehicles of people living and working in Jätkäsaari, will be considered.

The main approach is to first collect sufficient information about the vehicle flows using different types of sensors (e.g. cameras, radars, lidars and under-the-street sensors) and then process the information using ML and simulation models to create the necessary situational awareness and predictions of traffic flows. This information is then utilised for optimizing the timing of the traffic lights in the area and on the route to the closest freeway to increase the traffic throughput in the critical directions as well as to advise the drivers on best routes and parking options. A key collaboration partner here is the Smart Junction projects, which is deploying additional traffic sensors in the area and developing the traffic light controlling solution.

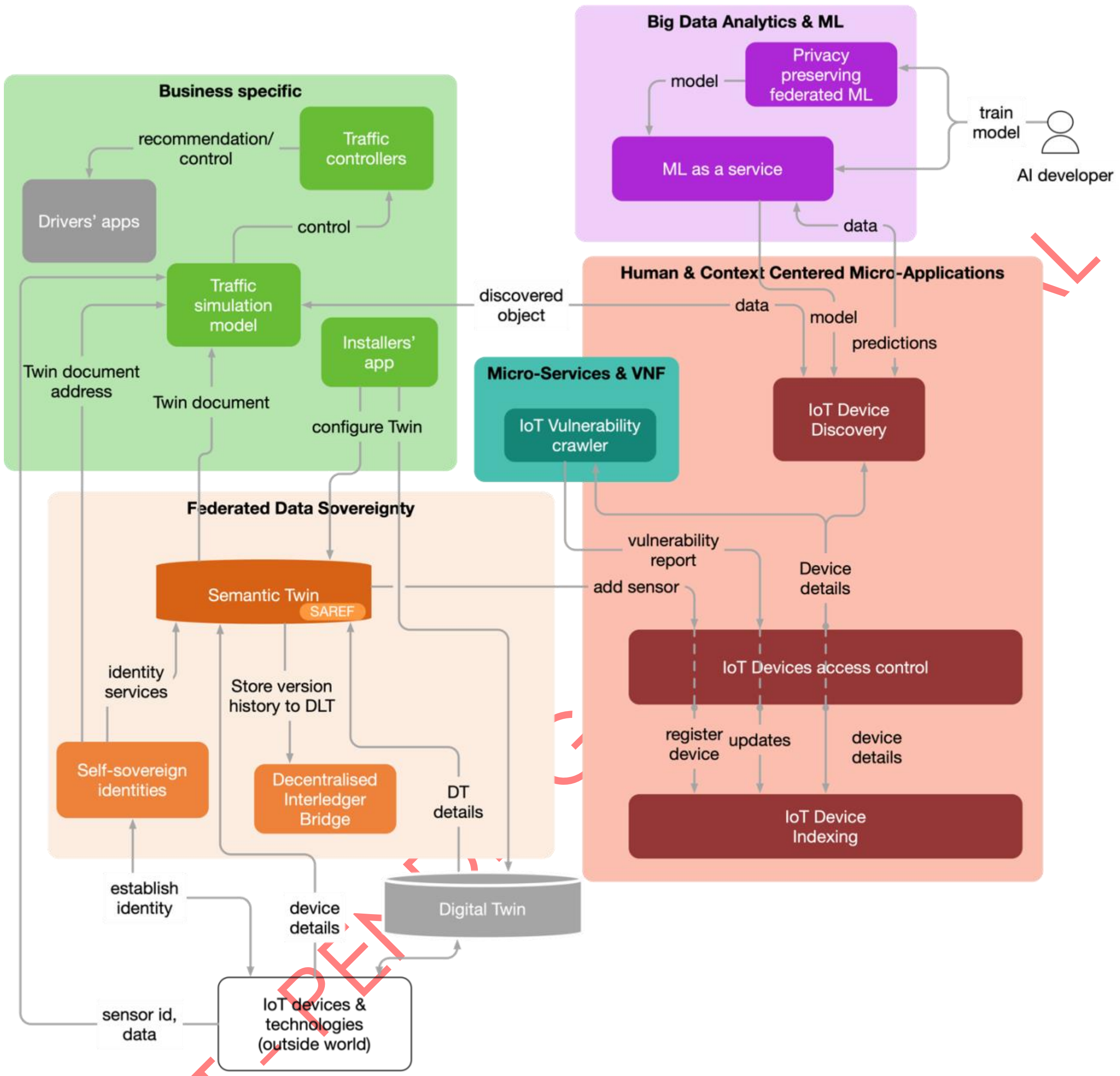


Figure 12 – Architecture instantiation for UC1.

The IoT-NGIN project contributes multiple components to improve and deploy the solution as shown in Figure 12. One area of interest is the easy deployment and management of the IoT devices. Here, IoT-NGIN is developing the Semantic Twin (ST) concept, which can be utilised to describe the device's and its Digital Twin's (DT's) capabilities in an machine-readable manner. The Semantic Twin component builds on SAREF ontologies to ensure interoperability, on Self-Sovereign Identities (SSIs) to improve privacy, and on Distributed Ledger Technologies (DLTs) via the Decentralised Interledger Bridge (DIB) to enable an immutable shared history to further increase trustworthiness of the STs. The Installer app can then be utilised to help configure the STs and DTs of the installed devices in an easy-to-use and secure (enabled by the IoT Device Access Control component) manner.

The second focus for the IoT-NGIN components are the ML-based vehicle recognition and simulation models. Here, the MLaaS platform and federated ML components support the recognition of the IoT Device Discovery.

The third focus area is the cybersecurity of the IoT devices, which is supported with the Vulnerability Crawler that relies on the IoT Device Access Control and Indexing components to scan the devices deployed in the area for potential vulnerabilities.

3.1.1.1 Business-specific components

Four main business-specific functionalities can be identified in this use case as described in the following subsections.

3.1.1.1.1 Traffic simulation model

The traffic simulation model utilises the information from the different sensors to build the situational awareness and predictions of the traffic in the area.

3.1.1.1.2 Traffic controllers

The traffic controller utilises the traffic predictions from the simulation model to optimise the traffic flows in the critical directions by managing the timings of the traffic lights in the area, but it also makes the congestion information available to the different drivers' applications and other information channels such as Traffic Announcements on car radios' RDS systems so that drivers can better time and route their movements as well as find suitable parking slots faster to minimise unnecessary circulation.

3.1.1.1.3 Drivers' application

Drivers' applications cover a wide range of existing application that the driver is likely to utilise in connection with travelling to or from Jätkäsaari. Examples include Google/Apple/Open Street Maps used for navigating, so providing better predictions of congestion information to these map services could help provide better routing information. Similarly, the ferry company's (Tallink's) app that is already used by many passengers for their tickets could also integrate traffic information to guide the drivers. Finally, parking apps used to pay for the parking services could also help find available parking spaces by utilizing the parking predictions.

3.1.1.1.4 Installers' application

Installer's app is a new app prototype being developed in IoT-NGIN for helping the installer configure the Semantic and Digital Twins related to the IoT devices being installed and maintained. Each device has a sticker with a unique QR code that the app can then scan using the mobile device's camera to easily and securely discover the Twins related to that device and to allow secure access for managing the information within the Twins.

3.1.2 UC2 - Crowd Management

The Crowd Management use case is very similar to UC1 one except it focuses on managing pedestrians and public transit passengers on the streets of Jätkäsaari and ferry passengers in the harbour terminal.

The approach is, therefore, also nearly identical to the UC1: collect information using sensors such as cameras and Bluetooth beacon detectors and process it to create predictions of the movements of the crowds. Finally, different existing apps and other channels can, again, be used to disseminate useful information to the crowds and service providers such as taxis to help improve traffic flows.

The IoT-NGIN components are, as shown in Figure 13, also used for identical purposes as in the UC1 detailed in the previous section.

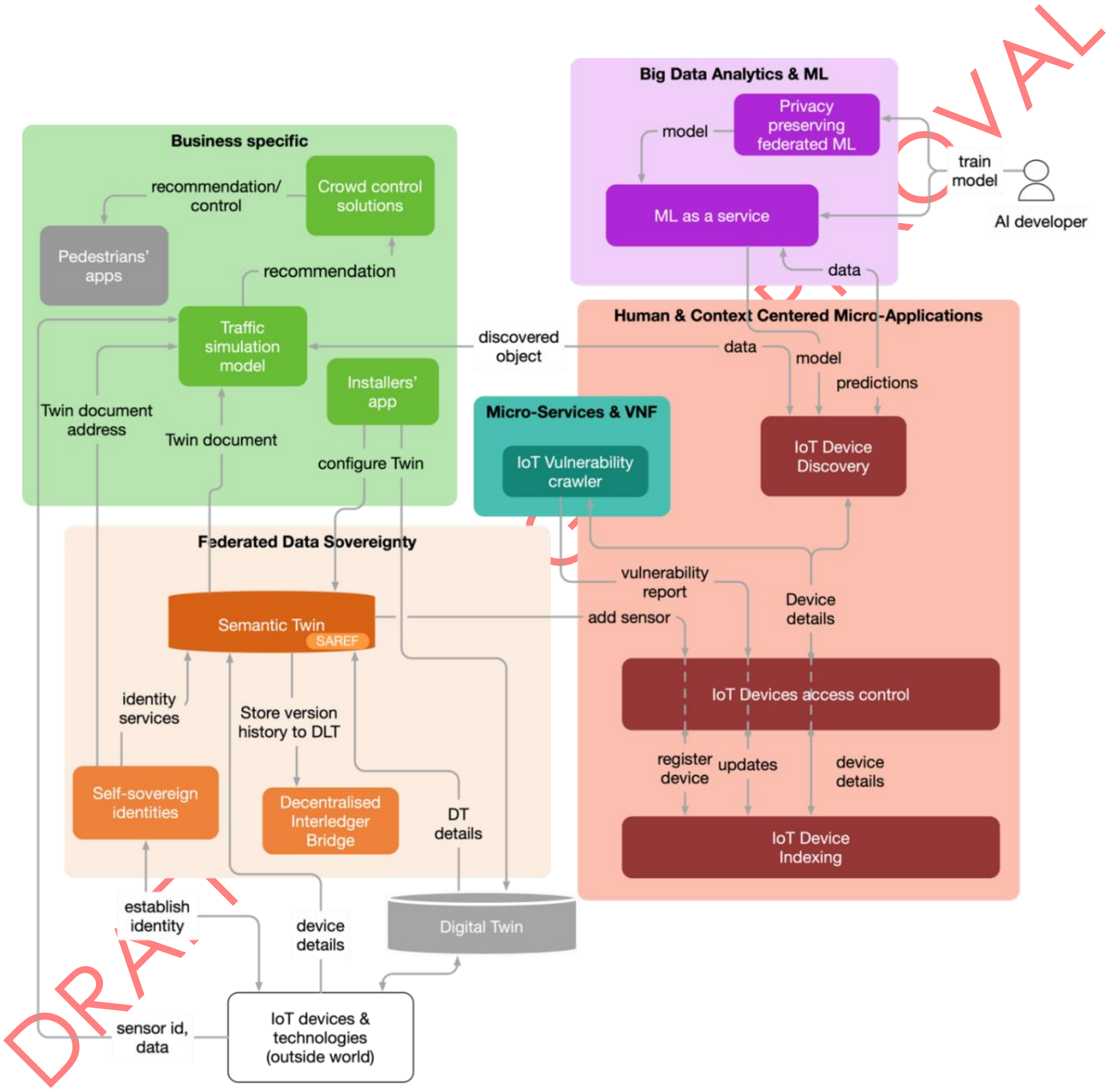


Figure 13 – Architecture instantiation for UC2.

3.1.2.1 Business-specific components

Again, four main business-specific functionalities can be identified in this use case as described in the following subsections.

3.1.2.1.1 Crowd simulation solutions

Same as UC1 except for the crowds.

3.1.2.1.2 Crowd control model

The crowd is controlled by changing the timings of the traffic lights for pedestrians but also, e.g., by providing the congestion predictions to related service providers such as taxis companies and e-scooter services to better match the available capacity to the need for rides.

3.1.2.1.3 Pedestrians' application

These, again, include different existing applications the crowds are likely to utilise. Examples include the different maps used for navigation and public transport information as well as the apps for using the public transport services including different MaaS apps. Also, the ferry company app can provide relevant advice to the crowds.

3.1.2.1.4 Installers' application

The app and its usage is identical to UC1.

3.1.3 UC3 - Co-commuting solutions based on social networks

UC3 is designed to be built on top of the UC1 and UC2. The main goal of UC3 is utilising open data sets and citizen data to design accessible on-demand co-commuting platforms for solutions and enrich the awareness of the transportation needs via use of social networks.

The approach of the use case includes harvesting data from the Twitter feed to the co-commuting platform, performing analysis of the "tweets" for the demand prediction model, and disseminating the generated content to potential 3rd party applications, such as ride-sharing apps, ferry company apps, scooter apps etc.

The IoT-NGIN components shown on Figure 14 are, again, used for the same purposes as in the previous use-cases.

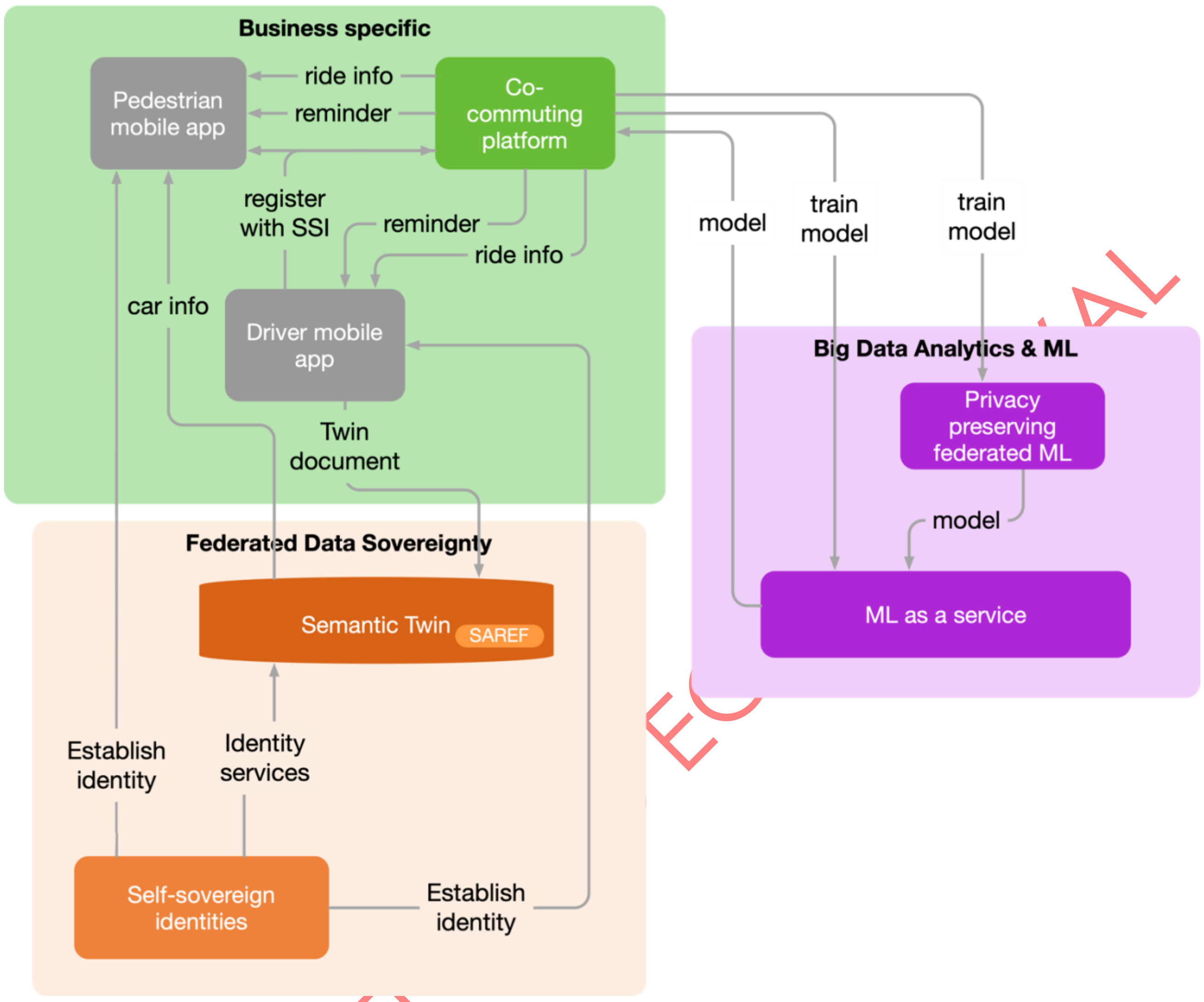


Figure 14 – Architecture instantiation for UC3.

3.1.3.1 Business-specific components

Three main business-specific functionalities can be identified in this use case as described in the following subsections.

3.1.3.1.1 Co-commuting platform

Social network data integration into the Co-commuting platform is considered a key enabler for this use case. Twitter is selected as an integration target due to its high utilisation in the tested environment, thus generating data with sufficient quality to estimate commuting demand.

The Co-commuting platform builds on the Urban Open Platform, giving 3rd party organizations easy access to the co-commuting data. Urban Open Platform (UoP) is an Open Data platform developed as part of the FinEst project, is focused on collecting sensor data between cross-border cities, city systems, city infrastructure of private/public sector. In the context of UC3, UoP acts as the data processing layer for Co-commuting platform, performing such tasks as data integrations, Exchanging ML training data sets, communicating with 3rd party applications, and generating high demand reminders between sub-systems. Such reminders are already used by taxi operators offering pre-booking functionality. In addition to reminders data exchange between sub-systems includes routes of the passengers, commute times, trip features, and ride information.

The listed extracted data is then processed by the Co-commuting platform, to therefore indicate potential demand to the ride-sharing applications. It is also expected that such 3rd party applications might benefit from the outputs by expanding their offering based on the demand, such as ferry organizations offering taxis as extra services.

3.1.3.1.2 Driver and pedestrian mobile application (3rd part applications)

As stated in D7.2, the scope of the project does not include a ride-sharing service so the UC3 will focus on the information flows coming from twitter feeds to demonstrate the potential benefit of the IoT-NGIN concept towards the applications of public transportation, ride-sharing, harbor applications etc.

As with the previous use-cases, this might again include different existing applications used by commuters. Such examples include ferry operators' apps used for ticket sales, extra ferry services, checking the crowdedness in the harbor. Applications providing the rented scooters is another example, benefitting from predicting the demand to therefore provision the right number of scooters on the place. Lastly, taxi service apps are likely to be used by commuters, thus predicted demand allows proper allocation of assets to the place.

3.2 Smart Agriculture Living Lab

3.2.1 UC4 - Crop diseases prediction. Smart irrigation and precision aerial spraying

The "Crop diseases prediction. Smart irrigation and precision aerial spraying" use case aims at enhancing the Smart Farmer's work related to the irrigation and spraying processes. Specifically, the IoT-NGIN tools are employed to

- Improve the farmer's awareness about the crop/field conditions, while exposing enhanced cyber-physical interaction to facilitate the irrigation process. This is achieved by presenting relevant data/metric and exposing advanced device discovery and access control capabilities to facilitate the farmer's interaction with the monitoring/irrigation devices.
- Maintain healthy crops, by decreasing the occurrence of crop diseases through advanced detection or prediction, while requiring less physical effort and rationalised investments. IoT-NGIN provides tools for effective training of Machine Learning models for crop disease prediction, which are then deployed on Unmanned Aerial Vehicles (UAVs) performing live predictions while flying over the Smart Farm.
- Ensure protection of Smart Farm against cyber-attacks, which could affect individual farms or even regions, depending on the extent or the spread of such attacks. The Smart Farm's cybersecurity relies on scanning for potential vulnerabilities and tracing potential attack patterns, as well as detecting potential attacks at the network level. In addition, the technical robustness of the ML models and operation is protected via ML-based attack detection against model and poisoning attacks, while ML models are securely executed in SEE.

On top of these, IoT-NGIN aims to reduce network overheads and increase resource usage efficiency by leveraging edge computing and Digital Twins in the Smart Farm.

IoT-NGIN offers these capabilities through the architecture instantiation depicted in Figure 15. As shown in the figure, network data are forwarded to IDI and dispatched to the IoT Vulnerability Crawler for detecting potential vulnerabilities, which are communicated to both IDI and MTD Network of Honeypots for deploying honeypots in the context of those vulnerabilities. However, Synelixis' SynField [23] and UAVs are the main data sources for the UC, which provide monitoring data/metrics and images to IDI, protected by the IDAC component.

The Smart Farmer that is close to a SynField device may access its data through the "Smart Agri app". The app scans the SynField's QR code and retrieves its ID through IDD QR. It then sends images of the device to "IDD - Computer Vision" to identify the device. This is performed via model serving through the MLaaS platform. If the device is recognised, it requests SynField's data from IDI through IDAC, which performs access control, based on the requester's identity and proximity to the device. If access is granted, the SynField data are presented in the app over SynField image in the form of AR objects. The user is also provided AR buttons to start or stop the irrigation on this device.

In addition, the drone images are used to train the disease prediction model via Federated Learning (FL). This is initiated by an AI developer calling the PPFL API. Moreover, synthetic data of the "GAN-based IoT Attack Dataset Generator" are used to perform a data poisoning attack, which is detected by the "Malicious Attack Detector" (MAD).

After training, free of attacks, the model is stored in MLaaS model storage and its deployment can be triggered by the AI developer through IDAC and IDI to the UAV. Upon reception of a new model, the UAV performs inference on the acquired images, which is the "Crop Disease Prediction" in the figure.

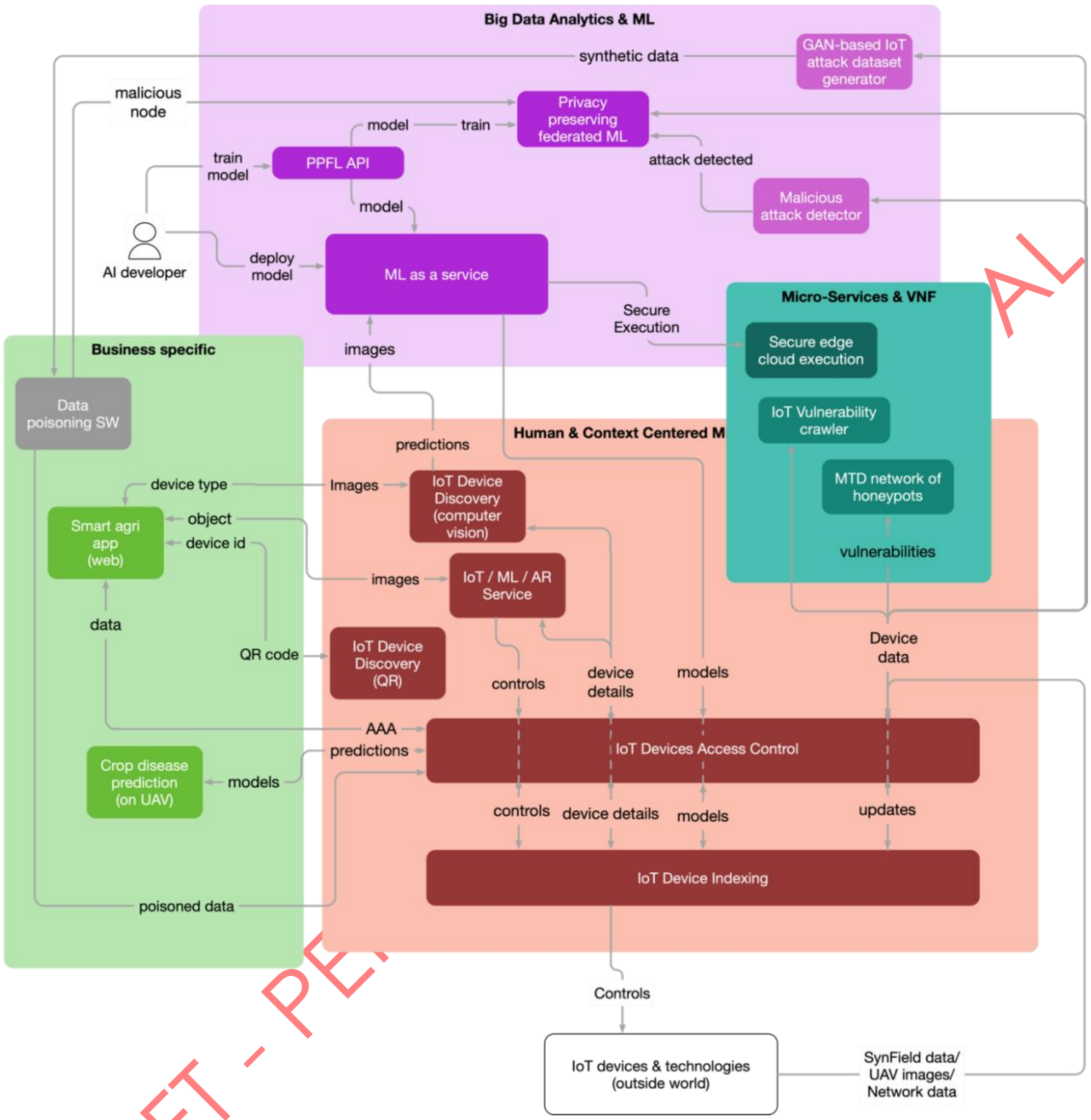


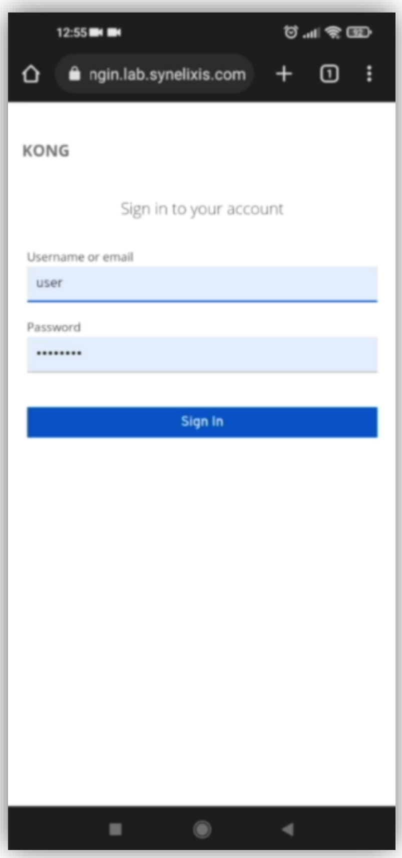
Figure 15 – Architecture instantiation for UC4.

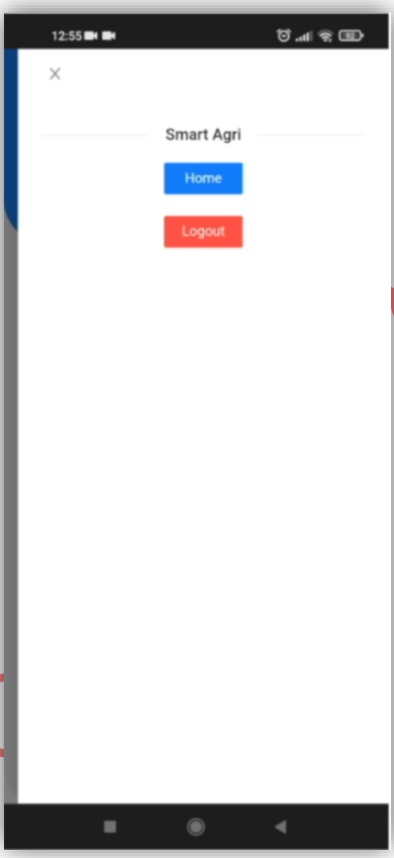
3.2.1.1 Business-specific components

In this section, the components identified specifically for the execution of UC4 are described. Specifically, the "Smart Agri Application" and "Crop Disease prediction" are such components.

3.2.1.1.1 Smart Agri application

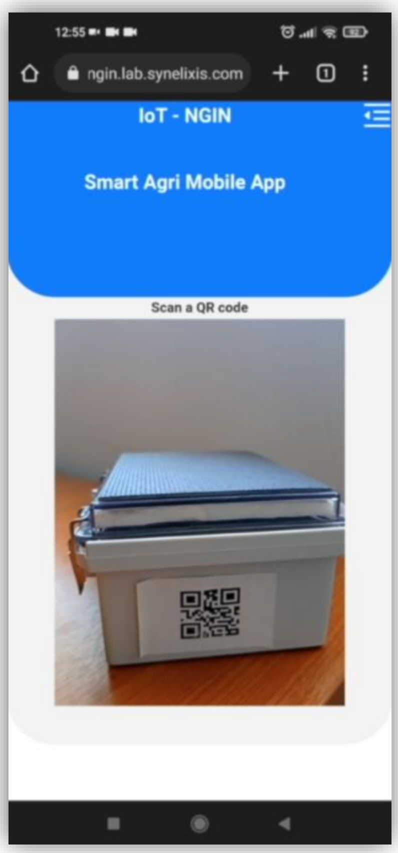
The Smart Agri app provides the interface for the Smart Farmer to experience enhanced cyber-physical interactions with SynField devices in the context of introducing ambient intelligence in the Smart Farm.



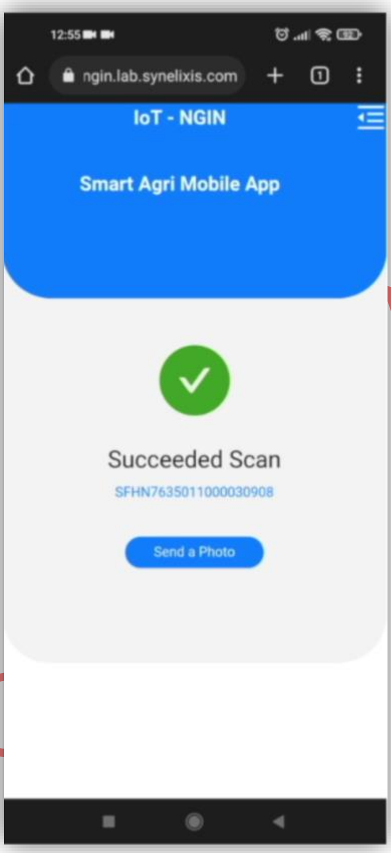


(a) (b)

Figure 16 – Login (a) and logout (b) options in the Smart Agri app.



(a)



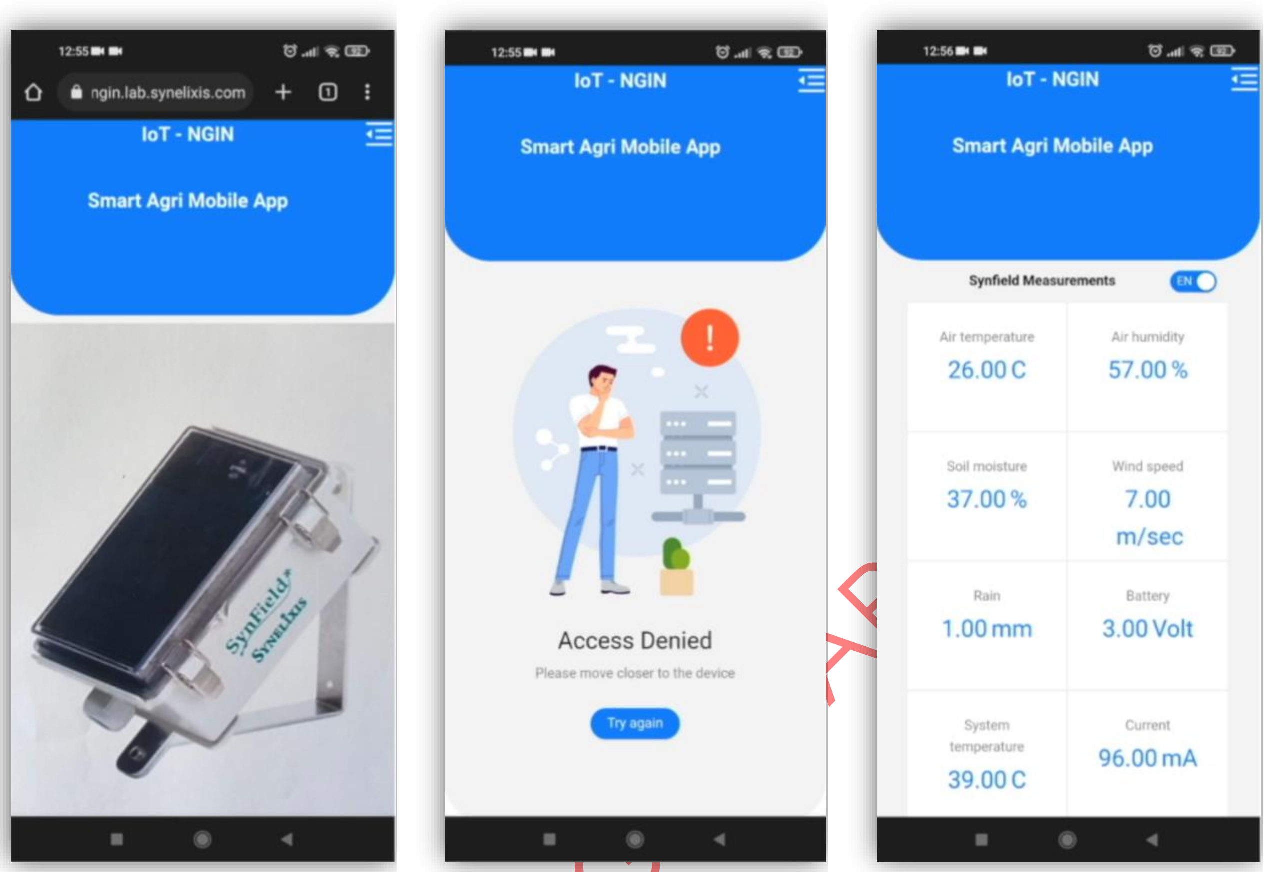
(b)

Figure 17 – Scanning a QR code (a) and informing about successful scan result (b) in the Smart Agri app.

The Smart Agri App has been developed as a React [24] web application, which interfaces with the IDD Code Scanning, IDD – Computer Vision and IDAC components. Indicative screens of the app are provided. The login and logout options are presented in the screens of Figure 16. This functionality relies on the Keycloak plugin of IDAC. Next, the screens supporting the QR scanning operation are presented in Figure 17 for a successful scan. This is supported by IDD – QR.

Next, the IDD -Computer Vision functionality is triggered by the screen of Figure 18(a). In the second part of the same figure, the user is informed that access is denied and is prompted to move closer to the device. This indicates that the proximity constraints set for the requester and the device are not satisfied. Last, the (c) part of Figure 18 illustrates the current version of the screen after successful access. The data are SynField data of the requested device, which have been provided through IDI.

In the next version of the Smart Agri app, the AR functionality will be incorporated for data presentation and device actuation.



Access Denied

Please move closer to the device

Try again

Synfield Measurements

Air temperature 26.00 C

Air humidity 57.00 %

Soil moisture 37.00 %

Wind speed 7.00 m/sec

Rain 1.00 mm

Battery 3.00 Volt

System temperature 39.00 C

Current 96.00 mA

(a) (b) (c)

Figure 18 – Acquiring image for device recognition (a), screen of denying access due to proximity criterion not satisfied (b), and (c) SynField Data presentation when access is granted in the Smart Agri app.

3.2.1.1.2 Crop disease prediction

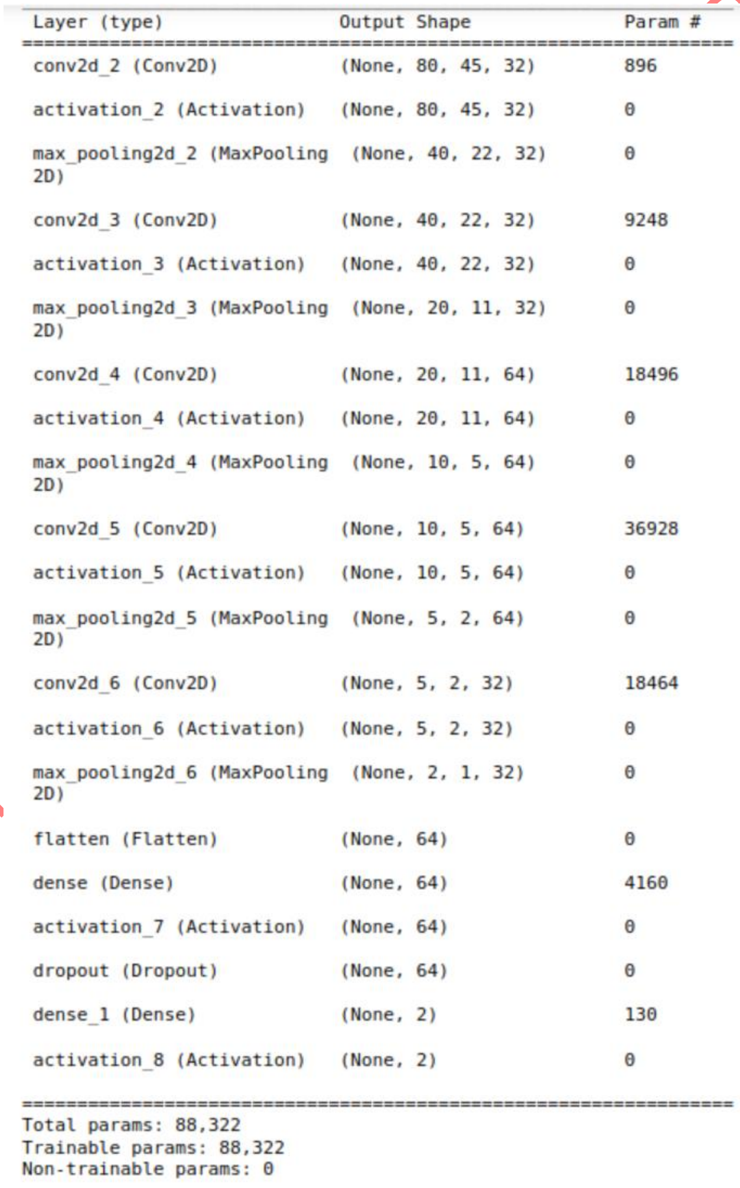
The Crop Disease Prediction component leverages an AI service which is based on an ML model for recognition of Esca [25] disease in vineyards. In this section, the training process followed for the Convolutional Neural Network (CNN) model created for Esca disease recognition is described. Specifically, will be a breakdown of the dataset selection, the architecture of the CNN model, and the model's performance.

