



Hochschule für Technik  
und Wirtschaft Berlin

University of Applied Sciences



## Developing a comprehensive Digital Twin Solution for a Battery Management System in Intra-Logistic Robots

---

### Project Overview and Documentation

#### Supervisors

Prof. Ute Dietrich, Charlene Zander

#### Project Team

Tuna Tröndle, Samuel Molloy, Azmaeen Hoque, Mandana Ebrahimian

#### Project Deadline

2024-07-29

## TABLE OF CONTENTS

---

<b>1.</b>	<b><i>Introduction.....</i></b>	<b><i>1</i></b>
1.1.	Project Overview .....	1
1.2.	NEURA MAV 500 Specifications and Features .....	1
<b>2.</b>	<b><i>The System .....</i></b>	<b><i>1</i></b>
2.1.	Data source for development.....	1
2.1.1.	Portainer .....	4
2.1.2.	Node-Red .....	4
2.1.3.	Flow Explanation.....	6
2.1.4.	MQTT and Mosquitto.....	7
2.1.5.	InfluxDb.....	9
2.1.6.	Grafana.....	14
2.2.	Eclipse BaSyx.....	17
2.3.	AAS Server.....	17
2.3.1.	Functionality in Our Context.....	18
2.3.2.	Benefits for Our Work .....	19
2.3.3.	Server Deployment.....	19
2.3.4.	How to deploy the server locally with Docker .....	20
2.4.	AASX Package Explorer V3.....	26
2.4.1.	Key Features .....	26
2.4.2.	Benefits.....	26
2.5.	Connecting Sensors to AASX.....	27
<b>3.</b>	<b><i>Discussion.....</i></b>	<b><i>32</i></b>
3.1.	Challenges .....	32
3.1.1.	Navigating Industry 4.0 .....	32

3.1.2.	Configuration of Components.....	32
3.1.3.	Working with Open Source.....	33
<b>4.</b>	<b><i>Glossary</i>.....</b>	<b>34</b>
<b>5.</b>	<b><i>Appendix</i>.....</b>	<b>37</b>
5.1.	CAN-Bus, CANopen, and Node ID.....	37
5.2.	PDO .....	38

## 1. INTRODUCTION

---

### 1.1. PROJECT OVERVIEW

This project, in collaboration with InMediasP GmbH, aimed to develop a digital twin for the battery management system (BMS) of an intra-logistic robot named MAV. The project spanned six months. We leveraged the Asset Administration Shell (AAS) to seamlessly integrate real-time BMS data, including critical parameters like State of Charge (SOC) and State of Health (SOH). By ensuring compliance with the standards set by the Industrial Digital Twin Association<sup>1</sup>, we aimed to enhance transparency and optimize battery utilization phases, ultimately improving sustainability.

Creating a Digital Twin was however too broad of a scope for a 6-month project with University students. We were also unable to work on optimising the battery utilization phase. We achieved a Digital Shadow, that just visualises the battery statistics.

Our international team comprised of students from HTW Berlin and the University of Technology Sydney, bringing together a diverse range of IT and engineering backgrounds.

## 2. THE SYSTEM

---

### 2.1. DATA SOURCE: MAV & THE LAB

Our data source was an intra-logistic robot known as NEURA MAV 500. It was designed for industrial material handling applications – we only needed it for its industry battery system; please see **Figure 1** below for reference.

---

<sup>1</sup> <https://industrialdigitaltwin.org/>



**FIGURE 1: MAV ROBOT**

The MAV was equipped with 3 industrial batteries from the brand “VARTA”, specifically their “Easy Blade 48” batteries. Please see attached file titled **Appendix 3** for the specifications document of the battery. The batteries have a built-in BMS that allow communication through CAN-BUS<sup>1</sup> (CANopen) connection.

The MAV was contained within the HTW Berlin Lab; it was shared amongst a few groups for learning purposes.

The in-lab setup started from the CAN-Bus connection between the BMS of each battery and a Raspberry Pi<sup>2</sup>. The big-picture is that the Raspberry Pi collected the raw messages and processed them through functions deployed by a low-code logic platform known as Node-RED. After processing, the Node-RED logic/flow sent the data to Influx DB, a database platform specialised in time-series data. From this point, we were able to use the processed data in other platforms, such as developing visualisation on the Grafana platform and embedding the visualisation in the AAS platform.

Please see [Appendix 4](#) for notes on the application-level identification system of the batteries, and the amalgamation process of data from multiple batteries in one system.

The CAN-bus communication from the battery system to the Raspberry Pi is through the application-level object known as Process Data Object<sup>4</sup> (PDO). VARTA provides documentation for how to read the PDOs being sent out from the battery. Please see **Figure 2** below for specifications on single-battery PDOs. The specification is assuming the battery with Node ID 1 is sending the data. If a different Node ID sends the data, the “CAN-ID” increments. E.g. If Node ID 5 sends the data, the CAN-ID increments by 4 and becomes 0x185 for TPDO\_1. That is, the “PDO” column is identifying what kind of data is being sent, and “CAN-ID” is identifying both the data content and the battery node in a hexadecimal ID.

PDO	CAN-ID	Type	Node-ID	Data				Event
				Battery Voltage		Battery Average Current		
TPDO_1	0x181	PDO	1	uint32 [mV]		int32 [mA]		1s

PDO	CAN-ID	Type	Node-ID	Data				Event
				Max. FET Temp.	Max. Cell Temp.	Charge Voltage Req.	Charge Current Req.	
TPDO_2	0x281	PDO	1	int16 [0.1°C]	int16 [0.1°C]	uint16 [mV]	uint16 [mA]	1s

PDO	CAN-ID	Type	Node-ID	Data				Event
				Battery Cap.	Battery Full Cap.	Battery Rem. Cap.	not used	
TPDO_3	0x381	PDO	1	uint16 [mAh]	uint16 [mAh]	uint16 [mAh]		1s

PDO	CAN-ID	Type	Node-ID	Data				Event
				Information Status	Warning Status	Error Status	Charge Control Status	
TPDO_4	0x481	PDO	1	uint16	uint16	uint16	uint16	100ms

Abbildung 7: Single Battery CAN-Frame [45]

FIGURE 2: PDO SPECIFICATION FOR INDIVIDUAL BATTERY DATA

The master battery data is a summary of the whole system. It has its own PDO specifications; please see **Figure 3** below. Note that TPDO\_8 is summarising every state of each module/battery; this way only one PDO needs to be checked for monitoring the system for faults and errors. Additionally, Node-ID being 27 indicates that it is the master battery data.

PDO	CAN-ID	Type	Node-ID	Data			Event
				Master Voltage (Max. Battery Voltage)	Master Current (sum of all modules)		
TPDO_5	0x19B	PDO	27	uint32 [mV]	int32 [mA]		1s

PDO	CAN-ID	Type	Node-ID	Data			Event
				Max. Battery FET Temperature	Max. Battery Cell Temperature	Master Design Capacity (sum of all modules)	
TPDO_6	0x29B	PDO	27	int16 [0.1°C]	int16 [0.1°C]	uint32 [mAh]	1s

PDO	CAN-ID	Type	Node-ID	Data			Event
				Master Full Charge Capacity (sum of all modules)	Master Remaining Capacity (sum of all modules)		
TPDO_7	0x39B	PDO	27	uint32 [mAh]	uint32 [mAh]		1s

PDO	CAN-ID	Type	Node-ID	Data				Event
				Master Information Status Register	Master Warning Status Register	Master Error Status Register	Master Charge Control Status Register	
TPDO_8	0x49B	PDO	27	uint16	uint16	uint16	uint16	200ms

Abbildung 8: Battery Master CAN-Frame [45]

FIGURE 3: PDO SPECIFICATION FOR MASTER BATTERY DATA

#### 2.1.1. PORTAINER

Portainer serves as the foundation for the Raspberry Pi setup. It is a graphical user interface (GUI) to manage Docker containers in a user-friendly platform. It is also run within a docker container itself, thus it needs to be installed first through the normal Docker container setup process.

There are many container-management software available. Portainer is popular amongst them as an open-source lightweight option. There is no strict requirement to use this application for container management; the base Docker functionalities can be complicated but are enough for all management needs.

The container management needs for this project are very basic, thus Portainer is used to keep things simple.

#### 2.1.2. NODE-RED

Node-RED is an open-source programming tool that offers a low-code alternative for connecting hardware devices, APIs, and other services in an Internet of Things (IoT) context. It runs on Node.js (JavaScript) code.

