



COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu

D3.6 – IoT
Data Pre-
processing
Module v2



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 955310

D3.6 – IoT Data Pre-processing Module v2

Dissemination Level:	Public
Deliverable Type:	Demonstrator
Lead Partner:	Hypertech
Contributing Partners:	UCL, AU, UEDIN, BOC-AG, QUE, NT, OLOD
Due date:	30-04-2022
Actual submission date:	29-04-2022

Authors

Name	Beneficiary	Email
Papafragkakis, Apostolos	Hypertech	a.papafragkakis@hypertech.gr
Giannakis, Giorgos	Hypertech	g.giannakis@hypertech.gr
Vassiliadis, Michalis	Hypertech	m.vassiliadis@hypertech.gr
Katsigarakis, Kyriakos	UCL	k.katsigarakis@ucl.ac.uk
Lilis, Georgios	UCL	g.lilis@ucl.ac.uk
Rovas, Dimitrios	UCL	d.rovas@ucl.ac.uk
Teizer, Jochen	AU	teizer@cae.au.dk
Chronopoulos, Christos	AU	chrichr@cae.au.dk
Bosché, Frédéric	UEDIN	f.bosche@ed.ac.uk
Bueno Esposito, Martín	UEDIN	martin.bueno@ed.ac.uk
Falcioni, Damiano	BOC-AG	damiano.falcioni@boc-eu.com
Andriopoulos, Panos	QUE	panos@que-tech.com
Triantafillou, Giannis	QUE	g.triantafillou@que-tech.com
Koletsas, Giorgos	QUE	g.koletsas@que-tech.com
Straka, Martin	NT	straka@novitechgroup.sk
Archontidis, Alexandros	OLOD	aarchontidis@olympiaodos.gr
Papakostas, Konstantinos	OLOD	kpapakostas@olympiaodos.gr

Reviewers

Name	Beneficiary	Email
Moraitis, Panagiotis	QUE	p.moraitis@que-tech.com
Falcioni, Damiano	BOC-AG	damiano.falcioni@boc-eu.com

Version History

Version	Editors	Date	Comment
0.1	A. Papafragkakis, G. Giannakis	11.03.2022	ToC
0.2	A. Papafragkakis, G. Giannakis	31.03.2022	Input from involved partners
0.3	A. Papafragkakis, G. Giannakis	04.04.2022	First Draft
0.4	A. Papafragkakis, G. Giannakis	21.04.2022	Final Draft for Internal Review
0.5	P. Moraitis, D. Falcioni	28.04.2022	Deliverable internal Review
0.6	A. Papafragkakis, G. Giannakis	29.04.2022	Review Comments Addressed
1.0	A. Papafragkakis, G. Giannakis	29.04.2022	Submission to the EC Portal

Disclaimer

©COGITO Consortium Partners. All right reserved. COGITO is a HORIZON2020 Project supported by the European Commission under Grant Agreement No. 958310. The document is proprietary of the COGITO consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights. The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use, which may be made, of the information contained therein.

Executive Summary

The present deliverable aims at documenting the second version of the IoT Data Pre-Processing Module and consequently at reporting on the second iteration of COGITO Tasks T3.3 “IoT solution in Construction Phase” and T3.4 “IoT data Pre-Processing module” design and development activities. In summary, the IoT Data Pre-processing Module is in charge of gathering raw IoT data coming from sensorial devices installed or worn at the construction site as well as pre-processing them and generating timeseries datasets that are ultimately fed to the COGITO Digital Twin platform.

During the early stages of the project, several meetings were held among the COGITO partners to elicit the input requirements of COGITO applications that shall act as the main IoT data consumers. In the context of T3.3, using the MoSCoW (Must-Should-Could-Would have) prioritisation method, COGITO partners specified the requirements of the IoT solution that is envisaged, covering the respective input data to the Work Order Definition and Management (WODM), the Process Modelling and Simulation (PMS) and the ProactiveSafety applications of the COGITO architecture. Hitherto, resources tracking (on-site workers and heavy machinery) has been defined as the single IoT data input to those applications. Based on an initial list of the IoT solution’s requirements (e.g., accuracy, latency, sensing range, total cost of acquisition and deployment in pilot sites, and security), a survey of Real-Time Location Systems (RTLS) techniques and relevant technologies was conducted, contrasting their capability in capturing resources’ location while meeting those requirements. The results of that study indicated that Bluetooth 5.1 with Angle of Arrival (AoA) is a promising technology for indoor positioning/pathfinding that meets the COGITO requirements. Hence, it was selected as candidate for initial experimentation in a lab environment. A survey was also performed on well-known and well-established vendors that realise commercial solutions for short range wireless indoor positioning using Bluetooth 5.1 with AoA. Among them, *Quuppa* was found to fulfil COGITO needs, offering a development kit with comprehensive training for experimentation. The kit includes locators and tags along with a controller for running the provided software suite for planning, deploying, and configuring the system in a dedicated environment. The development kit was purchased, installed and tested at Hypertech’s premises (leading Tasks T3.3 and T3.4). Despite the fact that *Quuppa* was selected for the experimental setup, it is worth mentioning that the COGITO’s IoT Data Pre-processing Module is being developed as a technology-agnostic middleware capable of fusing the various RTLS technologies and techniques currently available in a unified and consistent manner. Hence, it provides a potential for enhancing accuracy by using multiple positioning systems and circumvents vendor lock-in.

Given the abovementioned IoT solution, the first version of the IoT Data Pre-processing Module was prototyped, capable of ingesting raw IoT data coming from the IoT solution, processing them and feeding them to the COGITO Digital Twin platform as documented in D3.5 “IoT Data Pre-Processing Module v1”. Since then, the module has undergone additions and changes based on actual experimentation, testing, bug identification and feedback received by other COGITO partners through both bilateral discussions and mini workshops. Such updates, along with the objectives, the sub-components, the technology stack and the implementation tools, the inputs/outputs and the exposed APIs of the IoT Data Pre-Processing Module are documented in this deliverable.

Table of contents

Executive Summary	3
Table of contents	4
List of Figures	6
List of Tables	7
List of Acronyms	8
1 Introduction	10
1.1 Scope and Objectives of the Deliverable	10
1.2 Relation to other Tasks and Deliverables	11
1.3 Structure of the Deliverable	12
1.4 Updates to the first version of the IoT Data Pre-Processing Module	12
2 Overview of the IoT RTLS techniques, related technologies and available solutions	13
2.1 Positioning Technologies	13
2.1.1 RF-based	14
2.1.2 Non-RF based	16
2.2 Positioning topologies	16
2.2.1 Remote Positioning	16
2.2.2 Self-Positioning	17
2.2.3 Indirect Remote Positioning	17
2.2.4 Indirect Self-Positioning	17
2.3 Positioning techniques - algorithms	17
2.3.1 Geometric techniques	18
2.3.2 Proximity	19
2.3.3 Scene Analysis – Fingerprinting	19
3 IoT Data Pre-processing Module	21
3.1 Implementation Details	21
3.1.1 Communication subcomponents	21
3.1.2 Database subcomponent	22
3.1.3 Core application subcomponent	22
3.1.4 Module status monitoring & alerting subcomponent	22
3.2 Prototype Overview	23
3.2.1 Proprietary RTLS solution	23
3.2.2 Small-scale test deployment	26
3.3 Technology Stack and Implementation Tools	31
3.4 Input, Output and API Documentation	32
3.4.1 Contract-first API / ontology definition	32
3.4.2 Synchronous Data Communication (Restful API)	33
3.4.3 Asynchronous Data Communication (Apache Kafka)	34

3.5	Application Example	34
3.5.1	API response examples	34
3.5.2	Screenshots of the Module Status Monitoring & Alerting Dashboards.....	41
3.6	Licensing	44
3.7	Installation Instructions.....	44
3.8	Development and integration status	44
3.9	Requirements Coverage	44
3.9.1	Functional Requirements	45
3.9.2	Non-functional Requirements.....	45
3.10	Assumptions and Restrictions.....	46
4	Conclusions.....	47
	References.....	48

List of Figures

Figure 1: IoT solution requirements prioritised using the MoSCoW method	10
Figure 2: Summary of the indoor positioning technologies	10
Figure 3: Overview of potential topologies used for positioning	16
Figure 4: Overview of the most common RTLS techniques	17
Figure 5: Graphical representation of the angle-based (angulation) techniques	18
Figure 6: Graphical representation of the distance-based (lateration) techniques	18
Figure 7: The IoT Data Pre-processing Module implementation architecture (arrows define the direction of data flow)	21
Figure 8: Overview of the Quuppa system [14]	23
Figure 9: Then angle-based technique (AoA and AoD) [16]	23
Figure 10: The distribution of BLE, Wi-Fi and Quuppa proprietary channels in the ISM frequency band....	24
Figure 11: Comparison of Quuppa with other relevant technologies [14]	24
Figure 12: The received Quuppa development kit along with a separately purchased PoE switch	26
Figure 13: Using the provided planning tool (QSP) to create and plan a new project	27
Figure 14: Creation and configuration of simulation scenarios	27
Figure 15: Actual run of the simulation scenario	28
Figure 16: Snapshot of the generated floor map presenting the position of the simulated tags	28
Figure 17: Installation and cabling of the Quuppa locators above the false ceiling	29
Figure 18: “Identification” and “Focusing” of the locators using the Quuppa QSP	30
Figure 19: The Quuppa original and third-party tags purchased for testing	31
Figure 20: The Apicurio web application used to model the COGITO’s IoT API	32
Figure 21: Snapshot depicting part of the rendered YAML file while defining the API	33
Figure 22: Dashboard for the RTLS backend – status & project analytics	42
Figure 23: Dashboard for the Core application – streaming processing pipeline status	42
Figure 24: Dashboard for the API status	43
Figure 25: Dashboard for the UDP handler (input middleware) status	43
Figure 26: Dashboard for the Kafka broker status	44

List of Tables

Table 1: Relation to other COGITO project's deliverables	11
Table 2: Summary of the technology stack and implementation tools.....	31

List of Acronyms

Term	Description
2D	Two-dimension
3D	Three-dimension
AoA	Angle of Arrival
AP	Access Point
API	Application Programming Interface
BLE	Bluetooth Low Energy
BT	Bluetooth
CSV	Comma-Separated Values
DTP	Digital Twin Platform
FCC	Federal Communications Commission
FW	Firmware
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
Hz	Hertz
IC	Integrated Circuit
IMU	Inertial Measurement Unit
IR	Infrared
IR-UWB	Impulse Radio Ultra-Wideband
ISM	Industrial, Scientific and Medical frequency band
JSON	JavaScript Object Notation
kNN	k-Nearest-Neighbour
LOS	Line-of-Sight
mTLS	Mutual Transport Layer Security
NLOS	Non-Line-of-Sight
NN	Neural Network
PCB	Printed Circuit Board
PHY	Physical Layer
PoC	Proof of Concept
PoE	Power over Ethernet
QPE	Quuppa Positioning Engine
QSP	Quuppa Site Planner
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
RTTT	Round Trip Travel Time
SMP	Smallest M-vertex Polygon

SVM	Support Vector Machine
TDoA	Time Difference of Arrival
TLS	Transport Layer Security
ToA	Time of Arrival
ToF	Time of Flight
OTA	Over-The-Air
UUID	Universally Unique Identifier
UWB	Ultra-Wideband
VM	Virtual Machine
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Network


1 Introduction

1.1 Scope and Objectives of the Deliverable

The scope of this deliverable is to report on the work carried out within T3.3 and T3.4 towards designing, developing and delivering the second release of the COGITO's IoT Data Pre-processing Module along with the updates since the completion of its first version in M16 (documented in the deliverable D3.5). In alignment with the COGITO's Description of Action (DoA), the IoT Data Pre-processing module aims to filter raw data acquired by an IoT solution installed at the construction site thereby generating datasets that can be directly stored in the COGITO Digital Twin Platform.

No	Requirement	MoSCoW Priority	Comment
1	worker location tracking (coordinates, timestamp)	MUST HAVE	No preference between : Global coordinates / Relevant to a reference point
2	Heavy machinery location (coordinates, timestamp)	MUST HAVE	No preference between : Global coordinates / Relevant to a reference point
3	Accuracy of the measurement for requirements 1,2	SHOULD HAVE	Meter accuracy is enough
4	IMU data	COULD HAVE	for productivity relevant use-cases
5	material-component location tracking	COULD HAVE	
6	Tools location tracking	COULD HAVE	

No	Requirement	MoSCoW Priority	Comment
1	Worker location tracking (coordinates, timestamp)	MUST HAVE	No preference between : Global coordinates / Relevant to a reference point
2	Heavy machinery location (coordinates, timestamp)	MUST HAVE	No preference between : Global coordinates / Relevant to a reference point
3	Accuracy of the measurement for requirements 1,2	SHOULD HAVE	Meter accuracy (sub-meter if possible ~50cm)
4	Bearing (direction)	COULD HAVE	To clarify what this is
5	Bearing Accuracy	COULD HAVE	
6	Speed	COULD HAVE	
7	Speed Accuracy	COULD HAVE	


Workflow Management



Health & Safety

Figure 1: IoT solution requirements prioritised using the MoSCoW method

According to the results of the stakeholders' requirements elicitation and the first and second version of the COGITO system architecture, capturing information about the staff, machinery and other resources' location is fundamental to COGITO's Workflow Management and Health & Safety Digital Twin applications. Thus, the proposed IoT solution, outcome of T3.3 must attach importance to the selection of innovative and accurate localisation methods. In that sense, using the MoSCoW prioritisation method, COGITO partners have specified the requirements of the IoT solution to cover the needs of the Workflow Management and the Health & Safety applications. The complete list of the defined requirements is presented in Figure 1.

