SUNSPEC ALLIANCE

Webinar:

Unlocking Modbus

Accelerate Adoption with SunSpec DevKit & Open-Source Tools

July 31st, 10am PST

Antitrust and Intellectual Property Admonition

- All SunSpec meetings are conducted in accordance with the SunSpec Antitrust Policy and Intellectual Property Provisions defined in the SunSpec Member Agreement. This agreement can be found at: https://sunspec.org/sunspec-membership-agreement
- SunSpec strictly prohibits market participants, and their employees who take part in these activities, from using their participation as a forum for engaging in practices or communications that violate antitrust laws.
- Confidential or proprietary information should not be discussed in open session. Please contact SunSpec management if you have any questions.

FEATURED SPEAKERS

SUNSPEC ALLIANCE

Vish Ganti
President & COO

Niels Basjes
Cybersecurity Innovator

Peter Grace
DevOps Engineer

DER SECURITY

bol.

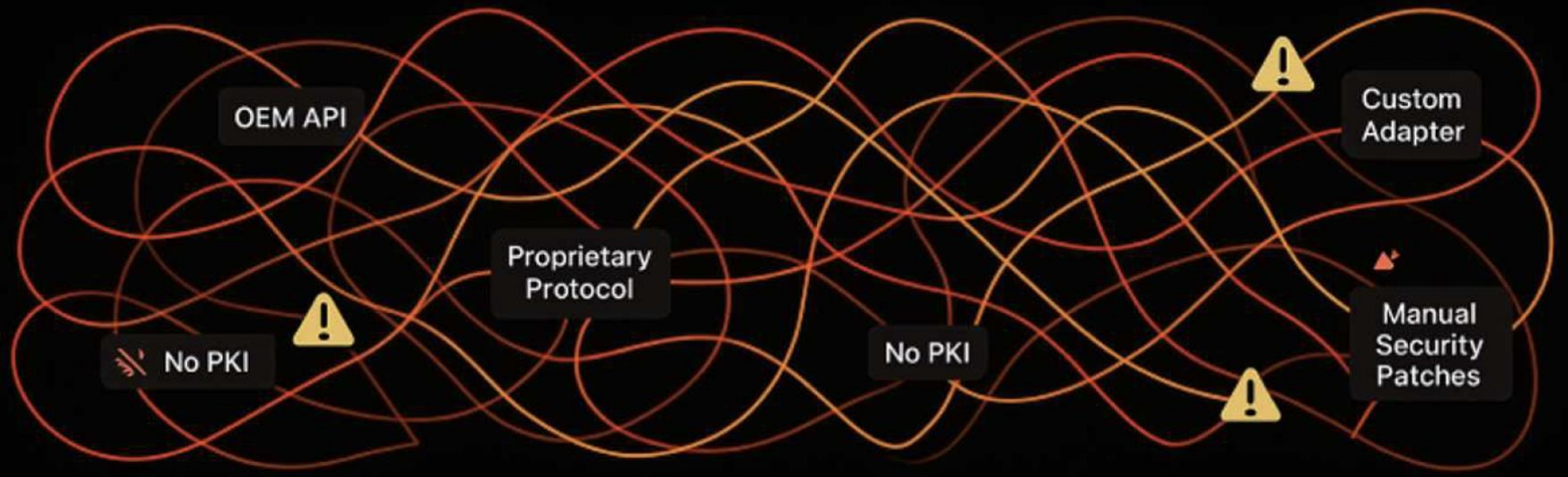
StackBlitz

Agenda For Today

- SunSpec Open-Source Program
- Modbus Mapping Schema Tool
- SunSpec Modbus Rust Stack with TLS
- SunSpec DevKit – Sneak Peak
- SunSpec Alliance Tech Summit
- Q&A
- Open Source Repos and Links

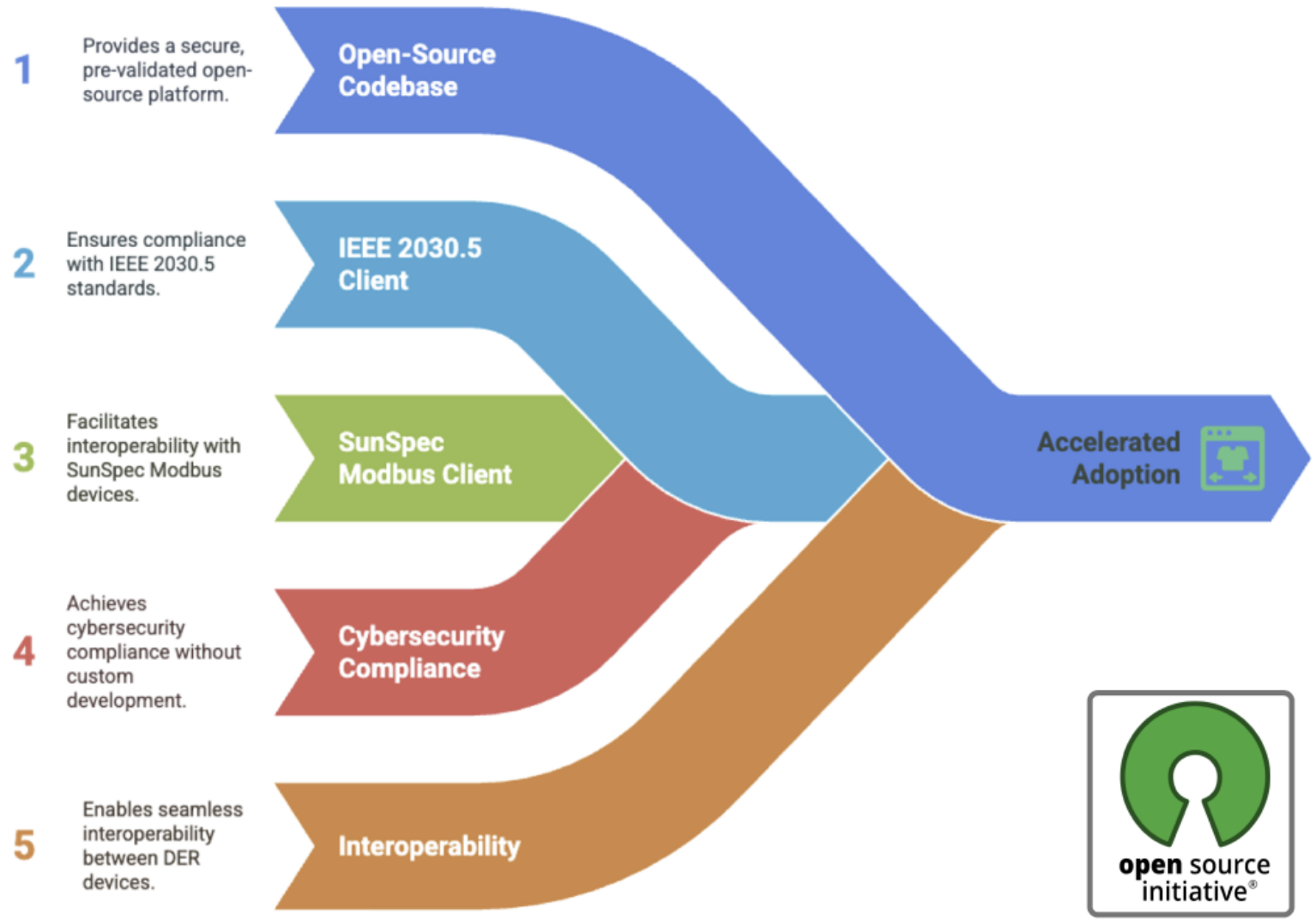
Today's DER Landscape

- Hundreds of custom OEM APIs
- Miilions wasted on integration and patchwork security
- No standardized onboarding
- Every new project = starting from scratch



Integration chaos. No interoperability. No security.

