

NOVAL

## D4.4

# Enhanced IoT Tactile & Contextual Sensing/Actuating (Final Version)

**WORKPACKAGE** WP4

**DOCUMENT** D4.4

**REVISION** V1.0

**DELIVERY DATE** 30/04/2023

**PROGRAMME IDENTIFIER** H2020-ICT-2020-1

**GRANT AGREEMENT ID** 957246

**START DATE OF THE PROJECT** 01/10/2020

**DURATION** 3 YEARS

© Copyright by the IoT-NGIN Consortium

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



## DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties, and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered using this information.

## ACKNOWLEDGEMENT

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

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

<b>PROJECT ACRONYM</b>	IoT-NGIN
<b>PROJECT TITLE</b>	Next Generation IoT as part of Next Generation Internet
<b>CALL ID</b>	H2020-ICT-2020-1
<b>CALL NAME</b>	Information and Communication Technologies
<b>TOPIC</b>	ICT-56-2020 - Next Generation Internet of Things
<b>TYPE OF ACTION</b>	Research and Innovation Action
<b>COORDINATOR</b>	Capgemini Technology Services (CAP)
<b>PRINCIPAL CONTRACTORS</b>	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), NETCOMPANY-INTRASOFT SA (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Robert Bosch Espana Fabrica Aranjuez SA (BOSCHN), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), ENTERSOFT SA (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
<b>WORKPACKAGE</b>	WP4
<b>DELIVERABLE TYPE</b>	REPORT
<b>DISSEMINATION LEVEL</b>	PUBLIC
<b>DELIVERABLE STATE</b>	FINAL
<b>CONTRACTUAL DATE OF DELIVERY</b>	30/04/2023
<b>ACTUAL DATE OF DELIVERY</b>	26/05/2023
<b>DOCUMENT TITLE</b>	Enhanced IoT Tactile & Contextual Sensing/Actuating (Final Version)
<b>AUTHOR(S)</b>	M. Montagud (I2CAT), M. Hjej (I2CAT), J. Escrig (I2CAT), M. Catalan (I2CAT), A. Gonos (OPT), A. Voulkidis (SYN), Ch. Betzelos (SYN), M. Sophocleous (EBOS), A. Sabatino (ENG), A. Corsi (ENG), Aukusti Kankaanpaa (ABB)
<b>REVIEWER(S)</b>	T. Velivassaki (SYN), F. Bellesini (EMOT)
<b>ABSTRACT</b>	SEE EXECUTIVE SUMMARY
<b>HISTORY</b>	SEE DOCUMENT HISTORY
<b>KEYWORDS</b>	Augmented Reality, Ambient Intelligence, Tactile internet, device discovery, access control

## Document History

Version	Date	Contributor(s)	Description
V0.1	01/02/2023	i2CAT	Table of Contents
V0.2	08/03/2023	i2CAT	Updates on ToC
V0.3	25/04/2023	EBOS, SYN	Inputs from EBOS and SYN
V0.4	16/05/2023	I2CAT, ENG	Inputs from i2CAT and ENG
V0.5	18/05/2023	ABB, I2CAT	Inputs from ABB and version ready for review
V0.5.1	19/05/2023	EMOT	Peer review
V0.5.2	22/05/2023	SYN	Peer review
V0.6	26/05/2023	I2CAT	Document finalization
V0.7	26/05/2023	CAP	Quality Check
V1.0	26/05/2023	I2CAT	Final version

# Table of Contents

Document History .....	4
Table of Contents .....	5
List of Figures.....	7
List of Tables.....	10
List of Acronyms and Abbreviations.....	11
Executive Summary .....	12
1 Introduction.....	13
1.1 Intended Audience.....	13
1.2 Relations to other activities .....	14
1.3 Relations to previous deliverables in WP4 .....	15
1.4 Document overview .....	17
2 State-of-the-Art on Ambient Intelligence via IoT-AR .....	19
3 Ambient Intelligence in IoT-NGIN .....	22
3.1 Converging to a Digital Twin solution .....	22
3.1.1 IoT Device Indexing (IDI) .....	23
3.1.2 IoT Device Access Control (IDAC) .....	24
3.2 IoT-AR Modules.....	25
4 Implementation for the IoT-NGIN Living Labs .....	30
4.1 IoT Device Discovery (IDD) .....	30
4.1.1 AI-based Object Recognition for the Smart Agriculture & Smart Energy UCs.....	30
General Info / Requirements / Installation .....	30
4.1.2 Industry 4.0 .....	37
4.2 IoT Device indexing (IDI) .....	38
4.3 IoT Devices access control (IDAC) .....	38
4.4 AR modules.....	38
4.4.1 Industry 4.0: BOSCH Factory .....	38
4.4.2 Industry 4.0: ABB Factory .....	44
4.4.3 Smart Agriculture/Energy .....	47
5 Installation and user guidelines .....	54
5.1 IoT Device Discovery (Computer Vision) .....	54
5.1.1 Installation guidelines .....	54
5.1.2 User guidelines .....	57
5.2 IoT Device Discovery (UWB) .....	59
5.3 IoT Device Discovery (VLP) .....	59

5.4	IoT Device indexing .....	59
5.5	IoT Devices access control.....	59
5.6	AR modules / apps .....	60
5.6.1	Industry 4.0 (BOSCH Living Lab) .....	60
5.6.2	Industry 4.0 (ABB Living Lab).....	61
5.6.3	Smart Agriculture/Energy .....	61
6	Conclusions.....	80
7	References .....	81
Annex 1	Public Repo with IoT-AR Resources .....	84

DRAFT - PENDING EC APPROVAL

## List of Figures

Figure 1. Work packages structure.....	14
Figure 2. Overview of output and input modalities to interact with ICT systems and services .....	19
Figure 3. Example of visual representation methods assessed in [10]: (a) virtual text with semi-transparent layer; (b) virtual cube with color to visualize the IoT information .....	20
Figure 4. IoT Device Indexing – functional description .....	24
Figure 5. High-level architecture of the IDAC module. ....	25
Figure 6. High-level sequence diagrams for AR presentation and actuation cases.....	26
Figure 7. AR module and interfaces for #UC10.....	27
Figure 8. Sequence diagram of AR module interactions for #UC10.....	28
Figure 9. EfficientDet architecture.....	31
Figure 10. Directory Tree of workspace .....	32
Figure 11. Samples of the training set for the Synelxis SynField device .....	33
Figure 12. Samples of the training set for emotion Charging Station.....	33
Figure 13. Labelling Interface.....	34
Figure 14. The label of the object.....	35
Figure 15. Results from model training .....	36
Figure 16. Metrics to characterize the performance of the trained model.....	36
Figure 17. Model testing.....	37
Figure 18. High-level architecture of the developed AR app for Smart Industry #UC6 .....	39
Figure 19. Detection of visual targets and superimposing related information, using Vuforia .....	40
Figure 20. Request of Internet privileges from the Magic Leap .....	41
Figure 21. Communication with IDI module from Unity .....	42
Figure 22. AR information displayed for the #UC6 (Bosch factory) .....	42
Figure 23. Alarm system to inform about near distances between sensors .....	43
Figure 24. Simplified architecture schematic of the #UC7 AR application.....	44
Figure 25. Code for decoding the hyperlink from QR code .....	45
Figure 26. Code for populating interaction buttons.....	46
Figure 27. Example of highlighted component and component tag on a digital cabinet model .....	47
Figure 28. Call Service MLaaS of the VM.....	47
Figure 29. Anchor Box e % Discovery .....	48
Figure 30. AR on QRCode.....	48
Figure 31. IDI: Get a token from Keycloak .....	49

Figure 32. Test Historical Data on IDI: requests "Get last n records/ID Device" (n=2) .....	49
Figure 33. AR on QRCode (with the values of the sensors of the SynField node) .....	49
Figure 34. Software of the AR app for Smart Energy UC .....	52
Figure 35. AR on QRCode.....	52
Figure 36. Virtual button .....	53
Figure 37. AR on QRCode.....	53
Figure 38. Requests Device Index N=2 (with the values of the sensors of the SynField node).....	53
Figure 39. Output of the nvidia-smi command where the supported CUDA version is displayed .....	55
Figure 40. Terminal output confirming the creation of a service, device and subscription ...	57
Figure 41. Terminal output after executing the Computer Vision Device Discovery demo ...	58
Figure 42. Unity Download.....	62
Figure 43. Unity Installs .....	63
Figure 44. Create a New Unity project (I) .....	63
Figure 45. Create a New Unity project (II) .....	63
Figure 46. Unity project created .....	64
Figure 47. Save Unity project.....	64
Figure 48. Download Vuforia Engine.....	65
Figure 49. Login and Agree the Terms (Vuforia).....	65
Figure 50. Software License (Vuforia).....	65
Figure 51. Import Vuforia SDK into Unity.....	66
Figure 52. Example image of the SynField device .....	66
Figure 53. Example image of EV Charging Station .....	67
Figure 54. Target Manager Tab in Vuforia.....	67
Figure 55. Create Database on Vuforia .....	67
Figure 56. Add Target in Vuforia .....	68
Figure 57. Setting Target parameters in Vuforia .....	68
Figure 58. Download Vuforia Database to Unity Editor.....	69
Figure 59. Compiling Vuforia database .....	69
Figure 60. Import package to Unity .....	69
Figure 61. Delete main camera on Unity .....	70
Figure 62. Add AR Camera from Vuforia Engine .....	70
Figure 63. Add image target from Vuforia Engine.....	71
Figure 64. Load Image from Database .....	71
Figure 65. Unity screen after having loaded Image from Database .....	72
Figure 66. Add a button from UI .....	72



Figure 67. Change Render Mode to World Space .....	72
Figure 68. Set Event Camera as AR camera .....	73
Figure 69. Reset the canvas position.....	73
Figure 70. AR on QRCode.....	74
Figure 71. Create a new C# file .....	74
Figure 72. Link a script to an AR image .....	75
Figure 73. Unity Components for the Project .....	76
Figure 74. Build Settings.....	76
Figure 75. Choosing Android in the Build Settings.....	77
Figure 76. Setting Player Parameters / Info .....	77
Figure 77. Creating a Build .....	78
Figure 78. .apk created .....	78
Figure 79. .apk installed and running .....	79

DRAFT - PENDING EC APPROVAL

## List of Tables

Table 1. Relation of WP4 activities to other WPs and tasks.....	14
Table 2. Relation of D4.4 compared to previous WP4 deliverables (D4.2 and D4.3) .....	15
Table 3. APIs defined and used by the AR module for the #UC10 .....	29
Table 4. Discovery methods and objects to be detects in the IoT-NGIN Living Labs .....	30

DRAFT - PENDING EC APPROVAL

## List of Acronyms and Abbreviations

AGV	Autonomous Guided Vehicles
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
BiFPN	Bi-directional Feature Pyramid Network
CV	Computer Vision
DT	Digital Twin
EV	Electric Vehicle
GUI	Graphical User Interface
IDD	IoT Device Discovery
IDAC	IoT Device Access Control
IDI	IoT Device Indexing
IAM	Identity and Access Management
IoT	Internet of Things
GPS	Global Positioning System
ML	Machine Learning
MlaaS	Machine Learning as a Service
MQTT	Message Queuing Telemetry Transport
NGSiv2	Next Generation Service Interfaces v2
OS	Operating System
SoTA	State-of-the-art
SSI	Self Sovereign Identities
TPU	Tensor Processing Units
UC	Use Case
UI	User Interface
UWB	Ultra Wide Band
VLP	Visual Light Positioning
VR	Virtual Reality
WP	Work Package

## Executive Summary

Ambient Intelligence will be an essential part of the next-generation Internet of Things (IoT). IoT-NGIN aims to provide a set of components, modules, and methods to enable Ambient Intelligence and Contextual IoT Sensing / Actuating in a set of relevant use cases and living labs, assisted with Augmented Reality (AR) presentation and interaction methods and tools.

This deliverable reports on the contributions of IoT-NGIN in that direction, within the scope of its Work Package 4 (WP4) "Enhancing IoT Tactile & Contextual Sensing/Actuating". In particular, the main outcomes of this deliverable include:

- A comprehensive state-of-the-art review on Ambient Intelligence via IoT-AR.
- An update of the technical specifications and interfaces for the final versions of four key components developed within WP4 to enable the envisioned use cases, namely: IoT Device Discovery, IoT Device Indexing, IoT Device Access Control, and IoT presentation and interaction.
- Final implementation details for the above-mentioned components and tools, for each foreseen Living Lab and Use Case.
- Installation and user guidelines of the tools for the above-mentioned components and tools, and links to their open-source code on GitLab.

In addition, a set of open access IoT-AR resources, both developed within the project, as well as third-party ones, have been gathered to let the interested audience build their targeted AR app or tool, allowing rich interaction with a diverse set of multi-modal IoT environments.

This deliverable reports on the final version of the technological contributions from WP4, so its content can be more comprehensively comprehended if read in conjunction with its predecessor deliverables (D4.2 and D4.3), which reported on the requirements for each component and use cases, initial and updated designs for all envisioned tools / components, as well as the developments of specific components, like IoT Device Discovery, IoT Device Indexing, and IoT Device Access Control.

# 1 Introduction

Ambient Intelligence will be an essential part of the next-generation Internet of Things (IoT). IoT-NGIN aims to provide a set of components, modules and methods to enable Ambient Intelligence IoT in a set of relevant use cases and living labs, assisted with Augmented Reality (AR) presentation and interaction methods and tools. This deliverable reports on the contributions of IoT-NGIN in that direction, within the scope of its Work Package 4 (WP4) "Enhancing IoT Tactile & Contextual Sensing/Actuating". The main outcomes reported in this deliverable include:

- Comprehensive state-of-the-art review on Ambient Intelligence via IoT-AR.
- Updated technical specifications and interfaces of the final versions of the four main components developed within WP4 to enable the envisioned use cases, namely:
  - IoT Device Discovery (IDD), which is responsible for recognition and/or positioning of IoT Devices, with a key focus on Computer-Vision based image recognition in this deliverable.
  - IoT Device Indexing (IDI), which manages device-related information and supports Digital Twin functionality.
  - IoT Device Access Control (IDAC), which intervenes communications among IoT-NGIN components and devices and provides configurable multi-criteria access control.
  - IoT-AR, which is responsible for presentation of, and interaction with, the IoT sensors / devices.
- Final implementation details for the above-mentioned components and tools, for each foreseen use case.
- Installation and user guidelines of the tools for the above-mentioned components and tools, for each foreseen use case, and links to their open-source code on GitLab.

In addition, a set of open access IoT-AR resources, both developed within the project and third-party ones, have been gathered to let the interested audience build their targeted AR app or tool, allowing rich interaction with a diverse set of multi-modal IoT environments. All such resources are collected in Annex 1 and on a Wiki at the project's Gitlab [1].

This deliverable reports on the final version of the technological contributions from WP4, so its content can be more comprehensively comprehended if read in conjunction with its predecessor D4.2 [2] and D4.3 [3] deliverables, which reported on the requirements for each component and use cases, initial designs for all envisioned tools / components, as well as developments of specific components, like IDD, IDI and IDAC.

## 1.1 Intended Audience

The target audience of this deliverable includes mainly IoT system providers, engineers and AR developers and/or technical staff, who would be interested in developing ambient intelligence and contextual sensing / actuating in IoT systems. Through this report, in combination with D4.2 and D4.3, they may identify the relevant ambient intelligence requirements in application domains covered by the IoT-NGIN Living Labs and Use Cases (UCs), and find the technical specifications and concrete implementation for all associated components related to IDD, IDI, IDAC and IoT-AR presentation and interaction.

Furthermore, most of the developed tools and components are publicly shared, so they can be downloaded, tested and fine-tuned based on the specific purposes.

Moreover, the report is useful internally, to the members of the development and integration team of the IoT-NGIN consortium, involving mainly WP3-WP6 partners, the Living Lab teams from WP7, but also to the whole Consortium and third parties, especially those participating via Open Calls, for integration, validation and exploitation purposes. Useful feedback could be also received from the Advisory Board, including both technical and impact creation comments.

## 1.2 Relations to other activities

The activities of WP4 are strongly connected to the rest IoT-NGIN activities, as sketched in

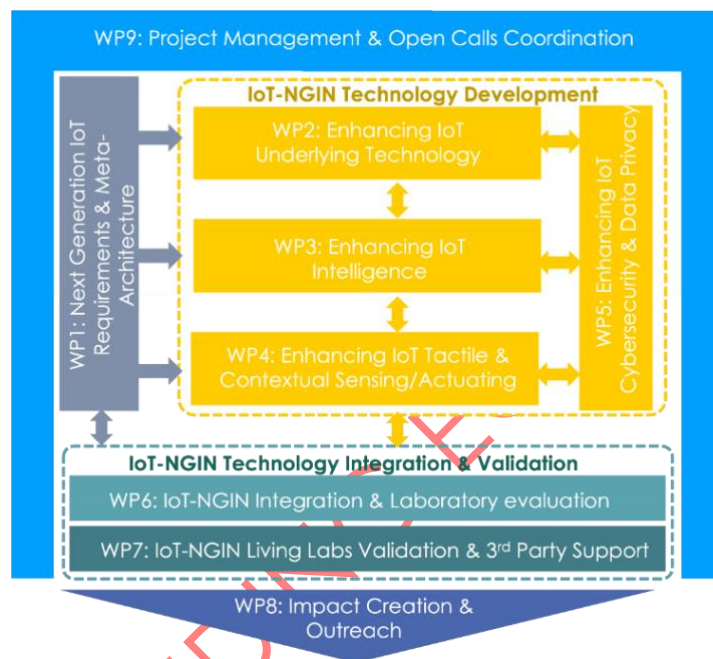


Figure 1. Work packages structure

Table 1 describes these relationships with further details.

Table 1. Relation of WP4 activities to other WPs and tasks

WP	Relation to WP4 and D4.3
WP1	The definition of the Living Labs' UCs and elicitation of user/system requirements has served D4.3 and D4.4 for the (updated) definition of the IoT Ambient Intelligence components implemented in tasks T4.2, T4.3 and T4.4. Also, the role of the tools and modules deployed in WP4 are represented in the meta-architecture resulting by the activities in WP1.
WP2	No direct interaction is foreseen among WP4 and WP2. WP2 provides communication enhancements which could support the communication requirements for tactile communications in next-generation IoT.

<b>WP3</b>	The training and sharing/serving of Artificial Intelligence (AI) models of the IoT Device Discovery components can be conducted via the Machine Learning as a Service (MLaaS) platform developed in WP3. Moreover, deployment of AI models into physical devices may be passed via the Digital Twin functionality of the IDI module.
<b>WP5</b>	The IDI and the IDAC modules deployed in WP4 interact with different components developed within the scope of WP5, such as the IoT Vulnerability Crawler. Moreover, IDAC integrates the Self-Sovereign Identities module.
<b>WP6</b>	WP4 components will be integrated with the rest of project's technologies and frameworks within WP6, while some of the WP4 components may be used in the development of application logic for the use cases or the Open Calls.
<b>WP7</b>	WP4 components have been implemented and used in several Living Labs and Use Cases (UCs), as reported in D4.3.  WP4 supports 3 <sup>rd</sup> parties by offering a specific set of functionalities (such as the IoT-AR repository).
<b>WP8</b>	WP4 provides notable outcomes and results for supporting impact creation activities. Moreover, it considers feedback (e.g., from the market analysis and business modelling tasks), which could be relevant for the WP4 design and development.

### 1.3 Relations to previous deliverables in WP4

Deliverables in WP4 are somehow iterative, although they have been planned to minimize the amount of redundance, just reporting on new or evolved aspects and/or contribution in every new recent deliverable. Table 2 summarizes the relation of D4.4 compared to previous WP4 deliverables, in particular D4.2 [2] and D4.3 [3].

Table 2. Relation of D4.4 compared to previous WP4 deliverables (D4.2 and D4.3)

## D4.4 - Enhanced IoT Tactile &amp; Contextual Sensing/Actuating (Final Version)

		D4.2	D4.3		D4.4	
State-of-the-art (SoTA) on Ambient Intelligence and Tactile IoT		Object recognition Object positioning Device authentication 5G Edge computing	Brief SoTA analysis on AR		In-depth review of IoT-AR	
Requirements from UCs		Initial requirements User stories	Update of the requirements		Already covered in D4.3	
WP4 architecture		Initial description	Updated description		Already covered in D4.3	
WP4 Modules						
	IoT Device Discovery	Initial description Initial interfaces	Updated description		Updates on the implementation of interfaces	
	IoT Device Indexing	Initial description Initial interfaces	Updated description Interfaces update		Already covered in D4.3	
	IoT Device Access Control	Initial description Initial interfaces	Updated description Interfaces update		Already covered in D4.3	
	AR/MR	Initial description Initial interfaces	Updated description		Updated description Updated Interfaces	
Implementation in the LL						
	Smart Agriculture		IDD	Presented	IDD	Updated
			IDI	Presented	IDI	Completed in D4.3
			IDAC	Presented	IDAC	
			AR	Initial	AR	Completed
	Industry 4.0		IDD	Initial	IDD	Completed
			IDI	Not started	IDI	Completed



Smart Energy		IDAC	Not started	IDAC	Completed
		AR	Initial	AR	Completed
		IDD	Initial	IDD	Completed
		IDI	Initial	IDI	Completed
		IDAC	Initial	IDAC	Completed
Smart City	-	AR	Not started	AR	Completed
		(IDI & IDAC use presented in WP7)		(IDI & IDAC use presented in WP7)	
Installation guidelines		IDD (Computer Vision, CV)	Not started	IDD (CV)	Completed
		IDD (Ultra Wideband( UWB))	Presented	IDD (UWB)	Completed in D4.3
		IDD (Visible Light Positioning, VLP)	Presented	IDD (VLP)	Completed in D4.3
		IDI	Presented	IDI	Completed in D4.3
		IDAC	Presented	IDAC	Completed in D4.3
		AR	Not started	AR	Completed

## 1.4 Document overview

The rest of the document is organized as follows.

Section 2 provides a comprehensive state-of-the-art review on Ambient Intelligence and Contextual Sensing and Actuating via IoT and AR technologies, as a motivation of the work conducted under the umbrella of IoT-NGIN, and its WP4 in particular.

Section 3 provides a short summary on the functionality of IDI and IDAC as tools towards developing digital twins, and reports updates regarding the specifications of the interfaces and APIs for IoT-AR components/modules, compared to the ones reported and described in D4.3 [3].

Section 4 reports the implementation of the AI-based object recognition methods specifically for the Smart Agriculture use case and of all associated AR apps and methods for the Smart Agriculture, Smart Energy and Industry 4.0 use cases.

Section 5 provides the installation and usage guidelines for the components described in Section 4.

Finally, Section 6 summarizes the conclusion and the next steps toward the implementation and validation of the pilot actions for each envisioned use case.

DRAFT - PENDING EC APPROVAL

## 2 State-of-the-Art on Ambient Intelligence via IoT-AR

An efficient combination of, and coordination between, AR and IoT technologies makes it possible to bridge the real world, virtual world (i.e., digital content), and ambient environment (i.e., information from IoT sensors), thereby increasing the richness of information and opening up new possibilities for context-aware ambient intelligence application development.

In addition, AR technologies have become an excellent means of providing rich multi-modal applications and services, in terms of input modalities, output modalities, and interaction modalities (Figure 2), thus expanding the possibilities provided via classical Information Communications Technology (ICT) systems / services [4].

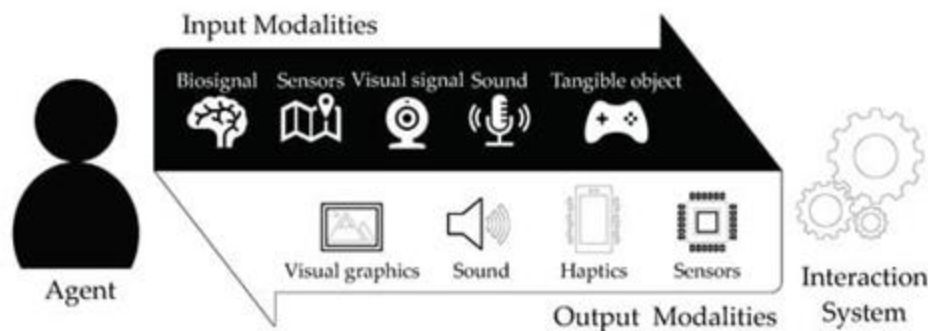


Figure 2. Overview of output and input modalities to interact with ICT systems and services

The applicability of AR-enabled interaction systems in the IoT ecosystem is diverse, including relevant use cases and scenarios, such as: monitoring environmental sensors; smart farming, smart manufacturing or smart cities; application for virtual visits, etc.

Hereafter, a review of state-of-the-art works and studies involving AR and IoT technologies is provided as proof of relevance of the wide applicability and the diverse set of challenges and opportunities available in this domain.

Up to date, many scientific studies and surveys have deeply analyzed the joint potential of IoT and AR technologies (e.g., [4], [5], [6], [7], [8]), exploring potential use case scenarios, and assessing associated technological challenges.

The study in [9] proposed the use of AR as immersive analytical tool for maintenance tasks, with a key focus on improving the safety of workers and supporting their decision making.