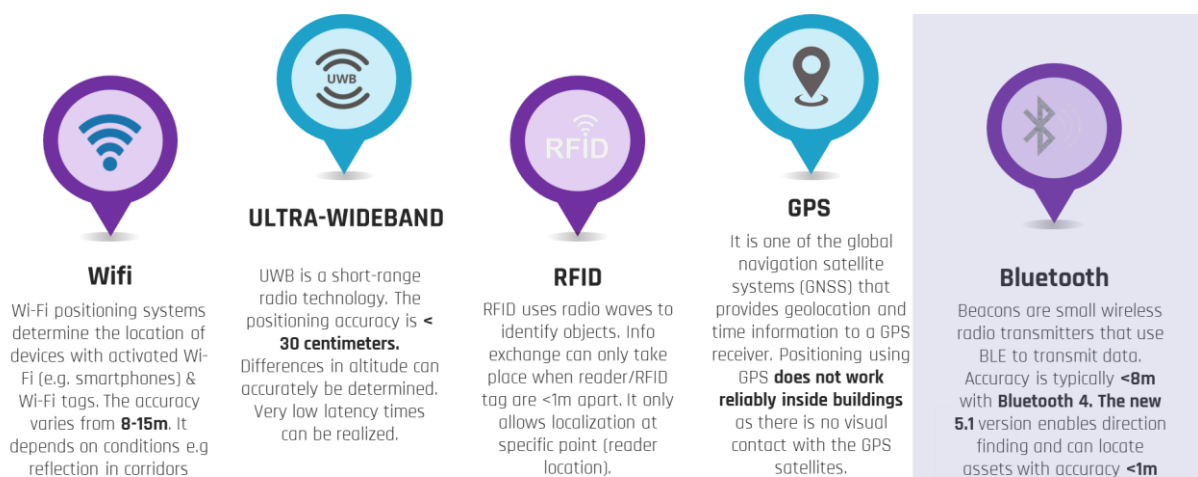


Figure 2: Summary of the indoor positioning technologies

Given this list of requirements, to deliver an appropriate IoT solution, initially, a thorough survey of a wide range of hardware IoT infrastructures, followed by an evaluation of available communication protocols, has been the main objective of T3.3. The results of that survey are presented in Section 2. Various indoor

positioning technologies (Figure 2) were reviewed; their characteristics were collected and evaluated, concluding to a proprietary solution that is proposed and introduced in Section 3.2.

Following the proposed IoT solution, the first and the second version of the IoT Data Pre-processing Module was designed, developed and delivered, constituting the outcomes of T3.4. It comprises of the Communication, Database, Core and Status Monitoring application sub-components presented in Section 3.1.

Since this work reflects the activities undertaken for the initial release of the module, it adheres to certain assumptions/restrictions and might not fully implement all the fine-grained functionalities that shall be offered in the IoT Data Pre-processing Module's final version (e.g., a full integration with the DTP is anticipated in its final version, to be delivered within T8.1 "End-to-end ICT System Integration, Testing and Refinement", as part of the COGITO components' refinement activities).

1.2 Relation to other Tasks and Deliverables

Table 1 depicts the relations of this document to other deliverables within the COGITO project; the latter should be considered along with this document for further understanding of its contents.

Table 1: Relation to other COGITO project's deliverables

Del. Number	Deliverable Title	Relations and Contribution
D2.1	Stakeholder requirements for the COGITO system	Analysis of the end-user requirements in order to create the necessary inputs for defining the COGITO IoT Data Pre-Processing Module and its interactions with other components, along with a thorough description of the business scenarios, use cases and system requirements tailored to the project's goals and therefore setting the skeleton for the IoT Data Pre-Processing Module development.
D2.3	COGITO Evaluation Methodology	Development of the evaluation methodology that determines the means (Key Performance Indicators) that will be used to evaluate and assess the COGITO outputs during the pre-validation and validation phases of the project, i.e., quantifying their impact on (i) construction's time, productivity, cost, safety, and sustainability and (ii) accelerating the digitisation rate of the construction sector.
D2.4	COGITO System Architecture v1	The intermediate version of the COGITO architecture report that includes the specifications of the IoT Data Pre-Processing Module concerning its hardware and software requirements, the programming language(s), and the status of its development, its functional and non-functional requirements, and its inputs/outputs, along with the format, method, endpoint and protocol for each data type and interface.
D2.5	COGITO System Architecture v2	The second version of the COGITO architecture report that includes updates on the specifications of the IoT Data Pre-Processing Module (hardware/software requirements, programming language(s), development status, functional/non-functional requirements, inputs/outputs, along with the format, method, endpoint and protocol for each data type and interface).

D3.2	COGITO Data Model & Ontology Definition and Interoperability Design v1	Documentation of the first version of the COGITO ontologies (available online in the COGITO ontology portal), particularly the resources relevant to the IoT Data Pre-Processing module.
D3.3	COGITO Data Model & Ontology Definition and Interoperability Design v2	Documentation of the second version of the COGITO ontologies (available online in the COGITO ontology portal), particularly the resources relevant to the IoT Data Pre-Processing module.
D7.1	Digital Twin Platform Design & Interface Specification v1	Report on the Digital Twin Platform's detailed architecture, providing details on its timeseries database for storing IoT data, and the supported communication protocols for interfacing with the IoT Data Pre-Processing Module (e.g., Kafka/MQTT/HTTPS).

Functionalities, inputs, outputs, and exposed APIs that are introduced in this document should be considered in conjunction with regular updates being performed on the COGITO ontologies in the context of T3.2 “COGITO Data Model, Ontology Definition and Interoperability Design” activities. Furthermore, the input and output format of the IoT Data Pre-processing Module, documented here, should be reflected on the Extract, Transform and Load (ETL) services that are being developed within T7.2 “Extraction, Transformation and Loading Tools (ETL) & Model-Checking”.

1.3 Structure of the Deliverable

To address aspects relevant to the scope of T3.3 and T3.4, Section 1 summarises the work that has been conducted up to M18. Furthermore, D3.6's relevance to other COGITO tasks and deliverables along with the deliverable's structure, are introduced in Section 1.2 and 1.3, respectively.

A survey conducted towards designing, developing and delivering the COGITO's IoT solution feeding the IoT Data Pre-processing Module with raw IoT data, revealed several IoT RTLS techniques, related technologies and available solutions; they are presented in Section 2.

Section 3 summarises the module's development conducted up to M18 within T3.4. Having concluded to the IoT Data Pre-Processing Module architecture, its sub-components are introduced in Section 3.1. The RTLS solution proposed, purchased and installed at Hypertech's premises for experimentation purposes is presented in Section 3.2. The input and output formats alongside the exposed APIs of the module are documented in Section 3.4. The remaining parts of Section 3 present the technology stack, assumptions, restrictions, installation instructions and licensing matters.

Finally, in Section 4, conclusions are provided along with the release plan for the final iteration of the COGITO's IoT Data Pre-Processing Module as part of T8.1.

1.4 Updates to the first version of the IoT Data Pre-Processing Module

The first version of the IoT Data Pre-processing Module was documented in detail in the deliverable D3.5; since its submission (M16) the following additions/changes have taken place:

- addition of a **monitoring and alerting subcomponent targeting the status of the module** as described in Section 3.1; example snapshots are provided in Section 3.5;
- inclusion of **third-party tags** purchased for testing and experimentation as described in Section 3.2;
- additions and changes in the **implementation stack and technologies used** are reflected in Section 3.3;
- restructuring of the provided **Restful API** is detailed in the examples of Section 3.5.

2 Overview of the IoT RTLS techniques, related technologies and available solutions

The terms positioning and localization are often used interchangeably; more formally, positioning is the process of determining a point in space using a coordinate system (e.g., x, y, z) while localization refers to the linking of a position with semantics and attributes of the tracked entity and its location [1]. RTLS stands for “Real-Time Locating/Location/Localization System” and is the term commonly used when referring to systems providing identification and positioning (tracking) of persons or assets in real or near-real time. More specifically, an RTLS registers the two-dimensional (2D) or three-dimensional (3D) location of a tracked entity (spatial data), links it with temporal data (timestamps it) and makes it available (possibly after processing it) to any further interested parties/applications. The tracking data can either be provided uniformly over time (granularity defined by a pre-set sampling frequency) or upon triggering an event. Location data from RTLS systems are sometimes collected in conjunction with other sensor data such as biometric (heart rate, oxygen, galvanic skin response) or movement (inertial measurement units (IMUs), body joint and angle tracking through video or depth) data [2]. An RTLS might be realised utilizing different positioning techniques and technologies; this section serves as a compact overview/survey on the RTLS potential implementations along with a comparison of each of the latter’s advantages and disadvantages.

In general, an RTLS should address the following generic requirements [3]:

- **Accuracy:** the provided location data has to be consistent with the actual position of the tracked element. Unless this is the case, an RTLS cannot be relied on to be integrated within the context of an automation process. Factors such as the coverage area, the output data format and the update rate also fall under the same category since they can ultimately affect the system’s accuracy directly or indirectly.
- **Robustness:** the system might have to be deployed in a challenging, rapidly evolving environment; obstacles might scatter or block the signal or even prevent the optimal installation of the equipment. Additionally, the system might have to be unintrusive (e.g., to ensure user acceptance), posing further limitations.
- **Scalability:** should the requirements change, an RTLS solution must be expandable with the least possible effort and architectural modifications. In this context, the cost of the system planning, provisioning, deployment and operation have to be considered well in advance.
- **Availability:** depending on the use case an RTLS might require virtually no downtime; countermeasures such as back-up (redundant) equipment might thus be necessary further adding to the complexity and cost of a solution.

A more comprehensive list of criteria for the selection and assessment of an RTLS can be found in [4].

The expected level of accuracy directly impacts the selection of the physical implementation as well as the required complexity of the localization technique. More particularly, one can define three levels of accuracy:

- **Presence:** a system that uses binary indicators to illustrate whether a mobile node to be tracked is present (or not) in a designated area.
- **Proximity:** a system providing relative information on where a mobile node is located in relation to a set of fixed nodes (anchors), i.e., inferring the mobile node’s location based on the anchor’s location (coarse localization).
- **Positioning:** a system providing the absolute position of a mobile node in a predefined coordinate system.

2.1 Positioning Technologies

In general, depending on the technology used, an RTLS can be classified using two broad categories, RF-based and non-RF-based ones. It should be stressed that the vast majority of commercially available RTLS make use of RF-based technologies. An overview of the RTLS taxonomy based on the technology used is provided in [5], [6], [7], [8], [9].

2.1.1 RF-based

In order to perform location measurements, RF-based systems are commonly based on data exchange and/or exploitation of the properties of transmitted and received signals such as the power level (signal strength) or the angle of arrival. Such systems comprise of fixed nodes (anchors) installed at precisely known locations and mobile tags worn by the personnel or attached to assets to be tracked. As the tags move around in space, they transmit or receive signals from/to the anchors respectively and depending on the technology and algorithms used, the formers' position in space can be estimated.

Regarding the selection of tags, depending on what entity is going to be tracked (i.e., whether persons or assets are of interest), the following points should be taken into consideration:

- Physical dimensions (size & shape), weight
- Power requirements
 - Battery life, battery replacement/recharging possibility
- Durability
 - IP protection class
 - Operational temperature & humidity
 - Mechanical robustness
- Tag functionality
 - Accelerometer, push buttons
 - Visible feedback (e.g., LEDs, flashing lights, display)
 - Aural feedback (e.g., siren, buzzer)
 - Haptic feedback (vibration motor)
 - Environment sensor data (e.g., temperature, humidity, atmospheric pressure)

2.1.1.1 GNSS/GPS

The Global Navigation Satellite System, GNSS (or more particularly the North-American Global Navigation System, GPS) is a typical satellite-based positioning system. Due to their ubiquitous coverage around the globe, they are the de-facto standard for outdoor location tracking and positioning. Such a technology is mature and is associated with a low-cost of provision (e.g., the majority of the smartphones in the market have integrated GNSS/GPS receivers).

The main requirement is line-of-sight conditions (LOS) with the satellites in order to acquire a position fix. Due to the low transmission power and the impenetrability of the various structural materials (e.g., concrete, rebar etc) they are deemed with poor indoor coverage and consequently inadequate accuracy in such use cases.

In an effort to expand GPS capabilities for indoor environments, the Assisted GPS (AGPS) uses a location server with a reference GPS receiver that relays GPS data to a mobile terminal assisting in the identification of weak GPS signals. The average accuracy is typically in the order of tens of meters.

2.1.1.2 Bluetooth/BLE

Conventional Bluetooth is a widespread technology for Wireless Personal Area Networks (WPANs). Its range is in the order of 10-15m using small, low-power, inexpensive transceivers; the spectrum of devices it is built-in ranges from mobile phones to smart TVs and home appliances.

Bluetooth Low Energy (BLE) is an evolution of the conventional Bluetooth technology involving significantly less power consumption which in turn allows for the use of smaller batteries (e.g., coin cell batteries). The single-charge device operational life ranges from months to years depending on the power consumption profile used and the battery type selected.

Bluetooth and BLE are not interoperable as they make use of different PHY and link-layer.

Using advertisement messages (e.g., including the ID of the device) in only three predetermined channels, BLE can accomplish very quick communication with the infrastructure (anchors), typically in the order of 1 s or less. BLE 5.0 also includes provisions for direction finding making it a suitable candidate for use in the

scope of location systems. Depending on the implementation, BLE is capable of achieving accuracy in the order of a few meters or even sub-meter.

2.1.1.3 Ultra-wideband

Ultra-wideband refers to a set of RF communication technologies that involve signal transmission using a bandwidth in excess of 500 MHz; the available frequency range spans from 3.1 GHz up to 10.6 GHz. The most prominent form of UWB communication is the so-called Impulse Radio UWB (IR-UWB), namely the transmission of low power, ultra-short duration pulses (in the order of ns).

Due to the very large occupied bandwidth, the signals are resilient to narrowband RF interference from neighbouring links; also, due to the high temporal resolution the short pulses offer, IR-UWB exhibits excellent performance in multipath conditions.

With respect to indoor positioning, UWB can provide extremely high accuracy (in the order of cm). It is suitable for 2D or 3D positioning using TDoA, AoA or fingerprinting techniques (see section 2.3 for more information on the techniques).

Despite the high accuracy advantages, the hardware required is complex and of high-cost effectively limiting the applicability of UWB solutions.

2.1.1.4 Wi-Fi

Wi-Fi or more generally the collection of IEEE 802.11 standards for Wireless Local Area Networks (WLANs) is a well-established, widespread technology used for wireless communication at high data rates. The use of Wi-Fi for positioning bears many advantages as it exploits ubiquitous, already existing infrastructure, it is easy to deploy, is cost-effective and allows for simultaneous communication and positioning functionalities.