Open Source to Accelerate DER Adoption, Interoperability and Security



Modbus Evolution

Legacy Modbus

Simple protocol lacking security features.

SunSpec Modbus

Standardized data model for DER devices.

Secure SunSpec Modbus

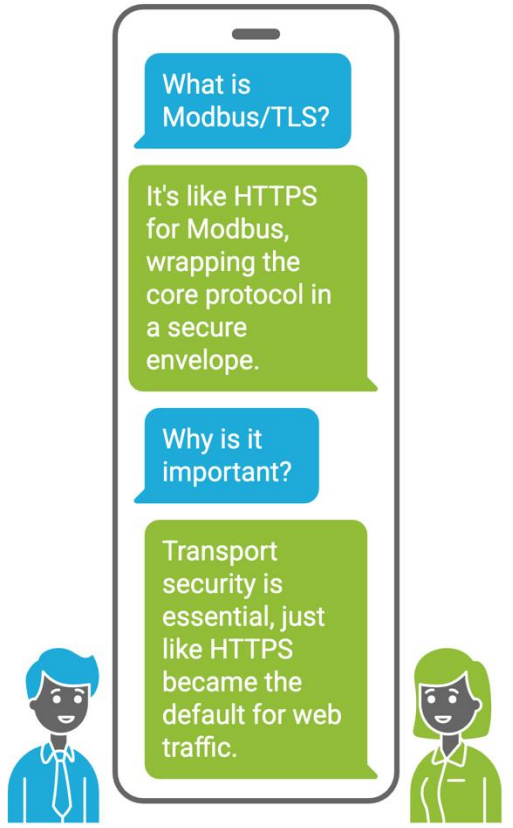
Modbus with added security and authentication.

Open-Source Enablement

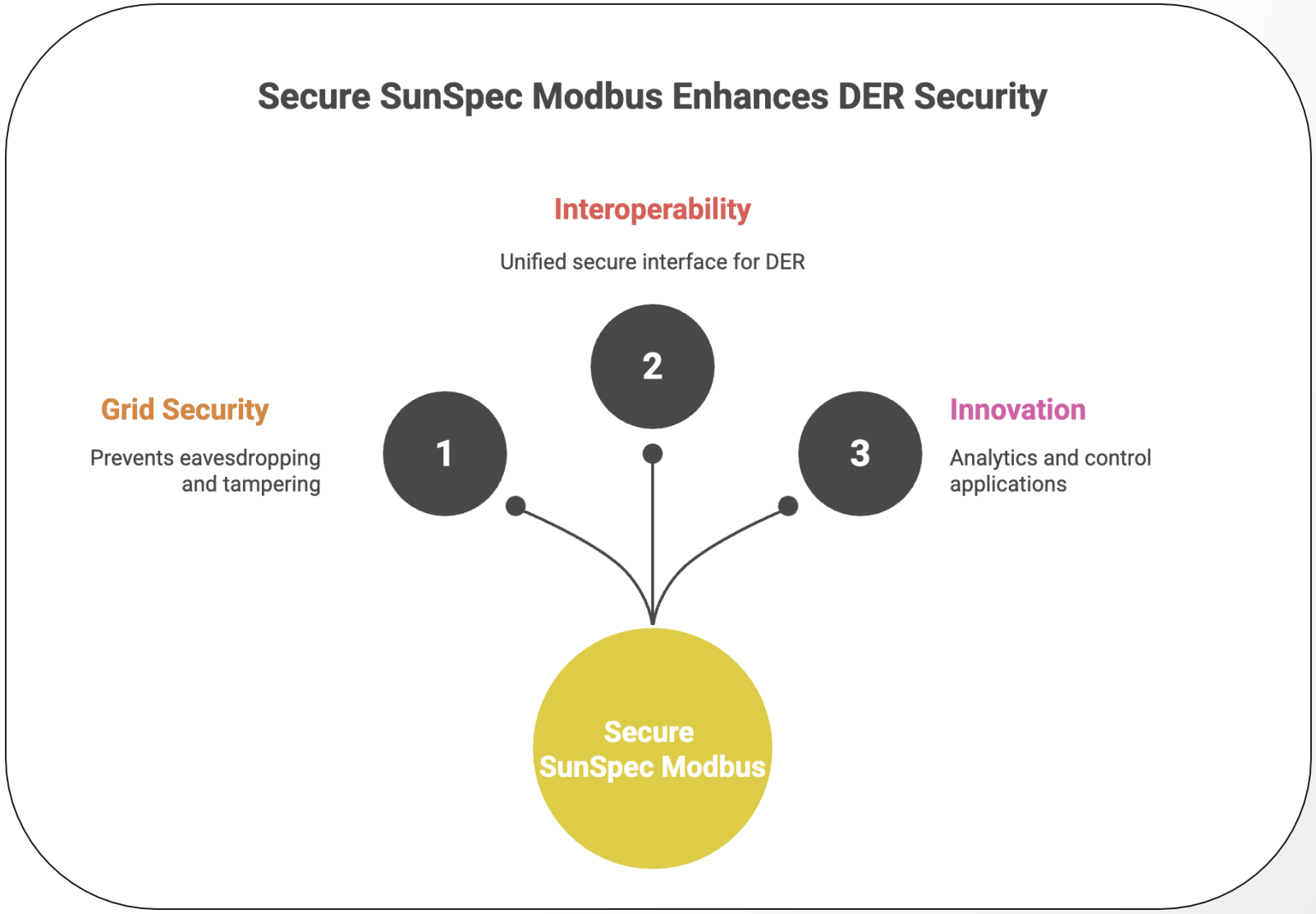
Community driving evolution with open-source projects.