The work in [10] introduces the use of AR as a support to IoT data visualization. The proposed IoT-AR system superimposes data from IoT sensors onto real-world objects, supporting object interaction (see Figure 3). The system includes a low-cost multi-camera sub-system, connected to Internet, which allows an accurate measurement of the 3D coordinates of the target objects for displaying the associated information in the appropriate location and perspective, avoiding visibility issues, and for a richer interaction. The system is applied in a smart precision agriculture use case to monitor crops in an environmentally sustainable, natural, and comprehensive manner, and its potential and benefits are highlighted compared to classical visualization and annotation methods.

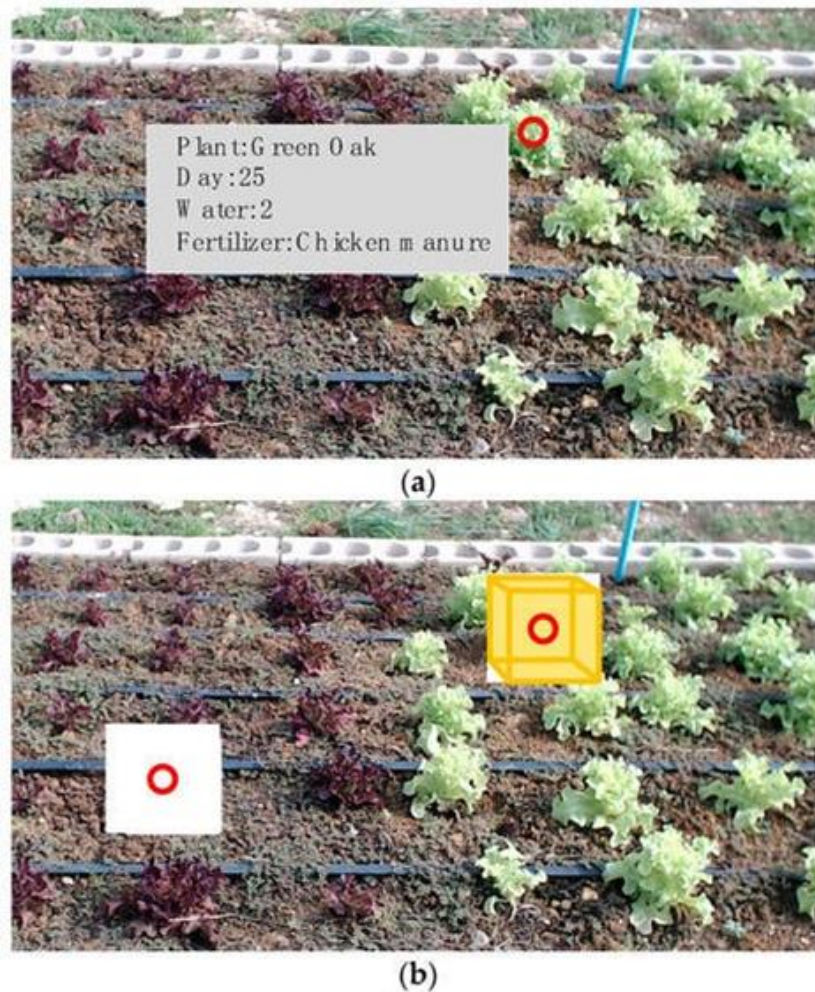


Figure 3. Example of visual representation methods assessed in [10]: (a) virtual text with semi-transparent layer; (b) virtual cube with color to visualize the IoT information

The work in [11] highlights how AR has become a promising technology to provide useful information through perceivable modalities (e.g., visualizations, sounds, vibrations) from data sources that may be unperceivable (e.g., readings from sensors or signals, ratings of a service from other users).

The study in [7] discusses and highlights the potential of providing a synergistic integration between IoT and AR, by providing a literature review on three key aspects, and identifying still remaining challenges and opportunities for each of them: (i) distributed and object-centric data management (including for AR services); (ii) IoT object-guided tracking; (iii) seamless interaction (especially with complex artifacts) and content interoperability.

The study in [12] analyzes up to 79 works (published from 2000 to 2018) having used IoT and AR technologies across a wide set of platforms and use cases. It reveals that over 60 of the proposed solutions utilized vision (light) as input and output modalities, while other modalities, such as sound, touch, motion, and temperature, were used in less than 20 of the proposed solutions in these works. Similarly, the study in [13] analyzed up to 14 works (published from 2014 to 2018) having proposed multimodal interaction systems using AR. From these works, eight input modalities (gesture, speech, face, touch, adaptive, motion, tangible, and click) were identified.

More recently, the study in [4] provides an in-depth review of twenty three (23) studies that have proposed multimodal interaction systems combining IoT and AR technologies, as a

proof of evidence of the huge diversity and possibilities in this domain. That study also reflects on the need for interoperable and open components, modules, interfaces and architectures to support such a diversity of input, output and interaction modalities, which in turn motivates the interaction, communication, processing and AR developments from IoT-NGIN, many of them reported in this deliverable.

DRAFT - PENDING EC APPROVAL

## 3 Ambient Intelligence in IoT-NGIN

This section provides a brief summary of WP4 components supporting the development of Digital Twins, namely IDI and IDAC, and reports on updates regarding the specifications and interfaces of IoT-NGIN IoT-AR modules.

### 3.1 Converging to a Digital Twin solution

A Digital Twin (DT) can be seen as a virtual representation of a physical entity, which may be used for extracting context or knowledge about the physical entity operation in real time or predicting its behavior under changing circumstances without intervening with the actual physical entity. Functionally, DTs are used for supervision, interaction, and prediction of assets with significant benefits arising from the capability of querying and interacting with the digital representation, even in cases that it would be impossible or impractical to do so with the physical asset.

DTs have been first introduced by the aerospace industry and have been revolutionary in that area, as tests regarding new parts have been made possible on DTs, without the risk of damaging the physical equipment in case of test failures. The application surface for DTs has been significantly extended since their initial occurrence, providing similar benefits in a number of domains, such as other Smart Industry, Smart City, or Smart Energy use cases, among others.

The realization of a DT can be leveraged by IoT, AI and advanced analytics, as well as advanced visualization technologies, but also immersive technologies such as AR. IoT is a significant driver, enabling monitoring the status of the physical asset and enabling creating context around its operation. AI and analytics have also a strong share of the predictive capability of the DT. Indicatively, sophisticated Machine Learning (ML) models may be leveraged to predict potential failures, based on real-time data coming from the physical asset, while mathematical models may well simulate the asset's behavior and help decision making under different what-if scenarios. In addition, (3D) visualization facilitates the perception of the asset status and possibly its placement in its environment. Last, but not least, immersive technologies and formats, like AR, provide enhanced interfaces for human-DT interaction in the context of building human-centric applications or symbiotic human-machine systems.

DTs provide obviously significant benefits in many relevant industries. However, they are application and asset-specific, so there is no single solution or technology which may address the challenges in every application domain. A DT is a combination of technologies, which additionally comes with challenges for its effective realization.

A specific challenge relates to the quality, appropriateness or even format of data collected and used by the DT to infer the asset's situation. Collected data must be usable by the DT, which means that they represent the necessary asset properties and are of sufficient quantity and freshness in order to create credible context. Moreover, interoperability is important, as the DT might need to integrate information from multiple sources with diverse interfaces.

Moreover, security must be ensured at the connection points with both the devices and querying entities. Each endpoint represents a potential security vulnerability, which may allow cyber-attackers take control of the DT and thus cause damage even to the physical devices.



Within WP4, these two challenges which relate with the data collection and security of bidirectional interfaces of the DT are addressed through the IDI and IDAC components.

Both IDI and IDAC have been presented in D4.3 [3], reporting their final specifications. Updates and fixes are being applied since then on the basis of the feedback received from the deployment in the context of the IoT-NGIN use cases.

Next, a short functional description of each component is provided.

### 3.1.1 IoT Device Indexing (IDI)

The IDI module is used to connect to IoT devices, collect their measurements, keep their DT appropriately updated as well as keep a record of the changes in the data. The component also supports the communication of commands to the device via its DT. The IDI component, as depicted in Figure 4, consists of the following sub-components:

- FIWARE Orion Context Broker (Orion) [14], which is used to store the latest updates of an IoT device's data. Orion holds a Next Generation Service Interfaces v2 (NGSiv2) representation of the device's data. An external user can query the data of the device without directly interacting with the device itself or can even pass commands through the component.
- FIWARE IoT Agent [15], which is used as a way for devices to communicate with Orion, in cases they have not adopted the NGSiv2 representation for their data. The IoT Agent receives the device's data in their native communication protocol (HTTP, Message Queuing Telemetry Transport (MQTT), etc.), makes the appropriate transformation and passes the NGSiv2 representation of the information to the Context Broker. An important feature of the IoT Agent is the usage of two mandatory headers, which are used to implement the multitenancy [16] feature.
- Historic Data Registry [17]. Since Orion does not by default hold the changes that occur in a device's data, the Historic Data Registry is used to keep a record of the changes in the data. The structure of the Registry keeps in line with the multitenancy architecture of Orion and the IoT Agent. The Historic Data Registry communicates with Orion through its *notification mechanism* [18]. A subscription to receive notifications has to be made for the Registry. Depending on the different tenants and for which devices a record is desired to be kept, there could be multiple subscriptions that need to be created.
- Inactivity Checker [19], which is used to determine whether a device is new (has just been registered) or not, as well as its activity status. The activity status is determined for all the available devices, based on their last update.

The component is available as open source on the H2020 IoT-NGIN group of the public GitLab instance [20].

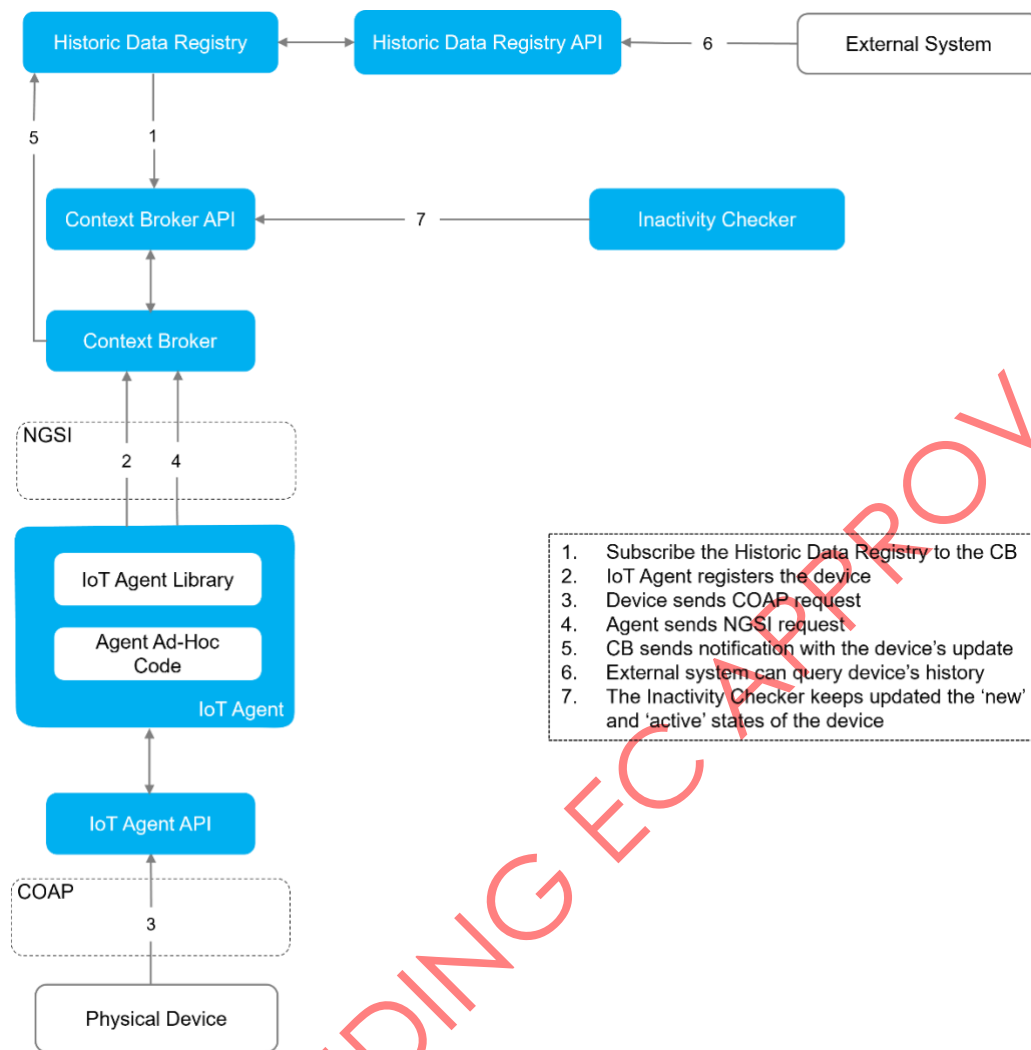


Figure 4. IoT Device Indexing – functional description

### 3.1.2 IoT Device Access Control (IDAC)

The IDAC module handles the access to the resources of the IoT-NGIN project. It is responsible for the protection of services and the authentication and authorization of users. It employs an Application Programming Interface (API) Gateway which applies multi-criteria resource-centred controls on requests for IoT-NGIN resources, before forwarding the requests to such involved resources. Moreover, IDAC implements criteria-based controls in the form of independent pluggable modules, which can then be selected to be applied in each resource or not. IDAC's high-level architecture is depicted in Figure 5.



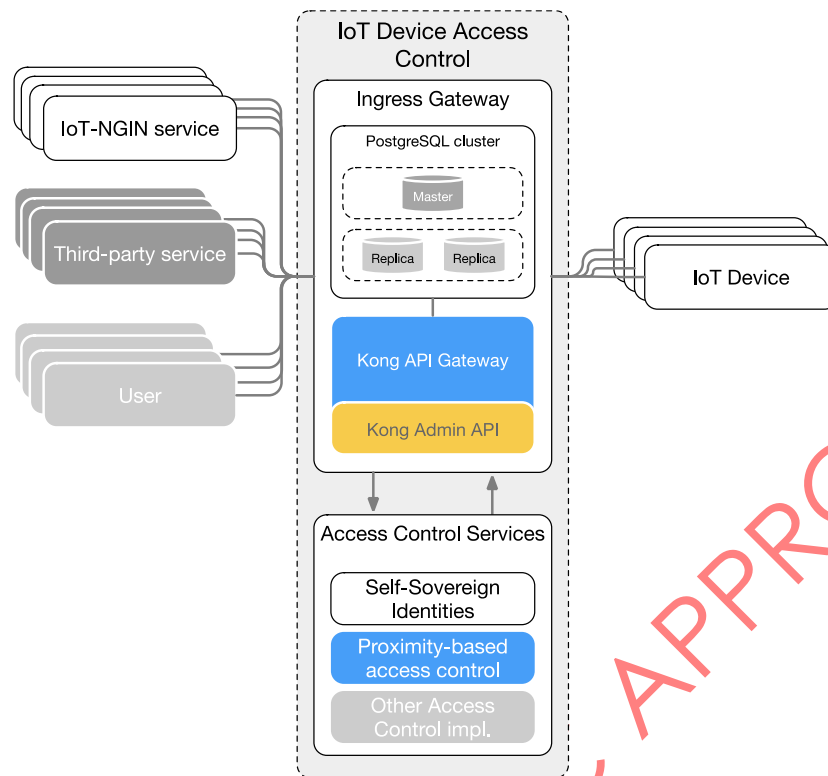


Figure 5. High-level architecture of the IDAC module.

In IoT-NGIN, the Kong Gateway API [21], an open-source project that enables the use of a variety of security methods, like oauth [22] and jwt tokens [23], has been adopted. In particular, the following custom plugins have been implemented in IoT-NGIN:

- Proximity: A plugin that enables ambient intelligence based protection. It compares the distance between the device of a user making a request and the requested device and, if it is lower than a configurable threshold, it allows access.
- Keycloak: A plugin that enables Keycloak authentication for both public and private clients.
- SSI (Self Sovereign Identities): A plugin that enables authentication based on the Privacy Preserving SSI component.

The module also uses the Konga Graphical User Interface (GUI), an interface for Kong that allows quick and easy management of the Kong services and plugins.

The component is available as open source on the H2020 IoT-NGIN group of the public GitLab instance [24].

## 3.2 IoT-AR Modules

An in-depth overview of the scenarios and requirements of all IoT-NGIN UCs making use of IoT and AR technologies has been provided in Section 2 of D4.3 [3], namely: Smart Agriculture (UC #4); Industry 4.0 (UCs #6 #7); and Smart Energy (UC #10).

On the other hand, D4.3 [3] provided a comprehensive overview of the communication workflow (in Section 3) and associated interfaces/protocols/messages (in Annex 1) between the IoT-AR components and the associated IDI module for: (i) discovering information updates; (ii) being provided with periodic updates; and (ii) actuating the status of the registered sensors therein. This is summarized in Figure 6. In addition, Section 3 of D4.3 [3] also

## D4.4 - Enhanced IoT Tactile &amp; Contextual Sensing/Actuating (Final Version)

provided an overall picture of the development frameworks, underlying scenarios and AR devices to be used in each of these IoT-AR use cases.

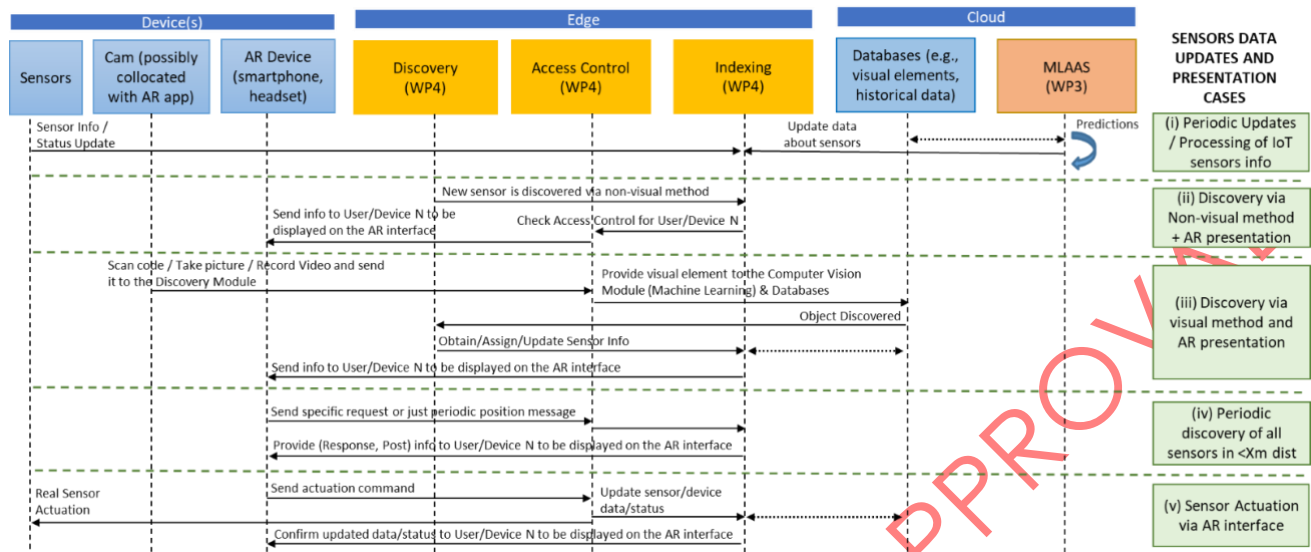


Figure 6. High-level sequence diagrams for AR presentation and actuation cases.

The implementations of the IoT-AR modules and associated apps have closely followed the specifications and designs reported in D4.3, which are detailed in Section 4. This subsection just briefly reports on those specifications and interfaces of the IoT-AR modules for Industry 4.0 and Smart Energy UCs that have realized slight changes compared to D4.3.

#### **Industry 4.0 (#UC6)**

The unique modification compared to what was reported in D4.3 is that the usage of MQTT as communication protocol with the IDI module, which was indicated as an alternative and feasible option in D4.3, has been disregarded after having detected some incompatibilities with some of the features provided by the IDI. Thus, all features and interfaces have been implemented via API REST protocol, which was also identified in D4.3 as a valid (and already tested) alternative.

#### **Industry 4.0 (#UC7)**

The AR application for #UC7 was initially planned for training, sales, manufacturing, and/or maintenance scenarios. All potential purposes were explored, and maintenance-related scenarios were selected for further development.

The specification for content and access control have been refined and slightly redefined. In addition to the 3D model of the cabinet (.fbx), the dynamically retrieved data now includes model target files (.xml and .dat) and model component metadata files (.wri and .xls). As the data is proprietary, access control is necessary to restrict access to only authorized parties. A proprietary Identity and Access Management (IAM) service is used for this purpose. QR-code scanning and decoding for identifying the cabinet are done locally on the display device, and cloud service is not required.

#### **Smart Agriculture (#UC4) & Smart Energy (#UC10)**

While the specifications and interfaces were reported in D4.3, they have been customized for the real implementation in the different Living Labs as part of the Smart Energy and Smart Agriculture UCs.

This section reports on the design specification for the mobile phone AR app, combining Unity and Vuforia, including a seamless integration with all involved components of the IoT-NGIN ecosystem.

The mobile phone AR app covers both the Smart Agriculture and the Smart Energy LL. Both of the apps working with the same architecture, and they change only on the sensor to be detected and also in the fact that the mobile AR app in case of the Smart Energy is equipped with a further functionality that enable the actuation on the sensor. The focus lies in developing an AR app that enhances the user experience by overlaying digital information onto Electric Vehicle (EV) charging stations and Synfield devices / sensors. The primary objective of this app is to provide users with real-time insights and contextual information about sensors, bridging the physical and digital worlds, paving the way for near-future interaction between EV user and charging station via smart glasses/smart contact lenses. In general, the implementation involves the Unity 3D environment, creating image targets using Vuforia as QR codes, establishing communication channels with external services, and seamlessly overlaying digital information onto real devices / sensors.

Figure 7 sketches the component diagram and interfaces for the mobile AR module / app for both the Living Labs.

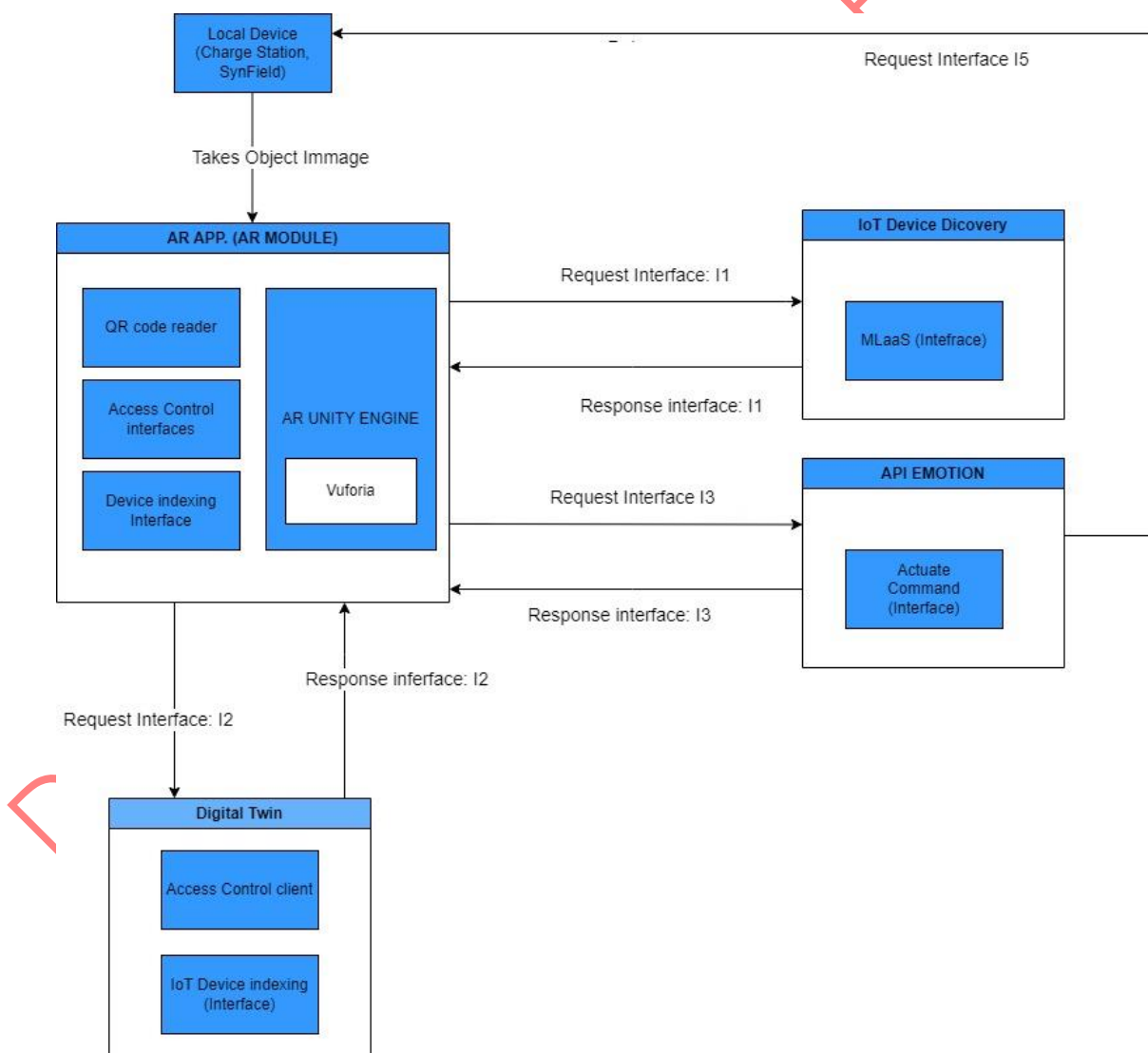


Figure 7. AR module and interfaces for #UC10

Figure 8 provides a comprehensive sequence diagram to depict the dynamic and temporal aspects of the implementation and how the different modules/interfaces interact.