#### Key Features

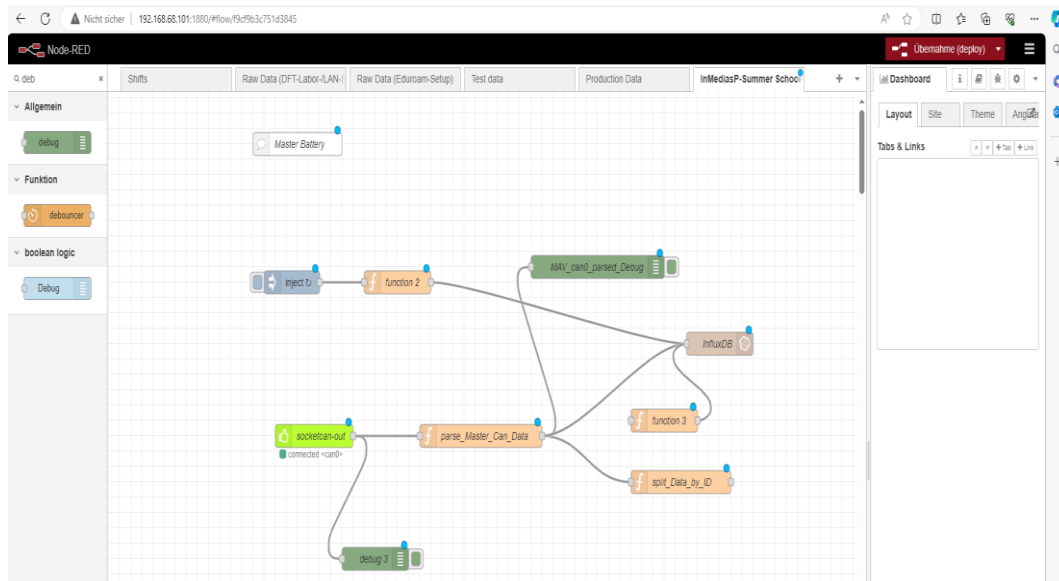
<u>Feature</u>	<u>Description</u>
Low-Code environment	Node-RED uses a flow-based programming approach, where functions and services are created by dragging nodes onto a workspace and connecting these nodes to transfer data between them. This creates a “flow” that is visualised by the nodes and connections. This low-code approach makes it easier to develop applications and reduces bugs and errors.
Node Library	Node-RED offers a library of predefined nodes for a variety of functions and integrations. Users can also create custom nodes with JavaScript code.
Real-Time Feedback	Changes to flows can be implemented and tested in real-time, reducing development time.
Integration with Platforms	Node-RED integrates with: <ul style="list-style-type: none"> <li>• IoT messaging protocols like MQTT</li> <li>• Databases like InfluxDB</li> <li>• Cloud platforms such as Microsoft Azure</li> </ul>

TABLE 1: KEY FEATURES OF NODE-RED

### Node-RED Flow

It was decided to use Node-RED, which was already established on the Raspberry Pi that was being used by the other lab researchers. We created a new flow as illustrated in **Figure 4** below. The general idea of this Node-RED flow is to handle incoming raw CAN-Bus data, process and parse it to extract meaningful information, then store this information in a cloud-base InfluxDB database for further use. The flow includes nodes for data injection, processing, splitting based on message IDs, and debugging to ensure the data is correctly handled at each stage.





**FIGURE 4: NODE-RED FLOW**

### 2.1.3. NODE-RED FLOW EXPLANATION

- Inject Node (inject)
  - Purpose: This node is used to manually inject a message into the flow, typically for testing purposes.
  - Details: It initiates the flow.
- Function Node (function 2)
  - Purpose: This node processes the incoming message using JavaScript code.
  - Details: It modifies the data from the inject node as needed before passing it to the next node.
- Debug Node (debug 3)
  - Purpose: This node is used to display messages in the debug sidebar of the Node-RED editor.
  - Details: Useful for troubleshooting and ensuring the flow works as expected.
- SocketCAN Output Node (socketcan-out)
  - Purpose: This is the source of the raw CAN-Bus data.
  - Details: It inputs the raw CAN-Bus data from the cable-connected network into the flow. This allows the raw data to be processed further down the flow.
- Function Node (parse\_Master\_Can\_Data)

- Purpose: This node parses the raw CAN data.
- Details: It processes the CAN data to extract relevant information.
- Function Node (split\_Data\_by\_ID)
  - Purpose: This node splits the data based on specific IDs.
  - Details: It routes the data to different paths based on the ID of the message.
- Function Node (function 3)
  - Purpose: This node further processes the split data.
  - Details: It handles additional data transformation as needed.
- InfluxDB Node (InfluxDB)
  - Purpose: This node writes data to an InfluxDB database.
  - Details: It stores the processed CAN data for later analysis.
- Debug Node (MAV\_can0\_parsed\_Debug)
  - Purpose: This node outputs the parsed data to the debug sidebar.
  - Details: Useful for verifying the parsed data

#### 2.1.4. MQTT AND MOSQUITTO

MQTT is a message-based protocol designed for Machine-to-Machine (M2M) communication, ideally for the Internet of Things (IoT). It enables data transmission between devices in an environment that often is resource-constrained, and/or a network that is overloaded. In our project, we use MQTT to transmit data from the Battery Management System (BMS) of a transport robot via a Raspberry Pi to Node-RED, allowing this data to be visualized in real-time on a Grafana dashboard.

##### Key Features of MQTT

Feature	Description
Lightweight and Efficient	<u>Low Overhead</u> <ul style="list-style-type: none"> <li>● MQTT causes minimal network overhead, making it ideal for environments with limited bandwidth.</li> </ul>

	<p><u>Resource-Conserving</u></p> <ul style="list-style-type: none"> <li>• It is suitable for devices with restricted resources, such as sensors and mobile devices.</li> </ul>
Pub/Sub Architecture (Publish/Subscribe)	<p><u>Asynchronous Communication</u></p> <ul style="list-style-type: none"> <li>• MQTT uses a publish/subscribe messaging model, where clients publish messages to topics and subscribe to topics of interest.</li> </ul> <p><u>Decoupling</u></p> <ul style="list-style-type: none"> <li>• This model decouples the producers and consumers of messages, allowing flexible and scalable communication.</li> </ul>
Reliability	<p><u>Quality of Service (QoS) Levels</u></p> <ul style="list-style-type: none"> <li>• MQTT offers different QoS levels, providing varying guarantees for message delivery (e.g., at most once, at least once, or exactly once delivery).</li> </ul> <p><u>Message State Management</u></p> <ul style="list-style-type: none"> <li>• It ensures messages are transmitted securely and reliably, even in unstable networks.</li> </ul>
Easy Implementation	<p><u>Widespread Support</u></p> <ul style="list-style-type: none"> <li>• There are numerous MQTT libraries and clients for various programming languages and platforms.</li> </ul> <p><u>Open Standards</u></p> <ul style="list-style-type: none"> <li>• MQTT is an open standard, ensuring interoperability between different systems and manufacturers.</li> </ul>

Security	<u>Encryption</u> <ul style="list-style-type: none"> <li>MQTT can use SSL/TLS to encrypt communication.</li> </ul> <u>Authentication</u> <ul style="list-style-type: none"> <li>The protocol supports various mechanisms for client authentication.</li> </ul>
----------	---

TABLE 2 MQTT FEATURES

#### 2.1.5. INFLUXDB

InfluxDB is an open-source time-series database designed to store and analyse data points collected at regular intervals. In our system, InfluxDB plays a crucial role in storing sensor data from the Battery Management System (BMS).

The time-series nature of InfluxDB is well-suited for the efficient storage and querying of sensor data, filtering through time constraints. This allowed us to analyse trends, identify patterns, and monitor the health and performance of the physical asset in real-time. For our system, we have used InfluxDB Cloud, hosted on Amazon Web Services, which allows access to all team members once they are added as users to the database.

##### Access and Querying Data

The database is accessible via the InfluxDB Cloud website and is linked to a user account. InfluxDB offers various tools, including ones that generally require some technical knowledge but enable more powerful and specific data queries.

##### Data Storage

InfluxDB stores data points based on the combination of measurement, series, and timestamp. This structure allows for quick querying of specific data segments. Data is usually compressed to optimize storage space, especially for large datasets collected over long periods.

## Data Retrieval

A range of data retrieval methods exist in InfluxDB, making it a flexible choice to suit a range of use-cases.

- InfluxDB UI Data Explorer      The InfluxDB user interface features a built-in data exploration tool that allows you to write queries directly in the browser.
- InfluxDB Command-Line Interface (CLI)      The Influx command-line tool enables interaction with InfluxDB from your terminal and the execution of queries.
- Client Libraries      InfluxDB provides client libraries for various programming languages (Python, Java, etc.). These libraries allow developers to integrate InfluxDB access and querying functionality into their custom applications.

## Limitations

While InfluxDB is a powerful tool that meets product requirements, it should be noted that there are also some limitations to its efficacy. The primary shortcoming is its comparatively high storage requirements. For large data volumes, InfluxDB can require significant storage space, which can increase resource costs as the service scales.

## Use Case

For our use case, Influx DB was set up on the cloud, leveraging AWS servers to ensure a secure and reliable environment. It is crucial to use cloud-based systems for enhanced safety, scalability, and ease of access. We created a data bucket with a retention period of 30 days to efficiently manage our data.

Using the cloud setup allows for better data protection and disaster recovery options, ensuring our data is always available when needed. To facilitate seamless integration and data retrieval, we generated an API token, enabling API calls for various programs, including Grafana in our case. This setup not only enhances our data management capabilities but also provides the flexibility to use other compatible applications and tools as required. It's a shared account system accessible by invitation from an admin, allowing invited users to access, modify the database, and connect data points.

