D5.5 Pledger Applications for the use cases II

Document Identification			
Status	Final	**Due Date**	31/08/2022
Version	0.6	**Submission Date**	12/09/2022

Related WP	WP5	**Document Reference**	D5.5
Related Deliverable(s)	D5.1	**Dissemination Level (*)**	PU
Lead Participant	i2CAT	**Lead Author**	August Betzler (i2CAT)
Contributors	HOLO, i2CAT, FILL	**Reviewers**	Lara Lopez (ATOS)
			Francesco Iadanza (ENG)

Keywords:
Use Case Applications, Development, Testing, Validation

This document is issued within the frame and for the purpose of the PLEDGER project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 871536. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

List of Contributors	
Name	Partner
August Betzler	i2CAT
Estela Carmona	i2CAT
Bruno Cordero	i2CAT
Jordi Marias	i2CAT
Nour Fendri	HOLO
Carina Pamminger	HOLO
Verena Stanzl	FILL
Michael Gillhofer	FILL
Tobias Gann	FILL
Lukas Hettmann	FILL

Document History			
Version	Date	Change editors	Changes
ToC	02/06/2022	August Betzler (i2CAT)	Created ToC
0.1	15/07/2022	HOLO, i2CAT, FILL	First round of inputs
0.2	03/08/2022	HOLO, i2CAT, FILL	Second round of inputs, ready for internal review
0.3	25/08/2022	ATOS, ENG, i2CAT	Internal review and addressing comments
0.4	6/09/2022	HOLO, i2CAT, FILL	Version addressing all comments for quality check
0.5	09/09/2022	ATOS	Quality assurance review
0.6	12/09/2022	ATOS	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	August Betzler (i2CAT)	06/09/2022
Quality manager	Carmen San Roman (ATOS)	09/09/2022
Project Coordinator	Lara Lopez (ATOS	12/09/2022

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
3D	three-dimensional
API	Application Programming Interface
AR	Augmented Reality
BSC	Barcelona Super Computing
CAD	Computer-Aided Design
CAM	Cooperative Awareness Message
CPU	Central Processing Unit
DENM	Decentralized Environmental Notification Message
DLT	Distributed Ledger Technology
Dx.y	Deliverable number y belonging to WP x
XAML	Extensible Application Markup Language
GPS	Global Positioning System
GPU	Graphics Processing Unit
HMD	Head-Mounted Displays
HTTP	Hypertext Transfer Protocol
LED	Light-emitting diode
MB	Megabyte
MQTT	Message Queuing Telemetry Transport
OBU	On Board Unit
QoE	Quality of Experience
QoS	Quality of Service
P2P	Peer-to-peer
PDF	Portable Data Format
PLC	Programmable Logic Controller
RAM	Random Access Memory
RDNS	Risk Detection and Notification System
RPC	Remote Procedure Call
SDK	Software Development Kit
Tx.y	Task number y belonging to WP x
UC	Use Case
UI	User Interface
V2X	Vehicle-to-Everything
VM	Virtual Machine
VPN	Virtual Private Network
VRU	Vulnerable Road User
WebRTC	Web Real-Time Communication
WP	Work Package

Executive Summary

This deliverable reports the outcomes of Task (T) 5.1 “Application development”, describing the final application development of the three Pledger use cases (UCs). The three use cases reflect real needs of edge computing resources, thus serving as demonstrators of how Pledger can help to satisfy the needs of the applications.

The three applications implemented in T5.1 implement the functionalities of each UC, namely mixed reality applications on the edge (UC1), the enhancement of global road safety for vulnerable road users (UC2), and the manufacturing of data mining with edge infrastructures (UC3). This document is based on the previous iteration of the T5.1 deliverable, D5.1 “Pledger Applications for the use cases I” [1]. The previous iteration serves as foundation for the introduction and justification of each application development, along with the explanation of the features to be developed and their purpose.

In this second iteration of the deliverable, a dedicated Section is provided for each UC, where the outcomes of the application development are detailed (Sections 2, 3 and 4, for UC1, UC2 and UC3, respectively). For some of the developed application modules, after the submission of D5.1, several new features were added to the application development roadmap to enhance the UC functionality further. For both, the initially defined features and these newly introduced features, the final descriptions are provided. Any relevant testing that has been conducted to assure the readiness of the applications for the validation and pilot phase are reported in this document as well on a per-UC basis. It is elaborated how the key functionalities targeted at the beginning of the project have been implemented for all UCs, sometimes adding more features that were initially planned to improve either the functionality or the integration points necessary for the applications to work with the Pledger platform and to fully benefit from its features. This includes, amongst other things, the definition and implementation of application level metrics that, once collected and exported to the Pledger monitoring subsystem, can be used to apply orchestration policies, such as offloading or scaling of the application modules.

Overall, as also detailed in the Conclusions (Section 5), the expected functionality of the three use cases and the necessary integration points with other Pledger modules was implemented, fulfilling the requirements for the pilot deployment and operation, as well as the final validation and evaluation activities of the project.

This deliverable is directed towards anyone who is interested in the details of the application development and final application status for the three Pledger use cases.

1 Introduction

1.1 Purpose of the document

This deliverable describes the final updates and development performed as part of task 5.1 (T5.1) "Application development" on the three use case (UC) applications. Each application development has been supervised and coordinated by its respective UC leader. Even though the development for each application and the applications per se are completely independent from each other in each UC, the UC leaders integrated feedback and comments from the periodic work package 5 (WP5) meetings in order to consider keys aspects like integration points with the Pledger platform (e.g., application metrics) and how the value of the Pledger platform can be highlighted in each UC application.

With the information provided in this deliverable, the reader will be able to understand the final development steps performed and how the application was tested and validated for the pilot phase of the project.

1.2 Relation to other project work

After the initial definition of the use cases in WP2, the detailed description of the software modules that are used in each UC were described in the first iteration of this deliverable, namely D5.1. The outcomes shown here are directly related to T5.3. "Pilots operation and monitoring" and T5.4 "Validation and Evaluation", given the final version of the applications developed in T5.1 are used for the pilot operation the final validation and evaluation experiments.

