University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Industrie 4.0-compliant Digitalization of Real-time Operational Data of a Pick and Place Module within a Digital Factory

Author:

Carlos Josue Rene Avila Carrillo (7025691)

Examiners:

Prof. Dr. Armando Walter Colombo
M.Eng. Jeffrey Werman

M.Eng. Industrial Informatics - Semester Project - SS 2024

September 13, 2024

Hochschule Emden/Leer · Constantiaplatz 4 · 26723 Emden · http://www.hs-emden-leer.de

Contents

List of Figures

List of Tables

Abstract

asdfasdf # Preamble

0.1 Abbreviations

AAS GUI I4.0

1 Introduction

The fourth industrial revolution is bla bla bla and aims to enable bla bla bla digital transformation of manufacturing and production industries. In a general sense this concept is known by other names such as: smart manufacturing, industrial internet of things (IIoT), to name a few. Mention something about digital twins too!!!!!

The Plattform Industrie 4.0[1] is a German government initiative aimed at advancing the the fourth industrial revolution. The Plattform Industrie 4.0 name for this concept is **Industrie 4.0**.

1.1 Industrie 4.0

Industrie 4.0 refers to the intelligent networking of machines and processes for industrie with the help of information and communication technology. Its fundamental purpose is to facilitate cooperation and collaboration between technical objects (assets), which means they have to be virtually represented and connected [6].

1.1.1 RAMI4.0

The Plattform Industrie 4.0, in partnership with many other stakeholders, has created the **DIN SPEC 91345**. This DIN SPEC describes the RAMI4.0 which is a reference architecture model and provides an architecture for technical objects (assets) in the form of layers, and allows them to be described, tracked over their entire lifetime and assigned to technical and/or organizational hierarchies. It also describes the structure and function of Industrie 4.0 components as essential parts of the virtual representation of assets. [6]. Figure 1 shows a visual representation of the RAMI4.0.

[1] https://www.plattform-i40.de/IP/Navigation/DE/Home/home.html

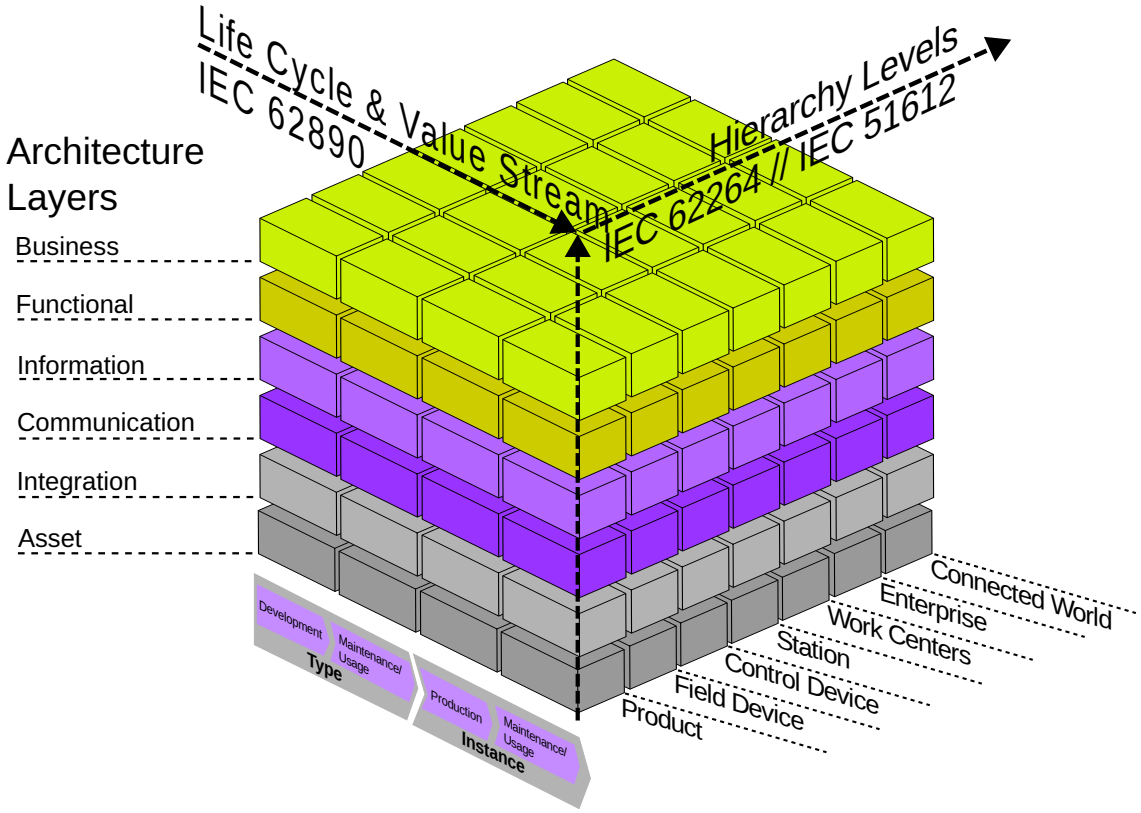

Figure 1: RAMI4.0 model

1.2 Digital Factory

The HS emden leer faculty of bla bla takes part in this overaching endeavor by creating an implementation of a factory following the principles of Industrie 4.0. This factory has academic and educational purposes and thus it is being transformed from a traditional production line into an I4.0-compliant work center by means of student projects, where each project iteratively augments and improves each work unit of the factory.