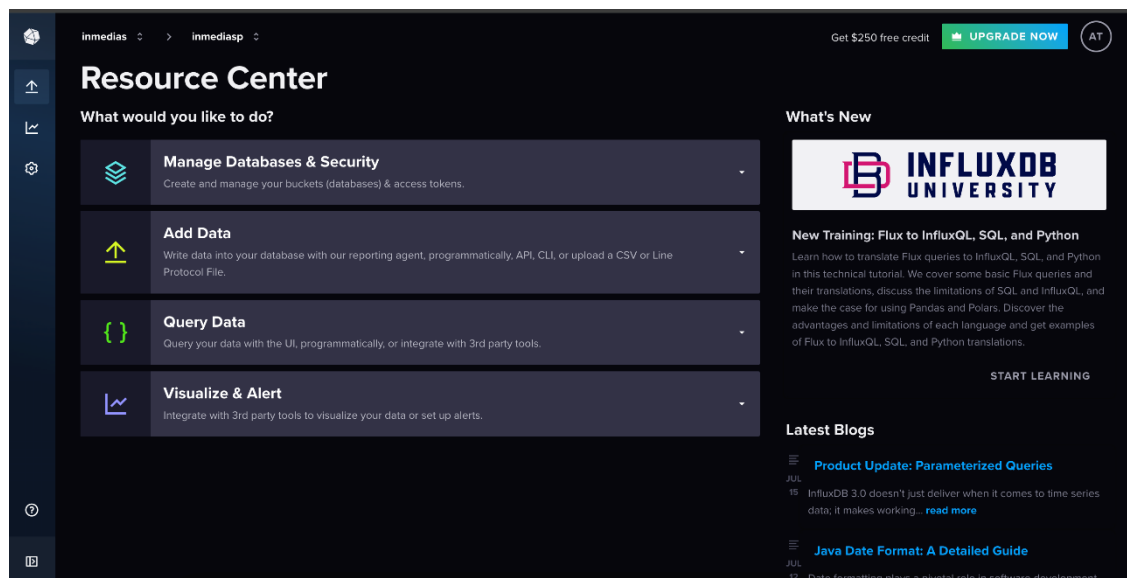


FIGURE 4: INFLUX DB HOME-PAGE FOR OUR PROJECT

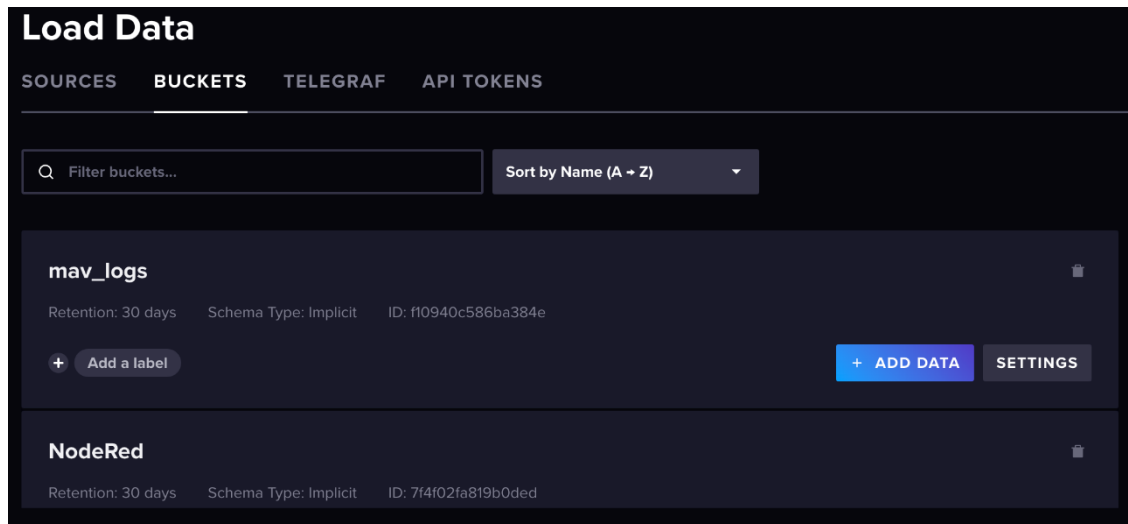


FIGURE 5: DATA BUCKET : MAV\_LOGS

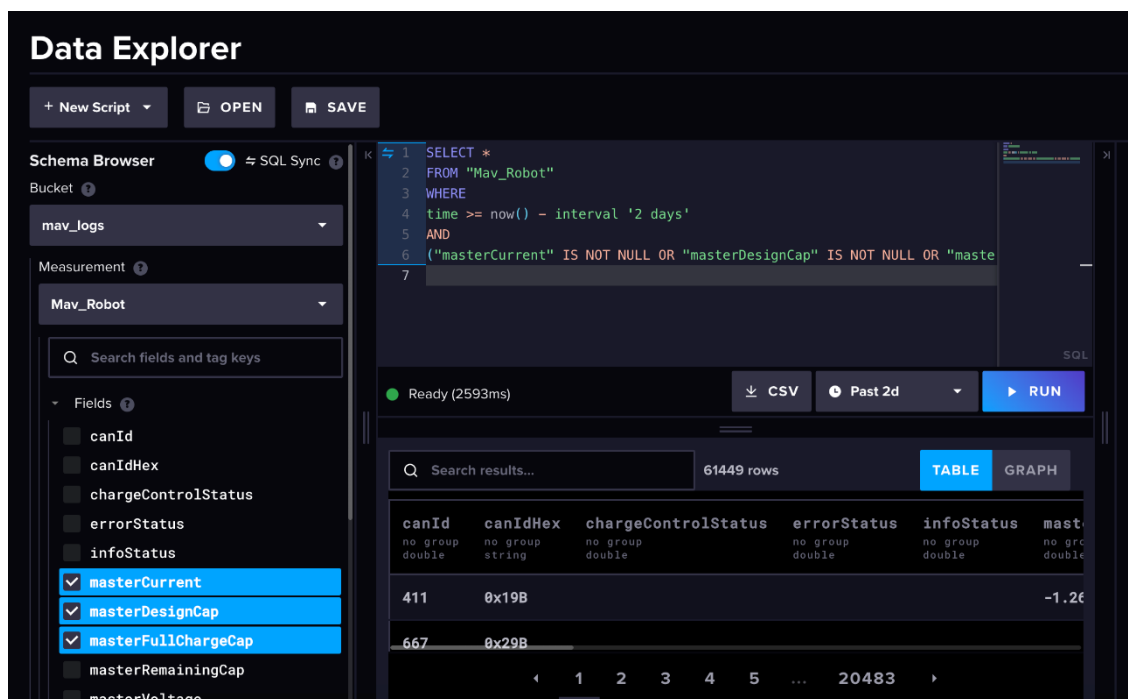


FIGURE 6: GENERATING SQL QUERIES USING DATA EXPLORER

We are using Data Explorer to leverage data exploration capabilities. This tool allows us to execute SQL commands and access the data seamlessly. Additionally, it enables the efficient generation of tables for better data management and analysis. API Tokens are created by selecting the data Bucket and if only reading, writing or both is allowed.

Ready (2593ms)

CSV

Past 2d

RUN

Q Search results...

61449 rows

TABLE

GRAPH

masterVoltage	maxCellTemp	maxFETTemp	nodeId	pdo	time
no group double	no group double	no group double	no group double	no group string	no group dateTime:RFC3339
54.25			27	TPD0_5	2024-07-15T09
	28.3	28.6	27	TPD0_6	2024-07-15T09
			27	TPD0_7	2024-07-15T09
			27	TPD0_7	2024-07-15T09
	28.3	28.7	27	TPD0_6	2024-07-15T09
54.247			27	TPD0_5	2024-07-15T09
54.247			27	TPD0_5	2024-07-15T09

1

2

3

4

5

...

8779

FIGURE 7: MAV EXAMPLE DATA: MASTERVOLTAGE, MAXCELLTEMPERATURE, NODEID, PDO, TIME



Generate a Custom API Token

×

Description

Describe this new token

Q Filter Access Permissions...

Resources	Read	Write
<div> <div>▼ Buckets</div> <div> <div>All Buckets</div> <div></div> <div></div> </div> <div>Individual Bucket Names</div> <div> <div>mav_logs</div> <div></div> <div></div> </div> <div>NodeRed</div> <div></div> <div></div> <div>_monitoring</div> <div></div> <div></div> <div>_tasks</div> <div></div> <div></div> </div>		
<div> <div>▼ Telegrafs</div> </div>		
<div> <div>▼ Other Resources</div> </div>		

FIGURE 8: GENERATING API-TOKENS

#### 2.1.6. GRAFANA

Grafana is a powerful, open-source analytics and visualization tool, ideal for time-series data. It supports a variety of sources, such as by directly connecting to databases like InfluxDB, or receiving data via communication protocols like MQTT, enabling the seamless consolidation and management of data from different systems in a single dashboard.

#### Key Features

Advanced Visualization Options	Grafana offers a wide range of visualization tools such as graphs, bar charts, and heatmaps, allowing for quick and efficient analysis and presentation of complex datasets.
Real-Time Monitoring and Alerting	Grafana enables real-time monitoring of systems and can automatically send notifications when data exceeds predefined thresholds. This is crucial for proactive maintenance and quick response to potential issues.
Scalability and Performance	Grafana is designed to grow with the project's demands, capable of handling large datasets and offering fast data processing and visualisation for real-time analyses.

**TABLE 3: GRAFANA FEATURES**

### **Use Cases**

Grafana is frequently used for network monitoring, server and application monitoring, IoT and sensor data monitoring, as well as business intelligence and analytics. It allows companies to effectively respond to changing conditions and make informed decisions based on real-time data.