Wi-Fi-based positioning solutions are for the most part based on RSS measurements combined with a propagation model or on fingerprinting (see section 2.3 for more information on the techniques).

Their accuracy is regarded as low to medium (a few meters at best).

2.1.1.5 ZigBee

ZigBee or IEEE 802.15.4 is a standard developed to allow low-cost, low-power, low data rate, and close proximity RF communication for WPANs or WSNs. Its range spans from 10 to 100m depending on the transmitted power and the environment. Zigbee traditionally forms mesh topologies; when used for positioning it can provide the rough location (in the order of a few meters or tens of meters) making it suitable in applications where the presence or proximity are adequate.

2.1.1.6 RFID

RFID systems comprise of a set of tags that communicate with one or more readers; tags provide their ID and potentially other data stored or pre-programmed in their ICs. They can be classified in two categories, active and passive tags.

Active tags require a power source (e.g., a battery or supercapacitor) in order to operate. They can initiate communication to nearby reader(s) either periodically or upon triggering an event.

Passive tags either reflect back the signal from the reader or are energized wirelessly by harvesting the energy transmitted by the RFID reader. Passive tags only respond to the reader who is responsible for initiating the communication.

RFID technology is regarded as low-cost; nevertheless, it offers limited positioning accuracy and is oftentimes used merely to detect presence or proximity. Also, another disadvantage is that there is a risk of tag collision, i.e., the reader is incapable of simultaneously processing multiple tags.

2.1.1.7 Near-Field Electromagnetic Ranging (NFER)

This technique is based on a physical property of the electromagnetic signals, namely that the phase difference between the electric and magnetic field components is 90° at the transmitter's antenna,

decreasing with increasing distance. Therefore, performing a phase comparison the distance to the transmitter can be coarsely estimated.

2.1.2 Non-RF based

2.1.2.1 Infrared (IR)

Infrared positioning makes use of infrared light pulses to locate objects; a transmitter emits IR pulses and neighbouring readers receive them after which the position can be estimated. The main drawbacks of this technology are primarily that Line-of-Sight is utterly essential (opaque obstacles block the signal) and moreover that interference from other infrared sources including sunlight can render the system non-operational.

2.1.2.2 Ultrasound

Ultrasound positioning is conceptually similar to the RF ones, the main difference being that instead of RF signals ultrasonic acoustic pulses are emitted and received. A neighbouring receiver is able to calculate the distance to the transmitter according using the Time-of-Flight (see section 2.3 for more information on the techniques). Generally, Line-of-Sight (LOS) conditions are required. Ultrasound systems are less prone to multipath than RF signals and also immune to RF interference, although they can suffer from interference from other nearby ultrasound sources.

2.2 Positioning topologies

The topologies presented below are typically applicable (although not limited) to RF-based localisation systems. Depending on the entity that undertakes the localization procedure (calculation of the position), four main topologies can be distinguished [10]; irrespective of the topology, the system comprises of the mobile node(s) to be tracked and several fixed nodes (anchors) with precise knowledge of their position in space. A virtual coordinate system is thus formed.

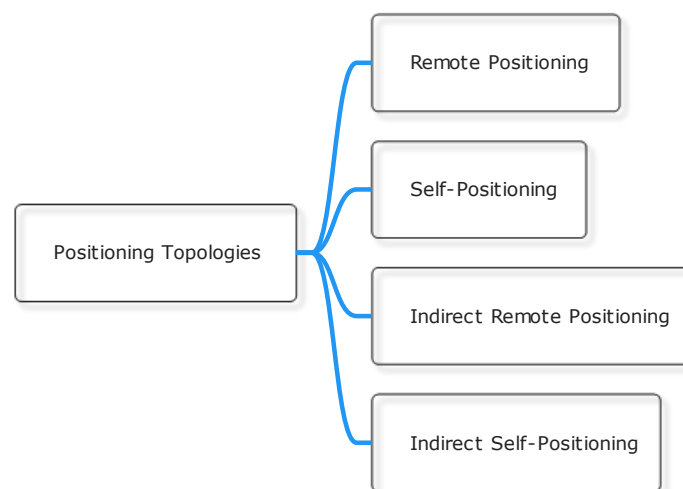


Figure 3: Overview of potential topologies used for positioning

2.2.1 Remote Positioning

The mobile node (to be tracked) transmits data advertising its presence to the several fixed devices (anchors); the data are received by the anchors and forwarded to a centralised localisation engine (deployed on- or off-site) which performs the necessary calculations to obtain the position. The mobile node does not have access to its location information.

2.2.2 Self-Positioning

This case is the inverse of the previous one; the mobile node itself receives signals transmitted by the anchors using which it is able to calculate its position in space. The location information is restricted to the mobile node.

2.2.3 Indirect Remote Positioning

A special case of self-positioning where the positioning is performed on the mobile node and then shared with an external system or observer.

2.2.4 Indirect Self-Positioning

Indirect self-positioning is a special case of remote positioning; after performing the positioning calculations using a remote/centralised localisation engine the result is propagated back to the mobile node.

2.3 Positioning techniques - algorithms

Depending on the technique used to determine the position of a mobile node, the following classification can be established. Most of the techniques and algorithms can be applied to both outdoor and indoor tracking applications [3], [11].

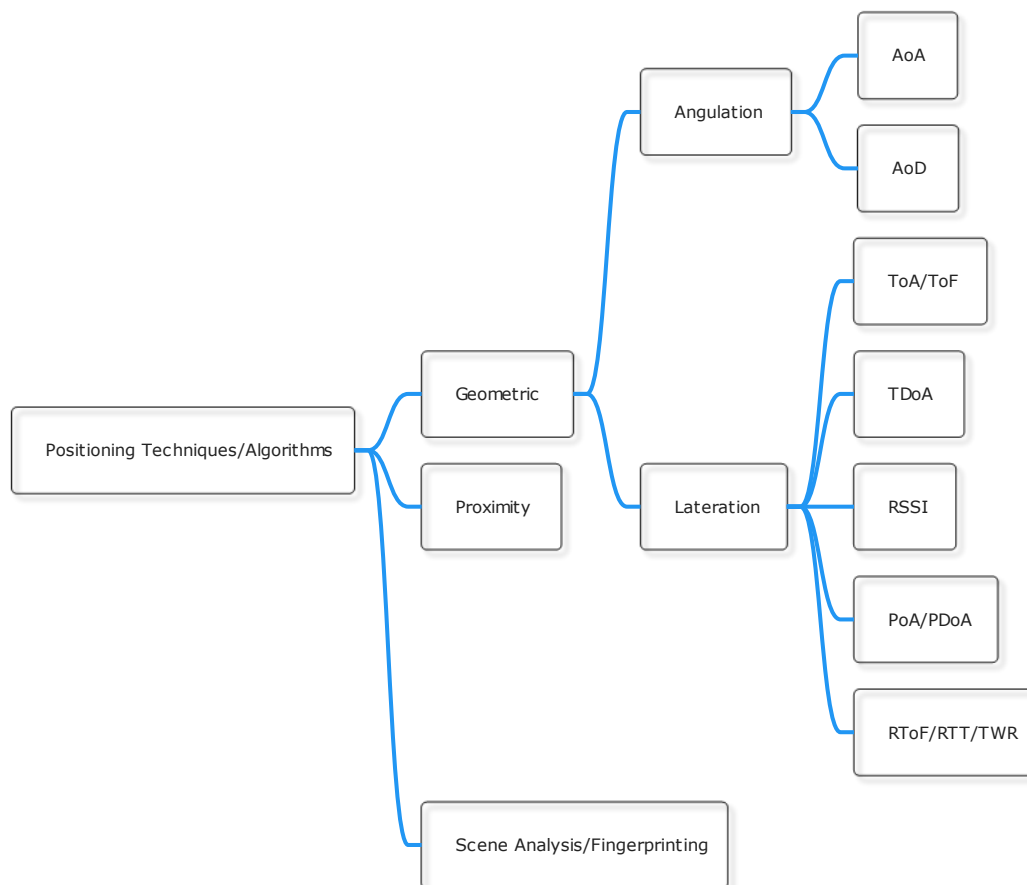


Figure 4: Overview of the most common RTLS techniques

2.3.1 Geometric techniques

2.3.1.1 Angle-based (angulation)

Angle-based techniques make use of the incident angles through which the signal reaches the receivers (Angle of Arrival) or is transmitted by the transmitters (Angle of Departure) in a system as in Figure 5; by intersecting two direction lines formed between transmitter-receiver pairs, the 2D tracked object's position can be derived.; for the 3D position, the intersection of three direction lines is required (and hence the knowledge of three angles). In general, the hardware required for the operation of such systems is complex and costly; additionally, the farther away from anchors the tracked object is, the lower the achieved accuracy.



Figure 5: Graphical representation of the angle-based (angulation) techniques

2.3.1.1.1 Angle of Arrival (AoA)

Using directional antennas (antenna arrays) on the receiver side, the angle at which the signal reaches an anchor can be determined; then using angulation (estimation of the intersection point of the direction lines), the position is estimated.

2.3.1.1.2 Angle of Departure (AoD)

Again, using directional antennas or antenna arrays, nevertheless on the transmitter's side, the angle at which the signal departs to reach an anchor is used to form a direction line. The intersection of two or more direction lines yields the position estimate.

2.3.1.2 Distance-based (lateration)

Distance-based techniques, also commonly referred to as lateration, produce a position estimate for a moving node by measuring its distances (range) from multiple reference points (anchors). The position estimate is calculated such that the set of distances is best fit as in Figure 6. When three distances are used the method is called trilateration, while when more than three are used, it is referred to as multilateration.

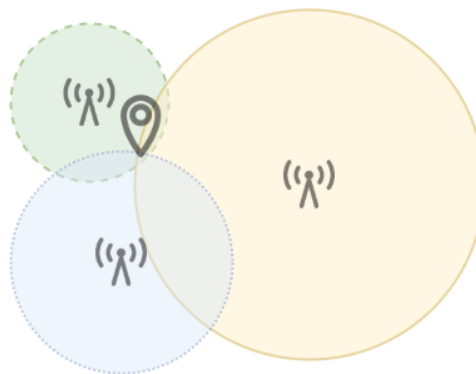


Figure 6: Graphical representation of the distance-based (lateration) techniques

2.3.1.2.1 Time of Arrival (ToA) / Time of Flight (ToF)

Such systems include temporal information (timestamps) in their signals signifying the time the signal was transmitted; upon receipt on the receiver's side, the signal timestamp is extracted and subtracted from the corresponding current timestamp, yielding the so-called time of flight, i.e., the time the signal required to propagate from the transmitter to the receiver. This time duration multiplied by the propagation speed of the signal (typically radio frequencies, sound or light, known for the transmission medium, i.e., air) provides an estimate of the distance (circle of radius r) across the transmitter-receiver. With three distances known, a trilateration method as in Figure 6 can be performed to assert the tracked entity's location. A major drawback of this technique is that it requires very tight synchronization among the transmitter & receiver clocks in order to yield accurate results.

2.3.1.2.2 Time Difference of Arrival (TDoA)

This technique closely resembles the aforementioned ToA/ToF technique; a signal is simultaneously transmitted/received by multiple anchors (common start of transmission/receipt accordingly) and upon receipt on the receivers' side the relative arrival time difference between the different signals is calculated; the resulting trilateration is then performed using hyperbolae instead of circles. This technique is also demanding in terms of clock synchronization regarding the anchors.

2.3.1.2.3 Received Signal Strength Indicator (RSSI)

RSSI is based on the fact that the signal strength decreases as the distance between transmitter-receiver increases. Using an appropriate path-loss model [12], the received signal level can be associated with a distance; using trilateration the location of the tracked object is then obtained. This method does not impose any requirements in terms of clock synchronization, however, the fact that assumes a particular environment (and thus a corresponding path-loss model) has a direct impact on the provided accuracy of the results. Also, noise or Non-Line-of-Sight (NLOS) conditions can severely deteriorate the consistency of the location information.

2.3.1.2.4 Phase of Arrival (PoA) / Phase Difference of Arrival (PDoA)

PoA uses the received carrier's phase (or phase difference) to estimate the distance between transmitter-receiver; to circumvent phase wrapping the phase can be evaluated using multiple frequencies. The distance can then be extracted using the rate of phase change [3].

2.3.1.2.5 Roundtrip Time of Flight (RToF)/Round Trip Time (RTT)/Two Way Ranging (TWR)

The RToF is conceptually similar to ToA/ToF, however, it bypasses the strict requirements on clock synchronization by measuring the time-of-flight of a signal traveling from a transmitter to the receiver and back, commonly referred to as the round-trip-time.

2.3.2 Proximity

The simplest yet least accurate localization technique; instead of providing precise position data (i.e., in the form of coordinates), information merely about whether a node is in the vicinity of an anchor is provided. This type of information can be obtained by applying a threshold to the received signal power on the anchors' side. Further simplifying this technique, data can be limited to binary information about the presence or absence of a mobile node within a space depending on whether a tag signal can be detected or not accordingly.

2.3.3 Scene Analysis – Fingerprinting

RF-based scene analysis techniques, also referred to as fingerprinting, involve the use of algorithms that try to estimate an entity's location by comparing the current location's signal features with the ones from a set of locations previously collected in a database. Consequently, such a procedure is executed in two phases, an offline and an online one. During the offline phase, a set of signal features for each location is sampled and processed; during the online phase, current measurements are used to match with one of the previously collected sets of features (fingerprints).

The signal feature most commonly used in the context of location fingerprinting is RSS measurements while the estimation algorithms are based upon pattern recognition principles and techniques such as [7], [13]:

- Probabilistic Methods
- k-Nearest Neighbour (Knn)
- Neural Networks (NN)
- Support Vector Machine (SVM)
- Smallest M-vertex Polygon (SMP)

Fingerprinting can typically utilize existing installations (e.g., Wi-Fi) and is the preferred positioning method when the procurement and/or deployment of new infrastructure is not an option. The provided accuracy is usually in the order of a few meters [7].