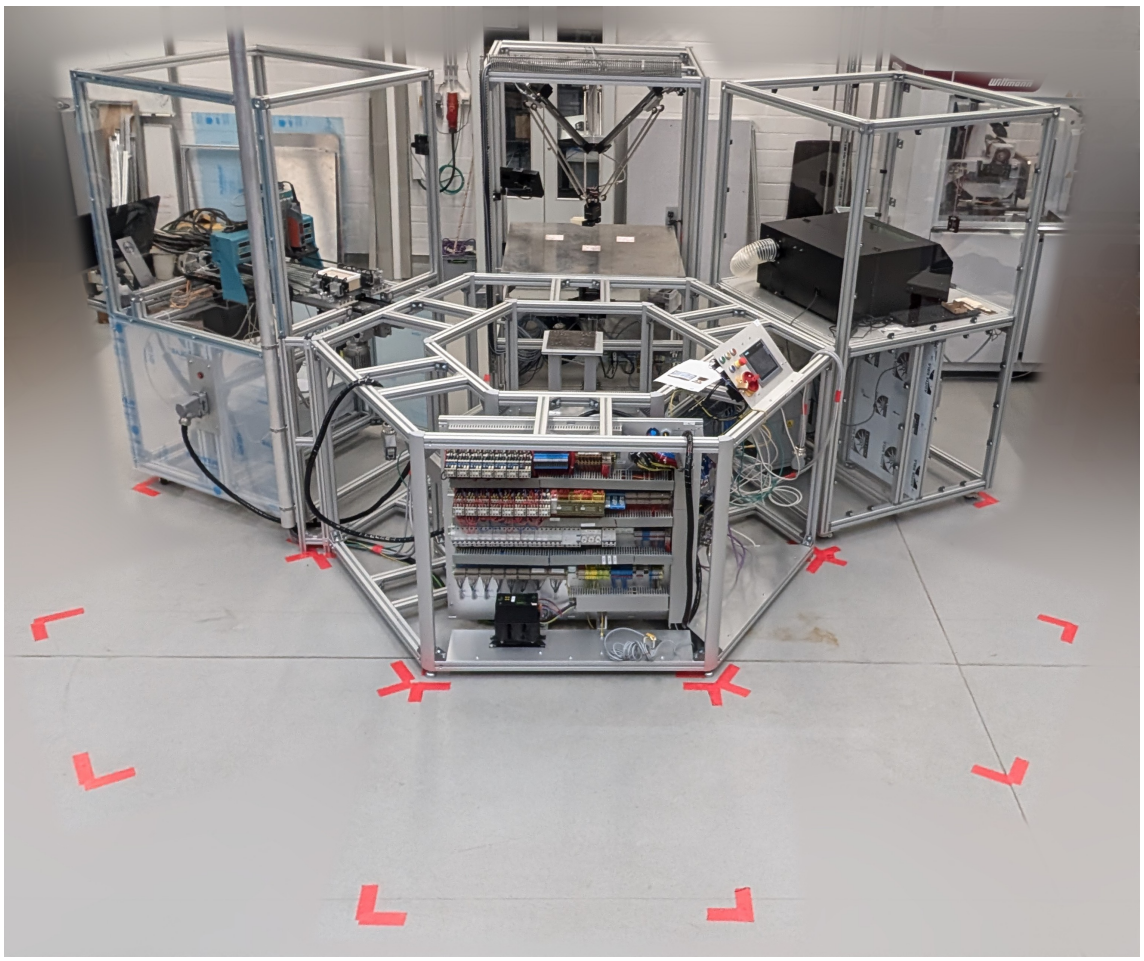

Figure 2: Digital Factory

1.2.1 Pick and place module

One of these work units is the pick and place module. Worked previously under the name of Delta robot, the module is an aluminum structure with XYZ dimensions blah blah bla bla. It is composed of the following harware and software:

- An Igus 3-axis Delta Robot with its iRC robot control which hosts a modbusTCP/IP service
- An in-house made gripper controlled with an ESP32 development board
- A Raspberry Pi XYZ
- A touch screen connected to the Raspberry Pi for desktop display
- An adjustable height table base

As a result from the previous work on the module there is a python library that interfaces with the robot control's modbusTCP service which greatly simplifies the interaction with the delta robot. There is also an Arduino based firmware

for the ESP32 to control the gripper and a python library that wraps the communication to the board using a serial interface.

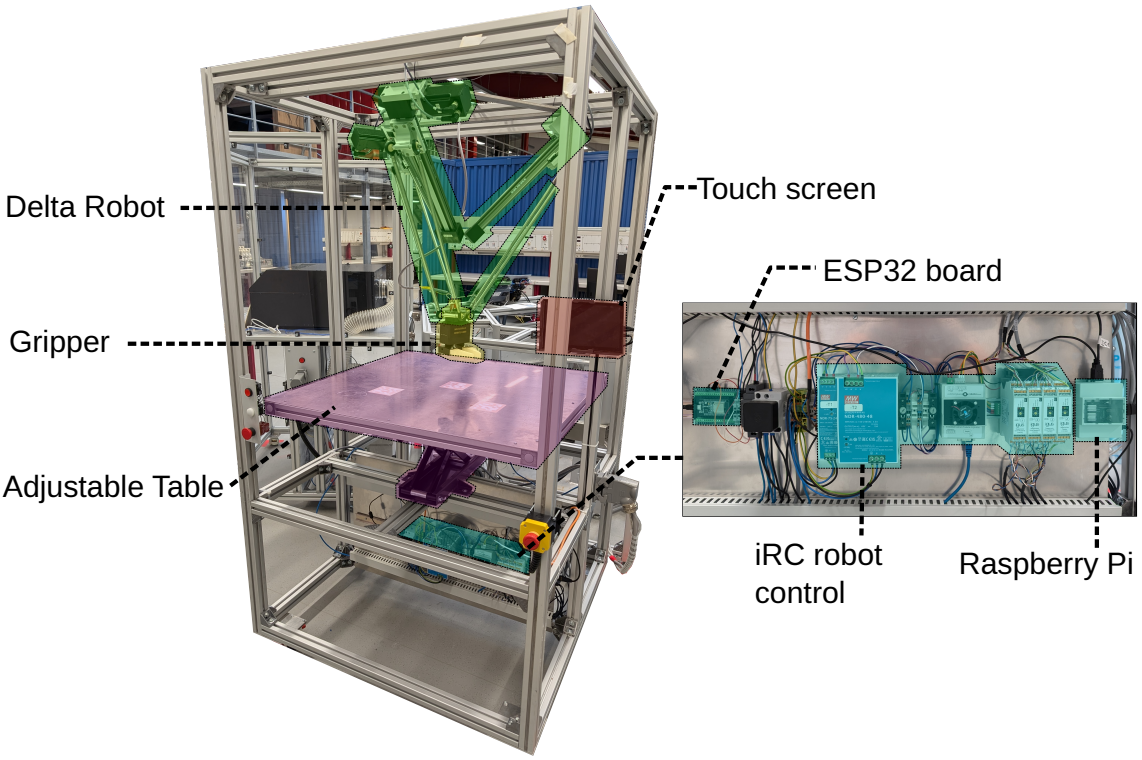

Figure 3: Pick and place module components

1.3 Objectives

The objective of this project is to present a functional implementation of the vertical integration, communication, and information of real-time operational data of an asset in the context of Industrie 4.0.