### **Implementation and Security**

For projects requiring a cloud-based solution, Grafana can be deployed on platforms like Amazon Web Services via Docker containers. This simplifies management, enhances accessibility and scalability, and reduces maintenance effort. Grafana's security features, including support for various authentication mechanisms and detailed access controls, ensure the protection of sensitive data.

### **Benefits for Our Work**

Grafana offers a combination of powerful visualizations, easy integration, automation, real-time monitoring, and user-friendliness – making it an ideal choice for our project. The ability to clearly and interactively present complex data supports not only daily monitoring and management but also long-term strategic planning and optimization.

### **Use Case**

Grafana can be used either locally or on Grafana Cloud. However, using Grafana Cloud introduces additional limitations, such as increased cloud and storage costs that must be considered. In our case, we used Grafana Cloud to access InfluxDB Cloud via API-Token.



FIGURE 8: CURRENT GRAFANA DASHBOARD

#### DASHBOARD URL:

<https://batterydashboard.grafana.net/d/EDL3X4XNY86BKA/dashboard-inmediasp-new?from=2024-06-16T18:11:44.628Z&to=2024-07-16T18:11:44.628Z&timezone=browser&refresh=>

The latest iteration of the dashboard is accessible via URL and is publicly available to everyone, however the database it is accessing is only holding data for up to 30 days. This setup allows for integration into the Submodel of the AAS Shell Explorer in subsequent steps. However, for industrial use cases, it is crucial to consider security and implement regulated access to ensure data protection and compliance.

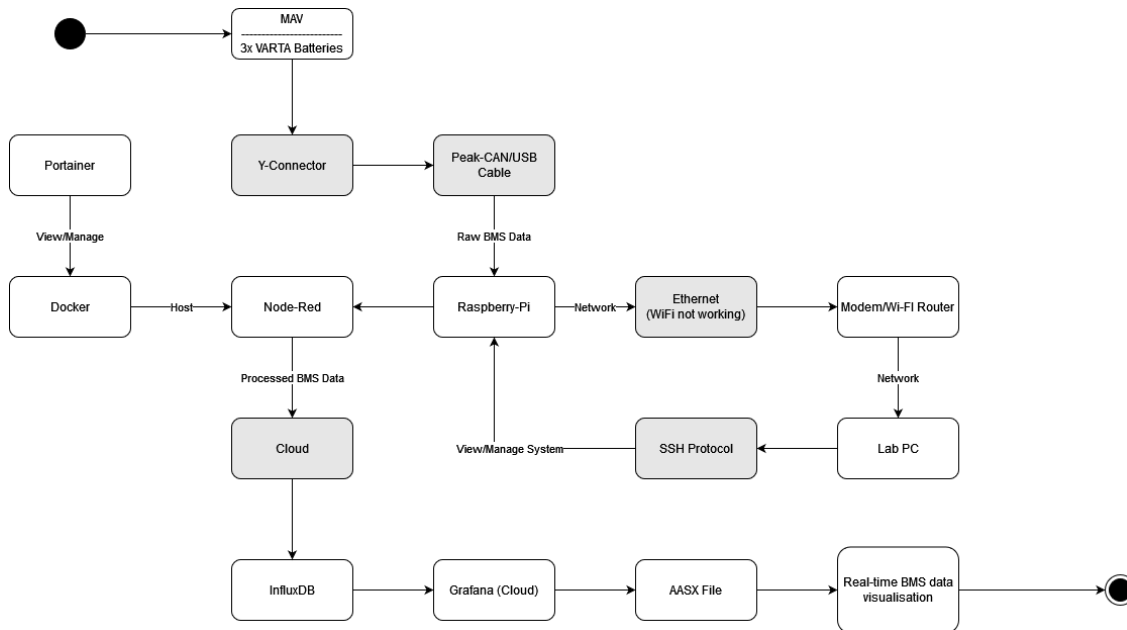
Grafana does have the capability to view fiscal year, quarterly, and yearly data. Users can change historical data and adjust the time range directly on the Grafana page. The refresh time is set to 5 seconds to enable real-time implementation, but users can also modify the refresh interval as needed.

Grafana is primarily a data reading and analysing tool that allows users to present different types of graphs and time-series data representations. For our project, it was required to represent voltage,

current (ampere), temperature, capacity (SoC), state of health (SoH), and include a warning status. The relevant SQL commands are documented in **Appendix 6.3**.

## 2.2. PROJECT WORKFLOW

**Figure 6** below illustrates the workflow used in the final solution setup.



**FIGURE 6: PROJECT WORKFLOW**

## 2.3. ECLIPSE BASYX

Eclipse BaSyx is a middleware framework designed to simplify the use of Asset Administration Shells (AAS) for creating digital twins. This document outlines how BaSyx interacts with AAS to manage digital representations of physical assets.

BaSyx leverages AAS as the cornerstone technology for building digital twins. An AAS serves as a digital identity for a physical asset. It incorporates links to sub models that provide specific details, simulations, and services relevant to the asset.

## 2.4. AAS SERVER

The Eclipse AASX Server is a companion application to the Eclipse AASX Package Explorer. It functions as a local service for hosting and serving Industrie 4.0 Asset Administration Shell (AAS) packages in the AASX format.

The AASX Server focuses on providing functionalities related to AASX package management:

- **Hosting:** The server acts as a central repository for AASX packages, allowing access and interaction with the information they contain.
- **Serving:** It facilitates retrieval of AASX packages by clients through various protocols, including:
  - REST API
  - OPC UA
  - MQTT (machine-to-machine communication protocol)

## Versions

The Eclipse AASX server comes in three main variants, catering to different user needs:

1. **Core:** This is a server built on .NET Core 3.1, offering the core functionalities like API access through the mentioned protocols.
2. **GUI:** This variant extends the core functionalities with a graphical user interface (GUI) built using the Blazor framework. The GUI allows users to explore the hosted AASX packages directly within a web browser.
3. **Windows:** This variant leverages the .NET Framework 4.7 and is designed specifically for Windows environments. It provides the same functionalities as the core variant.

### 2.4.1. FUNCTIONALITY IN OUR CONTEXT

For our specific needs, the key functionalities of the AASX Server (GUI) were:

- **Centralised Hosting:** The server acted as a central repository for storing and managing our AASX packages. This ensured easy access and control over these packages.

- **Interactive Exploration (GUI):** The web-based GUI provided a user-friendly interface for us to explore the contents of the hosted AASX packages. This visual representation of data streamlined our ability to understand and interact with the information within the packages.
- **Potential for Further Exploration:** While we primarily focused on data exploration, the server's capabilities extend to serving AASX packages through various protocols (REST API, OPC UA, MQTT). This opens doors for potential future integrations and functionalities.

#### 2.4.2. BENEFITS FOR OUR WORK

Using the Eclipse AASX Server (GUI) offered several advantages for our work:

- **Simplified Management:** The centralised repository simplified the management of our AASX packages, reducing the need for scattered storage and access methods.
- **Enhanced Understanding:** The interactive GUI facilitated a more intuitive understanding of the data within the AASX packages, compared to working with raw data formats.
- **Foundation for Future Development:** The server's underlying functionalities provide a foundation for building upon our work. The potential integrations with communication protocols could be explored for further advancements.

#### 2.4.3. SERVER DEPLOYMENT

For wider accessibility and to eliminate system dependencies, we deployed the Eclipse AASX Server (GUI) on Microsoft Azure. Azure is a cloud computing platform that provides a range of services, including virtual machines for hosting applications. However, to achieve a more efficient and scalable deployment, we utilised Docker containers.

**Docker Deployment:** We created a Dockerfile that packaged the AASX Server (GUI) application and its dependencies into a containerized image. This image can be easily deployed and run on any system that supports Docker.

**Deployment on Azure Web App Service:** Leveraging Microsoft Azure's Web App Service, we deployed the Docker container image. The Web App Service provides a managed environment for running containerized applications, taking care of infrastructure management and scaling.

This approach offered several advantages:

- Accessibility: The server became accessible to anyone with an internet connection and a web browser. This eliminated the need for users to have the server software installed on their local machines.
- Scalability: Azure Web App Service allows for automatic scaling based on user traffic, ensuring optimal performance.
- Reduced Maintenance: By using Docker containers and Azure's managed services, we minimised the need for manual server management and maintenance.
- Platform Independence: Docker ensures the application runs consistently regardless of the underlying Azure infrastructure.

#### 2.4.4. HOW TO DEPLOY THE SERVER LOCALLY WITH DOCKER

This guide walks you through setting up an AASX Server using Docker. This lets you run the adminshellio/aasx-server-blazor-for-demo application and manage your AAS files all in one place.

##### **Before You Begin**

- Docker Up Set-up: Make sure you have Docker installed on your machine. Download and install it from the official website: <https://docs.docker.com/get-docker/>
- Using docker-compose.yml
  - The recommended way to set up the AASX Server is with a file called docker-compose.yml. This file defines how the server will run in your Docker environment. Here's a breakdown of what's inside:
- The aasx-server Service
  - This section defines a service named aasx-server that will run the actual AASX Server container. Think of a container as a self-contained package with everything the server needs to operate.

## YAML Configuration options

Below is an explanation of each configuration option which is in the YAML file.