Dataset Selection

The dataset used is the ESCA dataset used in [26]. The dataset provides a set of RGB (Red, Green, Blue) images of grapevine leaves. The images consist of two different classes. The first class belongs to the grapevine leaves that are infected by the Esca disease and the second belongs to the healthy grapevine leaves. The dataset contains 1770 images in total where 888 belong to the Esca class and the rest 882 images belong to the healthy class. To increase the size of the dataset M. Alessandrini et al. [26] suggest applying data augmentation to the images. Specifically, they used the original images and applied 14 different data augmentation techniques (e.g., horizontal flip, blur, saturation, brightness, etc.) to increase the amount of data. Therefore, after the data augmentation process, the final size of the dataset is 12,432 for the Esca class and 12,348 for the healthy class.

Model Architecture

The model that was used for the training process is a CNN model. The CNN architecture consists of 5 Convolutional 2D layers followed by ReLu activation function and 5 2D Max Pooling with a 2 x 2 pool size. In the final stage, a flatten, two dense layers, with ReLu and softmax activation functions have been attached respectively to classify the provided input training images. Between the ReLu and softmax activation function also a dropout layer has been added. Figure 19 shows the model architecture with 80 x 45 as the input size of the images. At this point, it should be mentioned that for the model training a reduction to image resolution from 1280 x 720 to 80 x 45 was applied. The reduction of image resolution offers the opportunity to deploy the model to low-cost devices with low-resolution camera and reduced image size. Moreover, there is a trade-off when the resolution of the image decreases but the difference in the performance from 1280x70 to 80x45 resolution of the images in our case is about 4% accuracy.


```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 80, 45, 32)        896

 activation_2 (Activation)   (None, 80, 45, 32)        0

 max_pooling2d_2 (MaxPooling  (None, 40, 22, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 40, 22, 32)        9248

 activation_3 (Activation)   (None, 40, 22, 32)        0

 max_pooling2d_3 (MaxPooling  (None, 20, 11, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 20, 11, 64)        18496

 activation_4 (Activation)   (None, 20, 11, 64)        0

 max_pooling2d_4 (MaxPooling  (None, 10, 5, 64)        0
 2D)

 conv2d_5 (Conv2D)           (None, 10, 5, 64)         36928

 activation_5 (Activation)   (None, 10, 5, 64)         0

 max_pooling2d_5 (MaxPooling  (None, 5, 2, 64)         0
 2D)

 conv2d_6 (Conv2D)           (None, 5, 2, 32)          18464

 activation_6 (Activation)   (None, 5, 2, 32)          0

 max_pooling2d_6 (MaxPooling  (None, 2, 1, 32)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 activation_7 (Activation)   (None, 64)                0

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

 activation_8 (Activation)   (None, 2)                 0

=================================================================
Total params: 88,322
Trainable params: 88,322
Non-trainable params: 0
```

Figure 19 – Architecture of the CNN model.

Model training and performance

Before the training process data split was carried out. From the dataset 60% was used for training, 15% for validation, and the rest 25% for testing. shows information about the size of the dataset and the batch size. The model was trained on the NVIDIA TITAN X GPU (Graphics processing unit) with 32 GB RAM (Random-access memory) and the duration of the model training lasted about an hour. The number of epochs used to train the network is 50 for all the experiments with 32 batch size.

Table 1 – Information about batch size and split of dataset.

Epochs	Batch size	Total size	Training dataset	Testing dataset	Valitation dataset
50	32	24.780	14.868	6.195	3.717

The accuracy of the model reached 0.9667 on the test data. Figure 20 visualises the performance of the model during every 5 epochs.

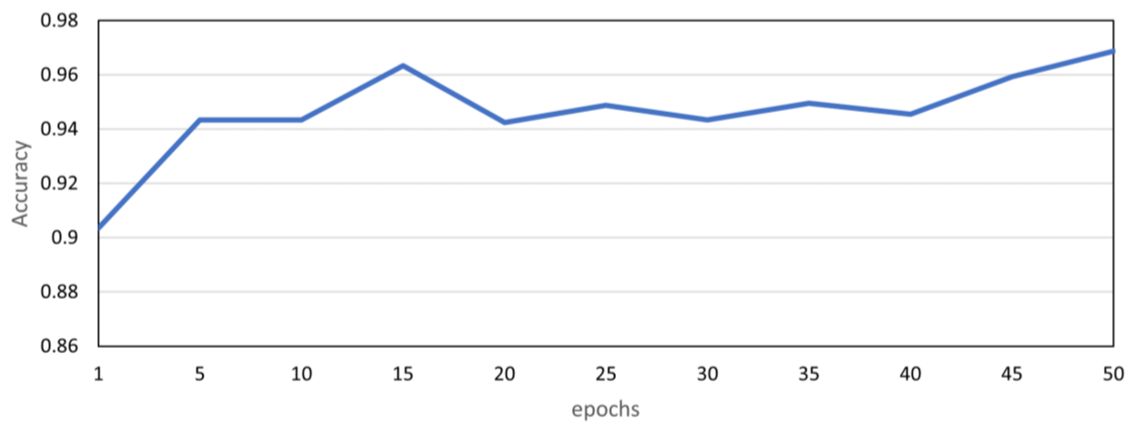


Figure 20 – Performance of the model.

Since the training model process is completed, with satisfactory results, an inference script is created that receives a single image as input or a directory of images. Image preprocessing, such as converting images to tensors, is applied to the images, and afterward the script loads the trained model and predicts if the grapevine leaves of the image are healthy or not.

3.2.2 UC5 - Sensor aided crop harvesting

UC5 - Sensor aided crop harvesting aims to enhance the harvesting process in Smart Farms. Specifically, AGLVs will alleviate the farmers from the heavyweight task of carrying full crates, while ensuring efficient and safe movement of them within the farm, effectively avoiding obstacles, like trees, or humans which may be in the farm.

The operation of the AGLV leverages the IoT-NGIN capabilities, as depicted in Figure 21. The data sources here are the sensors of the AGLV, e.g., camera, ultrasonic sensor, etc. Acquired data, that are used for training an ML model for object detection, are stored at the IDI component residing at the edge.

An AI developer may trigger model deployment through the MLaaS platform, which is then sent on the AGLV through IDI, after passing access controls by IDAC. Upon reception of the new model, the AGLV stops the already executed one and runs the new model. Then, it uses the model to detect obstacles and identify potential collisions while moving in the farm.

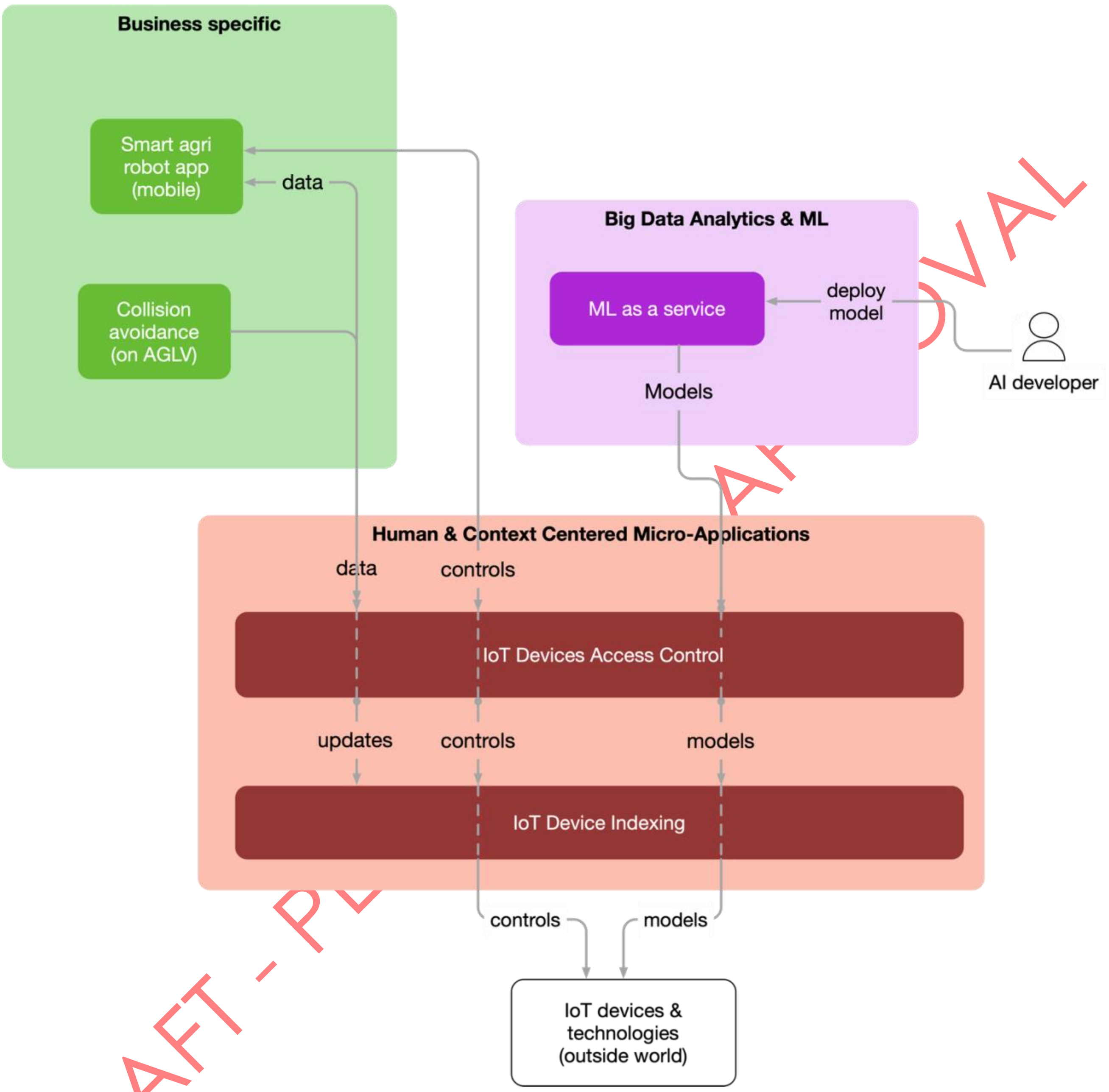

Business specific
Smart agri robot app (mobile)
data
Collision avoidance (on AGLV)
Big Data Analytics & ML
ML as a service
deploy model
Models
AI developer
Human & Context Centered Micro-Applications
data
controls
IoT Devices Access Control
updates
controls
models
IoT Device Indexing
controls
models
IoT devices & technologies (outside world)

Figure 21 – Architecture instantiation for UC5.

3.2.2.1 Business-specific components

UC5 includes two business specific components, which are required for performing the pilot activities. First, a "Smart Agri robot app" is used to manage and present information about the AGLV, in order to facilitate testing. Moreover, the "Collision Avoidance" service is mandatory to identify potential collisions, based on the detected objects.

3.2.2.1.1 Smart Agri robot application

The Smart Agri robot application is mainly aimed for supporting the test activities. It provides options to the user to manually control the AGLV wheels, as well as set device configurations and receive monitoring information at the test stages. App screenshots have been already provided in D7.2 for the presentation of the preparation activities for UC5.

3.2.2.1.2 Collision avoidance

As the goal of UC5 is to deliver an Automated Guided Land Vehicle (AGLV), which is able to move autonomously within an orchard, carrying the harvested crops, the Collision Avoidance service is the main driver for this use case. The service is based on an ML model for object detection, which has been trained and stored in the MLaaS model storage. The model per se has been developed within WP3 and will be further enhanced with the availability of datasets from the pilot site. The Collision Avoidance service is intended to run on the device (AGLV), together with the trained model. Based on the ML predictions regarding the detection of humans or trees, the Collision Avoidance service will trigger control actions on the AGLV, leading to a change in the AGLV's trajectory, in order to avoid the potential collision, but also to be able to reach the set destination.

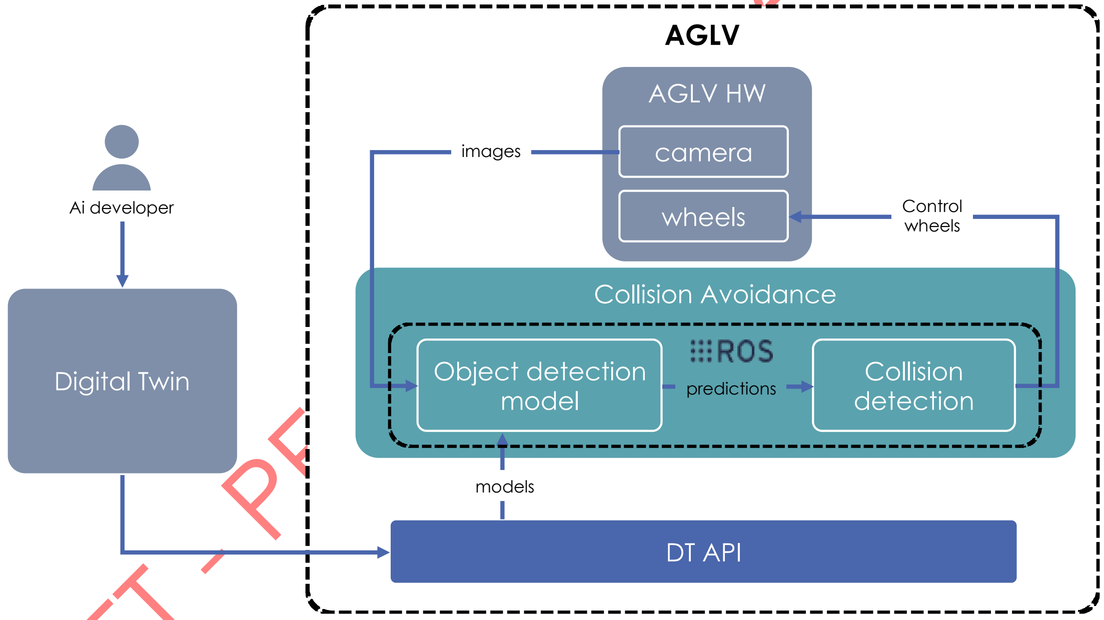


Figure 22 – Collision avoidance service for UC5.

The following operations are carried out by the Collision Avoidance service, leveraging on the Robot Operating System (ROS) [27]:

- Receive and deploy ML model
- Receive images from AGLV's camera
- Detect objects
- Identify potential collision, based on ML object detection and SLAM
- Send control commands to the AGLV's wheels

As a first step of the development of the Collision Avoidance service, the retrieval of ML models stored in MLaaS platform (at cloud resources) has been already completed and validated on the AGLV hardware. Following the logic of IoT-edge-cloud integration presented in section 2.2.1, the ML model URL has been communicated to the AGLV hardware through the Digital Twin.

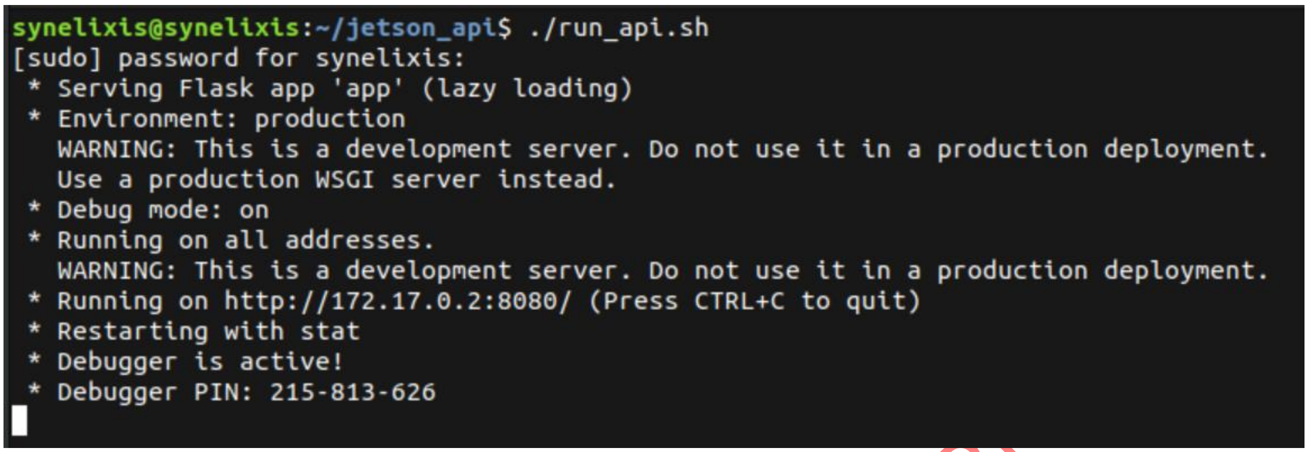
First, the AGLV has been registered in the *Devices* of the Device Indexing component, by issuing a POST request to IoT Agent of IDI at the endpoint URL http://{iot-agent}/iot/devices, with the following body.

```
{
   "devices": [
      {
         "device_id": "jetson6",
         "entity_name": "urn:ngsi-ld:Jetson:4",
         "entity_type": "Jetson",
         "endpoint": "xxxx",
         "protocol": "IoTA-JSON",
         "transport": "HTTP",
         "commands": [
            {
               "name": "url",
               "type": "command"
            }
         ]
      }
   ]
}
```

Next, the URL of the ML model, stored in MLaaS Minio service, is communicated to the AGLV through the IoT Agent. For the purposes of this test, a Minio installation in Synelixis premises has been used. The URL is provided to the AGLV by issuing a POST request at the IDI service accessible at the endpoint URL https://{orion-url}/v2/op/update, with the following request body.

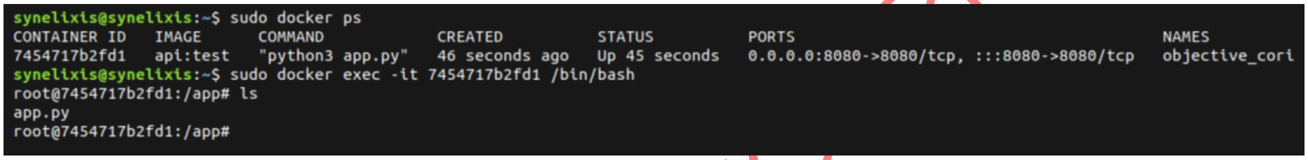
```
{
    "actionType": "update",
    "entities": [
        {
            "type": "Jetson",
            "id": "urn:ngsi-ld:Jetson:4",
            "url": {
                "type": "command",
                "value": "minio/testminio/test.txt"
            }
        }
    ]
}
```

In the following, the logic developed at the AGLV side for receiving updates like the above is illustrated via screen captures. A python script is initiated in Figure 23 which executes the docker container of the DT API, implemented as a FLASK app. The docker container execution is shown in Figure 24.



```
synelixis@synelixis:~/jetson_api$ ./run_api.sh
[sudo] password for synelixis:
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 215-813-626
```

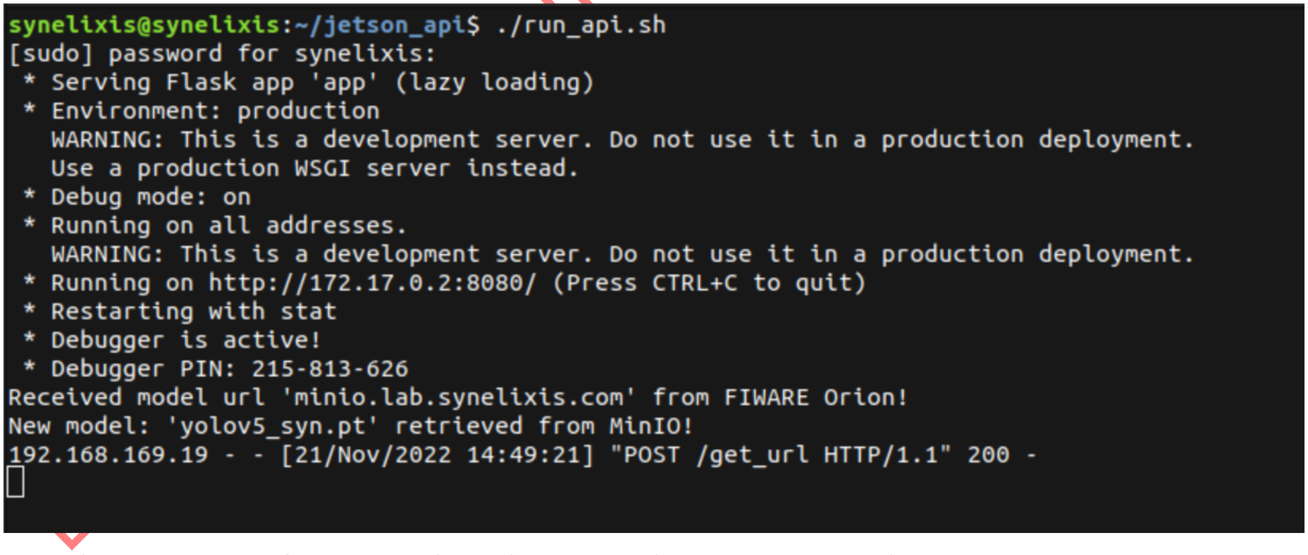
Figure 23 – Initiation of DT API at the AGLV hardware.



```
synelixis@synelixis:~$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND            CREATED          STATUS          PORTS                                       NAMES
7454717b2fd1   api:test   "python3 app.py"   46 seconds ago   Up 45 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   objective_cori
synelixis@synelixis:~$ sudo docker exec -it 7454717b2fd1 /bin/bash
root@7454717b2fd1:/app# ls
app.py
root@7454717b2fd1:/app#
```

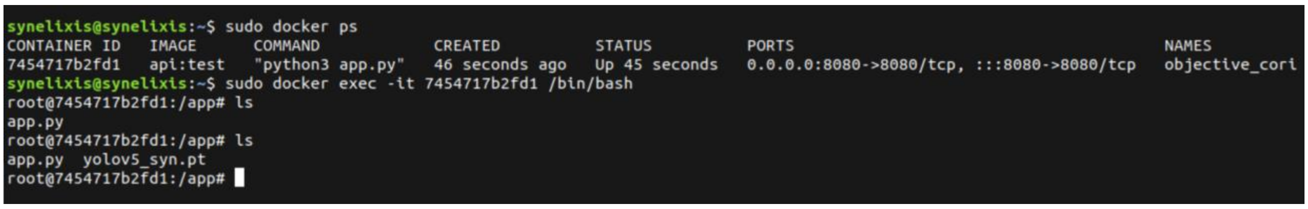
Figure 24 – Execution of Docker container of the DT API at the AGLV hardware.

With the DT API running, it is now possible to receive updates from the IDI component. The request providing the URL of a new ML model has been received, as shown in Figure 25. Then, the DT API retrieves the model from the provided URL and deploys it locally. The last step is shown in Figure 26.



```
synelixis@synelixis:~/jetson_api$ ./run_api.sh
[sudo] password for synelixis:
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 215-813-626
Received model url 'minio.lab.synelixis.com' from FIWARE Orion!
New model: 'yolov5_syn.pt' retrieved from MinIO!
192.168.169.19 - - [21/Nov/2022 14:49:21] "POST /get_url HTTP/1.1" 200 -
```

Figure 25 – URL of ML model for object detection has been received by DT API at the AGLV hardware.



```
synelixis@synelixis:~$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND            CREATED          STATUS         PORTS                                       NAMES
7454717b2fd1   api:test   "python3 app.py"   46 seconds ago   Up 45 seconds  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   objective_cori
synelixis@synelixis:~$ sudo docker exec -it 7454717b2fd1 /bin/bash
root@7454717b2fd1:/app# ls
app.py
root@7454717b2fd1:/app# ls
app.py  yolov5_syn.pt
root@7454717b2fd1:/app# █
```

Figure 26 – ML model for object detection has been deployed at the AGLV hardware.

3.3 Industry 4.0 Living Lab

3.3.1 UC6 - Human-centred safety in a self-aware indoor factory environment

This UC aims to improve the efficiency of processes within Industry 4.0 in relation to:

- The autonomous movement of Automated Guided Vehicles (AGVs). AGVs should be able to move between source and destination safely, avoiding collisions with human workers or Human Driven Vehicles (HDV) present at the industry facilities while AGVs are moving, but also avoiding falling of containers during emergency braking.
- Enhancing the presentation of information related to such AGVs, providing Augmented Reality (AR) interfaces which will provide information about AGVs, HDV and humans, as well as about potential collisions.

IoT-NGIN contributes to the concept of Ambient Intelligence (AmI) by adopting and developing a set of components and modules, enablers henceforth, that strategically integrated aim at providing innovative beyond state-of-the-art solutions. In the context of the Industry 4.0 Living Lab, and its associated "Human-centered safety in a self-aware indoor factory environment" Use Case, four types of enablers can be highlighted.

IoT Device Discovery (IDD): These include two main categories of methods using specific sensors, hardware setups and software solutions to discover and detect specific IoT sensors or elements: (i) visual methods, like advanced computer vision based recognition methods from external cameras and based on trained models for specific dynamic objects (e.g., vehicles, persons, carts, etc.), simpler methods from built-in cameras on smartphones or Augmented Reality (AR) headsets for static visual targets, and even methods based on Visual Light Positioning (VLP) using cameras and LED lamps; and (ii) non-visual methods, based on Ultra Wide Band (UWB) trilateration using beacons and tags.

IoT Device Indexing (IDI): This module allows registering and retrieving information (e.g., characteristics, status, etc.) related to the target sensors or elements being detected or to be detected.

IoT Device Access Control (IDAC): This module can grant or reject access to the information registered in the IDI module, based on different criteria (e.g., authentication, roles, proximity, etc.).

IoT AR Service (IAR): This service interfaces with the IDI module to get the information of detected sensors. It also implements methods to present the obtained information and interaction modalities in an intuitive and meaningful manner, minimizing the cognitive load and attention split.

By integrating these enablers and leveraging their distinct but complementary features, two business-specific components have been envisioned to meet the requirements of this use case, namely the "Collision Detection" and the "Device app".

IoT-NGIN supports UC6 through the architecture instantiation of Figure 27. This UC focuses on three variants of IDD, namely the VLP, the UWB and the Computer Vision variant for the recognition of objects and other entities such as humans, which are registered in IDI, after IDAC controls, The ML model for the Computer Vision variant is provided through the MLaaS platform to IDD, which then uses this model to perform predictions related to the detection of objects. The output of all three IDD variants is used as input to the Collision Detection component. Moreover, the AR functionality is provided to the user via the "Device app", running on the worker's mobile phone, tablet or headset.

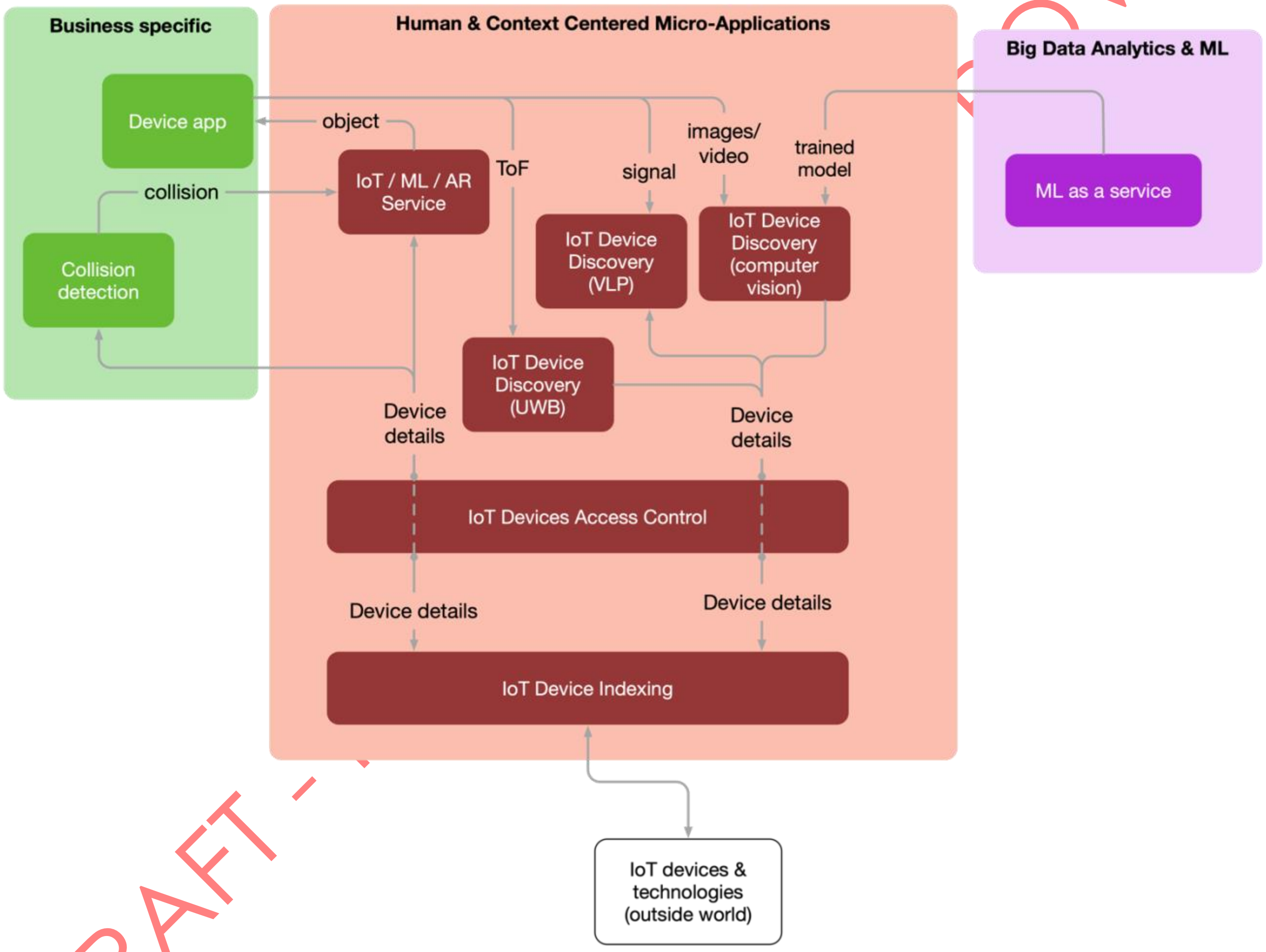


Figure 27 – Architecture instantiation for UC6.

3.3.1.1 Business-specific components

In the following subsections, the business specific components for UC6 are briefly presented.

3.3.1.1.1 Device application

Device application interfaces the IDI and IDAC modules in order to detect and get information from specific sensors or elements. In some cases, it can also implement specific

IDD methods, like detection of visual targets using an AR camera or QR code scanning. Such features for information presentation and interaction are provided by an instance of the IAR component running on the device application. The information can be provided just once under request, periodically, or in an event-driven manner, depending on the needs and/or preferences, and it is presented as an AR overlay (ideally close to the detected element), such that attention can still be kept on the explored real environment. In addition, the device application can also incorporate the required interfaces to actuate target sensors, via the IDI and IDAC modules.

3.3.1.1.2 Collision detection

All IDD methods provide the sensors' / elements' positions, even in indoor scenarios where GPS coverage is not available, by relying on a relating coordinates system. In the same manner, it is also possible to obtain the position of the AR device, running the Device Application component, by means of: (i) attaching an UWB tag to it; (ii) being detected by a computer vision-based method (e.g., as it will be used by a person); (iii) including an own ad-hoc positioning system; or combinations thereof. With the information especially, positions) from all detected sensors / elements, it is then possible to implement a Collision Detection module to provide: (i) a comprehensive situational awareness of the sensors / elements being active or detected environment (i.e. the factory in this UC), by e.g. displaying their position on a map, even by detecting those elements / sensors outside the user's field of view (e.g., behind walls), and potentially filtered by configurable distance thresholds; and (ii) provide alarms, and potentially actuate sensors (e.g., stop a moving element) in case that proximity thresholds are exceeded between specific types of sensors (e.g., two AGVs, an AGV and a person, etc.). Such information and features provided by Collision Detection thus contribute to an enhanced safety in and Industry 4.0 environments.

It is important to remark that the particular implementation for UC6 consists of deploying and running the IAR service and Collision Detection module as part of the Device application component. However, they have been described in separate subsections given that specially the Collision Detection module could be deployed and run e.g., on the Edge and interface other presentation and interaction methods (e.g., a 2D dashboard), depending on the needs and/or resources available for each particular implementation.

3.3.2 UC7 - Human-centred augmented reality assisted build-to-order assembly

The "Human-centred augmented reality assisted build-to-order assembly" use case (UC7) in ABB's facilities involves the assembly and wiring of ABB's drive cabinet products. The aim of this use case is to facilitate the assembly and wiring process of ABB's drive cabinet products through:

- 3D digital cabinet model for precise cabinet builds
- detailed, up-to-date documentation for all components and wiring routes, available at the physical site
- advanced human-machine (AR) interfaces to visualise assembly instructions.

IoT-NGIN hosts this use case through the architecture depicted in Figure 28. The 3D cabinet models are registered and retrieved via the IDI component, after passing authorization checks via the IDAC component. The models are retrieved by the AR service, which provides the appropriate guidance via AR tools. The information is visualised at the Device app at the assembly site.

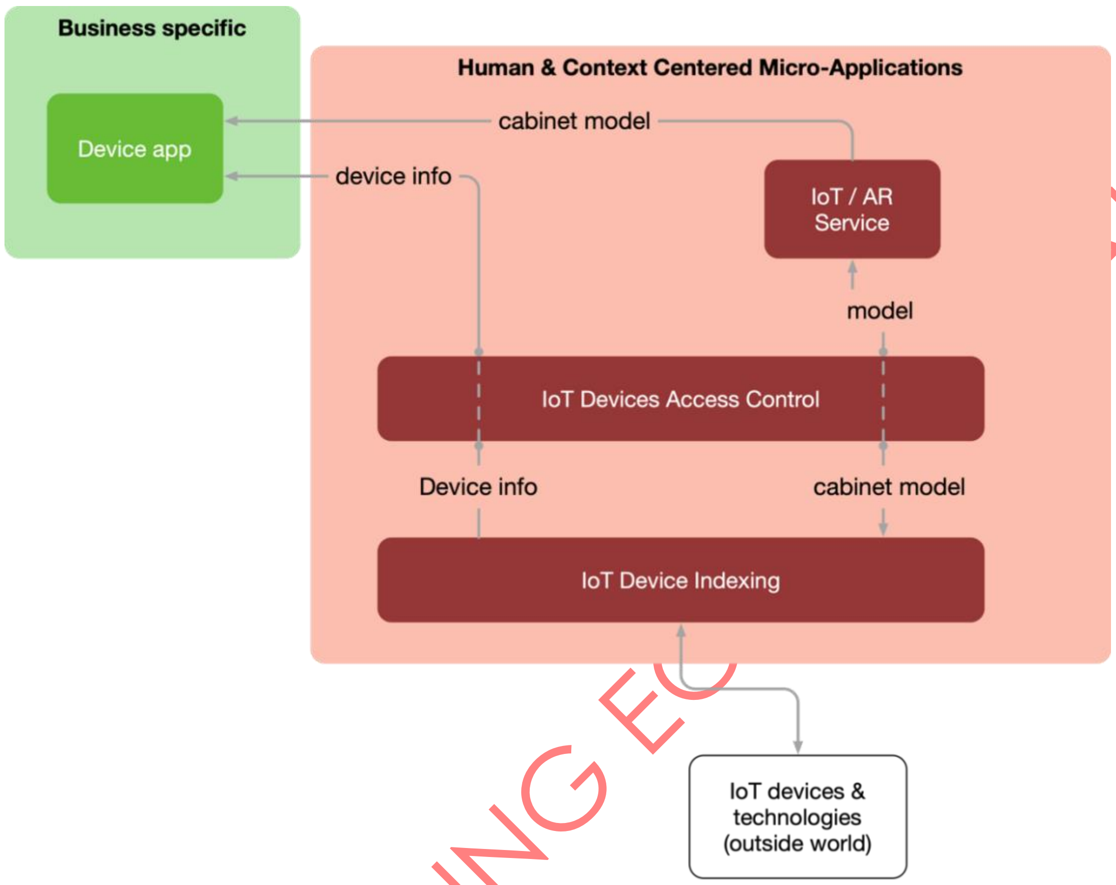


Figure 28 – Architecture instantiation for UC7.