1.3 Structure of the document

This Document has an overall of 4 Sections, including this introduction section (Section 1). The Sections 2, 3 and 4 are dedicated to UC1, UC2, and UC3, respectively. Each of these sections briefly recapitulates what each UC application is for and how it is composed. It then presents any relevant updates to the application modules defined in the previous iteration of the deliverable. The outcomes of the validation for each application are provided as well in these Sections. The document closes with the Conclusions in Section 5.

2 UC1: Mixed reality applications on the edge

2.1 Short Application description

UC1's objective is to improve the capabilities of Augmented Reality (AR) solutions by combining them with edge computing technologies and load allocation and optimization tools from the PLEDGER components to provide high-level services in industrial environments. The industrial environment in which the pilots for this use case will be run is the one provided by FILL in UC3.

The application developed in UC1 allows to explore and implement three case studies that integrate machine data into Mixed Reality interfaces and optimize them via edge computing technology:

- Fast Prototyping
- Remote Support
- Training

These case studies have a high demand for the visualization of three-dimensional (3D) Computer-Aided Design (CAD) Models as holograms. However, the number of polygons, the size, and the quality of the model represents a challenge for the limited computational power of the smart glasses. Therefore, outsourcing these computations to an external source like an edge device coupled with the adequate resource availability dramatically boosts the performance, enabling up to 50 times larger data visualization and resulting in a high level of detail and resolution.

To ensure the enhancement of user experience and functionality of the three case studies, UC1 implements its own remote rendering solution - Interactive Streaming for Augmented Reality (ISAR) - alongside PledgAR Workspace which is a feature rich application providing useful tools for each of the case studies. The Pledger Framework itself enhances the overall functionality and flexibility of PledgAR Workspace and solves several pain points. Those will be addressed in the paragraphs below.

2.2 Final report on application modules

2.2.1 Module: ISAR

ISAR Software Development Kit (SDK) is a cross-platform remote application rendering solution that enables streaming of AR applications, used as a plugin by the PledgAR Workspace. Using the SDK allows the visualization and interaction with high-polygon content by making use of edge computing resources. Furthermore, this is a tool with an ever-growing list of features. The following features are supported:

- **Data transmission**

  Application data is streamed via the Web Real-Time Communication (WebRTC) protocol and received by the Client application on the smart glasses.
- **Control input Transmission**

  As the application is running on a Virtual Machine (VM), the control input, i.e. hand input, must be transmitted from the smart glasses to the application.
- **Audio Transmission**

  PledgAR Workspace works with audio cues. Audio support is the result of development efforts related to the ARtwin project [2].
- **Bandwidth Control**

  Allowing the limitation or definition of bandwidth through Application Programming Interface (API) call. Bandwidth has an impact during the encoding and decoding phase and therefore directly affects the user experience.
- **Multiplayer Session Support**

  During a multi-player session input and data must be shared across all participants on runtime. This feature had to be added on both the application side, as well as ISAR side.

- **Performance Improvements** (e.g., encoding, streaming)
  General optimization of the way data will be transmitted.

2.2.2 Module: Distributed Ledge Technology (DLT) - WHISPER Protocol

In the previous report about the UC1 application development, the use of DLT to enable security and trust was motivated and the basic structure of the DLT-based solution was presented. During the application development and testing the possible ways to use DLT and which data to secure were evaluated. Among the options evaluated, the protection of the data stream or the user management were considered. However, these were dropped due to a decrease of performance due to added delays (when using DLT to secure data stream) or due to undesired shifting of the development focus from the actual use of a blockchain-based solution towards implementing a user management system. Eventually, a concept on the use of DLT was defined that was both useful and would not take a toll on the performance of the application. The designed use is the securing of the signaling server which is needed during the handshake – when the Application and the Client Application establish a connection – through the DLT component:

ISAR allows the Augmented Reality Head Mounted Device (AR HMD) (e.g., HoloLens 2) to make use of the resources of a more powerful edge device. The establishment of this peer-to-peer (P2P) connection requires an important handshake phase known as signaling. During signaling, peers exchange sensitive data relevant for the establishment of a secure communication channel. However, common methods utilized for signaling don't provide enough data security and are susceptible to malicious hacker attacks. Therefore, a private DLT running the Whisper protocol - based on Ethereum – is created and hosted at Holo-Light's premises. This private DLT enables secure messaging. The security of Whisper (Figure 1) is further detailed in deliverables D4.4 [3] and D4.5 [4].

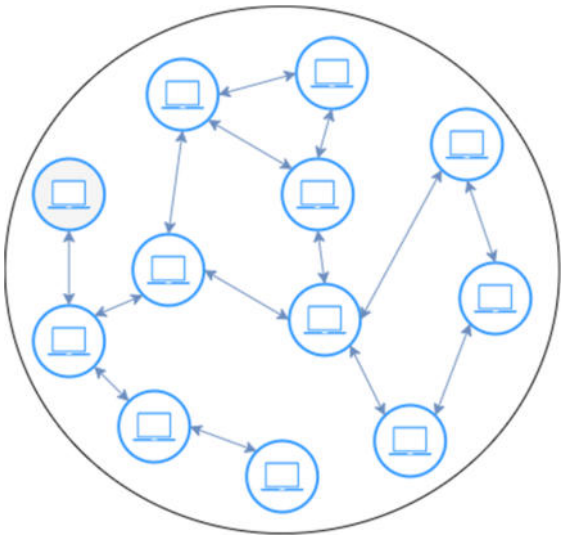

Figure 1: Private Whisper DLT topology with a set of nodes connected to each other

Since the last iteration of the deliverable (D5.1 [1]), we used Geth (Go Ethereum) to create a private Ethereum blockchain with 5 nodes, each running with the Whisper flag (-shh) activated. The nodes are represented as Hyper-Text Transfer Protocol (HTTP) endpoints that only authorized parties (i.e., Virtual Private Network (VPN)-protected) can connect to. The hosted DLT permits users to establish connections with its nodes and utilize all the functionalities provided by the Whisper protocol using JSON-Remote Procedure Call (RPC) requests. Further, an Extensible Application Markup Language (xaml)-based client application was developed for the HoloLens to enable the secure signaling on the user's side.

The 1st Prototype of the DLT-based solution was demonstrated during the 1st review meeting and the final version, along with its integration, will be validated as part of the pilot execution.