The main objective of this project is to present a functional implementation of the pick and place module as an Industry 4.0 component by exposing real-time operational data as a digital twin using an AAS served through an OPC UA server, allowing the module to be monitored and controlled through an I4.0 compliant communication interface.???

This document is not only a report on the semester project but also aims to be a manual or guide into how to use the RAMI4.0 as a conceptual tool to digitalize an asset.???

2 Methodology

This section describes how to use the RAMI4.0 as the theoretical basis to digitalize the pick and place module in the context of Industrie 4.0. The output of this section will provide the definitions, technologies, and requirements for the actual implementation.

2.1 Asset position in the RAMI 4.0

The RAMI4.0 is a reference model that when applied yields a more concrete, implementation-independent model of the digitalization of an asset. In a broad sense applying the RAMI4.0 involves three steps:

1. Determine the position of the asset in the Hierarchy Levels axis.
2. Determine the position of the asset in the Life cycle & Value stream axis.
3. Determine definitions, technologies and general requirements of each architecture layer.

Steps 1 and 2 help to determine **what** data and information is relevant to digitalize. Step 3 helps to determine **how** to digitalize the relevant data. This is, how the data and information is going to be made available to the business as shown in Figure 4.

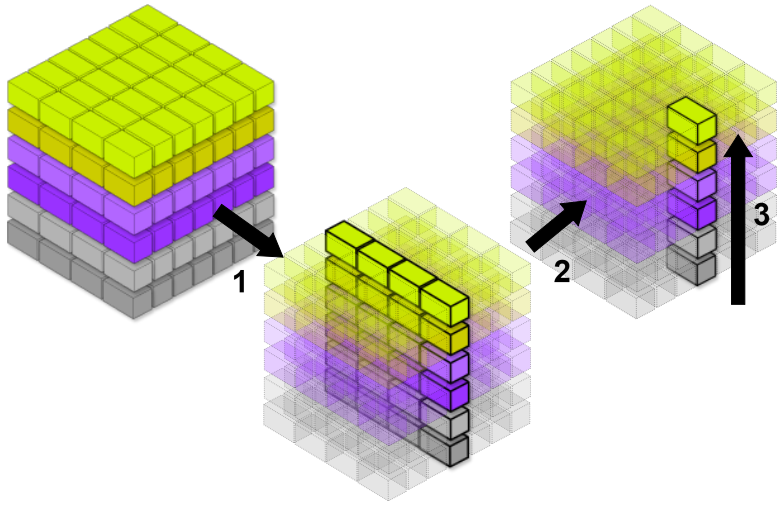

Figure 4: Three-step application of the RAMI4.0

The following three sections explain these steps in more detail.

2.1.1 Position in the Hierarchy Levels Axis

The Hierarchy levels axis of the RAMI4.0 is based on the role-based hierarchy model of the IEC 62264[5]. This axis describes the assets of an organization that are involved in the manufacturing and business processes. Figure 5 provides a broad picture of where the pick and place module is located in the hierarchy levels axis and how it relates to other components in the organization.

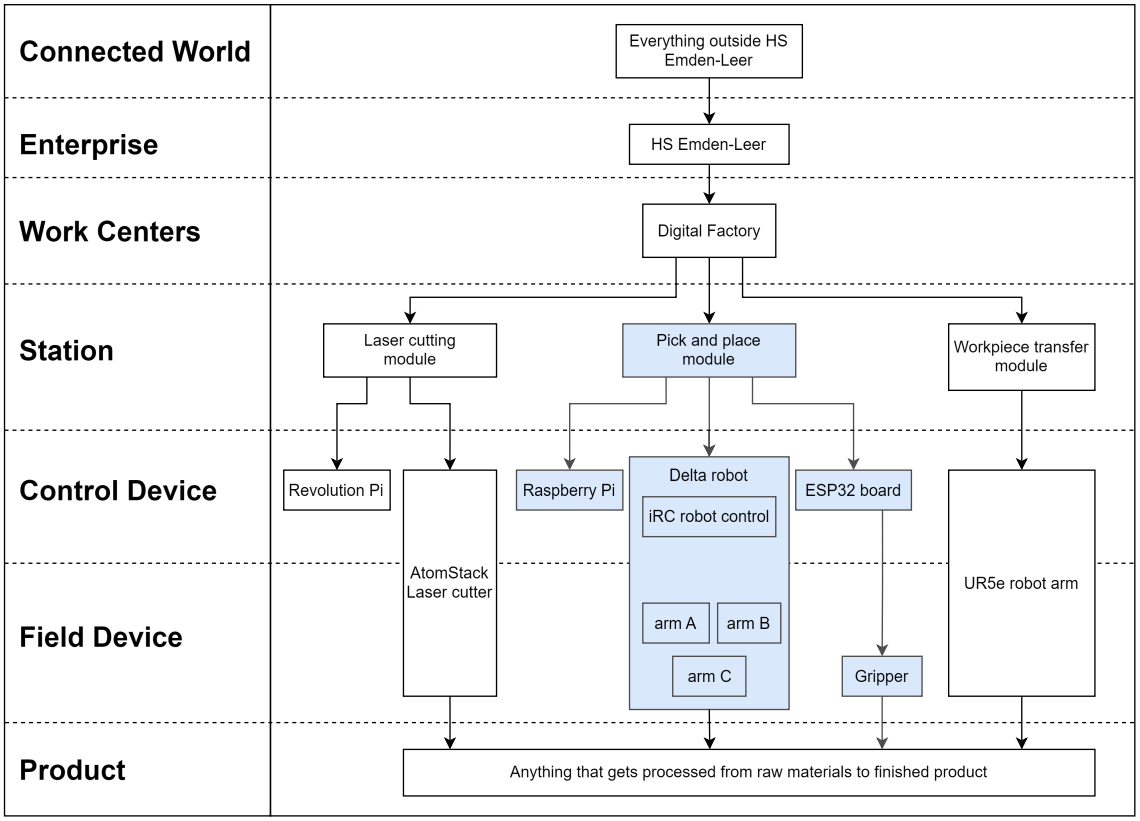

Figure 5: Hierarchy levels overview

A common, yet incomplete, definition of a station is an asset that is composed of sensors and actuators. A better definition of a station is an asset that has the equipment to manipulate a product (has sensors and actuators), has well-defined manufacturing capabilities and throughput capacities, it performs a segment of the manufacturing process, and is used for Level 3 functions (functions of manufacturing operations management). For more details on this refer to [5, chap. 5, “Hierarchy Models”]

The pick and place module is then classified as a **Station**.