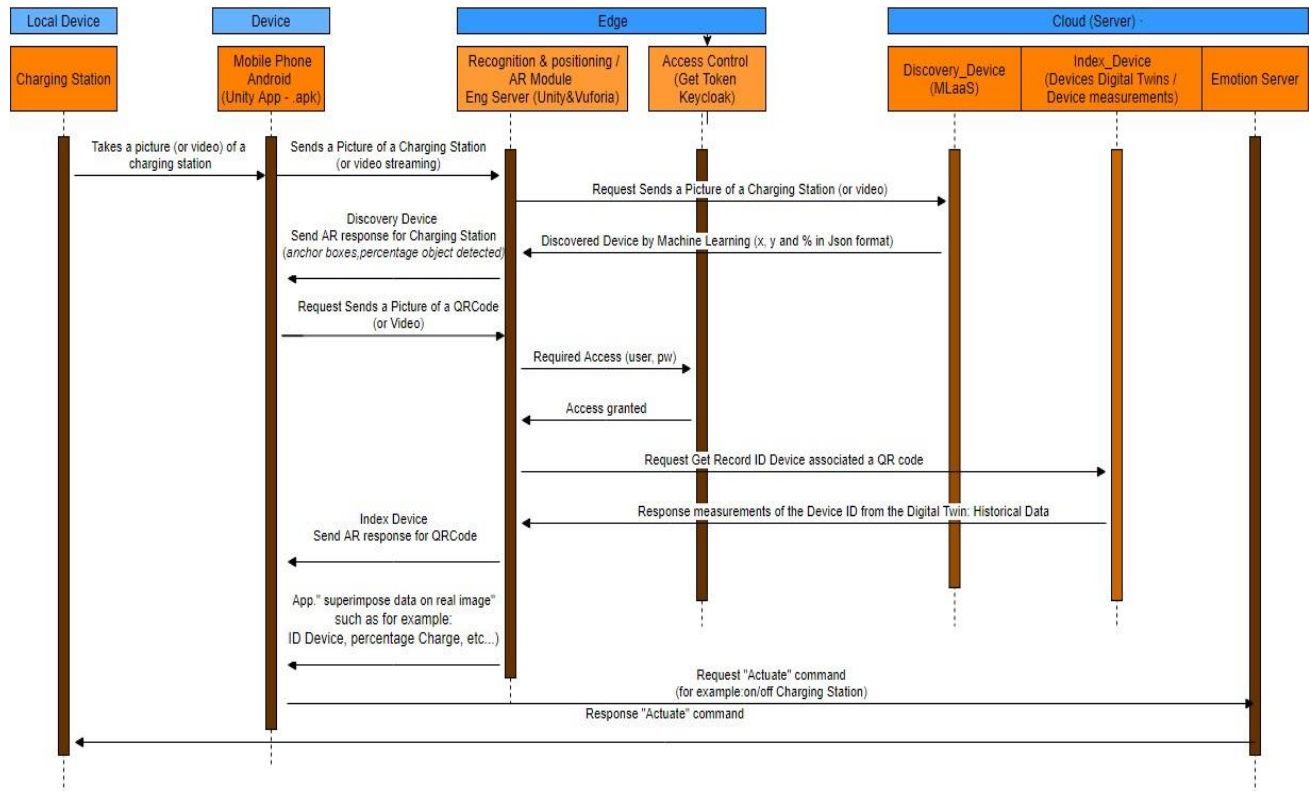


Figure 8. Sequence diagram of AR module interactions for #UC10

Next, a concise description of the APIs incorporated in the final architecture of the AR modules as specified in Figure 7, as well as an overview of the interfaces utilized for communication between the different components, are provided.

#### Interface I1: Connection API to the MLaaS for Device discovery

The Device is identified by the AR app using MLaaS, which results in the recognition of a specific entity, such as an EV Charging Station. This recognition is visually represented on the mobile phone by displaying an anchored edge, depicted as green anchor boxes.

The AR app initiates a call to the MLaaS service, providing an image of the Device and received the inference outcome as a response. This inference outcome is processed by the AR app, generating a surrounding context based on the geometric coordinates. Additionally, the inference outcome provides a value that enables the determination of the detected object's providing also the accuracy of classification as a percentage shown on the frame.

#### Interface I2: Connection API to IDI

The invocation of this interface enables the utilization of the IDI component, which operates behind the IDAC component. By sending requests to Access Control's Keycloak-protected endpoints, users can access the indexing services provided by devices. This facilitates the retrieval of information on the devices' DTs, including the desired measurements of interest, such as sensor-detected data.

### Interface I3+I5: Realization of the "Actuate Command" (At least for the "Charging Station" in Smart Energy UC)

This API enables the display of an 'Actuate Command' on the mobile phone. When the user clicks this button, the application invokes an API to perform ON/OFF (i.e. start / stop) actions for the EV Charging Station, start / stop a charging session. This API is made available by EMOT (manager of the EV charging station).

Finally, Table 3 summarizes the defined and used APIs by the AR module for the #UC10.

Table 3. APIs defined and used by the AR module for the #UC10

API	Request/ Response	Protocol	Method	Expected Input	Expected Output
I1	Request	REST	POST	Image (.jpeg) or video frames	
	Response				File Json (X,Y for anchor box and % Discovery)
I2	Request	REST	GET	ID Device	
	Response				- Retrieve information about the digital twins of the devices (ex: % or Volt of the charging process)
I3+I5	Request I3	REST	GET	ID Device	ON/OFF of the Charging Station

## 4 Implementation for the IoT-NGIN Living Labs

This section provides a description of the new implementation or improvements of WP4 components toward the realization of the Living Labs, for each specific UC. It should be noted that the section is exclusively focused on those components for which new developments have been provided, since D4.3 [3] was delivered.

### 4.1 IoT Device Discovery (IDD)

IDD modules were implemented for three of the Living labs. In each of them, a specific IDD module / method was configured to detect a different type of device, as summarized in Table 4.

Table 4. Discovery methods and objects to be detects in the IoT-NGIN Living Labs

Living lab	Discovery method implemented	Objects discovered
Smart Agriculture	Computer Vision	SynField device
Smart Energy	Computer Vison	Charging station
Industry 4.0	Computer Vision	People, Autonomous Guided Vehicles (AGV), carts
	Ultra Wide Band (UWB)	AGV
	Visual Light Positioning (VLP)	AGV

The next section explains how the computer vision method of the IDD module has been implemented for the living labs of Smart Agriculture and Smart Energy. In addition, a short description of how the IDD module has been implemented in the Industry 4.0 use case can be found in section 4.1.2, although the detailed description of the implementation can be found in D7.3 [25].

#### 4.1.1 AI-based Object Recognition for the Smart Agriculture & Smart Energy UCs

##### General Info / Requirements / Installation

The main objective is to develop an AI-based object recognition model able to detect the Synelxis SynField IoT devices as well as the EMOT EV charging stations from photos taken by mobile devices in the field.

For this purpose, a methodology and framework were developed and followed to build the model described next.

The pretrained model 'EfficientDet D1 640x640' [26] was selected and 'TensorFlow object detection API' was used for the training purposes. TensorFlow object detection API is a framework for creating deep learning networks that solve object detection problems. There are already trained models but were customized for the Living Lab use cases using the



pretrained network weights for the 'EfficientDet D1 640x640'. The architecture of the model is shown in Figure 9.

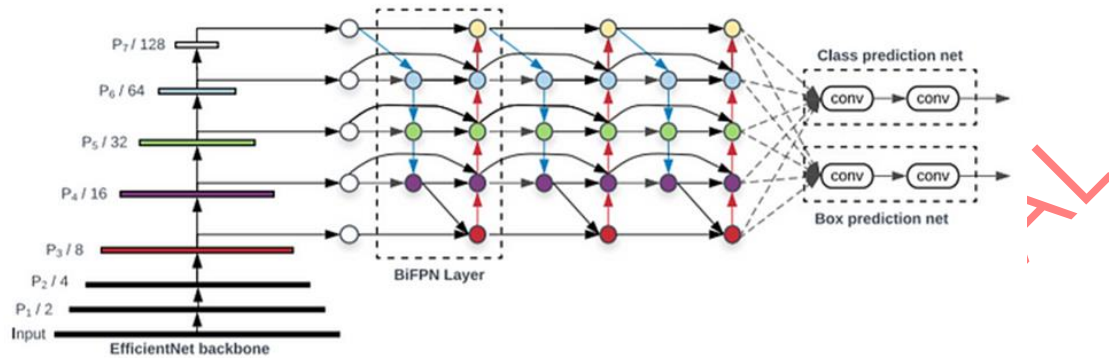


Figure 9. EfficientDet architecture

EfficientDet uses ImageNet pre-trained EfficientNet architecture as a backbone. The proposed Bi-directional Feature Pyramid Network (BiFPN) serves as the feature network, which takes level 3–7 features  $P_3$ ,  $P_4$ ,  $P_5$ ,  $P_6$ ,  $P_7$ , from the backbone network and repeatedly applies top-down and bottom-up bidirectional feature fusion. These fused features are fed to a class and box network to produce object class and bounding box predictions respectively. The class and box network weights are shared across all levels of features.

The output of the model is the detection of the required objects [Synelxis SynField device or EMOT EV charging station]. The whole process of setting up the model, the required APIs, how the input photos were labelled and how we carried out the training process is detailed below. The model has been built in python and requires the following packages:

- Python 3.8+
- Anaconda

### Installation Setup

The initial setup was split into 6 sections:

1. Workspace & Training Files Organization
2. Preparation/annotation of image dataset
3. Record generation
4. Training Pipeline configuration
5. Training and monitoring the model
6. Export resulting model object

### Workspace structure

The structure of our workspace is defined by the directory tree in Figure 10.

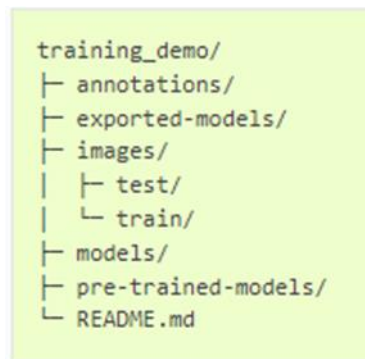


Figure 10. Directory Tree of workspace

An explanation for each of the folders/files shown in the above tree is provided next:

- **annotations:** This folder was used to store all \*.csv files and the respective TensorFlow \*.record files, which contain the list of annotations for our dataset images.
- **exported-models:** This folder was used to store exported versions of our trained model(s).
- **images:** This folder contains a copy of all the images in our dataset, as well as the respective \*.xml files produced for each one via the labeling engine 'Labellmg'. More details are provided in the next section.
- **images/train:** This folder contains a copy of all images, and the respective \*.xml files, which were used to train our model.
- **images/test:** This folder contains a copy of all images, and the respective \*.xml files, which were used to test our model.
- **models:** This folder contains a sub-folder for each training job. Each subfolder contains the training pipeline configuration file \*.config, as well as all files generated during the training and evaluation of our model.
- **pre-trained-models:** This folder contains the downloaded pre-trained models, which were used as a starting checkpoint for our training jobs.
- **README.md:** This is an optional file which provides some general information regarding the training conditions of our model. It is not used by TensorFlow in any way, but it generally helps when you have a few training folders and/or you are revisiting a trained model after some time.

### Training Methodology

The model was built in order to detect the object of interest (Synelixis SynField device or EMOT EV charging station). The model was trained by a series of photos as well as videos capturing the object from different angles. The input of the model is a photo of the area where the Synelixis SynField device or EMOT EV charging station resides. Figure 11 shows a few samples of our training set.





Figure 11. Samples of the training set for the Synelxis SynField device

For the SynField device, EBOS was provided with a total of 116 photos and two videos which were processed to give 5 frames per second. The two videos have a total duration of 63 seconds. Cutting them in 5 frames per second, a total of 332 additional photos were compiled.

For the EV charging station, eBOS was provided eleven videos which were processed to give 5 frames per second. The images from them were then filtered to keep only the suitable images that were clear. The total number of the images that were kept for the training purposes after the initial filtration and pre-processing were 1212 (Figure 12).

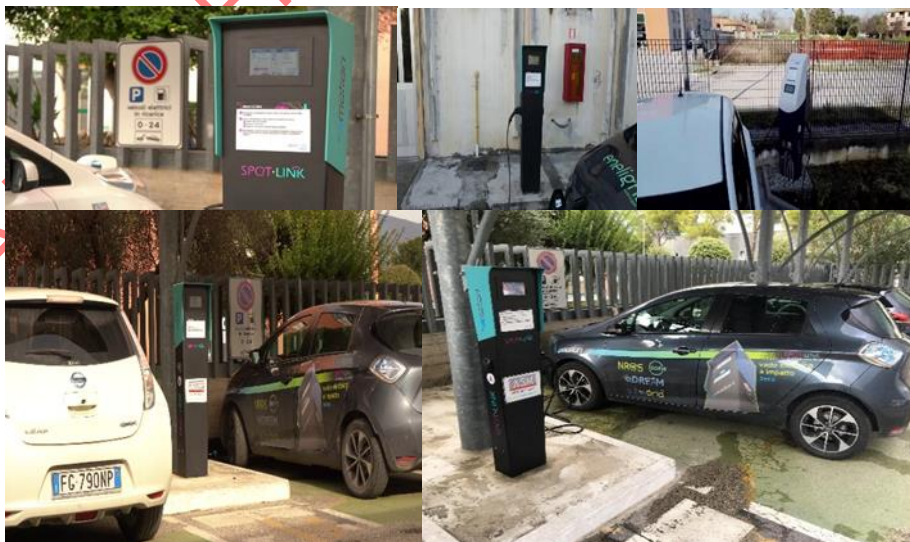


Figure 12. Samples of the training set for emotion Charging Station.

Thus, the total number of the images of both the objects of the interest were 1647. The dataset was split into three sets. A training set, a testing set and a validation set. The ratio for creating the above data sets was set at 90:5:5.

#### Dataset Preparation/TF Records Creation

The first step was to identify and annotate the photographs. There is some software available for this purpose, and 'labeling' was selected, which can be accessed and downloaded here: <https://github.com/tzutalin/labelImg/archive/master.zip>

Figure 13 shows the interface of such an application.

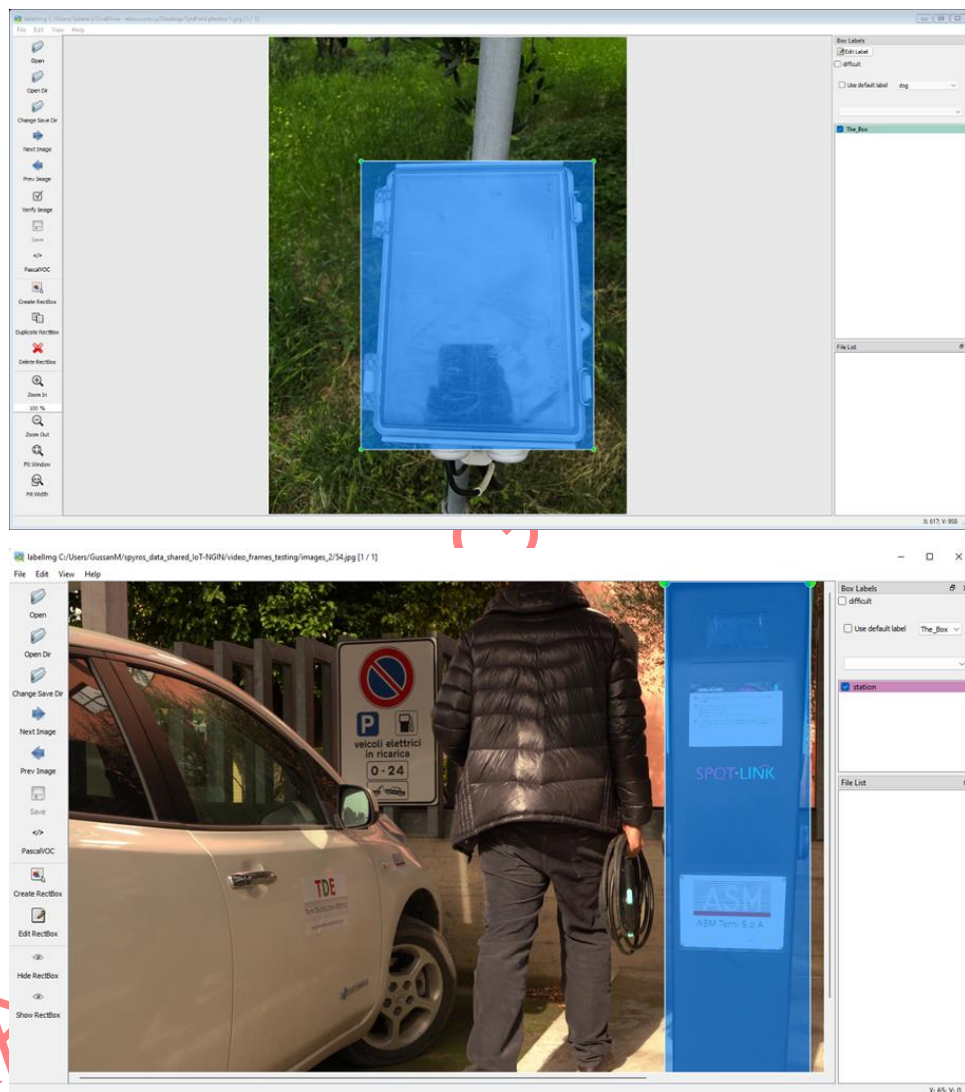


Figure 13. Labelling Interface

An XML version of the photographs was built, by labelling them. The photos were divided into training and test sets. A label map was required by Tensorflow, which translates each of the utilized labels to an integer value. Both the training and detection methods make advantage of this. The generated file was named 'label\_map.pbtxt' and has the format depicted in Figure 14.

```

item {
  id: 1
  name: 'SynField_Box'
}

item {
  id: 2
  name: 'Charging_Station'
}

```

Figure 14. The label of the object.

After the photos were annotated and the datasets defined into training and testing subsets, those annotations were converted into TFRecord format.

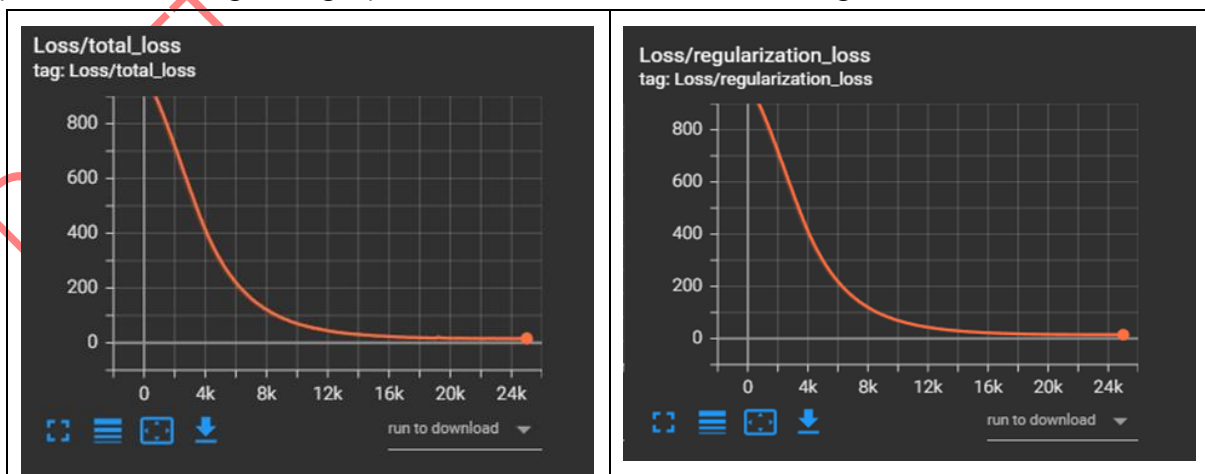
### Model Training

The model used was not developed from scratch; instead, we used one of Tensorflow's pre-trained models, *EfficientDet-D1-640x640*. This was saved in the pre-trained-models folder created.

Some changes at the configuration file *pipeline.config* were performed. Changes include:

- setting the number of the labelling classes (line 3). For our case it is 2.
- setting the batch size (line 131): We put 8. This varies based on your system.
- fine\_tune\_checkpoint (line 161):
  - *pre-trained models/ EfficientDet-D1-coco17 /checkpoint/ckpt-0*
- fine\_tune\_checkpoint\_type (167): Set to 'detection'.
- use\_bfloat16: false (line 168). Set this to false if you are not training on a Tensor Processing Units (TPU).
- label\_map\_path (lines 172, 182): "annotations/label\_map.pbtxt"
- input\_path (line 174): "annotations/train.record"
- input\_path (line 186): "annotations/test.record"

Once the training has started, its progress was monitored through Tensorboard [27]. The following graphs were obtained using Tensorboard. The model was trained for the 25000 steps for the training. The graphs for Tensorflow are shown in Figure 15.





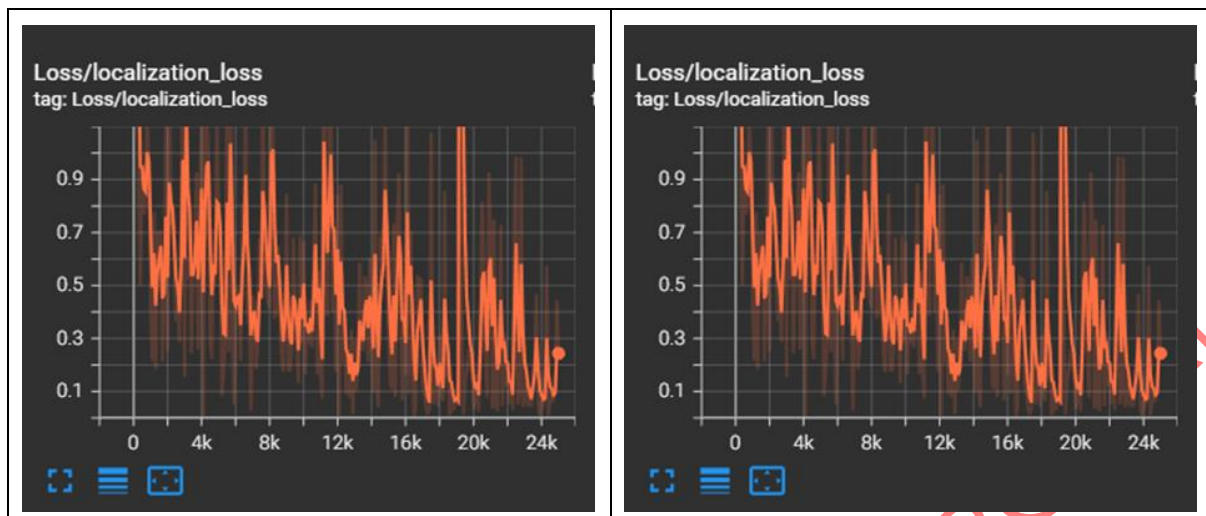


Figure 15. Results from model training

The model was successfully trained and evaluated against the COCO detection metrics [28] for each checkpoint. The checkpoint was created after each 1000 steps to monitor the performance of the trained model. The 12 metrics depicted in Figure 16 were used for characterizing the performance of the trained object detector.

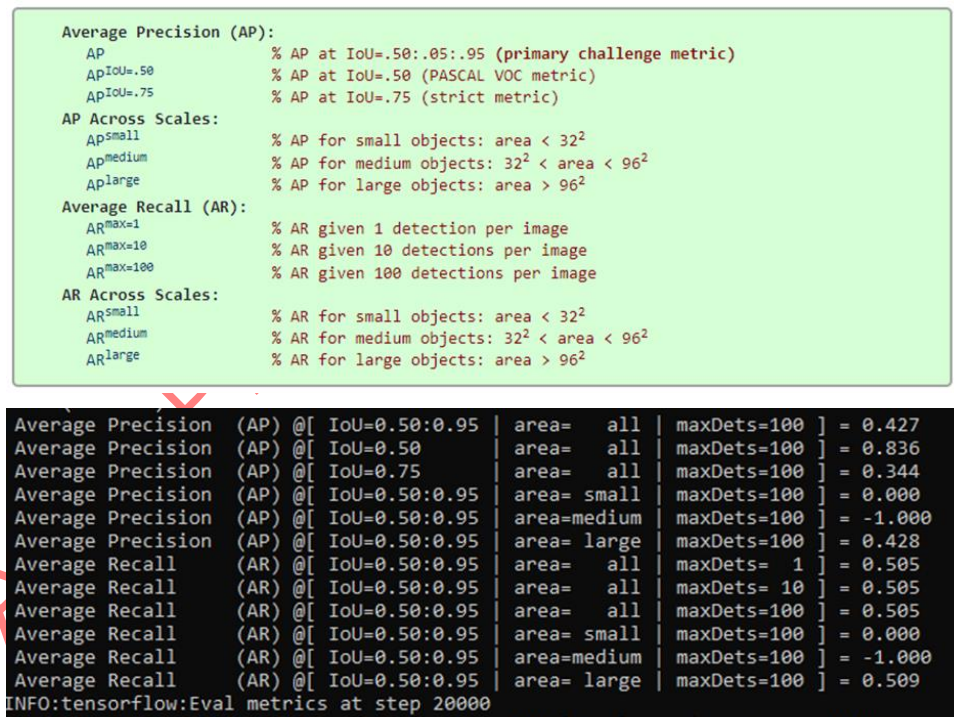


Figure 16. Metrics to characterize the performance of the trained model.