2.2.3 Module: PledgAR Workspace

PledgAR Workspace is a Unity3D based application that runs on an edge resource. The application provides the adequate features that enhance Training, Fast Prototyping, and Remote Support scenarios. The following features have been added since the last iteration phase:

- **Hand Collision Detection**: There are three types of hand collision (i.e., Hand Outline, Hand Outline+, and Hand Intersection). Each of these options provide different visual effects to alert the user of a collision with the model. The type will depend on the user's needs.

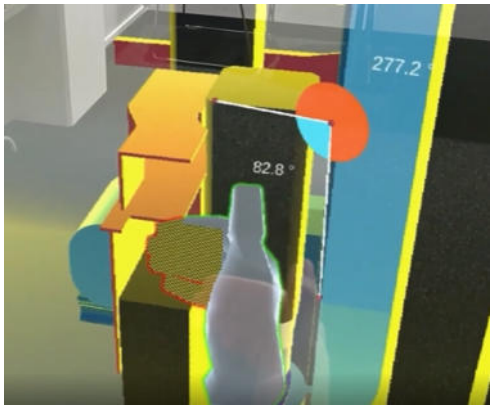

Figure 2: Hand Collision Detection

- **CAD Model Visualisation**: Using the application a user can import any CAD model and have it visualised and manipulated within their AR space. This feature is an extension of the original one, which supported the visualization of ".jt" and ".hlcad" files.
- **Multi-Media Player**: This feature permits users to load and display videos and portable data format (PDF) files within the AR space.

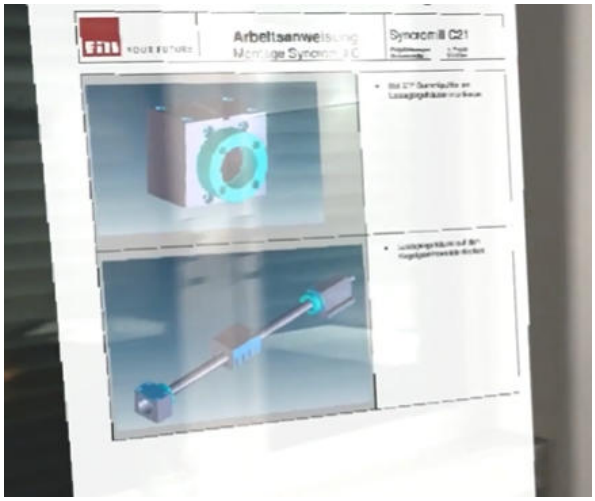

Figure 3: Multi-Media Player

- **Simplified Hand Menu**: The menu allows users to have a quick access to various CAD manipulation functionalities (e.g., grabbing, rotation, scaling, moving, etc) by flipping their hand and selecting the functionality they need.
- **Annotations – Measurements:** A user can measure angles and distances within their AR space. These measurements are realistic with a maximal offset of 5mm. These measurements stick to the selected area, can be saved and, therefore, read at any time. When no longer needed a user can simply delete them from the space.

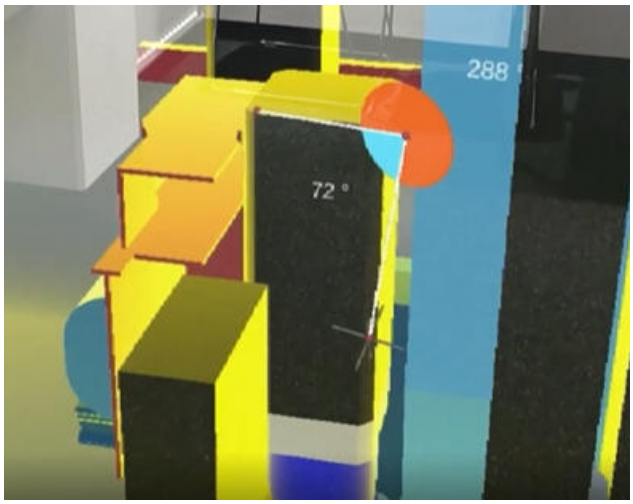

Figure 4: Annotations - Angle Measurements

- **Explosion View**: This functionality provides the user the ability to separate all available parts of a complex model at once. Furthermore, it supports multiple options to set the separation distance.
- **Multi-Player**: Allows multiple users to share a session and collaborate on the same models and render their changes in real-time.
- **Object Placement**: 3D objects can be placed freely in space and re/positioned after all relevant parts have been selected through the Hierarchy Tree.
- **User Interface (UI) Improvements**: Most users are new to the technology, hence, making the UI more user friendly and intuitive is key. Learnings have been used and improvements implemented.
- **Hand-Pose Capture**: Mostly used for training and allows users to capture a certain hand pose and leave it within the space to create useful training scenarios.

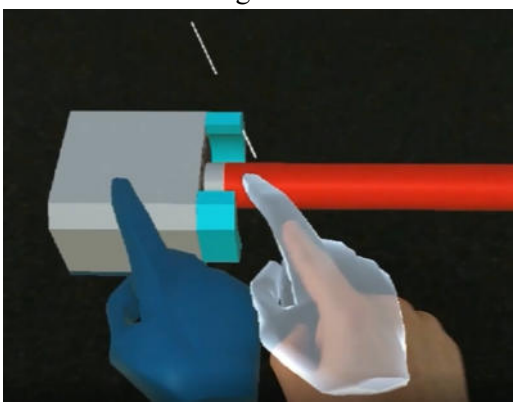

Figure 5: Hand-Pose Capture

2.3 Deployment and validation outcomes and observations

PledgAR Workspace has been fully developed. All functionalities necessary to enable the 3 planned scenarios (Fast Prototyping, Training and Remote Support) are supported. During the development period we reached many achievements, but also ran into some lasting issues. This section addresses what features of the application were successfully deployed, how they have been tested and what observations we have made.

