Knowledge-based services for and optimization of machines

Deliverable D1.5d

Description of demonstrator implemented for material handling demonstrator

Deliverable type:	Document
Deliverable reference number:	ITEA 18030 | D1.5d
Related Work Package:	WP 1 Use Cases, Requirements & Evaluation
Due date:	M38 (30 November 2022)
Actual submission date:	
Author(s)	Yangkoo Lee, Youngjae Lim, Wookeun Jeong, Wansik Choi
Confidentiality	Public
Version	v 1.0

Contributors:

Yangkoo Lee	ETRI
Youngjae Lim	ETRI
Wookeun Jeong	CIP
Wan Sik Choi	CIP

Table of Contents

Figures

1. Revision History

Version	Date	Description
v 0.9	11/09/2022	Initial draft
v 1.0	11/11/2022	Complete first version

2. Abstract

This demonstration will be implemented on an AMR-based logistics collaboration testbed built by ETRI. The use case in Korea consists of interaction scenarios between material handling devices and operators in a smart factory. For example, a specific AMR can be selected in a work process to carry out material transport, the most suitable AMR for a task is automatically recommended, or collaboration between AMRs can be achieved through the AMR's working status monitoring. In addition, when a crane loaded with a specific material is moving, it can collaborate with an AMR to perform tasks or share operating schedules between devices. And operators also participate in the material handling process as independent moving objects.

The AMR object constituting the entire testbed is attached with IoT sensors to generate data such as location, weight, and battery status, and transmits it to the digital twin platform through the IoT Gateway. A digital twin platform for data collection and storage, including data communication and message exchange protocols, is built on the testbed. The digital twin platform also provides query and information services based on an integrated data model. The operator monitors the work space using a visualized digital twin, establishes a task plan, and interacts with the work field.

As a result of the demonstration, Korea's use case will be utilized with other partners' use case results to use them as a target domain for connecting different digital twins and sharing data and this allows for the construction of a digital twin ecosystem that enables interaction between incompatible digital twins.

3. Description of demonstrator implemented for material handling demonstrator (ETRI, CIP)

3.1. Description and covered Use Cases

The general information of the use cases is summarized in the table below. the table describes use case definitions, responsibilities, possible contributions of partners to the use cases, interaction of use case with other work packages.

Responsible person:	Yangkoo Lee
Main contributor, demonstrator responsibility:	ETRI
Additional contributors and needed contribution:	• CIP: Digital twin platform, HMI system • Others:
Use case:	• Task Management • Automatic Charging • Loading Weight Control • Real-time Analysis
What is going to be validated with this demonstrator?	• WP1 – Use Cases, requirements and evaluation • WP2 – Interoperability of Digital Twin eco-systems • WP3 – Processing of multiple Digital Twin's data • WP4 – HMI system for controlling and monitoring Digital Twins
Start of project:	• 10/2019

Table 1: Description of demonstrator for material handling interaction

3.2. Outline of the demonstration of "Material Handling Interaction Use Case" (ETRI)

3.2.1. Indoor cooperative logistics system

For the digital twin-based manufacturing logistics use case, we built an indoor cooperative logistics system based on physical assets as shown in Figure 1. The use case infinitely repeats a scenario in which two AMRs check the weight of materials while uploading and downloading materials on both sides of the conveyor through collaboration. In the process, if the weight exceeds a given threshold, the AMR moves to the inspection station for confirmation by the administrator.

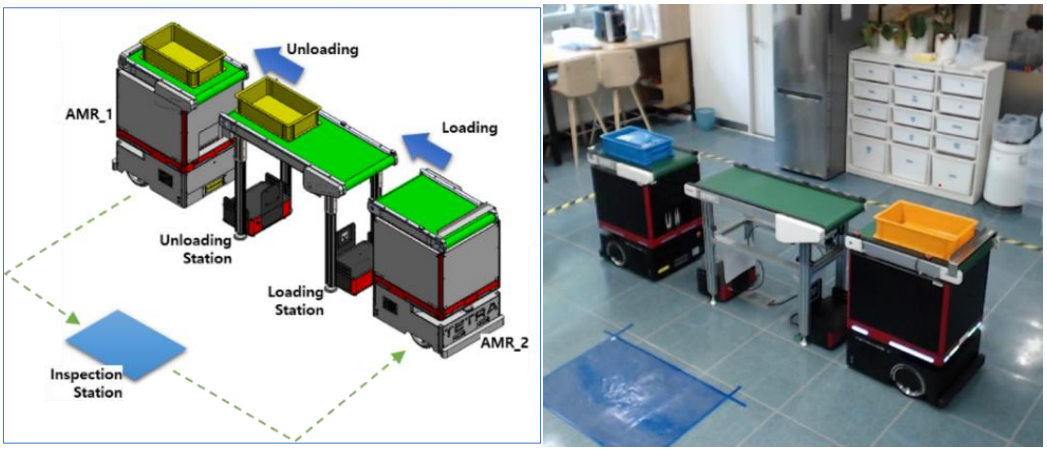

Figure 1. Real asset-based indoor cooperative logistics system that performs simple uploading and downloading tests on both sides of a fixed conveyor

3.2.1.1. System components

The collaboration-based indoor logistics testbed built to verify digital twin-based data analysis and interoperability technology consists of the following elements in the Figure 2.

- Two TETRA-DS5 robots (AMR) equipped with conveyor modules for loading and unloading
- Two charging station for TETRA-DS5 capable of automatic charging
- One stationary conveyor system with loading/unloading stations
- One environmental sensor set that collects environmental information (temperature, humidity, fine dust, etc.)
- One robot control server (ACS; AMR Control System) that manages robots and tasks, collects sensing information and transmits them to the digital twin platform

Figure 2. Hardware components of the indoor logistics testbed

3.2.1.2. Execute use case tasks

To apply the aforementioned indoor cooperative logistics use case scenario, the task manager of the AMR Control System (ACS) is used. Through the task manager's GUI as shown in Figure 3, various task scenarios can be designed and applied, and among them, the optimal scenario that is simple but can fully reflect the logistics characteristics is selected and performed repeatedly.

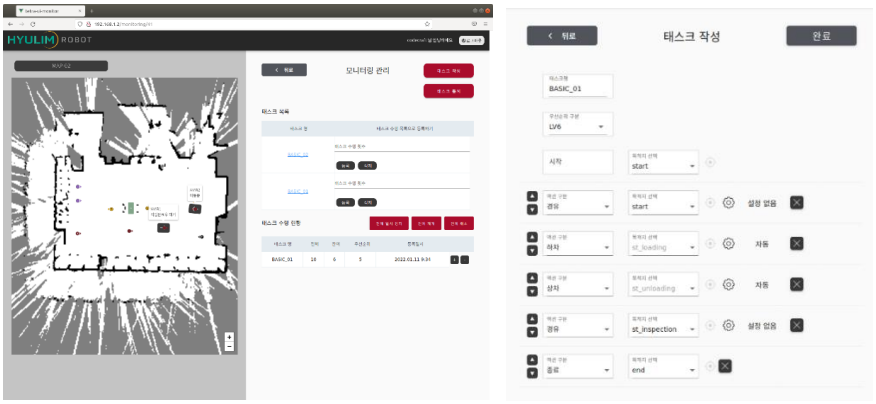

Figure 3. The execution screen of the task manager (task preparation, editing, execution, etc.) of the AMR control system

3.2.1.3. Real-time sensing data interface between ACS-HMI (MQTT)>

The message mapping between ROS - MQTT - Ditto based on Ditto's Thing models is defined as below to deliver the sensing data collected from each equipment constituting the indoor cooperative logistics system to the digital twin. Real-time sensing data collected from the AMR-based indoor logistics control system (ACS) is delivered to the digital twin platform using the MQTT topics in the list below and checked by MQTT tools as shown in Figure 4.

kr.re.etri.kmac:amr1/odometry
kr.re.etri.kmac:amr1/tetra_battery
kr.re.etri.kmac:amr1/weight_sensor
kr.re.etri.kmac:amr1/local_occupancy_map
kr.re.etri.kmac:amr1/navigation_path
kr.re.etri.kmac:amr1/operating_events
kr.re.etri.kmac:amr1/loader_events
kr.re.etri.kmac:amr1/command
kr.re.etri.kmac:conveyor/operating_events
kr.re.etri.kmac:env_sensor/env_data
kr.re.etri.kmac:st_charging/current_charging
kr.re.etri.kmac:st_charging/operating_events
kr.re.etri.kmac:st_inspection/operating_events
kr.re.etri.kmac:st_loading/operating_events
kr.re.etri.kmac:st_unloading/operating_events