3.3.2.1 Business-specific components

3.3.2.1.1 Device application

Digital models of the cabinets are developed which contain both mechanical and electronic CAD data. These digital models are used to visualise the different assembly phases to the assembly worker in the proprietary Smart Wiring™ software created by EPLAN. The models are also used to create an AR application for training, sales, and/or maintenance purposes. AR device application for HoloLens2/Mobile includes the following features: QR code scanning, tracking and model interactions.
Digital models and AR solutions has several benefits. Training of assembly works gets easier and more effective, productivity and quality of assembly work will increase and use of material can be reduced, for example.
The use case will take place at ABB's Helsinki factory site in the cabinet production facility. The main UC component is the digital model created by combining MCAD data (generated by Creo™ software) and ECAD data (generated by EPLAN's software suite). The digital model is then used to visualise assembly steps in EPLAN's Smart Wiring™ software, and to create an AR application utilizing the repository of software tools provided by T4.4.

3.3.3 UC8 - Digital powertrain and condition monitoring

UC8 "Digital powertrain and condition monitoring" aims to leverage IoT-devices, 5G telecommunication and cloud platforms to utilise novel ideas in the area of data engineering, analytics and condition monitoring. Specifically, these will be based on powertrain application data, i.e., data gathered from equipment involved in transforming energy provided by a power source into useful work.

In IoT-NGIN, UC8 is materialised through the architecture presented in Figure 29. Device registration and updates are supported by IDI and IDAC components. Last, but not least, the powertrain data are made discoverable and available through the Semantic Twin for the powertrain devices.

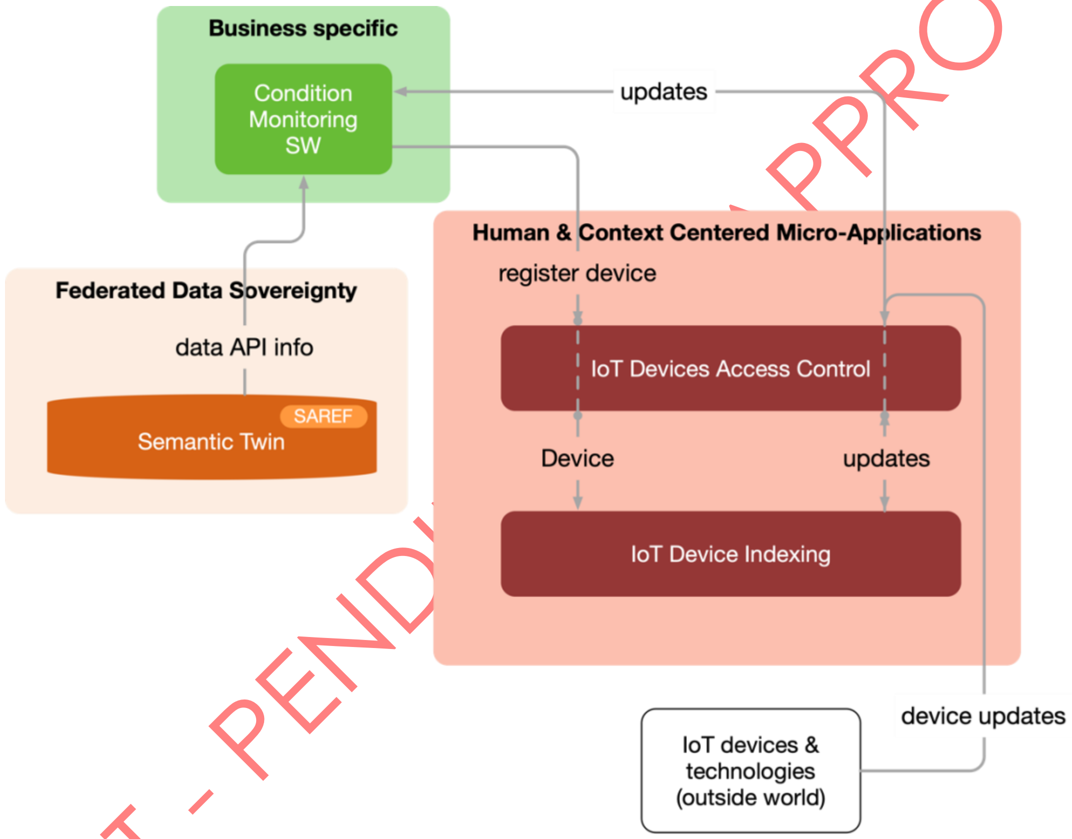


Figure 29 – Architecture instantiation for UC8.

3.3.3.1 Business-specific components

3.3.3.1.1 Condition monitoring software

In the context of this use case, a powertrain includes an AC motor and a variable speed drive responsible for its control. The goal in this use case is to leverage IoT-devices, 5G telecommunication and cloud platforms to utilise novel ideas in the area of data engineering, analytics and condition monitoring. The goal is to create a holistic view of the condition and status of each powertrain, especially the drive unit and the motor itself. Instead of using a traditional data-siloed site-specific approach, a decentralised and federated approach is taken, leveraging the IoT-NGIN paradigm and technologies.

A condition monitoring application needs to be able to access the sensor data gathered from powertrains in order to produce analytics results that can be used to monitor the condition of the devices.

The sensors and devices themselves are not directly capable of running any additional software. Thus, data is gathered to a RaspberryPi or Cassia gateway using available device-specific protocols (OPC DA/UA, plain Transmission Control Protocol (TCP) byte stream, MQTT, File Transfer Protocol (FTP)). The data can then be accessed from the gateway using any protocol of choice. Currently, the gateway device is running Node-RED which can be used to easily create endpoints of preferred protocol/format e.g., MQTT.

The various devices are connected to the gateway using a private 4G network and additional 4G capable gateway modems are used where needed (most sensor devices do not have built-in 4G/5G capabilities).

Twin description documents are created for the powertrains and sensors of the use case. The documents are used in application development, abstracting the underlying protocols used for a specific powertrain setup. More details about the solution are available in the deliverable D5.4.

3.4 Smart Energy Living Lab

3.4.1 UC9 Move from reacting to acting in smart grid monitoring and control

UC9 "Move from reacting to acting in smart grid monitoring and control" aims at monitoring and optimizing the grid operation. Specifically, the IoT-NGIN tools are employed to support the DSO towards the following:

- Forecasting of energy generation and consumption for the day ahead
- Optimise the grid operation, suggesting consumption pattern adaptation to avoid over- or under-voltage grid situations.
- Integrate Demand Response capability.

IoT-NGIN offers these capabilities through the architecture instantiation depicted in Figure 30. As shown in the figure, network data are forwarded to IDI and dispatched to both the IoT Vulnerability Crawler and the Malicious Attack Detector (MAD). The former scans for potential vulnerabilities, which are communicated to IDI as vulnerability reports, while the latter performs anomaly detection to identify potential attacks on the network level. Energy generation and consumption forecasting rely on ML models trained in the MLaaS platform, leveraging the Online Learning framework for training. Data coming from grid IoT devices are used as input for both the training and the inference processes of these services. Last, but not least, UC9 addresses data privacy and sovereignty, integrating the Semantic Twin and SSIs for facilitating device discoverability.

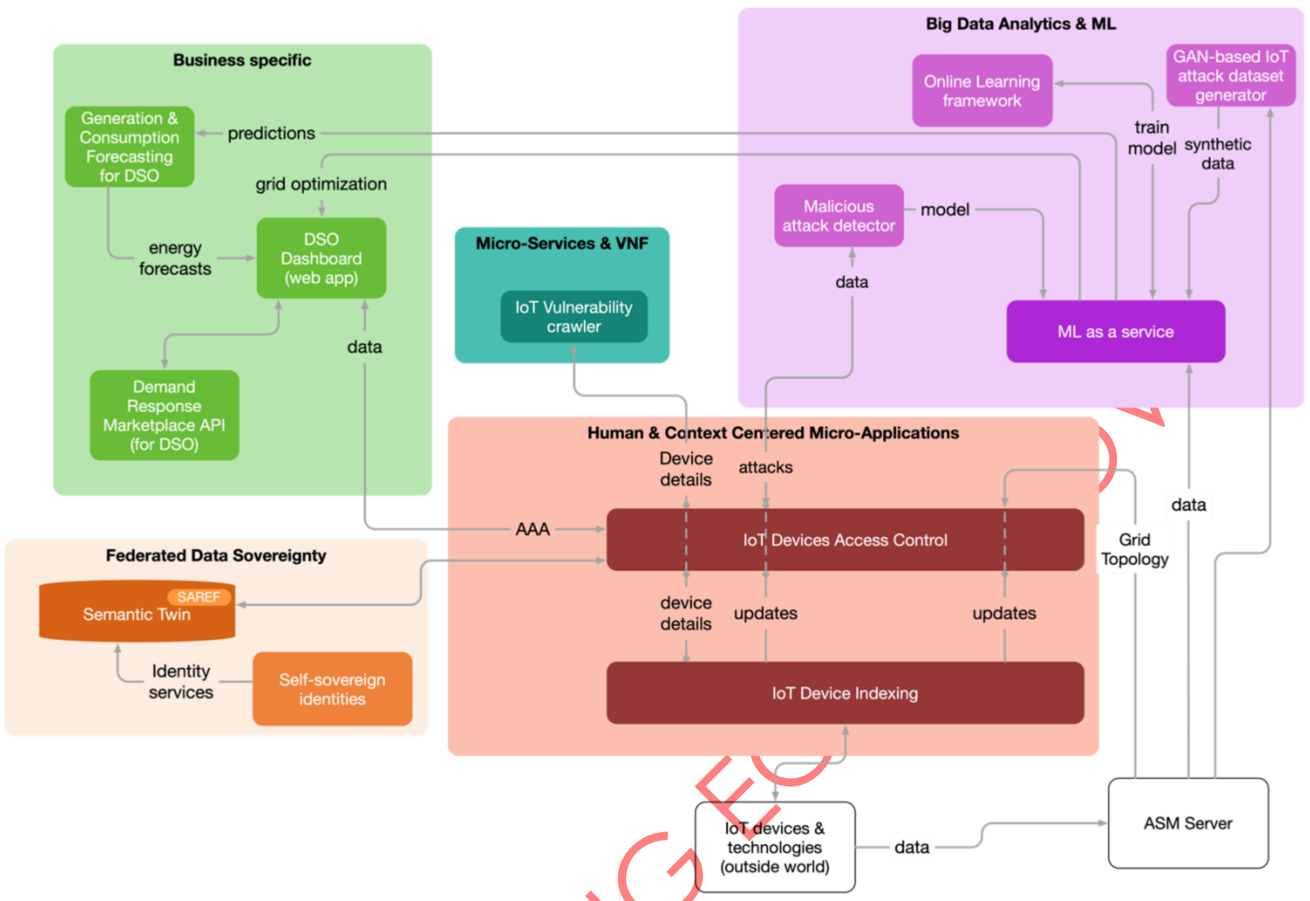

Big Data Analytics & ML
Online Learning framework
GAN-based IoT attack dataset generator
train model
synthetic data
Malicious attack detector
model
data
ML as a service
Business specific
Generation & Consumption Forecasting for DSO
predictions
grid optimization
energy forecasts
DSO Dashboard (web app)
Demand Response Marketplace API (for DSO)
data
Micro-Services & VNF
IoT Vulnerability crawler
Human & Context Centered Micro-Applications
Device details
attacks
AAA
IoT Devices Access Control
device details
updates
updates
IoT Device Indexing
data
Grid Topology
Federated Data Sovereignty
SAREF
Semantic Twin
Identity services
Self-sovereign identities
IoT devices & technologies (outside world)
data
ASM Server

Figure 30 – Architecture instantiation for UC9.

3.4.1.1 Business-specific components

In the following subsections, the business specific components for UC9 are presented.

3.4.1.1.1 DSO dashboard

DSO dashboard is a web platform developed by ENG in which the historical and real-time data of the monitored smart meters will be shown as well as the main DSO outcomes, namely:

- Self consumption rate (%)
- Self sufficiency rate (%)
- Reverser power flow (kWh)
- Flexible energy shifted (kWh)
- Voltage issues (#)
- RES used for EV charging (%)

Furthermore, DSO dashboard is developed to host the Generation and Consumption Forecasting service, together with the Grid Operation Optimization service, and Demand Response marketplace API.

3.4.1.1.2 Generation and consumption forecasting for DSO

In the context of the Smart Energy LL UC9 "Move from Reacting to Acting in Smart Grid Monitoring" forecasting services are required, in particular, the *Power Generation Forecasting and* the *Power Consumption Forecasting*. These services predict the behaviour of certain parameters of the electric grid. The grid includes metering devices to measure such parameters like the electric power, voltage or current, among others, in distinct nodes of the grid and publish them all in a MQTT broker.

Power Generation Forecasting and Power Consumption Forecasting services have been designed, implemented and tested in both UC9 as a joint collaboration between WP6 T6.2 and WP3 T3.2, by leveraging the MLaaS Online Learning framework. They have been integrated as part of the WP3 MLaaS platform and its technical details preliminary reported in D3.3. Currently they are being integrated into the Smart Energy dashboard.

These forecasting services are subscribed to the MQTT broker topics where grid nodes are publishing power (generated, consumed) data. Once incoming data is received, it is pre-processed, the forecasting DL model is retrained, and a new prediction for next 24-26 hours is published back on the topic.

Since the M23 release of these forecasting services (reported in D3.3) some improvements have been implemented in the forecasting model training and prediction process, including:

- date/time frames (days of week) have been added to the training data set as new features to improve the detection of weekly variability in the prediction process for power generation and consumption,
- weather conditions at the electric grid location have been added to the training data set as new features to improve the detection of climate-driven consumer behaviour in the prediction process for power generation and consumption,
- the online learning framework these forecasting services leverage on offers a new feature that shows explainable AI (XAI) which offers explanations on the way training set features influence the performance on predictions. Preliminary results show a memory effect on the training set, whereby it is shown recent metrics have stronger influence on the prediction than metrics collected hours before.

Borrowed from D3.3, Figure 31 shows the power generation forecasting for UC9.

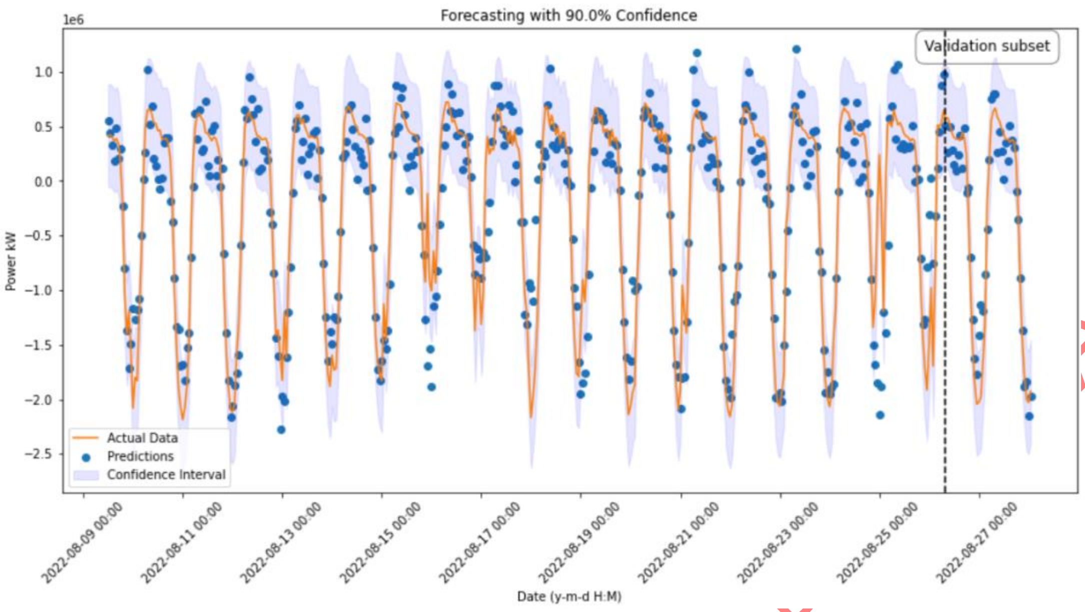


Figure 31 – Forecasting for power generation forecasting of UC9.

3.4.1.1.2.1 Grid Optimization

Smart Energy LL UC9 "Move from Reacting to Acting in Smart Grid Monitoring" requires the Grid Operation Optimization service. This service is being designed and implemented by a joint collaboration between WP3 T3.2 and WP6 T6.2. In the following we will provide a detailed specification of the Grid Operation Optimization scenario for UC9.

The Smart Energy LL uses an electric grid described in Figure 32.

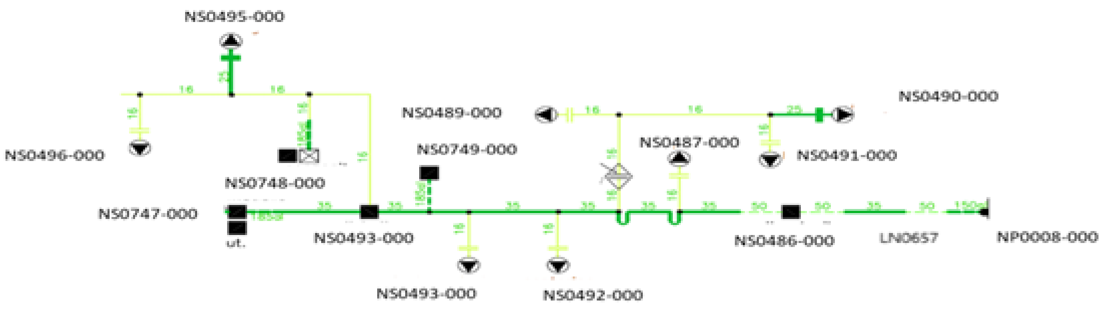


Figure 32 – Electric grid for Smart Energy LL.

The electric grid consists of 14 buses, one connected with the primary substation (NP0008-000) and the others connected with secondary substations. The primary substation injects main power to the grid, while the secondary ones provide power service to users, who may act as consumers, producers or prosumers. Each secondary substation offers power to the grid regulated by a contract for domestic and industrial users.

In this scenario the optimization process is specified by the main reinforcement learning entities, namely:

Agents

Agents, as entities that can introduce changes in the scenario environment through the application of actions, are the domestic and industrial consumers. These agents can modify their consumption behaviour by switching their peaks of higher consumption to other time slots. This can be done by reducing/increasing their power consumption in a certain percentage during time frames of high/small consumption.

States

States of the environment can be specified for the agents (I.e., the consumers) and the electric grid.

Consumers' states: the consumer states are a continuous of the power consumed over a day. In can be discretised on a range from minimum to maximum consumption, in steps of 5% for instance.

Electric grid (EG) states: the EG can transition through the following states (as described in D3.3):

- EG shows disturbances,
- EG does not show disturbances

The main disturbances EG can show are:

- Overloads: the power flowing in the transformers is greater than the rated power,
- **Over-voltages**: the voltage at a node is more than 1.1 times the nominal voltage,
- **Under-voltage**: the voltage at a node is less than 0.9 times the rated voltage.

Under these conditions the network can face serious consequences such as the damage of certain components, so circuit breakers must be opened to prevent this situation. UC9 considers over-voltages and under-voltages only, as there are no sensors to monitor the transformers or lines.

Actions

The possible actions taken by agents, that is, the domestic and industrial consumers are those that modify their consumption behaviour during the day time, concretely:

- Increase power consumption at time slot T by percentage P
- Decrease power consumption at time slot T by percentage P

P can be taken from a set of percentages [1%, 3%, 5%, 10, 15%] for example. T can be a discretization of the daytime [24 hours discretised in slots of 30 min] for example.

Rewards

For the electric grid, as described in D3.3, there are different optimization objectives, with associated rewards:

- Self-consumption ratio (SCR), defined as the ratio of the portion of the photovoltaic power-voltage (PV) production consumed by the loads over the produced energy (E_p) of the PV plant. It ranges between 0 % and 100 %. The optimization objective is to maximise self-consumption.
  - Positive reward $+R1$ if the self-consumption increases. $R1$ could be computed as proportional to the increment.
  - Negative reward $-R1$ if the self-consumption decreases. $R1$ could be computed as proportional to the decrement.
- Self-sufficiency ratio (SSR), defined as the portion of energy produced that has been consumed, out of the total energy consumed by the utility, i.e., the absorbed energy

(E_a) and the self-consumed energy. It ranges between 0 % and 100 %. The optimization objective is to maximise self-sufficiency.
  - Positive reward $+R2$ if the self-sufficiency increases. $R2$ could be computed as proportional to the increment.
  - Negative reward −R2 if the self-sufficiency decreases. $R2$ could be computed as proportional to the decrement.
- Power losses, defined as the dissipated energy across the EG. It is computed by summing over the segments of the EG of the dissipated energy by Joule effect. The optimization objective is to minimise power losses.
  - Positive reward $+R3$ if the power losses decrease. $R3$ could be computed as proportional to the decrement.
  - Negative reward −R3 if the power losses increase. $R3$ could be computed as proportional to the increment.
- Disturbances: EG should not show disturbances. High negative reward is expected if EG shows them. A better analysis of the EG disturbances is required to include this factor in the Reinforcement Learning (RL)-based optimization control.

The total reward RT in a single optimization step, starting from state $s\epsilon S$, selecting an action $a\epsilon A$, and transitioning to a new state $s'\epsilon S$ can be computed as a linear function of previously mentioned individual rewards:

$$RT = w1R1+w2R2+w3R3+fdist$$

where wi are normalised weights defined as hyperparameters of the RL model, and the function $fdist$ will compute the reward associated with the EG disturbances.

Based on the above-described scenario and the reinforcement entities required, the Grid Optimization service aims at **optimizing the consumers' demand**, as a **demand site management service** that optimises the grid behavior. The behavior is optimised by maximizing the rewards, that is:

- Maximizing the **grid SCR** and **SSR**,
- Minimizing the grid power loss,
- Discarding the occurrence of **grid disturbances**.

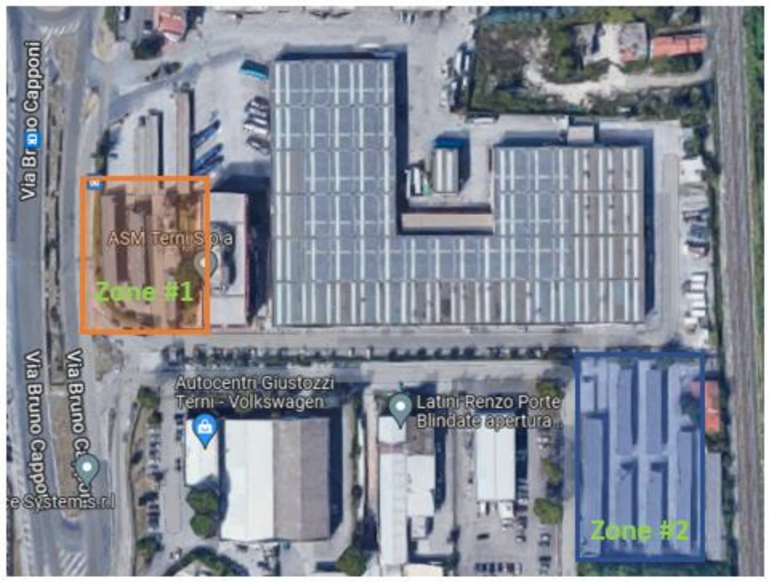
A RL-based optimization service requires a grid simulator that can:

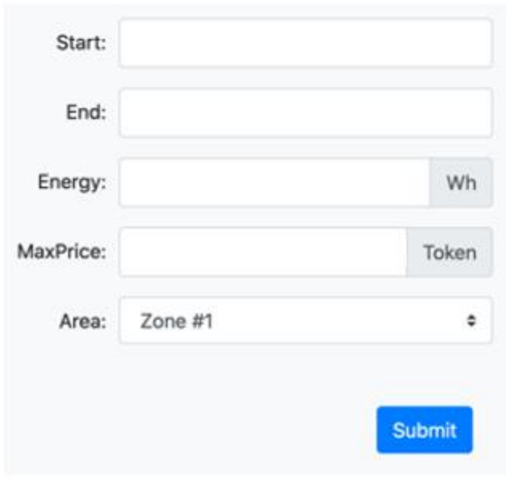
- Simulate the consumer's power consumption behavior, accepting actions to increase/decrease that consumption of specific day time frames,
- Simulate the grid behavior, after modifying the consumers behavior for a particular substation node, and return the observe state of the grid from where the RL-based optimization service can compute the grid SCR, SSR, power loss and occurrences of disturbances.

Currently, this electric grid simulator is under development. It will be used to train the RL-based optimization service, which will be released by M30 and integrated within the Smart Energy LL dashboard.

3.4.1.1.3 Demand Response marketplace API (for DSO)

Demand Response (DR) marketplace API enables P2P energy trading between DSO and energy users in an automated, decentralised and flexible way using Ethereum smart contracts; in particular DSO will be able to create a DR auction-based request (Figure 33), specifying when, where and how much energy flexibility have to be provided to improve power quality in its distribution grid.





Start:	
End:	
Energy:	Wh
MaxPrice:	Token
Area:	Zone #1

Figure 33 – DR request creation.

Figure 34 shows how an auction takes place using SOFIE [28] Marketplace component. First, the party wishing to sell something (Manager) creates a new auction, after which the potential buyers can submit their bids. Once the bidding time concludes, the Manager closes the auction and decides the winner. The winning bid can be the highest or the lowest depending on the type of auction as the Marketplace component supports different pricing models. The winner then pays for the item and the Manager delivers the item. Finally, the winner confirms successful reception of the item, thus creating a complete audit trail of the auction event. At any time, anyone with access to the Marketplace (Marketplace can be public or limited to members only) can view and audit the status of the Marketplace.

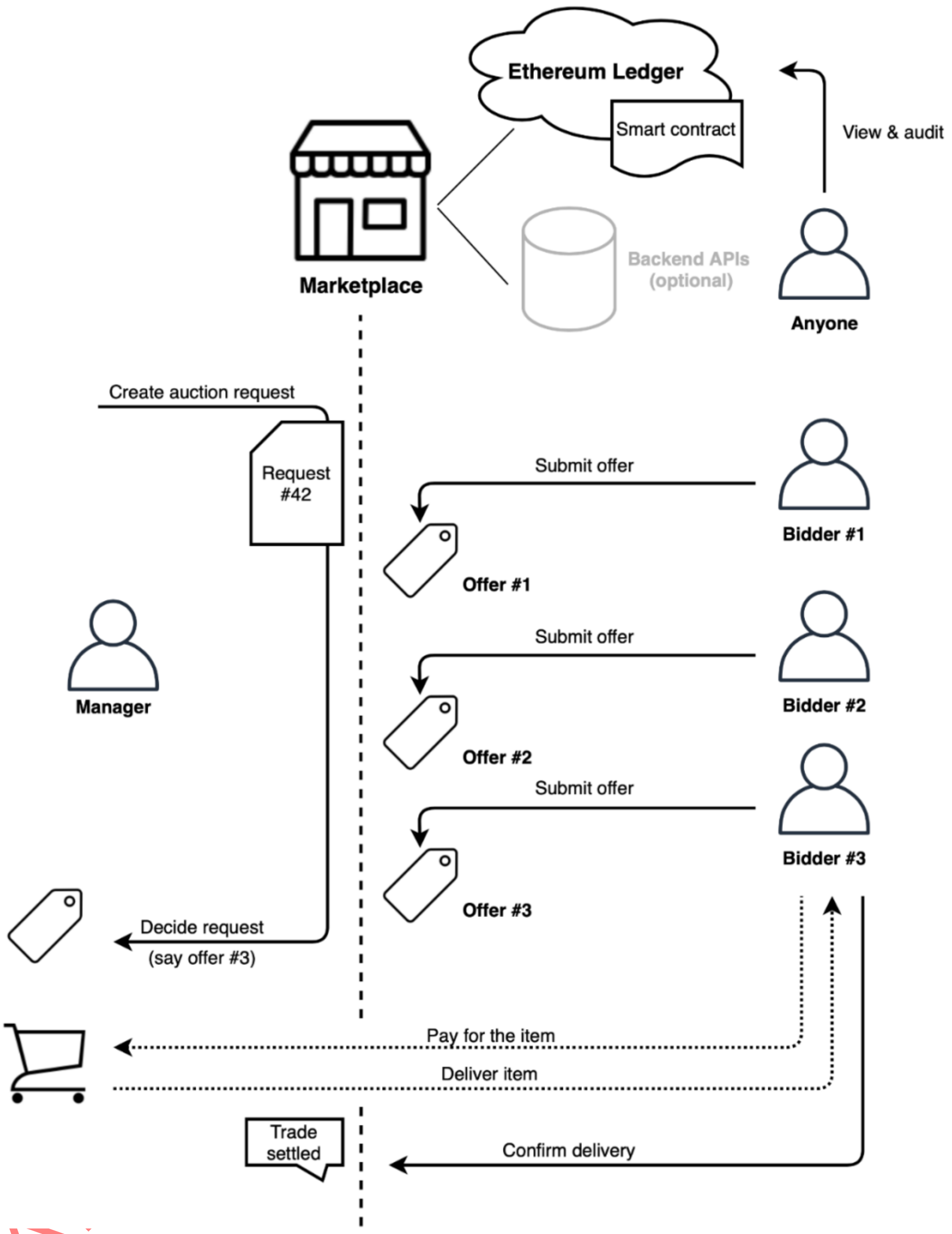


Figure 34 – Auction flow using SOFIE Marketplace component.

3.4.2 UC10 Driver-friendly dispatchable EV charging

The Driver-friendly dispatchable EV charging use case (UC10) aims at facilitating the integration of EVs into the grid and leveraging those asset's flexibility potential for responding to grid stabilization motives.

IoT-NGIN supports UC10 through the architecture depicted in Figure 35. Here, as well, network data feed vulnerability scanning and ML-based attack detection to protect the IT energy infrastructure from potential cyberattacks. Moreover, energy generation and consumption forecasting for the DSO relies, as in UC9, on the MLaaS platform, with training taking place in the Online Learning framework.

In order to leverage the charging stations' flexibility, monitoring data are forwarded to IDI, creating a Digital Twin at the edge. Such data feed the CPO dashboard which presents energy/financial efficiency information per charging station to CPOs. Moreover, EV users may visualise on-board diagnostics data related to their EV. EV users may also access charging station information through advanced interfaces, which present this information through AR objects. This also requires that the charging station is recognised through ML models offered via the MLaaS platform.

UC10 integrates a Demand Response Marketplace API for CPO, which ensures the reliability of the transactions through immutable storage of DR campaign info, supported by the "Decentralised Interledger Bridge", protected via SSI technologies.

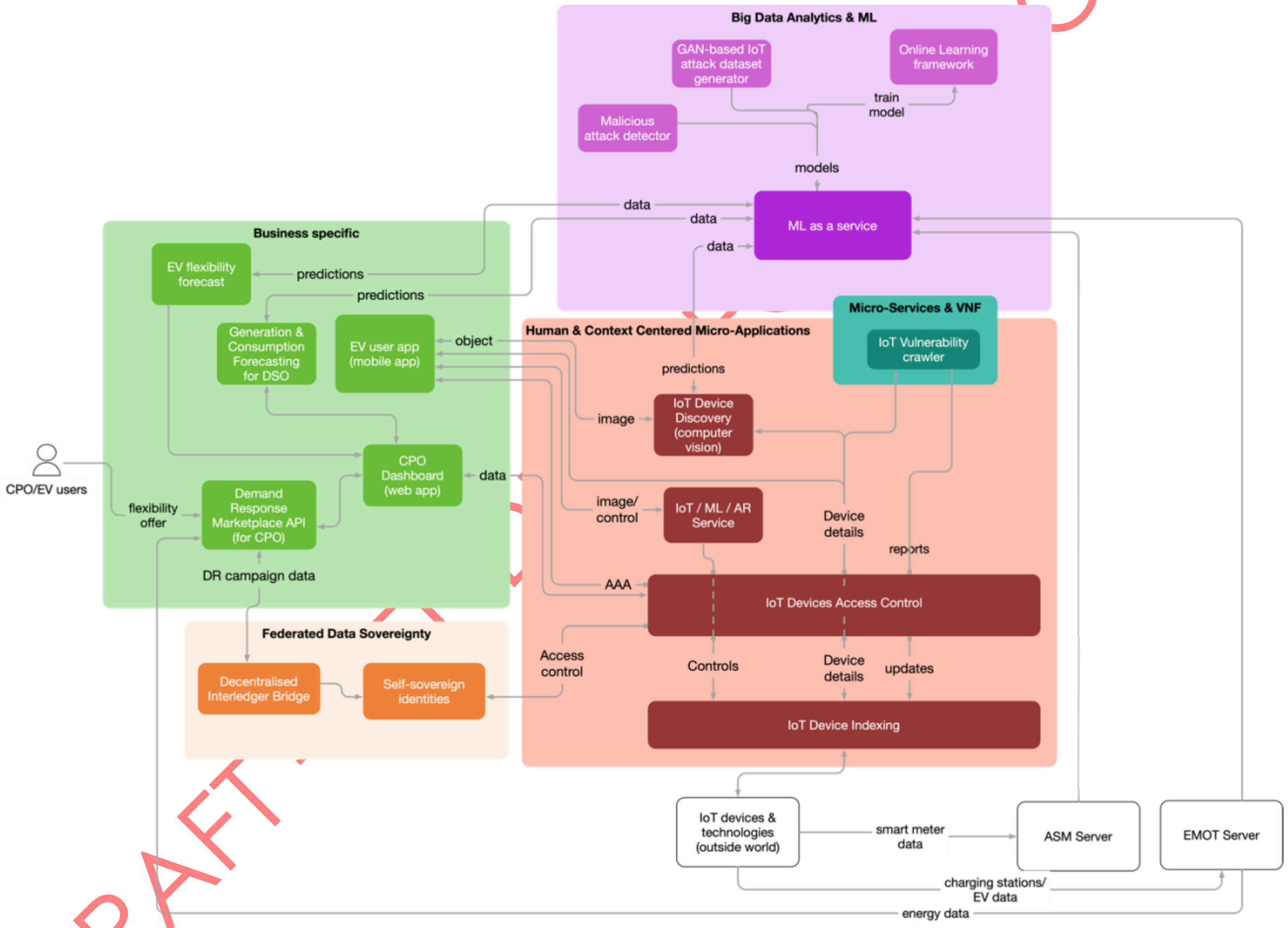

Big Data Analytics & ML
GAN-based IoT attack dataset generator
Online Learning framework
train model
Malicious attack detector
models
data
data
data
ML as a service
Business specific
EV flexibility forecast
predictions
predictions
Generation & Consumption Forecasting for DSO
EV user app (mobile app)
object
Human & Context Centered Micro-Applications
Micro-Services & VNF
IoT Vulnerability crawler
predictions
image
IoT Device Discovery (computer vision)
CPO/EV users
CPO Dashboard (web app)
data
flexibility offer
Demand Response Marketplace API (for CPO)
image/ control
IoT / ML / AR Service
Device details
reports
DR campaign data
AAA
IoT Devices Access Control
Federated Data Sovereignty
Access control
Controls
Device details
updates
Decentralised Interledger Bridge
Self-sovereign identities
IoT Device Indexing
IoT devices & technologies (outside world)
smart meter data
ASM Server
EMOT Server
charging stations/ EV data
energy data

Figure 35 – Architecture instantiation for UC10.