> *From now on “pick and place module” and “pick and place station” refer to the same asset.*

2.1.2 Position in the Life Cycle & Value Stream Axis

The Life cycle & value stream axis is used to describe an asset at a particular point in time during its lifetime, from its conception and design, to its production and value-added use right up to its disposal[6].

Because the main goal is to digitalize real-time operational data then the pick and place station can be classified as an **instance** in **Usage** phase.

2.1.3 Architecture Layers

The Architecture Layers axis describes the digitalization architecture in terms of properties and system structures with their functions and function-specific data in the form of layers [6].

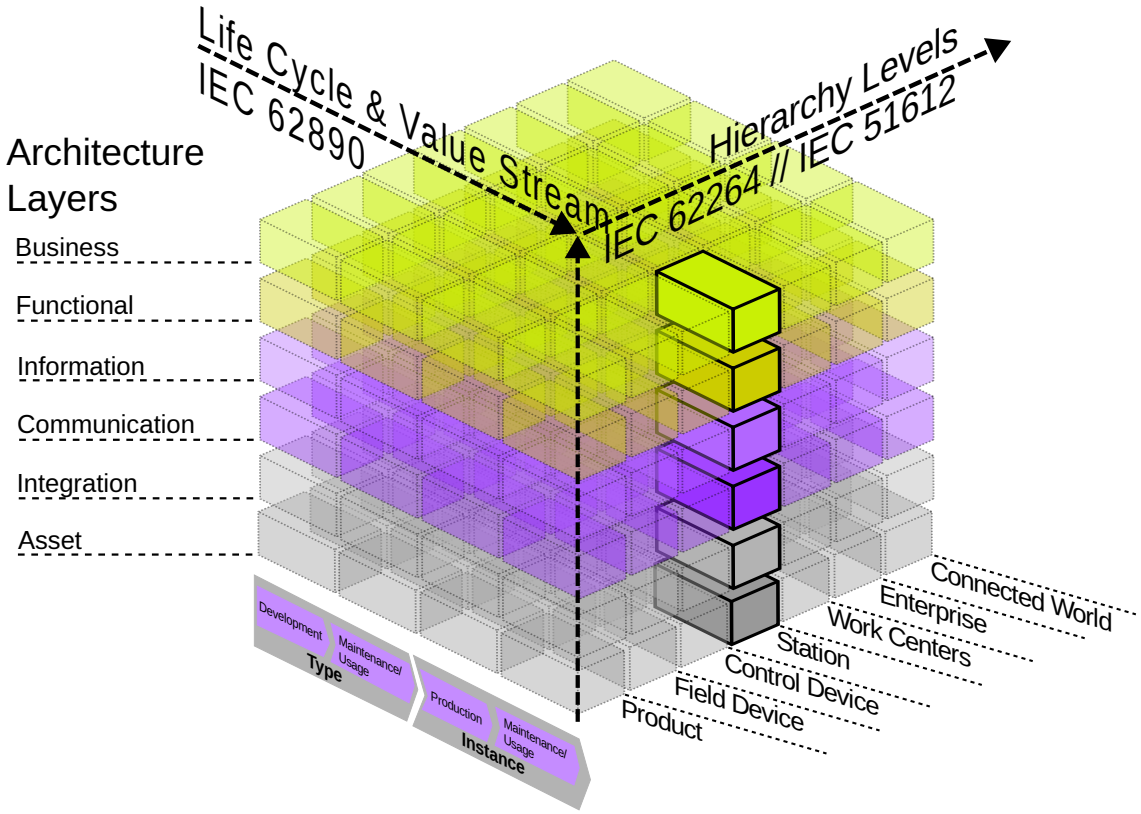

Figure 6: Architecture layers for station instance in maintenance/usage phase

By applying the RAMI4.0, the scope of the generic model is reduced and focused on the digitalization of the maintenance/usage data of a station instance. A visual representation of the result of applying the RAMI4.0 is shown in Figure 6.

A more concrete description of a solution can now be provided. The following subsections provide a description of each layer in relation to the pick and place station as well as the definitions, technology, and other general requirements for implementation. The actual description of the implementation will be described in Section 3.

2.1.3.1 Business Layer

The business layer describes the commercial view. This can be understood not only as profit but also as what gives value to the organization.

The context in which the pick and place station provides value will be defined as: **Academic research and implementation of I4.0-compliant technologies**.

2.1.3.2 Functional Layer

This layer describes the logical functions that enable the business. In the context of Industrie 4.0 these functions are defined as Capabilities. Capabilities are implementation-independent descriptions of the function of a resource to achieve a certain effect in the physical or virtual world[1].

Capabilities can be described at various levels of abstraction allowing them to be more or less specific as well as composed of other capabilities. For example: "Transport" is a capability. "Pick and place" is a more specific way to transport so it can also be a capability. Further more, "Pick and place" can be composed of the "Grip" and "Move" capabilities.

Capability descriptions are outside the scope of this project, still they provide a way to represent the logical functions defined in this layer.

For the pick and place station a logical function would be the **"Pick and place"** capability.

2.1.3.3 Information Layer

This layer describes the data and information that is used by the functions in the functional layer.

Industrie 4.0 introduces the concept of the Asset Administration Shell (AAS) submodels. AAS Submodels are representations of different aspects of an asset and are used to organize the data and information and provide a separation of concerns[2, Annex A, section V, "The Concept of Submodels"].

Using the AAS is a requirement for this project.

Because this project is focused on digitalizing real-time operational data then the information layer will include an **"Operational Data" submodel**. This submodel will contain properties and operations providing data such as the position of the delta robot as coordinate properties, a move operation, status of the robot movement, and so on.

2.1.3.4 Communication Layer

This layer describes the access to the information in an Industrie 4.0-compliant way. In other words, how to locate, read, and write the information.

Part of the AAS definition is the description of services, interfaces, and interface operations to access the information within. At the technology-specific level the information in the AAS can be accessed through HTTP, OPC UA, and MQTT interfaces[3].

As a requirement for this project the **communication must be done through OPC UA**.