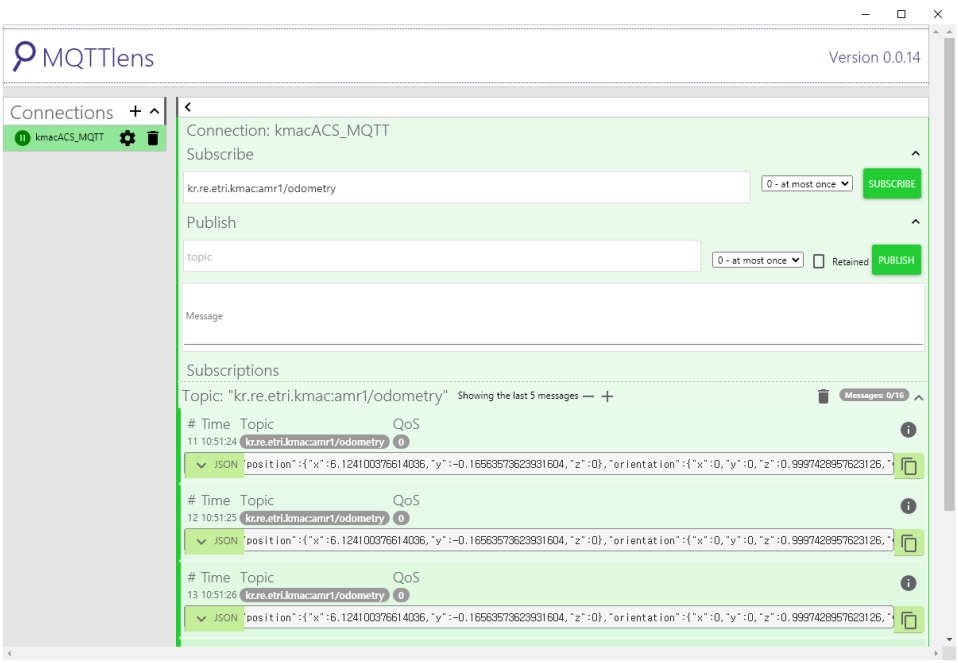

Figure 4. Major MQTT topics list and real-time data reception test

3.2.2. Interaction use case with digital twin of virtual asset

To test the interaction between digital twins in the physical AMR-based indoor cooperative environment, we built a virtual crane and forklift as a digital twin, and implemented a simulation module that performs material transport in real time while sharing the work area of real assets. Virtual assets generate working data in real time through this simulation module and are visualized on HMI through 3D modelling as shown in Figure 5 and 6. The HMI displays the task state of physical and virtual assets in real time by mapping the real-time working data from simulation module to the 3D model.

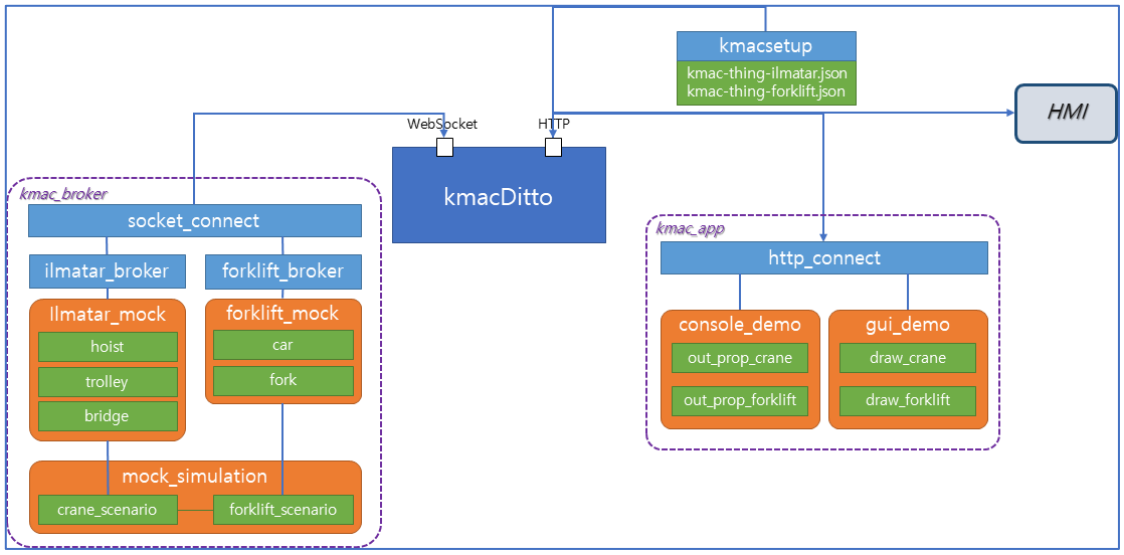

Figure 5. Configuration of the digital twin interface for virtual assets

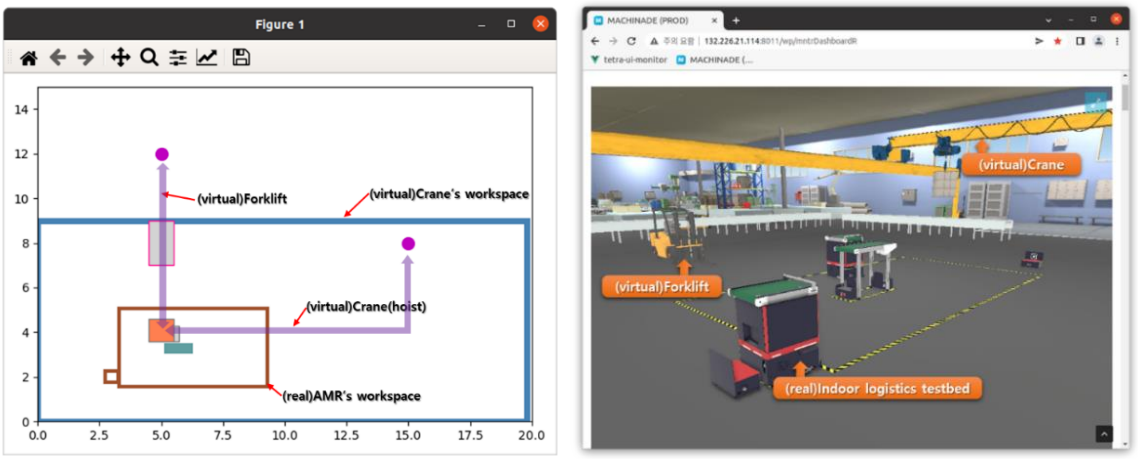

Figure 6. Material transport simulation of virtual assets (crane, forklift) and integrated operation with real asset

For the test, we set up a real asset-based indoor logistics operation consisting of two AMR units and one conveyor and a material transportation operation consisting of a virtual crane

and a virtual forklift. In this task environment, the process of interaction between different digital twins is performed.

If the virtual crane unloads the material at a specific location within the AMR's work area, the AMRs' movement path is interrupted until the virtual forklift loads the material after a while, and the avoidance maneuver of each AMR is activated. In this process, two different digital twins interact through a Ditto-based digital twin broker interface.

Figure 7 shows the avoidance maneuver. As shown in Figure 7, when the virtual crane unloads the material on the AMR route, the route planning of AMR is disturbed by this virtual material, and AMR changes the route to avoid it.

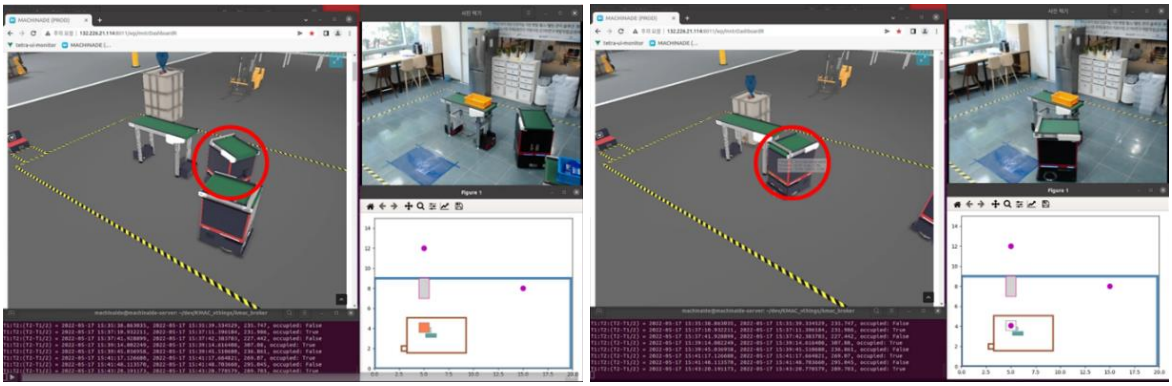

Figure 7. AMR Avoidance Maneuvering by Interaction with Virtual Assets

3.3. KOREA Machinaide (KMAC) Digital Twin Platform (ETRI)

3.3.1. Eclipse Ditto, An Open-source based Digital Twin Platform

Eclipse Ditto is an IoT technology that implements a software pattern called Digital Twin, and is an open-source framework that enables objects to be used as web services through Digital Twin. Since Eclipse Ditto as shown in Figure 8 is not a technology that supports the IoT platform itself, it does not directly define or implement IoT protocols to communicate with devices, and focuses on back-end scenarios by providing web APIs to simplify work on devices in customer apps or other back-end software.

