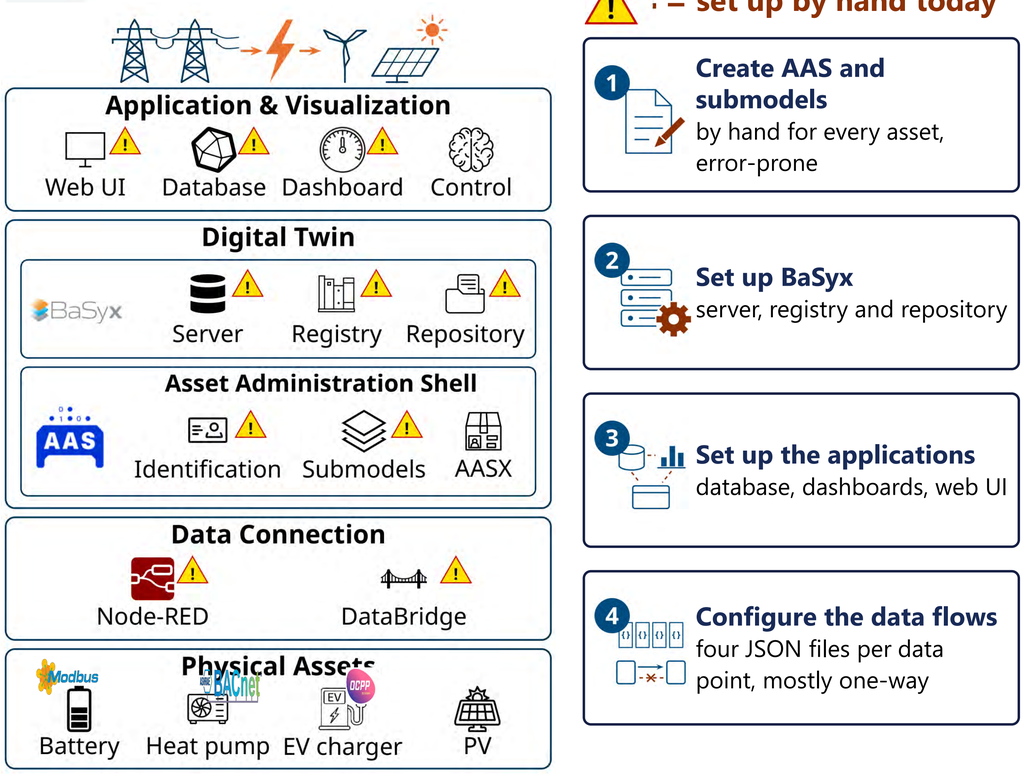
BASYX AUTOTWIN: Automated Configuration of Digital Twins and their Data Connectivity

Mahmoud Mustafa, Georg Frey
(Chair of Automation and Energy Systems (AES), Saarland University)

Every asset of a regional energy system needs a digital twin based on the Asset Administration Shell (AAS). Today, building one is manual work on every layer: shell and submodels, BaSyx infrastructure, database, dashboards and data connection are configured by hand, and data mostly flows one way. **BaSyx AutoTwin automates this chain:** from a single GUI session it generates the AAS with custom and IDTA submodels (AID, AIMC, TimeSeries), an InfluxDB bucket, Grafana dashboards and the DataBridge configuration for reading *and* writing. An example with eight simulated assets and five protocols runs without hand-written configuration.

Motivation & Problem Statement

Regional energy systems have many heterogeneous assets and protocols. A **digital twin per asset** (Asset Administration Shell, Eclipse BaSyx) gives every application one uniform interface. Yet each layer is still **set up by hand**: error-prone and mostly one-way – a digital shadow, not a twin.



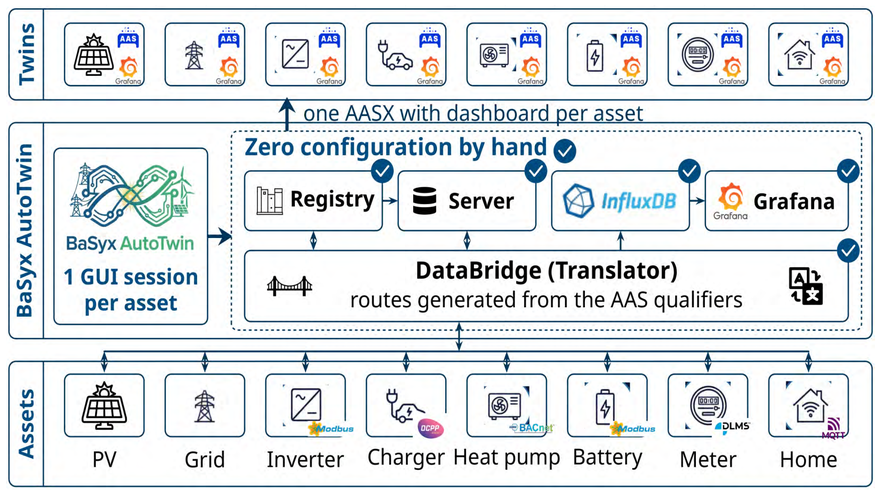
Result today: a digital *shadow*, not a twin – and it does not scale.
Goal: generate the whole chain from one GUI session, with data connectivity **in both directions**.