- 2018 security profile adds TLS & X.509
- Upgrade path similar to HTTP → HTTPS
- Accelerate the adoption

Modbus/TLS: HTTPS for Modbus



Enhanced Grid Security



Open Source + Standards = Fast-Track Fix

- **Open Source = Resilience** – acts as insurance against industry headwinds and Black-Swan events
- **Focus on Real IP** – stop rewriting "glue code"; invest in differentiated innovation
- **Reduce Integration costs** – firms keep paying to address integration problems
- **Modbus Dominance** – vast majority of field devices still speak Modbus
- **The Two Core Issues**
  - **Custom Register Maps** – every vendor invents a new Modbus map
  - **Insecure by Default** – no authentication, no encryption on most deployments

Simplifying Modbus Integration

Raw (Custom) Modbus Maps

Complex, binary, and inaccessible

Schema Mapping

Define data structure with schemas

Open-Source Tools

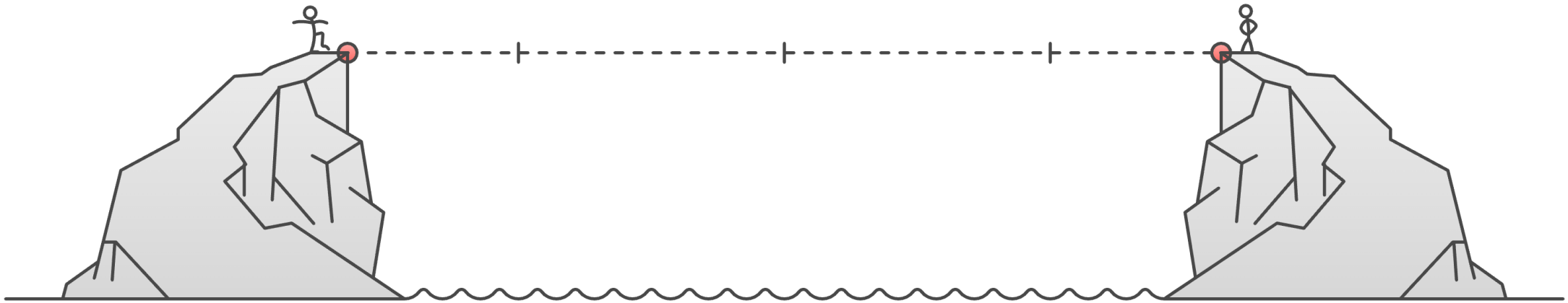
Utilize community-driven libraries

Gateway Integration

Bridge Modbus to MQTT

Usable SunSpec Modbus Data

Structured, accessible, and scalable

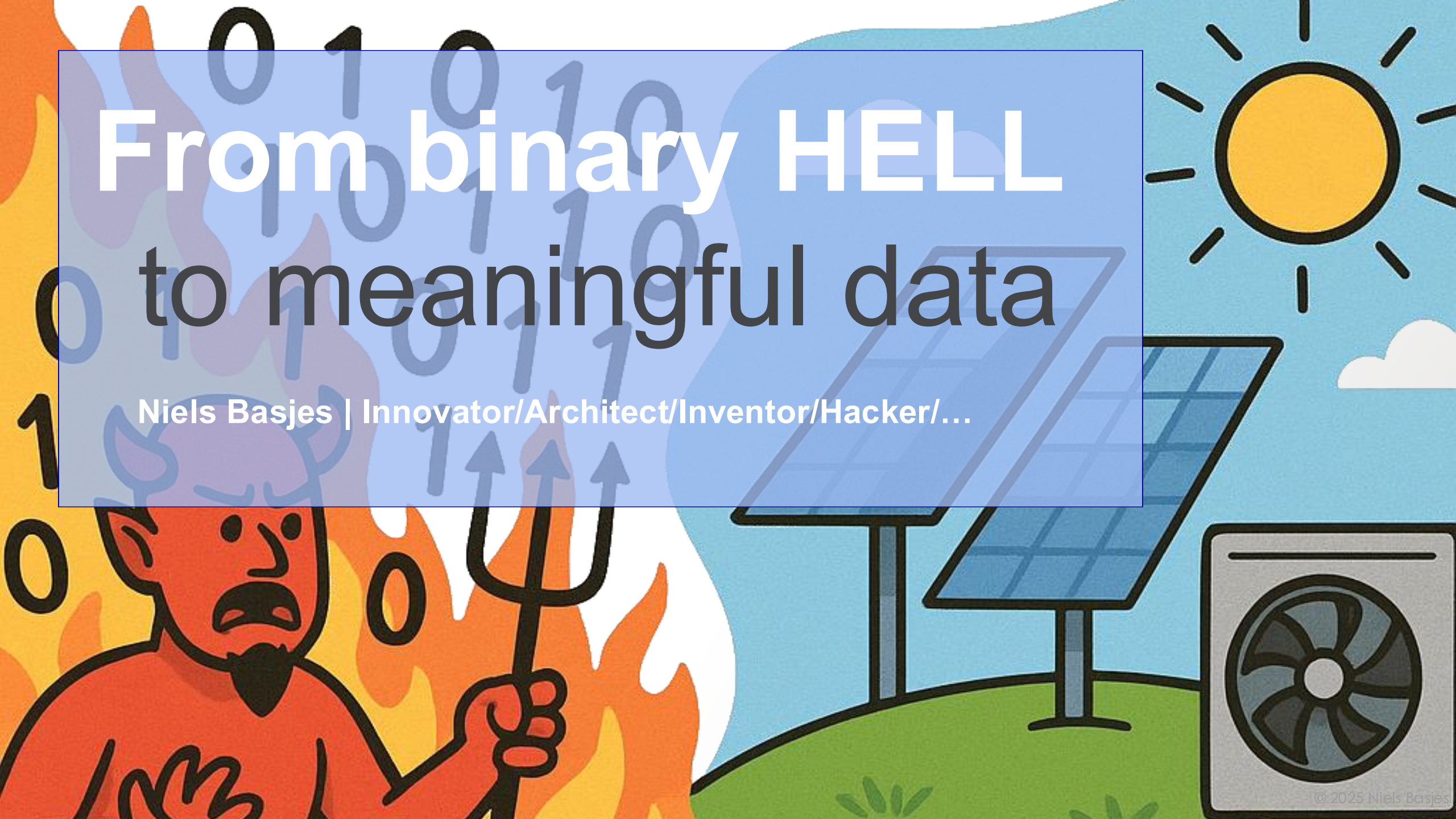


Demo Time!