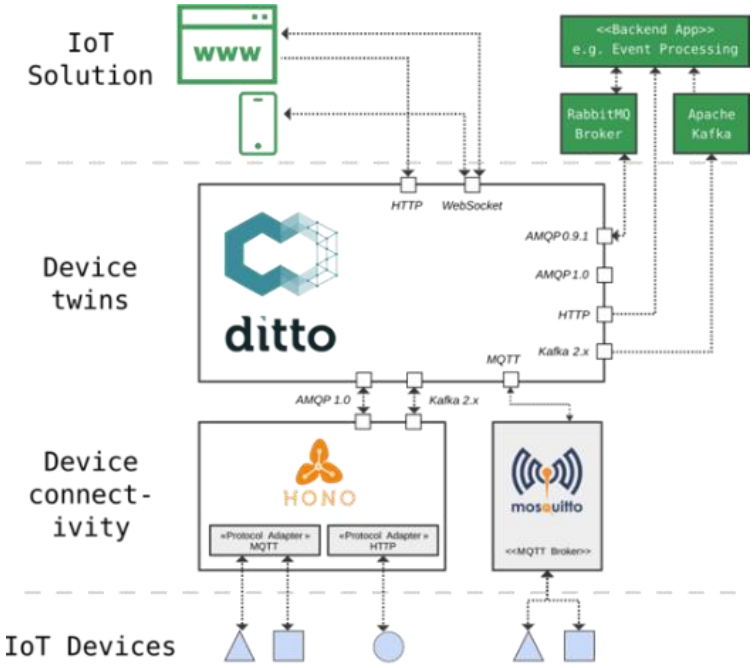

Figure 8. Construction of Eclips Ditto

3.3.2. Thing definition of assets and registration in Ditto

To obtain the Ditto Interface, not only each Asset that constitutes the KMAC Testbed but also the virtual "things" need a procedure to define and register the "thing" model as json to suit the Ditto protocol.

The definition of "thing" for the 9 assets constituting the test bed and the definition of the "thing" model for the virtual asset consisting of virtual cranes and forklifts are defined as json files consisting of attributes field, which are property information of assets, and features field, which are various sensing data, as shown below.

Thing model files for Assets in the indoor Logistics Testbed (9 Real Assets)	**Thing model files for vcrane & vforklift (2 Virtual Assets)**
• kmac-thing-amr1.json • kmac-thing-amr2.json • kmac-thing-conveyor.json • kmac-thing-env_sensor.json • kmac-thing-st_charging.json • kmac-thing-st_inspection.json • kmac-thing-st_loading.json • kmac-thing-st_unloading.json • kmac-thing-workspace.json	• kmac-thing-crane.json • kmac-thing-forklift.json

```
{
    "thingId": "kr.re.etri.kmac:amr1",
    "policyId": "kr.re.etri.kmac:policy",
    "attributes": {
        "twin_group": "device",
```

```
        "model_name": "TETRA-DS5 Mobile Platform",
        "manufacturer": "Hyulim Robot",
        "placed_time": "2021-08-10 15:00:30.12345",
        "description": "Automatic mobile robot for material handling",
        "specification": {
            "width": 0.49,
            "length": 0.592,
            ~~
        }
    "features": {
        "odometry": {
            "properties": {
                "pose": {
                    "position": {
                        "x": 0.0,
                        "y": 0.0,
                        "z": 0.0
                    },
                    "orientation": {
                        "x": 0.0,
                        "y": 0.0,
                        "z": 0.0,
                        "w": 0.0
                    }
                },
            ~~
        }
```

Figure 9. Example of "thing" model (json): "kmac-thing-amr1.json"

The thing model files of each defined asset are registered in Ditto as the PUT Request below.

```
### create Thing of AMR1
curl -u {userid}:{password} -X PUT -d '@dt_models/kmac-thing-amr1.json'
 'http://localhost:8080/api/2/things/{thingId}' -H 'Content-Type: application/json'
```

After each asset is registered in Ditto, when Ditto normally collects real-time data of the asset, users can access the real-time data of the digital twin using GET Request as follows.

APIs that have access to Digital twin are executed in accordance with the Eclipse Ditto HTTP API document (https://www.eclipse.org/ditto/http-api-doc.html).

```
### Get Data from AMR1 twin
curl -u {userid}:{password} -X GET 'http://localhost:8080/api/2/things/kr.re.etri.kmac:amr1'
```

3.3.3. Digital Twin Integration Platform (CIP)

Digital Twin Integration Platform (DTIP) is comprised in such a way that with Ditto at its center, it synchronizes real-time state values occurred in various IoT devices, provides web based simulation capabilities via collected information for AMR based indoor collaboration environment. Also DTIP can support the reflection of non-real-time event occurred in virtual space to actual space.

As shown in Figure 10, DTIP architecture composed of four separated modules: DT WAS, TwinConnect, Monitoring, and 3D Engine. Each module communicates with each other via WebSocket or REST API.

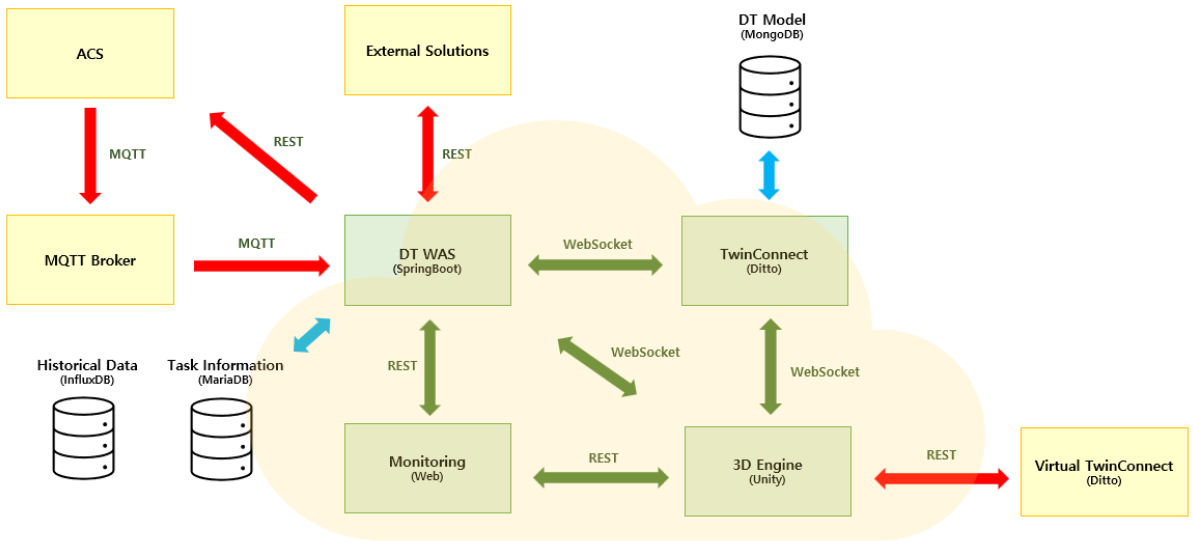

Figure 10. Architecture of Digital Twin Integration Platform(DTIP)

Functions of each DTIP module are described as in below.

No.	Name	Definition
1	DT WAS	- Performs the role of external gateway and comprises internal Back-end logic. - Maintains realtime information of TwinConnect by analyzing mqtt payload transmitted via MQTT Broker, and stores the history data to InfluxDB. - Sends non-realtime events occurred in platform to external
2	TwinConnect	- Stores as well as maintains realtime object information as "thing" model, and sends corresponding information to DT WAS or 3D Engine for utilization.
3	Monitoring	- Displays the data transmitted via DT WAS by composing Front-end UI - Receives non-realtime events occurred at Web or 3D Engine and sends them to DT WAS
4	3D Engine	- Displays 3D screen using history information transmitted through realtime information or DT WAS transmitted from TwinConnect

Digital Twin Integration Platform (DTIP) is provided in the cloud environment, and any user with valid account can access via web or mobile app. For external interface, DTIP provides "thing" model information through RESTful API (Figure 11). The API also is provided to the user with valid account, and authentication is carried out by issuing JWT.