2.1.3.5 Integration Layer

This layer describes how to read and write data from the asset (the physical world) into the information world. In other words, this layer represents a bridge that allows the communication layer to interact with the asset.

An **HTTP API** will be used to integrate the asset's data with the upper layers.

2.1.3.6 Asset Layer

This layer describes the real world. In this case the asset is the **pick and place station**.

Figure 7 shows an overview of the technologies and other definitions to be used for the architecture layers.

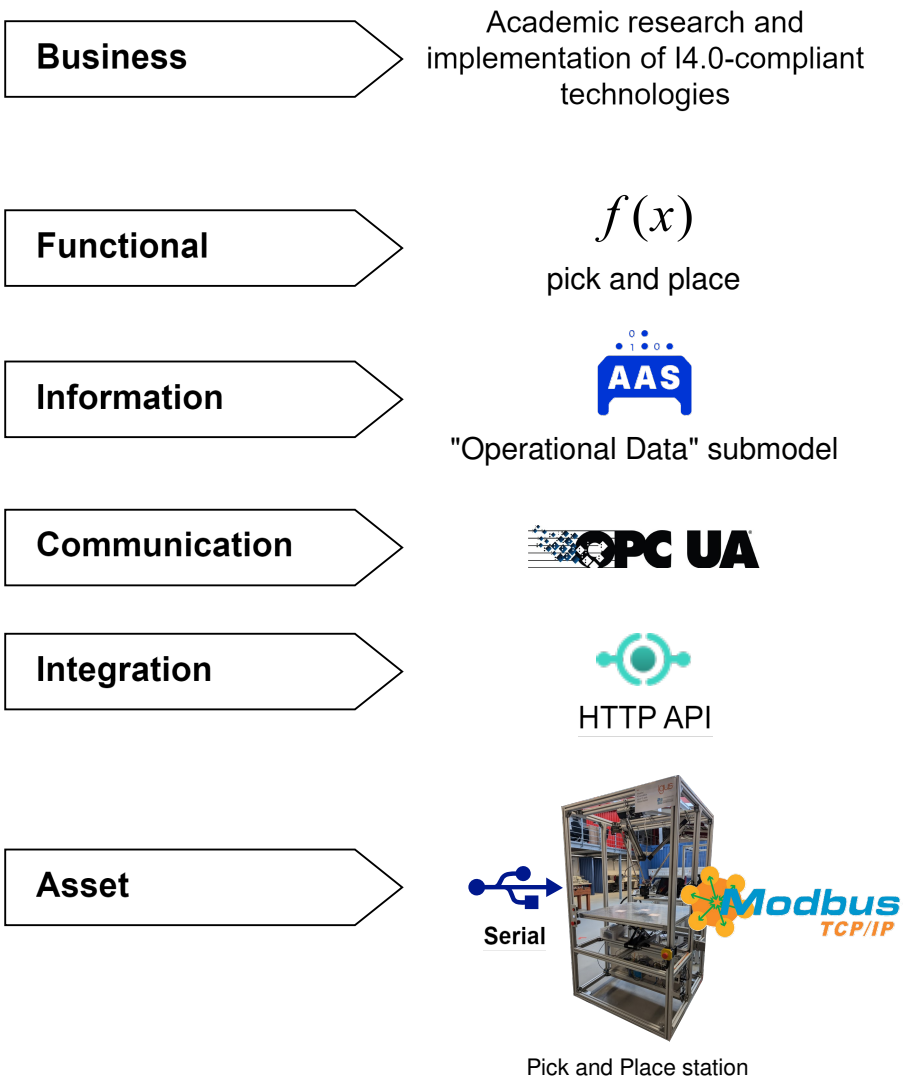

Figure 7: Architecture Layers technologies

3 Implementation

Following from the architecture layers definitions, technologies, and requirements described in Section 2.1.3 the information layer is to be implemented using AAS submodels and the communication layer is to be implemented as an OPC UA service.

3.1 AAS and the AASX Package Explorer

The AASX Package Explorer is a tool to view, create, and edit AAS[4]. Version 3 of the AASX Package Explorer was used to create the information model of the AAS of the pick and place station. The general structure is as follows:

- AAS: PickAndPlaceAAS
  - Submodel: OperationalData
    - Property: isModuledInitialized
    - Property: isModuleBusy
    - Property: robotSpeed
    - Property: tableDistance
    - SMC: robotPosition
      - Property: x
      - Property: y
      - Property: z
    - SMC: gripperPosition
      - Property: opening
      - Property: rotation
    - Operation: initializeModule() -> result
    - Operation: moveRobot(x, y, z, speed) -> result
    - Operation: moveGripper(opening, rotation) -> result
    - Operation: pickAndPlace(xInitial, yInitial, xFinal, yFinal, objectWidth, objectHeight) -> result

Figure 8 shows the GUI of the AASX Package Explorer with the OperationalData submodel expanded out.