The COCO evaluation metrics were used to validate the performance of the object detection trained model but there were few metrics that cannot be validated against as for example, since for the analyzed case there is only one object per image that needs to be detected and not multiple detections of the object in the same image. The prediction time was 0.18 seconds (average) for the 24 unseen images that the model was tested on.

### Model Testing

The trained model was tested for the objects of interest and it was successful in identifying both the Synfield device as well as the EV charging station. The detection of the model on both objects of interest is shown in Figure 17.

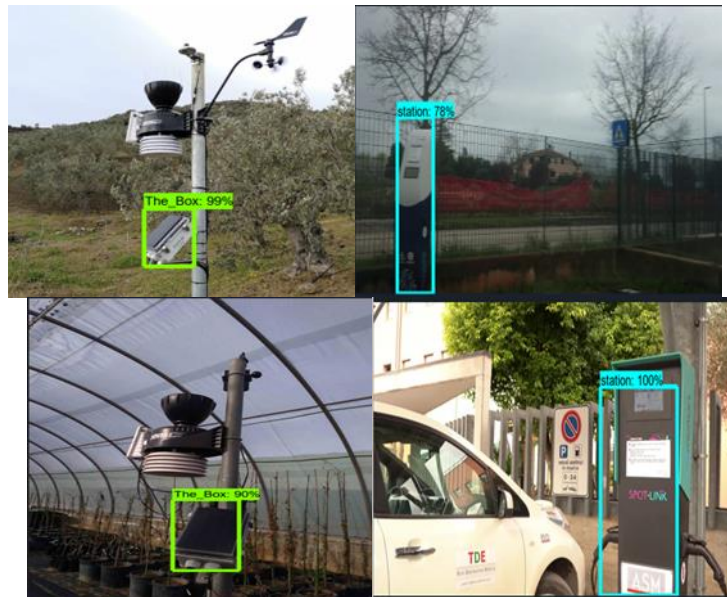


Figure 17. Model testing

### Json Format for the Model Output

The final output from the trained detection model was the detection label of the object that was detected as well as the confidence score with which it was detected according to the coordinates of the anchor boxes.

The resultant json string has the following fields of information:

- The name corresponds to the category that is detected either the "Synfield\_Box" or "Charging\_station".
- The \_n in each x, y axis names show the normalized coordinates of the anchor boxes.
- The simple x,y axis names show the real coordinates of the anchor boxes.

## 4.1.2 Industry 4.0

The three different methods of the Device Discovery module (CV, UWB and VLP) were described in D4.2 [2] and D4.3 [3] and implemented in the Industry 4.0 Living Lab, more specifically in the #UC6 "Human-centered safety in a self-aware indoor factory environment", deployed in the factory owned by Bosch in Aranjuez (Spain).

Each of these three methods was implemented to discover a different type of object in this UC: (i) CV method can detect, track and position people, AGVs, Tuggers, and loaded and empty carts; and (ii) UWB and VLP methods can position and track the AGV and at the same time they can identify particular AGVs, which is not always possible with the CV method. That

is why these different methods can complement each other in this use case, as explained in D7.3 [25].

## 4.2 IoT Device indexing (IDI)

For the IoT Device Indexing (IDI) component, no special implementation needs to be made per use case. The component has been deployed on the Living Labs and is used for the execution of the pilot scenarios. An example of using it in Smart Agriculture has been provided in D4.3 [3].

## 4.3 IoT Devices access control (IDAC)

For the IoT Device Access Control (IDAC) component, no special implementation needs to be made per use case. The component has been deployed on the Living Labs and is used for the execution of the pilot scenarios. An example of using it in the Smart Agriculture and Smart Energy Living Lab has been provided in D4.3 [3].

## 4.4 AR modules

### 4.4.1 Industry 4.0: BOSCH Factory

An AR tool has been designed to increase the safety and situational awareness in Industry 4.0 use cases. In particular, within the scope of #UC6, the goal is to provide ambient intelligence and context awareness by displaying the location, position and related information about a variety of sensors and elements (i.e., AGV, carts and persons), both static and moving, all over the factory.

The AR tool has been developed to run on a Magic Leap headset, so the worker does not have to deviate his/her visual attention while being provided with the IoT-AR information, as both the real-world environment and the IoT-AR information will be displayed on the same field of view.

A distinctive feature of the developed AR tool compared to state-of-the-art ones (reviewed in Section 2) can be highlighted: it is able to display relevant and meaningful information about sensors/elements not only detected via its build-in AR camera, but also by other external visual (CV, VLP) and non-visual (UWB) IDD methods, which all work in cooperation with the IDI and IDAC components of the IoT-NGIN platform. In addition, the AR tool is able to provide a comprehensive situational awareness in the factory, by displaying a mini map of it, with the position of all far and close by sensors being detected inside the factory.

Thus, by using the implemented AR app, workers will be able to move safely, avoid accidents with AGVs or pedestrians, and have a comprehensive and rich situational awareness about the elements and commodities inside the facility, as well as relevant information about them. Likewise, managers can use this kind of information to optimize processes and boost overall efficiency. In conclusion, an AR app like the one developed can increase operational effectiveness and worker safety in different business sectors (e.g., manufacturing, logistics, and retail), by offering rich, valuable and meaningful situational awareness, information and notifications.

The functionality of the AR tool is provided by a number of software elements that operate in a coordinated manner: (i) Magic Leap One headset; (ii) Unity, with Vuforia Engine [29]; and (iii) back-end technology that links the AR app modules to the rest of the IoT-NGIN platform components, and also includes all other core components to provide the desired functionalities.

Figure 18 provides a high-level overview of the software components of the designed AR app, and its interfaces with other modules of the IoT-NGIN platform.

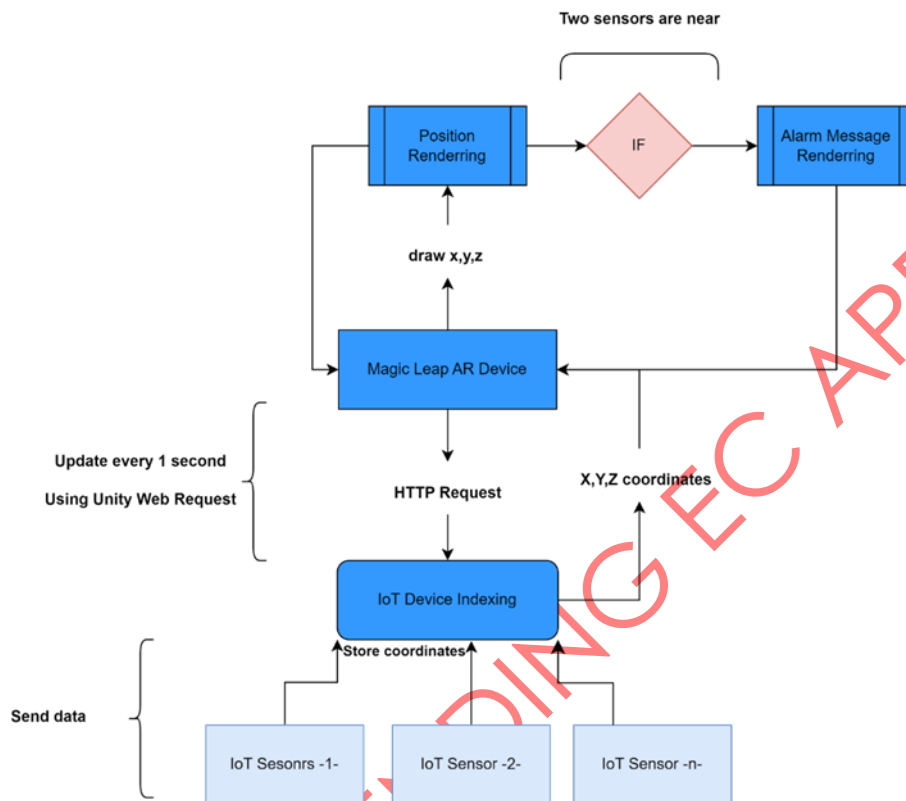


Figure 18. High-level architecture of the developed AR app for Smart Industry #UC6

Next, its main components are described.

### **Detection of Visual Targets**

The AR app makes use of image targets and Vuforia Engine to provide tracking features. Image targets are used to detect certain markers within the environment, while Vuforia is utilized to overlay virtual material or information onto the real-world environment.

These image / visual targets can be strategically positioned around the physical environment to serve specific purposes, like obtaining and superimposing info about elements (e.g., products or machines in the factory) of the environment (see Figure 19, as an example), or even as a support for device positioning in indoor (see next subsection).





Figure 19. Detection of visual targets and superimposing related information, using Vuforia

### **Ad-hoc internal device positioning method for indoor scenarios**

In outdoor scenarios, the position of an AR device can be accurately known by relying on Global Positioning System (GPS) technology. However, GPS is not available in indoor environments, so alternative methods for device positioning would be needed.

One direct approach to overcome this is to track the AR device, or the worker wearing it, by using any of the IDD methods developed in IoT-NGIN project (i.e., CV, UWB, VLP). In addition, an ad-hoc internal device positioning method based on the Magic Leap One headset has been designed and implemented. The Magic Leap One headset makes use of a combination of cameras, sensors, and computer vision algorithms to accurately track the position and orientation of the headset in real-time. On the one hand, since the tracking provided by the AR device itself is limited to a certain view, i2CAT has adjusted such a system using a Virtual Reality (VR) camera that follows the user in the specified trajectory to support the device's tracking process. The obtained data can be processed and used to determine the user's current location in the physical environment. Special relevance is given to the obtained x and y coordinates (since z values will be around 1.7). On the other hand, image targets can support in the device positioning / tracking process. Visual markers can be strategically placed around the physical environment, and can be recognized by the AR camera when moving around it. After detecting an image target, the app can then calculate the user's position and even the associated orientation/direction. The obtained position coordinated can then be stored internally and/or posted to the IDI module, as for any other sensor/element, and can also be used to provide live or event-driven notifications to the worker or AR app's user.

In addition, the position from the AR device can also be correlated and refined with the one(s) obtained by using other discovery methods applied to the AR device or worker, so that an accurate and reliable position can be obtained.

### **Communication between IoT-AR and IDI modules via API REST**

As documented in Annex 1 of D4.3 [3], API REST communication interfaces and messages between the IoT-AR and IDI modules have been implemented.



From Unity, and using the Magic Leap device, it is firstly necessary to request privileges to access Internet and to display information with low latency (Figure 20).

```
// Request access to the Internet. This privilege is required for applications that need to
// access web services or other online resources
MLPrivileges.RequestPrivilege(MLPrivileges.Id.Internet);

// Request access to the local area network. This privilege is required for applications that
// need to communicate with other devices on the same network
MLPrivileges.RequestPrivilege(MLPrivileges.Id.LocalAreaNetwork);

// Request access to low-latency lightwear. This privilege is required for applications that
// need to display content with low latency, such as AR applications
MLPrivileges.RequestPrivilege(MLPrivileges.Id.LowLatencyLightwear);
```

Figure 20. Request of Internet privileges from the Magic Leap

To extract the positions of IoT sensors and/or elements registered in the IDI module, Unity web requests via API REST are sent every 0.5s. The following code snippet (Figure 21) shows how it can be done in case of using a CSV file to store such an information. In case of using the IDI module, the communication interfaces are identical.

```
IEnumerator DownloadFile()
{
    while (true) // Loop infinitely
    {
        UnityWebRequest uwr =
        UnityWebRequest.Get("http://84.88.36.130:11000/database/test_data?test_name=
        pos_prueba_MIA_09-05-2333");
        Debug.Log("Downloading file...");
        yield return uwr.SendWebRequest();
        if (uwr.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError(uwr.error);
        }
        else
        {
            Debug.Log("File downloaded successfully!");
            if (uwr.downloadHandler.data.Length <= 0)
            {
                continue;
            }
            csv = System.Text.Encoding.UTF8.GetString(uwr.downloadHandler.data);
            Debug.Log(csv);
            string[] records = csv.Split("\n");
            Debug.Log(records.Length);
        }
    }
}
```

```
// Get the last record (skip the header)
string lastRecord = records[records.Length - 2];
string[] fields = lastRecord.Split(',');
float x = float.Parse(fields[3], CultureInfo.InvariantCulture.NumberFormat);

float z = float.Parse(fields[4],
    CultureInfo.InvariantCulture.NumberFormat);

// Transform the location of the AGV depending always on
the same origin
AGV.transform.position = new Vector3(x, 0, z);
Debug.Log("AGV position: " + AGV.transform.position);
}

// Wait for one second before the next iteration
yield return new WaitForSeconds(0.5f);
}
```

Figure 21. Communication with IDI module from Unity

### **AR display of information about sensors/elements**

The meaningful data from the retrieved information (position, status...) from the IDI module, either being the last entry or historical data from the registry, will be shown on the AR display, at an appropriate position and with an appropriate format.

Figure 22 shows an example from a pre-pilot trial at BOSCH factory displaying virtual positions of an AGV (blue 3D model) and a person (red avatar), together with their overall position in the factory displayed on also an overlaid mini map (see next sub-section). All such visual information will be complemented with additional info from the detected elements/sensors, beyond their position, like speed, load, id, etc.

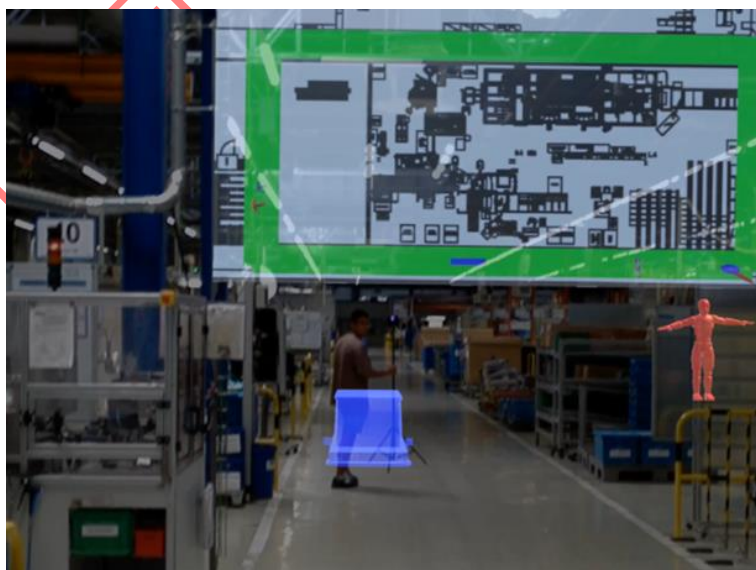


Figure 22. AR information displayed for the #UC6 (Bosch factory)

### **Mini map display with info about available sensors**

In large environments, like a factory, it becomes very useful to get provided with an overall and comprehensive situational awareness. For this purpose, as the Bosch factory has been equipped with camera, leds and UWB tags in different places to detect the sensors/elements (i.e., AGVs, persons, carts...) from/at different parts of it, a new functionality has been developed to provide such a requirement via the AR app. It consists of displaying a mini map on a corner of the field of view, showing the tracked area (see green area of the map displayed in Figure 22), and the information/position of all detected elements in such area, as depicted in Figure 22. A different color and 3D model / icon is used to differentiate between different types of sensors / elements.

The position of all sensors/elements displayed as part of the mini map is retrieved from the IDI module. By default, all sensors/elements are displayed, but a filter could be activated to just display only those sensors/elements within a pre-defined distance from the AR device.

#### **Alarm based on close distance (or detection of collision)**

Beyond displaying information retrieved from the IDI module, the AR app can provide event-driven and/or periodic visual or auditive alarms based on different triggers: (i) distance between any two sensors below a threshold (see code in Figure 23); (ii) risk of collision between a moving sensor and an element of the factory; (iii) a certain parameter of the detected sensors/elements is relevant to the worker using the AR device; (iv) a new relevant sensor/element has been detected; etc.

```
using UnityEngine;

...

public class DetectCollision : MonoBehaviour
{
    public GameObject object1;
    public GameObject object2;
    public string message = "Objects are near each other!";
    private void Update()
    {
        float distance = Vector3.Distance(object1.transform.position,
        object2.transform.position);

        // change the value to adjust the proximity threshold
        if (distance < threshold)
        {
            Debug.Log(Warning!! Objects too close!!);
        }
    }
}
```

Figure 23. Alarm system to inform about near distances between sensors

## 4.4.2 Industry 4.0: ABB Factory

### Overview

The aim of the AR application for #UC7 is to help field service engineers or end customers firstly to identify and locate electrical components inside the physical drive cabinet, and secondly to understand the functionality of those components. A drive line-up can consist of multiple cabinets forming a few or even tens of meters long configuration. Identifying the correct cabinet and then the correct components inside can be time-consuming and frustrating. The application first identifies the individual drive unit and then dynamically loads the 3D- and technical data related to the unique drive and its components. This is achieved by utilizing the digital cabinet models developed within the #UC7.

The AR application consists of several components working together to provide the user with information upon interactions. The architecture is presented in Figure 24. The AR application runs on mobile devices, utilizing Unity with Vuforia Engine, QR-code scanning library, and backend storing and providing the drive related data. Next the main components of the application are presented.

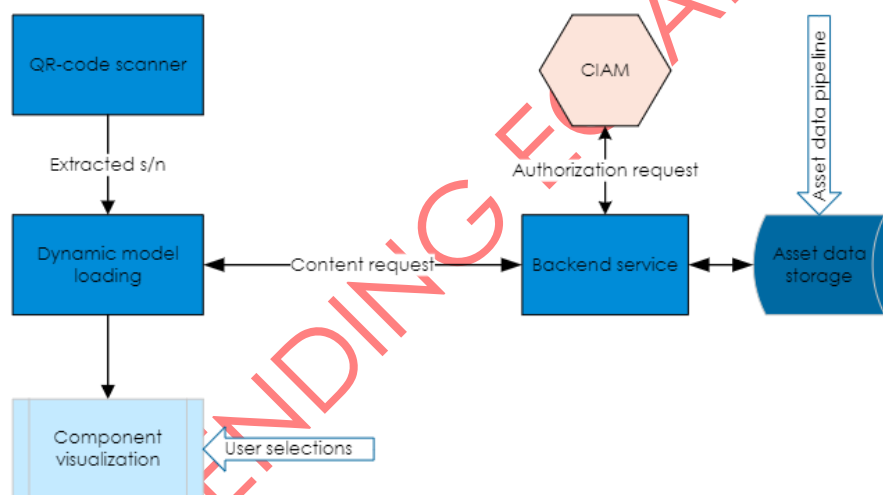


Figure 24. Simplified architecture schematic of the #UC7 AR application

### QR-code scanning

The drive units ship with unique QR-codes from the factory. These QR-codes include a hyperlink to a product documentation portal. However, this portal is not implemented in this case. Instead, the hyperlink comprises the drive unit's serial number, which will serve as a key input for downstream processing. The AR application employs a third-party library to decode the hyperlink from the QR-code, which is shown in Figure 25, and subsequently extracts the serial number from the hyperlink string.

```

void Start()
{
    var renderer = GetComponent<RawImage>();
    webcamTexture = new WebCamTexture(1024, 1024);
    renderer.texture = webcamTexture;
    //renderer.material.mainTexture = webcamTexture;
    StartCoroutine(GetQRCode());
}

IEnumerator GetQRCode()
{
    IBarcodeReader barCodeReader = new BarcodeReader();
    webcamTexture.Play();
    var snap = new Texture2D(webcamTexture.width, webcamTexture.height, TextureFormat.ARGB32, false);
    while (string.IsNullOrEmpty(QrCode))
    {
        try
        {
            snap.SetPixels32(webcamTexture.GetPixels32());
            var Result = barCodeReader.Decode(snap.GetRawTextureData(), webcamTexture.width, webcamTexture.height, RGBLuminanceSource.BitmapFormat.ARGB32);
            if (Result != null)
            {
                QrCode = Result.Text;
                if (!string.IsNullOrEmpty(QrCode))
                {
                    Debug.Log("DECODED TEXT FROM QR: " + QrCode);
                    break;
                }
            }
        }
        catch (Exception ex) { Debug.LogWarning(ex.Message); }
        yield return null;
    }
    webcamTexture.Stop();
}

```

Figure 25. Code for decoding the hyperlink from QR code

### **Dynamic model loading**

The dynamic model loading component takes the drive serial number as input and generates then a request to the backend service. The request triggers the invocation of the IAM service and prompts the user to provide login credentials.

Upon successful login, the request is routed to the backend service, which responds with the 3D model (.fbx), model target files (.xml and .dat), and model part/component details (.wri and .xls). This data is utilized to populate the application's interaction features, and to enable tracking based on the model target files. Figure 26 shows how interaction buttons are populated with 3D model data.

```

public static void CreateProductButtons(string locationName)
{
    List<WriDataModel> buttonData = new List<WriDataModel>();
    buttonData = GlobalData.WriDataList.Where(c => c.FunctionDesignation == GlobalData.LastClickedFunction && c.MountingLocation == locationName).ToList();

    Transform[] test = GlobalData.page3.GetComponentsInChildren<Transform>();

    foreach (Transform transform in test)
    {
        var a = transform.name.ToString();
        if(transform.name == "Content3")
        {
            foreach (Button item in transform.gameObject.GetComponentsInChildren<Button>())
            {
                Destroy(item.gameObject);
            }
        }
    }

    foreach (WriDataModel buttonPD in buttonData.Where(c => c.ProductDesignation != null))
    {
        GameObject model = GameObject.Find(buttonPD.ID);

        GlobalData.m_Renderer = model.GetComponentInChildren<MeshRenderer>();
        if (GlobalData.m_Renderer == null) continue;

        GameObject newButton = Instantiate(GlobalData.buttonPrefab, GlobalData.buttonParent3.transform);
        newButton.GetComponentInChildren<TextMeshPro>().text = buttonPD.ProductDesignation;
        newButton.AddComponent<WriDataModel>();
        newButton.GetComponent<WriDataModel>().ID = buttonPD.ID;
        newButton.GetComponent<WriDataModel>().FullDt = buttonPD.FullDt;
        newButton.GetComponent<WriDataModel>().FunctionDesignation = buttonPD.FunctionDesignation;
        newButton.GetComponent<WriDataModel>().MountingLocation = buttonPD.MountingLocation;
        newButton.GetComponent<WriDataModel>().ProductDesignation = buttonPD.ProductDesignation;
    }
}

```

Figure 26. Code for populating interaction buttons

### Component interaction

Once the AR scene has been populated with loaded data and the tracking is acquired, the GUI is enabled for the user. The GUI includes a feature that generates component selection buttons based on the 3D-model structure and the user's selections. The selection path starts from the top level, and if the chosen component has child components, new selections are generated. Once the user picks a component, its model is turned visible, and a label game object is updated with selected component location + offset, and the label content is updated according to the target component device ID. Figure 27 provides an example of label and highlighted component. Together with component highlight, the label visualizes the location of the component within the physical device. Additionally, the GUI has a 2D component to display the component technical details, if needed.



Figure 27. Example of highlighted component and component tag on a digital cabinet model

### 4.4.3 Smart Agriculture/Energy

#### Smart Agriculture

The AR app underwent thorough testing of its core functionalities within the context of the Smart Agriculture Living Lab. The testing was conducted in accordance with the architecture and component interactions outlined in Chapter 3, composed of the following steps:

- Reference Image Capture: Utilizing an Android mobile phone, we captured an image of the SynField device as a reference.
- Sending Image/Frame to MLaaS: The captured image/frame was then transmitted to the MLaaS platform for device detection.
- Response from MLaaS: Upon transmitting the image to MLaaS, a response is obtained as depicted in Figure 28, presented here for convenience the response message obtained.

```
'prediction': '{"name": "SynField_Box", "score": "61%", "ymin_n": [0], "xmin_n": [0], "ymax_n": [1], "xmax_n": [0], "ymin": [0], "xmin": [0], "ymax": [324], "xmax": [0]}'
```

Figure 28. Call Service MLaaS of the VM

- The response of the MLaaS allows to create the view of the "anchor box" and the % Discovery on the screen of the Android mobile phone (Figure 29). An example of this message is also provided below:

```
"{'name': 'SynField_Box', 'score': '99%', 'ymin_n': [0.5127112], 'xmin_n': [0.6133435], 'ymax_n': [0.81261986], 'xmax_n': [0.7942085], 'ymin': [984.4055557250977], 'xmin': [662.4109554290771], 'ymax': [1560.2301406860352], 'xmax': [857.7452087402344]}"
```





Figure 29. Anchor Box e % Discovery

- The subsequent step involves describing the AR application, which primarily focuses on displaying sensor information associated with the specific SynField artefact on the screen of the Android mobile phone. This visualization on the mobile phone is achieved by first identifying the device ("ID\_DEVICE") through a QR code affixed to the SynField device.