Niels Basjes

Modbus Schema Toolkit

https://modbus.basjes.nl
https://github.com/nielsbasjes/modbus-schema


From binary HELL
to meaningful data
Niels Basjes | Innovator/Architect/Inventor/Hacker/…
© 2025 Niels Basjes

Niels Basjes

Hack^h^h^h^h Innovator…

About me

- IT Architect/Innovator
  - Large scale systems design
  - Streaming BigData DataScience
  - Cyber security
- Opensource fanatic
- I talk a lot
  - youtube.basjes.nl

01

At home

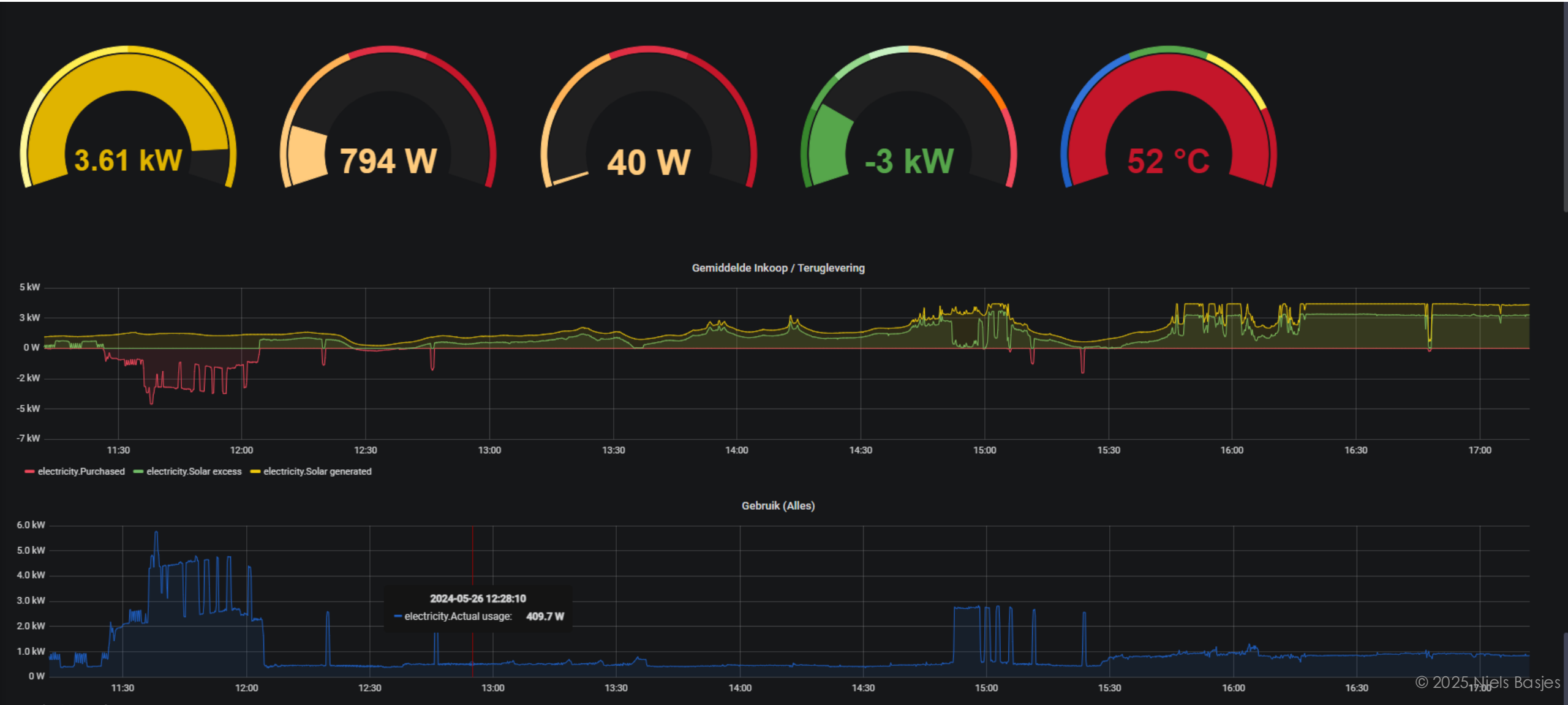
I try to be energy efficient

My Green Energy devices

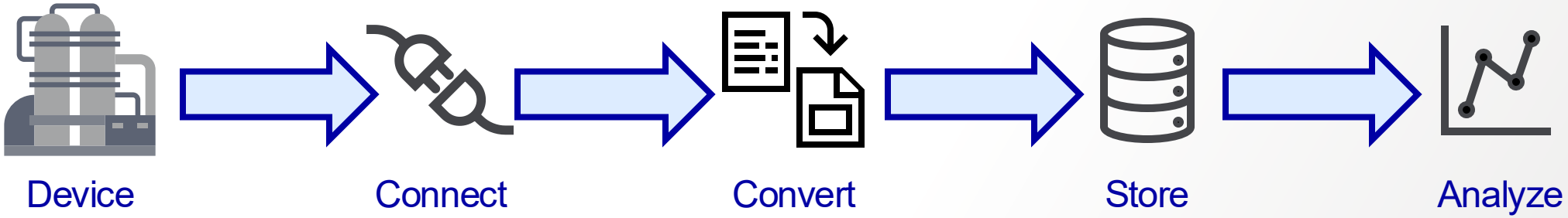
- Electricity meters
- Solar Inverter
- Heat pump