In order to enable the integration with the Pledger Framework, several design choices for the integration endpoints with the DLT and the orchestrator were made that impacted the application development: for the DLT, the necessary application development for securing the signaling (handshake) for the client

connectivity has been developed and the private DLT has been implemented, capable of featuring the Whisper protocol. Further, the exportation of metrics to the Recommender was another important development point: Developing the integration points for the interaction with the Recommender is crucial to allow the autoscaling scenarios needed by the UC1. During the development process multiple options for metric-gathering were addressed, starting from measuring the click count of the Central Processing Unit (CPU), file loading time, Graphics Processing Unit (GPU) usage, and more, in order to identify those affecting the Quality of Experience (QoE). After extensive research and many unsuccessful prototypes, most metrics could be ruled out due to technical limitations. What finally did work was the measurement of RAM availability on a VM supporting GPU pass through, running the operating system Windows 10 and connecting to the Recommender via a Kafka integration. This integration is a small script which pulls the RAM values from the hardware and transmits it in a specific frequency to the Recommender. With further development of the ISAR module an additional metric was introduced, bandwidth control. Here a script monitors decoding and encoding values to calculate latency. These values are then sent to the Recommender as well as the benchmarking tools for the next steps.

Over the span of the project, the full functionality of the UC1 application has been validated. A full test cycle consists of approximately 100 small to large scale test scenarios, across categories such as performance tests, network tests, feature tests, UI tests, hardware tests, just to name a few. The development of project specific app features or integration endpoints required multiple functional tests to ensure proper functionality. Here stress tests, and general functional tests have been conducted. Prototype tests were the basis for further development decisions, and they have been conducted all through the development process.

8 different types of VMs have been validated until the final solution using Linux and QEMU was eventually found, capable of running the application services. The discarded options failed due to the GPU pass-through requirement, or the compatibility with the PledgAR Workspace, or ISAR module.

Stress testing was used during the metric definition for the Recommender, as well as for the Multi-Player feature. For the latter, 10 developers were equipped with a HoloLens, using PledgAR Workspace in a staged prototyping session. This stress test was done with different network limitations and the results used to improve the performance of the ISAR module. During the in-house development and testing, as well as the FILL-site testing phase of the application a series of observation was made, concerning both the functionality of the application, but also the integration points developed as part of the application. In the following, we briefly list these observations:

2.3.1 Observations for the PledgAR Workspace

- Performance
  - Depending on the network performance (Wi-Fi), latency in the multi-Player session was problematic. Latency can decrease the resolution which puts a strain on the user's eyes. This resulted in the definition of the bandwidth scenario.
- Functionality
  - During an interaction, the hands do not render at times. This issue was linked to CPU performance issues and the original driver for looking into CPU as a possible metric to be exported.
  - There is a slight off-set between the real hands and the captured hands. Investigation is ongoing but will likely depend on AR reprojection and result in further optimization.
- Pledger Framework
  - Using the orchestrator simplifies the increase and decrease of bandwidth greatly and makes it efficient through automation. The user does not have to manually apply the changes anymore.
- Security
  - Some components would require a VPN access which we cannot provide, due to insurance related limitations. Hence, the setup of a separate and secure network is required.

2.3.2 Observations for the Integration endpoints with Pledger

- The Orchestrator automatically launches the VM instance and the UC3 application efficiently.
- Using Kafka as message broker was simple and effective. Metric values are transmitted accurately and speedy allowing the proper functionality without causing delays or an increase in latency.
- Using the Whisper protocol to secure the handshake does not add latency during the connection process and secures the vulnerable data.
- If the usage of available RAM is exceeded, the application has to re-start, which is at present a poor user experience. The behaviour of the application itself would have to be adjusted to allow a convenient RAM usage, but this is not a minor endeavour and not within the scope of this project.
- Finding a VM host that supports GPU pass-through took a considerable effort, which was not anticipated.

Any users or adopters of the mixed reality application require the physical setup to be in place. The setup includes the Hololens 2, a Linux-based PC (or laptop) with a powerful GPU (i.e.: Nvidia RTX 3060), a stable and high throughput end-to-end network connection and ideally a power bank with a USB-C cable to charge the HoloLens 2 with. This because, the HoloLens 2 has a limited battery life roughly 1 hour of constant usage. The VM and Client Application must be running so that they can connect to the Pledger Framework.

3 UC2: Edge Infrastructure for Enhancing the Safety of Vulnerable Road Users

3.1 Short Application description

The application developed for UC2 can be broken into several pieces of software. First, the core functionality is implemented by the Risk Detection and Notification Service (RDNS), a service that is deployed in the compute infrastructure and which is managed and orchestrated by Pledger. It processes the location data obtained from the on-board units (OBUs) carried by vulnerable road users (VRUs) like bicycles and scooters and from the tram detection service, determining when there is a possible risk situation. If such a situation is detected, notifications or critical warning are sent, respectively, depending on whether the VRUs are far from or close to the tram station, where the accidents between VRUs and pedestrians entering or leaving the tram can happen.

In order for the OBUs mounted on VRUs to be able to connect to the infrastructure, to share their location and to receive warnings, a vehicular software stack is deployed in UC2, both on the OBUs and the infrastructure. This software stack is necessary to enable the vehicle-to-everything (V2X) communications protocol and is referred to as V2X Stack. On the infrastructure side, road-side units (RSUs) are present, radio nodes that are managed by the Pledger platform. The design of the physical OBUs and the RSUs is part of the application development, given the hardware has been specifically curated and composed for this use case.

The tram detection service is the last component of the UC2 application. Two options for the implementation of the tram detection have been considered throughout the application development phase: i) in a collaborative effort with Barcelona Super Computing Center (BSC), dedicated cameras using image processing and object recognition are deployed and an interface is designed to allow for this information to be injected to the RDNS ii) an OBU is deployed in the tram, just like for the VRUs, allowing to communicate with the RSUs and to share its position. Eventually, the second option, i.e., deploying an OBU in the tram was chosen. The reasoning behind the choice is explained in the dedicated subsection below.

In the following subsections, the final report on the development of these application modules is given. Note that at the point of writing the previous iteration of the deliverable, the core functionality was already implemented and reported. As such, some components, like the V2X Stack, don't have any updates to report.

3.2 Final report on application modules

3.2.1 Module: V2X Stack

As reported in D5.1 [1], the software implementing the vehicular communication stack was already fully developed in M20, as such, no updates are to be reported.