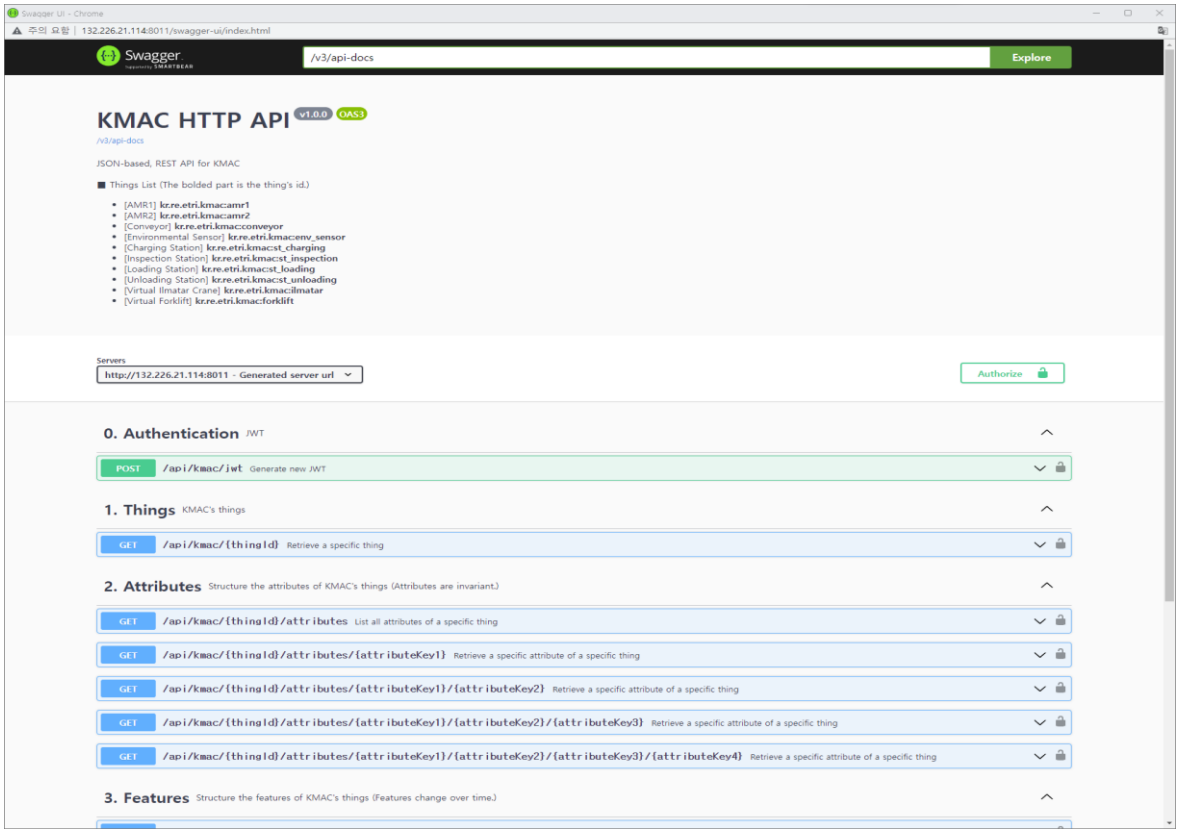

Figure 11. REST API documentation

Digital Twin Integration Platform (DTIP) provides services as Tomcat that is open source container, and can be started or stopped with the following commands.

```
### Start Digital Twin Integration Platform
service start tomcat9
### Stop Digital Twin Integration Platform
service stop tomcat9
```

With the start of service, functions are provided via web and mobile (Figure 12).

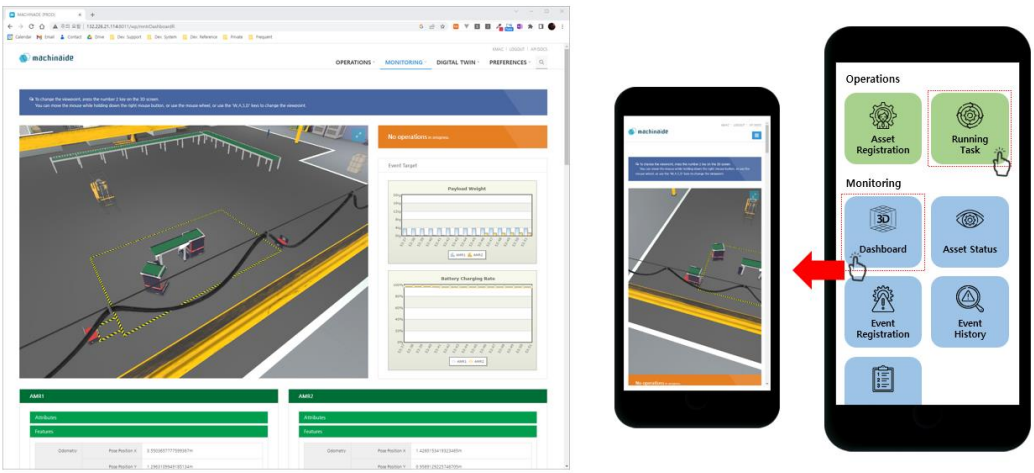

Figure 12. Web and mobile screenshots

3.4. Data Acquisition & Data Processing (ETRI-CIP)

The digital twin platform provides various types of storage structures depending on the characteristics of the data. In KMAC DT platform the digital twin model represents the physical assets based on the "Thing" provided by the Ditto framework, and the "Thing" is stored on the platform via the NoSQL database. Among information such as spatial environment, physical devices, and tasks of the IoT layer, data with static attributes are managed as a relational database (RDB), and data with dynamic attributes are managed through a time series database (TSDB). To this end, the DT platform configures multiple databases using the following DBMS, and can provide data services by linking each database as needed.

- MongoDB: Digital twin model management (e.g. thing file)
- MariaDB: Static data management (e.g. spatial information, environmental information, IoT device specification information, task information, user information, etc.)
- InfluxDB: Dynamic data management (e.g. device movement information, real-time event information, IoT sensor information, etc.)

3.4.1. Concept of data collection and storage design

Among the multiple databases, MongoDB is the basic database provided by Ditto, so there is no separate design, and it follows the storage policy of the "thing" file (Json) provided by Ditto. RDB and TSDB are designed based on the detailed items of the predefined "thing" data model. The "Attributes" items described in the "thing" model are classified as static data and become the basis for RDB design, while the "Features" items become the design basis for TSDB.

Based on the data flow in Figure 13, real-time data generated from the IoT layer is collected through ACS, and ACS issues the collected data as an MQTT topic and transmits it to the DT platform. The MQTT topic corresponds 1:1 to the detailed parameters defined in the "Features" item of the "thing" data model. Therefore, the MQTT topic is considered to be the data item that constitutes "thing", and the database of the DT platform is modeled in a form suitable for individual DBMS based on thing's technical content and MQTT topics.

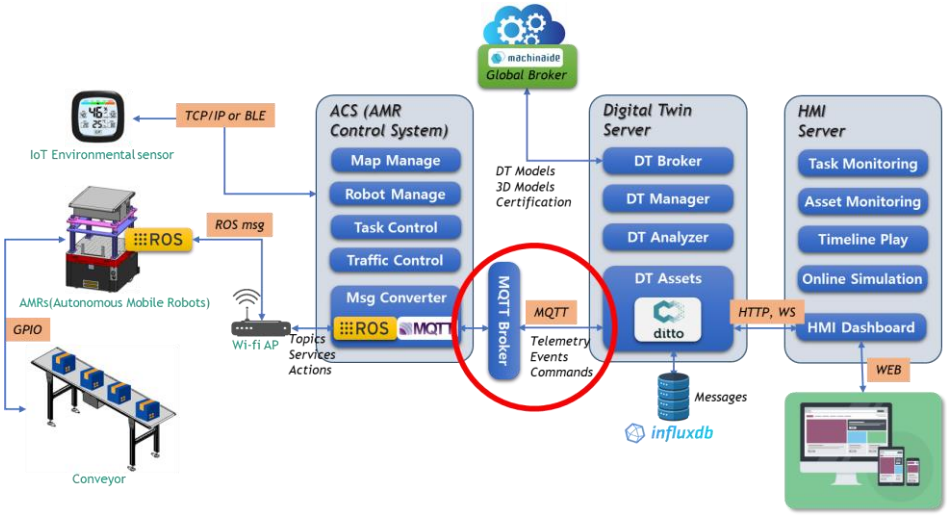

Figure 13. data flow on KMAC DT platform

The database is not designed to be subordinate to RDB or TSDB, but when building an physical database, it is divided into RDB and TSDB. The relationship between the database terms of RDB and TSDB is interpreted based on the following comparison table, and the data is modeled based on this.