### Configuration Options

- **Container Name:** This sets a friendly name for the container, `aasx-server`, for easier identification.
- **Docker Image:** This specifies the exact image to use, which is [docker.io/adminshellio/aasx-server-blazor-for-demo:main](https://hub.docker.io/r/adminshellio/aasx-server-blazor-for-demo).
- **Automatic Restart:** This ensures the container restarts automatically if your system reboots (unless it crashes).
- **Port Mapping:** This connects your computer's port 8080 to the container's port 5001. This allows you to access the AASX Server in your web browser at `http://localhost:8080`. You can use whatever open ports you have.
- **Environment Variables:** These are special settings for the container:
  - **Kestrel\_\_Endpoints\_\_Http\_\_Url:** This tells the web server inside the container to listen on port 5001.
  - **Local File Access:** This connects the directory on your computer where the .aasx files are located (`./AASfiles` in the example) to a directory inside the container (`/AasxServerBlazor/aasxs/`). This lets you store your AAS files on your machine and access them from the server.
- **Running the Server:** This overrides the default command with specific options:
  - **--no-security**
    - Caution! This disables security measures. Only use this for development, not production environments!
  - **--data-path /usr/share/aasxs**
    - This sets the location for AAS files within the container.
  - **--external-blazor YOURURL**
  - **Optional:** This sets the URL for an external Blazor application.



## Getting Started

1. **Create the docker-compose.yml file:** In your project directory, create a new file named docker-compose.yml and paste the provided configuration (see appendix).
2. **Store your AAS files:** Make sure your AAS files are located in the ./AASfiles directory on your computer or the relevant directory you named in the YAML file.
3. **Build and Run:** Open your terminal and run these commands:
  - `docker-compose build`
  - `docker-compose up -d`
4. This builds the Docker image and starts the AASX Server in the background. You can now access it at `http://localhost:8080` in your web browser.

Note: If you want to use a new release of the image, add the `--pull` flag to get the latest version of the build.

### File Used

#### Docker-compose.yml

services:

aasx-server:

container\_name: aasx-server

image: docker.io/adminshellio/aasx-server-blazor-for-demo:main

restart: unless-stopped

ports:

- 8080:5001

environment:

- Kestrel\_\_Endpoints\_\_Http\_\_Url=http://\*:5001

volumes:

## Replace ./AASfiles with the location of your .aasx files

- ./AASfiles:/AasxServerBlazor/aasxs/

command: --no-security --data-path /usr/share/aasxs --external-blazor YOURURL

## How to deploy the server on cloud

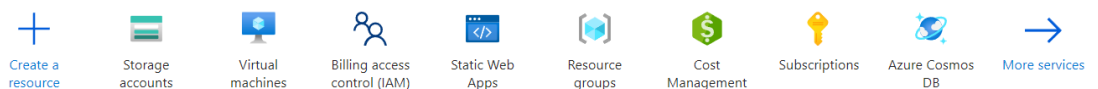
Once you have set up your AAS Server and successfully have it running locally, you are able to deploy to the cloud to allow for access anywhere with an internet browser and internet connection. This is useful for actual use beyond demonstration purposes. For our deployment of the AASX server Microsoft Azure was chosen as the cloud provider. This was due to past experience using Azure and knowledge of the tools it provides. Note you can use any cloud provider you are comfortable with or already have data with as they all offer web-app deployment.

## Required

- Azure account
- Docker compose file
  - Dockerhub image we used (adminshellio/aasx-server-blazor-for-demo:main)
  - You will need to configure this to and make sure it can run locally as you intend before hosting on cloud.

Once you have the requirements complete the following steps:

1. Click create a new resource



2. Choose or search for 'Web App' and click create.

Popular Azure services [See more in All services](#)



### Virtual machine

[Create](#) | [Docs](#) | [MS Learn](#)



### Web App

[Create](#) | [Docs](#) | [MS Learn](#)

3. Give your web app a name and select what resource group you would like it to be a part of.
4. You then have the option of deployment via code, container or static web app. For our purposes we are doing container deployment as it is the most seamless.

Publish \*

☒ Code ☐ Container ☐ Static Web App

Runtime stack \*

Select a runtime stack ▼

Operating System

☒ Linux ☐ Windows

Region \*

East US ▼

Not finding your App Service Plan? Try a different region or select your App Service Environment.

5. Select Linux as your operating system and select the most appropriate region.

- Next choose your image source, for us we used docker hub as the source and doing docker compose as the option, this lets us deploy a web-app based off a docker compose file.

### Create Web App ...

The screenshot shows the 'Create Web App' wizard in the Azure portal, specifically the 'Container' tab. The wizard is divided into several sections with radio button options:

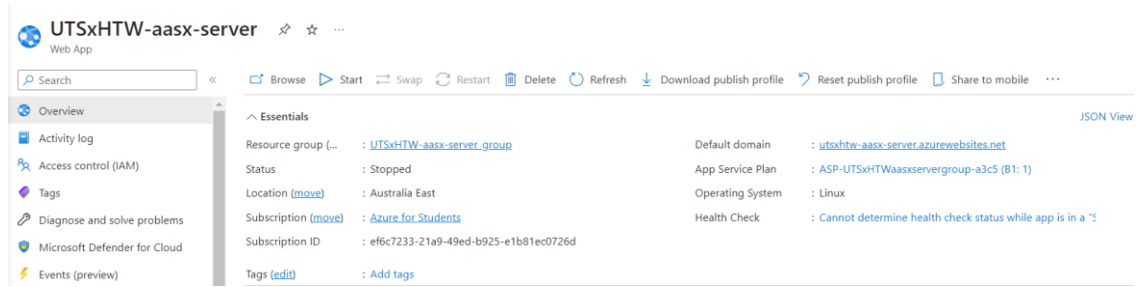
- Basics** (selected), Networking, Monitoring, Tags, Review + create
- Select your preferred source for container images.** You can change these settings and other dependencies after creating the app. [Learn more](#)
- Sidcar support (preview)**
  - ☐ Enabled
  - ☒ Disabled
- Image Source \***
  - ☐ Quickstart
  - ☐ Azure Container Registry
  - ☒ Docker Hub or other registries
- Options**
  - ☐ Single Container
  - ☒ Docker Compose (Preview)
- Docker hub options**
  - Access Type \***
    - ☒ Public
    - ☐ Private
- Configuration File**
- Configuration**

- Once the docker compose file is uploaded you can create and access your server on the cloud. For our example we used the adminshellio/aasx-server-blazor-for-demo:main image which can be found on dockerhub. You will need to configure the docker-compose file based on your set up preferences.

### How to run the server

Once you have access to the server follow the steps below to run it.

- Log into <https://portal.azure.com/>
- Search for the name of the app service (example: UTSxHTW-aasx-server)
- Click start in the header bar for the overview page.



4. You can then click the domain link to access the server, this domain link is what you provide into the ASX explorer when connecting to the server. Note: It may take a few minutes to start the web-app.
5. When done using please stop the web-app as it charges by the hour.

## 2.5. AASX PACKAGE EXPLORER V3

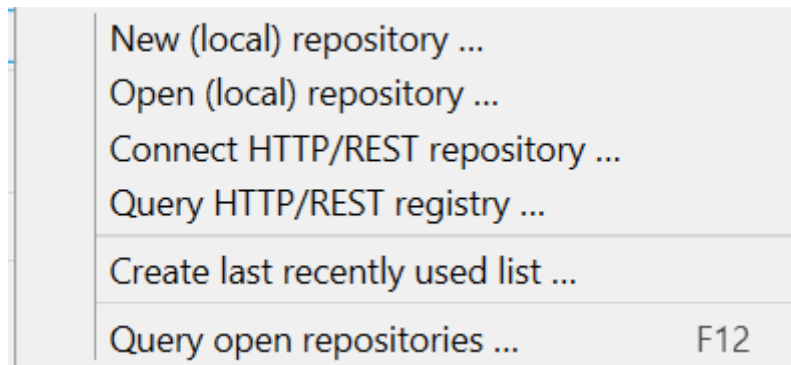
This document describes a Windows-based application designed for viewing, editing, and managing .AASX files. The application also provides functionality to connect to a dedicated AASX Server, enabling real-time synchronization between the desktop application and a web-based application.

### 2.5.1. KEY FEATURES

- **.AASX File Support:** View and edit the content of .AASX files.
- **Editing Functionality:** Modify the contents of .AASX files as needed within the application.
- **AASX Server Connection:** Establish a connection with a designated AASX Server.
- **File Uploading:** Upload edited .AASX files to the connected AASX Server.
- **Real-Time Synchronization:** Changes made to .AASX files within the application are reflected in real-time on the server's web application.

### 2.5.2. BENEFITS

- **Centralised Management:** Manage .AASX files from a user-friendly desktop application.
- **Collaborative Editing:** Leverage the server connection for real-time collaboration on .AASX files through the web application.
- **Improved Workflow:** Streamline workflows by editing and uploading .AASX files directly within the application.

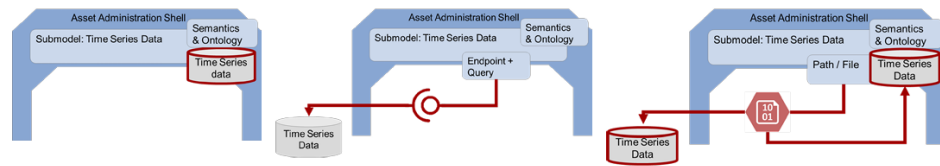


To connect click Connect HTTP/REST repository and enter the URL of the server.