dsmr.basjes.nl

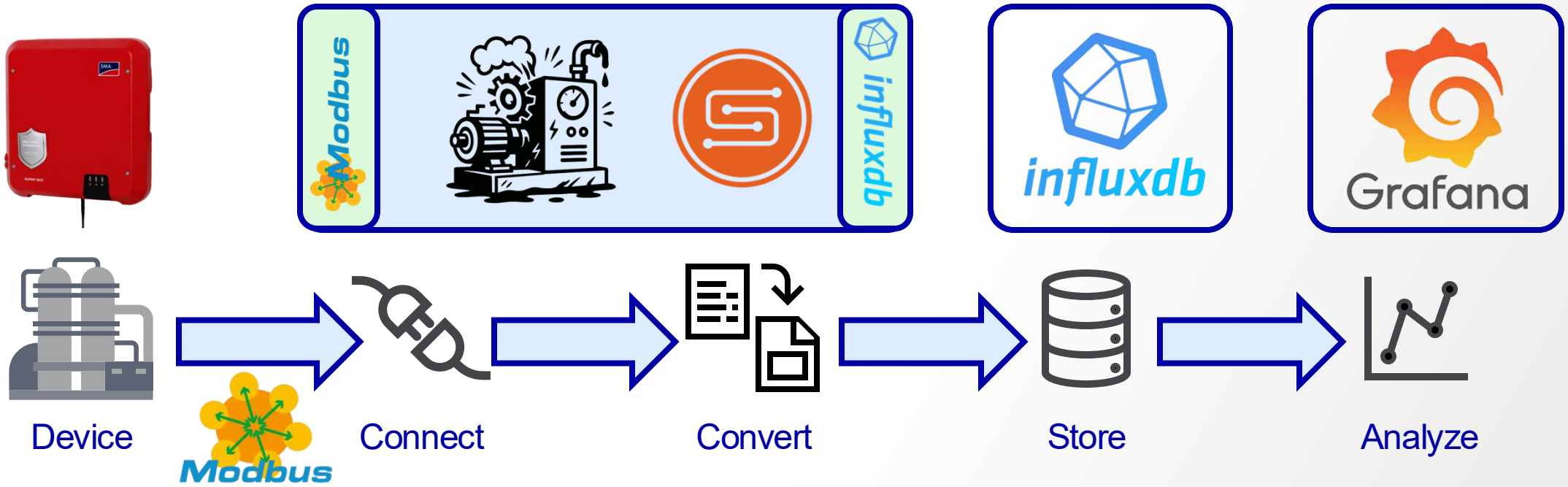
My home grown dashboard on Grafana



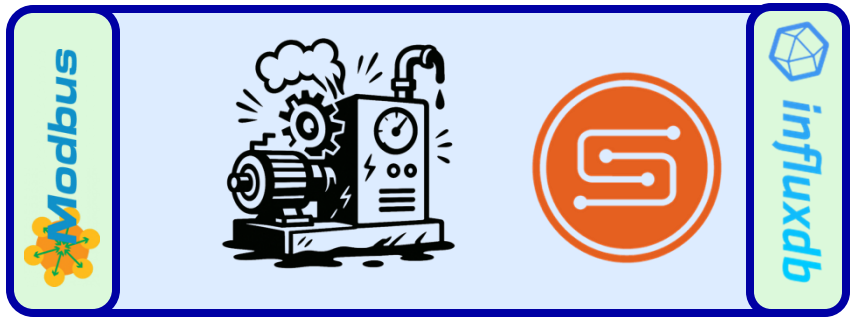
What are my systems doing?

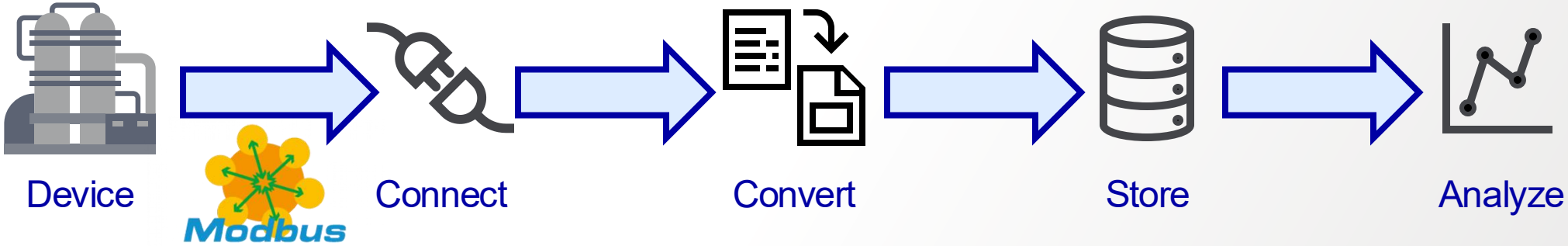


Getting SunSpec data (2019)



Getting Power consumption data (2019)





The things I faced (2019-2023)

- The actual Modbus implementation code exists.
- Adding meaning to the bits.
  - Almost non-existent.
- Modbus mapping information is confusing.
  - Some notations are “Off by 1” and “Base 10 oriented”
- My Modbus devices are slow
  - 100-200ms per request is “normal”.
- Optimizing the queries is hard.
  - Especially when some registers return errors.
- Writing code for each device is a lot of work and error prone.
  - Device specific code

Specific example field

- My solar inverter has a field with the type/model of the device
  - 16 registers (32 bytes) UTF-8 encoded
  - My inverter model “SB3.6-1AV-41”
  - Encoded using 16 registers starting at 4x40021
  - ```
    S B   3 .   6 -   1 A   V -   4 1
    0x5342 0x332E 0x362D 0x3141 0x562D 0x3431 0x0000 0x0000
    0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
    ```
    ```

You NEED the schema to read the data!

- Exactly read the field: **16 registers @ 4x40021**

```
0x5342 0x332E 0x362D 0x3141 0x562D 0x3431 0x0000 0x0000
0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
```

- Only read the first 15 registers: **15 registers @ 4x40021**
  - **Modbus error: INVALID_ADDRESS**

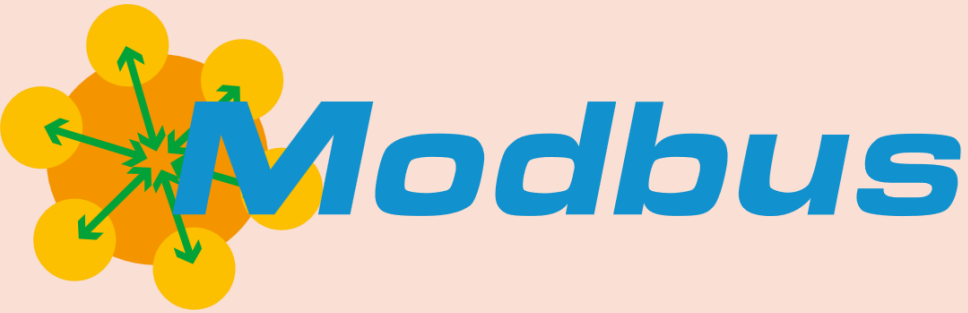
- Only read the last 15 registers: **15 registers @ 4x40022**