RDB(MariaDB)	TSDB(InfluxDB)
database	database
table	measurement
column	key
PK or indexed column	tag key (only string)
unindexed column	field key
SET of index entries	series

Figure 14 is a class diagram showing the relationship between the object (entity) and the object derived from the KMAC test bed. In the class diagram, classes in the blue box are constructed by RDB as objects defining static properties, and classes in the grey box are constructed by TSDB as objects defining dynamic properties.

The database design is described based on class diagrams classified into two groups. In each class layer, each class has an inheritance/related/aggregation relationship with other classes, and these relationships are mainly used to distinguish between main tables/subordinate tables/independent tables, and to infer join relationships.

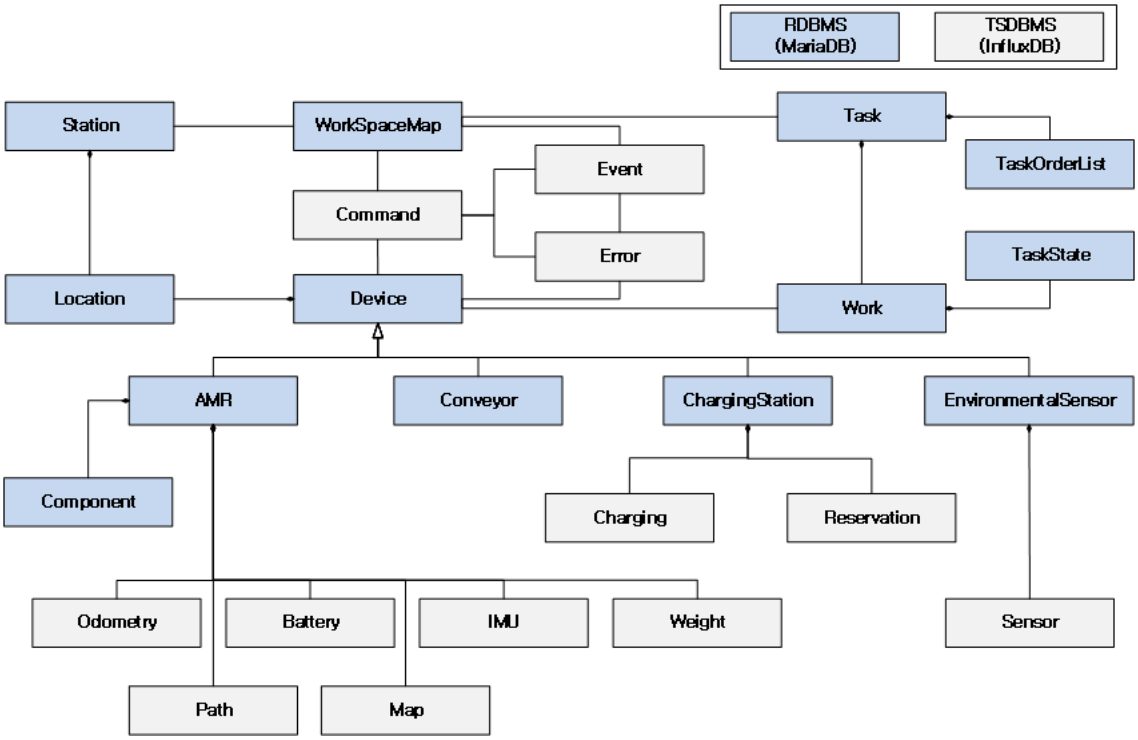

Figure 144. Class diagram (entity-relationship model)

Each individual class defined in Figure 14 corresponds to one table or measurement. The following table shows a table/measurement list defined as RDB/TSDB.

No.	Type	Name	Definition
1	Table	WorkSpaceMap	Full spatial information of test bed
2	Table	Station	Loading/Unloading/Inspection station information
3	Table	Location	Station's spatial location information
4	Table	Device	Logistics device information
5	Table	AMR	General information of AMR
6	Table	Component	Sensor of AMR configuration information
7	Table	Conveyor	General information of conveyor
8	Table	ChargingStation	General information of charging station
9	Table	EnvironmentalSensor	General information of IoT sensors
10	Table	Task	Definition information of Task
11	Table	TaskOrderList	Information of task order
12	Table	Work	Information of detail task
13	Table	TaskState	Task status/statistics information
14	Measurement	Odometry	Information of real-time odometry update
15	Measurement	Battery	Information of real-time battery update
16	Measurement	IMU	Information of real-time IMU update
17	Measurement	Weight	Information of real-time weight update
18	Measurement	Path	Information of real-time path update
19	Measurement	Map	Information of real-time map update
20	Measurement	Charging	Information of real-time charging update
21	Measurement	Reservation	Information of real-time reservation update
22	Measurement	Sensor	Information of real-time IoT sensor update
23	Measurement	Command	Information of real-time command detection
24	Measurement	Event	Information of real-time event detection
25	Measurement	Error	Information of real-time error detection

3.4.2. Data model for static data management

Figure 15 shows the schema structure of the RDB designed for classes classified into blue boxes in the class diagram of Figure 14. In the designed schema, objects (entities) described in "thing" utilize thing_id as the primary or foreign key, and objects (entities) not described in "thing" use serial numbers or identification codes to construct the schema.

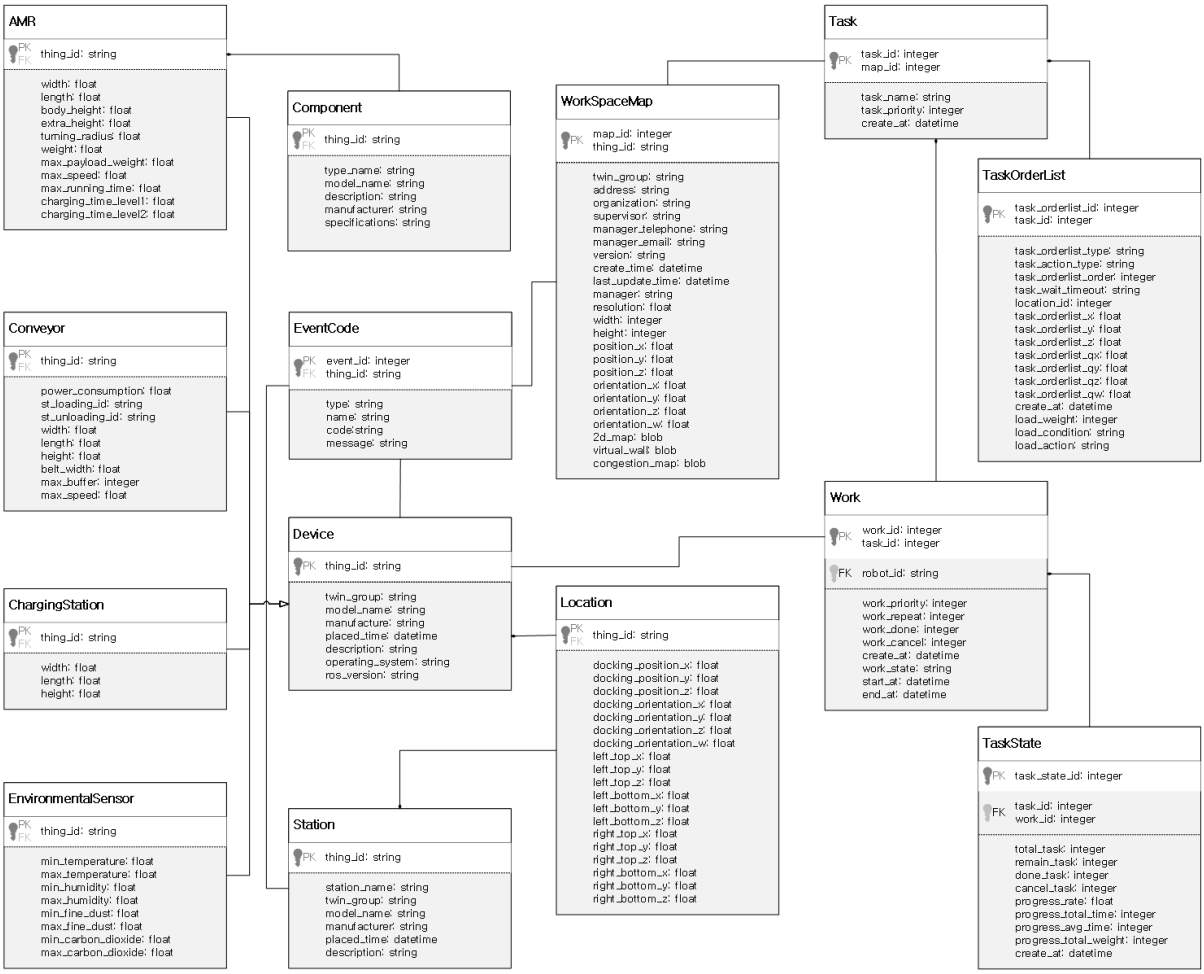

Figure 155. Entity-Relationship data model

3.4.3. Data model for dynamic data management

The TSDB is designed for classes classified as gray boxes in the class diagram in Figure 14. The designed TSDB is used to store a history of data that is updated in real-time among data collected from the IoT layer, and no separate schema is defined according to the design concept of TSDB. TSDB stores data in line protocol, and line protocol is described in the following format.