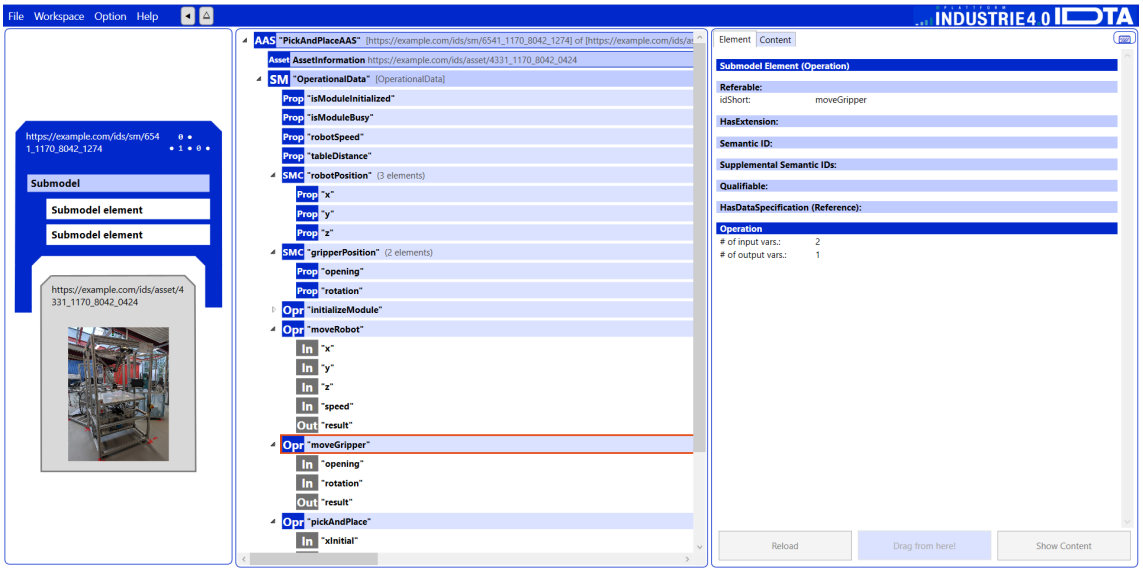

Figure 8: AAS Package Explorer GUI

3.2 FA³ST Service

This projects presents a new way to implement an AAS using the FA³ST service[2] tool which takes the AASX file and directly exposes the information model as an OPC UA service. The FA³ST service also provides an interface to configure how to link a property or an operation with an underlying HTTP endpoint, thus enabling real-time inuput/output of data from the asset. Figure 9 shows in a glance how the FA³ST service works. ??????

[2] https://www.iosb.fraunhofer.de/en/projects-and-products/faaast-tools-digital-twins-asset-administration-shell-industrie40.html

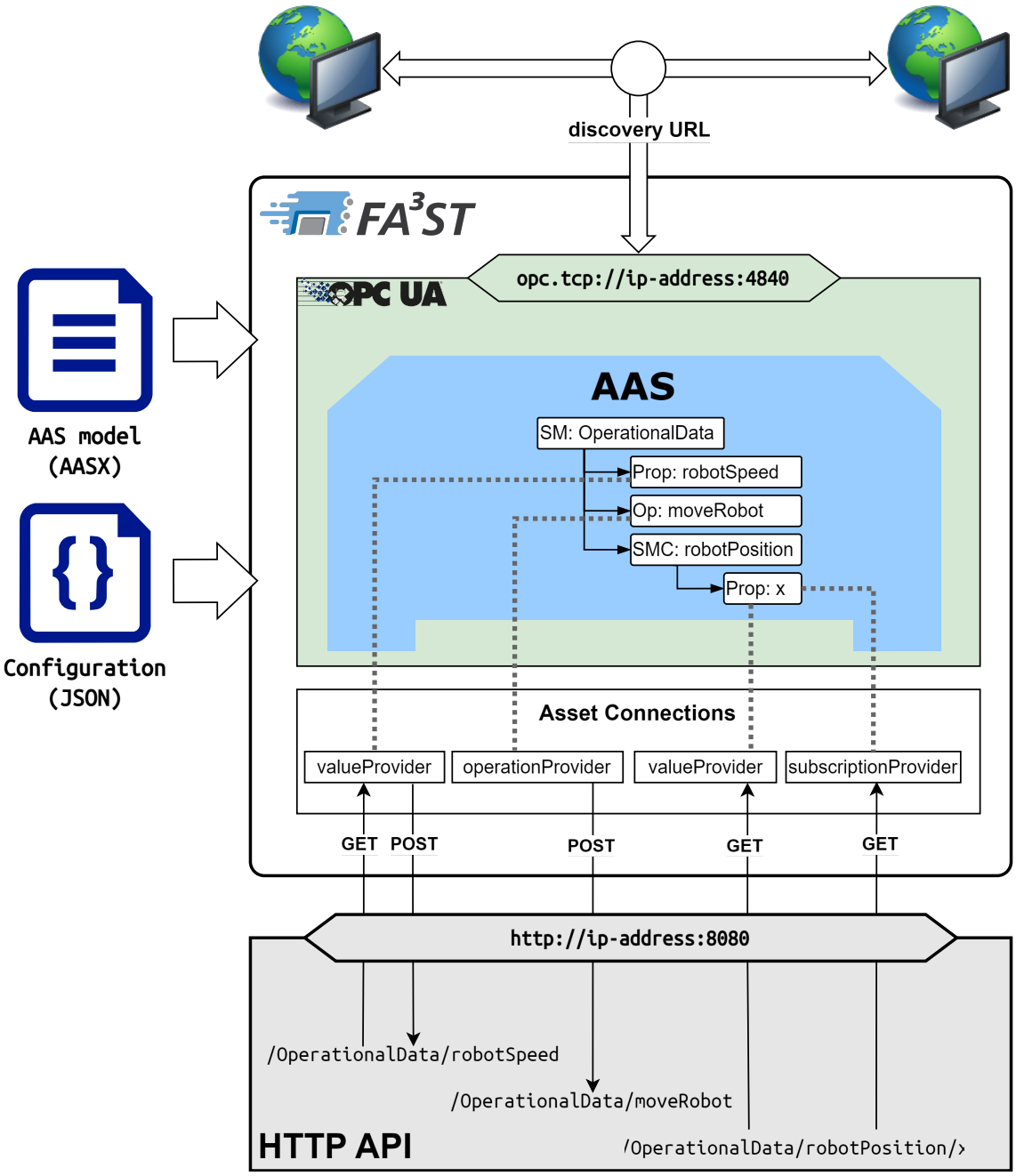

Figure 9: FA³ST service setup

3.2.1 Endpoint and Asset Connections

To expose the AAS through an OPC UA service as well as to synchronize the AAS with the pick and place station data the FA³ST service is configured with an OPC UA endpoint[3] and an HTTP asset connection[4]

[3] https://faaast-service.readthedocs.io/en/latest/interfaces/endpoint.html#opc-ua
[4] https://faaast-service.readthedocs.io/en/latest/interfaces/asset-connection.html#http

Two design considerations were used to keep track of all the mappings between the AAS submodel elements and the HTTP API endpoints, as well as to decouple their implementation:

1. The HTTP API endpoints will use the same path hierarchy as the AAS information model, starting from the submodel name and respecting the case.
2. All payload data (in requests and responses) must be in JSON format and must be structured as an object with the actual data in the `data` key. Examples are:

```
// scalars
{"data": 33.9}
// objects
{"data": {"success": false, "msg": "compilation error"}}
```

3.2.1.1 Read and write properties

Figure 10 shows an example of how read and write work on a property. This is valid for value and subscription providers. Notice that to read a value from the HTTP API the asset connection uses a GET request (GET requests don't have payload) and to write a value to the HTTP API it uses a POST request with a payload structured as mentioned. The response payload of the POST request does not matter because the AAS property is updated independently.