These AR features are achieved through these sub-steps:

1. Creation of ".apk" files using Vuforia and Unity applications. The ".apk" file is installed on the Android mobile phone (See the associated "Installation and user guidelines" subsection in Section 5).
2. When the QRCode affixed to the SynFiled sensor is framed by an Android mobile phone, via an ".apk" file, an AR screen appears on the mobile phone display.

Next, an example of the AR image displayed on the Android mobile phone is shown (Figure 30), when, by framing the QRCode, the data is not present (yet) on the IDI component, as the expected values are not present in the boxes.



Figure 30. AR on QRCode

1. By framing the QRCode, again via ".apk", the connection to the services for the IDI component is activated, but authentication and authorization via IDAC must first take



please. In particular some screenshots are shown below to show part of the Results tested on VM (Ubuntu OS), just for example:

- Figure 31: the token received for authentication (Access Control)
- Figure 32: Historical data on IDI for the last 2 records

[illegible]

Figure 31. IDI: Get a token from Keycloak

```
[{"id": "urn:ngsi-ld:Device:1", "type": "Test", "alias": null, "ip": null, "location": "38.027030, 23.732950"}, {"id": "urn:ngsi-ld:Device:1", "type": "Test", "alias": null, "ip": null, "location": "38.027030, 23.732950"}]
```

```
{ "id": "urn:ngsi-ld:Device:1", "type": "Test", "alias": null, "ip": null, "location": "38.027030, 23.732950"}, { "id": "urn:ngsi-ld:Device:1", "type": "Test", "alias": null, "ip": null, "location": "38.027030, 23.732950" }
```

Figure 32. Test Historical Data on IDI: requests "Get last n records/ID Device" (n=2)

Finally, the visualization of the AR app, always after having framed the QRcode and connecting to the services of the IoT Device Indexing, is shown in Figure 33. It can be seen that the values of the sensors obtained in *Response* and supplied by the DT are displayed in the individual boxes of the AR (on a black background):



Figure 33. AR on QRCode (with the values of the sensors of the SynField node)

Next, an excerpt of the developed software including all previous steps is provided. The complete software can be accessed and reviewed at the project's Gitlab [30] .

## D4.4 - Enhanced IoT Tactile &amp; Contextual Sensing/Actuating (Final Version)

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Vuforia;
5  using TMPPro;
6  using UnityEngine.Networking;
7
8  public class VBTN : MonoBehaviour
9  {
10     public VirtualButtonBehaviour Vb;
11     public GameObject window;
12     public TextMeshPro olive_fuit;
13     // Start is called before the first frame update
14     void Start()
15     {
16         Vb.RegisterOnButtonPressed(OnButtonPressed);
17         // Vb.RegisterOnButtonReleased(OnButtonReleased);
18         Debug.Log("start");
19         window.SetActive(false);
20     }
21
22     public void OnButtonPressed(VirtualButtonBehaviour Vb) {
23         Debug.Log("tes pressed");
24         GetData();
25
26         olive_fuit.GetComponent<TextMeshPro>().text = "Olive fruit fly [stage]:" + 64 ;
27         window.SetActive(true);
28     }
29
30     void GetData() => StartCoroutine(GetData_Coroutine());
31
32     IEnumerator GetData_Coroutine()
33     {
34         //string url = "https://keycloak.iotngin.lab.synelaxis.com/realms/iot-ngin/protocol/openid-connect/token";
35         string url = "https://keycloak.iotngin.lab.synelaxis.com/realms/iot-ngin/.well-known/openid-configuration";
36
37         using (UnityWebRequest request = UnityWebRequest.Get(url))
38         {
39             yield return request.SendWebRequest();
40             if (request.result != UnityWebRequest.Result.Success)
41             {
42                 Debug.Log(request.error);
43                 Debug.LogError("errore");
44             }
45             else
46                 Debug.Log(request.downloadHandler);
47         }
48
49         // WWWForm form = new WWWForm();
50         // form.AddField("username", username);
51         // form.AddField("password", password);
52         // form.AddField("client_secret", client_secret);
53         // form.AddField("client_id", client_id);
54         // form.AddField("grant_type", grant_type);
55         // using (UnityWebRequest request = UnityWebRequest.Post(url, form))
56         // {
57         //     request.SetRequestHeader("Content-Type", "application/x-www-form-urlencoded");
58         //     yield return request.SendWebRequest();
59         //     if (request.result != UnityWebRequest.Result.Success)
60         //     {
61         //         Debug.Log(request.error);
62         //         Debug.LogError("errore");
63         //     }
64         //     else
65         //         Debug.Log(request.downloadHandler);
66         // }
67
68     }
69
70     // public void OnButtonReleased(VirtualButtonBehaviour Vb) {
71     //     Debug.Log("tes released");
72     //     window.SetActive(false);
73     // }
74
75

```

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
using Vuforia;
```

```
using TMPPro;
```

```
using UnityEngine.Networking;
```

```
public class VBTN : MonoBehaviour
```

```
{
```

```
    public VirtualButtonBehaviour Vb;
```

```
    public GameObject window;
```

```
    public TextMeshPro olive_fuit;
```

```
    // Start is called before the first frame update
```

```
    void Start()
```

```
{
```

```
    Vb.RegisterOnButtonPressed(OnButtonPressed);
```

```
// Vb.RegisterOnButtonReleased(OnButtonReleased);
Debug.Log("start");
window.SetActive(false);
}

public void OnButtonPressed(VirtualButtonBehaviour Vb){
    Debug.Log("tes pressed");
    GetData();

    olive_fuit.GetComponent<TextMeshPro>().text = "Olive fruit fly [stage]:" + 64 ;
    window.SetActive(true);
}

void GetData() => StartCoroutine(GetData_Coroutine());

IEnumerator GetData_Coroutine()
{
    string url = "https://keycloak.iotngin.lab.synelixis.com/realms/iot-ngin/.well-known/openid-configuration";

    using (UnityWebRequest request = UnityWebRequest.Get(url))
    {
        yield return request.SendWebRequest();
        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.Log(request.error);
            Debug.LogError("errore");
        }
        else
            Debug.Log(request.downloadHandler);
    }
}
}
```

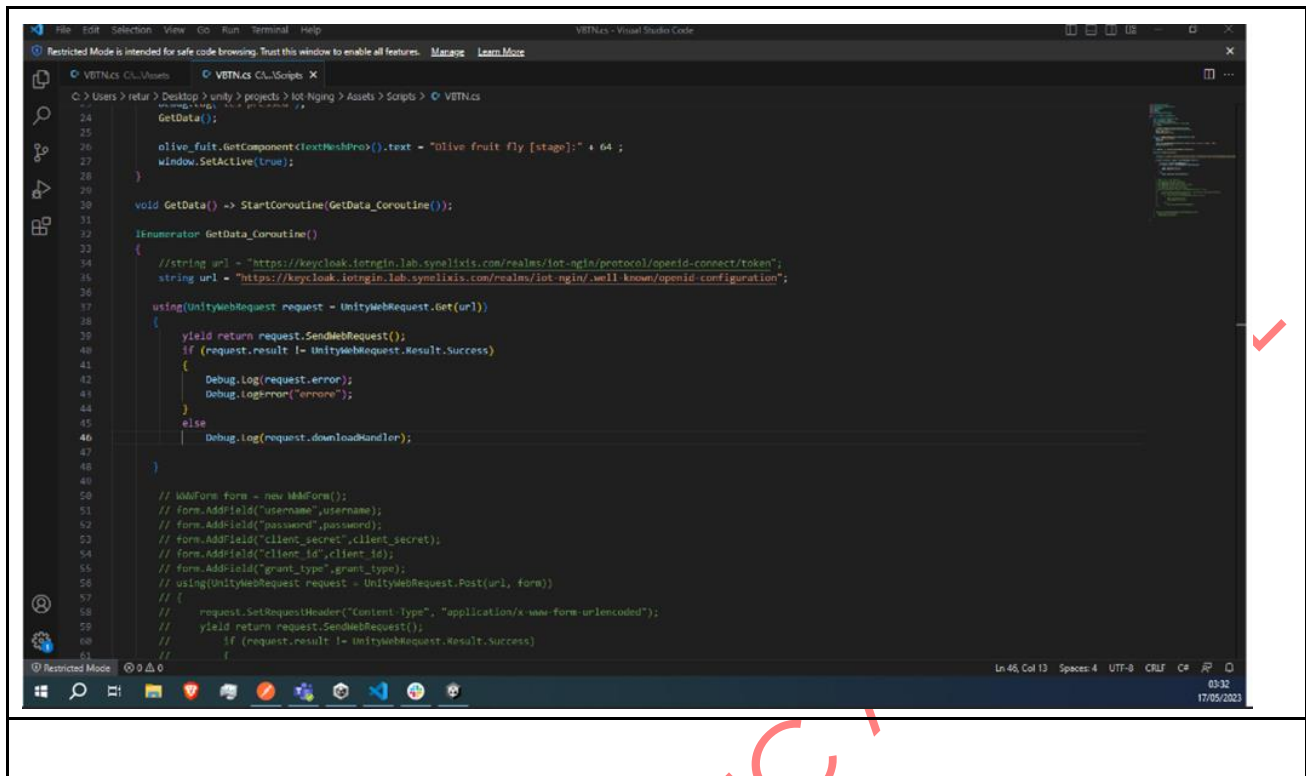


Figure 34. Software of the AR app for Smart Energy UC

Once this .apk being transferred and installed on an Android device, it will be observed on it as shown in Figure 35.

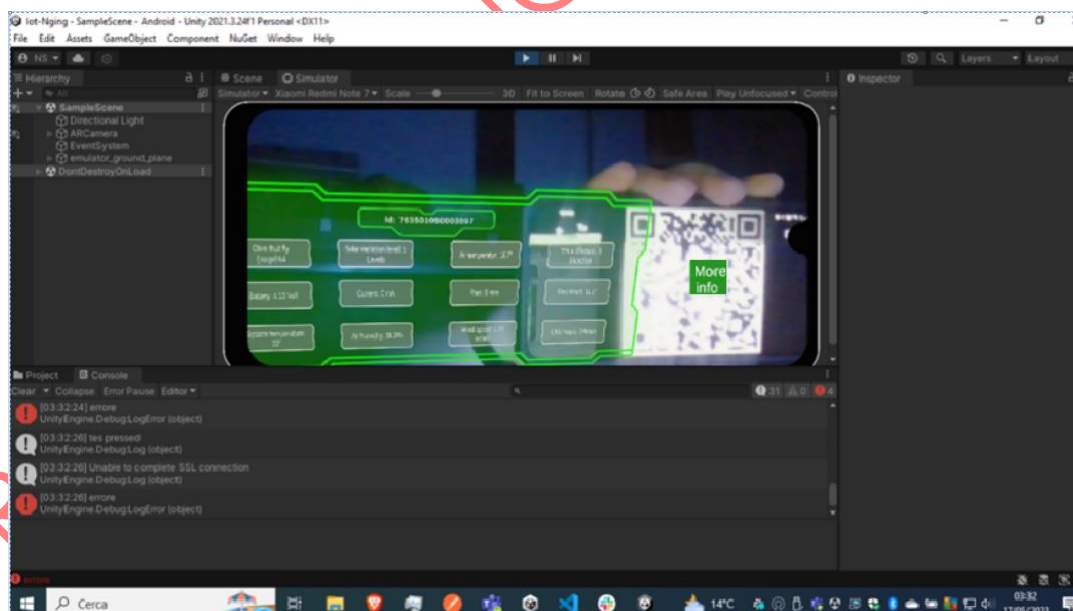


Figure 35. AR on QRCode

A virtual button has been added to start calling services (in this case the IDI service). When the "more info" button is clicked, the calls starts and the interface is populated in AR.





Figure 36. Virtual button



Figure 37. AR on QRCode

This virtual button allows obtaining more information, to be displayed in the various boxes present on the mobile phone display (included in the AR screen above) so as to dynamically obtain the enhancement of the data relating to the sensors connected, for example, to a specific SynField device.

For convenience, next a screenshot is provided to show part of the results tested on VM (Ubuntu OS), including the response we get when a given SynField device is detected. (Figure 38).

```
anasabattin@ubuntu: ~/Desktop/device-indexing-client-for-smart-agriculture-main$ pipenv run python main.py
Loading .env environment variables...
[{"id": "763501050003897", "type": "Synfield", "synfield_data": [{"sensing_service_id": 2599, "value": 6, "ontime": "2023-03-26T21:15:00.008967Z"}, {"sensing_service_id": 939, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 941, "value": 5, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 942, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 943, "value": 18.7, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 945, "value": 1.77, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 946, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 947, "value": 9, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 2165, "value": 0, "ontime": "2023-03-26T20:59:59.999999Z"}], [{"id": "763501050003897", "type": "Synfield", "synfield_data": [{"sensing_service_id": 2599, "value": 6, "ontime": "2023-03-26T21:15:00.008967Z"}, {"sensing_service_id": 939, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 941, "value": 5, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 942, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 943, "value": 18.7, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 945, "value": 1.77, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 946, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 947, "value": 9, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 2165, "value": 0, "ontime": "2023-03-26T20:59:59.999999Z"}]}]}

Loading .env environment variables...

[{"id": "763501050003897", "type": "Synfield", "synfield_data": [{"sensing_service_id": 2599, "value": 6, "ontime": "2023-03-26T21:15:00.008967Z"}, {"sensing_service_id": 939, "value": 4.13, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 940, "value": 22, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 941, "value": 5, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 942, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 944, "value": 66.3, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 943, "value": 18.7, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 945, "value": 1.77, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 946, "value": 0, "ontime": "2023-03-27T10:55:44.107613Z"}, {"sensing_service_id": 947, "value": 9, "ontime": "2023-03-27T10:55:44.107613Z"}, ...
```

Figure 38. Requests Device Index N=2 (with the values of the sensors of the SynField node)

## 5 Installation and user guidelines

The source code for each one of the components and tools described in this deliverable is available on the project's Gitlab, and the concrete reference is provided in the respective subsection. This section provides basic installation and usage guidelines for each of them.

### 5.1 IoT Device Discovery (Computer Vision)

#### 5.1.1 Installation guidelines

The following instructions are provided to be able to execute the demo of the IDD module based on Computer Vision.

The system does not require any special hardware, although systems with a CUDA-enabled GPU would be preferred. As for software, it is best to have Ubuntu 20.04 or Ubuntu 22.04.

The demo code for the IoT Device Discovery based on Computer Vision can be found on the project's Gitlab [31]. For the setup and installation, the following steps and considerations need to be considered:

1. Git clone

You can clone the repository using the following command.

```
$ git clone https://gitlab.com/h2020-iot-  
ngin/enhancing_iot_tactile_contextual_sensing_actuating/iot-device-  
discovery/computer-vision-device-discovery.git  
$ cd computer-vision-device-discovery
```

2. Requirements

The demo can either be executed in the host OS using a Python Virtual Environment or by using Docker.

- a. Local execution

In the root folder of this project there is the Pipfile file to create the virtual environment.

- i. Ensure that the system has pipenv installed correctly. If not, it can be installed with pip. Note that if pipenv was installed via the apt package manager, it should be uninstalled and re-installed with pip as the apt version is deprecated.

```
$ sudo pip3 install pipenv
```

- ii. Once pipenv has been installed, the virtual environment can be created with the following command.

```
$ pipenv install
```

### b. Docker execution

You can also use docker to execute the demo. There are 2 images available, depending on if you have a CUDA-enabled GPU or just CPU. In both cases you need to make sure that docker is installed and configured properly in your system.

- Installation instructions:  
<https://docs.docker.com/engine/install/ubuntu/>
- Post-installation instructions:  
<https://docs.docker.com/engine/install/linux-postinstall/>

#### ii. CPU only

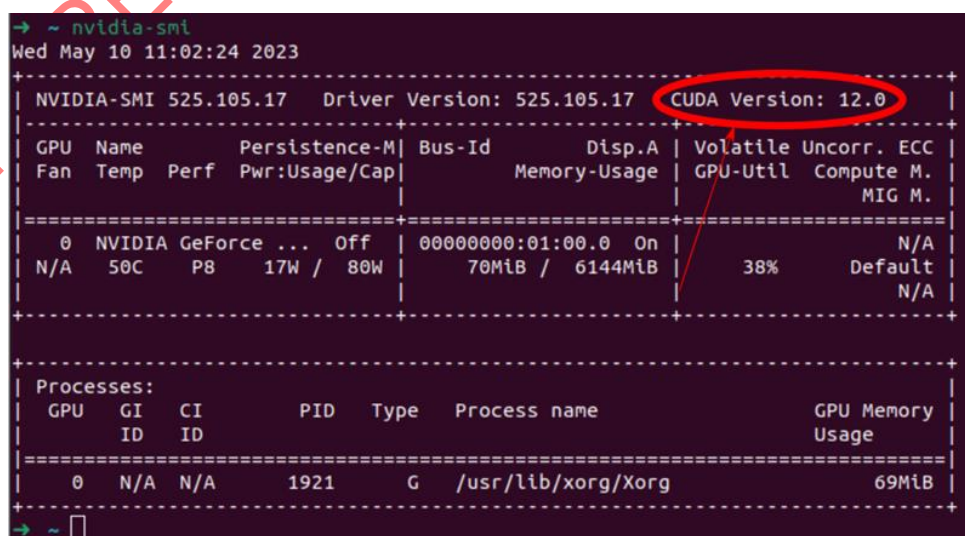
Once docker is properly installed, the following commands can be used to build and run the image. The process might take around 5-10 minutes, as it needs to download the necessary system and Python libraries.

```
$ cd docker/cpu
$ ./build_image.sh
$ ./run_image.sh $PATH_TO_ROOT_FOLDER
$ cd computer_vision_demo
```

**Note:** If the docker installation does not have rootless privileges, the `build_image.sh` and `run_image.sh` files can be edited to add the `sudo` command before the docker instructions.

#### iii. GPU

To use this image, you need to ensure that you have a driver and GPU that supports at least, CUDA 11.3.1. You can check this with the `nvidia-smi` command. You should be able to see the supported CUDA version in the top right corner of the output.



```
→ ~ nvidia-smi
Wed May 10 11:02:24 2023

+-----+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0   |
+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0  NVIDIA GeForce ...    Off          00000000:01:00.0 On   |  70MiB /  6144MiB |   38%     Default   |
| N/A   50C    P8           17W /  80W   |  70MiB /  6144MiB |             MIG M.   |
+-----+-----+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage      |
|=====+=====+
|  0     N/A  N/A         1921     G   /usr/lib/xorg/Xorg             69MiB     |
+-----+-----+
```

Figure 39. Output of the `nvidia-smi` command where the supported CUDA version is displayed

After checking that the driver and GPU support the necessary version, the image can be built and run. The process might take around 10-15 minutes to finish.

```
$ cd docker/gpu
$ ./build_image.sh
$ ./run_image.sh $PATH_TO_ROOT_FOLDER
$ cd computer_vision_demo
```

**Note:** If the docker installation does not have rootless privileges, the *build\_image.sh* and *run\_image.sh* files can be edited to add the *sudo* command before the docker instructions.

### 3. Device indexing and Access Control Configuration

By default, the configuration for the device indexing and access control is empty, so that the object detection and localization demo will not be publishing localization data to the server. If such service is required, then you can use your own deployment of the platform by modifying the missing parameters in the *config/device\_indexing\_defaults.py*:

```
# Keycloak
_C.KEYCLOAK.URL = "http://IP:PORT"
_C.KEYCLOAK.CLIENT_ID = ""
_C.KEYCLOAK.SECRET_KEY = ""
_C.KEYCLOAK.USERNAME = ""
_C.KEYCLOAK.PASSWORD = ""

# ACCESS CONTROL
_C.ACCESS_CONTROL.URL = "http://IP:PORT"
_C.ACCESS_CONTROL.ORION_URL = "http://IP:PORT"
_C.ACCESS_CONTROL.API_KEY = ""
_C.ACCESS_CONTROL.DEVICE_REGISTRY_URL = "http://IP:PORT"
_C.ACCESS_CONTROL.CBROKER_URL = "http://IP:PORT"
```

After the parameters have been correctly modified, a service, device and subscription need to be created in order to push data to the server. The following script can be used to do it:

- Local

```
$ pipenv run python3 setup_device_indexing.py
```

- Docker

```
$ python3 setup_device_indexing.py
```



After the execution, a message confirming the service, device and subscription creation should be displayed in the terminal, as shown in Figure 40.

```
Creating Keycloak Connection. Timeout: 60s
Connection to keycloak successful
First we make a request to provision a service group for a new device
Successfully provisioned a service group
Successfully registered the device
Successfully created subscription
```

Figure 40. Terminal output confirming the creation of a service, device and subscription

## 5.1.2 User guidelines

After finishing the configuration, there is a demo in the repository that can be used as the user guideline.

This demo performs object detection using a Deep Learning object detector network, [YoloX](#), on a pre-recorded video in the Bosch factory, as well as computes the location of each object in a local coordinate system. In more detail:

- Object detection: We use the YoloX [32] family of object detectors, specifically the Yolo-X model. The model was trained with a custom-labelled dataset that was created from videos that were recorded in the Bosch factory in May 2022. The model detects 3 classes of objects:
  - Class 0: person
  - Class 1: small cart
  - Class 2: AGV
- Localization: Once the positions of each of the detected objects in pixel coordinates are obtained, their positions in a local coordinate system in the factory are computed in meters. To perform a mapping between the pixel coordinate system in the image and the meter coordinate system in the factory, a set of 4 points in the factory and their respective points in the image are pre-selected, and an Homography is learned, i.e., a transformation that allows to transform a point in the image plane to a point in the factory coordinate system plane.
- Save results to cloud: If keycloak and the IDI and IDAC parameters have been configured properly, the demo also saves the localization data in the device indexing module.

The demo can be executed with the following commands:

- Local

```
$ pipenv run python3 demo.py
```

- Docker

```
$ python3 demo.py
```

If an output in the terminal just like in Figure 41 is seen, it means that the demo has been executed correctly.

```
Initializing detector...
Initializing localization...
Initializing device indexing client...
Creating Keycloak Connection. Timeout: 60s
Connection to keycloak successful
Processing video: 100%|██████████████████████████████████████| 679/679 [04:00<00:00, 2.83it/s]
Saving detection and localization data to CSV
Compressing output video with FFMPEG...
Demo finished!
```

Figure 41. Terminal output after executing the Computer Vision Device Discovery demo.

Finally, once the demo is finished, it will save several files with the results of the analysis:

- data.csv: In this file, the objects that have been detected and localized in each frame of the video can be checked. For each entry in the file, the following information is available:
  - timestamp: time in which the frame was processed, in Unix timestamp (seconds).
  - frame\_id: frame number.
  - x1, y1, x2, y2: top left, bottom right corners of the object in absolute [0-width, 0-height] coordinates.
  - x1 norm, y1 norm, x2 norm, y2 norm: top left, bottom right corners of the object in normalized [0-1] coordinates.
  - score: detection confidence of the object.
  - class\_id: class of the object. 0 is person, 1 is small\_cart, and 2 is AGV.
  - x meter, y meter: position of the object in the local coordinate system in meters.
- video.mp4: Video with the bounding boxes, score and class of all detected objects.

Furthermore, if the IDI server was configured correctly and the data was uploaded to it, the locations stored in it can be read, either by accessing all the measurements sent (1 measurement per frame) or the measurements for the last  $n$  frames:

- Local

```
$ pipenv run python3 read_historical_data.py --read-all-measurements
$ pipenv run python3 read_historical_data.py --read-last-n-measurements --last-
n N
```

- Docker

```
$ python3 read_historical_data.py --read-all-measurements
$ python3 read_historical_data.py --read-last-n-measurements --last-n N
```

By default, the `--last-n` parameter is set to 2. In either case, the format of the data follows, for each frame, the following structure:

```
"tracking_data": [
    {
        "locationx": "7.8",
        "locationy": "3.19",
        "locationz": "0.00",
```

```
"loctimestamp": "1970-01-20T11:42:11.350936",  
"loctech": "Computer Vision",  
"decawavetime": "27080",  
"class_id": "2.0"  
},  
....  
{  
  "locationx": "2.14",  
  "locationy": "6.37",  
  "locationz": "0.00",  
  "loctimestamp": "1970-01-20T11:42:11.350936",  
  "loctech": "Computer Vision",  
  "decawavetime": "27080",  
  "class_id": "0.0"  
}  
]
```

## 5.2 IoT Device Discovery (UWB)

The installation and user guidelines were provided in D4.3, Section 5.1 [3].

## 5.3 IoT Device Discovery (VLP)

The installation and user guidelines were provided in D4.3, Section 5.1 [3].

## 5.4 IoT Device indexing

The installation and user guidelines were provided in D4.3 [3], Section 5.2, and are also available on IoT-NGIN GitLab repository for IDI [31].