- line protocol: Measurement, tag, tag, tag, … field, field, …, …

The TSDB schema is defined by writing measurements that should be managed by history and data to be stored in line protocol, and the details are as follows.

Group	Measurement	line protocol (INSERT)
AMR	Odometry	Odometry,thing_id position_x,position_y,position_z,orientation_x,orientation_y,orientation_z,orientation_w,twist_linear_x,twist_linear_y,twist_linear_z,twist_angular_x,twist_angular_y,twist_angular_z,accel_linear_x,accel_linear_y,accel_linear_z,last_update,sampling_rate

AMR	Battery	Battery,thing_id charging_rate,power_consumption,charging_start_time,last_charging_completion_time,charging_state,last_update,sampling_rate
AMR	IMU_sensor	IMU_sensor,thing_id linear_acceleration_x,linear_acceleration_y,linear_acceleration_z,angular_velocity_x,angular_velocity_y,angular_velocity_z,last_update,sampling_rate
AMR	Weight_sensor	Weight_sensor,thing_id payload_weight,payload_class,last_update,sampling_rate
AMR	Local_occupancy_map	Local_occupancy_map,thing_id resolution,width,height,position_x,position_y,position_z,orientation_x,orientation_y,orientation_z,orientation_w,mapdata,last_update,sampling_rate
AMR	Navigation_path	Navigation_path,thing_id start_position_x,start_position_y,start_position_z,start_orientation_x,start_orientation_y,start_orientation_z,start_orientation_w,end_position_x,end_position_y,end_position_z,end_orientation_x,end_orientation_y,end_orientation_z,end_orientation_w,num_points,points,last_update
Conveyor	None	None
EnvironmentalSensor	env_data	Env_data,thing_id temperature,moisture,find_dust,carbon_dioxide,last_update,sampling_rate
Charging Station	Current_charging	Current_charging,thing_id charging_amrid,charging_percent,last_update,sampling_rate
Charging Station	Reserved_state	Reserved_state,thing_id next_amrid,last_update
Command	Command	Command,thing_id command_type,command_name,station_id,position_x,position_y,position_z,orientation_x,orientation_y,orientation_z,orientation_w,last_update
Event	Event	Event,thing_id event_type,event_name,event_message,docked_amrid,last_update
Error	Error	Error,thing_id error_type,error_name,error_message,last_update

3.4.4. Querying and Visualization

Querying data using Query Language provided by each database, and display the received responses on screen. Maria DB uses SQL, InfluxDB uses flux respectively. But MongoDB does not directly perform querying, instead it uses Ditto API for performing querying and receiving responses.

MariaDB user information is the screen displays of the results of data querying (Figure 16).

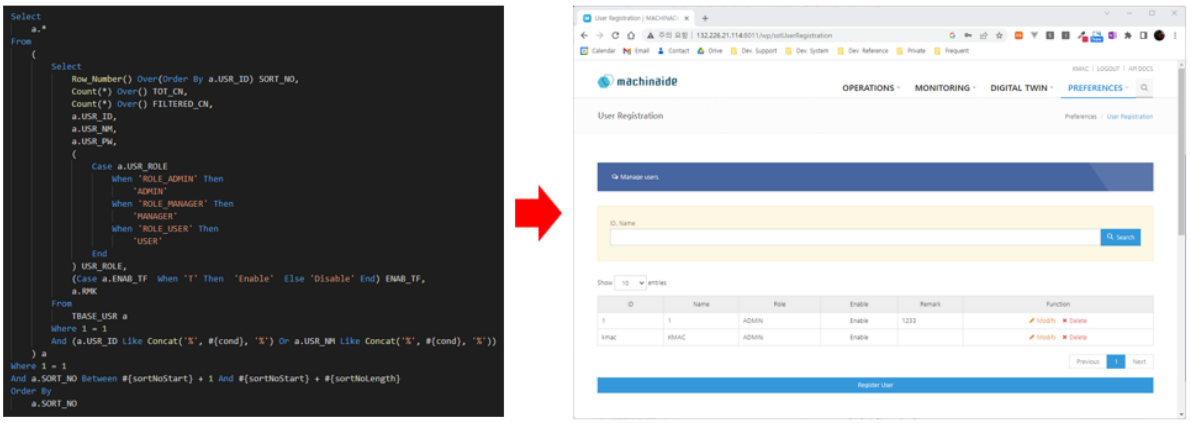

Figure 166. MariaDB's query and visualization

Environment Sensor history information of InfluxDB is the screen display of the results of data querying (Figure 17).

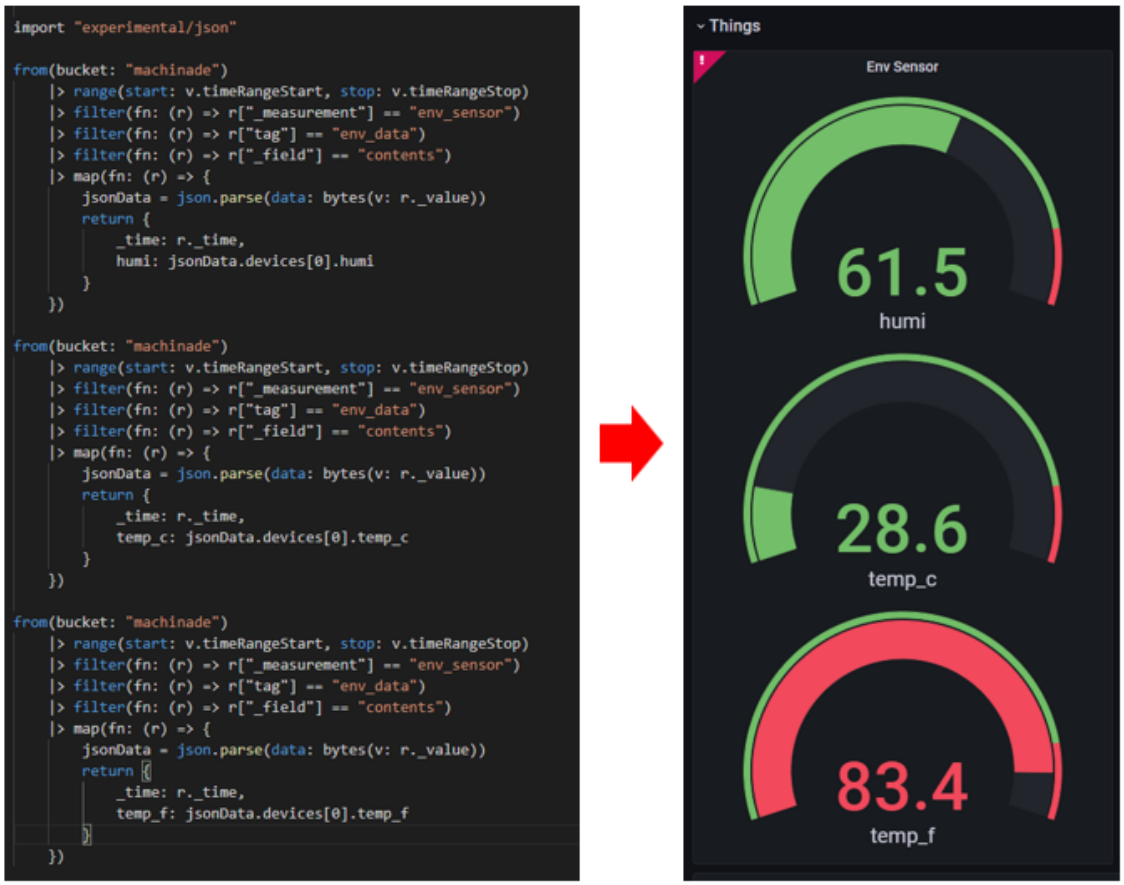

Figure 177. InfluxDB's query and visualization

Environment Sensor information stored in MongoDB is displayed by receiving the responses using Ditto API without direct querying (Figure 18).