```
0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF
0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF 0xFFFF
```

My new heat pump (Q4 2022)

- Thermia Calibra Cool 7
- Thermia send me the mapping
  - A PDF with > 450 fields

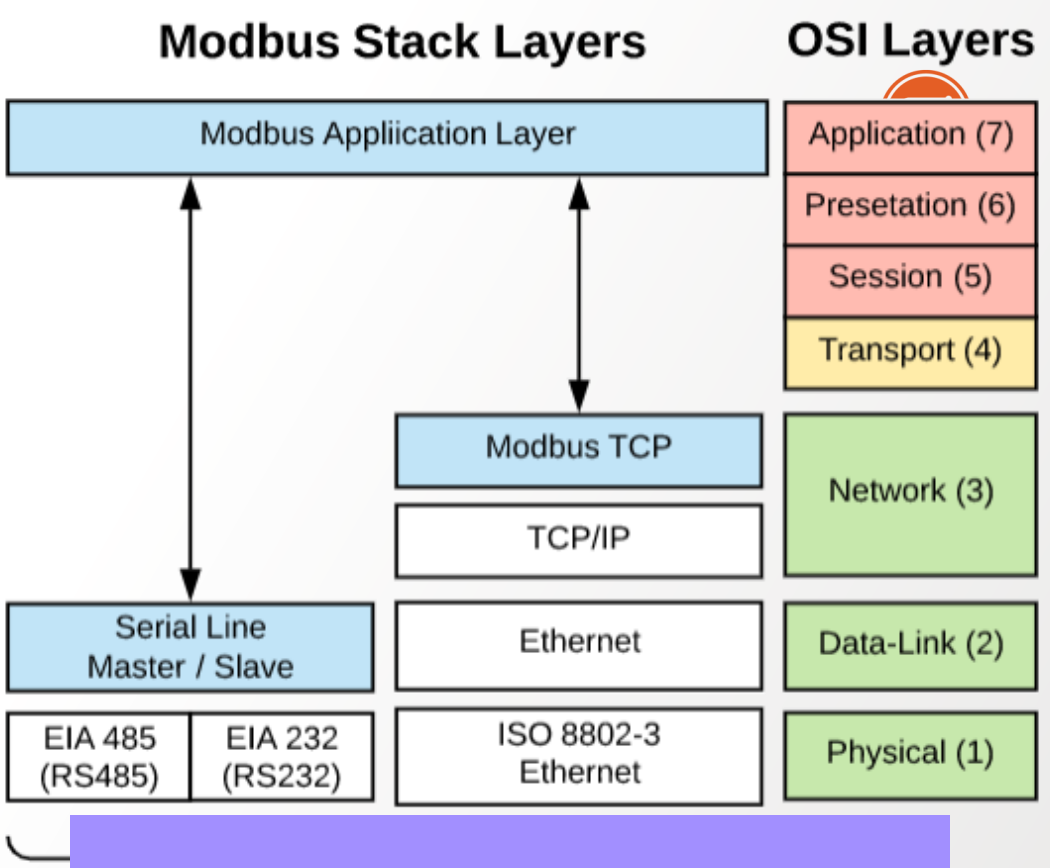
02



The common factor

Modbus

- Binary data exchange protocol
  - Developed in 1979 by Modicon
  - My devices offer this over
    - TCP/IP: Regular networking
    - RS485: Serial communication
- Essentially “Remote PEEK & POKE”
  - Get/Put single bits
  - Get/Put 16-bit registers
- Modbus does not tell you the meaning
  - Manufacturer provides the meaning



Modbus Mapping

- Translating the registers to their meaning.
- Most vendors
  - PDFs
  - With tables.

1.2.1 SDM630Modbus Input Registers

Address (Register)	Parameter Number	SDM630Modbus Input Register Parameter		Modbus Protocol Start Address Hex	
		Description	Units	Hi Byte	Lo Byte
30001	1	Phase 1 line to neutral volts.	Volts	00	00
30003	2	Phase 2 line to neutral volts.	Volts	00	02
30005	3	Phase 3 line to neutral volts.	Volts	00	04
30007	4	Phase 1 current.	Amps	00	06
30009	5	Phase 2 current.	Amps	00	08
30011	6	Phase 3 current.	Amps	00	0A
30013	7	Phase 1 power.	Watts	00	0C
30015	8	Phase 2 power.	Watts	00	0E
30017	9	Phase 3 power.	Watts	00	10
30019	10	Phase 1 volt amps.	VA	00	12
30021	11	Phase 2 volt amps.	VA	00	14
30023	12	Phase 3 volt amps.	VA	00	16

Data Format: 4 bytes (2 registers) per parameter. Floating point format (to IEEE 754)

Most significant register first (Default). The default may be changed if required -See Holding Register "Register Order" parameter.

Eastron®

Modbus Mapping: Needs math