Results

1	3	2	5	0
GUI session per asset	IDTA submodels	directions: read + write	field protocols	hand-written config files

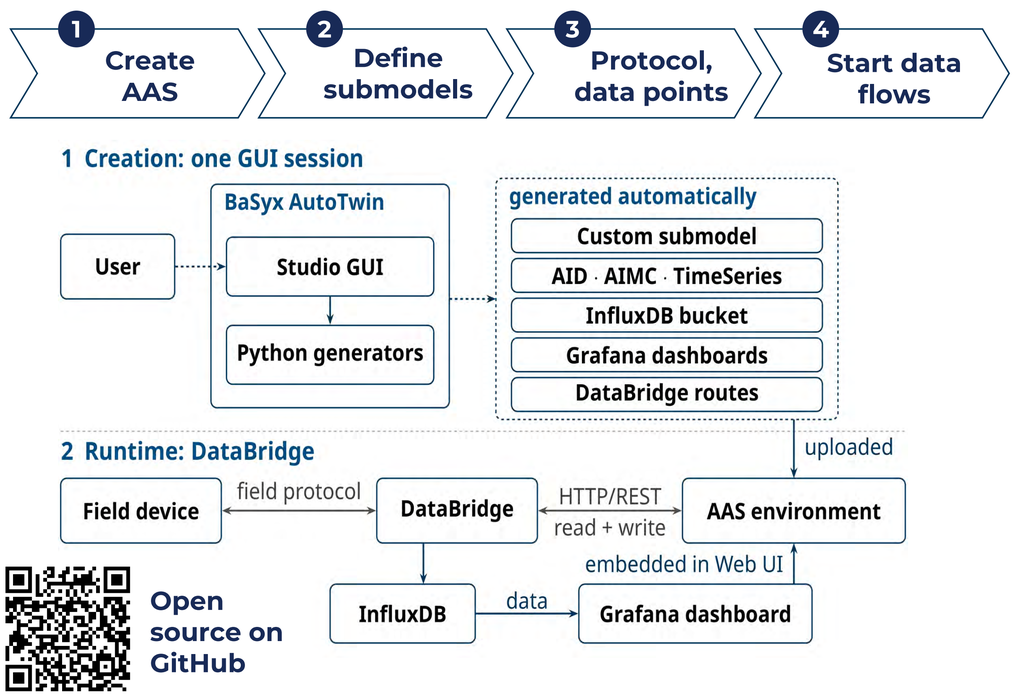
From one GUI session, without expert knowledge: the AAS with custom and IDTA submodels (AID, AIMC, TimeSeries), an InfluxDB bucket, a Grafana dashboard in the Web UI and the data connectivity.
Control works: a setpoint written in the shell reaches the device.
The data connection runs on one of **two DataBridge variants**:

	Custom DataBridge	Eclipse BaSyx DataBridge
Basis	Python	Apache Camel
Direction	read *and* write	read (no write-back yet)
Strength	rapid prototyping	many protocols, production-ready
Configured	automatically from the AAS qualifiers	



Method

Describe each data point once – generate everything else.
Protocol, address, type and direction are stored as **qualifiers with semanticIds**. From them, Python generators build the digital twin on AAS and Eclipse BaSyx: connection to the field device **in both directions**, database, and dashboards embedded in the Web UI.



Limitations & Outlook

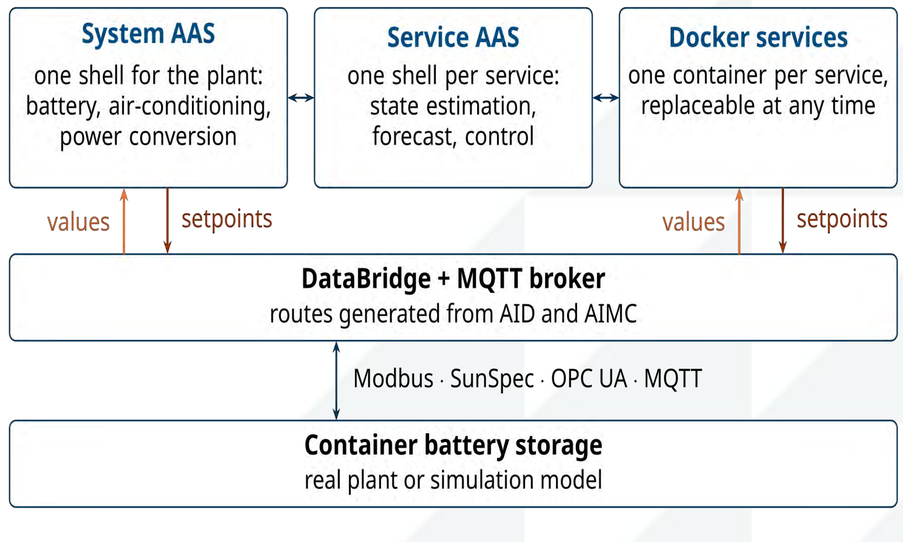
Limitations today

- tool under active development
- field devices simulated
- manual effort reduced, not gone
- simple data types only
- no security yet – lab use
- write side not standardized

Next steps

- **write side in the AID:** actions with semanticIds
- services that find assets by semanticId
- real hardware, TLS, OAuth2
- AutomationML import, complex types

Embedding (PESS 2026): the same building blocks carry a **software-defined digital twin framework**, shown on a containerized battery storage – every service gets its own shell and container.



Gefördert aus dem Transformationsprogramm Forschung und Wissenstransfer Saar durch:
Ministerium der Finanzen und für Wissenschaft SAARLAND
htw saar
UNIVERSITÄT DES SAARLANDES
dfki ai
IZES Institut für ZukunftsEnergie- und Stoffstromsysteme