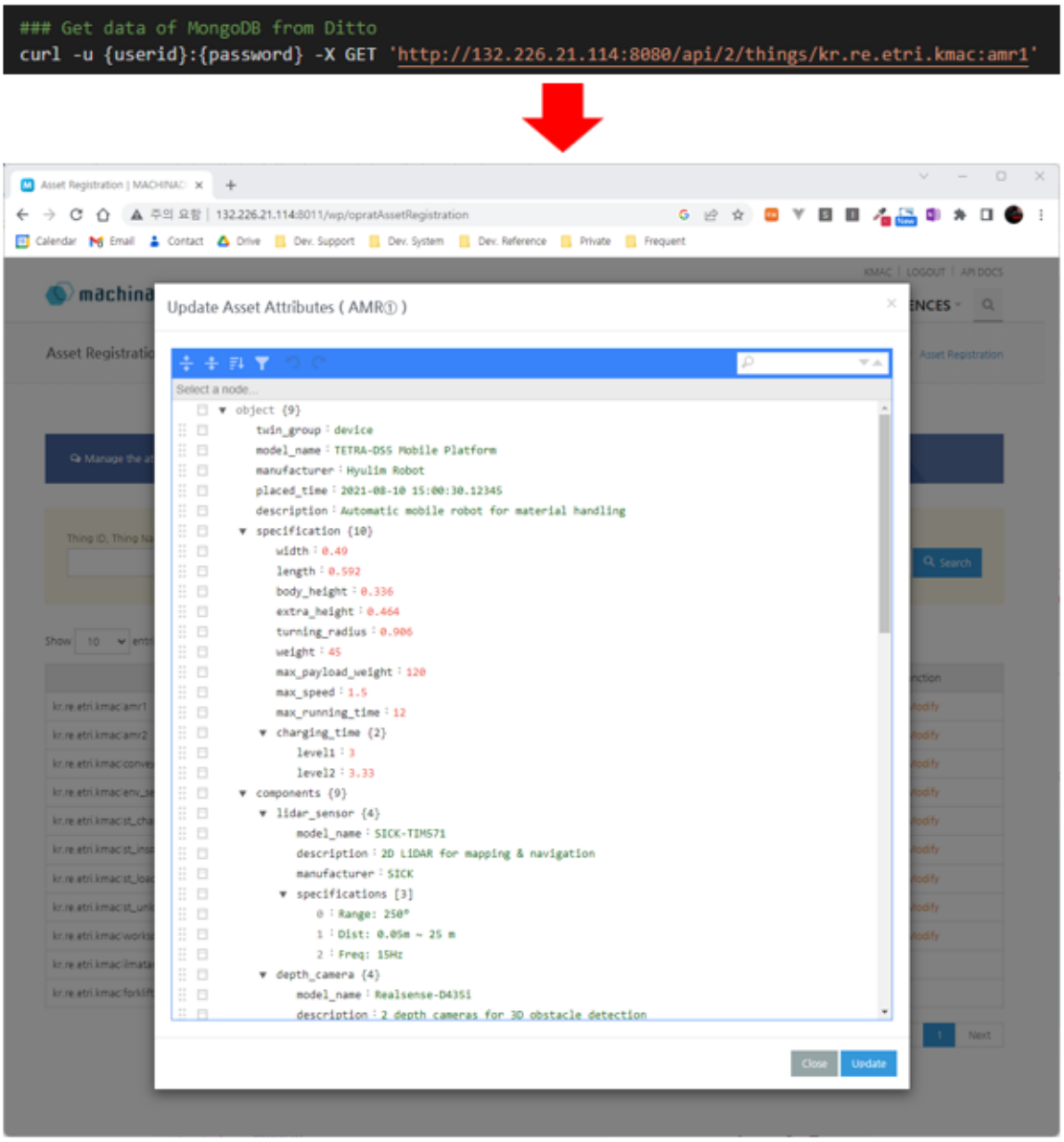

Figure 188. MongoDB's query and visualization

3.5. HMI Applications (CIP)

3.5.1. Concept of KMAC HMI system

KMAC's HMI is developed to derive functions necessary to verify interoperability between digital twins. For example, in the KMAC testbed, AMRs can interact with each other to perform given tasks, change task plans if necessary, and select appropriate processing logic in response to various contexts.

KMAC's HMI can provide more intuitive UI/UX-based insights to operators by monitoring and controlling the real site in a virtual environment by utilizing a digital twin. One of the main goals of the developed HMI is to identify the necessary functionalities and feasibility where

physical objects, digital twins, and humans are connected to interact each other. To this end, the HMI basically includes the functions of the existing control system. Also, it provides a method for efficiently collecting and storing IoT sensor data, and the resources and functions required to visualize it in 3D model. In addition, HMI is developed for running on the web environment so that the functionalities derived for HMI can be applied to the web-based digital twin ecosystem.

3.5.2. Key Features

KMAC HMI is implemented as a web application and provides services to users with access authority. Figure 19 shows the input window for user's login. The user accesses the web URL and logs in with user ID and password. A user who has successfully logged in can manage the KMAC testbed using the menus in the web-based HMI.

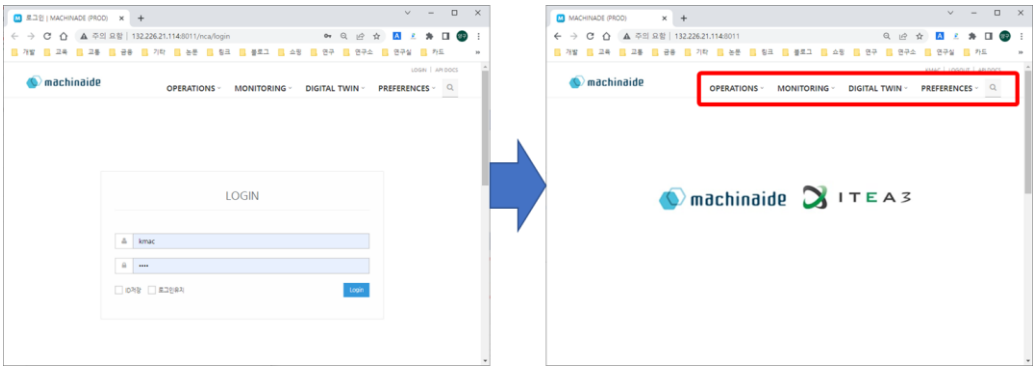

Figure 19. Log in process to KMAC HMI

The main functions of HMI consist of operation management, monitoring, event management, and history management. Operation management provides management of digital twin models registered in the digital twin platform. As shown in Figure 20, User can select the ID assigned to the digital twin object to modify the description or decide to enable/disable on the visualization of digital twins.

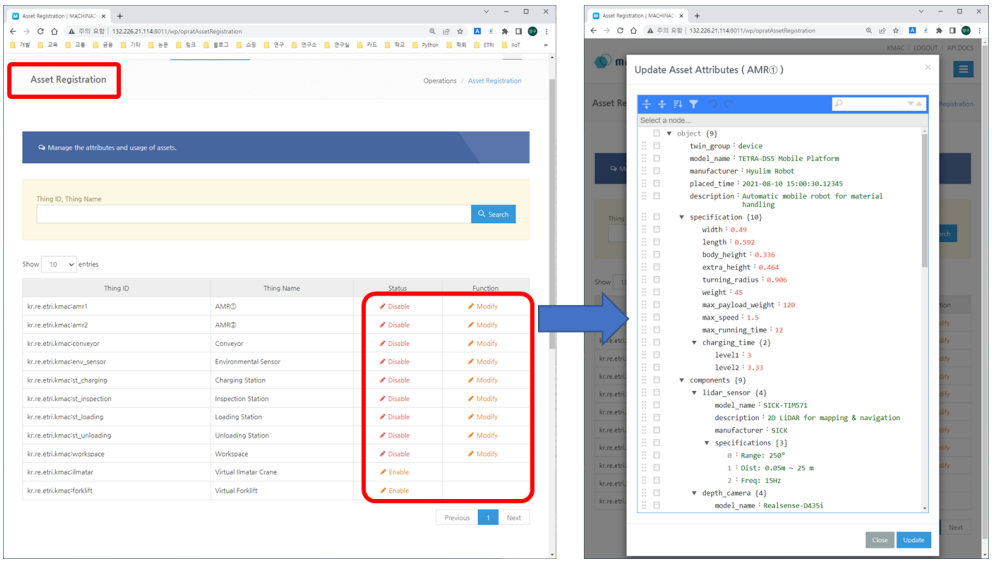

Figure 20. Operation management in HMI

Operators can register, edit, and delete task plans for field devices through the work management function provided by the HMI. Figure 21 shows registering and monitoring the task plan. The HMI can control the physical field through a digital twin synchronized to the virtual environment by command the physical device to execute the tasks according to the registered task plan.