- Actual value = int16(register bits) / scale

INPUT REGISTERS - Function codes: 4=read input registers						
Position number	Units	Reference to	Address	De Facto Address	Scale	Description
	A	9	69	30070	100	Electric meter L1 current (A)
	A	9	70	30071	100	Electric meter L2 current (A)
	A	9	71	30072	100	Electric meter L3 current (A)
	V	9	72	30073	100	Electric meter L1-0 voltage (V)
	V	9	73	30074	100	Electric meter L2-0 voltage (V)
	V	9	74	30075	100	Electric meter L3-0 voltage (V)
	V	9	75	30076	10	Electric meter L1-L2 voltage (V)
	V	9	76	30077	10	Electric meter L2-L3 voltage (V)
	V	9	77	30078	10	Electric meter L3-L1 voltage (V)
	W	9	78	30079	1	Electric meter L1 power (W)
	W	9	79	30080	1	Electric meter L2 power (W)
	W	9	80	30081	1	Electric meter L3 power (W)

Modbus Mapping: Lots of types

- Real mappings hold many types …
  - Ints, Floats, Booleans, Strings, Enums, Bitsets, …

Footnotes:
***1)** 1: Manual operation, 2: Defrost, 3: Hot water, 4: Heat, 5: Active Cooling, 6: Pool, 7: Anti legionella, 8: Passive Cooling 98: Standby 99: No demand 100: OFF
***2)** 1: OFF, 2: Standby, 3: ON/Auto
***3)** Different heat pumps have different number of available gears.
For instance: Commercial can have 10, while domestic can have 9 gears.
***4)** These applies to Smart grid function. 1: EVU, 4: Normal, 5: Comfort, 6: Boost
***5)** Should always be set to 1 i auto mode
***6)** Bit 0: Manual operation Bit 1: Defrost, Bit 2: Hot water, Bit 3: Heat, Bit 4: Active Cooling, Bit 5: Pool, Bit 6: Anti legionella, Bit 7: Passive Cooling, Bit 8: Reserved, Bit 9: Standby, Bit 10: No demand, Bit 11: OFF

***7)** 0=Disable immersion heater, 2=Internal immersion heater enabled

Modbus Mapping: SunSpec