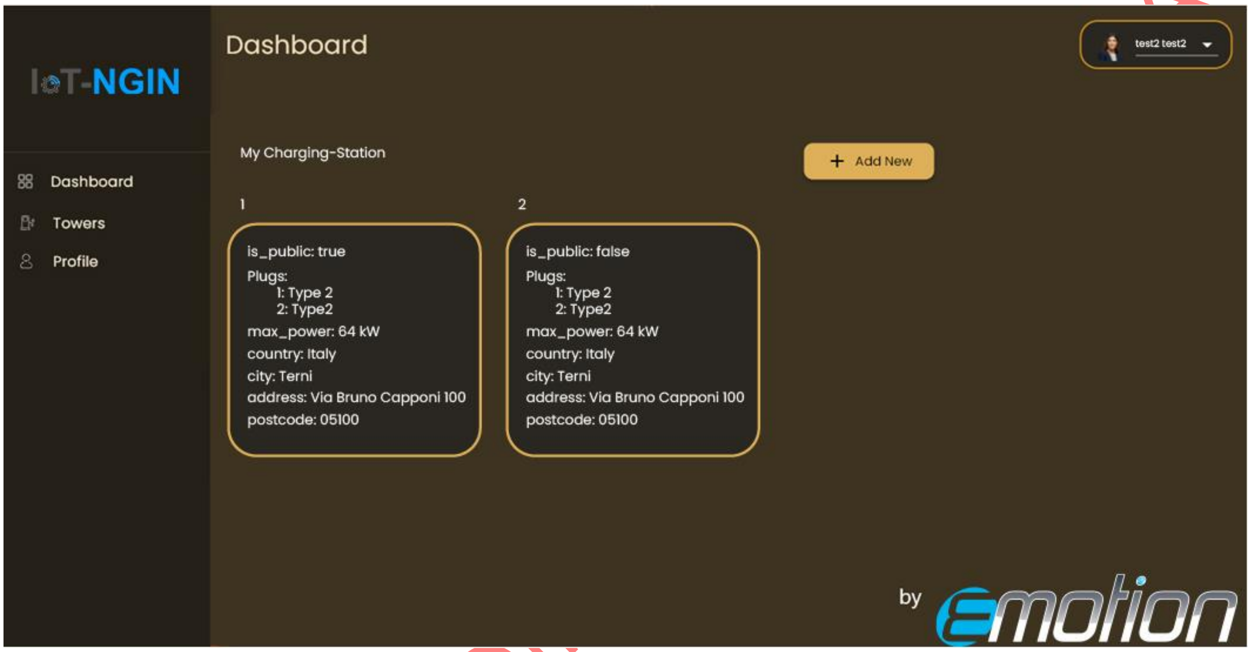
3.4.2.1 Business-specific components

In the following subsections, the business specific components for UC10 are presented.

3.4.2.1.1 CPO Dashboard

The CPO dashboard, presented in Figure 36, is a web platform developed by EMOT in which the historical and real-time data of the monitored charging stations (Figure 37) will be shown as well as the main CPO outcomes, namely:

- Renewable energy usage for EV charging (kWh, %);
- Money saved participating in DR campaign (€);
- CO_2 emissions avoided (kg).



My Charging-Station

1

is_public: true
Plugs:
1: Type 2
2: Type2
max_power: 64 kW
country: Italy
city: Terni
address: Via Bruno Capponi 100
postcode: 05100

2

is_public: false
Plugs:
1: Type 2
2: Type2
max_power: 64 kW
country: Italy
city: Terni
address: Via Bruno Capponi 100
postcode: 05100

Figure 36 – CPO Dashboard.

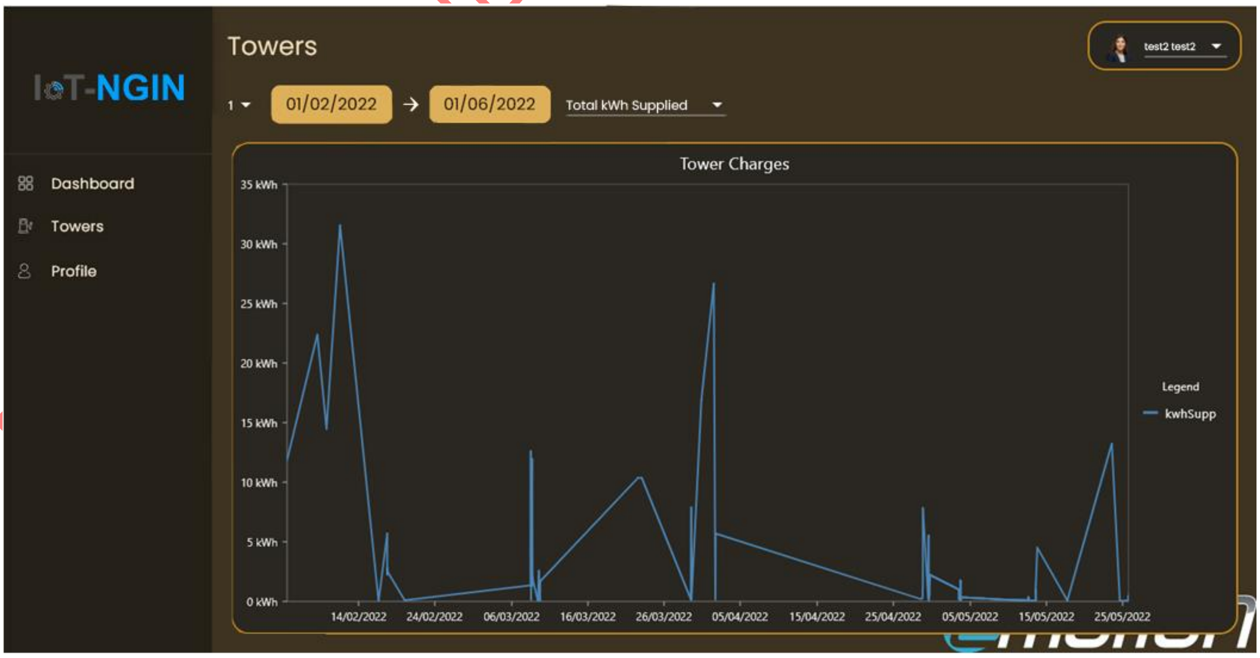


Figure 37 – Charging Station data.

Furthermore, the CPO dashboard is developed to host EV User Application, EV Flexibility Forecasting service and Demand Response marketplace API.

3.4.2.1.2 EV user application

The CPO dashboard is developed to host electric vehicle users in order to aggregate sufficient users for DR campaigns and provide energy flexibility to DSO, so as to balance the grid, increase the use of renewable energy and reduce the cost of EV charging.

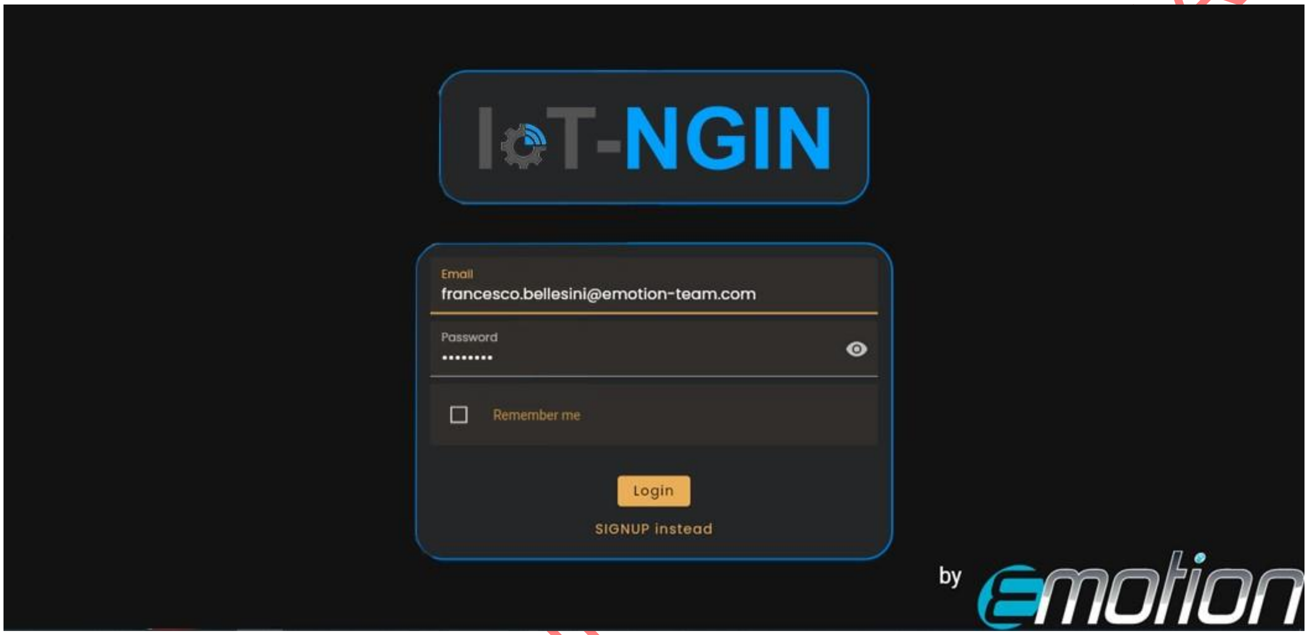
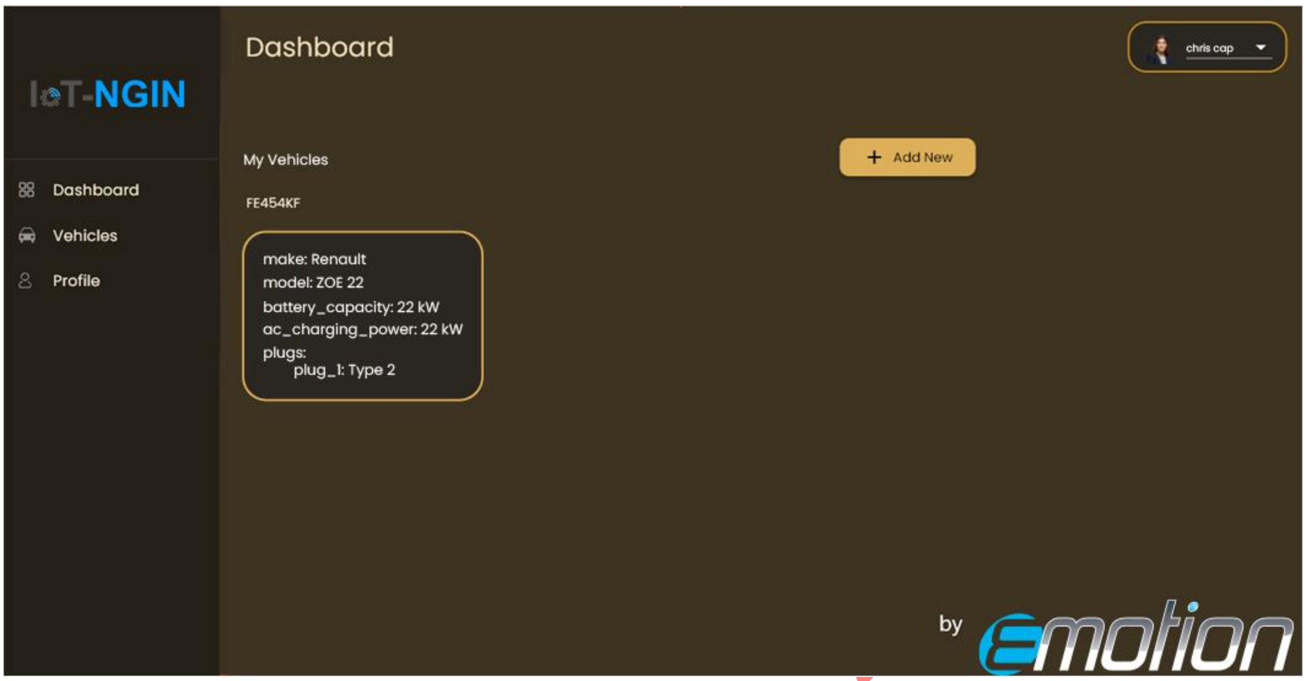


Figure 38 – Dashboard User Login

The CPO dashboard is enabled for electric vehicle users' registration (Figure 38), associating on-board diagnostic device (OBD) to each electric vehicle users (Figure 39); OBD is an IoT component to collect real-time data from the electric vehicle and send data to the server utilizing a TCP/IP communication (Figure 40).



make: Renault
model: ZOE 22
battery_capacity: 22 kW
ac_charging_power: 22 kW
plugs:
plug_1: Type 2

Figure 39 – EV User Dashboard.

Each EV user is enabled to watch on the dashboard historical and real-time data collected by OBD, as shown in Figure 40.

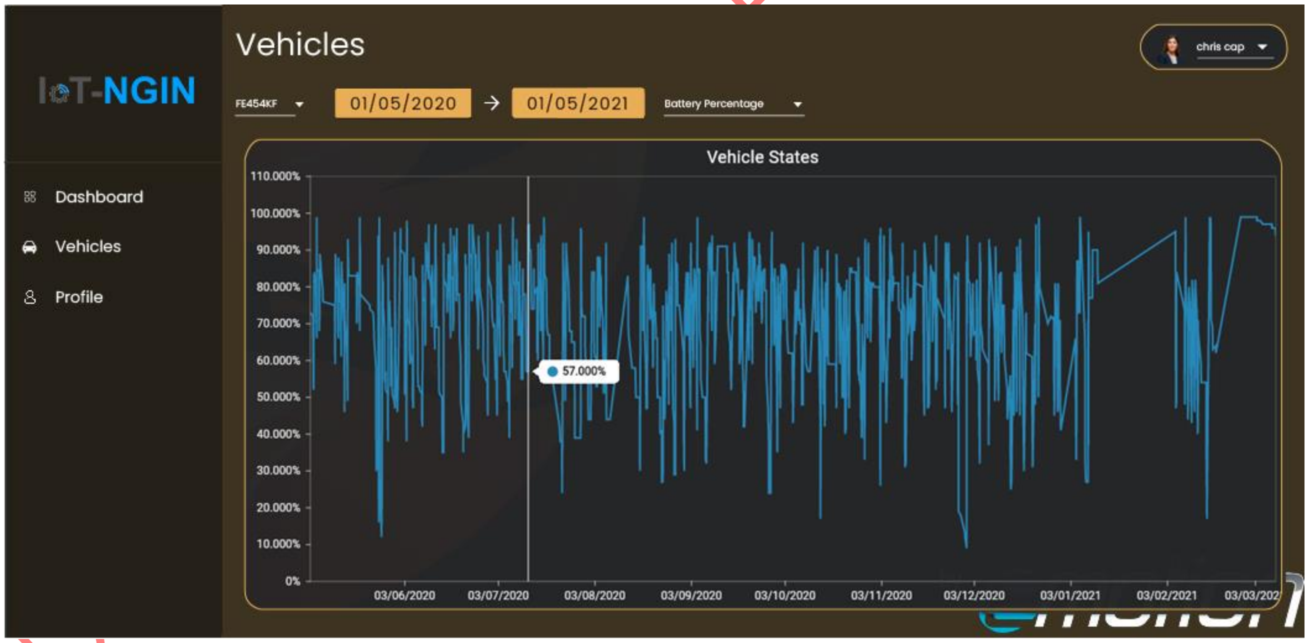


Figure 40 – EV data.

3.4.2.1.3 Generation and consumption forecasting (for DSO)

In the context of the Smart Energy UC10 "Control, and Driver-friendly dispatchable EV charging" forecasting services are required, in particular, the *Power Generation Forecasting*, the *Power Consumption Forecasting and the Energy Demand Forecasting*. These services predict the behaviour of certain parameters of the electric grid and EV chargers.

Power Generation Forecasting and Power Consumption Forecasting services have been designed, implemented and tested in UC10 as a joint collaboration between WP6 T6.2 and WP3 T3.2, by leveraging the MLaaS Online Learning framework. They have been integrated as part of the WP3 MLaaS platform and its technical details preliminary reported in D3.3. Currently they are being integrated into the Smart Energy dashboard. Energy Demand Forecasting service will be designed, implemented and texted in next development sprint and released in M30 version of the MLaaS platform.

Borrowed from D3.3, Figure 41 shows the power generation forecasting UC10.

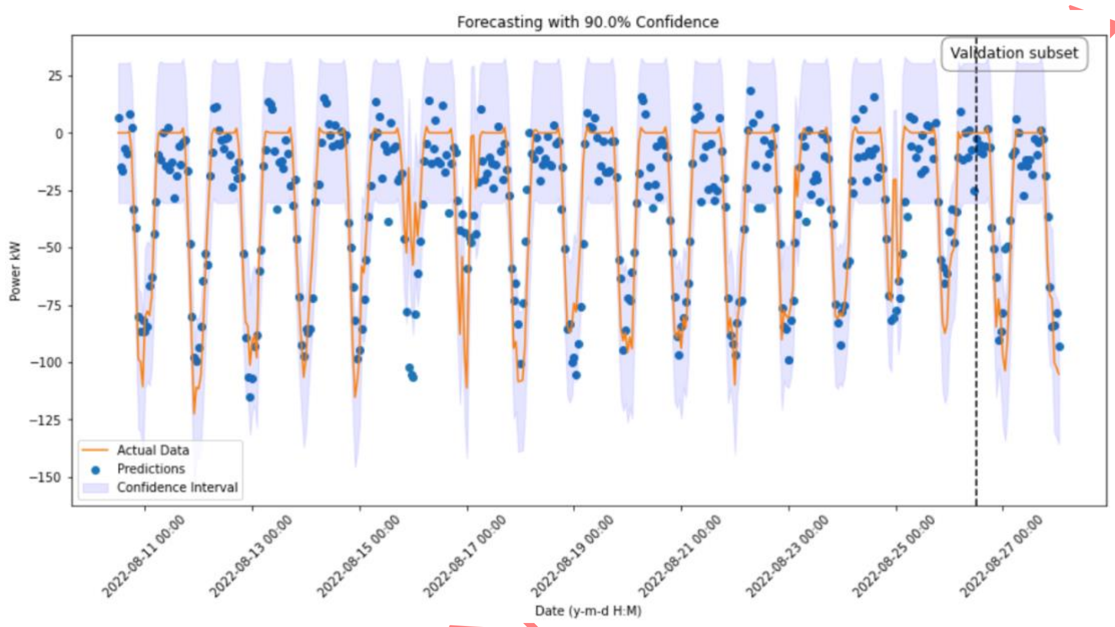


Figure 41 – Forecasting for power generation forecasting of UC10.

3.4.2.1.4 EV flexibility forecast

Smart Energy LL UC10 "Control, and Driver-friendly dispatchable EV charging" requires the Grid Operation Optimization service. This service is being designed and implemented by a joint collaboration between WP3 T2.2 and WP6 T6.2. A preliminary specification of the UC10 scenario for electric chargers and grid optimization was reported in D3.3. This document also describes a range of potential baseline solutions for the implementation of this Grid Operation Optimization service, based on AI reinforcement learning algorithms and libraries.

As for UC9, simulators for this scenario are required to train a RL-based optimization services. Both simulators for the EG and EV chargers are under discussion between WP3 and WP6 teams and under designed and implementation. They will be used to train the RL-based optimization service, which will be released by M30 and integrated within the Smart Energy LL dashboard.

Finally, EV flexibility provision forecasting is developed by leveraging the MLaaS Online Learning framework as for generation and consumption forecasting, focusing on SoC (state of charge) data collected by electric vehicles.

3.4.2.1.5 Demand Response marketplace API (for CPO)

Demand Response (DR) marketplace API enables P2P energy trading between CPO and DSO in an automated, decentralised and flexible way using Ethereum smart contracts; in particular CPO will be able to provide an offer to DR auction created by DSO. Once the bidding time concludes, DSO will close the auction and decide the winner. The winning bid will be the lowest, since DSO is interested in receiving energy flexibility provision service at the lowest price. After the energy flexibility has been supplied, it is confirmed by deployed smart meters and payment is immediately issued. Lastly, a complete audit trail of the auction event is provided and, at any time, anyone can view and audit the DR campaign status.

4 Implementation and testing the performance of 5G as the communications network for IoT-NGIN use cases

In this section, we describe:

- How we have completed the test series of the latency performance of live 5G in a laboratory environment supporting the communication of synthetic data streams representative of project use cases. Data streams and traffic patterns of project use cases were presented in D6.1. Since preparing D6.1, we have been able to analyse and test further use cases in order to complete the planned set of synthetic data tests proposed in D6.1. In this deliverable, we present tests with data streams of seven project use cases which cover all four vertical areas – smart agriculture, smart cities, smart manufacturing and smart energy. The tests with synthetic data use a range of protocols and traffic characteristics, such as protocols and message periods.
- Results of a new series of tests investigating an advanced 5G feature prototype for prioritizing and pre-scheduling uplink data and potentially improving network performance for IoT-NGIN use case data streams.
- Results of tests of the performance of a live 5G in a laboratory environment supporting the communication required by the edgePMU hardware. The edgePMU device generates high loads of data typical of smart energy use cases.
- TSN tests on CMC laboratory based on 5G Core connected to Analog Devices off the shelf TSN switches to validate they get in sync.

4.1 5G latency performance tests of project use cases

In this subsection, we describe the laboratory infrastructure used and the test results obtained in our sets of tests on:

- Live 5G in a laboratory environment supporting the communication of synthetic data streams representative of project use cases, and
- An advanced 5G feature prototype for prioritizing and pre-scheduling uplink data and potentially improving latency performance for IoT-NGIN use case data streams.

As we used the same infrastructure and test methodology for both sets of tests, we provide the description of the test infrastructure used in Section 4.1.1 below.

In Section 4.1.2, we provide the description and the results of the test series of a live 5G in a laboratory environment supporting the communication of synthetic data streams representative of project use cases.

In Section 4.1.3 we provide the description and the results of the test series of an advanced 5G feature prototype for prioritizing and pre-scheduling uplink data and potentially improving network performance for IoT-NGIN use case data streams.

In Section 4.1.4, we provide a conclusion of the subsection.

4.1.1 Description of 5G laboratory infrastructure and the test methodology

Objective

To verify the performance of 5G networks supporting IoT-NGIN use cases, one-way radio link latencies (time delay between the sending and the arrival of a message) were measured and compared for a range of message periods (time interval between the sending of consequent messages).

5G test infrastructure

For the performance tests, a 5G infrastructure was designed and configured in a laboratory environment. The infrastructure comprised a 5G standalone mobile network, hosting an edge cloud, a use case data simulator deployed on a PC, and a 5G radio modem connected to the PC. To record the latencies of transmitted data streams, the tool 'tcpdump' [29] was used at the data simulator and edge cloud side. The timestamps of the sent and received message were extracted from the recordings of tcpdump. Then, the latency was computed by subtracting the times of the received and the sent message. Both uplink and downlink traffic were traced so that the measurement of both one-way latencies was performed separately. By using NTP [30], the clocks on both sides were synchronised to ensure precise one-way latency measurement. More information about the test infrastructure is given in D6.1.

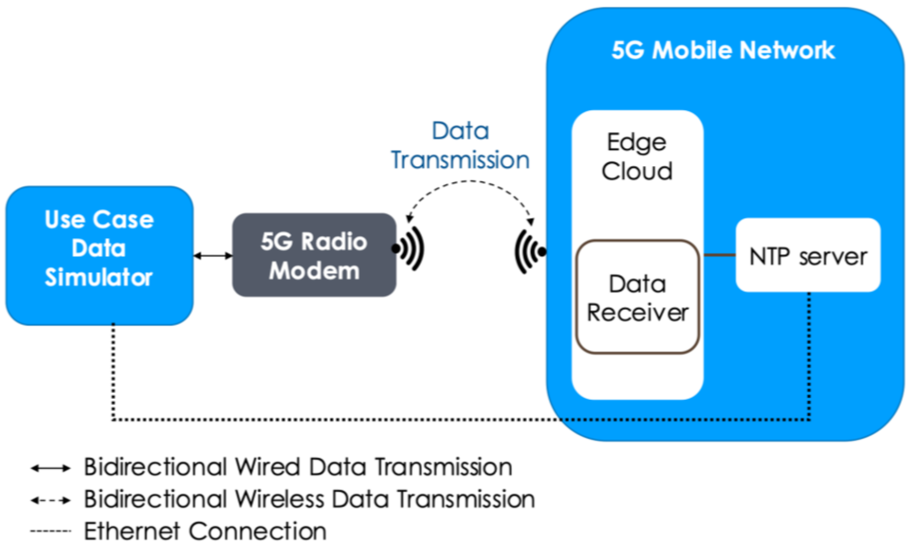


Figure 42 – 5G test infrastructure.

Limitations of the radio network

The components are optimised regarding throughput of eMBB, i.e., not for latency. This should be considered when looking at the results. In the future, there will be features to choose, e.g., a URLLC system could be chosen if latency should be very low and reliability high.

Data streams and communication traffic patterns

This section provides a tabular overview of the data streams and their communication traffic patterns transmitted in the 5G laboratory test infrastructure as well as a brief description of the individual communication protocols used.

In the following table, the terms unidirectional and bidirectional are used. Unidirectional describes the uplink traffic from the user to the base station, i.e., in this case from the PC to edge cloud. Uplink and downlink traffic is described by bidirectional. The traffic patterns are specified by message size and message period. The message size is defined by the sample data gotten from the living labs. It ranges from 10 bytes to 6.5 MB. The message period is the time interval between the sending of consequent messages and goes from 30 ms to 5 s. To investigate the possibilities of the 5G link to handle more frequent packets, a shorter message period than specified in the requirements was used in some test cases. This might become more relevant for future scenarios. The following traffic pattern was used:

Table 4 – Communication traffic patterns of IoT-NGIN use cases.

Use Case		Test Object	Comm. Protocol	Comm. Direction	Message Period [s]	Message Size [bytes]
Vertical	#					
Smart City	1	smart micro radars	WebSocket	Unidirectional	0.1	356
	2	Video transmission	RTSP	Unidirectional	Continuous video stream	6,501,000
Smart Agriculture	4	Synfield [23]	HTTP, MQTT	Bidirectional	0.03, 0.1, 0.3, 1	325
	5	Jpg file			0.1, 0.2, 0.5, 1	36,645
Smart Industry	8	Smart sensor	OPC UA	Bidirectional	0.1, 0.5, 1, 5	140
Smart Energy	9	PQA	HTTP	Unidirectional	0.1, 0.3, 1	1,073
		PMU	MQTT		0.1, 1	665
		smart meters	MQTT		0.1, 3, 5	1,000
	10	charging stations	MQTT	Bidirectional	0.1, 2	10
		electric vehicles	MQTT		0.1, 1, 3, 5	50
		smart meters	MQTT		0.1, 1, 3, 5	100

Table 4 does not include test results of use cases 3, 6 and 7, as resources for testing with live 5G in the first project reporting period were focused on testing those use cases placing stringent requirements on mobile network performance and not planned to be tested by other partners in the second reporting period of the IoT-NGIN project.

- UC3 (Co-commuting solutions based on social networks): The description of this use case available during the period when 5G tests were being planned by the IoT-NGIN project (end of 2021) focused on the transmission of data to the internet from standard mobile phones and mobile devices. The data is social network data streams and in this use case, the focus is on the use of twitter messages to identify opportunities for co-commuting. Such Twitter data streams are characterised by small message sizes and low requirements on data transmission rates. The use case plans the use of any available wireless network in the vicinity of the user as the carrier network including WiFi, 3G, 4G and 5G networks where available. The communications sector has many years of experience in the use of such networks to support such data streams and experience has shown that all of these networks are more than capable of adequately supporting such data streams. The added value of running tests of such data streams on a live 5G network using the resources of the IoT-NGIN project would be minimal and the project decided to focus its testing efforts on those use cases which place stringent requirements on the mobile network performance, and which were not planned to be tested by other partners such as I2CAT or CMC.
- UC6 (Human-centred safety in a self-aware indoor factory environment): As this use case has been planned to be lab-trialled in I2CAT's wireless network laboratory infrastructure, the added value of running tests using synthetic data would be minimal and the project decided to focus its testing efforts on those use cases which place stringent requirements on the mobile network performance, and which were not planned to be tested by other partners such as I2CAT. These tests will be reported in D6.3.
- UC7 (Human-centred augmented reality assisted build-to-order assembly): As this use case has been planned to be lab-trialled in CMC's wireless network laboratory infrastructure, the added value of running tests using synthetic data would be minimal and the project decided to focus its testing efforts on those use cases which place stringent requirements on the mobile network performance, and which were not planned to be tested by other partners such as CMC or I2CAT. These tests will be reported in D6.3.

MQTT (Message Queuing Telemetry Transport) works in a publish-subscribe architecture so that a client transmits data by publishing and receives data by subscribing. A client can be publisher, subscriber, or both at the same time. No point-to-point connections are established. Therefore, all clients are connected to a MQTT broker. The broker acts as a central unit. Every message is sent to it. The published messages are handled by distributing them to the corresponding subscribers. The tool for MQTT broker is called mosquitto. In our setup, the PC is publisher and subscriber at the same time. Packets are generated by using synthetic data (text files) as input. Then, the packets are published and transmitted as MQTT messages over the 5G link. The broker is deployed in the edge cloud of the 5G network and receives the messages. If a bidirectional link is specified, the packets or control messages are sent back to the subscriber on the PC.

HTTP (Hypertext Transfer Protocol) is a request-response protocol based on client-server model. It is an application layer protocol assuming an underlying reliable transport protocol like TCP. HTTP is mainly known from data communication of the world wide web. Before a message exchange can take place, a three-way TCP handshake is executed between client and server. The client sends a SYN message. The server answers with a SYNACK so that the client successfully finishes the handshake by an ACK. Then, the client sends a POST request via the 5G link which includes the synthetic data in message body. The server receives the packets and answers by sending a control message. This procedure is repeated for every new packet to be sent [31] Here, the client is the PC, and the server is the edge cloud. The server is established by opening a dedicated port to HTTP over which PC and edge cloud can communicate. Important to mention is that the communication is used without proxies.

OPC UA (Open Platform Communications Unified Architecture) is a machine-to-machine communication protocol based on server-client model. It results in a service-oriented architecture (SOA) and subscribe/publish communication. There are servers, clients, and nodes. A node is a basic abstract data in the network and can be a variable, an object, a method and much more. Nodes are handled by the server and the main way to communicate with clients. Clients can be subscribed to a node event so that they are informed when the node's data changes, i.e., the node publishes new data. OPC UA is commonly known from IoT applications and industry 4.0 [32] Here, the OPC UA is based on TCP. A node is created by the edge cloud which is responsible for the node's data (for simplicity the payload is only increasing) as the node cannot change its own payload, e.g., by measuring the temperature. The client is the PC which is subscribed to the node. The client begins to request the data and the node answers by sending a response including the data. This is done every time when the data is changing, i.e., when the payload is increased.

WebSocket (WS) is a communication protocol on layer 7 depending on TCP. Prior to sending messages, a three-way TCP handshake is run like in HTTP. Then, the client sends a WS handshake requests by using the HTTP upgrade header to switch from HTTP to WS. The server answers with the WS handshake response indicating success or failure of the upgrade. If the upgrade was successful, the data is sent in a POST request via the 5G link from the client to the server. The server uses a control message as an answer. The client-server connection is persistent and is finished automatically after the last control message is received. [33] Here, the client is the PC, and the server is the edge cloud. The server is established by opening a dedicated port to WS over which PC and edge cloud can communicate.

RTSP is short for Real Time Streaming Protocol. It is an application-level network protocol based on a TCP connection designed for multiplexing and packetizing multimedia transport streams (such as interactive media, video, and audio) over a suitable transport protocol. The transmission of streaming data itself is not a task of RTSP. Most RTSP servers use the Real-time Transport Protocol (RTP) in conjunction with Real-time Control Protocol (RTCP) for media stream delivery. Before messages are transmitted, the request types, the RTSP URL (rtsp://…) and the setup are defined. [34] Here, the edge cloud hosts the RTSP server which opens a dedicated port for the video transmission. The PC sends the video to the edge cloud. Only uplink is considered. When the transmission is finished, the TCP connection is closed.

Test methodology and statistical analysis approach

The following test methodology was performed stepwise for each test:

- The tcpdump was started on the PC,
- The tcpdump was started on the 5G edge cloud,
- A data stream was configured with the protocol, message size and message rate requested by the use case,
- The data simulator was started,
- The data simulator was stopped after a specified number of messages sent,
- The tcpdump was stopped on the simulator and the cloud,

- The trace files for each test sequence were collected, and finally the
- Latencies for each test were computed.

To ensure representative statistical results, the following measures were undertaken: a high number of samples were collected to minimise untypical behaviour. Therefore, each test was repeated several times collecting around 5,000 samples. An appropriate approach to interpret the results was chosen. Our approach was to calculate the average and the nth percentile and the reduction of latency:

- The **average** is also known as the arithmetic mean indicating the sum of all samples collected and divided by the number of samples.
- The nth **percentile** is used as a second statistical measure which means that n% of all samples are smaller than this value. It is a measure of reliability because it shows an overall picture of the sample distribution.
- **Reduction** describes the difference between lowest and largest value given in percentage which could be achieved when using optimal configurations compared to the weakest configuration, e.g., by using an optimal message period or communication protocol.

4.1.2 Results of 5G latency tests of IoT-NGIN use case synthetic data streams

Seven use cases were investigated representing four Living Labs (smart city, smart industry, smart agriculture, smart energy) by investigating the communication traffic patterns of the use cases, designing the data streams, and performing laboratory tests using those data streams. In the following, the results for each use cases are given using the statistical means introduced previously and summarised in a table.

Test results

This section provides the results of the 5G latency tests per use case. The latency results are shown as average and 99th percentile. In tests where traffic patterns such as message period or communication protocol were compared, we show the reduction of latency.

Traffic Flow and Parking Prediction (UC1)

In use case 1, sample data of 356 bytes was transmitted. Only uplink for a message period of 0.1 s was investigated. The results show that the average latency is 6.59 ms. The average 99th percentile is around 5 ms larger.

Table 5 – Results use case 1.

	Uplink
Average latency	6.59 ms
99th percentile	11.5 ms

Crowd Management (UC2)

In use case 2, a video of 6.5 MB was transmitted in uplink. The average latency is 8.88 ms and the average 99th percentile increases by 8 ms.

Table 6 – Results use case 2.

	Uplink
Average latency	8.88 ms
99th percentile	16.7 ms

Crop Disease Prediction: Smart Irrigation and Precise Aerial Spraying (UC4)

In this test, two communication protocols were tested: MQTT and HTTP. For both communication protocols, sample data of size 325 bytes were sent. The results show that HTTP has a worse performance because it uses a 3-way handshake each time before sending a message. The average latency is around 10ms and the 99th percentile is up to 18 ms. MQTT uses this handshake only at the beginning of the transmission so that its average latency has a minimum value of 8 ms and the 99th percentile shows the smallest value of 10 ms. Hence, MQTT has a latency reduction of at maximum 24 % and a 99th percentile reduction of up to 33 % in contrast to HTTP so that it performs better regarding latency values and reliability. MQTT can be recommended to improve latency because HTTP was used in its normal configuration with new handshake and request every time before a new packet was transmitted. The smallest values are achieved by using a message period of 0.03 s and 0.3 s. The results with those message periods when MQTT is used are shown in Table 7.

Table 7 – Results Use Case 4 using MQTT protocol.

	Uplink		Downlink	
	0.03 s	0.3 s	0.03 s	0.3 s
Average latency	8.91 ms	7.86 ms	7.84 ms	9.86 ms
Reduction of average latency compared to HTTP	14.01 %	23.07 %	24.48 %	7.46 %
99th percentile	13.53 ms	9.94 ms	10.76 ms	11.31 ms
Reduction of the 99th percentile compared to HTTP	25.14 %	27.51 %	27.22 %	33.22 %

Sensor Aided Harvesting (UC5)

In this test, two communication protocols were tested: MQTT and HTTP. A jpg file of 36,645 bytes was transmitted in this use case. The synthetic data tests result in larger values in HTTP than in MQTT. Especially in uplink, MQTT leads up to 42% better results. The average latency performance is around 10 ms in HTTP and up to 4 ms faster in MQTT. Even though MQTT has a larger deviation of values around the latency median, they are still smaller than the actual HTTP values. In the 99th percentile, HTTP reaches maximum values around 16ms. MQTT values are smaller or equal to HTTP. To ensure a high reliability as well as small latencies, MQTT with a message period of 0.1 s is recommended. The results with the message period of 0.1s when MQTT is used are shown in Table 8.

Table 8 – Results Use Case 5 using MQTT protocol and 0.1s message period.

	Uplink	Downlink
Average latency	5.8 ms	10 ms
Reduction of average latency compared to HTTP	42 %	10 %
99th percentile	11.5 ms	11.3 ms
Reduction of the 99th percentile compared to HTTP	14 %	28 %

Digital Powertrain and Condition Monitoring (UC8)

In this use case, the communication protocol was OPC UA. The uplink latency is around 9 ms and the downlink latency around 5 ms. The 99th percentile is in uplink around 15 ms and in downlink around 8 ms to 10 ms. A preferable message period is 0.1 s or 0.5 s which makes the results 5% in average faster in and up to 15 % the 99th percentile in downlink. An appropriate uplink message period can improve the results by 13 % in average and by 2 % in the 99th percentile. The small improvements can be explained by the independence of the latency values from the message periods. The best results with the message periods of 0.1s and 0.5s are shown in Table 9.

Table 9 – Results Use Case 8 with message periods of 0.1s and 0.5s.

	Uplink	Downlink
Average latency	8.4 ms	5.11 ms
Reduction of average latency with optimal message period	13 %	5 %
99th percentile	5.1 ms	8.4 ms
Reduction of the 99th percentile with optimal message period	2 %	15 %

Move from Reacting to Acting in Smart Grid Monitoring and Control (UC9)

Use case 9 uses multiple protocols. PQA (Power Quality Analyser) uses HTTP and has message size of 1073 bytes. Smart meters with a message size of 1000 bytes and PMU (phasor measurement unit) of 665 bytes use MQTT.