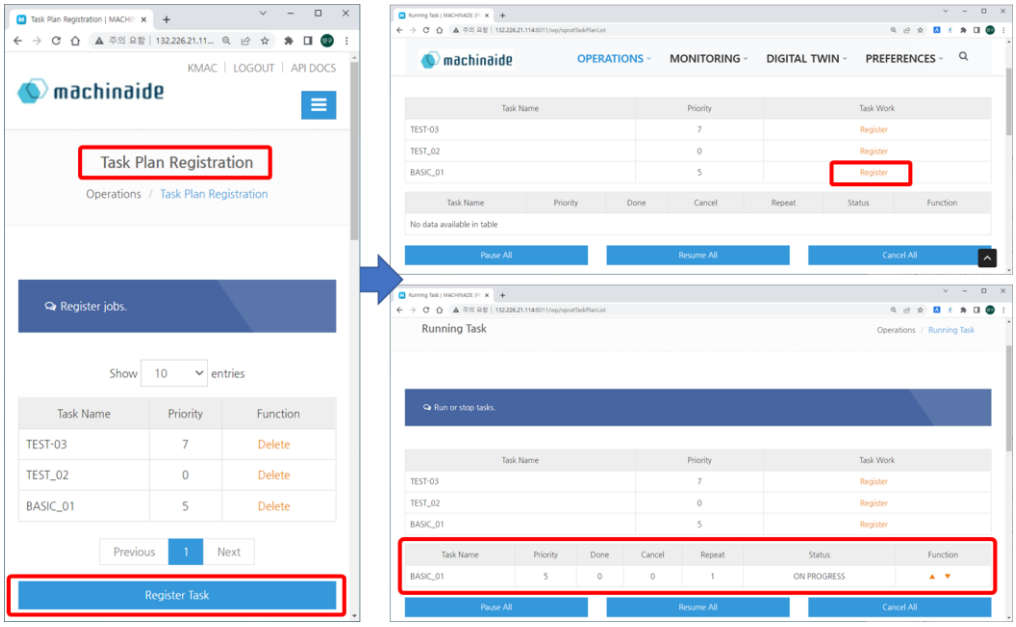

Figure 21. Task management in HMI

HMI provides a 3D visualization service for real-time monitoring of digital twins synchronized with physical assets. Figure 22 shows the 3D visualization process implemented by Unity engine. Users can monitor the digital twin in real-time by clicking Dashboard in the Monitoring menu. With digital twins realized through 3D visualization, operator can gain an intuitive view even outside of the real field, giving them insight into device operation.

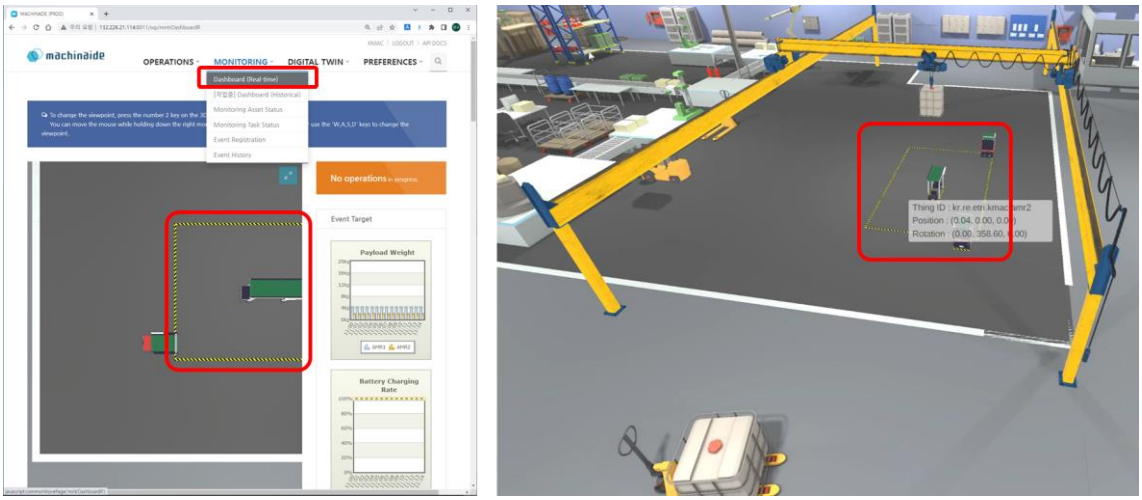

Figure 22. Real-time monitoring in HMI

When an operator wants to receive an alarm in response to a specific situation, the desired event can be registered and processed on the HMI. Figure 23 shows an example in which the user registers a specific event to the field devices. Events registered by Event Registration menu are linked with digital twin and when an event corresponding to a specific condition is detected, the event is reported to the operator so that the operator can actively control the device.

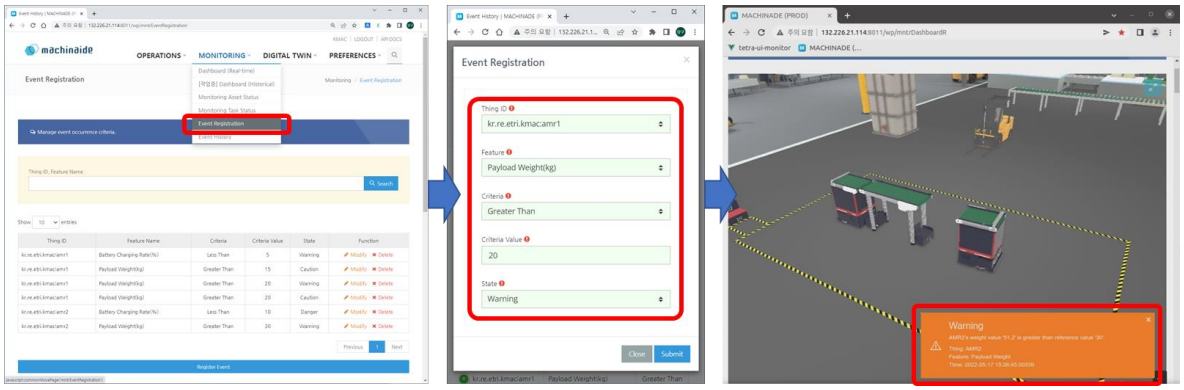

Figure 23. Event registration in HMI

3.6. Implementation schedule for demonstration

Action	Starting Term	Ending Term	Responsible
Usecase and Requirements definition	**2020/7**	**2021/6**	**ETRI**
Determination of DT platform spec.	**2020/7**	**2021/6**	**ETRI**
Determination of software for data management (MariaDB, InfluxDB, MongoDB)	**2021/1**	**2021/6**	**ETRI, CIP**
Determination of Testbed elements	**2021/1**	**2021/6**	**ETRI**
Building Testbed (AMRs, Conveyor, IoT sensors)	**2021/7**	**2022/2**	**ETRI**
Implementation of DT platform	**2021/7**	**2022/2**	**ETRI, CIP**
Implementation of Data management module	**2022/1**	**2022/6**	**ETRI, CIP**
Integration of software and hardware in testbed	**2022/1**	**2022/6**	**ETRI**
Creation of DT model	**2022/1**	**2022/6**	**ETRI**

Building multiple databases	**2021/7**	**2022/12**	**ETRI, CIP**
Collection of DT data	**2022/1**	**2022/12**	**ETRI, CIP**
Operation of testbed	**2022/1**	**2022/12**	**ETRI**
Development of event processing	**2022/1**	**2022/6**	**ETRI**
Development of HMI	**2022/1**	**2022/12**	**CIP**

1. Abbreviations

AMR	Autonomous Mobile Robots
API	Application Programming Interface
DBMS	Database Management system
DT	Digital Twin
GUI	Graphical User Interface
HMI	Human Machine Interface
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
PK	Primary Key
RDB	Relational Database
ROS	Robot Operating System
TSDB	Time Series Database
UI/UX	User Interface/User Experience
WAS	Web Application Server