3.2.2 Module: Tram Detection Service

For the most of the duration of the application development task (T5.1), the UC2 team assumed that integration would happen between the Barcelona Supercomputing Center (BSC, external to the project) and the UC2 partners, which would allow us to use their image processing to share the location of a tram with the RDNS when it enters the tram station. As such, in the RDNS module, an API was implemented that is capable of receiving the data format used by BSC to communicate the location information of the trams detected by their cameras. However, due to several delays and the lack of hardware (cameras) to be deployed on-street from BSCs side, the testing and integration phase came to a halt. Since this collaboration from BSC side was approached in a voluntary, "best effort" way, there was no way from Pledger's side to push for this integration further. As such, we proceeded with the backup option that

relies on deploying an OBU in a tram to detect when a tram approaches the station, just like for the bicycles and scooters.

To implement the approach using an OBU to detect the tram, we contacted with *Tram*, the Barcelona tram operator entity, and obtained a permission for the on-street (and "in-tram") testing, after also getting the approval to use the V2X radio technology inside the tram. In these tests, the detection range was tested and it yielded good results (the tram can be detected up to 200 m before arriving at the station).

In order for the RDNS to be able to differentiate OBUs carried by VRUs and the ones being deployed inside the tram, different identifiers are used: the Cooperative Awareness Message (CAM) used to periodically disseminate the position of each OBU has a field specifying the type of station, differentiating VRUs and trams, but which could be potentially more types, if required.

3.2.3 Module: RDNS Application

In M20, the full risk detection and notification functionality was already implemented in the RDNS (see D5.1 [1]) and it was showcased in the UC2 application demo presented during the first interim review. In the last phase of the application development task, the RDNS application was enhanced to feature a set of application metrics, with the goal to be able to export these metrics to the monitoring subsystem and to define thresholds and actions to be taken (both for the decision making and the Service Level Agreement modules).

In total, three metrics have been defined and implemented in the RDNS. These metrics are key indicators that allow us to determine whether the application is operating normally or if there are issues that could be caused by the lack of compute resources for the application. Note that in general in UC2 and the rest of use cases the radio and networking infrastructure is considered to be ideal. I.e., other issues, e.g. packet losses due to bad fibres or interference in the radio access are assumed to be absent, given these resources are not orchestrated by the platform and the platform cannot act upon such issues.

In the following, we describe the three metrics and explain briefly how they have been implemented.

- **Delay**: We consider the key application metric for UC2 to be the end-to-end delay between the OBUs and the RDNS. To obtain the end-to-end delay we measure the time it takes for packets sent out by the OBUs to reach the RDNS, where they are processed.

  In order to calculate the delay, timestamps are inserted at the sender-side into the V2X packets. The RDNS compares the timestamps to the local time when the packet is received. The VRUs are equipped with Global Positioning System (GPS), providing a very reliable time reference. On the RDNS, that runs in the compute infrastructure, the local time is obtained by a local network time protocol service, which delivers a high accuracy with an error of less than 10 ms. While this value is not ideal and introduces a slight error margin, it could be further improved with GPS[1].

  Whenever the application is operating normally, the delays are in the tenths of millisecond range, a sum of the radio transmission duration and the delivery of the message from the RSU towards the RDNS application. However, if compute resources available to the RDNS run low and the application starves, the delay increases noticeably, up to hundreds of milliseconds or more. Such resource shortage occurs, whenever the number of VRUs connected to the system increases considerably (thus increasing the compute load) and at the same time the compute resources allocated to the RDNS are not sufficient. As such, the Pledger platform can use this metric to determine when an upscaling or offloading node of the application to a less constrained compute might be necessary.
- **Packet Drops**: Another metric implemented to detect possible issues suffered by the RDNS application are packet losses. The number of packets lost is calculated by comparing the number of expected packets and the packets that actually are received. Once the RDNS detects new OBUs connected and to receive periodic messages from these stations with a fix rate. I.e. if there are two detected OBUs and the sending frequency is 10 packets per second, then the RDNS expects 20 packets

[1] Given the lab in which the UC is deployed is only a temporary location without GPS coverage, this feature is not available at the time of writing the deliverable.

per second. In the measurements performed with different compute resource allocations for the RDNS application and varying load (number of OBUs connected to the service), this error was not observed, even under high load. However, we assume that in case there are other physical issues in the infrastructure, this application metric could start increasing. This could allow to trigger a notification to the infrastructure provider to check for possible issues in the infrastructure.

- **Queue load**: Another metric that allows to determine whether the RDNS application is starving on computing resources is the queue load. The application has its own Message Queuing Telemetry Transport (MQTT) message queue in which incoming traces and location information sent by the OBUs are stored before being processed to detect possible risks. The state of this queue is exported as a metric that is exported to the monitoring subsystem. If the CPU resources assigned to the application are not sufficient to process all incoming messages immediately, the queue load increases, at the same time increasing the overall end-to-end delay. This metric is complementary to the end-to-end delay and considering them together gives a clearer picture of the application's status.

The event logs and traces stored in the app, while already present in the previous iteration of the application at the time of writing D5.1, they are now exported to a Kafka bus using a Kafka producer, so they can be uploaded to the DLT, where they are safely stored.

3.2.4 Vehicular Hardware: Gadget for VRUs

In addition to the OBUs and RSUs, a gadget has been designed and constructed to be able to notify the VRUs driving on the bicycle when there is a risk (see Figure 6 for the final design). The visual and acoustic signals which alert the presence of a tram while a vulnerable user is in the designated areas are programmed with an Arduino Leonardo board [5]. The visual signals consist of three Light-emitting diodes (LEDs) that light up depending on the conditions detected by the RDNS, showing the severity of the alarm if it's a critical alert or just a warning. Also, a buzzer is included in the panel to emit an acoustic signal to warn both the driver and pedestrians around.

Figure 6: The VRU panel designed for alerting the driver

On the vehicular side, every time the OBU receives a Decentralized Environmental Notification Message (DENM) message – which contains information related to a road hazard or an abnormal traffic condition - notifying the presence of a tram, it tells the Leonardo board through the writing of a message via serial. Listed below are the multiples states available for the panel:

- Off State: no tram presence on the nearest station, so the VRU driver could drive without receiving any warning. Only the 'Status' green LED is powered on, notifying that the panel is properly connected to the OBU.