If a POST request fails (response returns anything other than a 2XX status code) then the AAS property is not updated.

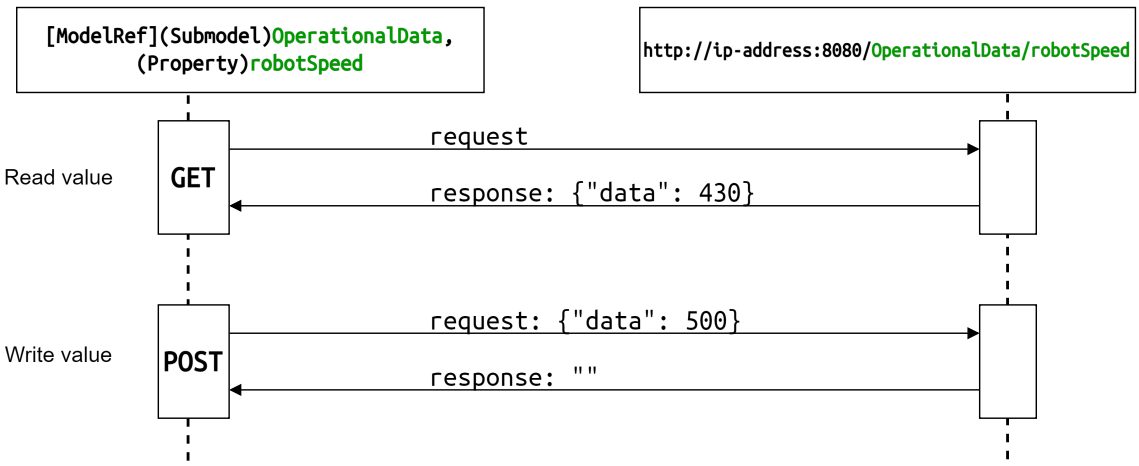

Figure 10: FA³ST asset connection for property read/write

3.2.1.2 Calling operations

Figure 11 shows an example of how an operation call works. This is valid for operation providers. Notice that the asset connection uses a POST request with a payload structured so that the input values of the operation are mapped to a key with the same name.

If an operation returns a value then it must be returned in the `data` key of the response payload.

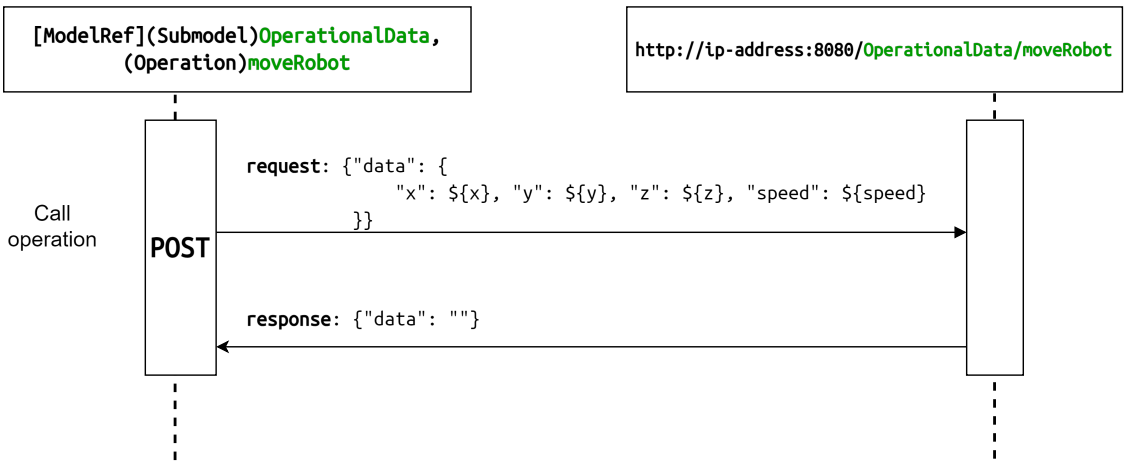

Figure 11: FA³ST asset connection for operation call

3.3 Hardware Integration

3.3.1 Raspberry Pi

A raspberry pi is used as the central hardware and software platform. It provides the computational and communication capabilities to communicate with the delta robot as well as the gripper as well as host the AAS over an OPC UA service.

3.3.2 Delta robot

The delta robot is controlled by its own robot control system. Thes system exposes a modbusTCP service which can be used to interact with the robot over a TCP network.

The python library from the previous project was refactored and improved. The most important improvement is on the homing sequence that initializes the robot. This homing sequence has to be run everytime the robot is started.

3.3.3 Gripper

The gripper is controlled by an ESP32 board. The communication between the ESP32 board and the Raspberry Pi is over a serial interface physically connected through a USB cable.

This project improves the gripper control firmware. Now the communication is done using JSON formatted messages which allow to issue commands to open and close the gripper as well as request status data.

For example, the following command rotates the gripper to the 120 degree position:

```
{"action": "rotate", "value": 120, "relative": false}
```

For more details refer to the README documentation on the project's source at `integration/esp32/README.md`.

3.3.4 Integration App

A python application was created to integrate the delta robot and the gripper and expose their functions over an HTTP API.

This API comes as a necessity because the FA³ST service asset connection only supports OPC UA, HTTP, and MQTT integrations. From these three options an HTTP API is the easiest to implement. See Section 3.2.1 for more information on how these two are integrated.

For more information on this app refer to the app's entrypoint script on the project's source at `integration/app/__main__.py`

3.4 Docker run environment

Finally, in order to keep the implementation free from software compatibility and integration issues, it was decided to use Docker containers as run environment. This way ???????

Figure 12 shows a diagram of the actual implementation.