## 2.6. CONNECTING SENSORS TO AASX

The Industrial Digital Twin Association (IDTA) is the central point of contact for all groups interested in leveraging digital twin technologies. They provide the necessary specifications for the AAS and maintain the compliance and schemas, so all AAS are consistent and interoperable. Below are their recommendations for how to implement time series data into the AAS.

However, it is to note that the IDTA are still exploring the functionality of time series within the AAS and have found that, “The defined sub-model offers more flexibility due to the different modelling approaches for time series segments, but this can lead to limitations in interoperability. I4.0 applications that access time series data may have to deal with proprietary systems and data formats .”(IDTA, 2023). Based on the support for time series data there are currently three recommended ways of accessing and storing this data. As seen below.



	InternalSegment	LinkedSegment	ExternalSegment
<b>Description</b>	InternalSegments allow an I4.0 application (defined in [7]) to manage the Time Series Data structure and content directly in the AAS.	LinkedSegments allow an I4.0 application to read out the endpoint and query to an external system to manage time series without the AAS.	ExternalSegments allow an I4.0 application to find a data or BLOB file within which time series data is stored.
<b>Suitable for</b>	<ul style="list-style-type: none"> <li>- Few data points</li> <li>- handover of data points</li> <li>- Permissions managed within the AAS</li> </ul>	<ul style="list-style-type: none"> <li>- Brownfield integration</li> <li>- Mass data</li> <li>- Permissions managed in external system</li> <li>- dynamic time series without continuous update of AAS</li> </ul>	<ul style="list-style-type: none"> <li>- Brownfield integration</li> <li>- Static time Series</li> <li>- Handover of time Series</li> <li>- Few accesses</li> </ul>
<b>Implications</b>	<ul style="list-style-type: none"> <li>- dynamic time series require continuous updating of the AAS</li> </ul>	<ul style="list-style-type: none"> <li>- I4.0 application must be able to work with endpoint and query</li> <li>- Data semantics and payload separated from each other</li> </ul>	<ul style="list-style-type: none"> <li>- I4.0 application must be able to handle different file formats</li> <li>- Data semantics and payload separated from each other</li> </ul>

**FIGURE 7: TYPES OF TIME SERIES SEGMENTS**

## Main Issues

- For data to be used in a AASX asset it needs to adhere to strict formatting, as the data coming from the BMS needs to be pre-processed to become information, it is not feasible to directly connect the the AASX via MQTT protocol. Since this would mean storing HEX data that is not useful.
- Due to the intricacies of the AASX file structure, it would be difficult to directly create the file from BMS data. Since information such as serial numbers, manufacturing data is not stored in the BMS.

## Solutions

- Perform data processing in Node-RED similar to what is done before it is stored in the InfluxDB.
- Find a JSON structure that is readable by the AASX package explorer.
- Push data from the BMS to a corresponding submodel using API calls.
- Can connect sensors in Node-RED and output as a JSON in the format that is readable by the AASX viewer.

- If this works, then we would be able to use API calls to put changes to an asset by referencing the asset ID.
- We would have a sub-model called logs, then we could store logs in the AASX.

### **Potential Workflow**

1. BMS data arrives: Raw data from the BMS is received.
2. Node-RED processing: The data is processed in Node-RED to extract relevant information, perform calculations, and potentially enrich it with additional data sources (if needed).
3. Convert to JSON: The processed data is converted into a well-defined JSON structure compatible with the AASX schema.
4. Push data to AASX: Node-RED uses API calls to push the JSON data to the corresponding sub-models within the AASX server.

## **2.7. SECURITY**

### **2.7.1. CURRENT STATE OF AAS SECURITY**

The security standards of the AAS have been set by IDTA and describe various security scenarios based on different use cases. In the projects implementation of AAS we have our server application which can load files and provides information to the repository via REST, it is evident that there are many different communication points in using the AAS and this means many points to secure. For example sharing AAS files between businesses the provider can share a copy of the file with confidential parts excluded or with them locked for access at a later date, this requires a form of access control. The preferred access control method for cross-company exchanges like the above as established by IDTA is Attribute Based Access Control (Plattform Industrie 4.0. Security der Verwaltungsschale: Diskussionspapier, 2017.) Attribute-Based Access Control (ABAC) is a method for granting access based on attributes of the subject, object, and environment. This allows for more fine-grained access control than traditional methods. In ABAC, rules are formulated that specify the conditions under which access is granted. These rules are based on the attributes of the subject, object, and environment.



ABAC can be used to control access to a wide variety of objects in Industrie 4.0. For example, an ABAC system could be used to control access to a machine by service technicians. The system could grant access to technicians with the appropriate qualifications, during certain times, and only if the machine is in maintenance mode.

ABAC offers several advantages over traditional access control methods. First, ABAC is more dynamic. The values of attributes can change over time, and the access control system can take these changes into account. Second, ABAC is more scalable. New subjects and objects can be added to the system without the need to change the access control rules. Third, ABAC is more secure. ABAC can be used to implement complex access control policies that can help to protect against unauthorized access.

These permission rules are stored within the Asset Administration Shells (AASs) themselves and can be provided by the asset manufacturer. The receiver of the AAS also has the ability to set their own access permission rules. These rules are based on the subject attributes, object attributes, and environment attributes, with the access permission rules evaluated locally to decide if the access is granted [SM1].

A prerequisite for ABAC in this context is a system with unique identities and secure identity management to identify the subjects that want to access the AAS and the objects, such as the whole AAS or submodels, that should be accessed.

### 3.3 BMS DASHBOARD INSIDE AASX

For our project, we integrated the real-time dashboard into a submodel of our Varta AASX file, which represents our Varta battery's digital twin. The dashboard URL is embedded using a text/URL subformat within the AAS shell. This integration allows users to seamlessly access and interact with the dashboard by clicking "Content" in the right corner. From there, users can view real-time data, monitor system performance, and make necessary adjustments, enhancing the overall functionality and usability of the digital twin representation.

This setup provides insights into the battery's operational metrics and facilitates proactive maintenance and decision-making. By allowing easy access to the dashboard, users can quickly identify and address

potential issues, ensuring optimal performance. The interface ensures that users can navigate the dashboard with ease, making it possible to customize views, set alerts, and analyse trends over time.