Most (conventional) fingerprinting techniques require a pre-deployment site survey (offline phase). This allows for the construction of a radio map representing signal strength at the various points in space; such a practice can be particularly time consuming and labour intensive while at the same time vulnerable to environmental changes. Also, considering the heterogeneity of devices available in the market, each with different RF front-end configurations, a large deviation in obtained fingerprints could yield inconsistent results. It should be noted that lately, various calibration-free fingerprinting methods have gained attention [13].

3 IoT Data Pre-processing Module

The IoT Data Pre-processing module builds on top of a location tracking system; as described in the previous sections of this deliverable, there is a multitude of technologies and techniques facilitating IoT location tracking; this is also reflected in the numerous commercial solutions available in the market. In absence of any serious standardization effort, most of them are tied to a particular technology and/or positioning technique, use proprietary hardware and data exchange protocols. They typically provide only predefined functionalities and accuracy, lacking interoperability with other platforms.

The rationale behind the development of the COGITO's IoT Data Pre-processing Module is the creation of a technology-agnostic middleware capable of fusing the various RTLS technologies and techniques currently available in a unified, consistent manner. This offers the potential to enhance accuracy by using multiple positioning systems in tandem, circumvents vendor lock-in, allows for data augmentation using custom pre- and post-processing schemes, and provides a unified API for data ingestion, delivery and monitoring.

3.1 Implementation Details

The module's implementation can be partitioned in the following logical subcomponents visually presented in Figure 7.

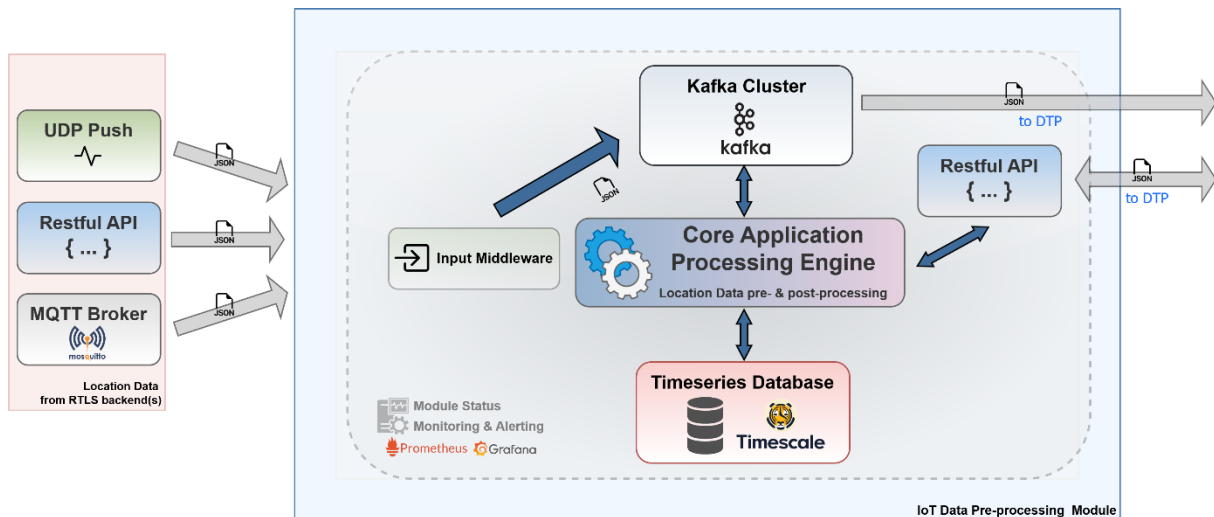


Figure 7: The IoT Data Pre-processing Module implementation architecture (arrows define the direction of data flow)

3.1.1 Communication subcomponents

Manage the communication of the module with the RTLS backend(s) and the DTP.

3.1.1.1 Backend Communication

- Establishes and maintains the connection with the proprietary IoT RTLS backend(s) (e.g., Quuppa) using an appropriate handler
- Requests or receives the location tracking data potentially using:
 - a RESTful API client - requests data over HTTP(S) using polling
 - a TCP/UDP socket to receive a continuous data stream of packets
 - an MQTT:
 - broker to allow data publication in topics
 - client to allow subscription to a topic and receive updated data
- Parses the input data (commonly JSON) to a format appropriate for further processing

3.1.1.2 DTP communication

- Creates and manages the available output REST API endpoints responding to requests on-the-fly

- Publishes the near real-time pre-processed data to corresponding topics on the Kafka cluster

3.1.2 Database subcomponent

Handles local data persistence and retrieval throughout the component. It implements:

- A database suitable for timeseries data (e.g., TimescaleDB or InfluxDB)
- A database wrapper providing an abstract way to communicate with the core application subcomponent

3.1.3 Core application subcomponent

The main subcomponent of IoT pre-processing module. It implements the following functionality:

- Aggregates the available data (possibly fusing them if multiple RTLS backends are available)
- Cleans/filters/pre- or post-processes them according to a preconfigured set of rules and algorithms
- Manages:
 - the distribution of the pre-/post-processed data (hand-over to the communication subcomponent for API or Kafka publishing)
 - the data retrieval in case an API call is received
 - the storage to the DB for back-up and performance meta-analysis

3.1.4 Module status monitoring & alerting subcomponent

As the IoT Data Pre-processing module involves continuous (24/7/365) operation on raw real-time data, its status as well as performance in terms of latency has to be monitored. To this end, a subcomponent based on industry-standard technologies was implemented featuring the following functionality:

- Monitoring of the **RTLS backend** status:
 - uptime history
 - communication latency/errors
 - resource usage, project information
 - tags registered incl. status
 - average accuracy (instantaneous and historical)
- Monitoring of the **RTLS input middleware** (currently limited to the UDP handler):
 - uptime history
 - input/output records processed
 - processing latency (instantaneous and historical)
- Monitoring of the **Core application**, i.e., the streaming processing pipeline:
 - uptime history
 - input/output records processed
 - processing latency (instantaneous and historical)
- Monitoring of the **API status**:
 - uptime history
 - processing latency/errors
 - resource usage
 - requests history
- Monitoring of the **Kafka cluster**:
 - uptime history
 - active connections history
 - resource usage history
 - read/write rates
 - queue latency
 - errors

3.2 Prototype Overview

3.2.1 Proprietary RTLS solution

After reviewing many commercial RTLS solutions and considering the module's requirements for location tracking, i.e., the fact that it mainly involves indoor scenarios where GNSS technologies are troublesome, the Quuppa system was selected.

3.2.1.1 Overview

3.2.1.1.1 Quuppa product description

Quuppa is an RTLS technology platform providing real-time tracking for tags and devices “using direction finding methods and advanced proprietary algorithms” [14]. The platform uses BLE-compliant technologies to achieve real-time sub-meter accuracy over a long range even in challenging environments [14]. It is a solution employed in several vertical markets including manufacturing, transportation, smart buildings, healthcare, sports, asset tracking and more [15].

3.2.1.1.2 Technologies used, PHY, topologies, localization methodology

The system comprises of anchor devices (termed “locators”) and tags that are attached to objects or worn/held by persons. Each Quuppa locator employs a directional antenna providing a cone-shaped radiation pattern; the locators are responsible for receiving the signals transmitted by the tags over BLE or a BLE-alike proprietary protocol.

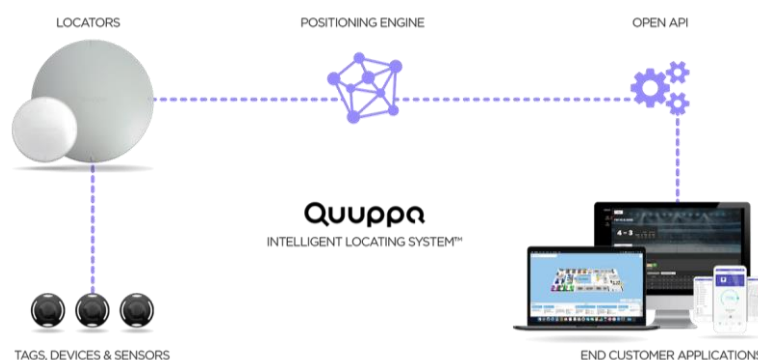


Figure 8: Overview of the Quuppa system [14]

All available data is relayed to a local or remote position processing engine software which after running proprietary algorithms produces the position estimate (Figure 8). Using the AoA positioning technique in the azimuth and elevation plane, the 2D position estimate of a tracked entity can be determined even using one locator; for the 3D position or when high accuracy is required, the position is calculated as the intersection of two or more lines as shown in Figure 9.

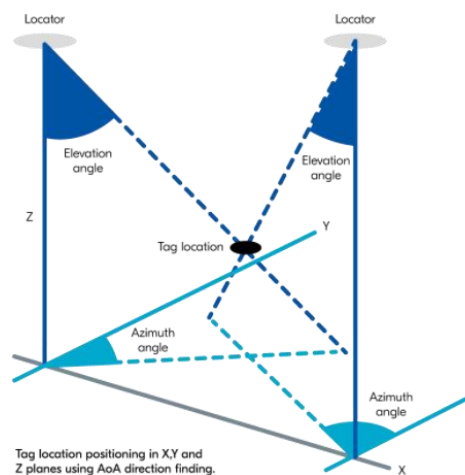


Figure 9: Then angle-based technique (AoA and AoD) [16]

The position estimate together with any other available attributes or meta-data is made available to a RESTful API or via UDP Push to preconfigured clients. System metrics and other relevant operational status indicators are also available via the REST API.

3.2.1.1.3 Key-features & advantages

Quuppa was chosen as the backend of the IoT Data Pre-processing module as it is a cost-effective yet highly effective RTLS providing a claimed accuracy down to 10 cm [15]. It features real-time location updates at a maximum update rate of 50 Hz and a latency down to 100 ms in the case of a standard system. It supports a multitude of BLE-capable tags and devices including smart phones.

Despite being fully interoperable with Bluetooth, Quuppa can provide resilience to possible interference by switching to proprietary channels at the edge of the ISM band as depicted in Figure 10.

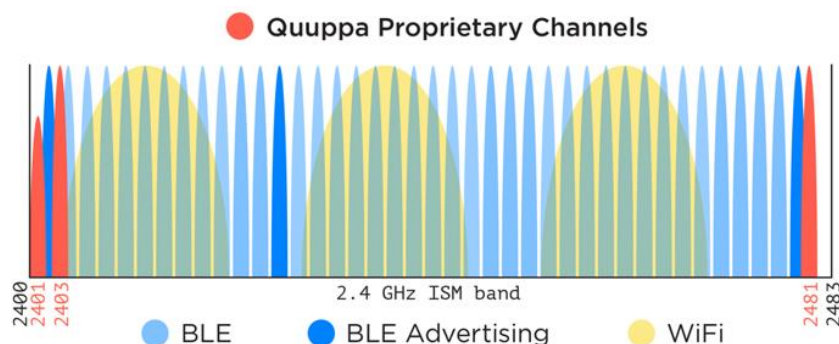


Figure 10: The distribution of BLE, Wi-Fi and Quuppa proprietary channels in the ISM frequency band

The computed tag positions can be integrated to existing/third party applications and services using both push and pull APIs (UDP push, REST API endpoints) in JSON and CSV formats. Besides the tag data (position, sensor data), the REST API allows the monitoring of the system status as well as execution of remote commands. The key features along with a comparison with other relevant technologies are summarised in Figure 11.

	Quuppa (Bluetooth Direction Finding)	Bluetooth® (RSSI)	UWB	Wi-Fi	RFID	GPS
ACCURACY	< 1m	1-5m	< 1m	5-20m	1-5m	5-20m
REAL-TIME	< 1 sec	10-30 sec	< 1 sec	10-30 sec	10-30 sec	10-60 sec
BATTERY CONSUMPTION	Low	Medium	High	High	Low	Very High
RANGE	upto 300m	upto 300m	upto 200m	upto 150m	upto 5m	Global
IOT GATEWAY*						
SMARTPHONE COMPATIBLE						
TOTAL COST** OF OWNERSHIP	€	€	€€	€€	€€	€€€
SCALABILITY	1000s of Tags	Unlimited	1000s of Tags	100s of Tags	1000s of Tags	Unlimited
SECURITY	★★★★★	★★★☆☆	★★★★★	★★★☆☆	★★★☆☆	★★★★★

* Capability to transmit data effectively for remote sensing

** Defined by cost of initial planning & deployment cost + 3-5 years maintenance

Figure 11: Comparison of Quuppa with other relevant technologies [14]

3.2.1.1.3.1 Tags

Quuppa tags can generally be used for both positioning and remote sensing. There is a substantial number of readily available Quuppa-approved tags that can be purchased depending on the use case; a tag can be used as is or be even further customised or redesigned to satisfy the requirements of a particular project. To this end, bare tag PCB modules as well as firmware libraries and relevant development resources are available through the customer's portal. It should be noted that apart from the Quuppa approved tags, smartphones and other BLE-capable sensors can be tracked when running (or emulating) the Quuppa firmware (i.e., made capable of transmitting a Quuppa specified radio packet).

The Quuppa tag firmware implements State Machine logic, allowing the designer to choose between different states triggered by events (e.g., accelerometer-triggered, button-push etc) or timeouts to prolong battery life, adjust transmission power and rate and minimize radio noise in the channel.

3.2.1.1.3.2 Locators

The locators serve as the anchor reference points of the system, are the ones that receive the signals from the tags and forward them to the Quuppa Positioning Engine (QPE) for further processing. Indoor as well as outdoor locators are offered, operational across a wide range of temperatures and humidity levels (the outdoors are also IP66 rated). Both indoor and outdoor types have provisions for PoE for combined data communication and power supply; they can also be powered through micro-USB.

3.2.1.2 Tools Available

3.2.1.2.1 Site planner

The provided site planner (Quuppa Site Planner, QSP) is a software tool allowing the user to quickly plan, visualise, deploy and configure the system by means of a Graphical User Interface (GUI); step-by-step wizards are also available to guide the user through the various available options and settings. Using this application, a site plan can be imported, serving as the base upon which the system will be planned. The locators, i.e., the hardware anchors necessary for the operation of the system can be placed anywhere on the plan and the expected system performance calculated based on the distances and the environment type is provided. This allows for fine tuning and optimization both before as well as after the actual deployment. Finally, the tool allows for tag programming and configuration while offering an API editor to customize the output format of its positioning engine.