## 5.5 IoT Devices access control

The installation and user guidelines were provided in D4.3 [3], Section 5.2, and are also available on IoT-NGIN GitLab repository for IDAC [24].

## 5.6 AR modules / apps

### 5.6.1 Industry 4.0 (BOSCH Living Lab)

#### 5.6.1.1 Installation guidelines

The AR app for #UC6 has been developed in Unity and to be run on a Magic Leap One device, with Lumin SDK 2.26.

In order to install the AR app 'as it is', its apk can be downloaded (size around 70 MB) from the project's Gitlab ( [33]).

Then, on the Magic Leap device, the next steps must be followed:

1. Install The Lab, which is the main PC application that runs the Magic Leap device.
2. Install the necessary tools inside the Lab, basically the Device Bridge Tool.
3. Drag and drop the app resources inside any folder of the Magic Leap Device bridge.
4. Connect Magic Leap to any available Wi-Fi.
5. Run the App. It is necessary to grant access to the camera and micro of the device.

If the code or settings of the AR app need to be modified, then Unity3D needs also to be installed, and the whole project needs to be cloned. The steps to follow include:

1. Install Unity hub.
2. From Unity hub, install Unity 2020 version (any should work).
3. Install the version of Unity that supports Lumin development.
4. Install MLSDK on your machine (i.e., the SDK of the Magic Leap (ML) Device).
5. Clone the repo ( [33]).
6. Open the project.
7. The scene of this AR app is called ML scene.
8. Navigate to the scene.

#### 5.6.1.2 User guidelines

After running the app, the user should wait a little bit while it starts collecting data from the IDI module or detecting visual targets via the AR camera.

The expected result is to observe 3D elements and associated visual information about sensors / elements being detected. Then, information from sensors in the field of view will be displayed. Information from all detected sensors, inside or outside the current field of view, will be displayed on the mini map.

The AR app must be launched from/at a given starting / reference point, which represents the origin of coordinates (0, 0, 0) for every considered discovery and positioning method in the use case. That origin point is indicated with visual indicators on the floor of the factory. After that moment, users can freely move while wearing the AR headset and will be provided with the info about the detected sensors/elements, and with the associated alarms/triggers.

The Magic Leap device and associated AR app have some limits in terms of tracking, meaning that if the user gets outside of the tracking area, the indoor positioning and any other positioning methods may fail.

So far, the AR app has been tested in the use case and #UC6 pilot scenario (BOSCH factory), with satisfactory results. Demo video: <https://www.youtube.com/watch?v=IhYtL7mgGjQ>

Finally, it must be remarked that this app has been fine tuned to work inside the BOSCH factory (#UC6 scenario), so it needs to be configured if it is planned to be used in other environments.

## 5.6.2 Industry 4.0 (ABB Living Lab)

### 5.6.2.1 Installation guidelines

The AR app for #UC7 has been created with Unity and can be installed on Android devices following the next steps.

1. Download the .apk file from the external source onto your Android device.
2. Open the Settings app on your Android device.
3. Scroll down and select "Security" (or "Privacy" on some devices).
4. Enable the "Unknown sources" option. This will allow you to install apps from sources other than the Google Play Store.
5. Open the "Downloads" app or the file manager app on your Android device.
6. Navigate to the folder where the .apk file was downloaded.
7. Tap on the .apk file to begin the installation process.
8. Follow the on-screen prompts to complete the installation.
9. Once the installation is complete, you can launch the app from your app drawer or home screen.

If modification on code or settings are required, these are done the same way as presented for #UC6 in section 5.6.1 under *installation guidelines*. For proper functioning in other use cases, the backend service API endpoints and identity and access management services need to be configured.

### 5.6.2.2 User guidelines

First the app must be opened. Next step is to scan a valid QR-code containing the key for generating model data request. After scanning, the application should prompt to input user credentials. Once access is granted, it should take a moment for the application to load the model and the data. Then the mobile device's camera should be pointed so that the target object is visible and after a few seconds the model tracking is acquired. The UI is now enabled and becomes visible, and it is now possible to start selecting different components or search for specific component. When selected, the component becomes visible, and the UI element shows data of the component. The UI element can be closed for better view.

It should be noted that the application has been highly customized for this specific use case.

## 5.6.3 Smart Agriculture/Energy

Unity together with Vuforia have been used to create an AR application capable of showing information of interest relating to a certain detected device on the display of an Android mobile phone, via HTTP requests to an IDI instance.

### 5.6.3.1 Installation guidelines

The process to follow in order to install the various components needed for this project is summarized next.

**A) Download and install Unity Hub. Also, install the version “2021.3.24f1” of Unity from Unity Hub.**

Below is the link to launch the installation procedure: <https://unity.com/download>

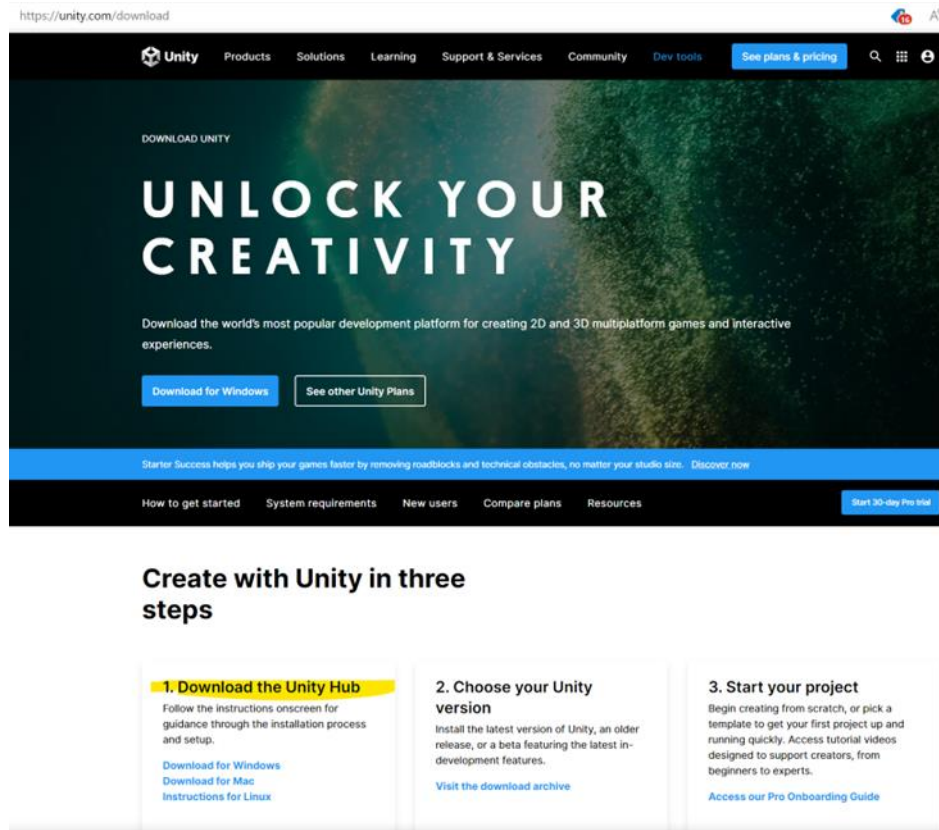


Figure 42. Unity Download

The version with a free license, as well as the underlying Operating System (OS) should be selected.

The Unity Hub supports the following Operating Systems:

- Windows 7 SP1+, 8, 10 (64-bit versions)
- macOS X 10.13+
- CentOS 7
- Ubuntu 18.04, 20.04

For instance, the Instructions for Linux can be found here: <https://docs.unity3d.com/hub/manual/InstallHub.html#install-hub-linux>

Once Unity Hub is installed, it should be opened, and the appropriate version of Unity with Android build support should be installed to build .apk files.



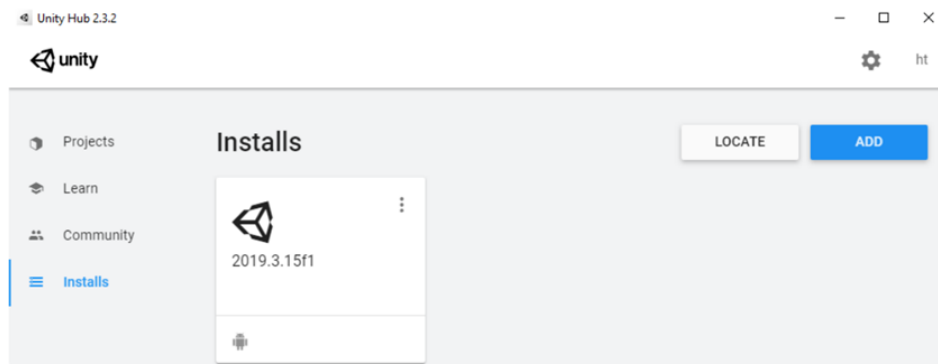


Figure 43. Unity Installs

Then, a new project should be created, as shown in Figure 44.



Figure 44. Create a New Unity project (I)

Give a project name, the 3D template option should be selected and "CREATE" should be clicked, as in Figure 45.

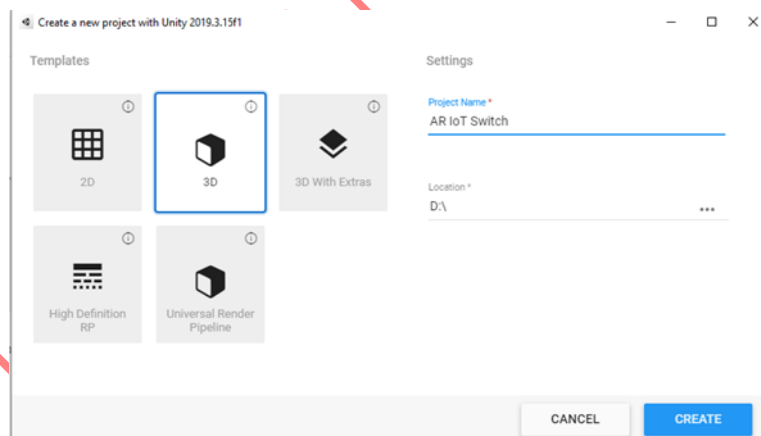


Figure 45. Create a New Unity project (II)

After that, the project will be created, as shown in Figure 46.

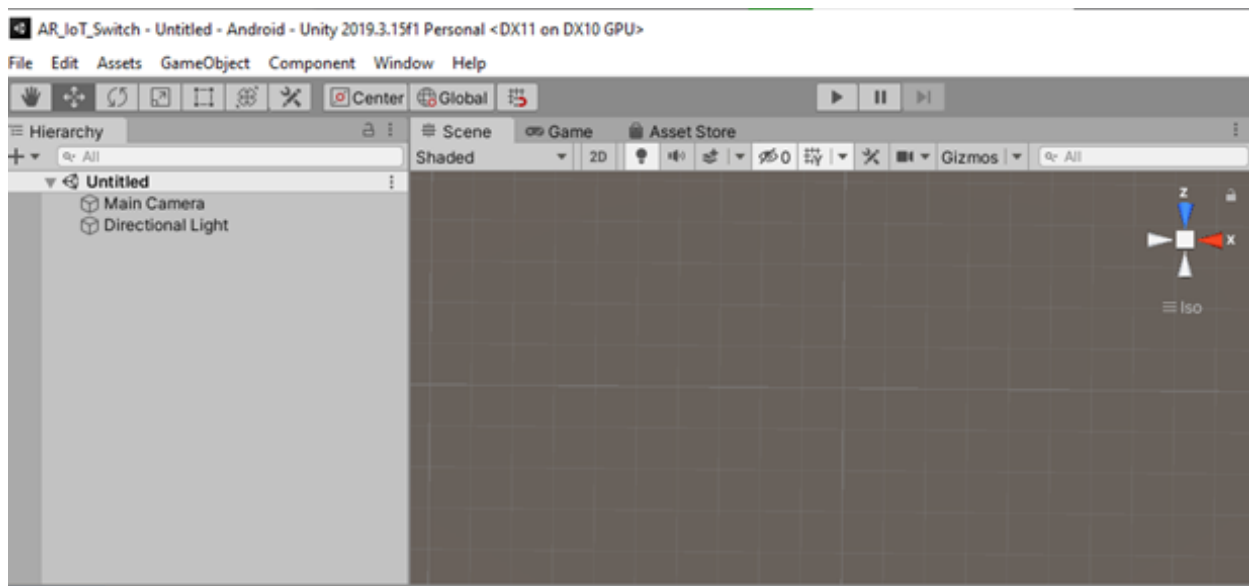


Figure 46. Unity project created

The scene can be saved through the “File” submenu, with the desired name (e.g., “IoT-Ngin”), as presented in Figure 47.

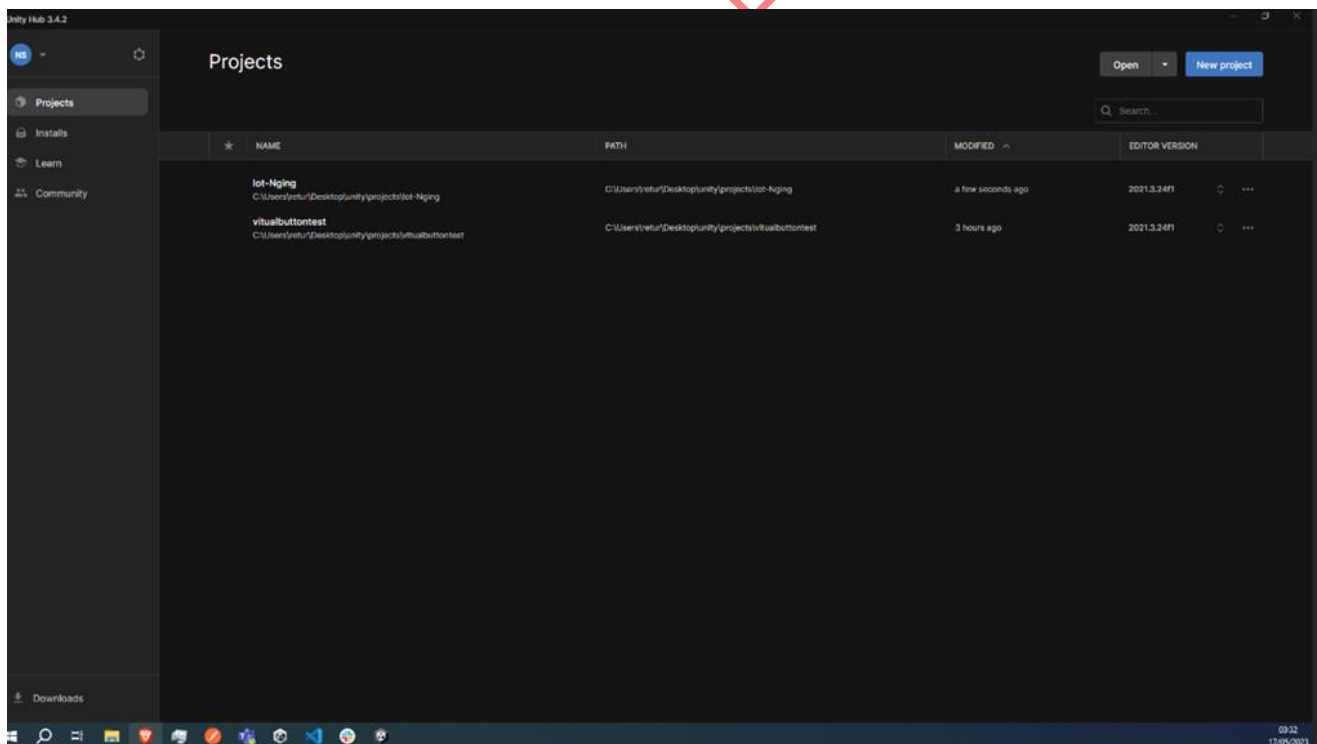


Figure 47. Save Unity project

## **B) Download the Vuforia Engine for your Unity.**

It can be installed via Unity Hub itself. To this, a developer account needs to be created at

<https://developer.vuforia.com/>

After that, it can be downloaded at: <https://developer.vuforia.com/downloads/sdk> (Figure 48).

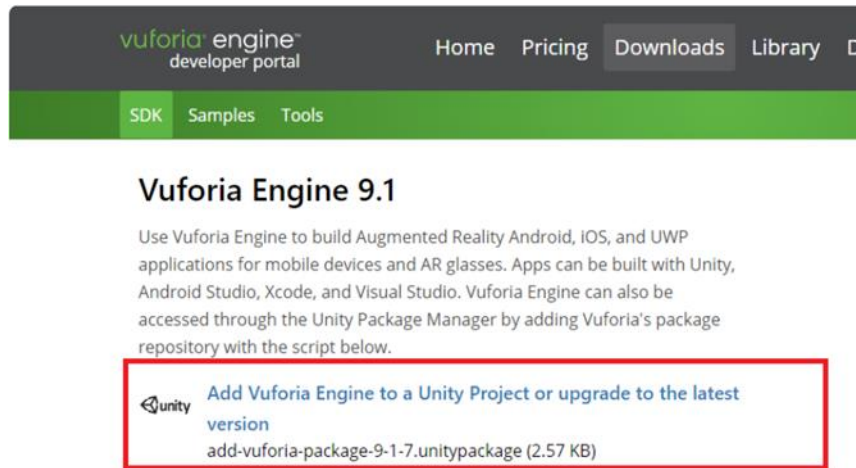


Figure 48. Download Vuforia Engine

Then, the user should login and agree to the terms, as shown in Figures 49 and 50.

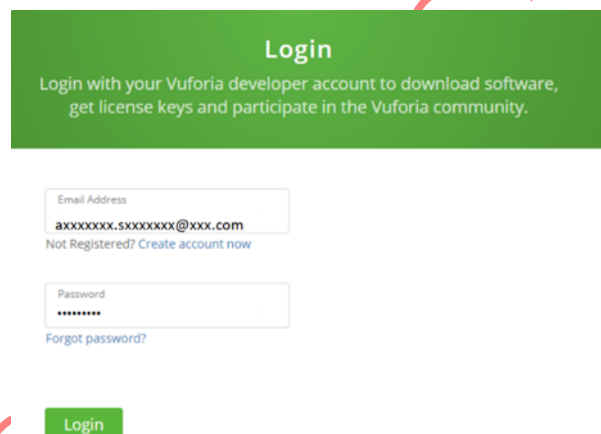


Figure 49. Login and Agree the Terms (Vuforia)



Figure 50. Software License (Vuforia)

The user should then open the downloaded file in Unity. After that, the user should click on Import (Figure 51).



Figure 51. Import Vuforia SDK into Unity

Now the Unity Software is ready along with Vuforia Engine.

For support, the user may visit this link: <https://developer.vuforia.com/support#report-issues>

### 5.6.3.2 User guidelines

As described in the previous subsection, once the 2021 3.24.f1 version of Unity has been downloaded and installed from Unity Hub, and the Vuforia Engine for Utility has been downloaded, we move on to the user side to carry out the next steps.

#### **STEP 1: Setup Target Image.**

Next, a target image is needed. Every time the camera detects this image, the information of interest will be displayed on the Android mobile phone display via AR.

For example, the SynField device image (Figure 52) and the EV Charging Station (Figure 53) as the target images are used in this guide.



Figure 52. Example image of the SynField device



Figure 53. Example image of EV Charging Station

Alternatively, any image of choice could be used (e.g., QRcode, etc.).

Now, the user should visit <https://developer.vuforia.com/vui/develop/databases>. Then, click on "Add Database" under "Target Manager" tab.

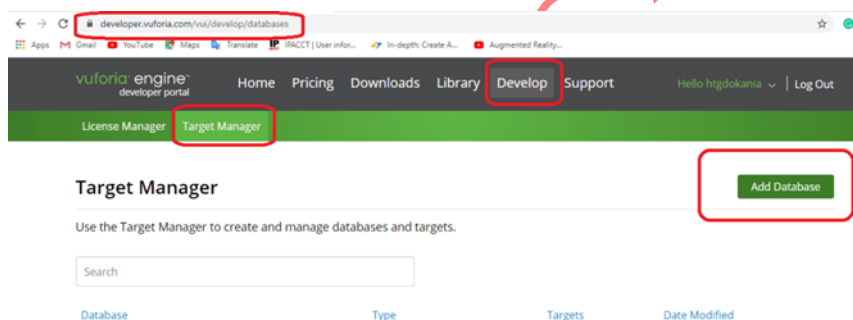


Figure 54. Target Manager Tab in Vuforia

The user should then give a name, select device as the "type", and click on "Create" (Figure 55).

### Create Database

Database Name \*

IoT

Type:

☒ Device

☐ Cloud

☐ VuMark

Cancel Create

Figure 55. Create Database on Vuforia

The user should then click on “Add target” and browse their target image as shown in Figure 56.

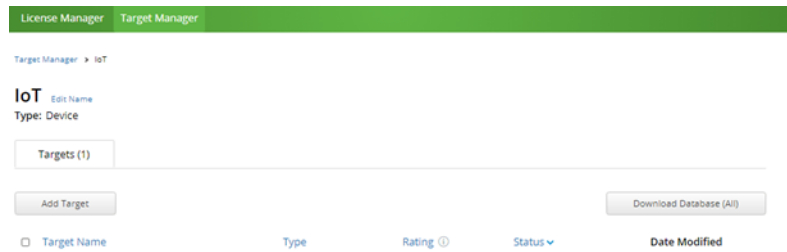


Figure 56. Add Target in Vuforia

The user should also give a “Width” of their choice (say 4) and a name (say bulb). Finally, they should click on “Add” (Figure 57).

**Add Target**


Type:



Single Image



Cuboid



Cylinder



3D Object

File:

WhatsApp Image 2020-06-12 at 4:17:29 PM.jpeg
Browse...

.jpg or .png (max file 2mb)

Width:

4

Enter the width of your target in scene units. The size of the target should be on the same scale as your augmented virtual content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image.

Name:

bulb

Name must be unique to a database. When a target is detected in your application, this will be reported in the API.

Cancel
Add

Figure 57. Setting Target parameters in Vuforia

The user should next click on “download Database(All)”, select Unity Editor and download (Figures 58 and 59).



### Download Database

1 of 1 active targets will be downloaded

Name:  
IoT

Select a development platform:

☐ Android Studio, Xcode or Visual Studio

☒ Unity Editor

Cancel

Download

Figure 58. Download Vuforia Database to Unity Editor

### Compiling Database



Compiling a database with Object targets may take several minutes

Cancel

Figure 59. Compiling Vuforia database

Last, the user should double click on the package downloaded and import it in Unity (Figure 60).

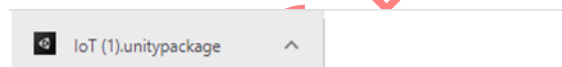


Figure 32: Example image

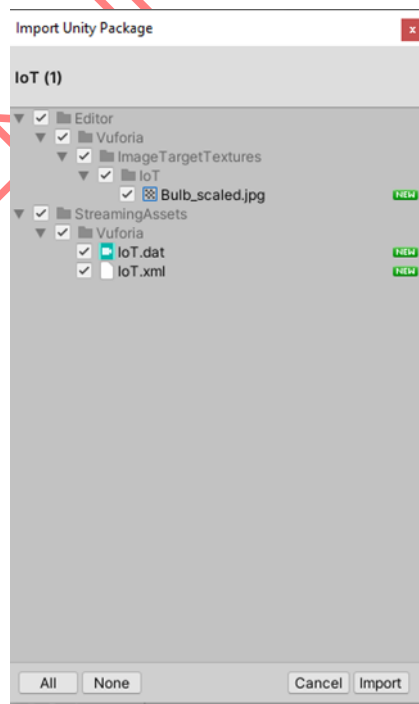


Figure 60. Import package to Unity

**STEP 2: Add screenshot above target image in Unity (AR)**

First, the user must open the scene and delete the main camera (just select it and press the delete key), as in Figure 61.

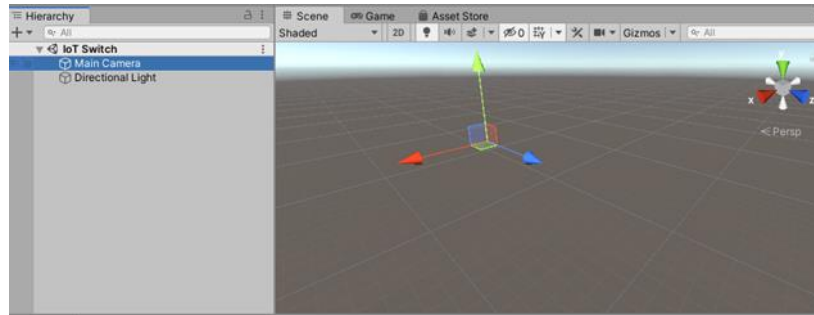


Figure 61. Delete main camera on Unity

Then, the user should add the AR Camera from Vuforia Engine (Figure 62).