For PQA, the latency is around 10 ms while the percentile is around 16 ms. Optimal performances can show 2 % faster results in average latency while the 99th percentile results in nearly no improvement. PQA has an average latency of 9 ms and its 99th percentile reaches 13 ms. Reducing the message period to 0.1s leads to an improvement of 4 % to 6 %. Since the improvements are very small, these two message types are nearly independent from the message period. For smart meters, the latency ranges from 8 ms to 10 ms and the percentile from 15 ms to 20 ms. In this case, an appropriate message period can improve the results in average by 19.4 % and in 99th percentile by 28.8 %. Since it has the highest percentile, smart meters have the worst reliability. The best reliability has PMU having a deviation of 0.5ms around the mean. For the smallest latency and the highest reliability, the message period of 0.1s is recommended for smart meters while the periods of the other use cases behave equally.

Table 10 – Results Use Case 9 with optimal message periods.

	PQA	Smart meters	PMU
Average latency	10.1 ms	8.37 ms	8.1 ms
Reduction of average latency with optimal message period	2 %	19.4 %	6.3 %
99th percentile	16.7 ms	14.5 ms	12.5 ms
Reduction of the 99th percentile with optimal message period	1.26 %	28.8 %	4.4 %

Driver-friendly Dispatchable EV Charging (UC10)

The three message types of use case 10 use MQTT as communication protocol. The message size is 10 bytes in charging station, 50 bytes in electric vehicles and 100 bytes in smart meters.

The average uplink latency is around 6 ms, and the average downlink (DL) latency is around 8 ms. By choosing the optimal message period the uplink results at maximum in 10 % improvement and in downlink in 3 %. Like in the previous use case, the results show the same characteristic that they seem to be independent from the message size, i.e., that the difference between the values is in uplink at maximum 0.5 ms and in DL 0.03 ms.
The 99th percentile is in uplink between 7 ms and 9 ms. In downlink, it is more distributed from 10 ms to 14 ms. This shows that uplink is more reliable than downlink. This is underlined by the general fact that the uplink is always smaller than downlink for both latency and 99th percentile. Moreover, there is a higher maximum reduction in DL except for the message type of charging stations. Since the uplink performances are very similar, the message periods can be chosen based on their best 99th percentile. Consequently, 1 s is recommended for electric vehicles and smart meters achieving an improvement of up to 35 %. For charging stations, all message periods can be taken since the reduction is at maximum 10 %.

Table 11 – Results Use Case 10 with optimal message periods.

	Uplink			Downlink		
	Charging stations	Smart meters	Electric vehicles	Charging stations	Smart meters	Electric vehicles
Average latency	5.69 ms	5.7 ms	5.69 ms	8.01 ms	7.6 ms	7.84 ms
Reduction of average latency with optimal message period	8.27 %	9.37 %	8.53 %	0.12 %	3.33 %	2.51 %
99th percentile	7.63 ms	7.85 ms	7.77 ms	13.93 ms	9.68 ms	9.15 ms
Reduction of the 99th percentile with	9.93 %	9.3 %	10.4 %	0.014 %	29.7 %	34.6 %

optimal message period						

4.1.3 Results of 5G latency tests of a prototype service-adaptive prescheduling feature

In addition to the standard configuration of the 5G network, we had the opportunity to test a new feature for service-adaptive prescheduling. Based on this feature the uplink data can be prescheduled and prioritised enabling faster time-to-content especially for small data transmission. Moreover, it supports a reduction in the air interface contribution for end-user latency resulting in decreased latency for specified radio modems configured with this mode.

In the test of this series, we generated background traffic using iperf on a second 5G radio modem connected to the same 5G network. The background traffic was running in parallel to the use case data streams without prioritised configuration to show that the synthetic data was prioritised to the background traffic. In the first round of tests, the radio modem used for MQTT data transmission was not prioritised while in the second round of tests, that radio modem was configured to be prioritised for service-adaptive prescheduling. The following use cases of test series 1 with MQTT as communication protocol were used:

- UC4 a): SynField with message period of 0.1s
- UC4 b): SynField with message period of 0.03s
- UC9 a): PMU with message period of 0.1s
- UC9 b): Smart meters with message period of 0.1s
- UC10 a): Charging station with message period of 0.1s
- UC10 b): Electric vehicle with message period of 0.1s

The results of the average uplink latency are illustrated in Figure 43. It shows that latencies are lower when the modem was prioritised. The improvements of using a prioritised modem were up to 7.5%. A greater effect of the feature may be achieved with more modems generating a higher load of background traffic.

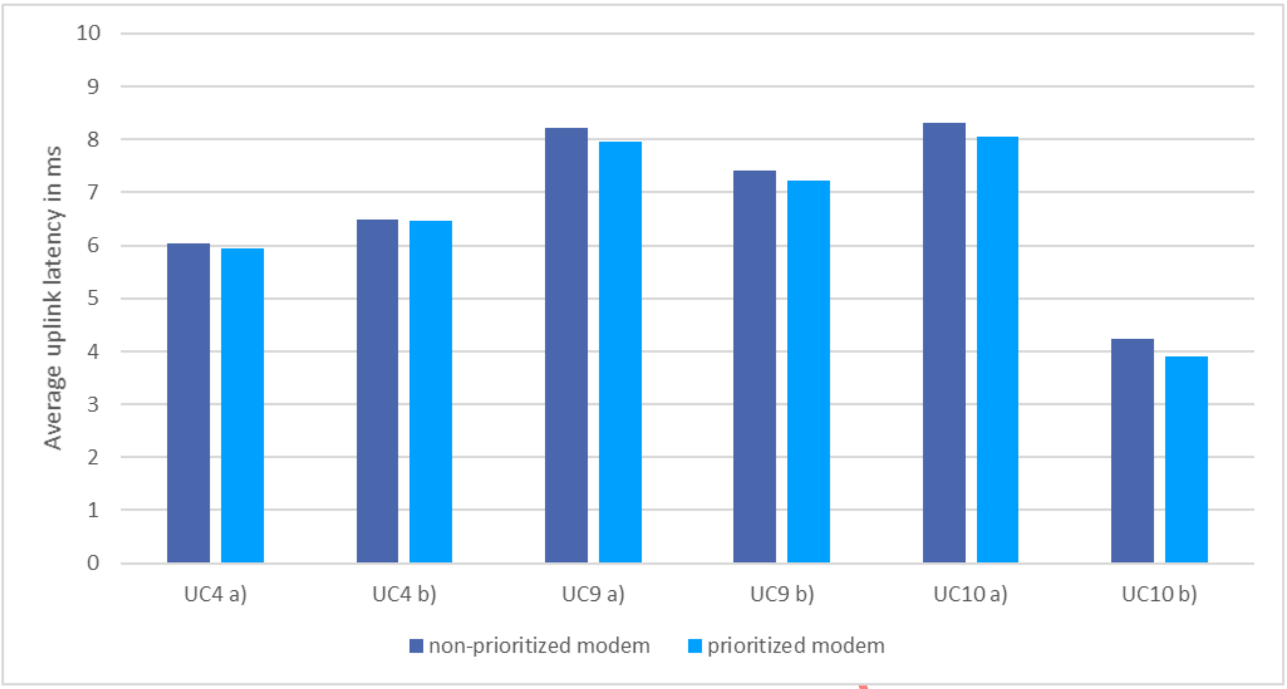


Figure 43 – Test results with service-adaptive prescheduling feature.

4.1.4 Conclusion of the 5G latency tests of project use cases and of the prototype prescheduling feature

Our tests demonstrated that 5G can support the latency requirements of the IoT-NGIN use cases. We gave recommendations how to optimise the latency by using optimal message periods and communication protocols.

Tests with the new service-adaptive prescheduling feature showed that 5G provide low latencies even under high load and the feature can be used to prioritise specific devices.

4.2 5G latency performance tests of edgePMU data streams

In this subsection, we describe the laboratory infrastructure used and the test results obtained in our sets of tests on edgePMU data streams in a live 5G laboratory environment.

In Section 4.2.1, we introduce the edgePMU concept.

In Section 4.2.2, we provide the description of the 5G laboratory infrastructure and the test methodology used for the tests.

In Section 4.2.3, we provide the description and the results of the test series of a live 5G in a laboratory environment supporting the communication of edgePMU data streams.

In Section 4.2.4, we provide a conclusion of the subsection.

4.2.1 Introduction to the edgePMU concept

The edgePMU is a new approach on how to calculate phasors by utilizing the edge cloud, that is available in the 5G network. In contrast to the classical concept, where the measurement of the samples and the estimation of the phasors is performed in a single device, the edgePMU splits this in two. As shown in Figure 44 the device, deployed in the field, acquires the samples and time tags them. The samples are then transmitted to an edge cloud service, where the phasor estimation is executed. This approach enables new possibilities for utilization of algorithms, as well as for more compute intensive estimation approaches.

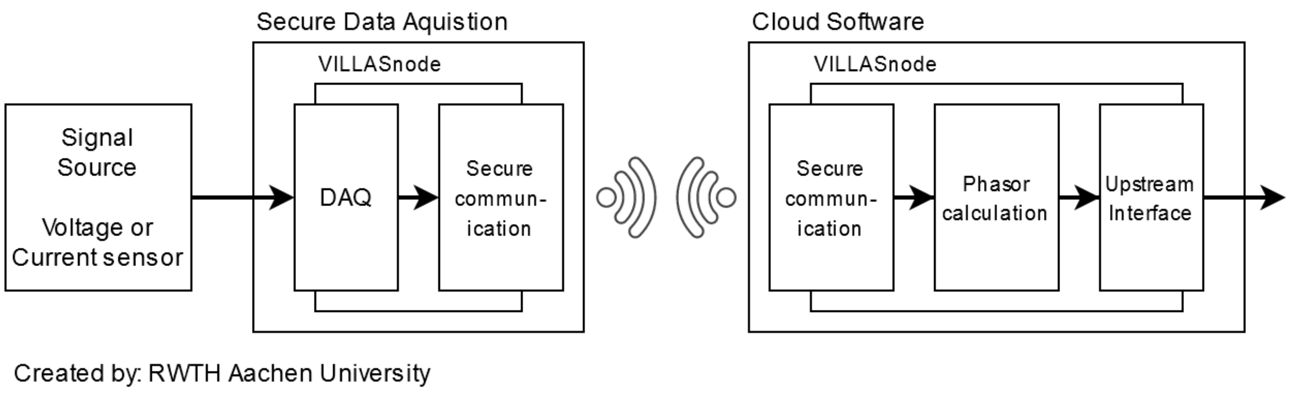


Figure 44 – edgePMU concept.

4.2.2 Description of 5G laboratory infrastructure and the test methodology for 5G edgePMU tests

Objective

To verify the 5G performance with data streams generated by the edgePMU, an edgePMU device was integrated in the setup used in test series 1. The data characteristics of the edgePMU show higher data rates and throughputs than the data characteristics of the project use cases investigated in test series 1.

Infrastructure

The 5G SA network used in test series 1 was modified by adding the following components:

- edgePMU hardware including Ethernet cables and PoE (Power-over-Ethernet)
- VM (virtual machine) located in the 5G edge cloud
- Switch to create a mirror link for capturing data traffic of the edgePMU

Since synchronization between the 5G network and the edgePMU was unavailable, a switch was integrated between the edgePMU and the WNC 5G modem to mirror the data of the edgePMU to the PC. The PC was time synchronised via NTP with the 5G network and time stamped packets were captured on the PC. In this way, precise one-way latency measurements were performed.

The uplink transmission between the edgePMU and the 5G network was traced. The edgePMU uses up to eight channels which can be turned on and off depending on the use case. For instance, voltage calculation needs three channels for each phase and one channel for the synchronization. The number of channels is changed in the configuration file on the Raspberry Pi.

In a real deployment, the edgePMU device would be connected with an analogue cable from the outside, e.g., from the power grid, to the DAQU. Since we performed the tests in a 5G laboratory, the analogue connection stayed open. The device hence generated dummy messages which had the same characteristics as if the device would be connected to a power grid.

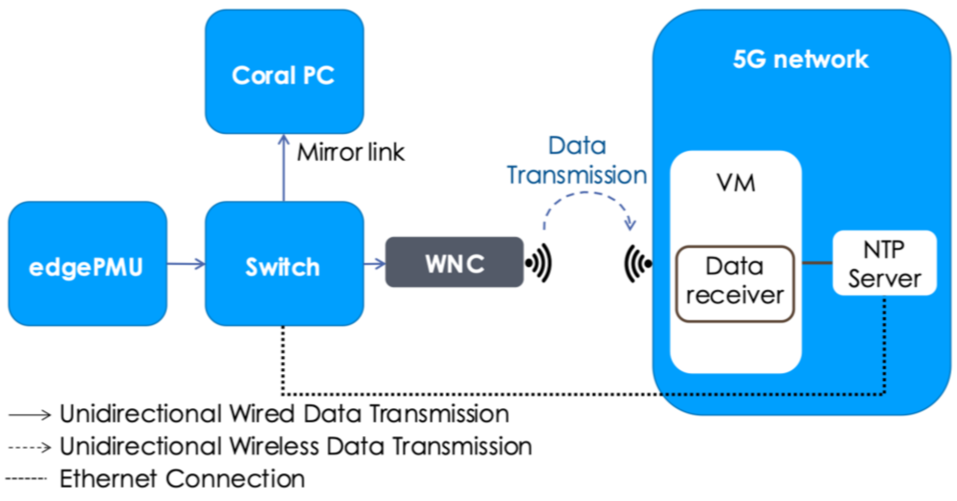


Figure 45 – Schematic diagram of the test infrastructure with the edgePMU.

Data streams

The signals generated by the edgePMU were transmitted as UDP packets to the VM via the 5G link. In contrast to TCP, UDP is a connectionless protocol and does not use ACKs, congestion control, packet ordering and retransmission.

Limitations

Since the setup has only minor changes, the limitations of the edgePMU tests are identical to the 5G performance tests.

Statistical aspects

Since the setup has only minor changes, the statistical aspects of the edgePMU tests are identical to the 5G performance tests of test series 1.

Traffic patterns

Three parameters were configured in the edgePMU: number of channels, sampling rate and vectorize. The number of channels depends on the use cases and ranges from one to eight. Sampling rate defines how many samples are generated per second. It can range from 1000 up to 60,000 Hz. Vectorize defines how many samples one UDP packet contains. 10,000 packets were sent for each test.

These configurable parameters have a direct impact on the communication characteristics. Table 12 shows the impact of the parameters on packet size, message period and throughput. Detailed information on the configurable parameters and which ranges have been used for the tests is provided in Annex 1.

Table 12 – Impact of the parameters on communication characteristics.

	Packet size	Message period	Throughput
Number of channels	X		X
Sampling rate		X	X
Vectorize	X	X	

Test methodology

The following test methodology was performed stepwise for each test:

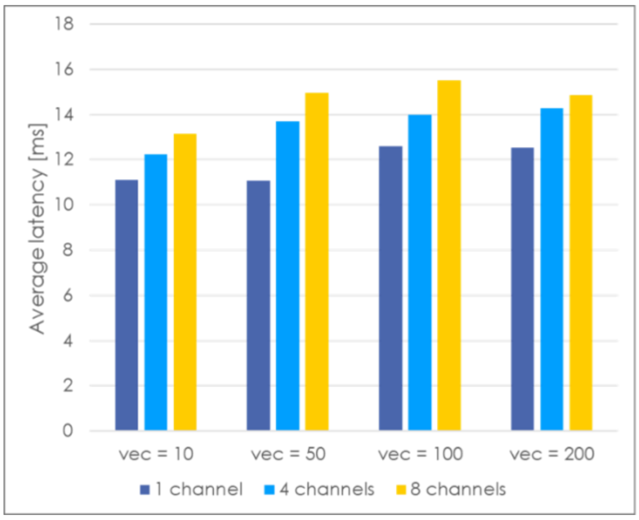
- The tcpdump was started on the PC,
- The tcpdump was started on the VM,
- The edgePMU was configured with the sampling rate and vectorize,
- The edgePMU was started,
- The edgePMU was stopped after a specified number of messages sent,
- The tcpdump was stopped on the PC and the VM,
- The trace files for each test sequence were collected, and finally the
- Latencies for each test sequence were computed.

4.2.3 Results of 5G latency tests of edgePMU data streams

In this set of tests, the sampling rate was set to 20kHz or 50kHz. Vectorize as well as the number of channels was varied. The number of channels was increased from 1 to 4 and 8 while the vectorize was stepwise increased from 10 to 50, 100 and 200. Figure 46 shows that a higher number of channels resulted in a higher latency which can be explained with the higher throughput.

With 20kHz sampling rate, the average uplink latency also slightly increases when the vectorize parameter was increased. This behaviour was not observed for a sampling rate of 50kHz where the average latencies of 4 channels, for example, stayed constant around 14 ms in all vectorize settings. A potential explanation could be that a sampling rate of 50kHz matched better with the scheduled transmission slots in the 5G radio.

Overall average uplink latencies lay in the range of 11 to 15.5ms for 20kHz sampling rate and in the range of 11 to 16ms for 50kHz sampling rate.



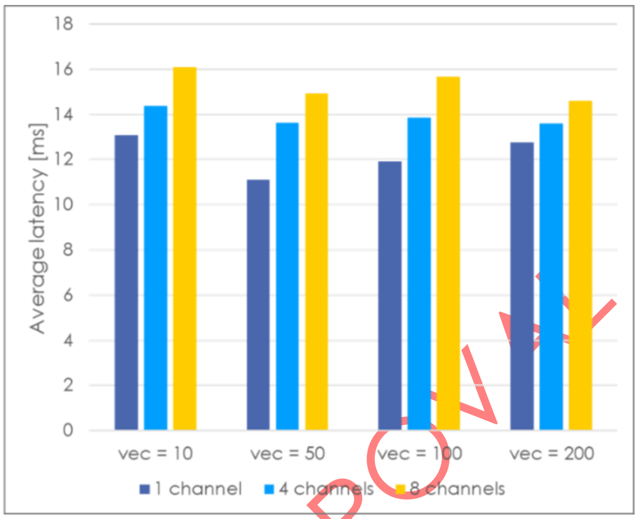


Figure 46 – Average uplink latency for a range of settings.
Left: Sampling rate 20kHz, right: Sampling rate 50kHz.

4.2.4 Conclusion of the 5G latency tests edgePMU data streams

Our tests demonstrated that the edgePMU device can be integrated in a live 5G network. The capabilities of 5G support the high throughput and low latency which the edgePMU requires.

4.3 Implementation and testing of TSN networks

The usage of 5G networks in industrial environments requires a set of features different from consumer mobile networks. 5GLAN or the integration of 5G devices as part of the fixed Local Area Networks (LAN) is a key enabler for connecting mobile with fixed industrial devices. Time Sensitive Networks (TSN) deliver additional functionality to provide deterministic communications between fixed and mobile devices.

Time-Sensitive Networking (TSN IEEE 802.1Q), is an Ethernet technology that provides deterministic messaging on standard Ethernet. When centrally managed, the TSN technology offers the capability of guaranteed delivery of messages with reduced jitter. TSN uses time-scheduling in providing deterministic communications and works at Layer 2 (L2) of the Open System Interconnection. The advantage of TSN working at L2 is that TSN entities (switches and bridges) only need the information contained in Ethernet headers to make forwarding decisions. In addition, the information carried in Ethernet frame payloads does not have to be limited to IP only, making TSN applicable in industrial applications with different application payloads.

5G would behave as standard TSN switch so all the complexity of mobile networks should be hidden and instead the 5G looks like a Logical (TSN) switch as shown in Figure 47.

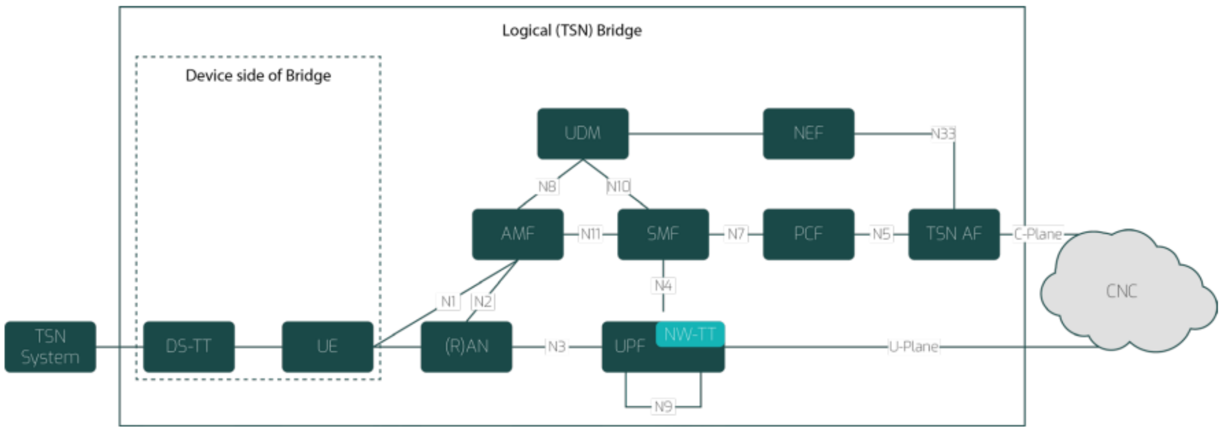


Figure 47 – 5G represented as TSN logical bridge.

This section describes the initial design and implementation of time synchronization modules such as Network TSN Translator (NW-TT) and the Device Side TSN Translator (DS-TT) defined in the 3GPP standards to synchronise the clocks of the mobile devices with the fixed devices.

The preliminary implementation of NW-TT and DS-TT has been used to test the accuracy of the time synchronization and results are shown in this section. The experimentation based on the IEEE Working Group (WG) recommendations and publications providing the necessary modifications to Precision Time Protocol version 2 (PTPv2). The IEEE standards specify the packets that needed to be modified to develop generalised Precision Time Protocols (gPTP). Before entering and exiting the 5G System (5GS), the gPTP messages are changed. These encompass all the needed packet header modifications and necessary calculations to achieve the synchronization accuracies of 900 nanoseconds as stipulated by the IEEE 802.1AS standard.

4.3.1 Description of the laboratory configuration and the test methodology

The implementation of the NW-TT is deployed in standalone server running Linux Ubuntu 20.04 LTS that is connected to the 5G Core network through the User Plane Function (UPF) network function. The UPF has direct link with the base station (gNB) to send the time synchronization that NW-TT receives from the Grand Master clock located in the fixed network.

The implementation of the DS-TT is deployed in a Linux server that incorporates 5G modems for connecting to the 5G network. The language of choice for this was "C" since it provided the different Linux libraries vital when writing code that interacts with the kernel. Linux offers multiple ready-to-use libraries and modules that make writing sockets that handle traffic between different nodes straightforward. Thus, the DS-TT will receive the time synchronization from the modem i.e., User Equipment (UE) that is integrated in the same server. The DS-TT will perform the required adaptation of the time synchronization and will forward to the TSN switch that will get synchronised with TSN switch located in the fixed network and connected to the NW-TT as shown in Figure 48.

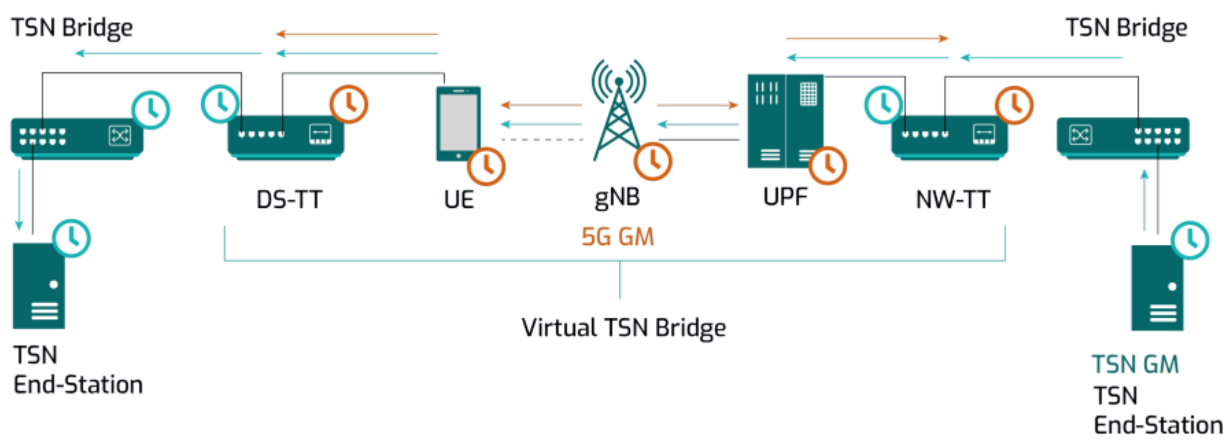


Figure 48 – Time synchronization test setup.

4.3.2 Results of tests of TSN synchronization

The TSN synchronization setup discussed in the previous section forms the basis upon which the final testing results are retrieved from. The final synchronization results after the messages are finally modified and sent by the DS-TT, are visible from the console of the TSN switch directly attached to the DS-TT. These synchronization results are visible through the GUI offered by the TSN switch. To achieve the experimentation results, a simple bash script that scrapes the synchronization results from the GUI is taken into use. The script goes occasionally, to the website page, retrieves the synchronization value reported and stores it in a text file.

To achieve substantial synchronization results test data, the command was run from Friday evening (17:00) to Monday morning (10:00). This resulted in around 637 synchronization results data points. A summary of the data is denoted in Table 13, which shows the minimum values recorded when synchronization was started as 0.9 μs and maximum to be around 15 milliseconds. The median values are about 328 μs. The summary data provided is just a general overview, to help understand better about synchronization and how it happens over a period, graph plots are more useful. MATLAB plots of the data during the whole duration of running the TSN modules gives a better trend and helps in further understanding the synchronization.

Table 13 - Summary of synchronization results.

Summary	Value [μs]
count	637.0
Mean	343.8
std	647.19
min	0.90
25th percentile	328.89
50th percentile	462.45
75th percentile	678.39

Max	15961.43

4.3.3 Conclusion of the test series of TSN synchronization

Figure 49 shows the plot using the whole dataset. As can be seen from the figure, during the first parts, when the TSN modules start, the synchronization accuracy fluctuates a lot. This continues for some time and even reaches a maximum value of 15961 μs. When the synchronization reaches the datapoint around 240, which is roughly after one day, the synchronization becomes stable with a synchronization accuracy of 328 μs. This is the synchronization value that stays for the rest of the time which is more than one day. This shows that the synchronization accuracy obtained in this the TSN modules implementation becomes stable over a period of time to a constant value.

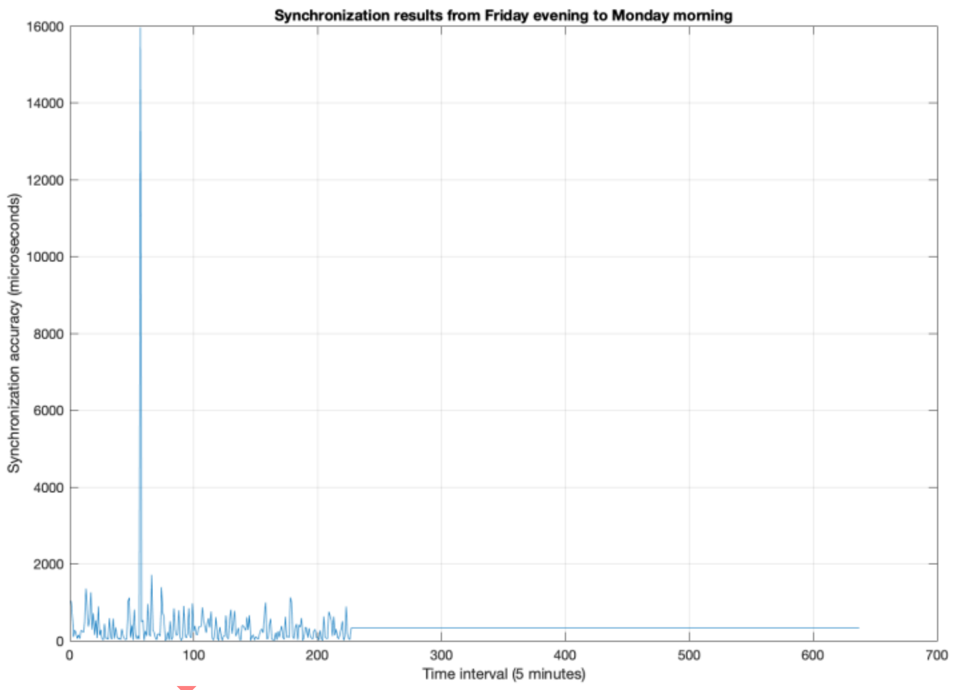


Figure 49 – Synchronization trend and eventual stabilization.

The results show that time synchronization of mobile devices using as clock reference a Grand Master in the fixed network is feasible. However, the variable delay of the radio link and the asymmetry of the delay in the radio link makes the accuracy lower than what is required for high-precision TSN devices.

As future work the usage of network slice isolated for time synchronization with similar delay for uplink and downlink could improve the accuracy of the synchronization.

5 Testing the IoT-NGIN components and evaluation of specific use cases

IoT-NGIN framework's holistic testing and evaluation requires the finalization of the integration activities that are necessary in order to realise the integrated IoT-NGIN platform prototype. The IoT-NGIN development, integration and testing plan, as described in section 2.2.3.2, indicates that the integration activities will conclude in M30. Therefore, in this section we provide an indicative selection of evaluation tests performed on IoT-NGIN developed components. The comprehensive integration tests will be presented in the following deliverable D6.3.

5.1 D2D Communication

5.1.1 Introduction

In cellular networks, D2D communication is defined as direct communication between two or more mobile devices without these being mediated by a Base Station. Moreover, D2D does not require direct communication with network infrastructures and therefore opens up a range of applications that can be used to enhance existing network infrastructures or leverage them to perform other independent services.

Within the fundamentals of D2D, we find that it operates in both inband and outband spectrum. D2D shares the same spectrum with cellular networks in the inband, resulting in it having a dedicated portion of the spectrum or competing for its resources. In contrast, in outband, D2D uses an unlicensed spectrum, typically being the ISM bands. Therefore, D2D communication is correlated with the integration of several available technologies, being LTE direct, Zigbee, Bluetooth Low Energy (BLE), or WiFi-Direct (WFD) part of them. However, the selection of these technologies will depend on the trade-off generated by them and the required use case.

One of the applications of IoT-NGIN is the realization of an implementation to extend network coverage. To this end, D2D is intended to be used to enable an out-of-coverage device to relay to another 5G-connected device in order to provide service to it (see Figure 50).

However, given the nature of mobile wireless technologies, the out-of-coverage device must be able to identify relay-enabled devices for coverage extension. In addition, it must have mechanisms that allow it to self-connect to other relays while in motion when necessary. It should be noted that in the event that an out-of-range device is in range of two or more relay devices, it should have mechanisms to select the relay that best suits its needs.

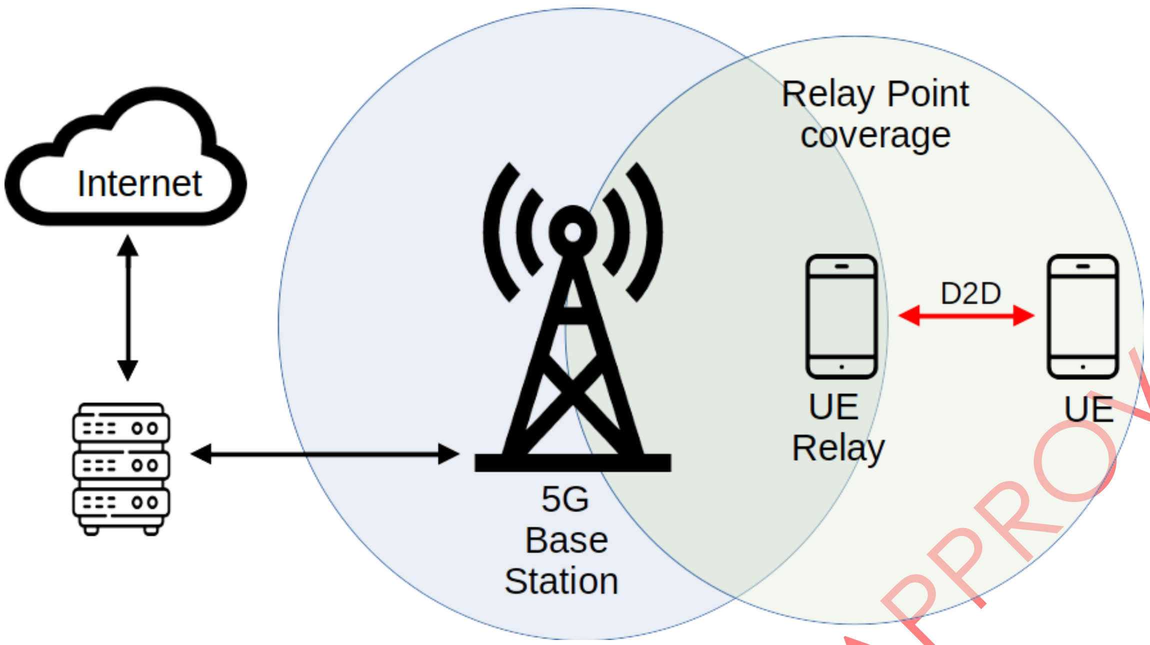


Figure 50 – D2D Out of coverage UE.

5.1.2 Testing Scenario

To establish a connection between an out-of-range device and the outside, we have created a TCP/IP connection scenario between UEs and a socket server. Our goal is to provide alert messages from the server to any UE registered to it. For this purpose, Sorbonne University and EBOS have set up the following configuration.

5G LAN: To maintain a controlled environment, a 5G LAN will be provided where all devices will belong to the same network using the same 5G core.

Message Forwarding: Given the peer-to-peer nature of the D2D, we have established two roles within the D2D. On one side, we have the UE relay, which will be any UE device located within the coverage radius of the 5G network. On the other hand, we have the out-of-coverage User Equipment (UE), which encompasses all UE outside the 5G coverage radius but within the range of the UE relay. Based on this, the UE relay must forward all messages corresponding to its D2D out-of-coverage UE.

Client-server architecture: To facilitate traffic between the UE and the outside, we have developed a TCP socket server to register the devices with which it will communicate. This includes both UE relay devices and out-of-coverage UEs. Then, the TCP server is notified with routing updates from each UE to create its routing table to establish communication with the registered UEs. This way, the TCP server will know which relay to address if a message is forwarded to a UE outside the coverage area.

Therefore, the configuration will consist of a 5G LAN directly connected to a TCP server and covering a set of mobile devices. Additionally, another group of devices will be located out of the coverage range of the 5G LAN (Figure 51).

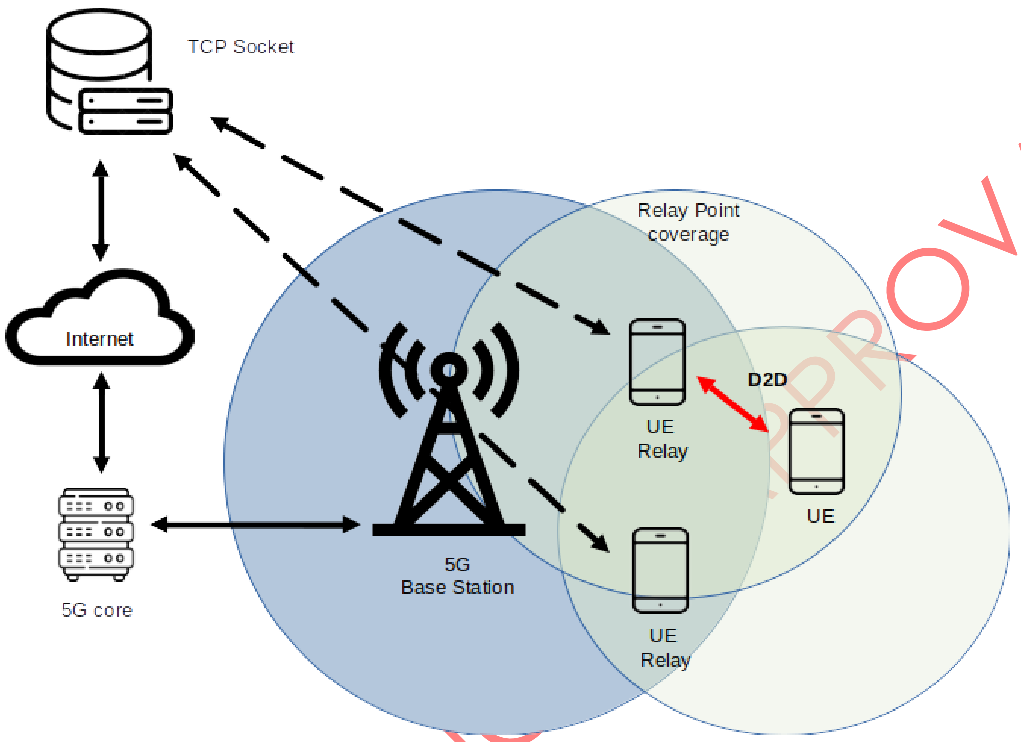


Figure 51 – 5G LAN directly connected to a TCP server and covering a set of mobile devices.

5.1.3 D2D Application

To establish D2D communications between UEs, Sorbonne University has developed an application for Android platforms called AtomD. This application allows devices to engage in D2D communications by performing device discovery via Bluetooth 5 and establishing communication links via WiFi direct. The current version of this application enables users to perform two ways of approaching D2D.