3.2.1.2.2 Simulator

The Quuppa System Simulator (QSS) allows the testing and validation of a planned system prior to its actual deployment. By creating a fully functional virtual RTLS, the user can evaluate the design choices they made and verify that the hardware meets the target requirements. One is then able to revise their plan accommodating any changes or lessons learned through the simulations, without spending money, time and effort on a physical installation.

3.2.1.2.3 Data retrieval

Quuppa offers a fully customizable API; data attributes and metadata can be selected/omitted depending on the particular use case. The API is published and accessible to third parties/applications as a RESTful endpoint, however, there is also an option for UDP push. The latter, enables the transmission of UDP packets to a target, user-defined client, alleviating the need for constant polling and allowing for faster streaming (minimizing latency). The exchanged file format can either be JSON or CSV. Any changes on the system status are either published at a pre-set sampling frequency (e.g., 2 Hz) or upon event triggering (e.g., the movement of a tag); the change of the system status is reflected simultaneously at the API endpoints and by means of UDP packets transmission.

3.2.1.2.3.1 RESTful API

In the context of COGITO, the restful API is not employed as a means to retrieve location data; in any case, it remains an option (in this case, data available by the Quuppa system is retrieved by regularly polling a restful endpoint)

3.2.1.2.3.2 UDP Push

This is the selected technique for retrieving data from the Quuppa system. It requires no action on the client side (the custom developed middleware – pre-processing application) which only has to receive the UDP bytes and decode them in a usable JSON format.

3.2.2 Small-scale test deployment

3.2.2.1 Purchase of a development kit

In order to evaluate the capabilities and performance of the Quuppa system a development kit was purchased. The kit contained [17]:

- 5 Quuppa Q17 Locators and mounting brackets
- 1 Quuppa Q17 Locator configured as a Focusing Locator (marked with a sticker)
- 5 Quuppa QT1-1 Tags
- 1 Quuppa Controller computer configured with the Quuppa software
- 1 USB-Ethernet Adapter for internet connectivity
- 1 Micro USB-USB cable for powering the Focusing Locator
- 1 Quuppa Development Kit Quick Start Guide

To deploy it the following extra components were used:

- PoE Switch
- UTP Cables
- Laptop computer
- Laser measurement tool
- Tripod

In Figure 12 the received kit along with the PoE switch used is presented (one of the mounting brackets was detached from a locator for the sake of this photograph).



Figure 12: The received Quuppa development kit along with a separately purchased PoE switch

3.2.2.2 Requirements' assessment, project planning & simulation

In order to achieve the best possible performance, a good understanding of the platform is important; this allows for optimal setting the various parameters and configuration options throughout the system. In the following figures the various tools provided by Quuppa are shown, namely the planning tool, QSP (Figure 13 and Figure 14), the simulator interface, QSS) in Figure 15 and a snapshot of the generated real-time location map (Figure 16).

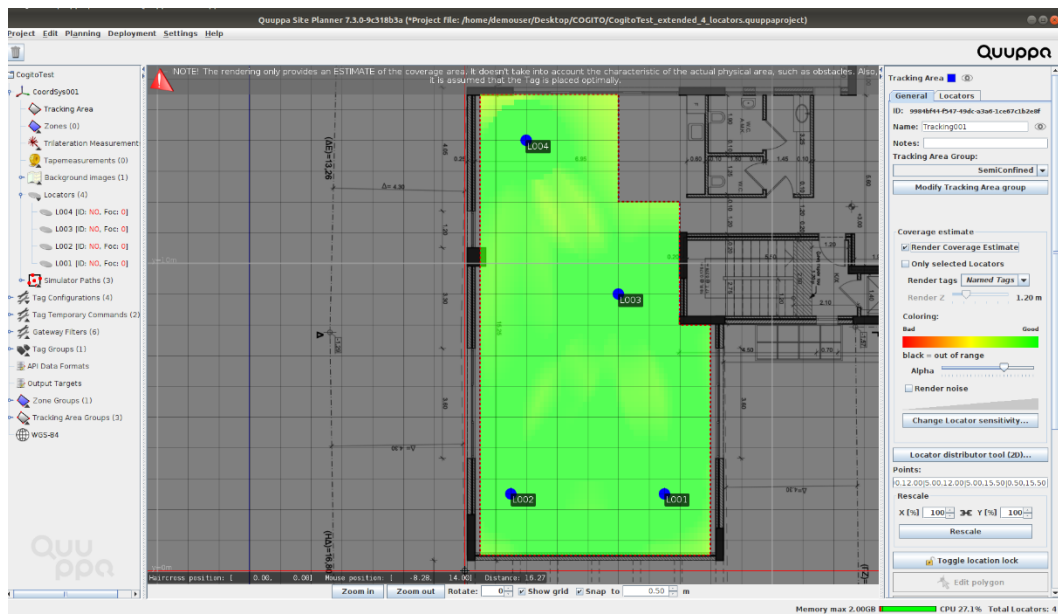


Figure 13: Using the provided planning tool (QSP) to create and plan a new project

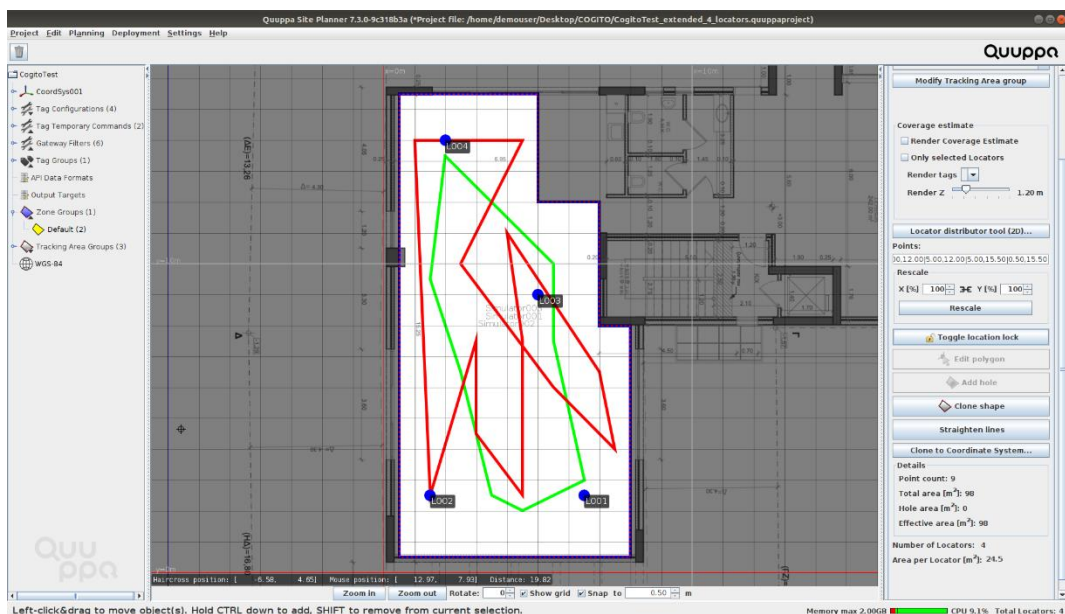


Figure 14: Creation and configuration of simulation scenarios

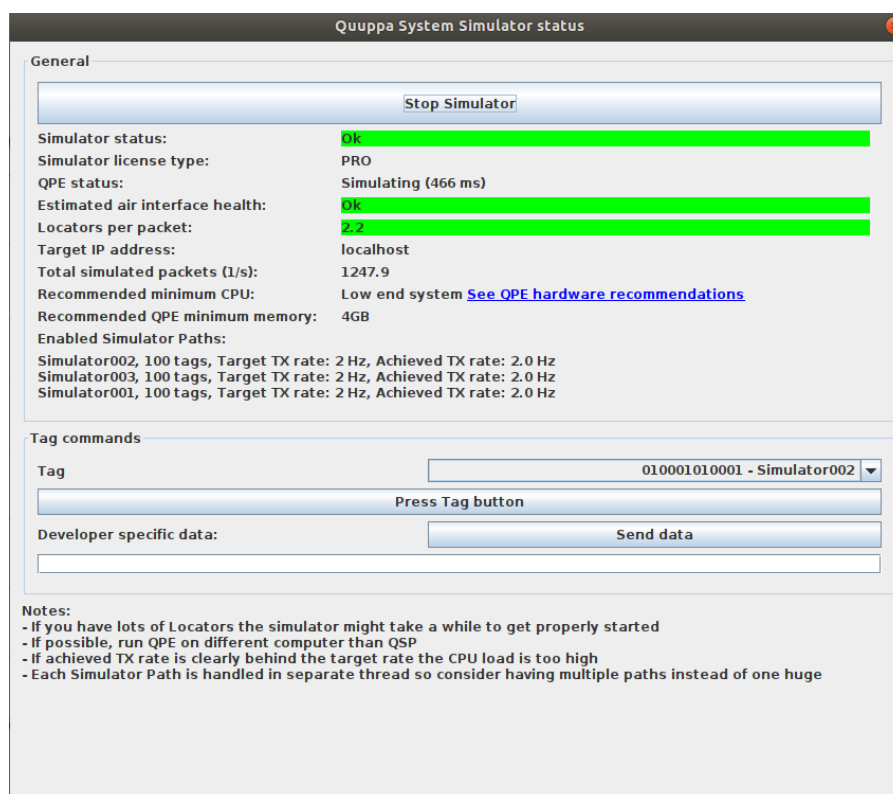


Figure 15: Actual run of the simulation scenario

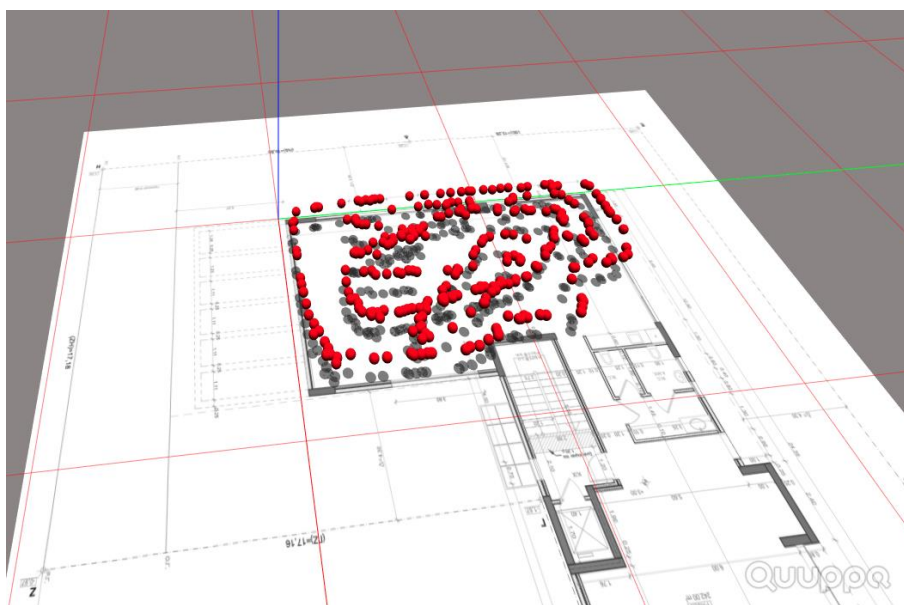


Figure 16: Snapshot of the generated floor map presenting the position of the simulated tags

3.2.2.3 Actual deployment and performance evaluation

After carefully studying the product documentation, familiarizing with the technologies involved as well as the installation and operational requirements, an actual deployment at the company's premises was decided; to this end, the first floor of Hypertech Energy Labs' offices was selected as the deployment location.



Figure 17: Installation and cabling of the Quuppa locators above the false ceiling

The deployment included the installation of four (4) locators at optimal locations according to the plans and simulations created by the Quuppa software tools. The devices were placed horizontally above the gypsum tiles of the false ceiling at a height of approximately 2.43m above the ground as shown in Figure 17; no fixing material was required. A single UTP Cat6 cable for each locator provides both power and data communication with the Quuppa Software running on the provided Intel NUC mini-computer through a PoE switch. The latter was also installed above the false ceiling at a strategically selected location to minimise cable runs and was connected to the Intel NUC by means of a single passive PoE adapter kit; the NUC itself is placed on a shelf right below the PoE's location. The entire installation is completely unobtrusive without any visible components or irreversible alterations of the building's elements.

After the locator's installation and cabling, a calibration procedure is required; the calibration consists of three phases: the definition of the locators' position relative to the used floor plan, an "identification" and a "focusing" phase. Throughout the calibration process, a Quuppa "focusing locator" was used, i.e., a regular locator booted in an alternate mode. The focusing locator was connected to a laptop with all the Quuppa related tools running in a VM. For practical reasons, as also suggested in the Quuppa manuals, the focusing locator was mounted on a tripod as in Figure 18. Before proceeding to the calibration phase, the system must enter a "deployment mode".

Using the floor plan as a relative coordinate system, the locator's coordinates can be estimated e.g., with respect to nearby walls or other structural components of the building. The distances were measured using a laser measurement tool; for each locator, the distance to two neighbouring building structures/elements is sufficient to define its position. Also, the height was measured, which identical for all locators.

The "identification" of the locators, i.e., the assignment of a unique ID to each locator is performed using the Quuppa QSP tool while placing the focusing locator under (or very close) to each locator. Each locator is automatically recognized and assigned an ID by the QSP software.

The next step is the "focusing" of the locators, i.e., a procedure that allows the QSP software to estimate each locator's orientation (elevation, azimuth and rotation). This is accomplished by pointing the focusing locator towards each locator from a distance and entering the orientation in the QSP focusing wizard. Generally, pointing towards the locator from two different positions is sufficient to fully define each locator's orientation.



Figure 18: “Identification” and “Focusing” of the locators using the Quuppa QSP

In order for the tags to be used, they have to be configured by placing them on top of (or very close to) the focusing locator while running the tag configurator tool available within the QSP software.

After finishing with the calibration phase the system is fully operational and ready to enter the tracking mode.