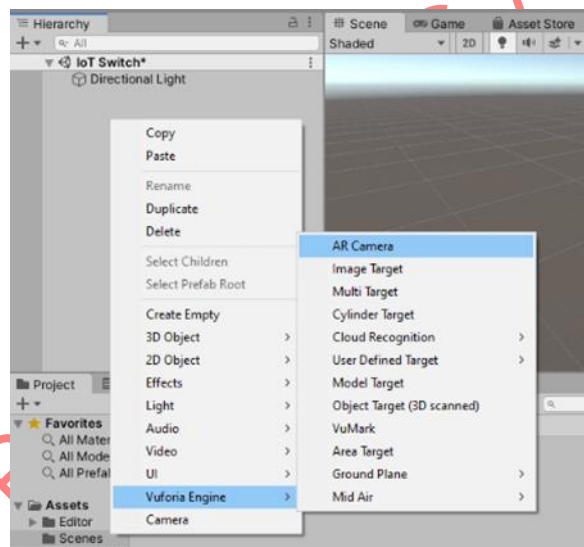


Figure 62. Add AR Camera from Vuforia Engine

Next, the user should add an image target from Vuforia Engine (Figure 63).

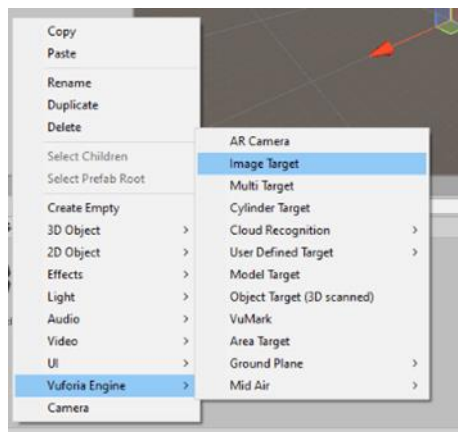


Figure 63. Add image target from Vuforia Engine

The user should select "Image target" and, in the inspector panel on the right side of the screen, load the image from database (Figure 64).

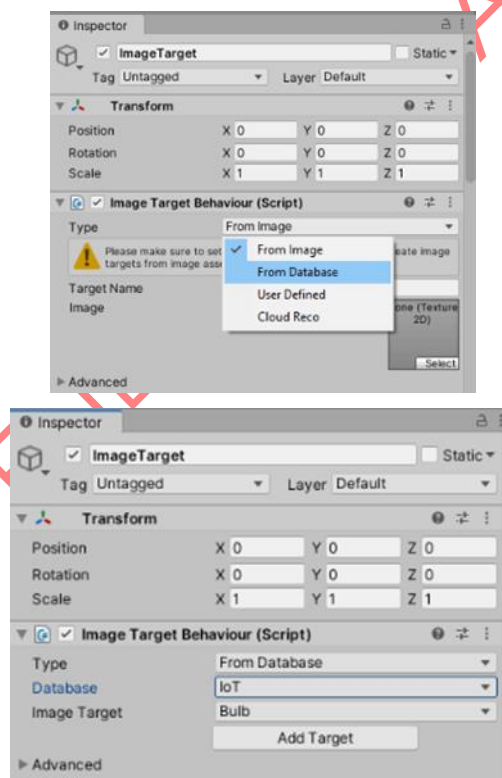


Figure 64. Load Image from Database

Once done, the screen will look something like this (Figure 65):

## D4.4 - Enhanced IoT Tactile &amp; Contextual Sensing/Actuating (Final Version)



Figure 65. Unity screen after having loaded Image from Database

The user should add a button from the UI. The user should right click on “ImageTarget” and select for example “button” from UI or another selection (Figure 66).

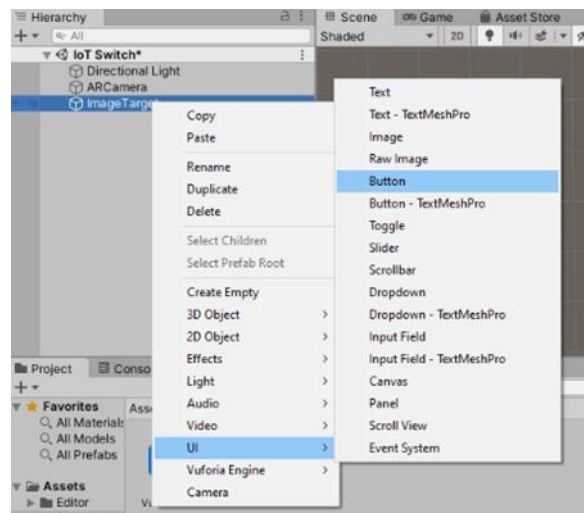


Figure 66. Add a button from UI

The user should click on “canvas” and change “Render Mode” to “World Space” (Figure 67).

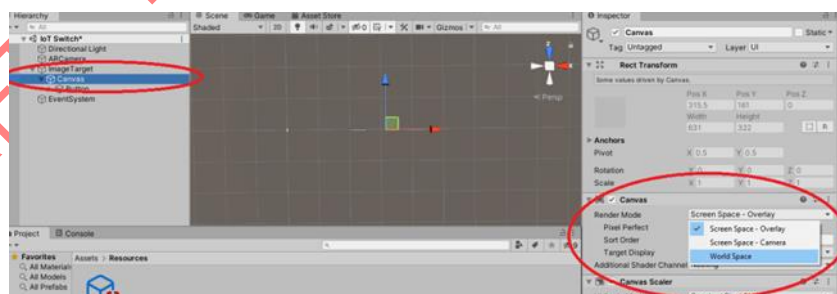


Figure 67. Change Render Mode to World Space

The user user should then set “Event Camera” as AR camera (simply dragging and dropping), as in Figure 68.

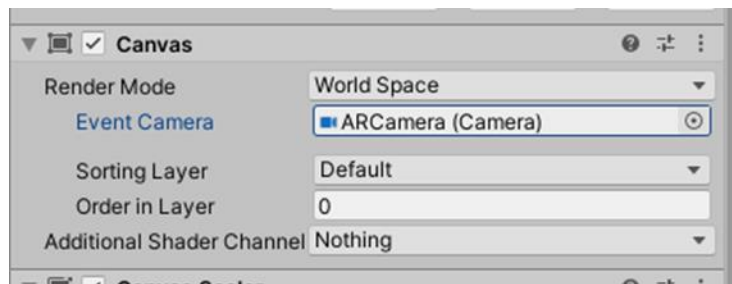


Figure 68. Set Event Camera as AR camera

The user should reset the canvas position to (0,0,0), so that the AR image seen on the mobile phone display (e.g., buttons) is in the same position as the target image (Figure 69).

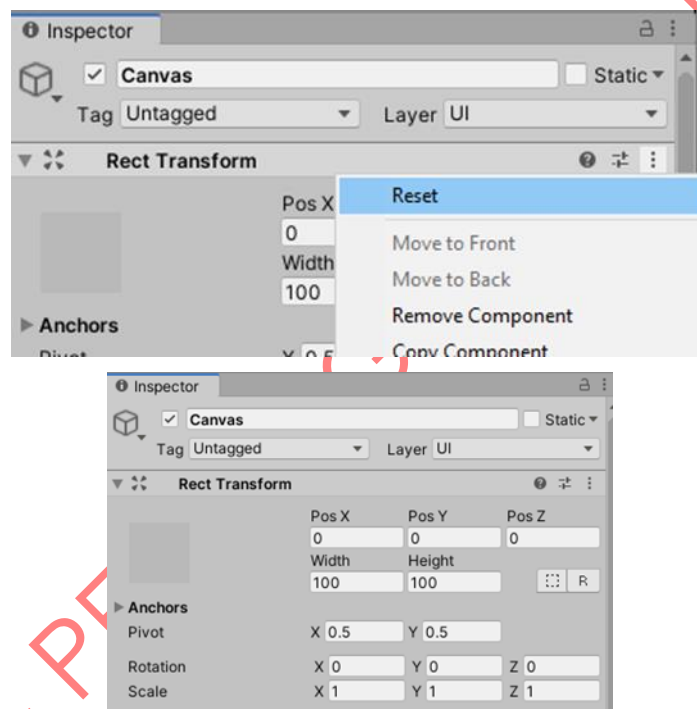


Figure 69. Reset the canvas position

Next, the user should adjust the "screen" of AR over the Target image. The user should rotate and scale to fit like this.

So now, if the target image is run and brought in front of the computer camera, the "screenshot obtained via AR" will be seen above it (Figure 70).



Figure 70. AR on QRCode

### **STEP 3: Script for Connection API to IDI services**

The user should write a script to get the information especially related to the IDI by creating HTTP requests.

The user needs a code that creates a request to the APIs related to the IDI component every time they frame for example the QRCode (The same is obtained for the invocations to the APIs related to the IoT Device Discovery),

First, inside the assets folder, the user should create a new folder "Scripts", and inside this folder create a new C# file with the example name "ClickUrl" or "CallServices" (Figure 71).

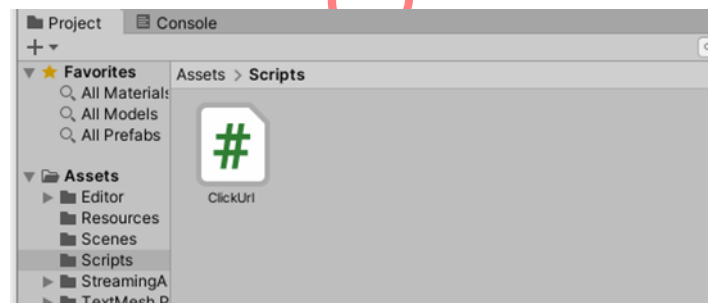


Figure 71. Create a new C# file

The user should add the following code to the C# file.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;
using TMPro;
using UnityEngine.Networking;

public class VBTN : MonoBehaviour
{
    public VirtualButtonBehaviour Vb;
    public GameObject window;
    public TextMeshPro olive_fuit;
    // Start is called before the first frame update
    void Start()
    {
        Vb.RegisterOnButtonPressed(OnButtonPressed);
    }
}
```



```

Debug.Log("start");
window.SetActive(false);
}

public void OnButtonPressed(VirtualButtonBehaviour Vb){
    Debug.Log("tes pressed");
    GetData();

    olive_fuit.GetComponent<TextMeshPro>().text = "Olive fruit fly [stage]:" + 64 ;
    window.SetActive(true);
}

void GetData() => StartCoroutine(GetData_Coroutine());

IEnumerator GetData_Coroutine()
{
    string url = "https://keycloak.iotngin.lab.synelaxis.com/realms/iot-ngin/.well-known/openid-configuration";

    using (UnityWebRequest request = UnityWebRequest.Get(url))
    {
        yield return request.SendWebRequest();
        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.Log(request.error);
            Debug.LogError("errore");
        }
        else
            Debug.Log(request.downloadHandler);
    }
}
}

```

The user should save the code and return to the Unity screen. Now they link this script to the AR image. (Just by dragging and dropping) (Figure 72)

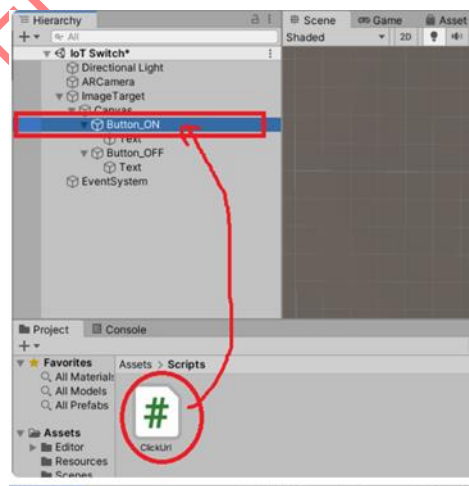


Figure 72. Link a script to an AR image

The various components of the Unity project created for this use case are displayed in Figure 73.

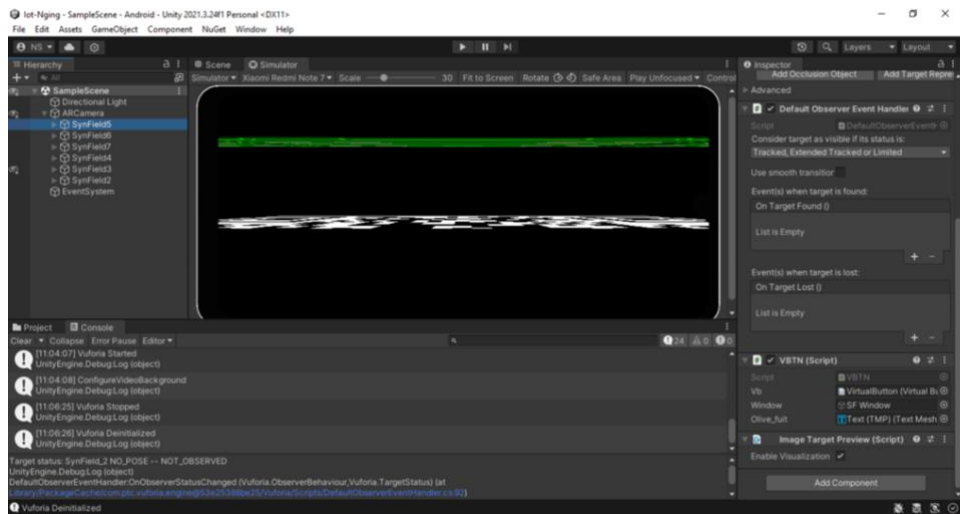


Figure 73. Unity Components for the Project

#### **STEP 4: Build an .apk file and test it on an Android Device**

Finally, It is time to **build** the **Android app**

The user should go to File > Build Settings (Figure 74)

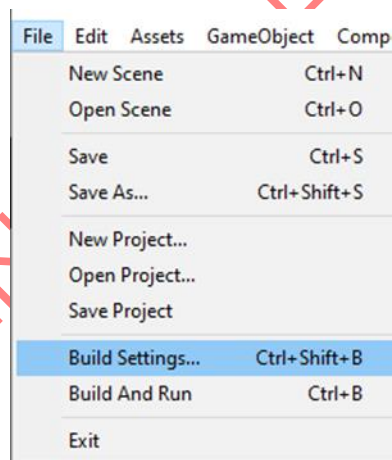


Figure 74. Build Settings

The user should switch to Android. [Click on Switch Platform] (Figure 75)

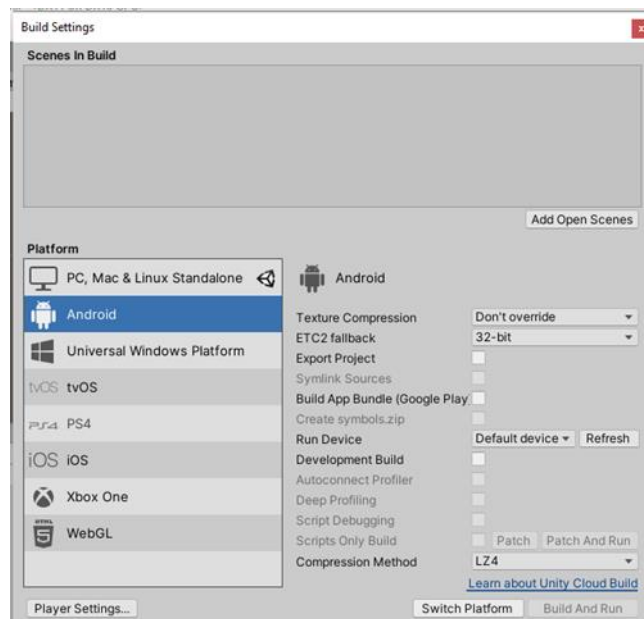


Figure 75. Choosing Android in the Build Settings

The user should click on *Player* settings and then add the preferred details / info. App icon can also be changed from here (for example Figure 76).

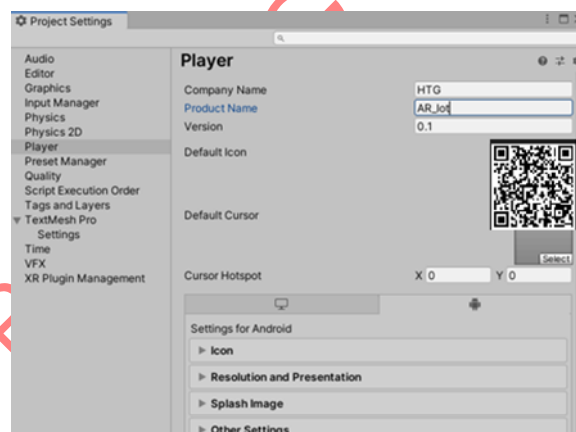


Figure 76. Setting Player Parameters / Info

Then, the user should click on "Build" (Figure 77).

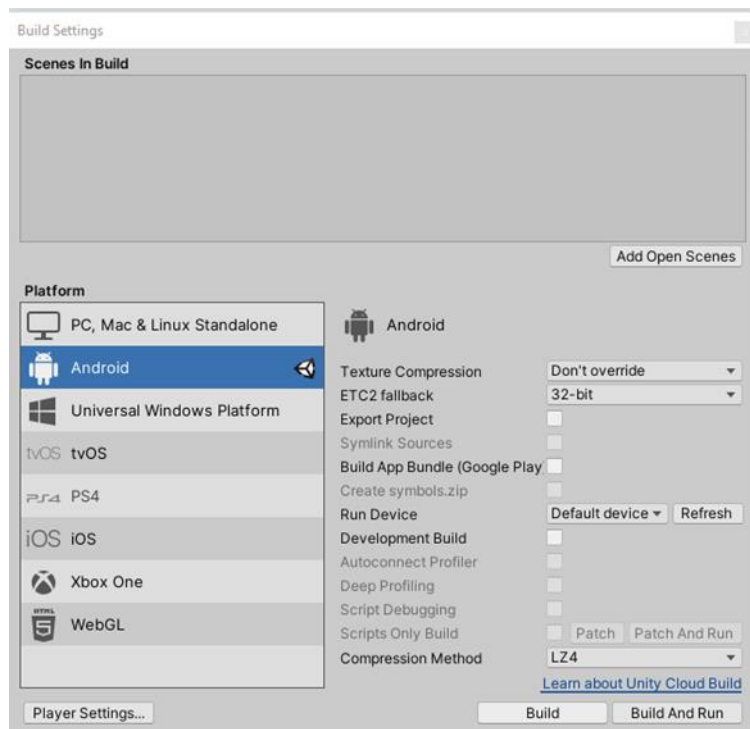


Figure 77. Creating a Build

Finally, the user should give a name to the .apk and save it.

It will start compiling and finish after some time. Once finalized, the .apk will be created (Figure 78).

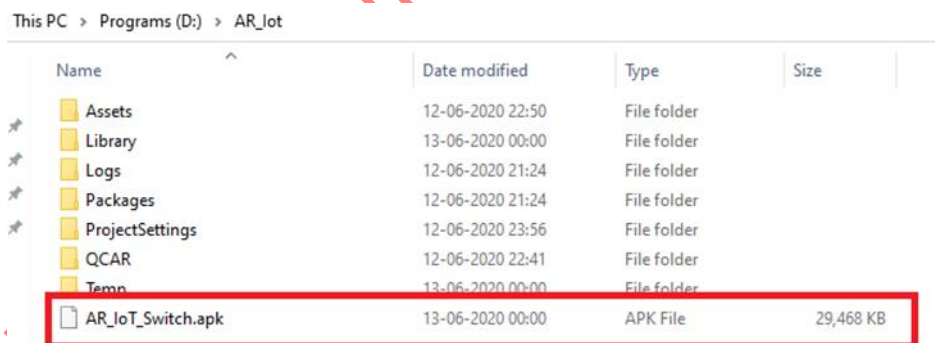


Figure 78. .apk created

### **Install and run the .apk**

Finally, the .apk must be transferred to the mobile devices, installed and run (Figure 76).



Figure 79. .apk installed and running

DRAFT - PENDING EC REVIEW

## 6 Conclusions

This deliverable has provided an update about the technical work and outcomes related to the design and implementation of components, methods, and tools to provide ambient intelligence and contextual sensing/actuating in the scope of the EU H2020 IoT-NGIN project.

The document represents the fourth deliverable within the scope of WP4 and, as originally planned, it updates D4.3 [3], provides final versions of the object recognition methods, and complete implementations of the AR modules and apps to support the envisioned IoT-NGIN use cases.

After providing a comprehensive state-of-the-art review on the IoT-AR field (Section 2), the deliverable has reported on the final specification for the IDI, IDAC and IoT-AR modules (Section 3). Then, the deliverable has reported on the evolved versions of visual recognition / IDD methods, as well as on the new development of AR tools and methods to support the envisioned use cases: Smart Agriculture, Industry 4.0, and Smart Energy. Finally, the deliverable has reported on the installation and usage guidelines for every new and updated component and tool described in it.

All such components, methods and tools should not only be conceived as key ad-hoc enablers for the envisioned use cases, but as modular and interoperable pieces to support advanced and context-aware sensing and actuating in the scope of ambient intelligence and tactile Internet, via intuitive, natural, and user-friendly multimodal AR interaction methods. In this context, a set of open-access IoT-AR resources (libraries, components, whole projects, tutorials, courses...) are shared in Annex 1, and on public Gitlab repository [1], to let third-party entities leverage the outcomes of the project and further advance in this field.

Future work will be targeted at making use of WP4 component to support the Living Labs as well as at deploying and running the associated pilot actions as part of the envisioned use cases.



## 7 References

- [1] IoT-NGIN, "Wiki with repository of open-access IoT-AR Resources," 2023. [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/ar-toolkit/ar-tools-info/-/wikis/home](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/ar-toolkit/ar-tools-info/-/wikis/home).
- [2] IoT-NGIN, IoT-NGIN, "D4.2 - Enhancing IoT Ambient Intelligence", H2020-957246 IoT-NGIN Deliverable Report, 2021.
- [3] IoT-NGIN, "D4.3 "Enhancing IoT Tactile & Contextual Sensing/Actuating"," H2020-957246 IoT-NGIN Deliverable Report, 2022.
- [4] J. C. Kim, T. H. Laine and C. Åhlund, "Multimodal interaction systems based on internet of things and augmented reality: A systematic literature review.," *Applied Sciences*, vol. 11, no. 4, 2021.
- [5] S. Lanka, S. Ehsan and A. Ehsan, "A review of research on emerging technologies of the Internet of Things and augmented reality," in *In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 770-774, 2017..
- [6] A. Badouch, S. D. Krit, M. Kabrane and K. Karimi, "Augmented Reality services implemented within Smart Cities, based on an Internet of Things Infrastructure, Concepts and Challenges: an overview," in *In Proceedings of the Fourth International Conference on Engineering & MIS 2018 (pp. 1-4)*, 2018.
- [7] D. Jo and G. J. Kim, "AR enabled IoT for a smart and interactive environment: A survey and future directions," *Sensors*, vol. 19, no. 19, p. 4330, 2019.
- [8] A. A. Syahidi, K. Arai, H. Tolle, A. A. Supianto and K. Kiyokawa, "Augmented Reality in the Internet of Things (AR+ IoT): A Review," *The IJICS (International Journal of Informatics and Computer Science)*, pp. 5(3), 258-265, 2021.
- [9] M. F. Alam, S. Katsikas, O. Beltramello and S. Hadjiefthymiades, "Augmented and virtual reality based monitoring and safety system: A prototype IoT platform," *Journal of Network and Computer Applications*, pp. 89, 109-119., 2017.
- [10] P. Phupattanasilp and S. R. Tong, "Augmented reality in the integrative internet of things (AR-IoT): Application for precision farming," *Sustainability*, vol. 11, no. 9, p. 2658, 2019.
- [11] G. C. C. P. A. & C. S. [Whi19] White, "Augmented reality in IoT," in *In Service-Oriented Computing-ICSOC 2018 Workshops: ADMS, ASOCA, ISYyCC, CloTS, DDBS, and NLS4IoT*, Hangzhou, Chin, November 12–15, 2018.
- [12] N. Norouzi, G. Bruder, B. Belna, S. Mutter, D. Turgut and G. Welch, "A systematic review of the convergence of augmented reality, intelligent virtual agents, and the internet of things," in *Artificial Intelligence in IoT*, Cham, Springer, 2019, pp. 1-24.
- [13] S. M. Nizam, R. Z. Abidin, N. C. Hashim, M. C. Lam, H. Arshad and N. A. A. Majid, "A review of multimodal interaction technique in augmented reality environment," *International*

*Journal on Advanced Science, Engineering and Information Technology* , vol. 8, no. 4-2, 2018.

- [14] "Telefonica, "Orion Context Broker," GitHub. [Online]. Available: <https://github.com/telefonicaid/fiware-orion/blob/master/README.md>," 2022. [Online].
- [15] "FIWARE, "IoT Agent," [Online]. Available: <https://github.com/FIWARE/tutorials.IoT-Agent>. [Accessed 2023].," [Online].
- [16] "FIWARE, "Multitenancy," [Online]. Available: <https://thinking-cities.readthedocs.io/en/release-v4.0/multitenancy/>. [Accessed 2023].," [Online].
- [17] "IoT-NGIN, "Historic Data Registry," [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/device-indexing/historic-data-registry](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/historic-data-registry). [Accessed 2023].," [Online].
- [18] "FIWARE, "tutorials.Subscriptions," [Online]. Available: <https://github.com/FIWARE/tutorials.Subscriptions>. [Accessed 2023].," [Online].
- [19] "IoT-NGIN, "Inactivity Checker," [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/device-indexing/inactivity-checker](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/inactivity-checker).," [Online].
- [20] "IoT-NGIN, "IoT Device Indexing," [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/device-indexing](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing). [Accessed 2023].," [Online].
- [21] "Kong Inc., "Kong Gateway," 2022. [Online]. Available: <https://konghq.com/products/api-gateway-platform>," [Online].
- [22] "IETF OAuth Working Group, "OAuth 2.0," [Online]. Available: <https://oauth.net/2/>," [Online].
- [23] "Y. Sheffer, D. Hardt and M. Jones, "JSON Web Token Best Current Practices," Internet Engineering Task Force (IETF), 2020," [Online].
- [24] "IoT-NGIN, "IoT Device Access Control," [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/access-control](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/access-control). [Accessed 2023].," [Online].
- [25] "IoT-NGIN, IoT-NGIN, "D7.3 - IoT-NGIN Living Labs use cases intermediate results ", H2020-957246 IoT-NGIN Deliverable Report, 2023.," [Online].
- [26] "Tensorflow, "EfficientDet D1 640x640," 2020. [Online]. Available: [models/tf2\\_detection\\_zoo.md at master · tensorflow/models · GitHub](https://github.com/tensorflow/models/blob/master/tf2_detection_zoo.md)," [Online].
- [27] "TensorBoard <https://www.tensorflow.org/tensorboard> [Accessed 2023].," [Online].
- [28] "COCO Dataset: COCO - Common Objects in Context (cocodataset.org) [Accessed in 2023].," [Online].

- [29] "Vuforia Engine, <https://developer.vuforia.com/> [Accessed in 2023]," [Online].
- [30] "IoT-NGIN's Gitlab. Software for the Smart Agriculture Use Case [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/ar-toolkit/ar-smart-agri.git](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/ar-toolkit/ar-smart-agri.git)," [Online].
- [31] "IoT-NGIN's Gitlab. Software for IDD – Computer Vision [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/ar-toolkit/ar-smart-agri.git](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/ar-toolkit/ar-smart-agri.git)," [Online].
- [32] "YoloX <https://github.com/Megvii-BaseDetection/YOLOX>," [Online].
- [33] IoT-NGIN, "AR in Industry 4.0," GitLab, 2023. [Online]. Available: [https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/ar-toolkit/ar-industry40](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/ar-toolkit/ar-industry40).
- [34] Unity Technologies, "Unity Augmented reality," 2021, [Online]. Available: <https://unity.com/unity/features/ar>.

## Annex 1 Public Repo with IoT-AR Resources

### Introduction

This Annex collects a set of open-access resources that allow rich interaction with IoT sensors and real-world objects via AR applications. These include both own developments from the IoT-NGIN consortium as well as collected third-party resources (e.g., Github projects, tools, online courses) that can become for the interested audience in these topics.

The same information included in this Annex is available on the project's Gitlab [1].

### Resources developed within the project (5)

#### **Resource 1.** AR device positioning method in indoor environments

##### Brief description and purpose of the resource

The AR positioning method for indoor environments is based on tracking where the device is located within a certain range. To achieve this, image tracking methods are employed, meaning that whenever the device tracks an image target, the system knows exactly where the device is located. This method gives very specific and certain positioning of the device. Also, this image tracking method is supported by an additional one based on placing virtual camera above the player (which is the device) in Unity3D engine, and connect the camera movements to the player/user movements. By doing this, the position of the player/user can be known and registered in the IDI, and/or shown in the minimap.

##### Code examples / snippets