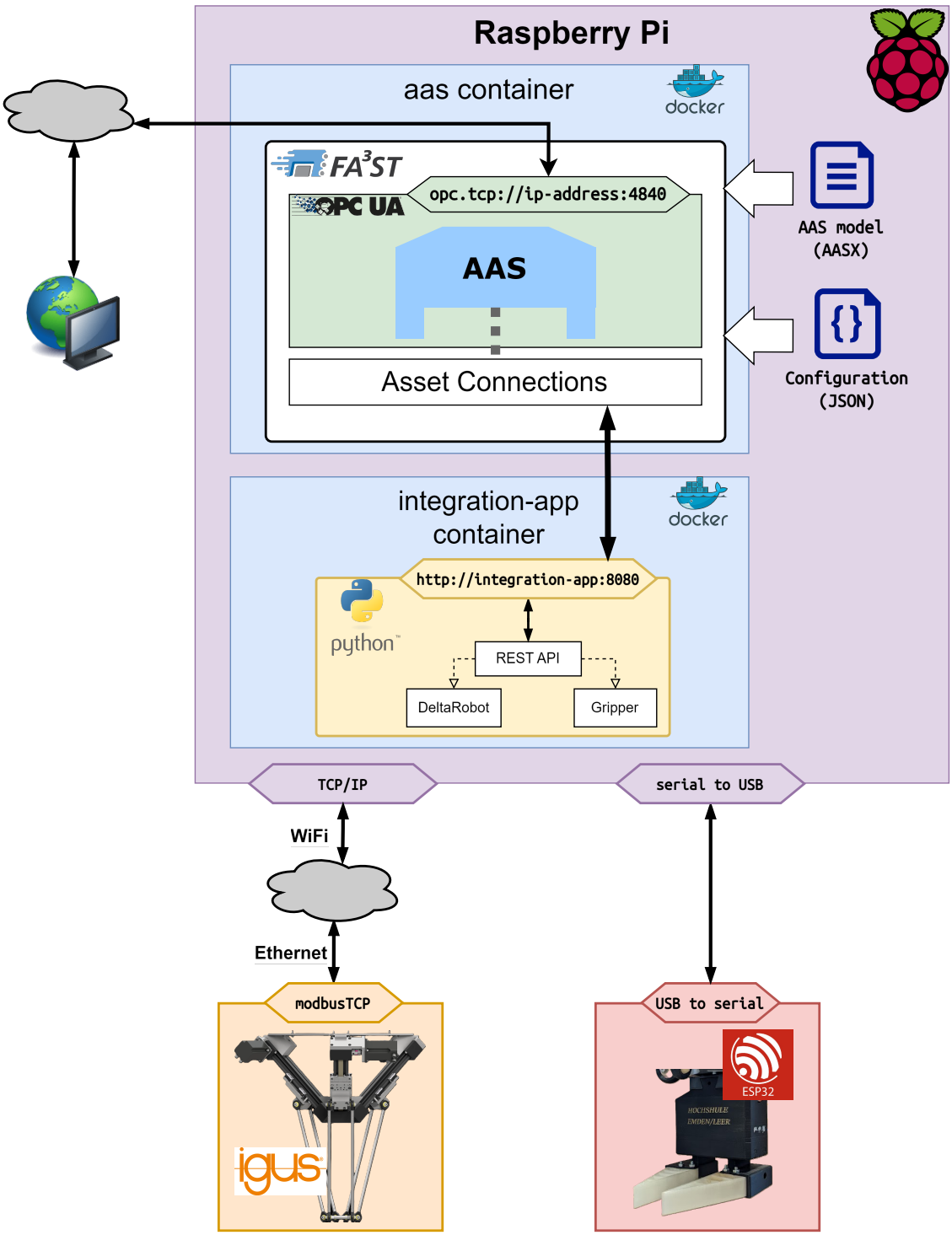

Figure 12: Pick and place station implementation

4 Results

5 Discussion

A common way to implement the communication of an AAS is to export the AAS information model to an XML file as the OPC UA NodeSet schema, then use a programming language with a suitable framework which can take the XML file as input and create an OPC UA server following the schema. This option was not considered an option as it has been already used in other projects and it was considered a cumbersome way to proceed.

A second option to consider was the AASX Server[5] tool. While it does allow the implementation of the OPC UA server directly from the AASX file it does not have a native way to interact with the underlying asset. This means that the service implemented cannot provide real-time input/output of data.

During the creation of this project the gripper was damaged to the point where it no longer opens or closes. The rotation still works but it is not precise. Despite these problems, the gripper was never able to actually grip something in a useful way.

[5] https://github.com/eclipse-aaspe/server

6 Annex A

6.1 Asset Connection Details

```
"assetConnections": [
  {
    "valueProviders": {
      "[ModelRef](Submodel)OperationalData, (Property)robotSpeed": {
        "path": "/OperationalData/robotSpeed",
        "format": "JSON",
        "template": "{\"data\": ${value} }",
        "writeMethod": "POST",
        "query": "$.data"
      },
      "[ModelRef](Submodel)OperationalData,
       ↪  (SubmodelElementCollection)robotPosition, (Property)x": {
        "format": "JSON",
        "path": "/OperationalData/robotPosition/x",
        "query": "$.data"
      }
    }
  }
]
```

The valueProviders object defines all the 1. The key value is a string referencing the path of the property in the AAS model 2. The path where the HTTP resource is located. Notice it matches the hierarchy of the property in the AAS model 3. The data format of the payload of the HTTP request and response 4. For properties that can be written, this denotes the JSON structure of the payload data 5. For properties that can be written, this denotes the HTTP method used to send the data. This can be POST or PUT. 6. The dot notation path locating the data to read from the HTTP response. The AAS property value will be set to this data.

Notice that for properties that are read-only, such as `OperationalData.robotPosition.x`, the `template` and `writeMethod` are removed.

7 References

[1] Plattform Industrie 4.0. *Capabilities of the Industrie 4.0 Components - Basics for the Development of Ecosystems*. 2020. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.pdf?__blob=publicationFile&v=1.

[2] Plattform Industrie 4.0. *Details of the Asset Administration Shell Part 1 Version 3.0*. 2023. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.pdf?__blob=publicationFile&v=1.

[3] Plattform Industrie 4.0. *Details of the Asset Administration Shell Part 2 Version 1.0*. 2023. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.pdf?__blob=publicationFile&v=1.

[4] Eclipse Foundation. *Eclipse AASX Package Explorer*. Accessed: 2024-09-06. 2024. URL: https://github.com/eclipse-aaspe/package-explorer.

[5] *IEC 62264-1: Enterprise-control system integration - Part 1: Models and terminology*. Second edition. International Electrotechnical Commission, 2013. URL: https://webstore.iec.ch/publication/6675.

[6] *Reference Architecture Model Industrie 4.0 (RAMI 4.0)*. DIN SPEC 91345. Berlin, Germany: Deutsches Institut für Normung, Apr. 2016. URL: https://www.din.de/resource/blob/229091/38ec5291c94e5d7e5e7bce7b3fc4c06e/din-spec-91345-pdf-data.pdf.