The first is to allow users to perform experiments to evaluate the performance of D2D. These include transmitting a single message over chunks of data, transmitting a fixed-size binary file over multiple chunks of data, and running various automatic link establishment procedures through a Discoverer node. From the tests performed to evaluate D2D, we conducted experiments to assess the time it takes for two devices to establish a D2D link using Bluetooth Classic and BLE. More specifically, we evaluated the delay of a Discoverer node (Figure 52) in performing neighbour discovery and connection establishment with an Advertiser node between distances of 0 m, 20 m, 40 m, 60 m, 80 m, and 100 m (Figure 53). A test has also been performed by transmitting binary files using the exact distances mentioned above.

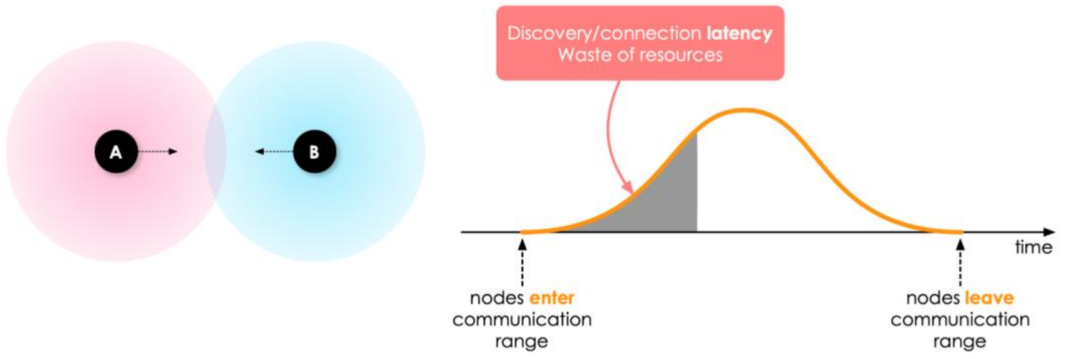

A
B
Discovery/connection latency
Waste of resources
time
nodes enter communication range
nodes leave communication range

Figure 52 – During D2D data exchange, part of the timespan is used for the connectivity process between devices.

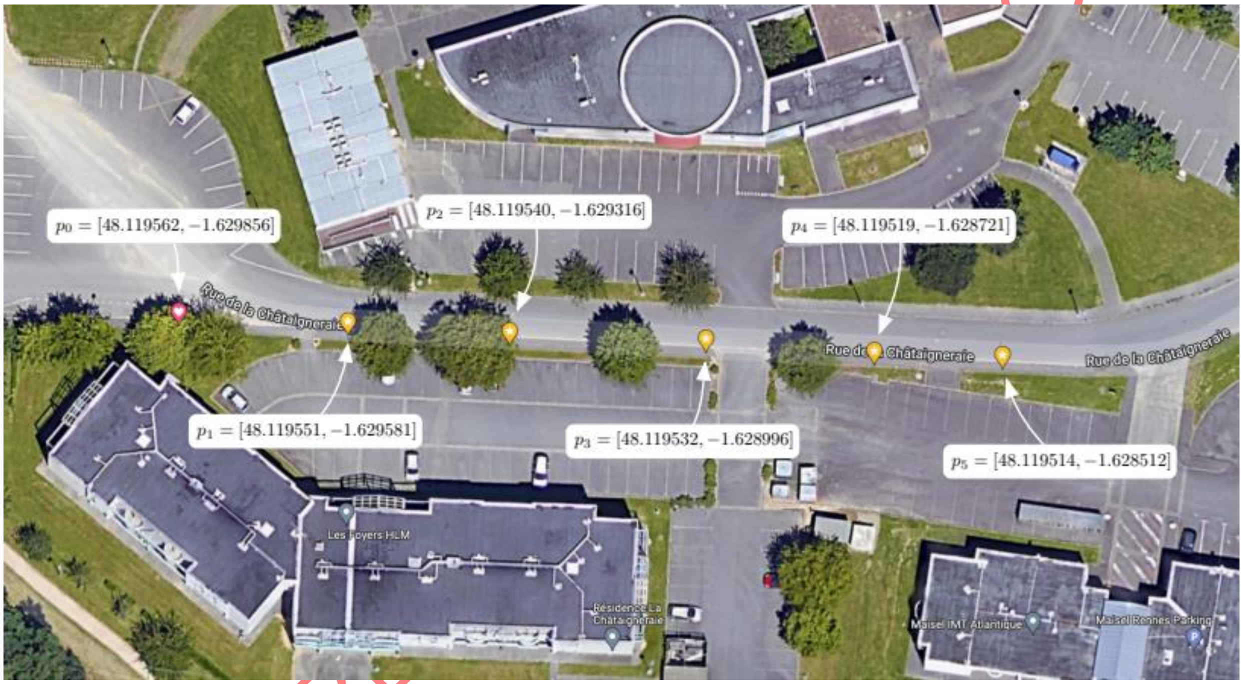

$p_0 = [48.119562, -1.629856]$
$p_1 = [48.119551, -1.629581]$
$p_2 = [48.119540, -1.629316]$
$p_3 = [48.119532, -1.628996]$
$p_4 = [48.119519, -1.628721]$
$p_5 = [48.119514, -1.628512]$
Rue de la Châtaigneraie
Les Foyers HLM
Résidence La Châtaigneraie
Maisel IMT Atlantique
Maisel Rennes Parking

Figure 53 – Experimental set-up at different distances where the discoverer is located at p_0 while the advertiser is located at distance from p_1 to p_5.

The second approach is to focus AtomD to make D2D connections with the feature of relay selection at the moment the devices are being discovered. In this way, if there is more than one candidate for a device to exchange traffic, then the relay selection algorithm will select the device that best suits the requirements of this device to perform a one-to-one communication. It should be noted that to establish connections to the outside, AtomD was provided with a library that allows connections to TCP socket servers in a persistent manner. This way, in case of any disconnection with any server, the application will retry in a constant way the re-connection to it.

5.1.4 Evaluation methodology

To manipulate the TCP socket server in a simplified way, Sorbonne University has developed a WebSocket-based daemon to communicate one or more web-based user interfaces and the TCP server.

This interface is made of a table with the necessary information to identify the different participating UEs and a text entry to send them a message. The table consists of 4 columns listed below:

Type: This column indicates which role the connected device belongs to, with Relay UE or Out-Of-Coverage (OC) UE as possible options. The UE Relay tag belongs to all devices directly connected to the TCP server. On the other hand, the OC UE tag identifies all the out of coverage devices that are connected to a UE relay.

Device ID: ID Corresponding to the device ID to which the row points. Its source can be found at https://developer.android.com/reference/android/provider/Settings.Secure#ANDROID_ID.

Connected To: Corresponds to the ID of the device to which the device is connected through D2D. In case no device is connected, this field will remain empty.

Send Message: In this column, there is a text field where the user can send messages to a device. If the device is a UE OC, the message will be sent to its corresponding relay to be forwarded later.

The life cycle of each connection

In order to register UEs on the TCP socket server, each newly connected device announces what type of role it is running. So, if the new connected device is a relay, it will communicate its information directly to the server. In case it is a new UE OC, it will tell the UE relay to inform the server of its existence. Then the server will register the UE OC on the same output as the UE relay.

In the event of disconnection by a UE OC, the UE relay will notify the server to update its routing table and unlink the UE OC from the connected devices. If a UE relay has disconnected, the server will remove that node from its registry, and the UE OC linked to it if it exists.

Finally, if a message has to be sent, the server identifies to which output the message has to be sent. In addition, it will provide within the message to be sent the ID of the destination UE to which it is to be communicated. Thus, if the message is sent to a UE OC, the UE relay can identify if this message belongs to it or should forward to its UE OC.

5.2 edgePMU

The RWTH Real time laboratory as described in D6.1 [3] is utilised to test the latest revision of the edgePMU now featuring integrated 5G connectivity. The laboratory allows for comparison of different types of PMUs under well controlled conditions.

The testing at has two major objectives.

- **Objective 1**: Investigate if the hardware modifications, that enable 5G connectivity, impact the measurement result precision.
- **Objective 2**: Investigate the impact of different real life communication channels. Specifically wired LAN as reference channel, then LTE and finally communication via a private 5G network.

5.2.1 Test description

Reference PMU

As reference measurement device, a commercial Alstom MiCOM P847 PMU is used. This device allows for extracting the phasors with the synchrophasor protocol C37.118 [35]. The phasors are then forwarded to a central database and compared to other measurements.

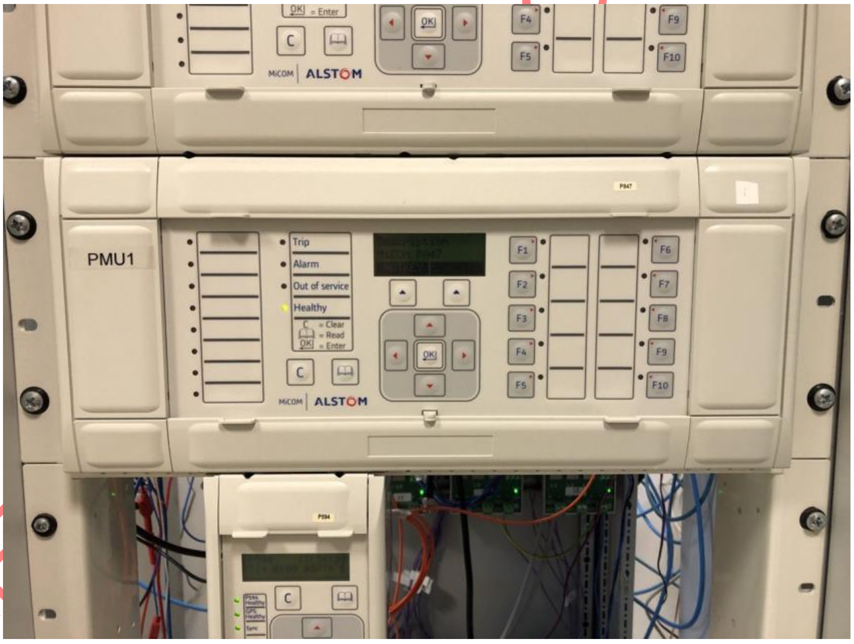


Figure 54 – FAlstom MiCOM P847 PMU.

Signal generator

As a signal generator, a TTI TG2000 [36] is used. This signal generator can provide up to 20 MHz with an amplitude of 10V on a single channel.

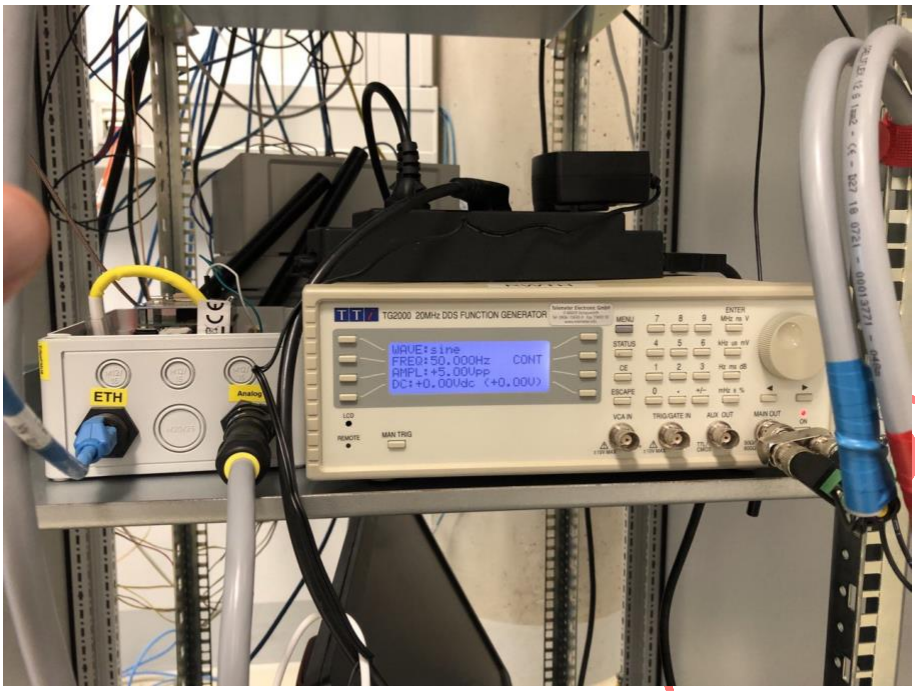


Figure 55 – TTI TG2000 DDS Function Generator.

Real time simulator

The RTDS real time simulator provides a test bed for PMUs. This test bed is utilised by using an analog output card and generating predefined signals in the time domain. As real time simulator a RTDS [37] device is used.

5.2.2 Testing scenarios

Scenario 1

In scenario 1 a generated signal is provided by a signal generator and both PMUs are synchronised via GPS. The measurements are sent to a database and analyzed. The schema of this scenario is shown in Figure 56.

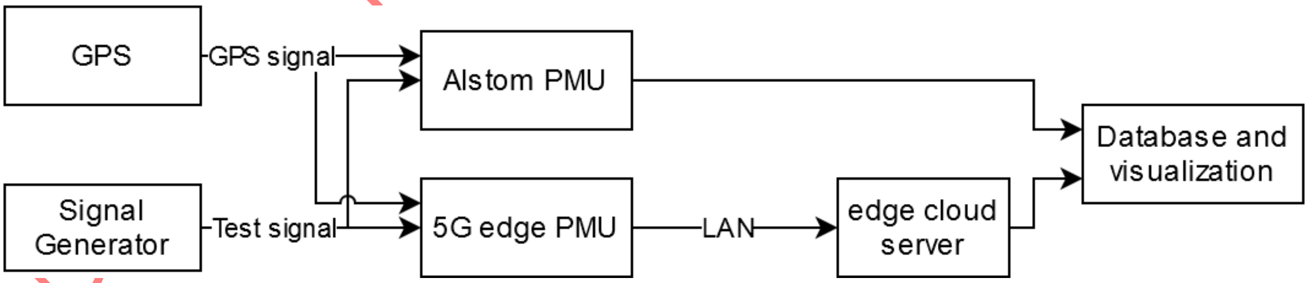


Figure 56 – 5G edgePMU scenario 1.

Scenario 2

In scenario 2 the real time simulator is generating the test signal waveform as well as a synchronization signal. With this approach, the measurements of the edgePMU are referenced to a virtual time base provided by the real time simulator. The results are then fed back to the real time simulator and compared to the expected values. The schema of this scenario is shown in Figure 57.

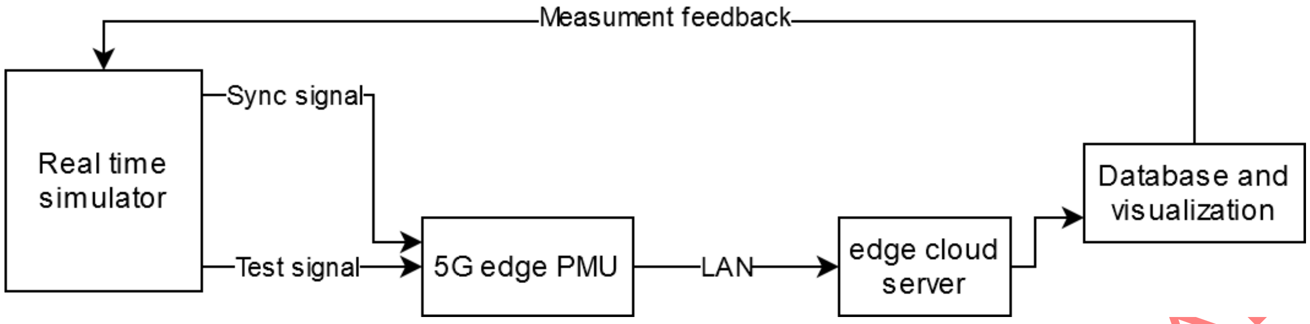


Figure 57 – 5G edgePMU scenario 2.

Scenario 3

In scenario 3 the 5G edgePMU is synchronised with GPS and signal generator is providing the test waveform. This test is done with different kinds of test networks and will provide insights into the real-life performance of the network under test. The schema of this scenario is shown in Figure 58.

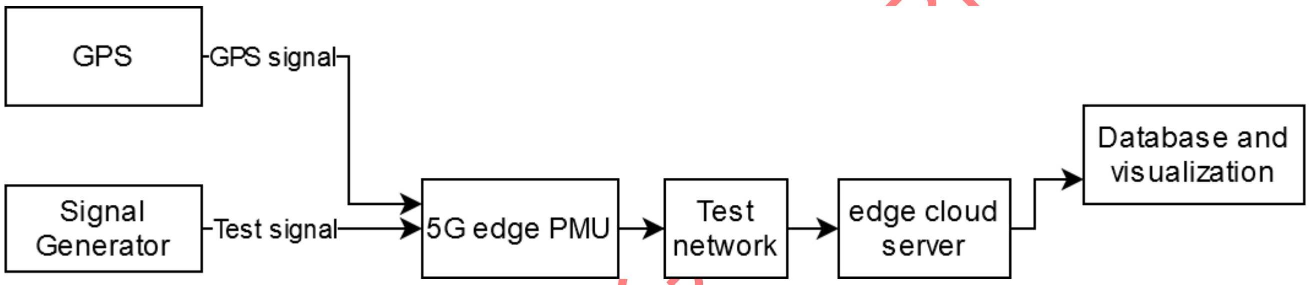


Figure 58 – 5G edgePMU scenario 3.

5.2.3 Tests for edgePMU Objective 1

Steady state analysis (Scenario 1)

- Run a test with a fixed frequency input and varying amplitudes.
- Run a test with fixed amplitude input and varying frequency.

Dynamic analysis (Scenario 2)

- Run the RTDS PMU test bench with different parameters

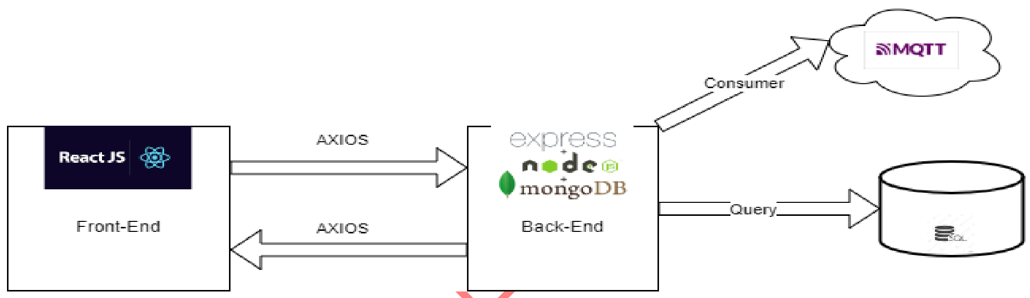
5.2.4 Tests for edgePMU Objective 2

Communication tests (Scenario 3)

- Set sampling rate to 10 kSmps/channel and measure the network parameters for LAN, LTE and 5G
- Set sampling rate to 20 kSmps/channel and measure the network parameters for LAN, LTE and 5G
- Set sampling rate to 50 kSmps/channel and measure the network parameters for LAN, LTE and 5G
- Set sampling rate to 60 kSmps/channel and measure the network parameters for LAN, LTE and 5G.

5.3 DSO Dashboard

5.3.1 Test description

Test: Validation of operation of AR app for EV charging station recognition	
Objective	The main objective of this test is to show data coming from smart meter and ML framework if inference is requested. This means for the CPO to be able to control production and energy consumption as well as show inferences result rely on the MLaaS service.
Components	• DSO Dashboard  MQTT Consumer React JS Front-End AXIOS AXIOS express node mongoDB Back-End Query SQL Figure 59 – DSO dashboard architecture.
Features to be tested	In order to increase network stability, DSO leverages the huge amount of data gathered from IoT devices to have real-time evaluation of the smart grid parameters. With this aim, data are continuously collected from the field and shown on the dashboard views. Moreover, in order to estimate energy demand for the DR campaign, DSO must have consumption and production forecasts at the grid level. Forecasting of energy demand, which will be calculated every day.
Requirements addressed	REQ_SE1_NF01, REQ_SE1_NF02
Test setup	This dashboard will be in place in the operator premises. For testing the OneLab infrastructure is taking in account.
Steps	**User registration** • The user accesses the application page • The user clicks on the sign-up link • In the registration page the user enters all the necessary data and clicks the SIGN UP button • A notification message if the registration was successful
	Login • The user accesses the application page • The user log-in with previously registered email and password.

	• The user clicks on the SIGN IN button • Device information divided by table is shown
	Navigation in home page • The user logs in • The user verifies that the data displayed in the table matches the device.
	Visualization of data • The user logs in • The user clicks on the tab choosing time span and the service needed • The dashboard shows the field devices consumption and or production data
	Prediction visualization • The user logs in • The user clicks on the device tab and forecast buttons and make a request for a given time interval • The ML inference as result is showed on the dashboard

5.3.2 Test outcomes

Sign Up

The page allows the registration of a new user.

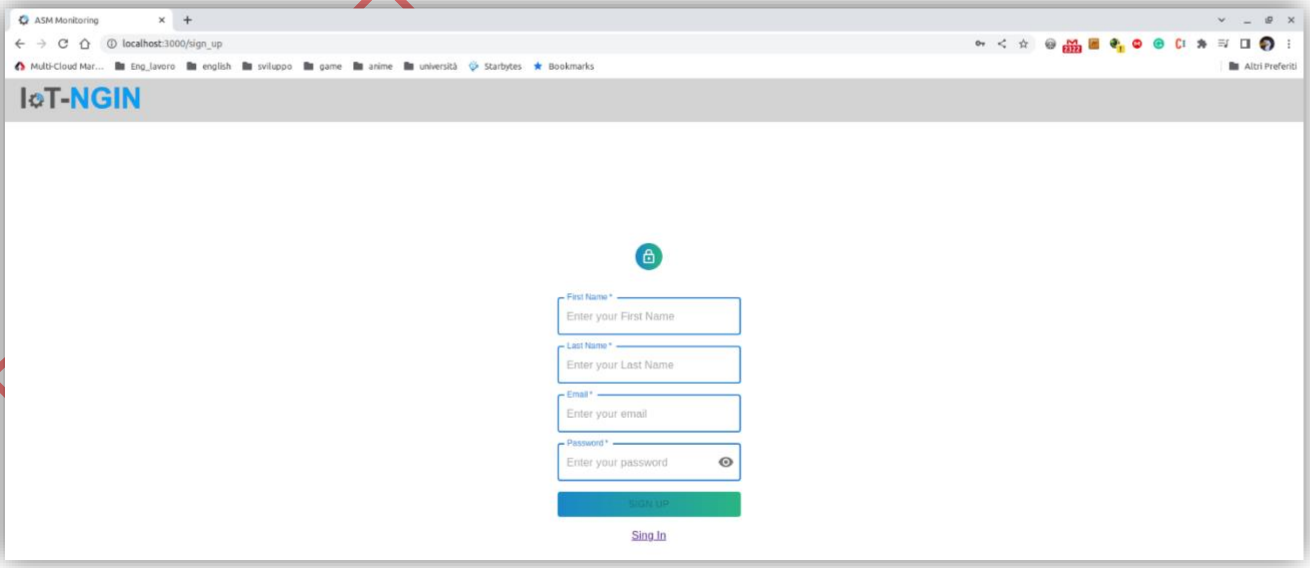


Figure 60 – SIGN UP page in the DSO Dashboard

Login

The page allows you to log in to get access to the monitoring tool (local database).

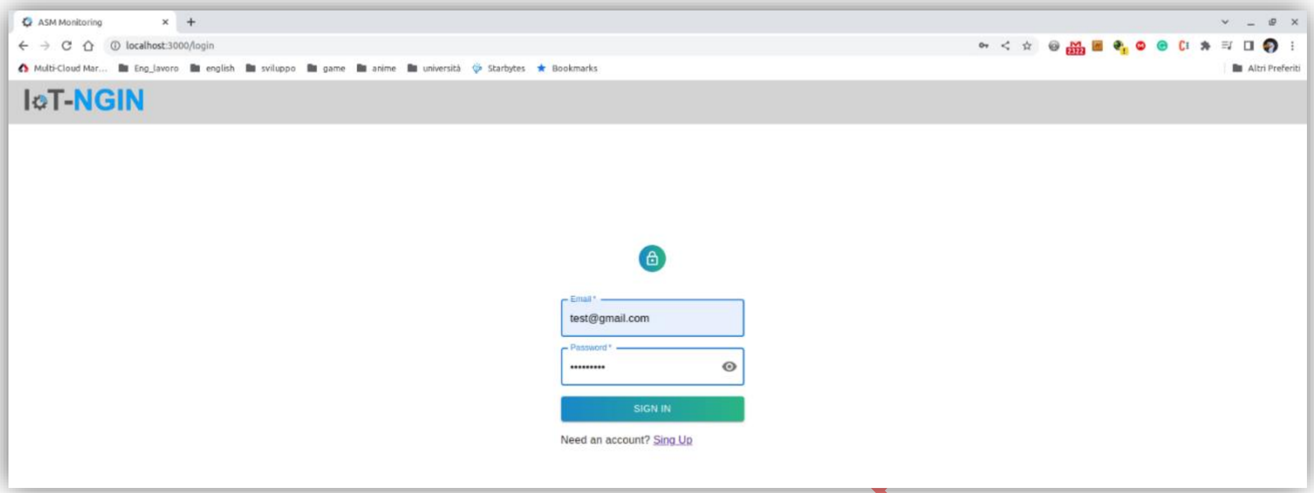
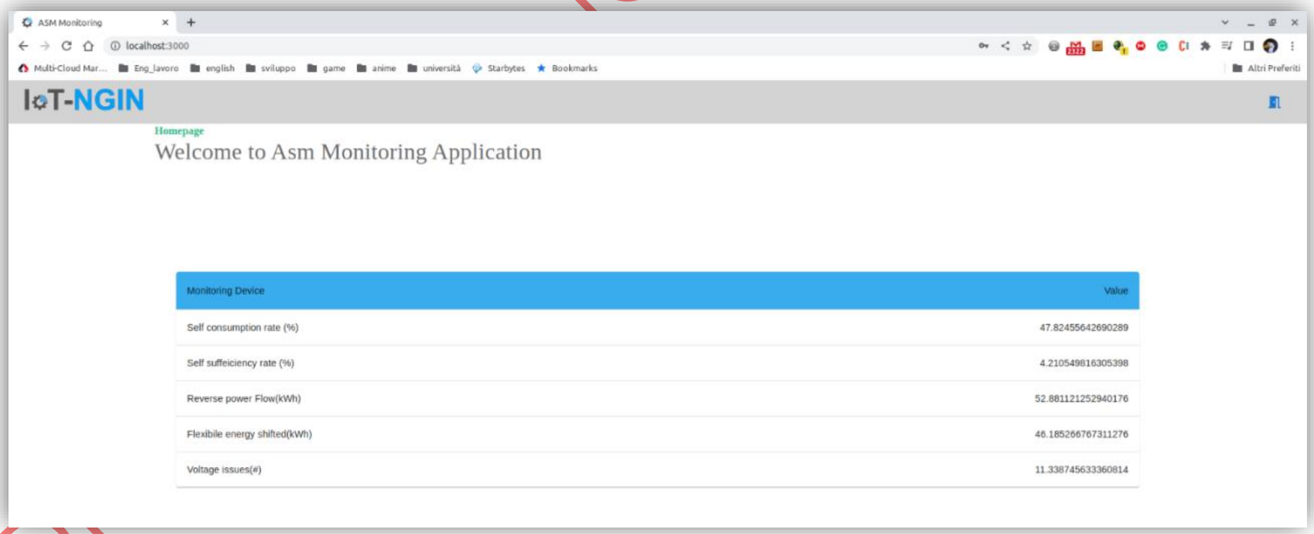


Figure 61 – SIGN IN page in the DSO Dashboard.

Navigation in Home page

The page is currently integrated with the MQTT bus provided by ASM, for now it shows the most important fields of the device in a table and is still under development. It is being decided whether to show graphs on this page as well.



Welcome to Asm Monitoring Application

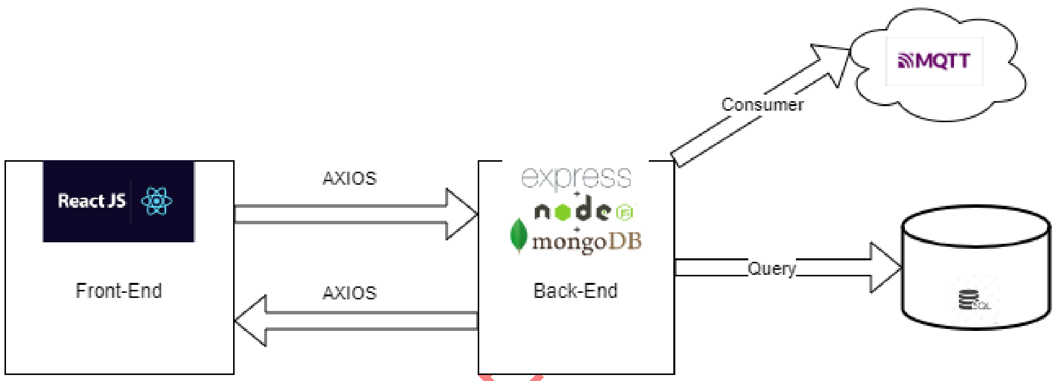
Monitoring Device	Value
Self consumption rate (%)	47.82455642690289
Self suffeiciency rate (%)	4.210549816305398
Reverse power Flow(kWh)	52.881121252940176
Flexibile energy shifted(kWh)	46.185266767311276
Voltage issues(#)	11.338745633360814

Figure 62 – home.

5.4 Augmented Reality App

These tests have the main purpose to demonstrate the planned tests for the Augmented Reality (AR) app that will be delivered early 2023. For the meanwhile we are describing the different test cases that will complete the full integration of the AR in the LL. The planned test is specified in Table 14.

Table 14 – Test specification for AR app (UC10).

Test: Validation of operation of AR app for EV charging station recognition	
Objective	The objective of this test is to recognise a given EV charging station, to identify it and to augment information on a mobile phone display. Finally, some commands are sent and executed on the real device.
Components	• AR tool for EV charging station recognition  Consumer MQTT React JS AXIOS express node mongoDB Front-End AXIOS Back-End Query Figure 63 – High-level architecture of the AR application for EV charging station recognition (UC10).
Features to be tested	The main purpose of this test is: • to check if the AR app is work accordingly with device indexing to identify the component • to check if the AT app is working properly and it is capable to augment the EV charging station image in the mobile camera • It is able to send command to the EV charging station
Requirements addressed	REQ_IN1_F10, REQ_IN1_F03, REQ_IN1_F06
Test setup	The application will be hosted in edge LL installation and the preliminary tests will start using also OneLab infrastructure The following data will be used in testing phase depending on the needed results • Logos and images to identify the EV charging station (EVCS) • GPS measurements to identify and locate the EVCS • VUFORIA database artifacts hosted in the cloud • Device indexing data from digital twin, for visualization • Data will send to the EV charging station to be applied to the real asset. This will be done through device indexing southbound interfaces.

Steps	• Camera on mobile phone connection • Retrieving real time images • Track of a target image in the EV charging station (EVCS) • Detection and localization EV charging station with Unity and GPS service • Identification of EVCS • Extraction data from Device Indexing • Augmented reality data presented on mobile camera phone • Data command send through device indexing southbound interfaces

5.5 Testing of the 5G Device Management API with IoT-NGIN use cases

Ericsson provided a laboratory infrastructure with a prototype 5G network to perform the functionality testing of the 5G Device Management API and the demonstration of its features with a Graphical User Interface (GUI). In this chapter, the following topics will be explained in detail:

- The objectives related to the 5G Device Management API,
- The features of the 5G Device Management API relevant for the IoT-NGIN use cases,
- The objectives of the tests of the 5G Device Management API,
- The laboratory infrastructure used for testing the proof-of-concept implementation of the 5G Device Management API,
- The limitations of the 5G Device Management API proof-of-concept implementation,
- The 5G-enabled advanced IoT-NGIN use cases defined for the smart agriculture and smart industry domains within the project, and
- A demonstration scenario investigated and presented during the first project review with an IoT-NGIN smart industry use case.

5.5.1 Introduction

In this subchapter, the objectives of our work related to the 5G Device Management API, the features of the API, the objectives of the tests of the API, the 5G Device Management API laboratory infrastructure, and the limitations of the 5G Device Management API proof-of-concept will be described.

Objectives related to the 5G Device Management API

Currently, 5G is becoming more important and accessible for various industrial domains. As many industrial domains consider deploying and using 5G in their environments, it becomes increasingly important to make the capabilities of 5G easier to access for users and developers working in these domains. The exposure of 5G network capabilities to users will not only enable faster development and integration of new services with 5G, but also faster adoption of 5G into industrial infrastructures. Therefore, we investigated requirements of IoT-NGIN use cases to see whether they could be addressed by the 5G Device Management

API and to define which 5G Device Management API features would be relevant and beneficial for the IoT-NGIN use cases.

One of our objectives was to define the requirements of IoT-NGIN use cases related to the API and a second was to define new 5G-enabled advanced use cases for vertical domains studied in IoT-NGIN. Starting from the current IoT-NGIN use cases, we defined 5G-enabled advanced use cases based on the likely future evolution of current IoT-NGIN use cases and prepared a demonstration of an advanced use case with the 5G Device Management API. This study helped not only to describe and evaluate the potential benefits of the API in these vertical domains, but also it helped to identify new requirements and new use cases to be investigated and proposed to standardization bodies.

Features of the 5G Device Management API

The concept of having a 5G Device Management API for allowing users to manage the connectivity of their devices is described under the 3GPP study called "Service Enabler Architecture Layer for Verticals (SEAL)" and this description is being standardised in 3GPP [38]. The specifications and features of 5G Device Management API are continuously investigated and defined by the 3GPP SA6 group.

The features of the 5G device management API proof-of-concept relevant for the IoT-NGIN use cases were defined and summarised in D2.2 [12]. For simplicity, a summary table is included below in Table 15.

Table 15 – Descriptions of the 5G Device Management API features.

5G Device Management API feature	Description of the feature
Provisioning and onboarding devices	It allows users to provide the unique identifiers of their 5G devices to the 5G network, so that these devices are accepted by and attached to the 5G network.
Getting a list of devices	It allows users to get a list of devices that are attached to the 5G network.
Creating, modifying, and removing device groups	It allows users to create device groups having same or different purposes, unite some devices in the same group and remove a device from a group.
Monitoring the quality of the communication links of devices	It allows users to monitor the quality of the communication links (e.g., signal strength, packet loss, latency) of their 5G devices.
Defining and changing QoS parameters of individual device connections	It allows users to set and change the values of Quality-of-Service (QoS) parameters for communication links of devices.
Getting location information of devices	It allows users to get the geographical location of their connected devices to track their mobility.

Objective of the tests with the 5G Device Management API

The objectives of tests with the 5G Device Management API were to create a test scenario to realise an IoT-NGIN use case in a lab environment with the API, to use the 5G Device Management API as an authorised user in order to test the functionality of its features, to validate that the features are working, and to demonstrate the API features through its GUI with one of the 5G-enabled advanced use cases defined for the smart industry domain within the project.

The 5G Device Management API laboratory infrastructure

In the laboratory, a prototype private 5G network, in which the 5G Device Management API proof-of-concept had been deployed, was used. The following components were integrated into the implementation:

- Raspberry Pi acting as data generator/s (e.g., devices of the use case),
- 5G edge cloud acting as data receiver/s (e.g., AI/ML services from IoT-NGIN technologies used in the use case),
- 5G Device Management API proof-of-concept hosted in the 5G mobile network,
- Graphical User interface (GUI) of the API hosted in the 5G mobile network,
- 5G radio dot, and a
- 5G radio modem.

Figure 64 below shows the overview of the laboratory infrastructure in which the 5G Device Management API was deployed. As illustrated with "yellow" lines in Figure 64, the infrastructure allows authorised users to use the GUI of the API to manage and monitor the 5G connectivity of devices, establish end-to-end communications between 5G devices and the edge cloud. After establishing all necessary connections through the API, the user can start sending data streams from 5G devices acting as data generators to the edge cloud acting as data receivers that is shown with "black" solid and dashed lines.

A local area private 5G network hosting the 5G Device Management API proof-of-concept implementation was used in the laboratory as seen in Figure 64 and several Raspberry Pi devices connected to different 5G radio modems were available.