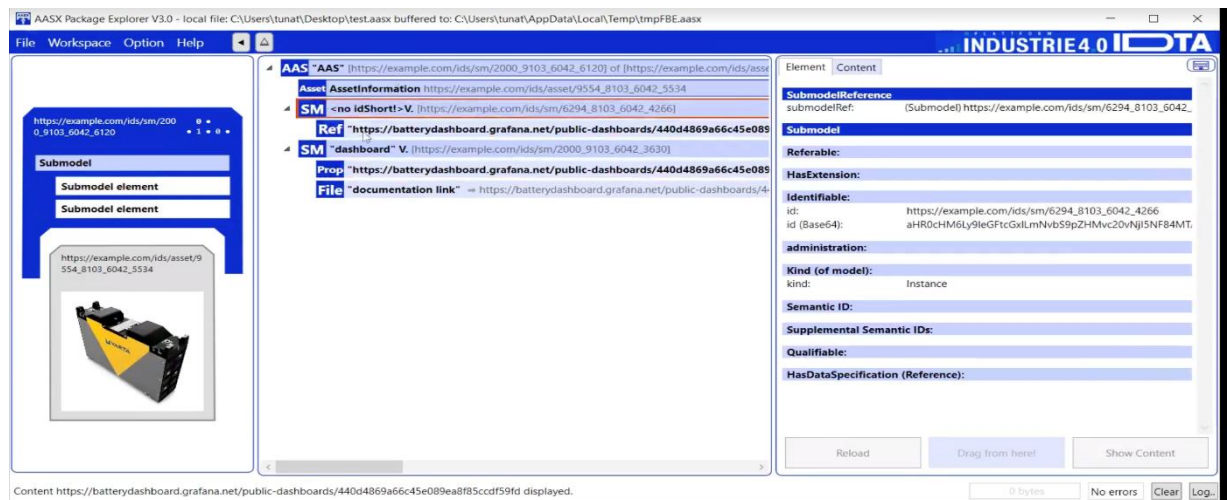


FIGURE 9: REAL-TIME DASHBOARD WITHIN THE AASX-FILE

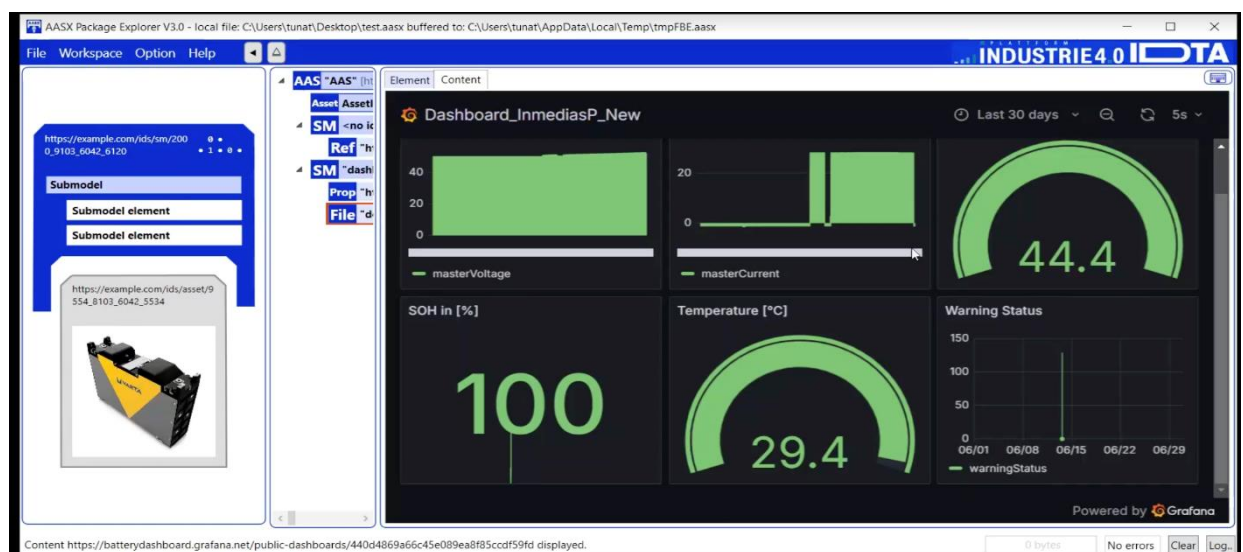


FIGURE 10: REAL-TIME DASHBOARD WITHIN THE AASX-FILE

### 3. DISCUSSION

---

#### 3.1. CHALLENGES

##### 3.1.1. NAVIGATING INDUSTRY 4.0

Of the main challenges faced during the early stages of the project was discovering the literature and notable companies providing materials on using Digital Twin technologies. Since this use of Industry 4.0 technologies has been adopted mostly in Europe a lot of the source material was in German and was more difficult to navigate compared to finding information on technologies used more prominently in Australia. Once the relevant industry leaders were discovered the process of learning was made easier. Communication with InMediasP proved very useful in overcoming this challenge as they pointed us in the right direction recommending companies such as BaSyx who are the main leader in open-source Digital Twin technologies.

##### 3.1.2. CONFIGURATION OF COMPONENTS

Finding the relevant tutorials to set-up and configure the necessary components needed for our project proved difficult for a number of reasons. The digital twin technologies framework has only been around since 2017 (Later, Grieves and Vickers add reference) so the amount of tutorials are limited and not thorough. Comparing this with the tutorials available for more widely used technologies like Docker, Azure, AWS there is not as well documented solutions. Another limitation of configuring the components is the support for multiple operating systems, currently Windows is the only supported OS for majority of the AAS software. Although it was more difficult to set up the components it was more rewarding in understanding the systems and how all the Digital Twin components work together as it was necessary to do deep dives into all documentation and information on forums. Lots of trial and error was used in configuring settings to get the components to work to the specifications required for the project since. This was especially prevalent in the hosting of the applications as most of the examples were hosted locally whereas for our project it was necessary to have a cloud hosted solution.

### 3.1.3. WORKING WITH OPEN SOURCE

The AASX Package Explorer and Server that was used in the project was part of an open-source effort supported by the Eclipse Foundation. While it was great to work with the open-source community and support these projects there were some drawbacks associated with relying on open source. The main issue found during the setup stage of the project was the outdated documentation. We found that the documentation provided on the GitHub page hadn't been updated to match recent developments. This was especially tricky when using the API calls since they weren't consistent with the documents. Also during the later stages of the project, a breaking change was pushed to the server which broke the functionality between the package explorer and the server. With this the ability to manipulate AASX files locally and push the changes to the server was lost.

## 4. GLOSSARY

---

Term	Definition
.NET	A software development framework from Microsoft for building various applications (web, mobile, desktop etc.)
AASX	Package file format used for the Asset Administration Shell (AAS) standard, which describes industrial assets.
Amazon Web Services (AWS)	A cloud computing platform offering a variety of services like storage, compute power, databases, etc.
Application Programming Interface (API)	A set of instructions and standards that allows applications to communicate with each other.
Asset Administration Shell (AAS)	An information model for representing industrial assets in a standardized way.
BaSyx	An open-source framework for implementing the Asset Administration Shell (AAS) standard.
Battery Management System (BMS)	An electronic system that manages a battery, including monitoring its state, charging and discharging.
Blazor	A web framework from Microsoft that allows building web UI with C# and .NET.
Bus	A communication system that allows devices to share data.
Command Line Interface (CLI)	A way to interact with a computer program using text commands.
Controller Area Network (CAN)	A robust communication protocol for devices in industrial automation.
Data Bucket	A storage location for large amounts of data in cloud storage.
Digital Twin (DT)	A virtual representation of a physical asset that mirrors its behavior and state.
Dockerfile	A text document that defines the instructions for building a Docker image.
Dockerhub	A cloud repository for sharing Docker images.

Eclipse	An open-source integrated development environment (IDE) popular for software development.
Flow	In Node-RED, a visual representation of a program logic using connected nodes.
Grafana	An open-source platform for data visualization and analytics.
Graphical User Interface (GUI)	A user interface that allows interaction with a computer program using visual elements like windows, icons, menus etc.
HTW Berlin	University of Applied Sciences Berlin (Hochschule für Technik und Wirtschaft Berlin).
Industrial Digital Twin Association (IDTA)	A consortium promoting the use of digital twins in industry.
Industrie 4.0 (I4.0)	A German initiative focused on advancing digitalization of manufacturing.
InfluxDB	An open-source time series database for storing and analyzing time-stamped data.
Information Technology (IT)	The technology used for storing, retrieving, transmitting, manipulating and protecting information.
IoT	Internet of Things - Network of physical devices embedded with sensors, software, and other technologies for connecting and exchanging data.
JavaScript	A high-level programming language commonly used for web development.
JSON	JavaScript Object Notation - A lightweight data format for human-readable exchange of information.
Machine-to-Machine (M2M)	Direct communication between devices without human intervention.
Microsoft Azure	A cloud computing platform offering similar services to AWS.
Mosquitto	An open-source MQTT message broker.
MQTT	Message Queuing Telemetry Transport - A lightweight messaging protocol for communication between devices.
Multi-Sensing Autonomous Vehicle (MAV)	An unmanned aerial vehicle equipped with various sensors.

MySQL	A popular open-source relational database management system (RDBMS).
Node	Refers to runtime environment for executing JavaScript code outside a web browser.
Node-RED	A visual programming tool for wiring together hardware devices, APIs and online services.
OPC	Open Platform Communications - A family of interoperability standards for communication in industrial automation.
Process Data Object (PDO)	A standardized data structure used in communication protocols like CAN bus.
Raspberry Pi	A credit-card sized computer commonly used for hobbyist projects.
REST	Representational State Transfer - An architectural style for designing web services.
State of Charge (SOC)	Remaining battery capacity relative to its full capacity.
State of Health (SOH)	A measure of a battery's overall health and capacity degradation.
Submodel	A part of an Asset Administration Shell (AAS) that describes a specific aspect of an asset.
Tokens	A piece of data used for authentication or authorization.
UA	Unified Automation - A communication protocol for industrial automation built on top of OPC.
UML	Unified Modeling Language - A standard graphical language for visualizing software systems.
University of Technology Sydney (UTS)	A university in Sydney, Australia.
VARTA	A company specializing in battery technologies.
YAML	Yet another markup language

## 5. APPENDIX

---

### 5.1. CAN-BUS, CANOPEN, AND NODE ID

A Controller Area Network (CAN) Bus is a message-based protocol used to allow communication between different components in a system, for example the electrical components in a vehicle.

It uses a “Shared Bus Topology”, and hence has no “Master” device. The devices/components/microcontrollers/nodes in the system share the network.

It allows the configuration of priority amongst the nodes. Priority is dictated by the “Node ID”, which is a number used to identify a node in the network (battery), the lower the number, the higher the priority.

CANopen is a higher-level protocol over the CAN system. It is beneficial because it unburdens systems developers from dealing with the CAN hardware-specific details. It standardised the communication into basic message units at the application level known as Communication Objects (COB). Each COB has a unique identifier to determine its purpose and priority in the network.

[https://en.wikipedia.org/wiki/CAN\\_bus](https://en.wikipedia.org/wiki/CAN_bus)

<https://www.can-cia.org/can-knowledge/canopen>

### 5.2. RASPBERRY PI

A Raspberry PI is a small computer run on a single circuit board, developed by the Raspberry Pi Foundation in association with Broadcom. It is an inexpensive option for various computer-related purposes, including industrial applications.

### 5.3. VARTA BATTERY SPECIFICATIONS

Please see attached “VARTA Battery Specifications” file.

### 5.4. VARTA BATTERY – ADDITIONAL INFORMATION

VARTA Batteries must be “configured” by triggering a built-in process whenever the following occurs:



- If the batteries are connected for the first time
- If one or more batteries are replaced with new batteries in an already configured system
- If new batteries are added to an already configured system

This process sets up the “Node ID” for each battery, for identification needs in communications; see **Appendix 1**. The Node ID can be from 1 to 27; they are assigned in order of the serial numbers of the batteries. If the system consists of more than 1 VARTA battery, a “master” battery will be determined; decided by the highest serial number of the batteries. The master battery communicates summarised data using Node ID 27; the master battery can still transmit its single-battery data as its own Node ID.