```
using UnityEngine;
using Vuforia;

public class IndoorPositioning : MonoBehaviour, ITrackableEventHandler
{
    private TrackableBehaviour trackableBehaviour;
    private bool isTargetDetected = false;
    private Vector3 targetPosition = Vector3.zero;

    void Start()
    {
        trackableBehaviour = GetComponent<TrackableBehaviour>();
        if (trackableBehaviour)
        {
            trackableBehaviour.RegisterTrackableEventHandler(this);
        }
    }

    void Update()
    {
        if (isTargetDetected)
```

```
{
    // Position your virtual object relative to the target position
    transform.position = targetPosition;
}
}

public void OnTrackableStateChanged(
    TrackableBehaviour.Status previousStatus,
    TrackableBehaviour.Status newStatus)
{
    if (newStatus == TrackableBehaviour.Status.DETECTED ||
        newStatus == TrackableBehaviour.Status.TRACKED)
    {
        isTargetDetected = true;
        targetPosition = trackableBehaviour.transform.position;
    }
    else
    {
        isTargetDetected = false;
    }
}
}
```

**Download URL:**

<https://gitlab.com/h2020-iot-ngin/enhancing-iot-tactile-contextual-sensing-actuating/ar-toolkit/ar-tools-info/-/wikis/home>

**Dependencies:**

These prerequisites offer the groundwork for integrating Magic Leap One, Unity 3D, and Vuforia-based indoor AR device placement. To create immersive AR experiences indoors, we use the potent programming environment Unity, the hardware and SDK from Magic Leap, and the image identification and tracking tools from Vuforia.

**Unity 3D:**

- Download and install the Unity 3D program. The version should be 2020.
- Package for Magic Leap Unity: Add the Magic Leap Unity package that Magic Leap has made available to your Unity project. The libraries and tools required for developing for Magic Leap are included in this bundle.
- Import the Vuforia Unity package that Vuforia has given into your Unity project. The Vuforia SDK and tools for marker-based and image recognition augmented reality are included in this bundle.

**Magic Leap One:**

- The actual Magic Leap One AR headset is required for testing and delivering your Unity application.
- Download and install the Magic Leap SDK, which offers the essential APIs and tools for creating AR applications for the Magic Leap One, from the Magic Leap website.
- Package for Magic Leap Unity's Technical Preview: Open your Unity project and import the Magic Leap Unity Technical Preview package. Additional Magic Leap-specific scripts and functionality for the Magic Leap One device integration are included in this bundle.

#### Vuforia:

- Account for a Vuforia developer: Register for an account at the Vuforia Developer Portal.
- License code for Vuforia. From the Vuforia Developer Portal, get a license key for the software. To use the AR capabilities of Vuforia in your Unity project, you must have this key.
- Package for Vuforia Unity: Create a Unity project and import the Vuforia package. The scripts, APIs, and tools required to integrate Vuforia into Unity are included in this package.

#### Installation and Usage Guidelines:

##### Unity3D:

- Installation: Use the Unity website (<https://unity.com/>) to download and install Unity.
- Use: Open Unity after installation, then start a new project or pick one from your library. By choosing the package files under "Assets" -> "Import Package" -> "Custom Package," you may import the Magic Leap Unity package and the Vuforia Unity package into your project.

##### Magic Leap One:

- Installation: The Magic Leap SDK must be installed in order to create applications for the Magic Leap One. To download and install the SDK, go to the Magic Leap Developer Portal (<https://creator.magicleap.com/learn/guides/developer-setup>) and follow the instructions.
- Usage: Launch your Unity project after installing the SDK. By choosing the package file under "Assets" -> "Import Package" -> "Custom Package," you may import the Magic Leap Unity Technical Preview package. This package enhances your project with Magic Leap-specific scripts, settings, and functionality. The Magic Leap One device may then be used in your Unity project by utilizing the Magic Leap APIs and functionalities.

##### Vuforia:

- Make an account on the Vuforia Developer Portal by visiting <https://developer.vuforia.com>
- Vuforia License Key: Log in to the Vuforia Developer Portal after making an account, then go to the License Manager area. Get the license key connected to your project by creating a new license.
- Installation: Launch the Unity project. By choosing the package file under "Assets" -> "Import Package" -> "Custom Package," you may import the Vuforia Unity package. The Vuforia SDK and required scripts for integrating Vuforia with Unity are included in this bundle.



- Use: Enter your Vuforia license key in the "Window" -> "Vuforia Configuration" menu when configuring Vuforia in Unity. To set up image targets, marker-based tracking, or other AR capabilities in your project, go to the Vuforia documentation and tutorials.

## Resource 2. Detection of Visual Target from an AR app

### Brief description and purpose of the resource

Using Unity3D and Vuforia, we are able to detect many image targets with the AR device. Vuforia offers Unity3D applications the tools and APIs necessary to integrate AR experiences. Vuforia provides a powerful picture tracking and recognition solution for image targets.

We can make engaging AR experiences in Unity3D that seamlessly overlay digital material over real-world objects or photos by utilizing Vuforia's image identification and tracking technology. We can concentrate on creating and implementing the AR interactions and content within our Unity project while the Vuforia platform handles the challenging computer vision duties.

### Code examples / snippets

```
using UnityEngine;
using Vuforia;

public class DetectTargets : MonoBehaviour, ITrackableEventHandler
{
    public GameObject arObject; // The AR object to display when the target is detected
    private TrackableBehaviour trackableBehaviour;

    void Start()
    {
        trackableBehaviour = GetComponent<TrackableBehaviour>();
        if (trackableBehaviour)
        {
            trackableBehaviour.RegisterTrackableEventHandler(this);
        }
        GameObject.SetActive(false);
    }

    public void OnTrackableStateChanged(TrackableBehaviour.Status previousStatus,
    TrackableBehaviour.Status newStatus)
    {
        if (newStatus == TrackableBehaviour.Status.DETECTED ||
            newStatus == TrackableBehaviour.Status.TRACKED ||
```

```

        newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
    {
        // Target is detected or tracked, show the AR object
        GameObject.SetActive(true);
    }
    else
    {
        // Target is lost, hide the AR object
        GameObject.SetActive(false);
    }
}
}

```

#### Download URL

<https://gitlab.com/h2020-iot-ngin/enhancing-iot-tactile-contextual-sensing-actuating/ar-toolkit/ar-tools-info/-/wikis/home>

#### Dependencies:

##### Unity 3D:

- The Unity website (<https://unity.com>) offers a download and installation of the Unity 3D program. For your particular operating system, adhere to Unity's installation guidelines.

##### Vuforia:

- Create an account on the Vuforia Developer Portal (<https://developer.vuforia.com>) to use the Vuforia Developer Platform.
- Key for Vuforia license: Log into the Vuforia Developer Portal after making an account, then go to the License Manager area. Get the license key connected to your project by creating a new license.
- Unity Vuforia package: Add the Vuforia Unity package to your project in Unity. The Vuforia Developer Portal has the bundle available for download. Go to "Assets" -> "Import Package" -> "Custom Package" and choose the package file to import. The Vuforia SDK and required scripts for integrating Vuforia with Unity are included in this bundle.

##### Visual Target Database

- Create a visual target database by going to the Target Manager area of the Vuforia Developer Portal. Upload your 3D models or photos as targets to the database. Once you've added targets. Do not forget to activate and download the database.

#### Installation and Usage Guidelines:

1. Using the Vuforia Engine AR package imported from the Unity Asset Store, install the Vuforia package in Unity3D. Create a Vuforia License Key tied to your project and

register as a Vuforia Developer. By selecting Window -> Vuforia Configuration from Unity's menu bar, you may set up Vuforia by inputting your license key.

2. Create the picture targets we'll be using for our AR experience. Image targets are actual pictures or things that Vuforia can identify and follow. Uploading our photos and creating target database files (either a.unitypackage or a cloud database) on the Vuforia Target Manager web page. We insert the target database file that was produced into our Unity project by double-clicking it or by using the Unity importer.
3. Create an empty GameObject and we add the Vuforia ARCamera component to it. This is how we use the Vuforia components in Unity. This element controls the tracking of picture targets and the AR camera. In Unity, we make a new Image Target GameObject. We select the relevant target from our imported database and attach the ImageTarget component to it. To represent the content or augmentations we want to display when the image target is recognized, we add child gameobjects to the image target gameobject.
4. When we start the Unity application on a compatible device, the Vuforia engine makes use of the camera to record the actual scene. Using the target database that has been setup, Vuforia analyzes the camera stream and looks for image targets. Vuforia tracks the location and orientation of an image target after it is identified.
5. Digital material or augmentations can be shown on top of an image target when it is being monitored. In our specific case whenever an image target is tracked by the AR device, an update of the position of the AR device is processed, and we are using this method or overlay some information about some packets inside the factory of BOSCH.

**Resource 3.** API REST endpoints for an AR app to communicate with and retrieve information from the IoT Device Indexing module (IoT agent database developed in IoT-NGIN)

Brief description and purpose of the resource:

The information from IoT sensors is usually stored in an IoT agent database or Device Indexing module. This resource aims to provide a communication channel with such a module. However, in order to eliminate dependences on the existence of such a module, the resource is based on retrieving the information from a CSV file stored on a server, linked to a specific device/sensor.

This module sends request every 0.5 second (configurable) to the database to retrieve the last value published in the CSV file and process it accordingly for visualization / presentation.

GET /devices Positioning:

- Retrieves a list of all IoT devices and positions that have registered.
- Device data returned as a CSV File, including the device ID, name, status, position.

Code examples / snippets:

```
using System.Collections;
using System.Globalization;
using System.IO;
using UnityEngine;
```

```

using UnityEngine.Networking;
using UnityEngine.XR.MagicLeap;
public class NewReader : MonoBehaviour
{
    public GameObject AGV;
    public string csv;
    void Start()
    {
        MLPrivileges.RequestPrivilege(MLPrivileges.Id.Internet);
        MLPrivileges.RequestPrivilege(MLPrivileges.Id.LocalAreaNetwork);
        MLPrivileges.RequestPrivilege(MLPrivileges.Id.LowLatencyLightwear);

        StartCoroutine(DownloadFile());
    }
    IEnumerator DownloadFile()
    {
        while (true) // Loop infinitely
        {
            UnityWebRequest uwr =
            UnityWebRequest.Get("http://84.88.36.130:11000/database/test_data?test_name=pos_prueba_MI
            A_09-05-2333");

            Debug.Log("Downloading file...");
            yield return uwr.SendWebRequest();

            if (uwr.result != UnityWebRequest.Result.Success)
            {
                Debug.LogError(uwr.error);
            }
            else
            {
                Debug.Log("File downloaded successfully!");
                if (uwr.downloadHandler.data.Length <= 0)
                {
                    continue;
                }
            }
        }
    }
}

```

Download URL:

<https://gitlab.com/h2020-iot-ngin/enhancing-iot-tactile-contextual-sensing-actuating/ar-toolkit/ar-tools-info/-/wikis/home>

Dependencies: -

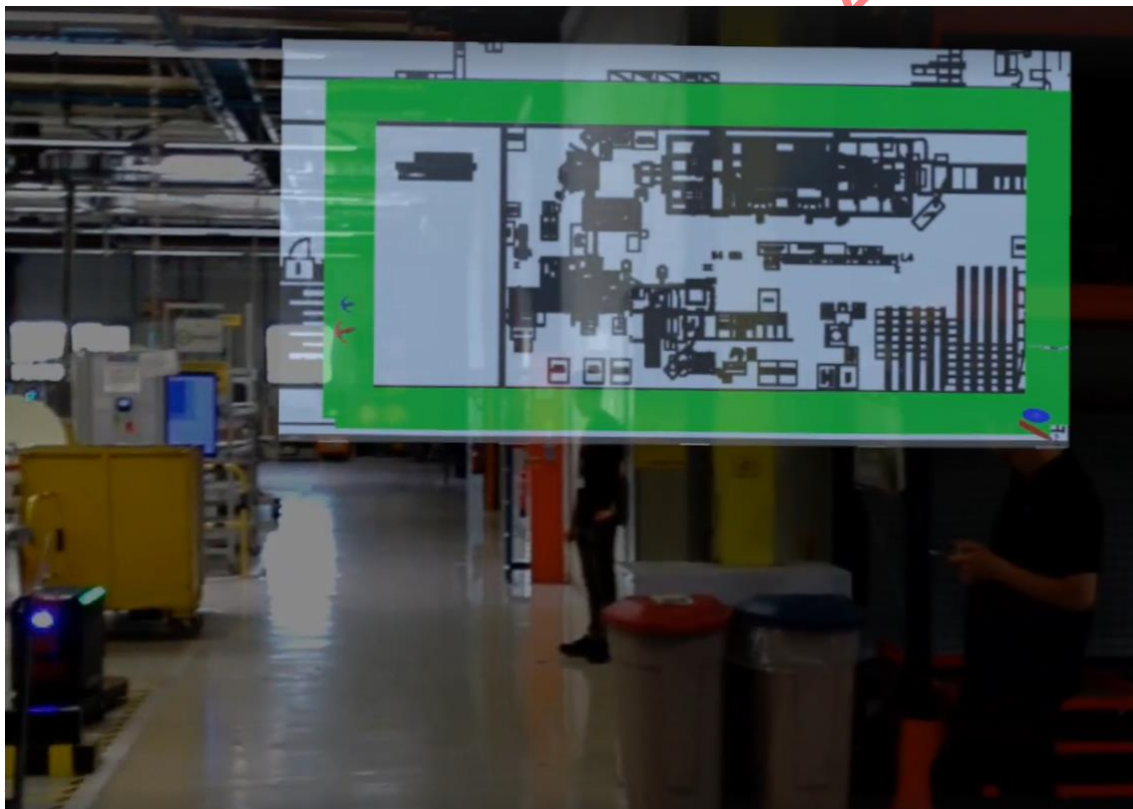
Installation and Usage Guidelines: See the Download URL

**Resource 4.** Display of a mini map of a location on the corner of the field-of-view of an AR app, with the dynamic information and position of all sensors therein

Brief description and purpose of the resource:

The Minimap aims at guiding the user during the AR experience, by displaying all the location of all elements / sensors detected in the physical environment on that mini map. Besides, the mini map is also useful in combination with the indoor positioning method (Resource 1), as the AR app's location can also be displayed, and the information to be shown could be filtered by distance, sensor type and/or status.

Code examples / snippets:



Download URL:

<https://gitlab.com/h2020-iot-ngin/enhancing-iot-tactile-contextual-sensing-actuating/ar-toolkit/ar-tools-info/-/wikis/home>

Dependencies: -

1. Unity.
2. Ascertain REST API endpoints.

3. Use UnityWebRequest to send HTTP queries to the API endpoints, or a third-party networking library, if Unity's networking features are not sufficient.

#### Installation and Usage Guidelines:

- Installation: UnityWebRequest is pre-installed with Unity and does not need to be installed separately, meaning there is no need to install nor import any library to Unity.
- Use: UnityWebRequest is used directly in scripts without the need for any further processes.

**Resource 5.** Alarm system when the distance between any two sensors is below a (configurable) threshold

#### Brief description and purpose of the resource:

An alarm system is designed and implemented inside the Unity3D app targeting the AR device, where it starts pumping some visual alarm messages in the field of view of the AR device whenever two tracked objects are close (2 m or less). Alarms could be also auditive or via vibration. The purpose of this system is to enhance situational awareness and safety, trying to avoid accidents.

#### Code examples / snippets:

using UnityEngine;

public class DetectCollision : MonoBehaviour

{

    public GameObject object1;

    public GameObject object2;

    public string message = "Objects are near each other!";

    private void Update()

    {

        float distance = Vector3.Distance(object1.transform.position, object2.transform.position);

        if (distance < 2f) // change the value to adjust the proximity threshold

        {

            Debug.Log(Warning!! you are too close to.....);

        }

    }

}

#### Download URL:

[https://gitlab.com/h2020-iot-ngin/enhancing\\_iot\\_tactile\\_contextual\\_sensing\\_actuating/ar-toolkit/ar-tools-info/-/wikis/home](https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/ar-toolkit/ar-tools-info/-/wikis/home)

#### Dependencies: -

- Unity



#### Installation and Usage Guidelines:

- Installation: just integrate this feature into an Unity3D project.
- Use: alarms will be automatically received by AR users.

#### Third-party Resources – Open-source projects (7)

##### **Project 1:** A location-based AR app for monitoring outdoor IoT sensors

**Summary:** Location-based mobile AR application for assisting professionals in conducting daily inspections on outdoor IoT sensors. Among other relevant features, the IoT-AR app allows for: (i) registering new sensors; (ii) displaying the locations of sensors and the user on a map; (iii) getting notifications based on.

**Demo:** [https://www.youtube.com/watch?v=pVpWK\\_tICZU](https://www.youtube.com/watch?v=pVpWK_tICZU)

##### **Involved Technologies / Components:**

- Firebase Authentication SDK
- Firebase database
- Unity, and associated plugins
- Vuforia SDK
- MQTT Client Library

**Link:** <https://github.com/VivianKuKu/CASA0022-Dissertation-Augmented-Sensor>

##### **Project 2:** A location-based AR app for monitoring outdoor IoT sensors

**Summary:** Proof-of-concept project that allows controlling distributed electronic switches (e.g., lights) from a smartphone via an AR app

**Demo:** [https://drive.google.com/file/d/1UlpIOXssT\\_Xx49pebvtX2wz1FHbVLv9b/view](https://drive.google.com/file/d/1UlpIOXssT_Xx49pebvtX2wz1FHbVLv9b/view)

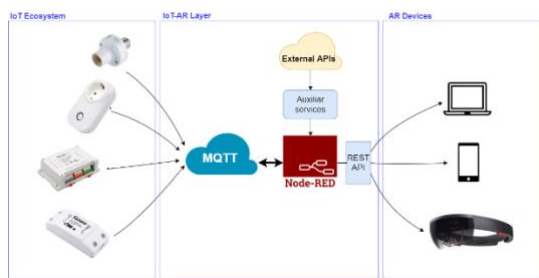
##### **Involved Technologies / Components:**

- IoT Controller (ESP8266)
- Unity, and associated plugins
- Vuforia SDK
- Artificial Intelligence (AI) Recommendation System

**Link:** <https://github.com/TMCheah/Virtual-Switch-Controlling-IoT-Devices-Using-AR>

##### **Project 3: IoT-AR Framework**

**Summary:** Open-source framework that eases the integration between AR apps and IoT devices, as well as the transfer of information among them, in real time and in a dynamic way

**Demo:****Involved Technologies / Components:**

- Node-RED (<https://nodered.org/>): programming tool for wiring together hardware devices, APIs and online services.
- Unity, and associated plugins
- HTTP(S) REST API
- MQTT

Link: <https://github.com/4m1g0/IOT-AR-Framework>

**Project 4: Weather conditions and water level inspection via AR**

**Summary:** Proof-of-concept project to provide real time data about weather conditions and water level from IoT sensors via an AR app.

Demo: -

**Involved Technologies / Components:**

- Arduino IDE
- Unity
- Google Firebase
- Flutter
- Hardware / Sensors: NodeMCU ESP8266 Wifi Module, DHT-11 Humidity & Temperature sensor, Rain Drop sensor and Soil moisture sensor

Link: <https://github.com/Mr-Sushant/IoT-AR>

**Project 5: AR for Preventive Maintenance**

**Summary:** Proof-of-concept project to provide preventive or periodic maintenance for machines (or robotics) via an AR app. It allows for: (i) displaying specifications and related information for an associated machine/piece/object or IoT sensor; (ii) displaying a simulation of how the machine/piece/sensor works or should work; (iii) show how it can be assembled and/or connected to other elements; (iv) provide a checklist to be monitored and validated.

Demo: Check demo videos on the Github page

## Involved Technologies / Components:

- Unity
- Vuforia SDK
- Shapr3 (<https://www.shapr3d.com/>): 3D CAD program

Link: <https://github.com/yudhisteer/Augmented-Reality-for-Preventive-Maintenance>

**Project 6: AR and IoT for Machine Monitoring (Phase II of Project 5)**

Summary: Proof-of-concept project to retrieve dynamic information and parameters (e.g., temperature, humidity) from IoT sensors via an AR app

Demo: Check demo videos on the Github page

## Involved Technologies / Components:

- Arduino IDE
- Unity
- Vuforia SDK
- Google Firebase
- MQTT / HTTP API REST
- Hardware / Sensors: NodeMCU, DHT-11 Humidity & Temperature sensor

Link: <https://github.com/yudhisteer/Augmented-Reality-and-IoT-for-Machines-Monitoring>

**Project 7: QR Code scanner for Unity**

Summary: Simple QR code scanner implementation for Unity utilizing ZXing.Net QR code decode/generator library. Build for Android/iOS/Windows

Demo: -

## Involved Technologies / Components:

- Unity
- ZXing.Net (QR code decode/generator library)

Link: <https://github.com/nickdu088/Unity-QR-Scanner>

Third-party Resources – Courses and Tutorials**Online Course**

- Platform: Udemy
- Title: Internet of Things using Augmented Reality in Unity

- Duration: 43 lessons, around 6h duration

URL: <https://www.udemy.com/course/internet-of-things-using-augmented-reality-and-unity-iotar/>

## Tutorials

### Summary:

- Tutorial 1) How to integrate IoT-AR to get information from an IoT sensor via an AR dashboard, using Unity and Vuforia
- Tutorial 2) How to interact with, and actuate, a switch via an AR app, using Unity and Vuforia, and a NodeMCU through HTTP requests

### Demo:

- Demo 1: <https://www.hackster.io/mithun-das/tutorial-augmented-reality-with-internet-of-things-iot-fb51e7>
- Demo 2: <https://iot4beginners.com/augmented-reality-based-iot-switch-using-unity-and-vuforia>

### Involved Technologies / Components:

- Arduino IDE
- Unity
- Vuforia SDK
- MQTT / HTTP API REST
- Microsoft VS Code
- Hardware / Sensors: NodeMCU ESP8266 Breakout Board - DHT11 Temperature & Humidity Sensor (4 pins) - Breadboard (generic) - USB Camera - jumper wires (generic) - Nodemcu LED with Resistor

Link: Check demo URLs