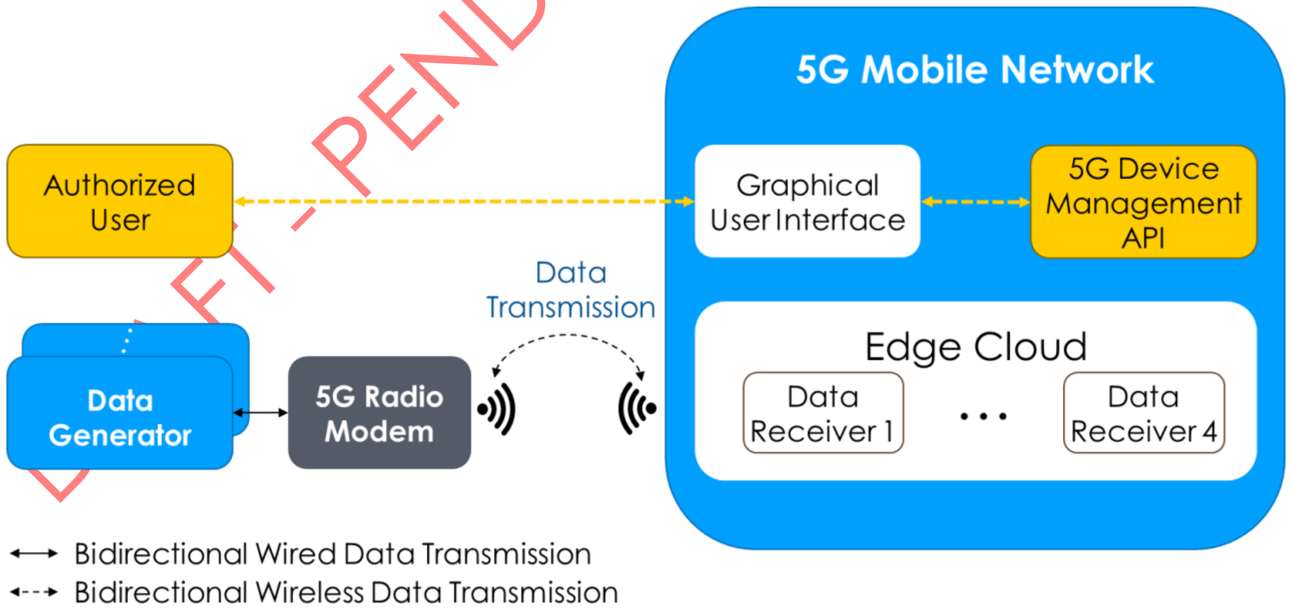


Figure 64 – 5G Device Management API Laboratory Infrastructure.

The required 5G connections between endpoints were provided using the graphical user interface (GUI) of the 5G Device Management API. The authorised user could send requests to the 5G Device Management API and effectively to the relevant core network functions through this GUI. Upon receiving the commands from the GUI, the 5G Device Management API further transmitted the commands to the related 5G network functions to perform the requested operation, such as adding a device to a device group. The success or failure info about this operation was then shown to the authorised user in the GUI. The communication illustrated with "yellow" lines allows the user to:

- manage and modify the 5G connectivity of devices,
- monitor the quality of the connectivity status of devices,
- create and remove device groups,
- manage the members of the device groups, and to
- modify the QoS levels of the connectivity of individual devices or devices within groups in the 5G mobile network.

While preparing a demonstration with the 5G Device Management API and its GUI, the functionality of relevant API features was tested. For example, we observed whether or not the devices were successfully onboarded to the 5G network, the requested device groups were successfully created, the devices were successfully added to the suitable device groups and the quality of the communication links of the devices were monitored in terms of latency and packet loss etc. were maintained.

Limitations of the 5G Device Management API proof-of-concept implementation

- The laboratory 5G network hosting the 5G Device Management API proof-of-concept implementation was a prototype network. The components were configured to enable new features through the API proof-of-concept and some of the configurations included solutions which have not yet been standardised. Therefore, the prototype network does not have the same characteristics or configurations as commercial 5G standard networks currently have.
- Currently, chips in many 5G devices do not provide the capability to have more than one active data transmission for a device, meaning that a device cannot send two different data streams at the same time from two different radio link channels. Therefore, a device can only have one active connection in a device group created by the API to send its data to a service hosted on edge cloud.

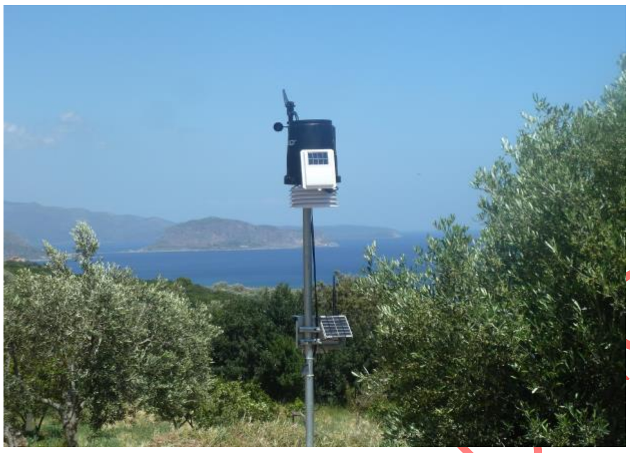
5.5.2 Defining a 5G-enabled advanced smart agriculture use case

This subchapter describes a 5G-enabled advanced IoT-NGIN use case for the smart agriculture domain. This use case was defined based on the investigations on IoT-NGIN use case requirements and investigations of how the 5G Device Management API could be beneficial for them in future. From the smart agriculture domain, one of the IoT-NGIN use cases, specifically the IoT-NGIN Use Case 4 – "Crop diseases prediction, Smart irrigation and precision aerial spraying", was investigated to define a more advanced use case including the 5G Device Management API. Both the original use case and the 5G-enabled advanced use case will be described in this subchapter.

The IoT-NGIN Use Case 4 - Crop diseases prediction, smart irrigation, and precision aerial spraying

IoT-NGIN Use Case 4 focused on using the data coming from sensors, is illustrated in Figure 65. Data from SynField devices developed by SYN, drone camera video images of the soil, leaf and weather conditions and ML services developed by IoT-NGIN combine to better predict the growing conditions of the crops and the need for further aerial spraying, as described in D1.1 [39]. The main objective of this use case is to detect diseases on crops and leaves located in the green field and to optimise the precision on aerial spraying based on real-time video streams coming from the drones [39].

In a smart agriculture field, many sensors would be located in the field to measure the temperature and humidity conditions of the soil and leaves, with several SynField devices and drones being deployed. To represent this use case on a field trial scale, Figure 65 illustrates, a single SynField device being used to integrate data from a number of sensors, aggregating their data and transmit it to the relevant ML services from IoT-NGIN technologies (on the left), and a drone equipped with a multi-spectral camera used to capture videos and images of the crops and leaves and to perform aerial spraying whenever necessary (on the right).



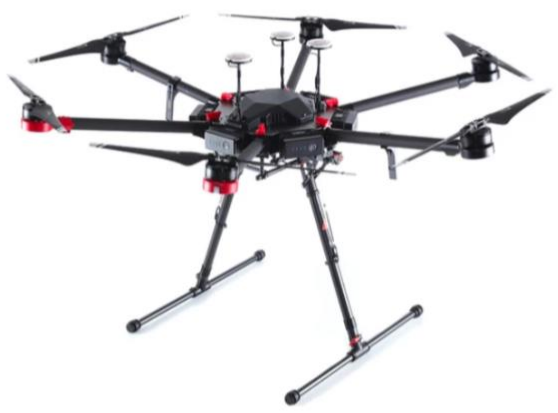


Figure 65 – A SynField device deployed in the field, a drone used for the crop disease detection and aerial spraying.

The 5G-enabled advanced smart agriculture use case based on the IoT-NGIN Use Case 4

Because the large-scale deployment of this use case requires many devices (e.g., SynField devices and drones) to be connected to the IoT-NGIN ML services to optimise disease prediction and irrigation, the use of the 5G Device Management API was considered as beneficial for the smart agriculture use case described above. The 5G Device Management API can be used to connect various devices to the 5G network, to create device groups, to set different QoS levels for the data traffic and to monitor the connectivity status of the devices. An advanced use case and a solution architecture for this use case were defined, including the 5G-API deployment, based on the IoT-NGIN's smart agriculture use case 4 and illustrated in Figure 66.

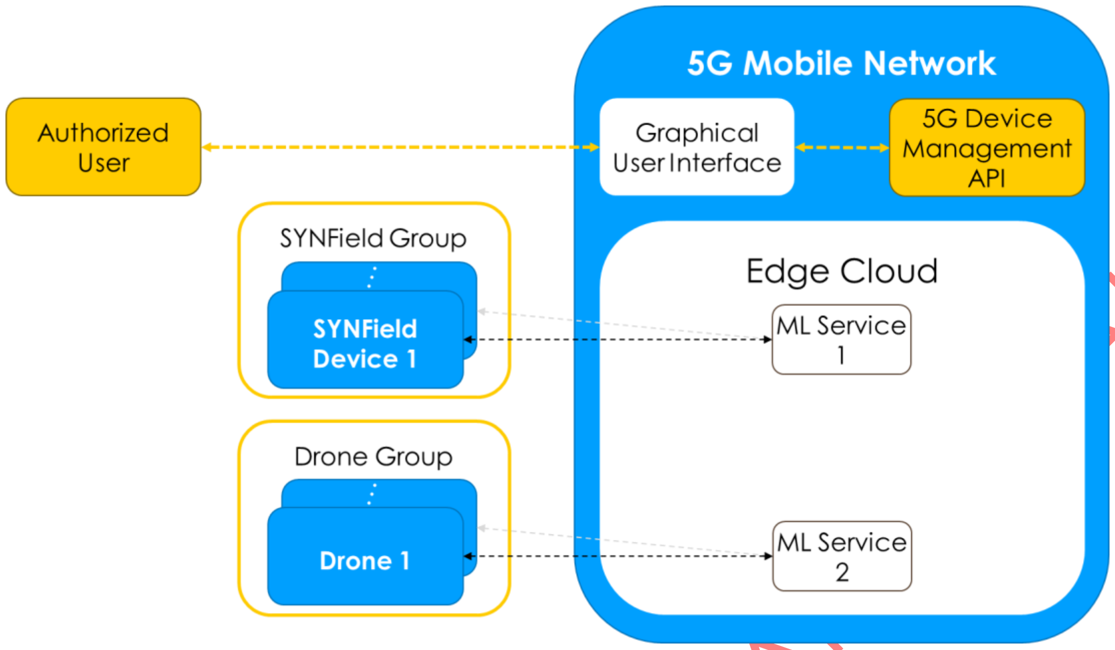


Figure 66 – Solution architecture defined for the 5G-enabled advanced smart agriculture use case.

As seen in Figure 66, the solution architecture consists of an authorised user from the smart agriculture domain, such as a farmer working in a field, a 5G mobile network hosting the 5G Device Management API, its GUI and edge cloud for hosting corresponding ML services from IoT-NGIN technologies, and some devices that were considered for the use case, such as SynField devices and drones. As the characteristics of the data being transmitted from SynField devices and from drones are different in terms of their load, transmission frequency, required bandwidth, and required latency, we considered separating their traffic and splitting the devices into two different device groups. Thus, the farmer can group all SynField devices into a "SynField Group" carrying the sensor data and she can modify the requirements of their communication links by changing their QoS level, whereas she can group all drones into a "Drone Group" carrying the video streams from the field and setting their QoS parameters to higher values. In addition, as the data processing of the sensor data and the video streams is performed by different services from IoT-NGIN technologies, having device groups can help the farmer manage connectivity to devices as groups and connect them to the relevant service hosted on an edge cloud, isolating the traffic of different devices as needed.

The potential benefits of the 5G Device Management API defined for the 5G-enabled advanced smart agriculture use case

The potential benefits of the 5G Device Management API based on the IoT-NGIN use case 4 were described in IoT-NGIN D2.2. To save the reader looking for this source, a summary of the benefits is included below as Table 16. The examples given for IoT-NGIN smart agriculture use case 4 are not only related to the current status of the use case studied in IoT-NGIN, but they also include potential future use cases as use case 4 evolves in the coming years.

Table 16 – Potential benefits of the 5G Device Management API to the 5G-enabled advanced smart agriculture use case.

5G device management API feature	Example - Why does a smart agriculture use case need or benefit from this API feature?

Provisioning and onboarding devices	Devices that are used in the smart agriculture use cases can be onboarded to the 5G network easily by a farmer without the need to contact a mobile operator or cellular systems expert. In addition, as each of these devices will have a unique ID. These unique IDs can be used to list devices and to do further analysis, such as vulnerability checks, on each device.
Getting a list of devices	The IoT vulnerability crawler being developed in WP5 in IoT-NGIN uses a "list of devices" as a component for security checking. If 5G can provide this information directly to the vulnerability tool, then the integration of new devices to the IoT-NGIN platform would be easier, digital twins can be created more quickly and vulnerability crawling process can start earlier.
Creating, modifying, and removing device groups	Creating device groups per specific purpose would be relevant for devices used in the smart agriculture use cases. For example, groups of devices could indicate e.g., devices monitoring a certain crop type or devices belonging to a single farmer.
Monitoring the quality of the communication links of devices	If the data rate of the communication link drops due to bad radio conditions, an alert could be sent to the drone inspecting the field. Based on the alert, the drone could change to a lower resolution codec and thereby adapt the video streaming quality to the link capacity.
Defining and changing QoS parameters of individual device connections	If the QoS parameters for the communication links of drones are low, the resolution of the images taken by the drone camera would also be low. By changing or increasing the relevant QoS parameters of the communication link, the resolution of the image or video could be increased, and the diseased crop can be identified more easily.
Getting location information of devices	For future use cases, it would be relevant to receive the coordinates of the drones location, e.g., the drone can be automatically moved from one location to another to perform aerial spraying of specific areas with diseased crops.

5.5.3 Defining a 5G-enabled advanced smart industry use case

This subchapter describes a 5G-enabled advanced IoT-NGIN use case for the smart industry domain. This use case was defined based on the investigations on IoT-NGIN use case requirements and of how the 5G Device Management API could be beneficial to them in future years. From the smart industry domain, one of the IoT-NGIN use cases, specifically the IoT-NGIN Use Case 6 – "Human-centred safety in a self-aware indoor factory environment", was considered as the basis for defining a more advanced use case including the 5G Device

Management API. Both the original use case and the 5G-enabled advanced use case will be defined in this subchapter.

The IoT-NGIN Use Case 6 - Human-centred safety in a self-aware indoor factory environment

Figure 67 illustrates IoT-NGIN Use Case 6, which is based on the needs of a factory owned by IoT-NGIN partner BOSCH. This use case focuses on detecting human workers in the factory with safety cameras, ultra-wide band sensors and Radio Frequency Identification (RFID) nodes and avoiding collisions between Automated Guided Vehicles (AGVs) and human workers walking on the factory floor as described in D1.1 [39]. The main objective of this use case is to improve the safety of workers in factories by enabling automated operations, particularly in those where workers and AGVs are working together [39].

In a factory setting, many such AGVs, safety cameras and workers will be on the factory floor. A representative scenario for this use case, illustrated in Figure 67, shows an AGV moving in a factory on the left, some workers walking in the factory floor in the middle section of the illustration and typical safety cameras in the factory used to detect people on the factory floor are shown on the right. The location data from sensors, RFID nodes and cameras are transmitted in the uplink direction to relevant IoT-NGIN AI/ML services for processing, whereas the control data is transmitted in the downlink direction from the AI/ML services to the AGVs to perform specific operations such as “start to follow a new and optimal route”, or “brake as there is a human or obstacle nearby”.

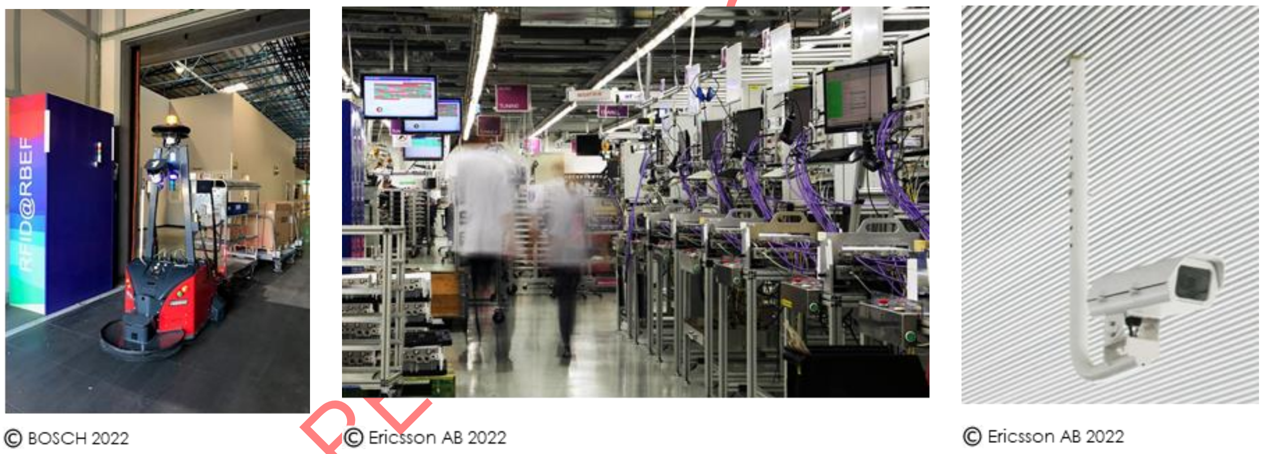


Figure 67 – Automated Guided Vehicles in the factory, workers walking in the factory, safety cameras deployed in the factory.

The 5G-enabled advanced smart industry use case based on IoT-NGIN Use Case 6

As many devices need to be connected to each other to maintain safety in a factory, the use of the 5G Device Management API was considered as beneficial for the smart industry use case described above. The 5G Device Management API can be used to manage 5G devices, connect them to the 5G network and monitor their connectivity statuses to support and improve automated operations and safety for the human workers in the factory. An advanced use case and a solution architecture for this use case were defined based on the IoT-NGIN’s smart industry use case 6 and illustrated in Figure 68.

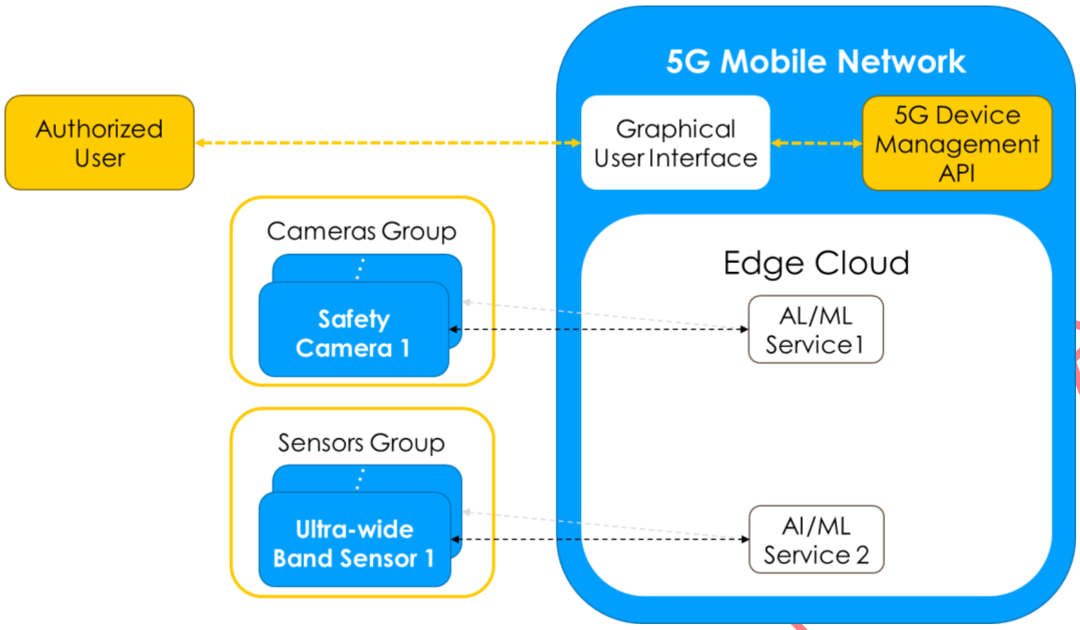


Figure 68 – Solution architecture defined for the 5G-enabled advanced smart industry use case.

As illustrated in Figure 68, the solution architecture consists of an authorised user from the smart industry domain, such as a human worker in a factory, a 5G mobile network hosting the 5G Device Management API, its GUI and edge cloud for hosting relevant IoT-NGIN AI/ML services, and some devices needed for the use case, such as safety cameras and ultra-wide band sensors. The characteristics of the data being transmitted from safety cameras and from ultra-wide band sensors differ as the safety cameras need to send video streams or pictures from the factory floor and require high bandwidth, whereas the ultra-wide band sensors need to send localization information about moving objects or obstacles and require less bandwidth compared to the safety cameras. For the use case, it was considered to split these two entities into two different device groups. Thus, the factory worker can group all safety cameras into a "Cameras Group" carrying the videos and pictures from the obstacles or moving objects in the factory and set the QoS requirements of their communication links to "high" values. Alternatively, she can group all ultra-wide band sensors into a "Sensors Group" carrying the localization information of the obstacles and objects in the factory.

In addition, to increase the security of the human workers in the factory and avoid potential incidents, the communications system needs to be reliable, the latency of data transmissions should be low, and the data processing should be done on the edge cloud, as near as possible to the devices. Having device groups can help to increase the system security, as the data traffic in one group is isolated from that of the other groups. This can reduce the chances of having communication problems in both groups for all devices and increase the chances of AI/ML services receiving at least some data to process and detect any potential collisions.

The potential benefits of the 5G Device Management API defined for the 5G-enabled advanced smart industry use case

The potential benefits of the 5G Device Management API for IoT-NGIN use case 6 are summarised in Table 17. The examples given for IoT-NGIN smart industry use case 6 are not only related to the current descriptions of the use case studied in IoT-NGIN, but they also include potential future use cases based on enhancing the scenario and communications features of use case 6.

Table 17 – Potential benefits of the 5G Device Management API to the 5G-enabled advanced smart industry use case.

5G device management API feature	Example - Why would a smart industry use case need/benefit from this API feature?
Provisioning and onboarding devices	Easy connection of devices or disconnection of devices from the 5G network is enabled.
Getting a list of devices	When there are many 5G devices available in a factory, it can be challenging to see which devices are powered on and still connected to the 5G network. Through the 5G device management API, a user can easily get a list of devices connected to the 5G network to see which devices are available and active in the network.
Creating, modifying, and removing device groups	Different communication priorities are needed for different operations in a factory. E.g., robot control is the most important operation in a factory and its data traffic needs high priority, whereas data being recorded for statistical purposes is not required so often, and this traffic could have a low priority compared to the data traffic for robot control. Device grouping helps in this situation by enabling the setting of different priorities for different machines, algorithms, and operations.
Monitoring the quality of the communication links of devices	Protocols used in Smart Industry use cases require deterministic communication. Knowing what packet loss is occurring and what latency the communication is experiencing is important. High latency in the communication to a production line could affect the whole line. E.g., if the 5G Device Management API provides the service with data on the current packet loss and latency of the communications, the service hosted on an edge cloud could take a proper action at an early stage avoiding the production line being totally stopped.
Defining and changing QoS parameters of individual device connections	The requirements of various operations in a factory can differ from each other, e.g., the control of an AGV requires low latency communications, whereas sending video streams over the 5G link would require higher bandwidth but may not be so dependent on low latency. When creating two separate device groups, a user can set different QoS values for each individual device or for devices in the same group.
Getting location information of devices	When a moving object is detected by sensors or cameras, the current location information for that connected asset, as well as the geographical coordinates of relevant sensors and cameras, could be provided by the 5G device management API to relevant services operating in the factory.

5.5.4 The demonstration of the 5G Device Management API with an IoT-NGIN smart industry use case

This subchapter describes the demonstration of the 5G Device Management API in relation with an IoT-NGIN smart industry use case, namely the IoT-NGIN Use Case 6 – "Human-centred safety in a self-aware indoor factory environment".

The API procedures used to perform the relevant API features, diagrams illustrating the output of these API procedures, and screenshots showing the end-to-end data transmission between devices and services hosted on edge cloud are presented in the deliverable. For the detailed descriptions of the features of the 5G Device Management API, see D2.2 [12]. As an example, Figure 69 demonstrates a replica of the GUI of the 5G Device Management API illustrating the resulting end-to-end connections between 5G devices and virtual machines on the edge cloud that can host the AI/ML services from IoT-NGIN technologies. The API and its related features were used to establish these end-to-end connections and to provide monitoring of the connectivity status of these connections. The figure shows the IP addresses of communication endpoints assigned by the 5G network to allow them to communicate.

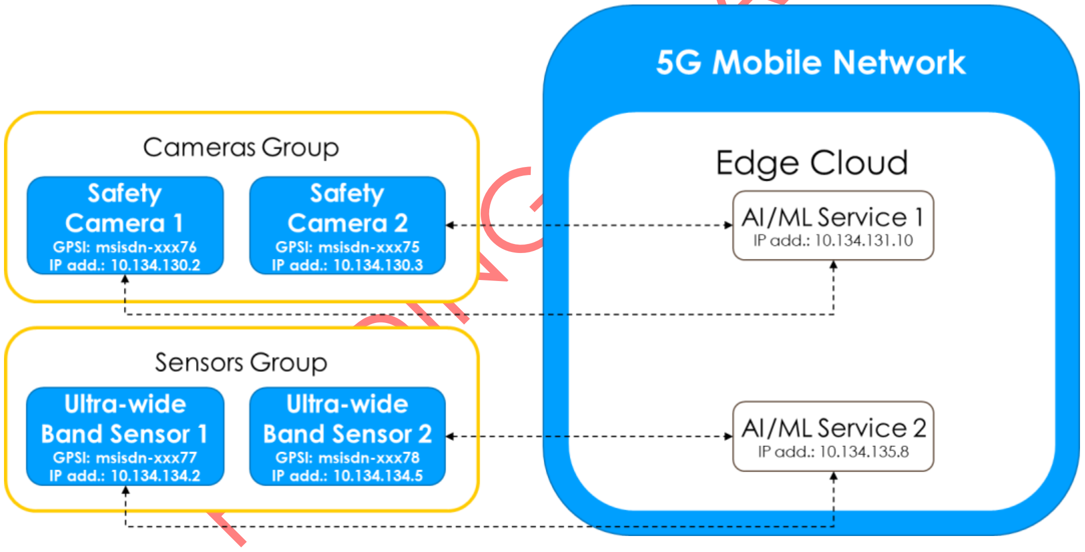


Figure 69 – A replica of the API GUI illustrating the end-to-end connections between devices and AI/ML services as representatives of the use case 6.

A set of pre-conditions had to be satisfied before we could start to use the 5G Device Management API or its GUI in the laboratory. These pre-conditions were that:

- The 5G Device Management API implementation is working properly in the laboratory,
- Raspberry Pis connected to 5G radio modems (called as "5G devices") are available in the laboratory and ready to send data, once they are attached to the 5G mobile network,
- Each 5G radio modem has a SIM card or eSIM available that allows them to be authenticated by the 5G mobile network, and that
- The user is authorised to access to the 5G mobile network, the 5G Device Management API and its GUI.

"Provisioning and onboarding devices" feature of the API:

To successfully be onboarded to the 5G mobile network, the subscription identity of each 5G device must be provisioned to the network. In the lab environment, multiple Raspberry Pis connected to 5G radio modems equipped with unique SIM information were used as 5G devices and onboarded to the network through the 5G Device Management API. The steps below were sequentially implemented:

- A unique device name was defined for each device, e.g., "Safety Camera 1", "Safety Camera 2".
- General Public Subscription Identifier (GPSI) numbers of devices were provided to the 5G Device Management API and effectively to the 5G core network. This enables the network to authenticate and then successfully onboard the intended device to the network.
- The GPSI numbers of each device were available in SIM cards of the 5G radio modems attached to each Raspberry Pis.
- Four different Raspberry Pis were onboarded separately in the lab environment. Each device was connected successfully to the network with default QoS parameters, and thus visible on the GUI.
- After onboarding performed successfully, each device was assigned with an IP address by the 5G core network.

The information of onboarded devices in the API GUI is given in Table 18.

Table 18 – 5G devices provisioned and onboarded to the 5G mobile network.

5G Device in the lab	5G Device Name	5G Device GPSI	IP address
"Raspberry Pi 1" connected to 5G radio modem	Safety Camera 1	msisdn-xxxx76	10.134.130.2
"Raspberry Pi 2" connected to 5G radio modem	Safety Camera 2	msisdn-xxxx75	10.134.130.3
"Raspberry Pi 3" connected to 5G radio modem	Ultra-wide Band Sensor 1	msisdn-xxxx77	10.134.134.2
"Raspberry Pi 4" connected to 5G radio modem	Ultra-wide Band Sensor 2	msisdn-xxxx78	10.134.134.5

"Getting a list of devices" feature of the API:

The user can easily access a list of onboarded devices using the 5G Device Management API. Once the feature of "Provisioning and onboarding devices" is successfully performed, the list of devices can be requested from the API which responses with the following output:

- Safety Camera 1
- Safety Camera 2
- Ultra-wide Band Sensor 1
- Ultra-wide Band Sensor 2

Detailed information of each device was given by the API as a list including the information on Table 18.

"Creating, modifying, and removing device groups" feature of the API:

The 5G Device Management API allows the user to create groups of 5G devices serving for specific purposes or having the same QoS requirements. To address the requirements of the IoT-NGIN Use Case 6, two different device groups were created in the laboratory using the 5G Device Management API. The steps below were sequentially implemented:

- Group names and the QoS parameters were defined by the user.
- The groups were named as "Cameras Group" and "Sensors Group".
- The QoS parameters relevant for the Use Case 6 were the priority level and the maximum bitrate (uplink and downlink). Therefore, these two parameters were set to the required values through the API.
- After both device groups were created, Security Cameras 1,2 and Ultra-wide Band Sensors 1,2 were added to the groups "Cameras" and "Sensors" respectively, and
- After a device becomes a member of a group, all the QoS parameters set for that group are automatically applied to the connectivity characteristics of the device.
- Table 19 shows the groups created, QoS parameters set for the groups and devices added to the groups as group members.

Table 19 – Device groups created through the 5G Device Management API.

Group Name	QoS Parameters		Group Members
	Priority Level	Maximum Bitrate	
Cameras Group	High	Default (10^4 kbps)	Security Camera 1 Security Camera 2
Sensors Group	Medium	Default (10^4 kbps)	Ultra-wide Band Sensor 1 Ultra-wide Band Sensor 2

"Monitoring the quality of the communication links of devices" feature of the API:

The 5G Device Management API enables the user to actively monitor the communication link of the devices such as monitoring of latency, packet loss, and bitrate. Among these parameters, latency monitoring was performed in the lab environment using the 5G Device Management API. The latency monitoring consists of round-trip time (RTT) measurement between two endpoints for a given duration of time.

The 5G Device Management API requires measurement duration and latency window for generating a plot. The following settings were used:

- The measurement duration was defined as 100 seconds meaning that the latency will be measured for the upcoming 100 seconds.
- The latency window was defined as 5 seconds meaning that the latency is measured every 5 seconds.
- As a result, the API measured the latency (RTT) every 5 seconds for 100 seconds and showed the results on a plot to the user.

"Defining and changing QoS parameters of individual device connections" feature of the API:

The user can set different QoS parameters per individual devices in a group through the 5G Device Management API. To validate this feature, the QoS parameters of the Security

Camera 1 from Cameras group was updated to have maximum bit rate of 2x10^6 kbps instead of the default value of 10^4 kbps. This process was completed by the 5G Device Management API, by sharing the device GPSI to the 5G core network to perform the required actions to change the intended parameter.

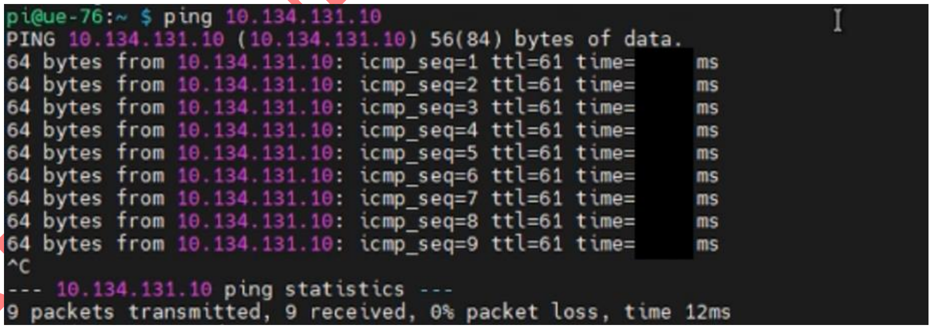
"End-to-end data transmission tests":

The grouping feature of the 5G Device Management API separated devices into two groups and allowed them to only communicate between group members but not with any other devices. To check the functionality of the API that has performed necessary steps to establish communication links between group members, ping tests were performed as follows:

- Ping messages using the Internet Control Message Protocol (ICMP) were transmitted from Raspberry Pis to the virtual machines which were hosted on the edge cloud in the lab environment.
- The groups were configured to share the data from the devices with two different virtual machines hosted on edge cloud acting as two different AI/ML services from IoT-NGIN use case 6. As devices on one group should only send their data to one specific virtual machine as a feature of device grouping, it was tested and validated that the other virtual machine cannot receive this data.
- The connections between devices and virtual machines were checked whether the ping messages were successfully received on the receiver side, and the echo reply message was received on the transmitter side.

The first set of tests was conducted for the devices added to the Cameras group. A cluster of ping messages were transmitted from a Raspberry Pi to each virtual machine acting as a AI/ML service hosted on edge cloud.

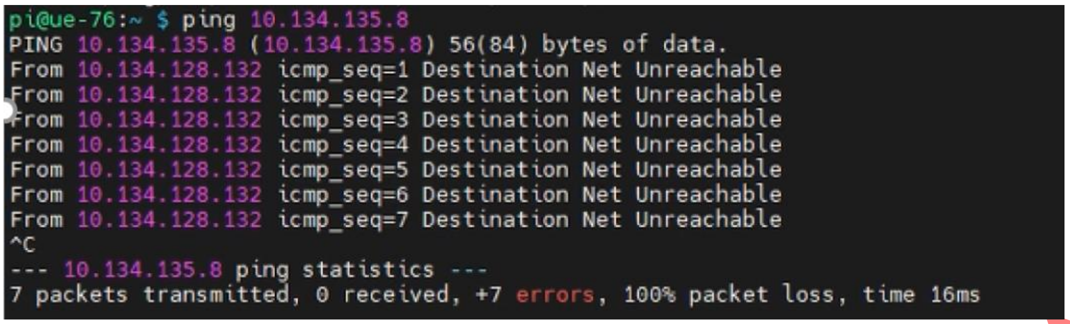
- As seen in Figure 70, from Security Camera 1 to the Service 1, cluster of ping messages were transmitted. Messages received successfully which indicates the existence of the connection between two endpoints.



```
pi@ue-76:~ $ ping 10.134.131.10
PING 10.134.131.10 (10.134.131.10) 56(84) bytes of data.
64 bytes from 10.134.131.10: icmp_seq=1 ttl=61 time=[redacted] ms
64 bytes from 10.134.131.10: icmp_seq=2 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=3 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=4 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=5 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=6 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=7 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=8 ttl=61 time=       ms
64 bytes from 10.134.131.10: icmp_seq=9 ttl=61 time=       ms
^C
--- 10.134.131.10 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 12ms
```

Figure 70 – End-to-end data traffic between Security Camera 1 and Service 1.

- As seen in Figure 71, from Security Camera 1 to the Service 2, cluster of ping messages were transmitted. Messages could not be received successfully by the receiver and failed, which was the desired output. This indicates that the connection between these two endpoints does not exist, and the Raspberry Pi in Cameras group can only share its data with the Service 1 that it is connected to.

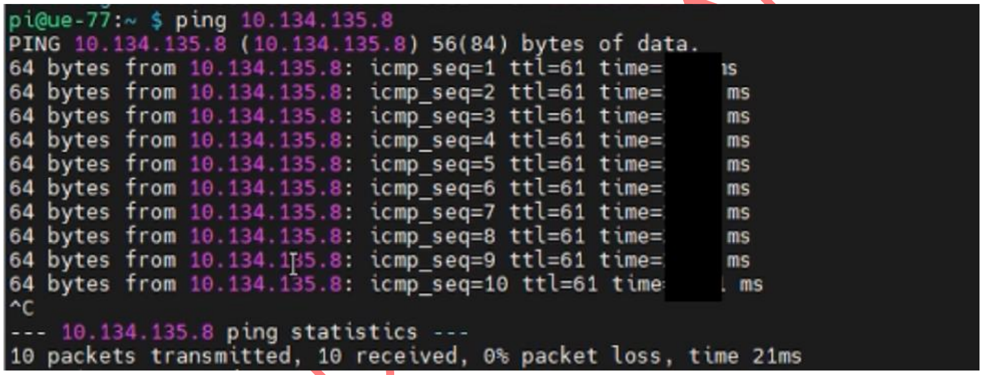


```
pi@ue-76:~ $ ping 10.134.135.8
PING 10.134.135.8 (10.134.135.8) 56(84) bytes of data.
From 10.134.128.132 icmp_seq=1 Destination Net Unreachable
From 10.134.128.132 icmp_seq=2 Destination Net Unreachable
From 10.134.128.132 icmp_seq=3 Destination Net Unreachable
From 10.134.128.132 icmp_seq=4 Destination Net Unreachable
From 10.134.128.132 icmp_seq=5 Destination Net Unreachable
From 10.134.128.132 icmp_seq=6 Destination Net Unreachable
From 10.134.128.132 icmp_seq=7 Destination Net Unreachable
^C
--- 10.134.135.8 ping statistics ---
7 packets transmitted, 0 received, +7 errors, 100% packet loss, time 16ms
```

Figure 71 – End-to-end data traffic between Security Camera 1 and Service 2.