Batteries that do not get “configured” will not have a valid Node ID, and hence will not be able to send any communications.

Additionally, note that there can never be an active Node ID 2 because of manufacturer settings; see **Figure 8**.

	Device Name	NMT State
0	Network	-
1	b	Operational
2	b	Pre-Op
3	b	Operational
4	b	Operational
5	b	Operational

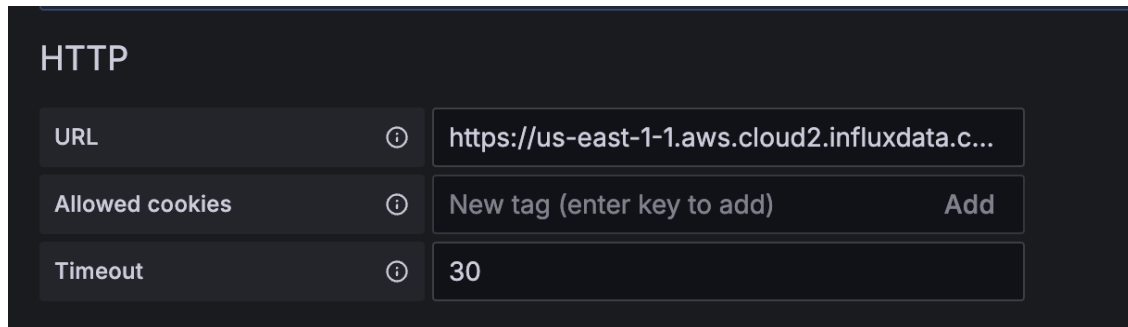
**FIGURE 8 - EXAMPLE OF SYSTEM WITH 4 BATTERIES IN PARALLEL**

## 5.5. PDO

A Process Data Object (PDO) is a Communication Object (COB) within the CANopen protocol. It is used for broadcasting high-priority control and status information about a node.

## 5.6. Influx DB

To extract the correct data, specific SQL commands need to be executed via the Grafana dashboard. Before entering the SQL commands, it is essential to select the correct data bucket from InfluxDB. The Grafana dashboard is connected to InfluxDB using a secured API token to read the data.

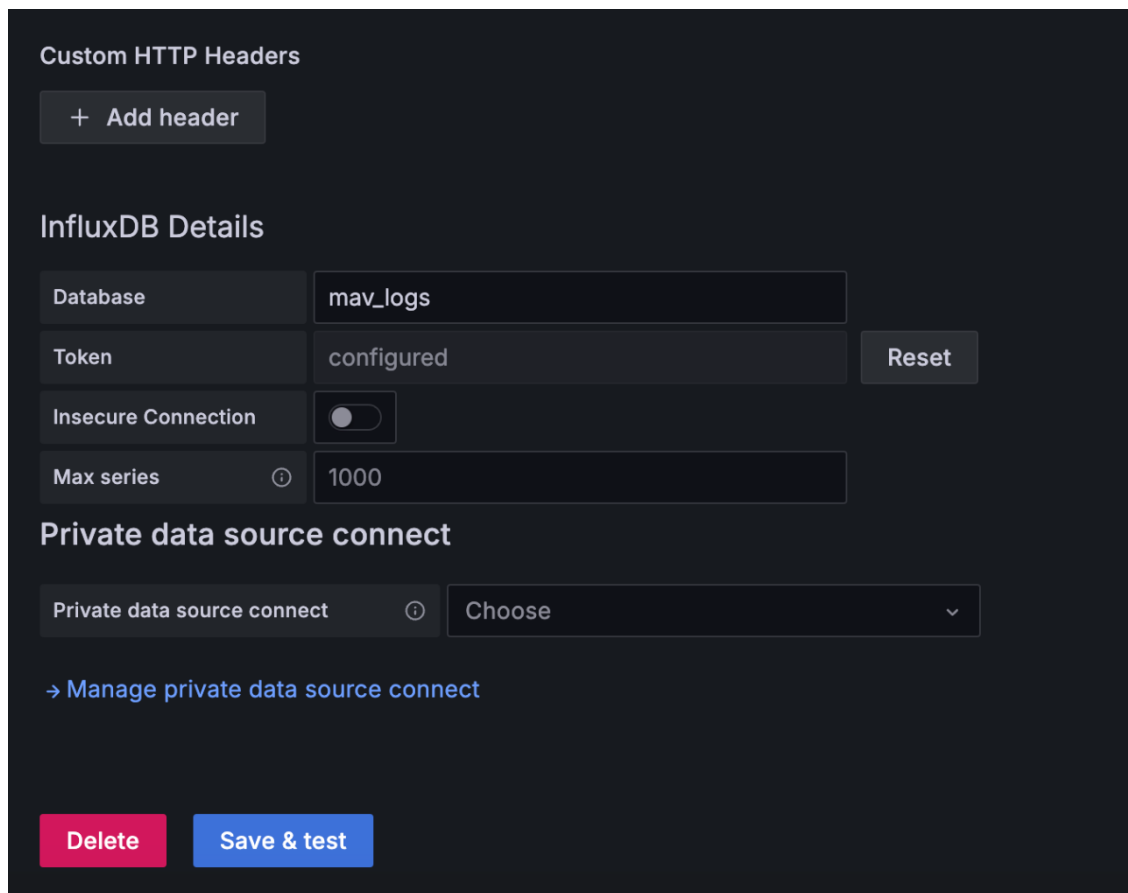


The screenshot shows the 'HTTP' configuration section in Grafana. It contains three rows of settings:

Setting	Value
URL	https://us-east-1-1.aws.cloud2.influxdata.c...
Allowed cookies	New tag (enter key to add) [Add]
Timeout	30

FIGURE 11: INFLUXDB URL ENTERED ON GRAFANA:

[HTTPS://US-EAST-1-1.AWS.CLOUD2.INFLUXDATA.COM/ORGs/8A887B77B4CADEF](https://us-east-1-1.aws.cloud2.influxdata.com/orgs/8A887B77B4CADEF)



The screenshot shows the 'InfluxDB Details' configuration section in Grafana. It includes a 'Custom HTTP Headers' section with an 'Add header' button. The 'InfluxDB Details' section contains the following fields:

Field	Value	Action
Database	mav_logs	
Token	configured	[Reset]
Insecure Connection	<input type="checkbox"/>	
Max series	1000	

Below the 'InfluxDB Details' section is the 'Private data source connect' section, which includes a 'Private data source connect' field with a dropdown menu set to 'Choose' and a link to 'Manage private data source connect'. At the bottom are 'Delete' and 'Save & test' buttons.

FIGURE 11: ENTERING THE API-TOKEN FROM INFLUX DB

## 6.4 GRAFANA

### SQL QUERIES:

Voltage [V]:

```
SELECT
"masterVoltage",
"time"
FROM
"Mav_Robot"
WHERE
"time" >= $__timeFrom
AND "time" <= $__timeTo
```

Ampere [A]:

```
SELECT
"masterCurrent",
"time"
FROM
"Mav_Robot"
WHERE
"time" >= $__timeFrom
AND "time" <= $__timeTo
```

Capacity [%]:

```
SELECT
"masterRemainingCap" / 1000.0 AS "RemainingCapacityInAh"
FROM
"Mav_Robot"
WHERE
"time" >= $__timeFrom AND "time" <= $__timeTo;
```

State of Health [%]:

```
-- Define a base query to fetch the latest non-null values for each parameter at three latest distinct timestamps
```

```

WITH RankedFCC AS (
  SELECT
    "time" AS TimeFCC,
    "masterFullChargeCap",
    ROW_NUMBER() OVER (PARTITION BY "masterFullChargeCap" ORDER BY "time" DESC) AS rk
  FROM
    "Mav_Robot"
  WHERE
    "time" >= $_timeFrom AND
    "time" <= $_timeTo AND
    "masterFullChargeCap" IS NOT NULL
),
RankedDC AS (
  SELECT
    "time" AS TimeDC,
    "masterDesignCap",
    ROW_NUMBER() OVER (PARTITION BY "masterDesignCap" ORDER BY "time" DESC) AS rk
  FROM
    "Mav_Robot"
  WHERE
    "time" >= $_timeFrom AND
    "time" <= $_timeTo AND
    "masterDesignCap" IS NOT NULL
)

-- Calculate SOH by combining the latest values of FCC and DC from the last three timestamps
SELECT
  a.TimeFCC AS "FCC Timestamp",
  b.TimeDC AS "DC Timestamp",
  'State of Health (SOH) %' AS "Description",
  ROUND((a."masterFullChargeCap"::float / b."masterDesignCap"::float) * 100, 2) AS "Value"
FROM
  RankedFCC a
JOIN
  RankedDC b ON a.rk = b.rk
WHERE
  a.rk <= 3 AND b.rk <= 3
ORDER BY
  a.rk, b.rk;

```

Temperature [°C]:

```

SELECT
  "maxFETTemp"
FROM
  "Mav_Robot"
WHERE
  "time" >= $_timeFrom
AND "time" <= $_timeTo

```

Warning Status:

```
SELECT
"warningStatus",
"time"
FROM
"Mav_Robot"
WHERE
"time" >= $_timeFrom
AND "time" <= $_timeTo
```