3.2.2.4 Testing and experimentation with additional tags

As already mentioned in previous sections, although Quuppa provides its own tags, there are also numerous Quuppa-compatible third-party tags available in the market. In fact, there are tags integrating sensors that allow for additional functionality, e.g., apart from positioning they can be used for remote sensing of the environment or the entity bearing them, or providing haptic, audible or visual notifications.

Such tags are either based on the officially released Quuppa firmware (transmitting proprietary packets) or operate by emulating it. To test whether attempting to use third-party tags would be reasonable within COGITO, two further types of tags were purchased, one offered as a commercial product ready to use and another one in the form of a fully customisable Printed Circuit Board (PCB) as described below. The third-party tags along with one provided by Quuppa as part of the development kit are depicted in Figure 19.

- Blueup Quuppa SafeX [18]:
 - full support of Quuppa Intelligent Locating System™
uses native Quuppa libraries, full support of the Quuppa RTLS software features, including back-channel for configuration and commands, OTA FW update, proprietary channels,
 - replaceable CR2477 battery,
 - 3-axis accelerometer,
 - alarm button,
 - 2 LED,
 - buzzer,
 - vibration motor,
 - wearable waterproof IP65 enclosure offered in three options:
 - SafeClip (clip for belt/pocket),
 - SafeCord (eyelet for lanyard),
 - SafeBand (silicone wrist-strap).

- onsemi SECO-RSL10-TAG-GEVB Asset Tag Evaluation Kit [19]:
 - used with Quuppa Intelligent Locating System™,
 - development platform based on the RSL10 System-on-Chip (SoC),
 - environmental sensors,
 - 3-axis accelerometer.



Figure 19: The Quuppa original and third-party tags purchased for testing

The actual functionality and potential advantage of utilising third-party tags (either the particular ones or even further ones available in the market) are going to be evaluated during the integration- and pre-validation-related tasks. In any case, depending on the pilot sites requirements such (or other third-party) tags seem to be a sensible option to consider.

3.3 Technology Stack and Implementation Tools

The technologies used throughout the module as well as the implementation tools and libraries are summarised in the Table 2 below.

Table 2: Summary of the technology stack and implementation tools

	Technologies Used	Implementation tools/libraries used
Communication subcomponent I Backend communication	<ul style="list-style-type: none"> • REST API • TCP/UDP sockets • MQTT broker/client • Apache Kafka cluster 	<ul style="list-style-type: none"> • Python (main libraries: qmqtt, anyio) • Mosquitto MQTT broker • Apache Kafka/Redpanda
Communication subcomponent II DTP communication	<ul style="list-style-type: none"> • REST API • Apache Kafka cluster 	<ul style="list-style-type: none"> • Python (main libraries: FastAPI, sqlalchemy/sqlmodel, faust, pydantic, orjson) • Apache Kafka/Redpanda
Database subcomponent	<ul style="list-style-type: none"> • Timeseries Database 	<ul style="list-style-type: none"> • PostgreSQL TimescaleDB

Core application subcomponent	<ul style="list-style-type: none"> In-house developed solution for data fusion and pre-processing 	<ul style="list-style-type: none"> Python (main libraries: Dask, streamz, faust, pandas, provision for inclusion of AI-related tools e.g., TensorFlow)
Module status monitoring & alerting subcomponent	<ul style="list-style-type: none"> Prometheus Grafana 	<ul style="list-style-type: none"> Python Prometheus' & Grafana's built-in tools

3.4 Input, Output and API Documentation

3.4.1 Contract-first API / ontology definition

As the API is designed from the ground up, virtually unlimited customisation can be achieved. After reviewing the input and output data that: (a) in the case of input could be required or would be readily available, or (b) in the case of output could be generated and could be of interest for the module itself and the third-party entities, a baseline input/output format was established. This approach, formally referred to as a *Contract-first* or *API-first* approach enables the development of an application completely tailored to the particular use case requirements imposed by the COGITO ecosystem.

A formal definition of the API was made possible using the OpenAPI 3 standards [20] and the YAML language. Thanks to tools like Apicurio [21] or Swagger [22], the creation of the API can be done step-by-step using a friendly GUI as in Figure 20. Apicurio can automatically generate the YAML file describing the API as shown in Figure 21, while both can automatically generate extended, concise documentation.

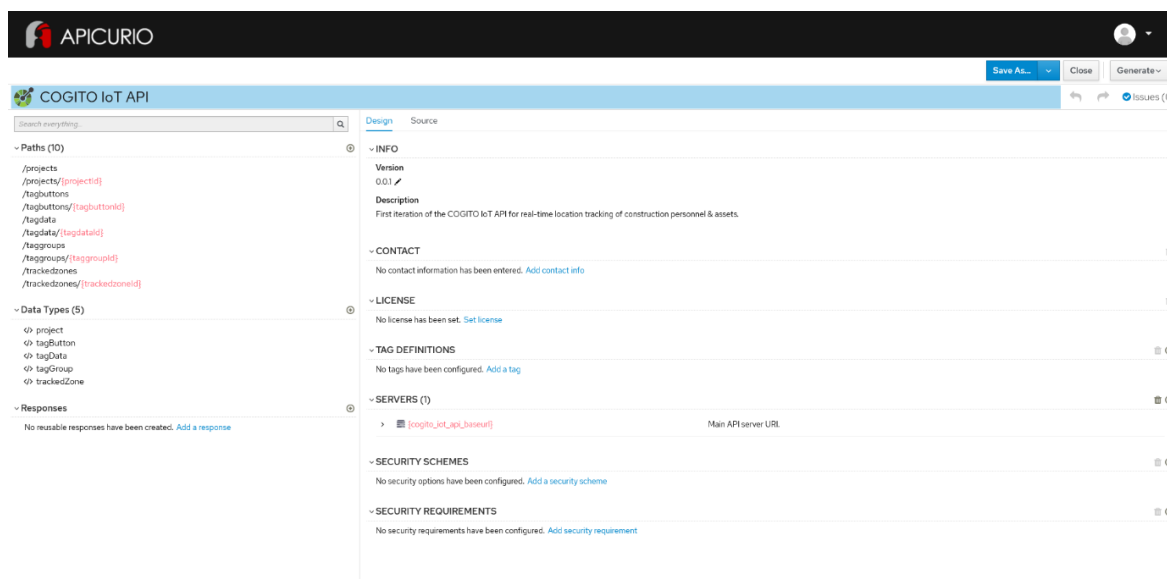


Figure 20: The Apicurio web application used to model the COGITO's IoT API

```

1  openapi: 3.0.2
2  info:
3    title: COGITO IoT API
4    version: 0.0.1
5    description: >-
6      First iteration of the COGITO IoT API for real-time location tracking of construction personnel &
7      assets.
8  servers:
9    -
10     url: '{cogito_iot_api_baseurl}'
11     description: Main API server URI.
12     variables:
13       cogito_iot_api_baseurl:
14         default: 'http://localhost/iot_api_test'
15         description: Endpoint for testing purposes.
16  paths:
17    /projects:
18      summary: >-
19        Path used to manage the list of COGITO projects/sites within which IoT RTLS functionality is
20        expected.
21      description: >-
22        The REST endpoint/path used to list 'Project' entities. This path contains a 'GET' operation to
23        perform the list task.
24      get:
25        responses:
26          '200':
27            content:
28              application/json:
29                schema:
30                  type: array
31                  items:
32                    $ref: '#/components/schemas/project'
33                description: Successful response - returns an array of 'Project' entities.
34            operationId: getprojects
35            summary: List All projects
36            description: Gets a list of all 'Project' entities.
37    '/projects/{projectId}':
38      summary: Path used to manage a single COGITO project/site within which IoT RTLS functionality is expected.
39      description: >-
40        The REST endpoint/path used to get single instances of a 'Project'. This path contains a 'GET'
41        operation used to perform the get task.
42      get:
43        responses:
44          '200':
45            content:
46              application/json:
47                schema:
48                  $ref: '#/components/schemas/project'
49                description: Successful response - returns a single 'Project'.
50            operationId: getProject
51            summary: Get a Project
52            description: Gets the details of a single instance of a 'Project'.
53      parameters:

```

Figure 21: Snapshot depicting part of the rendered YAML file while defining the API

3.4.2 Synchronous Data Communication (Restful API)

The endpoints considered in the REST API implementation convey information regarding:

- The project details, attributes and meta-data
- The tag data
- The tag extra/meta data, sensorial information & functions
- The tag groups
- The tracked zones within a project (IoT-solution specific)

Examples of the conveyed data can be found in section 3.5 below.

projects

GET	/api/v1/projects/	List all projects including basic details	▼
GET	/api/v1/projects/{project_id}	Retrieve all details for a particular project	▼

tagdata

GET	/api/v1/tagdata/	List basic data for all tags	▼
GET	/api/v1/tagdata/{tag_id}	Retrieve the full data for a particular tag	▼

tagmeta

GET	/api/v1/tagmeta/	List all tags extra/meta data & functions	▼
GET	/api/v1/tagmeta/{tag_id}	Retrieve tag extra/meta data & functions for a particular tag	▼

taggroups

GET	/api/v1/taggroups/	List all tag groups	▼
GET	/api/v1/taggroups/{group_id}	Retrieve all details for a particular tag group	▼

trackedzones

GET	/api/v1/trackedzones/	List all tracked zones	▼
GET	/api/v1/trackedzones/{zone_id}	Retrieve all details for a particular tracked zone	▼

3.4.3 Asynchronous Data Communication (Apache Kafka)

Regarding the asynchronous communication via publishing to an Apache Kafka cluster, only the **tagdata** and **tagmeta** information will be streamed grouped by project (serving as the topic name/id).

3.5 Application Example

3.5.1 API response examples

The following are example responses (not necessarily based on actual/real data) returned by the COGITO IoT module API in response to GET requests.

GET /api/v1/projects

```
[
  {
    "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
    "projectTitle": "Example COGITO IoT project",
    "projectLocation": "Address Name and Number, City, Country",
    "startDate": "2022-04-07T17:12:14.880878",
    "trackingType": 0,
    "trackedGroups": 0,
    "noTrackedZones": 1,
    "noTrackedItems": 6,
    "lastActive": "2022-04-20T14:41:53.078",
    "estAccuracy": 0.13316928033665384,
    "coordsOrigin": [
      38.017215285632595,
      23.796327919846476
    ]
  }
]
```



```

    ]
  }
]

```

GET /api/v1/projects/{project_id}

```

{
  "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
  "projectTitle": "Example COGITO IoT project",
  "projectLocation": "Address Name and Number, City, Country",
  "startDate": "2022-04-07T17:12:14.880878",
  "trackingType": 0,
  "trackedGroups": 0,
  "noTrackedZones": 1,
  "noTrackedItems": 6,
  "lastActive": "2022-04-20T14:41:53.078",
  "estAccuracy": 0.13316928033665384,
  "coordsOrigin": [
    38.017215285632595,
    23.796327919846476
  ],
  "groups": [
    {
      "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
      "groupName": "Workers",
      "groupType": 0,
      "tags": [
        {
          "tagId": "b46610b8-6e05-5796-88f5-6925ac2e54a8",
          "tagActive": 1,
          "lastSeen": "2022-04-20T14:41:53.078"
        },
        {
          "tagId": "0176711d-00a6-5dab-be9c-a807e6dace25",
          "tagActive": 1,
          "lastSeen": "2022-04-20T14:41:50.107"
        },
        {
          "tagId": "87844e0c-ec1c-55d3-9214-419e43caa5ad",
          "tagActive": 1,
          "lastSeen": "2022-04-20T14:31:44.609"
        },
        {
          "tagId": "dad21521-ec7a-5321-a60a-5d1963b29012",
          "tagActive": 1,
          "lastSeen": "2022-04-20T11:14:45.864"
        },
        {
          "tagId": "72c1491f-f35c-55f4-8d7f-5d29196944b5",
          "tagActive": 1,
          "lastSeen": "2022-04-20T11:07:14.32509"
        },
        {
          "tagId": "7c189ae9-5cc9-5efe-8217-1ee97d27bebe",
          "tagActive": 1,
          "lastSeen": "2022-04-20T09:31:17.815"
        }
      ]
    }
  ]
}

```


GET /api/v1/tagdata

```
[
  {
    "tagId": "0176711d-00a6-5dab-be9c-a807e6dace25",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:41:50.107",
    "assignedToGroup": {
      "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
      "groupName": "Workers",
      "groupType": 0,
      "assignedToProject": {
        "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
        "projectTitle": "Example COGITO IoT project"
      }
    }
  },
  {
    "tagId": "72c1491f-f35c-55f4-8d7f-5d29196944b5",
    "tagActive": 1,
    "lastSeen": "2022-04-20T11:07:14.32509",
    "assignedToGroup": {
      "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
      "groupName": "Workers",
      "groupType": 0,
      "assignedToProject": {
        "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
        "projectTitle": "Example COGITO IoT project"
      }
    }
  },
  {
    "tagId": "dad21521-ec7a-5321-a60a-5d1963b29012",
    "tagActive": 1,
    "lastSeen": "2022-04-20T11:14:45.864",
    "assignedToGroup": {
      "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
      "groupName": "Workers",
      "groupType": 0,
      "assignedToProject": {
        "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
        "projectTitle": "Example COGITO IoT project"
      }
    }
  },
  {
    "tagId": "7c189ae9-5cc9-5efe-8217-1ee97d27bebe",
    "tagActive": 1,
    "lastSeen": "2022-04-20T09:31:17.815",
    "assignedToGroup": {
      "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
      "groupName": "Workers",
      "groupType": 0,
      "assignedToProject": {
        "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
        "projectTitle": "Example COGITO IoT project"
      }
    }
  },
  {
    "tagId": "87844e0c-ec1c-55d3-9214-419e43caa5ad",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:31:44.609",
    "assignedToGroup": {
```

```

        "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
        "groupName": "Workers",
        "groupType": 0,
        "assignedToProject": {
            "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
            "projectTitle": "Example COGITO IoT project"
        }
    },
    {
        "tagId": "b46610b8-6e05-5796-88f5-6925ac2e54a8",
        "tagActive": 1,
        "lastSeen": "2022-04-20T14:41:53.078",
        "assignedToGroup": {
            "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
            "groupName": "Workers",
            "groupType": 0,
            "assignedToProject": {
                "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
                "projectTitle": "Example COGITO IoT project"
            }
        }
    }
]