- Warning State: if a DENM is received regarding the warning awareness, the OBU sends to the dashboard panel the string 'Warning'. The dashboard then activates the yellow LED and starts blinking it.
- Critical State: if the received DENM is about a critical awareness, the OBU will send the string 'Critical'. The yellow LED exits the blinking state and goes to a permanently turned-on status, the red LED is also turned on without blinking and the buzzer is activated with a linear tone.

To exit Warning or Critical states and to go back to the Off state, the OBU will send the string 'off' to the dashboard to turn off all buzzer(s) and LED(s). That way the Arduino Leonardo acts is as a state machine whose periodic reception of strings through the serial changes their states. Arduino Leonardo has been chosen due the built-in USB communication capability thanks to its ATmega32u4 microchip, easing communications through Serial/COM ports from the V2X stack deployed at the OBU. Note that this is a prototype developed for the project to validate the functionality of the use case. A second version, closer to a commercial solution, could be redesigned to be more compact or feature other audio-visual elements, such as a small screen.

3.3 Deployment and validation outcomes and observations

During the application development phase, tests and atomic validations of the application modules have been performed to deploy the UC2 application via the Pledger platform and to proof its functionality both with hardware (VRUs equipped with OBUs) and with simulated VRUs. These functional tests have been performed for the entirety of the application's functionality and for the integration endpoints developed as part of the application. For the deployment and testing of the UC2 application, from the beginning of the application development phase, the indoor lab replica of the testbed was used (which features the same elements as the on-street testbed, namely the radio nodes, as well as the far-edge, edge and cloud compute nodes). Since the same hardware will be used for the final on-street deployment that will be available for the final phase of the project[2], the entire application design and functionality will be ported to the new environment without any foreseeable issues.

The full functionality of the application was shown in the first interim review demo, where aforementioned testbed hardware was used and only the traces for both VRUs and tram were injected artificially into the system. Given the on-street deployment has not yet been finished at the time of writing this deliverable due to a plenitude of delays suffered, the final validations are to be performed as part of the Tasks T5.3 and T5.4.

In the laboratory-tests performed, we want to highlight that the expected functionality of the application could be confirmed and that all the components of the UC2 application (RDNS, V2X stack, tram detection service, OBUs/RSUs) have been successfully integrated. Note that two successfully tested features developed in this final reporting phase for the application development had not been tested by the time of writing the previous iteration of the deliverable: i) the collection and export of application metrics. It was validated that the 3 metrics are correctly measured by the application and that Pledger's monitoring system is capable of collecting these metrics to act upon them (Service Layer Agreements, triggers to scale/offload). ii) The audio-visual notification system for bicycle drivers has been built and integrated with the OBUs. It could be validated that both visual and acoustical signals can be well perceived during the tests, whenever an alert is sent out to a VRU.

With these two final features tested and validated, the UC2 application can be considered complete and fully functional. For any adopter or user to be able to use this application, the software modules that form the application need to be operational and the VRUs need to have the necessary radio connectivity enabled between their vehicle and the infrastructure. Note that both the orchestration and lifecycle management, as well as the configuration of the infrastructure including computation, networking and radio are handled by the Pledger platform. Further, the gadget that can notify bicycle or e-scooter drivers needs to be mounted on the respective VRUs; it will automatically connect to the infrastructure and from

[2] The availability of the on-street deployment at the time of writing is the second week of September 2022.

then on will generate alarms or warnings whenever risky situations are detected by the RDNS. The need of using an OBU also applies to the tram, as a means to share the position of the tram with the RDNS application. Adopters or users that desire for other vehicles (e.g. busses, or cars) to be detected and to form part of the risk detection, these vehicles could be also equipped with OBUs. Then, by adopting different station IDs that represent these vehicles and by defining new or additional danger areas the scope of the risk detection could be customized for new locations. This is implemented by adapting the application code and enables the use of the application in a large variety of topologies and potential scenarios.

4 UC3: Manufacturing the data mining on the edge

4.1 Short Application description

The UC3 applications deal with the analysis of the process stability of the SYNCROMILL machine [6]. SYNCROMILL machines characterize a unique, cost-effective, and innovative machine concept, allowing large quantities of parts to be processed with the highest levels of precision. To ensure high throughput rates and consistently high quality, a stable process is essential. To provide customers and engineers with a tool to facilitate this analysis and to derive appropriate action measures, the appropriate modules are developed in this task.
A stable process is characterized by approximately constant media consumption, meaning the machine uses constant amount of air and electricity during the process. A drift over time points out a possible leakage or malfunctioning electrical components. Furthermore, the parts produced by the machine are analyzed in terms of constant cycle time and quality to ensure the required output. Lastly, the thermal stability has a high influence on the quality of the parts produced. To ensure highest precision of 0.01 mm, machine operators usually run the machine in a warm-up mode before going into production mode and actively processing parts. The duration of this procedure is derived mostly from experience and rather too long than too short in order not to risk quality losses with the disadvantage of losing valuable production time. Determining the thermal stable window allows to determine the exact timepoint when the machine is ready for production and gain time for active process.
The main features for the modules Media Consumption and Analysis of Parts Produced have been already developed in the last iteration of this task and described in Deliverable D5.1 [1].

4.2 Final report on application modules

4.2.1 Module: Media Consumption

There are two variations of this module determining the average media consumption of the machine: pneumatical – analysing the air consumption, and electrical – analysing the electrical consumption. The implementation is the same for both analyses and follows the concept described in D5.1.

Since the last iteration of the deliverable, there have been no updates regarding this module and all features were already implemented in M20.

4.2.2 Module: Analysis of Parts Produced

This module implements the analysis of the parts produced by the SYNCROMILL machine, e.g., cylinder heads, oil pans, etc. The analysis includes the cycle time, the duration of the different subprocesses, the type and data matrix code of the part.

Since the last iteration of the deliverable, the UI has been improved to enable a more detailed analysis of the parts produced by the machine. The user can now select up to 5 parts and compare their subprocesses to identify the source of any deviation in the process. Possible reasons for deviations are repeated clamping due to imprecise placement of the part, or irregularities in the machining process resulting in longer machining duration. When observing frequent irregularities of the same type, this might point out a mechanical or electrical issue and the user can act before an actual breakdown in production. Furthermore, any alarms occurring during the processing of a part, are visualized, enabling the user their analysis and identification of deviations in the process.