The second set of tests was conducted for the devices added to the Sensors group. A cluster of ping messages were transmitted from a Raspberry Pi to each service hosted on edge cloud.

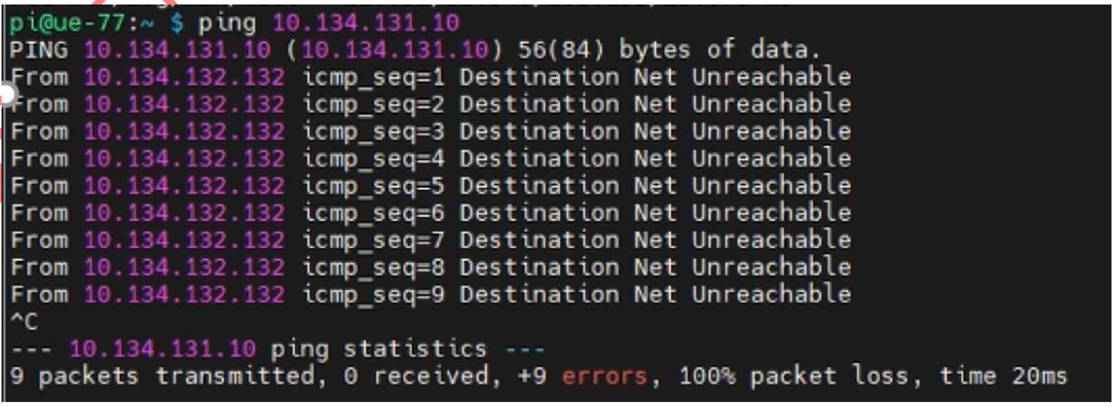
- As seen in Figure 72, from Ultra-wide Band Sensor 1 to the Service 2, cluster of ping messages were transmitted. Messages received successfully which indicates the existence of the connection between two endpoints.



```
pi@ue-77:~ $ ping 10.134.135.8
PING 10.134.135.8 (10.134.135.8) 56(84) bytes of data.
64 bytes from 10.134.135.8: icmp_seq=1 ttl=61 time=[redacted] ms
64 bytes from 10.134.135.8: icmp_seq=2 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=3 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=4 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=5 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=6 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=7 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=8 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=9 ttl=61 time=    ms
64 bytes from 10.134.135.8: icmp_seq=10 ttl=61 time    . ms
^C
--- 10.134.135.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 21ms
```

Figure 72 – End-to-end data traffic between Ultra-wide Band Sensor 1 and Service 2.

- As seen in Figure 73, from Security Camera 1 to the Service 1, cluster of ping messages were transmitted. Messages could not be received by the receiver and the ping test was failed, which was the desired output. This indicates that the connection between these two endpoints does not exist, and the Raspberry Pi in the Sensors group can only share its data with the Service 2 that it is connected to.



```
pi@ue-77:~ $ ping 10.134.131.10
PING 10.134.131.10 (10.134.131.10) 56(84) bytes of data.
From 10.134.132.132 icmp_seq=1 Destination Net Unreachable
From 10.134.132.132 icmp_seq=2 Destination Net Unreachable
From 10.134.132.132 icmp_seq=3 Destination Net Unreachable
From 10.134.132.132 icmp_seq=4 Destination Net Unreachable
From 10.134.132.132 icmp_seq=5 Destination Net Unreachable
From 10.134.132.132 icmp_seq=6 Destination Net Unreachable
From 10.134.132.132 icmp_seq=7 Destination Net Unreachable
From 10.134.132.132 icmp_seq=8 Destination Net Unreachable
From 10.134.132.132 icmp_seq=9 Destination Net Unreachable
^C
--- 10.134.131.10 ping statistics ---
9 packets transmitted, 0 received, +9 errors, 100% packet loss, time 20ms
```

Figure 73 – End-to-end data traffic between Ultra-wide Band Sensor 1 and Service 1.

5.5.5 Conclusion

The objectives of tests conducted with the 5G Device Management API were to:

- realise a 5G-enabled advanced use case from the use cases studied in IoT-NGIN in a laboratory environment,
- demonstrate the 5G Device Management API in use,
- describe potential benefits of the API on different vertical domains, and to
- demonstrate the API features through its GUI with a 5G-enabled advanced use case defined for the smart industry domain.

The results of the tests with the 5G Device Management API proof-of-concept validated that the API, its features, and its GUI, were working correctly. They showed that the ping tests were successfully run on the 5G device and on the 5G edge cloud. With these simple ping tests, we could show that operations, such as onboarding devices to the 5G network, creating isolated device groups, and monitoring the quality of the communication links of the devices, were simplified by using the API GUI.

Using the 5G Device Management API, it is possible to offer the ability to manage and monitor the 5G connectivity of devices to users or developers from any vertical domain, especially to those domains studied in IoT-NGIN. Furthermore, the API makes it possible to integrate more devices to mobile networks, automate their operations, and adopt 5G into private infrastructures in a faster and simpler manner. For instance, a worker from an industrial factory gains the capability to onboard more of their 5G-capable safety cameras and ultra-wide band sensors to a private 5G network serving the factory floor, to group their devices according to their location into various device groups, and to monitor the quality of the communication links of the 5G devices used in the factory.

6 Conclusions

In this deliverable the main activities of the IoT-NGIN project towards integration, instantiation in the Living Labs, as well as validation activities, focussed on 5G enhancements, have been presented.

Specifically, the integration of the IoT-NGIN framework based on the principle of cloud-native computing and through DevSecOps has been presented. GitLab CI/CD and Kubernetes are the main tools to automate the integration, build, testing and deployment processes to the extent possible and realise a cloud-native IoT-NGIN framework.

Moreover, the component-level architecture has been updated, following the realm of the work across the development teams of the project, while considering support of the LL use cases. The integration of IoT, edge and cloud, as well as the integration of ML services through the MLaaS platform have been also discussed. The current state of the IoT-NGIN framework configuration and deployment has been presented, based on the Helm charts for IoT-NGIN components and custom automation scripts.

Next, the usability of the IoT-NGIN framework is proven through use case specific instantiations of the architecture in the context of the 10 use cases across the 4 Living Labs of the project. Business-specific functionality is also presented, to leverage and complement the IoT-NGIN functionality in the relevant use cases.

On the validation side, results of validation tests of the 5G enhancements have been presented. These include latency performance tests of live 5G in a laboratory environment, as well as tests investigating an advanced 5G feature prototype for prioritizing and pre-scheduling uplink data and potentially improving network performance for IoT-NGIN use case data streams. In addition, the performance of a live 5G in a laboratory environment supporting the communication required by the edgePMU hardware is validated. Last, but not least, TSN tests on CMC laboratory based on 5G Core connected to Analog Devices off-the-shelf TSN switches to validate they get in sync.

Moreover, IoT-NGIN components, also in the context of specific use cases, have been tested, including the D2D communication enhancements and the new version of the edgePMU, featuring 5G connectivity. The 5G Device Management API has been tested against the use cases, showcasing its value in the Smart Agriculture and Smart Industry domains. Moreover, business specific functionality tests have been successfully conducted and presented.

The validation results will feed enhancements in the development of the IoT-NGIN components for the next integration cycle.

The future work includes integration towards the final version of the IoT-NGIN platform, as well as laboratory testing and evaluation results of the integrated prototype comprised by the complete spectrum of the project's developed technical solutions. The work will be documented in D6.3 "Interoperable IoT-NGIN meta-architecture & laboratory evaluation", due in the second quarter of 2023.

The IoT-NGIN framework is available as open source on the project's GitLab group [9].

7 References

[1] IoT-NGIN, "D1.2 - IoT meta-architecture, components, and benchmarking," H2020 957246 - IoT-NGIN Deliverable Report, 2021.

[2] IoT-NGIN, "D1.3 - IoT meta-architecture alignment and continuous technology watch," H2020 957246 - IoT-NGIN Deliverable Report, 2022.

[3] IoT-NGIN, "D6.1 - Integration guidelines & initial IoT-NGIN components integration," H2020 - 957246 - IoT-NGIN Deliverable Report, 2022.

[4] CNCF, "Cloud Native Computing Foundation ("CNCF") Charter," 2021. [Online]. Available: https://github.com/cncf/foundation/blob/main/charter.md.

[5] HELM, "Helm - The package manager for Kubernetes," [Online]. Available: https://helm.sh. [Accessed 2022].

[6] Kubernetes, "Kubernetes Documentation - Deployments," [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/. [Accessed 2022].

[7] Kubernetes, "Kubernetes Documentation - Secrets," [Online]. Available: https://kubernetes.io/docs/concepts/configuration/secret/. [Accessed 2022].

[8] Kubernetes, "Kubernetes Documentation - ConfigMaps," [Online]. Available: https://kubernetes.io/docs/concepts/configuration/configmap/. [Accessed 2022].

[9] IoT-NGIN, "IoT-NGIN GitLab Group," 2022. [Online]. Available: https://gitlab.com/h2020-iot-ngin.

[10] Kubeflow, "The Machine Learning Toolkit for Kubernetes," [Online]. Available: https://www.kubeflow.org/. [Accessed 2022].

[11] IoT-NGIN, "IoT-NGIN Dockerhub profile," DockerHub, [Online]. Available: https://hub.docker.com/repository/docker/iotngin.

[12] IoT-NGIN, "D2.2 - Enhancing IoT Underlying Technology," H2020-957246 IoT-NGIN Deliverable Report, 2022.

[13] IoT-NGIN, "D4.3 - Enhancing IoT Tactile & Contextual Sensing/Actuating," H2020 - 957246 - IoT-NGIN Deliverable Report, 2022.

[14] IoT-NGIN, "D3.3 - Enhanced IoT federated deep learning/ reinforcement ML," H2020 - 957246 - IoT-NGIN Deliverable Report, 2022.

[15] IoT-NGIN, "D5.4 - Enhancing IoT Data Privacy & Trust (Update)," H2020-957246 IoT-NGIN Deliverable Report, 2022.

[16] AIOTI, “Guidance for the Integration of IoT and Edge Computing in Data Spaces,” 2022.

[17] Telefonica, “Orion Context Broker,” GitHub, 2022. [Online]. Available: https://github.com/telefonicaid/fiware-orion/blob/master/README.md.

[18] IoT-NGIN, “IoT Device Indexing,” GitLab, 2022. [Online]. Available: https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing.

[19] D. Shyy, C. Watson and K. Woods, “6G and Artificial Intelligence & Machine Learning,” MITRE, 2021. [Online]. Available: https://www.mitre.org/news-insights/publication/6g-and-artificial-intelligence-machine-learning.

[20] IoT-NGIN, “D3.1 - Enhancing deep learning / reinforcement learning,” H2020 - 957246 - IoT-NGIN Deliverable Report, 2021.

[21] IoT-NGIN, “D3.2 - Enhancing Confidentiality preserving federated ML,” H2020 - 957246 - IoT-NGIN Deliverable Report, 2021.

[22] IoT-NGIN, “D7.2 – Trial site set-up, initial results and DMP update,” H2020 - 957246 - IoT-NGIN Deliverable Report, 2022.

[23] Synelixis, “SynField,” Synelixis, 2022. [Online]. Available: https://www.synfield.gr/.

[24] React, “React - A JavaScript library for building user interfaces,” [Online]. Available: https://reactjs.org. [Accessed 2022].

[25] L. Ouadi, E. Bruez, S. Bastien, J. Vallance, P. Lecomte, J.-C. Domec and P. Rey , “Ecophysiological impacts of Esca, a devastating grapevine trunk disease, on Vitis vinifera L.,” *PLOS ONE,* vol. 14, no. 9, 2019.

[26] M. Alessandrini, R. Rivera, C. Fuentes, L. Falaschetti, D. Pau, V. Tomaselli and C. Turchetti, “A grapevine leaves dataset for early detection and classification of esca disease in vineyards through machine learning,” *Data in Brief,* p. 106809, 2021.

[27] ROS, “Robot Operating System,” [Online]. Available: https://www.ros.org. [Accessed 2022].

[28] SOFIE, “SOFIE project website,” H2020 - 779984, [Online]. Available: https://www.sofie-iot.eu/. [Accessed 2022].

[29] “tcpdump,” [Online]. Available: https://www.tcpdump.org/.

[30] “Network Time Protocol,” [Online]. Available: http://www.ntp.org/.

[31] “http,” python, [Online]. Available: https://docs.python.org/3/library/http.html. [Accessed 22 February 2022].

[32] “Unified Architecture,” OPC Foundation, [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/. [Accessed 22 Februrary 2022].

[33] A. M. I. Fette, "RFC 6455," December 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6455. [Accessed 27 June 2022].

[34] A. R. R. L. H. Schulzrinne, "RFC 2326," 1998. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2326. [Accessed 27 June 2022].

[35] IEEE/IEC, "IEEE/IEC International Standard - Measuring relays and protection equipment - Part 118-1: Synchrophasor for power systems - Measurements," *IEC/IEEE 60255-118-1:2018,* 2018.

[36] AIM & THURLBY THANDAR INSTRUMENTS, "TG1000 & TG2000 Datasheet," [Online]. Available: https://resources.aimtti.com/datasheets/AIM-TG1000+TG2000_function_generators_data_sheet-Iss2A.pdf. [Accessed 2022].

[37] RTDS Technologies Inc., "RTDS website," [Online]. Available: https://www.rtds.com/. [Accessed 2022].

[38] "3GPP TS 23.434; Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows; Release 18," 2022.

[39] IoT-NGIN, "D1.1 - Definition analysis of use cases and GDPR Compliance," H2020-957246 IoT-NGIN Deliverable Report, 2021.

[40] "WebSocket," javascript.info, 26 June 2022. [Online]. Available: https://javascript.info/websocket. [Accessed 27 June 2022].

[41] IoT-NGIN, "D5.3 - Enhancing IoT Data Privacy & Trust," H2020-957246 IoT-NGIN Deliverable Report, 2021.

[42] "OPC Unified Architecture (UA)," 2008. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[43] "IEC 61784-2: Real-time Ethernet".

Annex 1 edgePMU parameters

The relation between the different edgePMU parameters is explained in the next paragraphs.

The packet size is influenced by the number of channels and vectorize. It holds the following relation:

$$Packet\_size = (20 + (number\_of\_channels - 1) * 4) * vectorize$$

For a vectorize of 10 and different numbers of channels, it leads to Table 20.

Table 20 – edgePMU packet size.

Number of channels	1	2	3	4	5	6	7	8
Packet size [bytes]	200	240	280	320	360	400	440	480

Important to mention is that packet fragmentation occurs if the packet size is larger than 1472 bytes, i.e., that the original packet is divided into smaller packets so that the channel capacity is used more efficiently. The packet limit is determined by the normal Ethernet MTU (Maximum Transmission Unit) which is 1500 bytes but without the IP header (20 bytes) and without the UDP header (8 bytes).

The message period is computed by

$$Message\_period = vectorize / sample\_rate$$

The throughput is influenced by the number of channels and the sampling rate. It follows the relation:

$$Throughput = 8 * (20 + (number\_of\_channels - 1) * 4) * sampling\_rate$$

The throughput has a factor 8 so that it is calculated in bits per second. To give an example, Table 21 presents the throughput for a fixed sampling rate of 50kHz.

Table 21 – edgePMU throughput.

Number of channels	1	2	3	4	5	6	7	8
Throughput [Mbps]	8	9.6	11.2	12.8	14.4	16	17.6	19.2

An overview of all traffic patterns is given in Table 22.

Table 22 – edgePMU traffic patterns.

Number of channels	Sampling rate [samples/s]	Vectorize [samples/packet]	Packet size [bytes]	Throughput [Mbps]	Message period [ms]
1 – 8	20k – 50k	10 – 200	200 – 9,600	3.2 – 19.2	0.2 – 10

Annex 2 IoT-NGIN Integration RoadMap

Table 23 – IoT-NGIN components details and interfaces.

IoT-NGIN components details			**Interfaces**		
Component Name	**WP/ Task**	**DESCRIPTION**	**INPUT** (from comp. x, y,z)	**OUTPUT** (to comp. x, y,z)	**Other Dependencies**
Secure Edge Cloud framework (for IoT micro-services)	WP2	An innovative "by design" framework using the most advanced programming language in security (Rust) and unikernels to support secure, deployable and scalable edge cloud execution framework for IoT focused micro-services.	MLaaS, Virtual Infrastructure Manager (VIM)	Virtual Infrastructure Manager (VIM), MLaaS	Apache TVM, TensorFlow, Pillow, Ninjga
5G Resources Management API	WP2	Open 5G Resources Management API to enable IoT devices to access the IoT-NGIN resources, providing connectivity, along with its native secure, tactile, low-latency and reliable connectivity facilities.	IoT devices (outside world), Slice & Orchestration Engine, Network Controller API, 5G device management API	IoT devices (outside world), Slice & Orchestration Engine, Network Controller API, 5G device management API	Microservices Management API, Network Controller API, 5G Device Management API

Slice & Orchestration Engine (Nework Controller API)	WP2	The 5GC includes network slice manager to separate devices and traffic with different requirements i.e. IoT traffic from consumer data. The 5GC includes features such as 5GLAN, TSN required specifically for private installation and integration with industrial infrastructure.	5G Resource Management API	5G Resource Management API (RAN controller, MANO, VIM)	Industrial 5G Core with network slice manager
5G device management API	WP2	This API aims to enable IoT devices to access the IoT-NGIN resources, providing connectivity, along with its native secure, tactile, low-latency and reliable connectivity facilities.	5G Resources Management API	IoT devices (outside world)	5G resource management API, 5G Core Network Functions
Device-to-Device communication (5G Coverage Extension)	WP2	D2D API which allows enhanced connectivity of IoT devices in 5G networks, implementing an advanced relay selection strategy, based on experimental evaluation of D2D links, as well as information sharing with the underlying infrastructure.	TSN Bridge, Network Controller API	5G Core Network Functions, IoT devices (outside world)	TSN Bridge
Privacy Preserving Federated Learning	WP3	The Privacy-preserving Federated Learning framework allows the development of more efficient ML models, which are trained considering data from multiple sources with increased privacy guarantees, without disclosing the data and protecting the information communicated, which could lead to data disclosure.	PPFL API	PPFL API	PPFL API

PPFL API	WP3	API that exposes Privacy Preserving FL interfaces, facilitating the communication with IoT cybersecurity and data privacy components (MAD and GAN-based attack dataset generator).	GAN-based attack dataset generator, Privacy preserving federated ML	MAD, Privacy preserving federated ML, MLaaS	MLaaS, Privacy preserving federated ML
MLaaS	WP3	MLaaS platform offering AI scientists and developers Data Management, ML training, Model Sharing and Prediction as a service, offering APIs and/or SDK to train and optimize AI models, supporting self-learning capabilities	Reinforcement Learning framework, Online Learning framework, Malicious Attack Detector (MAD). PPFL API,	Reinforcement Learning framework, Online Learning framework, Polyglot model sharing, IoT Device Discovery, Secure edge cloud execution framework	kubeflow, Kserve, Kafka, Camel-k, MINIO, Flower, Mlflow, Mosquito MQTT, Nginx, ONNX, Rust
Online Learning Framework	WP3	Online Learning (OL) service is responsible for i) the dynamic training of ML models for IoT applications, such as those implemented for the different Living Lab apps, as data become available, and also ii) for providing inferences (i.e., predictions), from these models, when requested.	MLaaS	MLaaS	MLaaS

Reinforcement Learning Framework	WP3	Depending on the algorithm of choice, the service will handle the lifecycle of the necessary components for the algorithm development and implementation.	MLaaS	MLaaS	MLaaS
Polyglot Model Sharing	WP3	Polyglot model sharing facilitates the communication of the ML models so that ML engineers, data scientists or AI developers who can use them as-is or for transfer learning. MLaaS uses a Polyglot Model Sharing	MLaaS, IoT devices (outside world)	IoT devices (outside world)	ONNX is used as library
Device Indexing	WP4	The IoT Device Indexing component enables the creation of a repository of IoT devices, allowing quickly querying about their status, basic characteristics and associated monitoring or other regularly updated information, thus indexing both the physical and digital twin of IoT devices.	IoT Devices access control, IoT/ML/AR service, IoT Vulnerability crawler, MTD network of honeypots	IoT Devices access control, IoT/ML/AR service, IoT Device Discovery, IoT Vulnerability crawler, MTD network of honeypots, IoT devices (outside world)	FIWARE Orion Context Broker, IoT agent, Historic Data Registry
IoT Devices Access Control	WP4	Security middleware between IoT devices and services, enabling pervasive security based IoT access control, extending static access rights by managing user rights by physical proximity or visibility as a flexible security and policy management API gateway	SSI, MTD network of Honeypots, IoT VC, IoT Device Indexing, IoT/ML/AR serivces, IoT Device Discovery	MTD network of Honeypots, IoT VC, IoT Device Indexing, IoT/ML/AR serivces	Keycloak

IoT Device Discovery (Computer-vision)	WP4	Discovery method based on computer vision implementing a function for object recognition.	IoT/ML/AR Service, IoT Device Access control, MLaaS, IoT Device Indexing	IoT/ML/AR Service and IoT Device Access control	
IoT Device Discovery (Computer-vision)	WP4	Discovery method based on computer vision implementing a function for face recognition.	IoT/ML/AR Service, IoT Device Access control, MLaaS, IoT Device Indexing	IoT/ML/AR Service and IoT Device Access control	
IoT Device Discovery (Visual Light Positioning - VLP)	WP4	Positioning based on frequency identification and clustering	IoT/ML/AR Service, IoT Device Access control, MLaaS, IoT Device Indexing	IoT/ML/AR Service and IoT Device Access control	
IoT Device Discovery (Ultra Wide Band - UWB)	WP4	Ultra-Wide Band positioning which is non-visual but based on radio signals based on positioning algorithm in NLOS (Non Line-Of-Sight) situations or noisy scenarios.	IoT/ML/AR Service, IoT Device Access control, MLaaS, IoT Device Indexing	IoT/ML/AR Service and IoT Device Access control	
IoT Device Discovery (Code Scanning)	WP4	The component is provided to cover the QR scanning needs of Uc	IoT/ML/AR Service, IoT Device Access control, MLaaS, IoT Device Indexing	IoT/ML/AR Service and IoT Device Access control	

GAN-based Data Generator	WP5	GAN-based dataset generator is used to create data poisoning attacks in FL.	N/A	MAD, PPFL API	Malicious Attack Detector (MAD)
Decentralized Interledger Bridge	WP5	Interlinking different types of distributed ledgers with atomic transactions enables new types of services and helps overcome the limitations of individual ledgers.	N/A	SSI, Semantic Twin, Polyglot model sharing	
Privacy Preserving Self-Sovereign Identities (SSI)	WP5	Proof-of-concept prototypes demonstrate how self-sovereign identities can be used to increase trust in IoT-applications while still effectively protecting the privacy of different parties.	DIB	IoT Devices Access control, Semantic Twin, Digital Twin	
Malicious Attack Detector (MAD)	WP5	Malicious Attack Detection against model and data poisoning attacks in Federated Learning networks. The approach employs ML techniques via Generative Adversarial Networks to learn and detect poisoning attacks in malicious nodes.	GAN-based attack dataset generator, PPFL API	MLaaS	
IoT Vulnerability Crawler	WP5	IoT Vulnerability Crawler able to identify vulnerable IoT nodes from network security perspective, leveraging on awareness of known vulnerabilities appearing in relevant well-known open repositories.	IoT Devices Access control, IoT Device Indexing	MTD network of Honeypots	

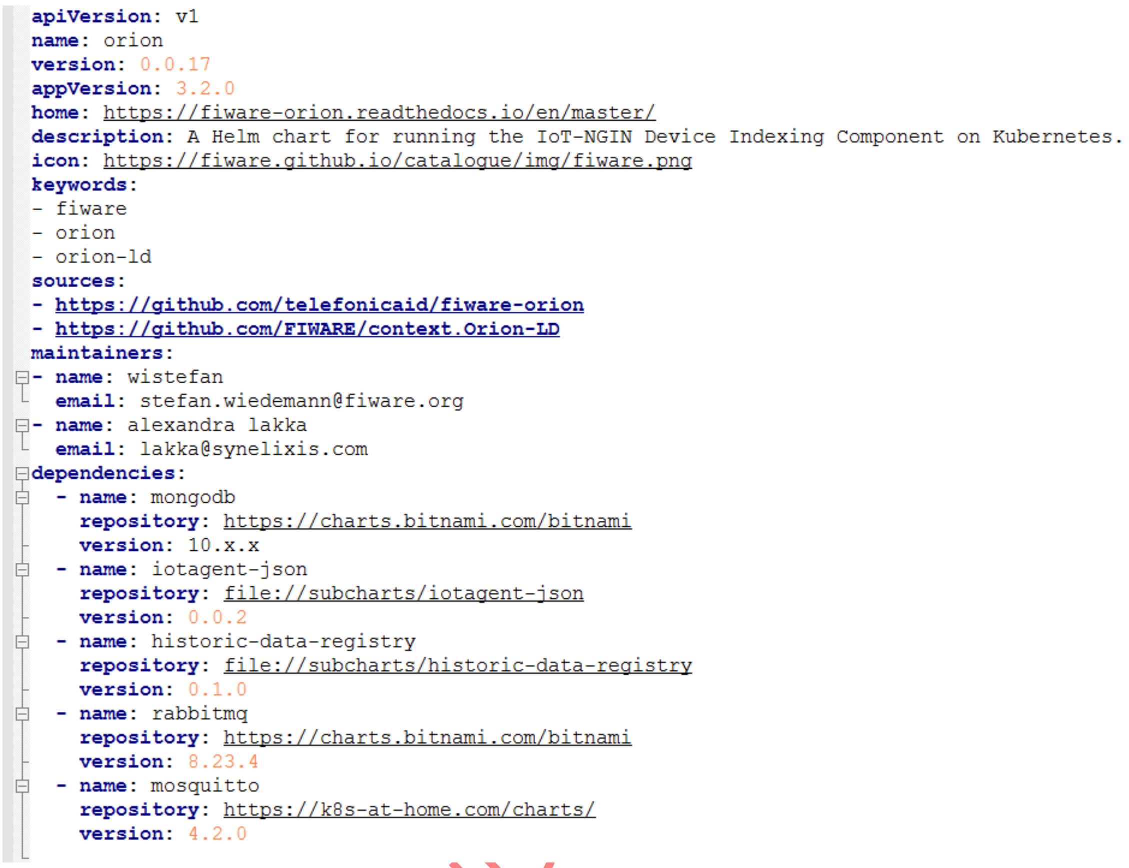
Semantic Twins	WP5	Semantic twin provides a semantic description of the digital twins and the related real-world entities, e.g., API endpoints, identity, relations to other twins, etc., incorporating digital Self-Sovereign Identities and SAREF ontologies	DIB, SSI	Digital Twin, IoT devices (outside world)	
Moving Target Defense (MTD) Honeypot Framework	WP5	The Moving Target Defence (MTD) can be exploited as part of an integrated cybersecurity solution or as a stand-alone cyber-defence mechanism offered either as a product or as a service that also includes the IoT network analysis process for added value. The latter, enables the option for providing a domain specific service.	IoT Devices Access control, IoT Device Indexing, IoT Vulnerability Crawler	IoT Devices Access control, IoT Device Indexing	
Quorum Network	WP6	Auxiliary Ethereum-based blockchain service (private)	N/A	DIB	Auxiliary private blockchain network
Smart Cities ML model(s)	WP7	Vehicle recognition, Human recognition, Crowd management via social network	N/A	N/A	To be integrated and available via the MLaaS
Smart Industry ML model(s)	WP7		N/A	N/A	To be integrated and available via the MLaaS

Smart Agriculture ML model - Crop disease detection	T6.2	UC4. Crop disease detection based on drone images	N/A	N/A	To be integrated and available via the MLaaS
Smart Agri ML model -Object detection	T6.2	Object detection model	N/A	N/A	To be integrated and available via the MLaaS
Smart Agri application (UC5) - Collision Avoidance	T6.2	Collision avoidance	IoT Device Access Control, IoT Device Indexing	IoT Device Access Control, IoT Device Indexing	MLaaS
Smart Agri application (AGV management)	T6.2	• AGV configuration and management	IoT Device Access Control, IoT Device Indexing	IoT Device Access Control, IoT Device Indexing	
Smart Agri web-application (Precision agriculture)	T6.2	• Including AI service for crop disease prediction • Will provide crop disease predictions to the user • Will provide SynField data to the user • Will allow the user to send actuation commands to the SynField devices	IoT Device Discovery (computer vision) , IoT Device Discovery (QR), IoT Devices Access Control	IoT Device Discovery (computer vision), Device Discovery (QR), IoT Devices Access Control	

Smart Energy Web Application - DSO dashboard	T6.2	Distribution System Operator (DSO) Dashboard	MLaaS, IoT Devices Access Control, Smart Energy Marketplace API (DSO), Generation & Consumption Forecasting for DSO	IoT Devices Access Control	
Smart Energy Web Application - CPO dashboard	T6.2	Charging Point Operator (CPO) Dashboard	IoT Devices Access Control, EV flexibility forecast, Demand Response Marketplace API (CPO), Generation & Consumption Forecasting (DSO)	Demand Response Marketplace API (CPO), Generation & Consumption Forecasting (DSO), IoT Devices Access Control	
Smart Energy ML model DSO	T6.2	Energy production & consumption forecasting system for DSO	N/A	N/A	MLaaS

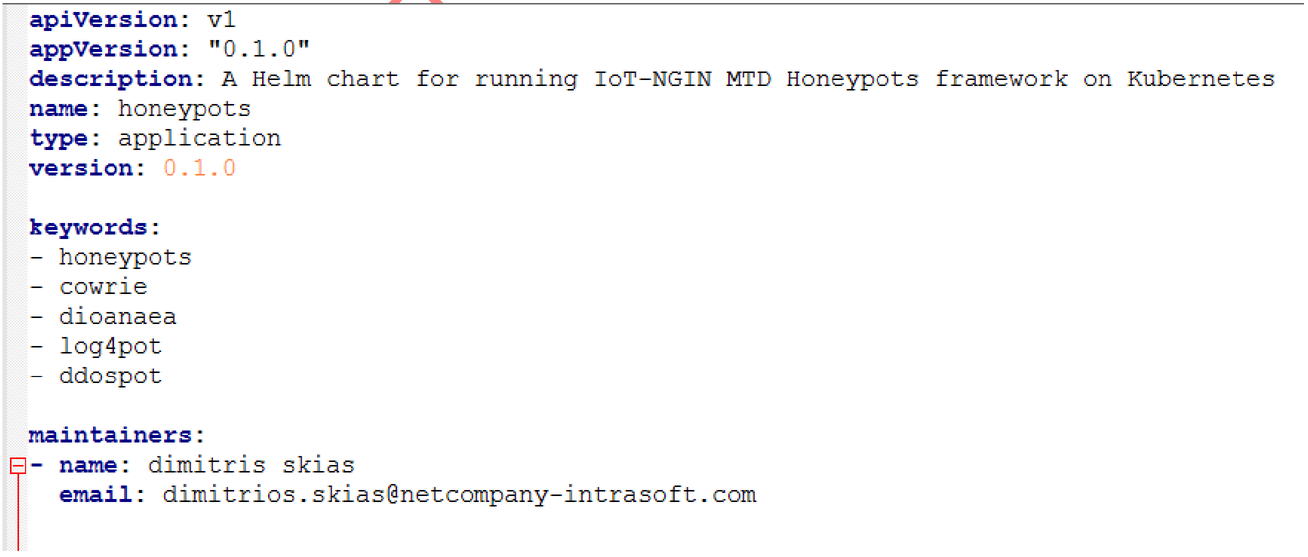
Smart Energy ML model (CPO)	T6.2	EV potential flexibility provision forecasting system for CPO	N/A	N/A	MLaaS
Smart Energy Marketplace API (DSO)	T6.2	Demand Response blockchain-based marketplace for DSO	DSO Dashboard (web app), DIB	DSO Dashboard (web app), DIB	
Smart Energy Marketplace API (CPO)	T6.2	Demand Response blockchain-based marketplace for CPO	DIB, CPO Dashboard (web app)	DIB	EMOTION server, CPO/EV users
Smart Energy AR Tool	T6.2	Device discovery & access control module for CPO	EV user app (mobile app)	IoT Device Access Control, IoT Device indexing	

Annex 3 IoT-NGIN components' HELM-charts



```yaml
apiVersion: v1
name: orion
version: 0.0.17
appVersion: 3.2.0
home: https://fiware-orion.readthedocs.io/en/master/
description: A Helm chart for running the IoT-NGIN Device Indexing Component on Kubernetes.
icon: https://fiware.github.io/catalogue/img/fiware.png
keywords:
- fiware
- orion
- orion-ld
sources:
- https://github.com/telefonicaid/fiware-orion
- https://github.com/FIWARE/context.Orion-LD
maintainers:
- name: wistefan
  email: stefan.wiedemann@fiware.org
- name: alexandra lakka
  email: lakka@synelixis.com
dependencies:
  - name: mongodb
    repository: https://charts.bitnami.com/bitnami
    version: 10.x.x
  - name: iotagent-json
    repository: file://subcharts/iotagent-json
    version: 0.0.2
  - name: historic-data-registry
    repository: file://subcharts/historic-data-registry
    version: 0.1.0
  - name: rabbitmq
    repository: https://charts.bitnami.com/bitnami
    version: 8.23.4
  - name: mosquitto
    repository: https://k8s-at-home.com/charts/
    version: 4.2.0
```

Figure 74 – Device Indexing Helm-chart.

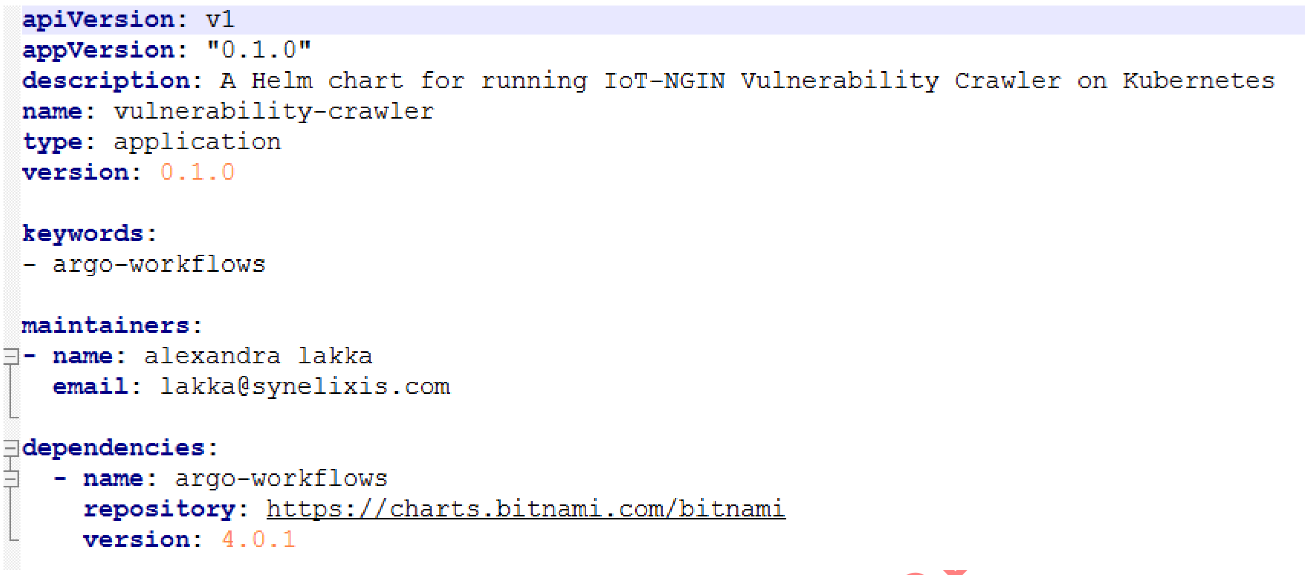


```yaml
apiVersion: v1
appVersion: "0.1.0"
description: A Helm chart for running IoT-NGIN MTD Honeypots framework on Kubernetes
name: honeypots
type: application
version: 0.1.0

keywords:
- honeypots
- cowrie
- dioanaea
- log4pot
- ddospot

maintainers:
- name: dimitris skias
  email: dimitrios.skias@netcompany-intrasoft.com
```

Figure 75 – MTD Honeypot framework Helm-chart.



```yaml
apiVersion: v1
appVersion: "0.1.0"
description: A Helm chart for running IoT-NGIN Vulnerability Crawler on Kubernetes
name: vulnerability-crawler
type: application
version: 0.1.0

keywords:
- argo-workflows

maintainers:
- name: alexandra lakka
  email: lakka@synelixis.com

dependencies:
  - name: argo-workflows
    repository: https://charts.bitnami.com/bitnami
    version: 4.0.1
```

Figure 76 – IoT Vulnerability Crawler Helm-chart.