```

GET /api/v1/tagdata/{tag_id}

```

{
    "tagId": "0176711d-00a6-5dab-be9c-a807e6dace25",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:41:50.107",
    "assignedToGroup": {
        "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
        "groupName": "Workers",
        "groupType": 0,
        "assignedToProject": {
            "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
            "projectTitle": "Example COGITO IoT project"
        }
    },
    "coords": [
        {
            "coordTimestamp": "2022-04-20T14:41:50.107",
            "coordX": 5.45,
            "coordY": 3.39,
            "coordZ": 1.2,
            "locationRadius": 0.16,
            "inferredAccuracy": 0,
            "inferredMovement": 0,
            "inferredSpeed": 0
        },
        {
            "coordTimestamp": "2022-04-20T14:41:40.088",
            "coordX": 5.45,
            "coordY": 3.39,
            "coordZ": 1.2,
            "locationRadius": 0.16,
            "inferredAccuracy": 0,
            "inferredMovement": 0,
            "inferredSpeed": 0
        }
    ]
}

```

```

        "coordTimestamp": "2022-04-20T14:41:30.075",
        "coordX": 5.45,
        "coordY": 3.39,
        "coordZ": 1.2,
        "locationRadius": 0.16,
        "inferredAccuracy": 0,
        "inferredMovement": 0,
        "inferredSpeed": 0
    },
    {
        "coordTimestamp": "2022-04-20T14:41:20.062",
        "coordX": 5.45,
        "coordY": 3.39,
        "coordZ": 1.2,
        "locationRadius": 0.16,
        "inferredAccuracy": 0,
        "inferredMovement": 0,
        "inferredSpeed": 0
    },
    {
        "coordTimestamp": "2022-04-20T14:41:10.044",
        "coordX": 5.45,
        "coordY": 3.39,
        "coordZ": 1.2,
        "locationRadius": 0.16,
        "inferredAccuracy": 0,
        "inferredMovement": 0,
        "inferredSpeed": 0
    }
  ]
}

```

GET /api/v1/tagmeta

```

[
  {
    "tagId": "b46610b8-6e05-5796-88f5-6925ac2e54a8",
    "acceleration": {
      "accelTimestamp": "2022-04-20T14:41:50.107",
      "accelX": 0.05,
      "accelY": 0.08,
      "accelZ": 0.04
    },
    "buttons": [
      {
        "id": "0",
        "lastPressed": "2022-04-20T14:45:53.078"
      },
      {
        "id": "1",
        "lastPressed": "2022-04-20T14:42:12.055"
      }
    ],
    "sensors": [
      {
        "id": "0",
        "value": "19.5",
        "sensorTimestamp": "2022-04-20T14:42:12.031"
      }
    ],
    "battery": {
      "batteryTimestamp": "2022-04-20T14:41:50.107",

```

```

        "voltage": 3.1,
        "minThreshold": 2.7
    }
}
]

```

GET /api/v1/tagmeta/{tag_id}

```

{
  "tagId": "b46610b8-6e05-5796-88f5-6925ac2e54a8",
  "acceleration": {
    "accelTimestamp": "2022-04-20T14:41:50.107",
    "accelX": 0.05,
    "accelY": 0.08,
    "accelZ": 0.04
  },
  "buttons": [
    {
      "id": "0",
      "lastPressed": "2022-04-20T14:45:53.078"
    },
    {
      "id": "1",
      "lastPressed": "2022-04-20T14:42:12.055"
    }
  ],
  "sensors": [
    {
      "id": "0",
      "value": "19.5",
      "sensorTimestamp": "2022-04-20T14:42:12.031"
    }
  ],
  "battery": {
    "batteryTimestamp": "2022-04-20T14:41:50.107",
    "voltage": 3.1,
    "minThreshold": 2.7
  }
}

```

GET /api/v1/trackedzones

```

[
  {
    "zoneId": "cb136571-02d3-40b0-8856-9ca9472baecf",
    "zoneName": "Main office 1st floor",
    "assignedProjectId": " b305f83a-6674-4ba5-99c1-091e5938b2dc",
    "trackingType": 0,
    "coordSystemAvail": 0,
    "boundaries": [
      [
        0,
        0
      ],
      [
        5.2,
        0
      ],
      [
        5.2,

```

```

        8.6
      ],
      [
        0,
        8.6
      ]
    ]
  }
]

```

GET /api/v1/trackedzones/{zone_id}

```

{
  "zoneId": "cb136571-02d3-40b0-8856-9ca9472baecf",
  "zoneName": "Main office 1st floor",
  "assignedProjectId": " b305f83a-6674-4ba5-99c1-091e5938b2dc",
  "trackingType": 0,
  "coordSystemAvail": 0,
  "boundaries": [
    [
      0,
      0
    ],
    [
      5.2,
      0
    ],
    [
      5.2,
      8.6
    ],
    [
      0,
      8.6
    ]
  ]
}

```

GET /api/v1/taggroups

```

[
  {
    "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
    "groupName": "Workers",
    "groupType": 0,
    "assignedToProject": {
      "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
      "projectTitle": "Example COGITO IoT project"
    }
  }
]

```

GET /api/v1/taggroups/{group_id}

```

{
  "groupId": "7ec6bf7e-14bd-42a4-8cad-9d5d60e789b2",
  "groupName": "Workers",

```

```

"groupType": 0,
"assignedToProject": {
  "projectId": "b305f83a-6674-4ba5-99c1-091e5938b2dc",
  "projectTitle": "Example COGITO IoT project",
  "projectLocation": "Address Name and Number, City, Country",
  "startDate": "2022-04-07T17:12:14.880878",
  "trackingType": 0,
  "coordSystem": 0,
  "trackedGroups": 0
},
"tags": [
  {
    "tagId": "b46610b8-6e05-5796-88f5-6925ac2e54a8",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:41:53.078"
  },
  {
    "tagId": "0176711d-00a6-5dab-be9c-a807e6dace25",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:41:50.107"
  },
  {
    "tagId": "87844e0c-ec1c-55d3-9214-419e43caa5ad",
    "tagActive": 1,
    "lastSeen": "2022-04-20T14:31:44.609"
  },
  {
    "tagId": "dad21521-ec7a-5321-a60a-5d1963b29012",
    "tagActive": 1,
    "lastSeen": "2022-04-20T11:14:45.864"
  },
  {
    "tagId": "72c1491f-f35c-55f4-8d7f-5d29196944b5",
    "tagActive": 1,
    "lastSeen": "2022-04-20T11:07:14.32509"
  },
  {
    "tagId": "7c189ae9-5cc9-5efe-8217-1ee97d27bebe",
    "tagActive": 1,
    "lastSeen": "2022-04-20T09:31:17.815"
  }
]
}

```

As already discussed in Section 3.3, the processed location data are published to a Kafka topic to ensure minimal latency; the data are provided in JSON format as illustrated above for the REST API. Nevertheless, current as well as archived data is also available via the prescribed REST API endpoints. System information-reporting and possibly the execution of remote commands affecting particular tags or tag groups is natively supported and could be available via REST API should it be officially required.

3.5.2 Screenshots of the Module Status Monitoring & Alerting Dashboards

As described in Section 3.1, a new feature of the IoT Data Pre-processing Module is the “Module Status Monitoring & Alerting subcomponent”, responsible for monitoring the status and providing analytics in the form of customisable dashboards. Snapshots of the dashboards can be found in Figure 22 to Figure 26.

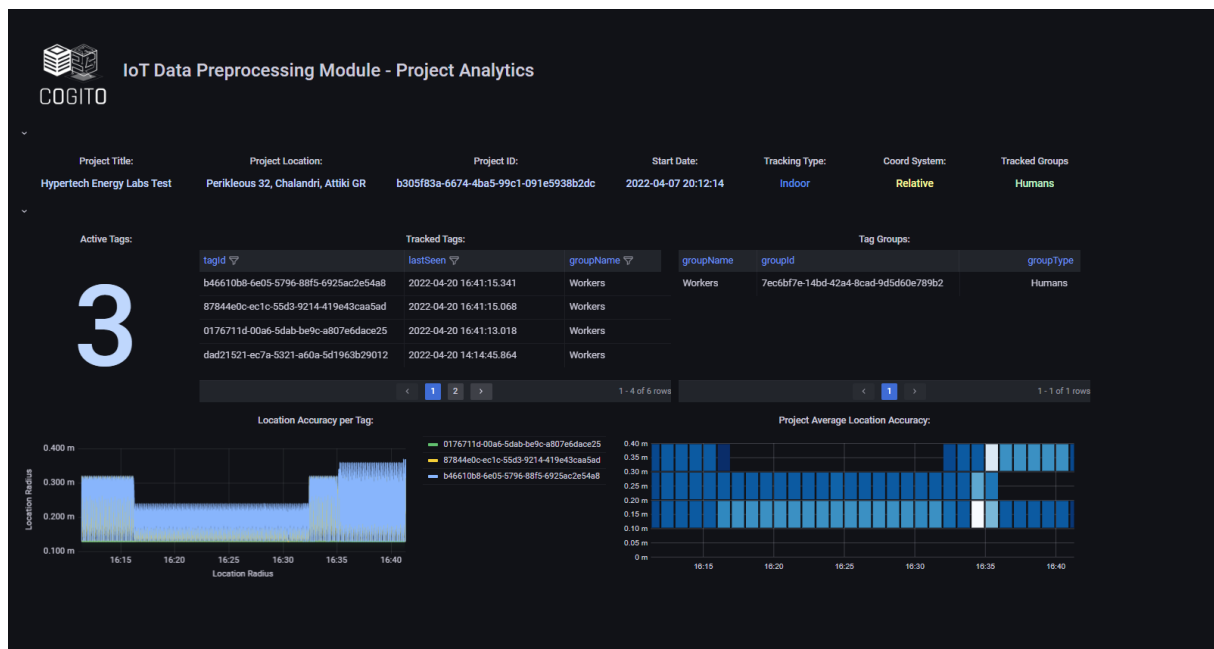


Figure 22: Dashboard for the RTLS backend – status & project analytics



Figure 23: Dashboard for the Core application – streaming processing pipeline status