An example is shown in Figure 7. All parts produced by the machine are visualized as individual dots over time, the y-axis being the cycle time. One part represents an outlier compared to the others. Selecting the outlier and a “normal” part and comparing the subprocesses shows, that a repeated clamping process (highlighted in the blue box) resulted in a longer cycle time, which can be observed as the outlier. In case this behavior becomes more frequent over time, the clamping device should be checked.

Figure 7: Comparing two parts produced with a repeated clamping process during one part.

Information included in these messages include or enable the extraction of sensitive information, e.g. the amount of parts produced in a certain time range or the amount of scrap. The access to this information should only be given to authorized parties, which is realized by using the DLT. The integration with the DLT is described in the Deliverable D5.4 [7].

4.2.3 Module: Thermal stability (new)

A further aspect to guarantee the process stability of a SYNCROMILL is to ensure that the machine and its mechanics are in a thermal stable window. As many components assembled in the machine are metallic, they expand during the process due to the heat generated by friction. SYNCROMILLs are used for fine machining with an accuracy of up to 0.01 mm, therefore, the uncontrolled expansion of the components must be prevented in order to guarantee this high accuracy. Usually, machine operators run the machines in warm-up cycles, i.e., moving the machine axes automated without raw material and production. The duration of this warm-up phase is often not known and is mostly based on experience, whereas other influencing factors (e.g., temperature in the shop floor) cannot be estimated. Therefore, operators play safe and warm up for too long rather than too short resulting in loss of valuable production time. This module has been developed to serve exactly this purpose and help the machine-operators to estimate the thermal state of the machine. This is not only useful, when starting the production at the beginning of a shift, but also after a downtime during production, where it's not clear, whether a warm-up cycle is needed again and for how long. For the analysis, no further sensor or temperature sensor is required, it's a completely data-driven approach, which is explained in the following.

4.2.3.1 Module Implementation Details

The data used for the analysis are the positions of the axes using two measurement systems (a direct and an indirect one mounted on each axis) gathered during one iteration of the warmup phase (moving the axis to the other side). The thermal stability indicator is determined via linear regression and the resulting gradient of the obtained line. This indicator rises as long as the machine's components are expanding and gets stable as soon as the components are in a thermal stable range. Based on the machine-operator can determine the time to stop the warm-up cycle and change into production mode.

To automatize the process, the module as well as the automation software of the machine has been extended. The module calculates the thermal stability indicator and determines the action to execute for the machine: "Start", "Stop", "Continue", to respectively start, stop or continue the warmup process based on the previously obtained results. Afterwards, the module publishes the result on the message broker of UC3 (RabbitMQ), from where a consumer publishes this result on the Programmatic Logical Controller (PLC) of the machine. The controller processes the result accordingly to control the machine in the desired way. Furthermore, the result is stored in an instance of MongoDB for the historical overview.

Figure 8 shows an example. One red dot illustrates the result (i.e., thermal stability indicator) of one warmup phase. During the warmup of the machine, the values rise, whereas they become consistent as soon as the machine is warmed up. The green box indicates the thermal stability window. As soon as this window is configured, the warmup process is automated and the machine changes into production mode as soon as the threshold is reached.

Figure 8: UI Thermal stability

4.2.3.2 Module Requirements and Virtualization

The module is virtualized as Docker containers with the following resource requirements:

Module Name	RAM	vCPUs	Storage (Solid State Disk)	Virtualization
Thermal stability	80 MB	1	N/A	Docker

4.2.3.3 Placement and scaling considerations

This module communicates directly with the PLC of the machine, while the physical machine is waiting for the response and result of the analysis to decide on the further procedure. In this process latency is relevant, therefore highest priority will be given to this component and it will stay on the edge. In case of performance issues or limited resources, other applications, like the media consumption pneumatic can be offloaded to the cloud to ensure an optimal Quality of Service (QoS) and QoE of all components. The QoS and QoE metrics have been identified in D5.3 [8].

4.2.3.4 Inter-Module dependencies and integration considerations

The "Thermal stability" component relates to the RabbitMQ as well as the MongoDB to store the results. The MongoDB stores the analytical results to provide a history for visualization on the Cybernetics Web UI to the user. This enables a better understanding of the process and the machine to the end user. RabbitMQ serves both as data source to retrieve the relevant data for the data analysis as well as the target of the result towards the PLC, from where a consumer transfers the result to the defined interface on the PLC. Therefore, integration has to be tested from machine (PLC) via RabbitMQ and application back to the PLC validating the expected behaviour of the machine.

4.2.4 Module: Monitoring App (new)

This module has been implemented as a new application module to ease the integration with the Pledger system, by monitoring relevant QoS to enable the Service Provider to monitor his/her application as well as to enable Pledger to monitor and optimize performance.

4.2.4.1 Module Implementation Details

The QoS metrics are based on available metrics of RabbitMQ, and Docker as described in Deliverable D5.3 [8]. These metrics are extracted using the APIs of RabbitMQ to monitor the message broker [9] as well as the Docker SDK for Python [10] to get the resource metrics of the different services deployed. Afterwards, a message containing the information is created and published on RabbitMQ, from where it is transferred to StreamHandler using the RabbitMQ-To-StreamHandler component described in Deliverable D5.3 [8].

The messages share a common format, where the "DocumentType" describes the type of metric documented, "InfrastructureProvider", "InfrastructureName", "NodeName" and "TimeStamp" give more information about the origin and time of the metric. The "DocumentBody" contains the relevant information/metric for the specific type of "DocumentType".

Three types are available for the UC3:

- SystemMonitoring: metrics about the current usage of computational resources of the system
- AppResourceMonitoring: metrics about the current usage of computational resources of the application
- ApplicationMetricMonitoring: application specific metrics, e.g. calculation_time, describing the time needed for an analytical result to be determined

4.2.4.2 Module Requirements and Virtualization

Module Name	RAM	vCPUs	Storage (Solid State Disk)	Virtualization
Monitoring App	60 MB	1	N/A	Docker

4.2.4.3 Placement and scaling considerations

This module is intended to be placed on the edge to enable the monitoring of the edge infrastructure as well as the performance of applications running on the edge. The module itself will not be managed by Pledger regarding performance optimization, but it enables the other modules to be managed by Pledger based on their QoS and performance.