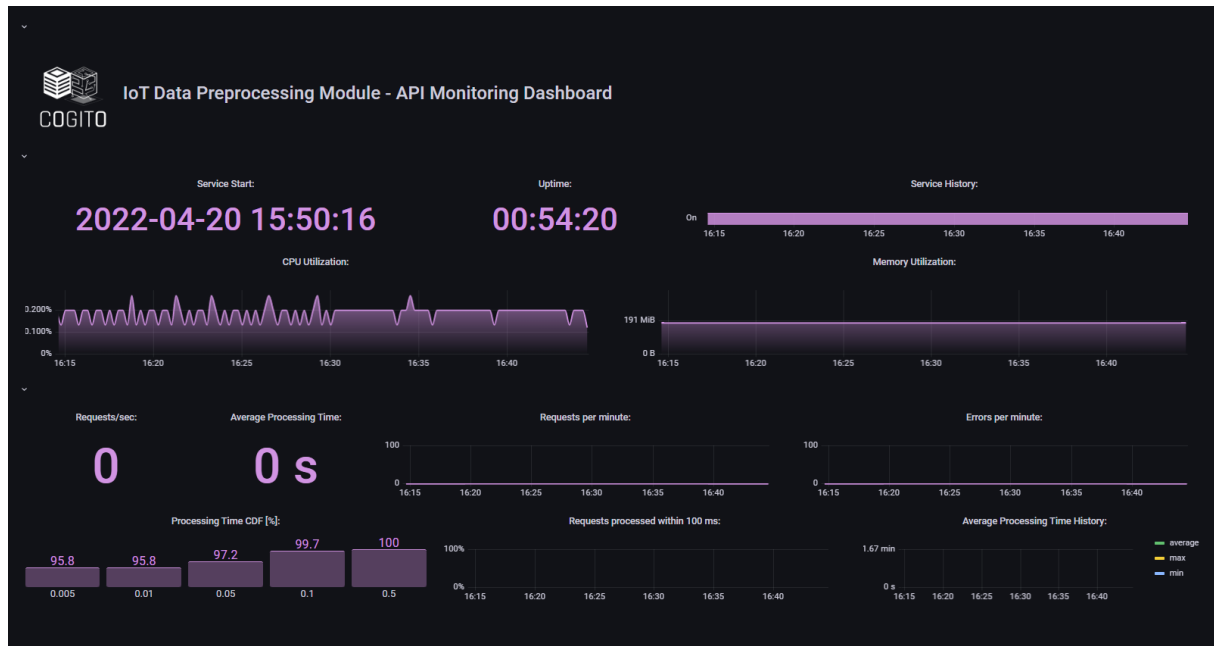


Figure 24: Dashboard for the API status

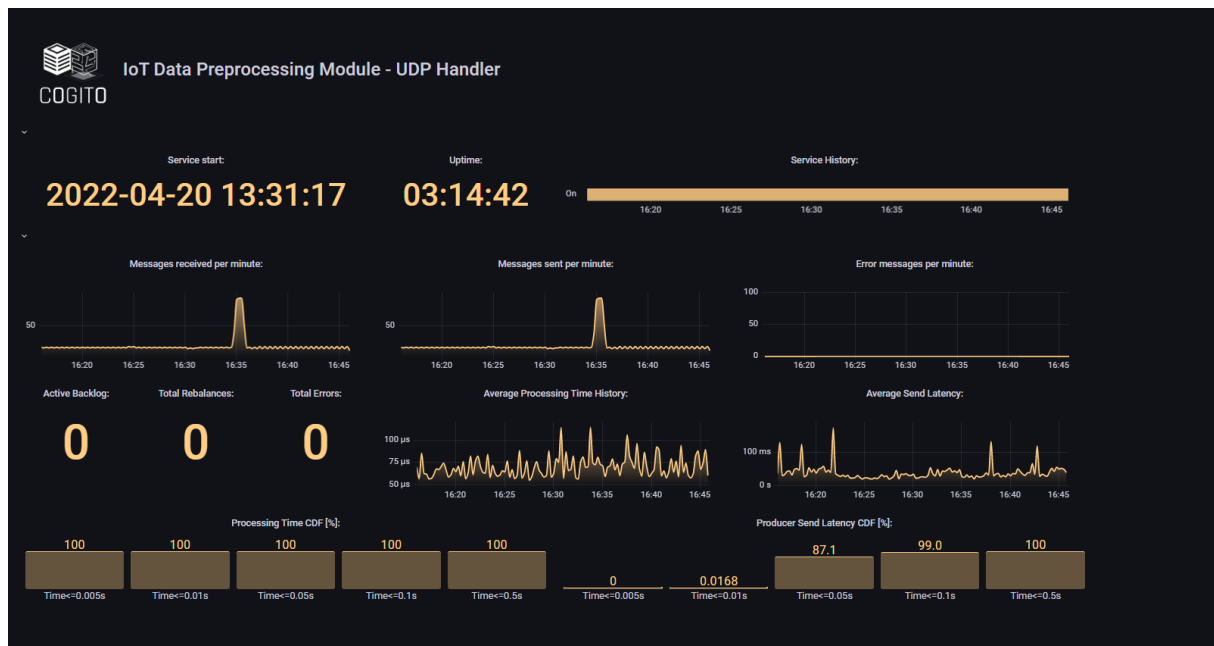


Figure 25: Dashboard for the UDP handler (input middleware) status

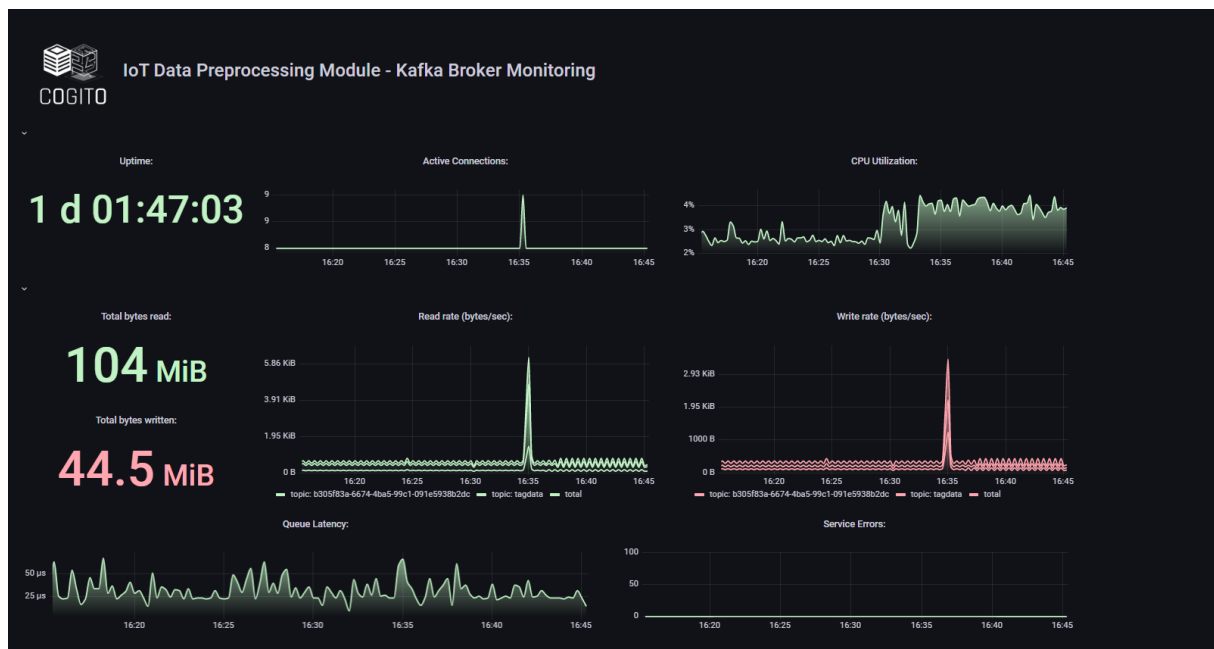


Figure 26: Dashboard for the Kafka broker status

3.6 Licensing

The IoT Data Pre-processing module is offered in the form of an open-source Software as a Service (SaaS) component.

3.7 Installation Instructions

The IoT Data Pre-processing module is offered to the COGITO platform as a web service; as such, no file download, installation, maintenance or other related operation is to be performed by entities other than its creators.

3.8 Development and integration status

As already explained in the previous sections of this deliverable, the COGITO IoT solution comprises of two main parts, the IoT RTLS backend(s) and the actual IoT Data Pre-processing Module responsible for ingesting the location data, processing them and delivering them to the Digital Twin Platform (DTP). After reviewing the potential options available, an IoT RTLS backend was selected (*Quuppa*), procured and installed locally; its performance and configuration options have been evaluated using the small scale (yet fully-fledged) deployment at the Hypertech's offices.

Regarding the pre-processing module itself, after bilateral discussions with the affected consortium parties, the baseline functionality, architecture and input/output formats have been defined.

The development of the subcomponents comprising the IoT module has reached a major milestone, currently providing all the required functionality; the finalisation of the development, including any refinements should run in parallel to the testing/pre-validation phase, namely during the task T8.1. After this phase, the module should be ready for integration and delivery.

3.9 Requirements Coverage

The architecture and implementation address all functional and non-functional requirements as defined in the deliverables D2.4 "COGITO System Architecture v1" and D2.5 "COGITO System Architecture v2".

3.9.1 Functional Requirements

- **Data Management**

The data from any IoT RTLS backend, including but not limited to the selected one (*Quuppa*) is retrieved, ingested, homogenized cleaned and potentially pre-processed, all in real- or near real-time before publishing it to the DTP. In the case of multiple available location backends (e.g., one for indoor and one for outdoor environments), the input data is fused and provided at the output in a consistent, transparent manner. The output data can either be published to a Kafka topic where the DTP can subscribe for updates or they can be requested (polled) using REST API endpoints; similarly, the input data can either be provided using a UDP data stream, by means of a REST API or using an MQTT-like protocol.

Regarding the provided output data, all requirements have been met, namely the following:

- A sensor measures the location
- A location includes: altitude, longitude and latitude
- A location also includes the time of the measure
- A location includes the accuracy of the measure

- **Archiving & Backup**

The DTP is responsible for storing the data provided to it by the IoT module; however, the transmitted data is going to be persisted also locally using a timeseries-oriented DB such as TimescaleDB, InfluxDB or similar. Such a DB is going to act as the “source of truth” regarding the location data and will allow for meta-analysis to investigate the performance of the system besides being a backup of the collected data.

- **Authorisation and encryption, pseudo-anonymisation**

The IoT Data Pre-processing module is not supposed to directly communicate with any other COGITO module or end-user except for the DTP; the DTP is responsible for delivering the location information to other modules and components using its local copy of data. As such, there is no need for role-based authorisation. Client authentication (i.e., DTP authentication) is accomplished using mTLS and encryption using TLS regarding both the REST API endpoints (over HTTPS) and the Kafka broker-client connection. Regarding the requirement for information pseudo-anonymisation, since the collected data only refer to tag identifiers (tag UUIDs), there is no direct connection with a particular person or object in the context of the IoT pre-processing module; this kind of semantics is linked within the DTP which is responsible for maintaining the integrity and anonymity of the collected data.

3.9.2 Non-functional Requirements

- **Interoperability**

Since the module essentially serves as a middleware between potential IoT backends and the DTP, interoperability is of paramount importance. The implementation covers the most-commonly used input/output formats ensuring out-of-the-box compliance, while there are also provisions for further data formats and communication schemes accommodating a wide range of available commercial RTLS technologies.

- **Security**

As already stressed, the IoT module does not collect or process any sensitive data; all collected data are anonymised and cannot be linked to a particular person or object within the scope of the IoT Data Pre-processing component. Nevertheless, all provisions and countermeasures have been taken in the architecture and the implementation to ensure maximum security and data integrity regarding all involved layers, namely the data ingestion, processing, storage and communication processes.

- **Performance**

The component’s implementation is accomplished using state-of-the-art software frameworks and libraries, ensuring maximum performance in terms of processing speed and resource usage.

- **Scalability**

The architecture and technologies used in the component inherently allow for both vertical as well as horizontal scalability should it be required.

3.10 Assumptions and Restrictions

Concerning the accuracy and reliability of the location/sensorial data provided, it should be noted that the impact of the indoor environment can be significant; the signal strength can be affected by many propagation effects such as absorption, scattering, reflection, multi-path etc. Also, in general, range and reliable signal reception are strongly interrelated. Careful planning and possibly simulation before any actual deployment is thus required.

Regarding the deployment of the IoT solution, practical matters tightly related to the specificities of a particular construction site such as the mounting options offered, available clearance for cable runs, power supply, internet connectivity should be considered.

In very challenging or mixed (indoor/outdoor) environments, it is highly likely that a combination of RTLS techniques might be required; such a use case is well accommodated within the context of this module and shall pose no considerable complications.

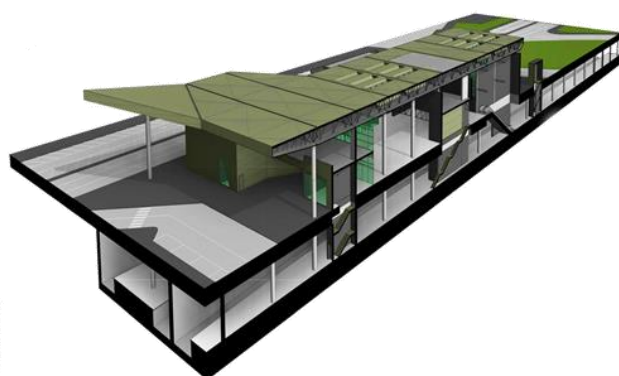
4 Conclusions

In this deliverable, the scope, the conceptual design as well as the implementation details for the COGITO's IoT Data Pre-processing Module were provided; a brief overview of the RTLS techniques and technologies available to be used as the solution's backend was also incorporated. A small-scale, yet fully-featured deployment of the selected RTLS backend at the Hypertech premises was presented; after several weeks of familiarisation, experimentation and fine-tuning, it was possible to evaluate the performance and identify possible issues or challenges. The module's development has reached a major milestone, thereby currently providing all the required functionality; in any case, any refinements or modifications required to integrate it with the rest of the COGITO ecosystem are expected to be completed during T8.1.

References

- [1] "Real time localization of assets in hospitals using quuppa indoor positioning technology," in *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Split, Croatia, 2016.
- [2] M. N. Kamel Boulos and G. Berry, "Real-time locating systems (RTLS) in healthcare: a condensed primer," *International Journal of Health Geographics*, vol. 11, no. 1, p. 25, 2012.
- [3] R. Mautz, *Indoor Positioning Technologies*, Zurich, Switzerland: Geodätisch-geophysikalische, 2012.
- [4] B. Gladysz and K. Santarek, "An Approach to RTLS selection," in *24th International Conference on Production Research (ICPR 2017)*, Poland, 2017.
- [5] F. Zafari, A. Gkelias and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568-2599, 2019.
- [6] D. Dardari, P. Closas and P. M. Djurić, "Indoor Tracking: Theory, Methods, and Technologies," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1263-1278, 2015.
- [7] H. Liu, H. Darabi, P. Banerjee and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067-1080, 2007.
- [8] R. Zhan, Z. Farid, R. Nordin and M. Ismail, "Recent Advances in Wireless Indoor Localization Techniques and System," *Journal of Computer Networks and Communications*, vol. 2013, 2013.
- [9] D. Stojanovic and N. Stojanovic, "Indoor Localization and Tracking: Methods, Technologies and Research Challenges," *Series: Automatic Control and Robotics, Facta Universitatis*, vol. 13, pp. 57-72, 2014.
- [10] R. Amsters, A. B. Junaid, S. Roy, P. Slaets, P. Aerts, M. Verheyen and E. Demeester, "A Literature Study of Indoor Positioning," KU Leuven, 2017.
- [11] R. Mautz, "Overview of current indoor positioning systems," *Geodezija ir Kartografija*, vol. 35, no. 1, pp. 18-22, 2009.
- [12] T. K. Sarkar, Z. Ji, K. Kim, A. Medouri and M. Salazar-Palma, "A survey of various propagation models for mobile communication," *IEEE Antennas and Propagation Magazine*, vol. 45, no. 3, pp. 51-82, 2003.
- [13] A. Mahtab Hossain and W.-S. Soh, "A survey of calibration-free indoor positioning systems," *Computer Communications*, vol. 66, pp. 1-13, 2015.
- [14] "Quuppa RTLS Technology Overview," [Online]. Available: <https://www.quuppa.com/overview/>. [Accessed 15 02 2022].
- [15] "Quuppa Continues to Deliver Industry's Most Comprehensive Location Solutions," [Online]. Available: <https://www.quuppa.com/quuppa-continues-to-deliver-industrys-most-comprehensive-location-solutions/>. [Accessed 18 02 2022].
- [16] "Direction Finding nWP-036," August 2020. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nwp_036.pdf. [Accessed 15 2 2022].

- [17] “Quuppa Development Kit Content,” [Online]. Available: https://quuppa.com/product-documentation/manuals/q/devkit_qsg/topics/DevkitQSG_included.html. [Accessed 15 02 2022].
- [18] “Blueup Quuppa SafeX,” [Online]. Available: https://www.blueupbeacons.com/index.php?page=safex_quuppa. [Accessed 11 04 2022].
- [19] “Onsemi SECO-RSL10-TAG-GEVB: RSL10 Asset Tag,” [Online]. Available: <https://www.onsemi.com/design/tools-software/evaluation-board/seco-rsl10-tag-gevb>. [Accessed 11 04 2022].
- [20] “OpenAPI Specification v3.1.0,” [Online]. Available: <https://spec.openapis.org/oas/v3.1.0>. [Accessed 18 02 2022].
- [21] “Apicurio Studio,” [Online]. Available: <https://www.apicur.io/studio/>.
- [22] “Swagger,” [Online]. Available: <https://swagger.io>.



COGITO

CONSTRUCTION PHASE
DIGITAL TWIN MODEL

cogito-project.eu



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310