4.2.4.4 Inter-Module dependencies and integration considerations

The Monitoring app is only depended on the message broker used on the UC3 edge infrastructure (RabbitMQ) and docker daemon to get the relevant statistics of both Docker and RabbitMQ. Furthermore, the results are published on the related exchange on RabbitMQ to be transferred via StreamHandler to Prometheus to make the results available to the Pledger system. This procedure has been described in the deliverable D5.3 [8].

4.3 Deployment and validation outcomes and observations

The modules "Media consumption" and "Analysis of parts produced" have been already tested and deployed successfully before and were documented in the first iteration of this deliverable (D5.1 [1]).

The module "Monitoring app" has been deployed and demonstrated successfully during the second pilot iteration while showcasing the integration of the UC application "Media consumption pneumatic" with the Pledger system and offloading the application to the cloud based on the load of the infrastructure. This scenario has been documented in the deliverable D5.4 [7] and is especially relevant for UC3, since ideally the application should run as much as possible on the edge but can be offloaded for a short time to meet in average the defined thresholds. This way, optimal performance with little increase in costs for additional third-party cloud resources can be ensured.

Lastly, the "Thermal stability" app has been deployed and validated successfully. This has been done in three phases: First, the data analysis was validated without the communication to the PLC and the warmup process has been observed. The results and the correlation of the thermal stability indicator's consistency and the mechanical expansion has been validated using an additional sensor to measure the exact position in space and the change due to expansion.

In the second iteration, the receiving of the analytical result and the derived instruction on the machine has been successfully tested with a continuous warmup cycle. The end-to-end integration with the resulting automated warmup process has been fully established.

Users can interact with the UC3 application by accessing the Cybernetics UI to analyse the manufacturing process. Based on these insights they can adapt the specific process to improve the efficiency of the machine. The components can be transferred to any other SYNCROMILL machine in a factory by setting up the relevant infrastructure and connectivity to the relevant controllers of the machine. However, as the components are tailormade for machines of type SYNCROMILL and their associated processes, they cannot be transferred to any other type of machine.

5 Conclusions

This deliverable serves as the final report about the application development performed by the three use cases within Task 5.1, finalizing in M33. The content of this document focuses on the development performed since the last deliverable produced in M20: i) on the finalization of already started work on the software modules that form the three use case applications and were already presented in D5.1, ii) on the extension of these software modules to include new functionality that was not foreseen originally, and finally iii) on any new software modules that were developed to complement the application with useful features. These three aspects are reported in Section 2, 3 and 4, dedicated to UC1, UC2 and UC3, respectively.

As described in these three sections, all use case applications have been implemented, tested and had their functionality validated. For each application, additional functionalities were developed to be able to better showcase the integration with Pledger. In particular, for all UCs effort was dedicated to create and export application metrics that can be used by the monitoring subsystem of the Pledger platform to detect whether the applications are operating normally or run into issues. Effort also was dedicated to implement the necessary functionality for other integration points, for example exporting information to the DLT by pushing information to the StreamHandler. With these additional features or functionalities in place, the necessary basis to perform the integrations with Pledger is present in each application.

Beyond features related to the integration, some of the applications have been improved with UI enhancements (UC1 and UC3), or completely new modules that provide additional features (e.g., Thermal Stability module for UC3). But not only additional software was developed, in UC2 a gadget that can be mounted on scooters and bicycles was designed, 3D-printed and equipped with the necessary electronics to be able to warn the users in case of possible risks.

There has not been any major shortcoming in the application development of the three UCs. As reported in the corresponding sections, the shortcomings are normally caused by the physical technologies (e.g., Wi-Fi) used or they are caused by external issues that are not directly related to how these applications are orchestrated and executed via Pledger. For example, in UC1, some performance limitations were detected, e.g., when many users connect via the multiplayer feature. This and other minor issues of UC1 are all reported in Section 0. In UC2, the tram detection feature meant to be fed with data from an external detection module (via image processing) for which an API was developed. Since this external module was not developed, eventually it had to be replaced with a feature that works with the same technology used for the bicycle tracking, i.e., using an OBU installed in the tram communicating with the Pledger infrastructure and sharing its position, but emitting a different identifier.

Overall, each use case application has been able to implement all the planned key features and the validations of the application functionalities have been successful. Further, the development has enabled the key features necessary for the integration with other Pledger components. With these two aspects, the full functionality and the necessary features for integration, the base for the final testing and evaluation phases after M33 has been laid out for each use case.

6 References

[1] PLEDGER. "D5.1: Pledger Applications for the use cases I", A. Betzler, 2022, Available: http://pledger-project.eu/D5.1.pdf. [Accessed 3 August 2022]

[2] ARtwin Project, https://holo-light.com/artwin-for-a-sovereign-ar-cloud/, Accessed [2 September 2022]

[3] PLEDGER. "D4.4: Trust, Security and Privacy related tools I", F. Iadanza, to be published in the project website http://www.pledger-project.eu/content/deliverables.

[4] PLEDGER. "D4.5 Smart Contracts and DApps implementation tools", N. Kapsoulis, to be published in the project website http://www.pledger-project.eu/content/deliverables.

[5] Leonardo Board, https://docs.arduino.cc/hardware/leonardo, Accessed [2 September 2022]

[6] SYNCROMILL C machine. https://www.fill.co.at/en/products/syncromill-c [Accessed 6 July 2022].

[7] PLEDGER. "D5.4: Pilots operation and monitoring II", V. Stanzl, Available: http://pledger-project.eu/content/deliverables. [Accessed 25 July 2022].

[8] PLEDGER. "D5.3: Pilots operation and monitoring I", V. Stanzl, Available: http://pledger-project.eu/content/deliverables. [Accessed 6 July 2022].

[9] "RabbitMQ monitoring", https://www.rabbitmq.com/monitoring.html [Accessed 6 July 2022].

[10] "Docker SDK for Python". https://docker-py.readthedocs.io/en/stable/index.html [Accessed 6 July 2022].