- Generic standard for solar/energy related devices
- Modular Modbus Mapping
  - Model per functionality: PV (Solar), Battery, IPv4, …
- Modbus Mapping for a specific SunSpec device:
  - Query the device for the implemented set of Models.
  - Generate the modbus mapping from
    - the device data
    - the known SunSpec models.

SUNSPEC ALLIANCE

Modbus Mapping: SunSpec math

- Part of Model 102 (simplified):
  - `{ "desc": "AC Current", "name": "A", "sf": "A_SF", "size": 1, "type": "uint16", "units": "A"},`
- later in this model:
  - `{ "name": "A_SF", "size": 1, "type": "sunssf" },`
  - sunssf: Scaling Factor; 1 register int16; values -10…10
  - `int16( A_SF register )` → `A_SF`
- The real calculation:
  - `A = uint16( A register ) * 10^A_SF`

03

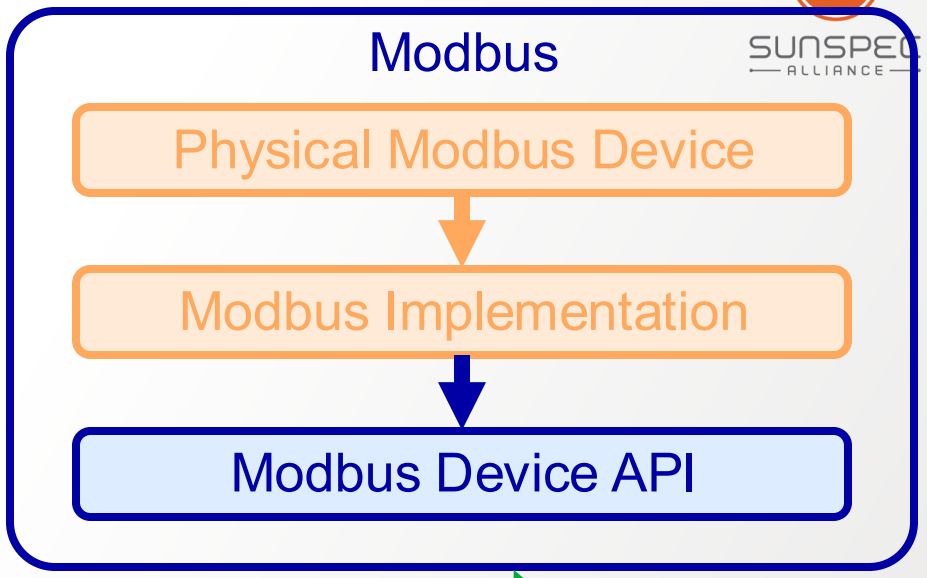
Modbus Schema

It’s just a hobby project …

Core goals

- Solve the Modbus Mapping problem once and for all.
- For all my devices
  - Including all types, math and complexities
  - Including SunSpec
- Usable for all my usecases
  - Read only
  - Optimal queries
  - Kotlin/Java/JVM
- Long term simple
  - A new device should be easy
  - A new integration should be easy

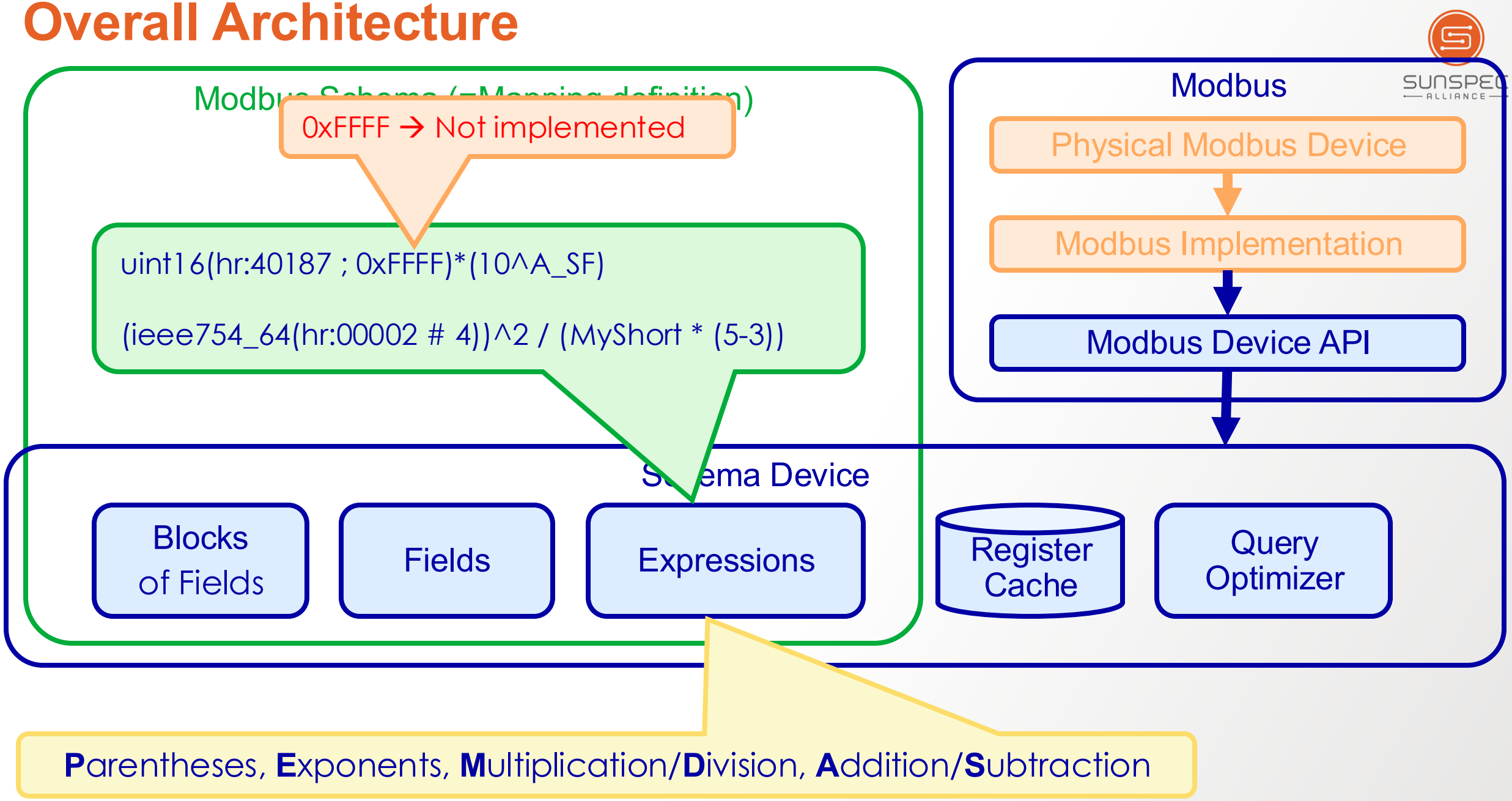
Overall Architecture



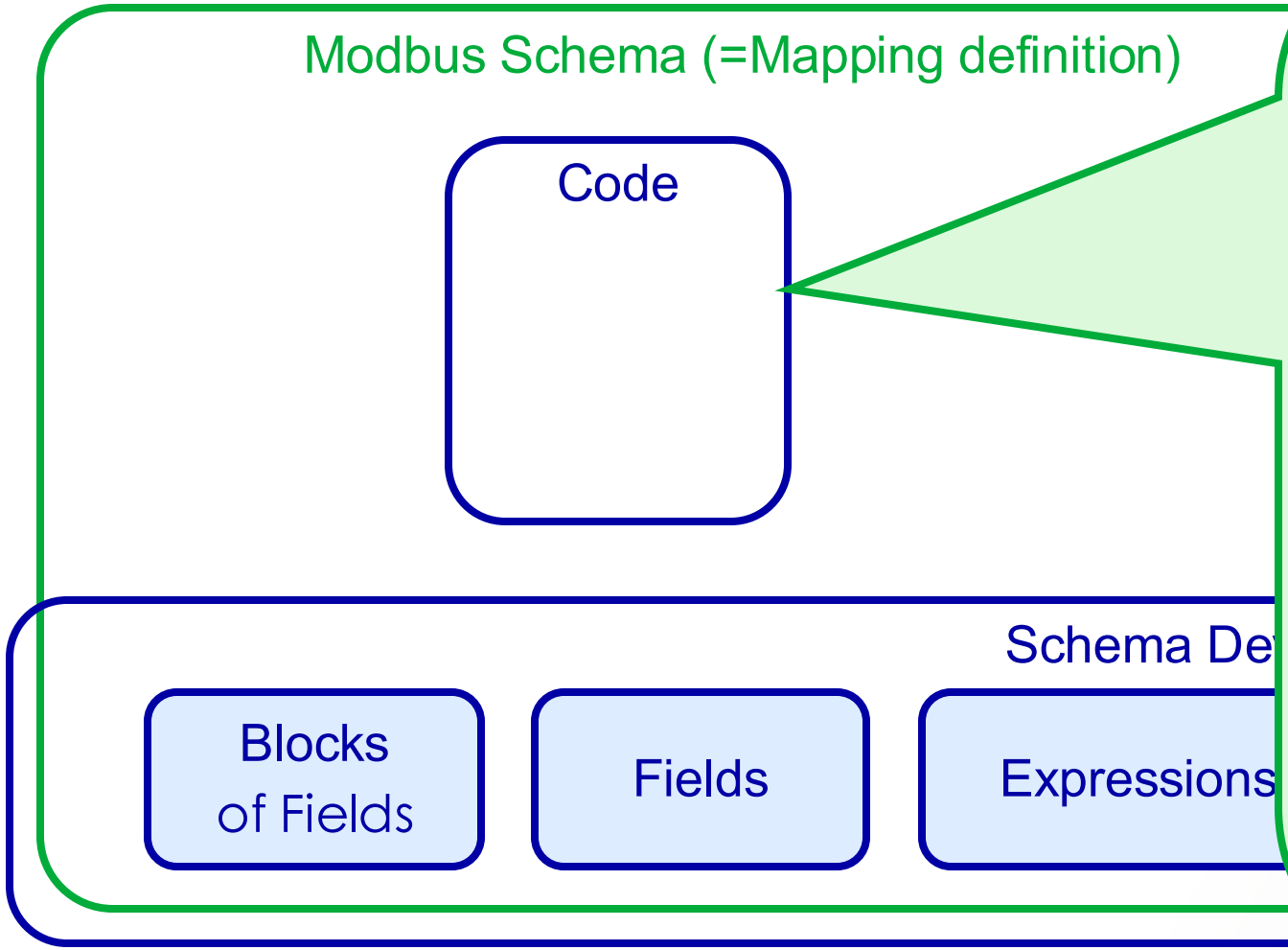
Supports many address notations

- Modicon:
  - 40124, "40124", "400124"
- Parsable (but still off by 1):
  - "4x124", "4x00124", "4x000124"
- Better notation:
  - "holding-register:123", "hr:123"

Overall Architecture



Overall Architecture

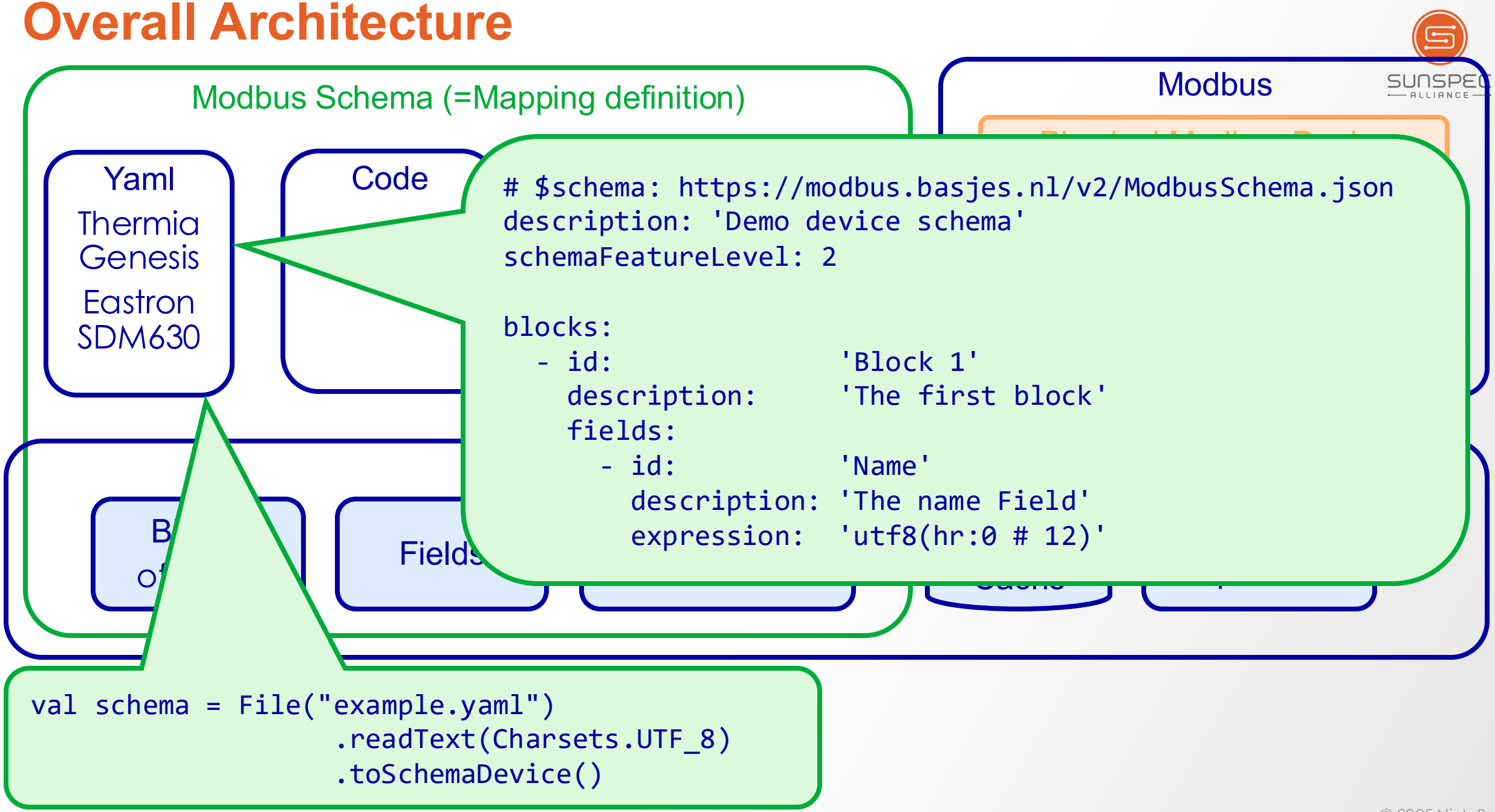



```
val schemaDevice = SchemaDevice()

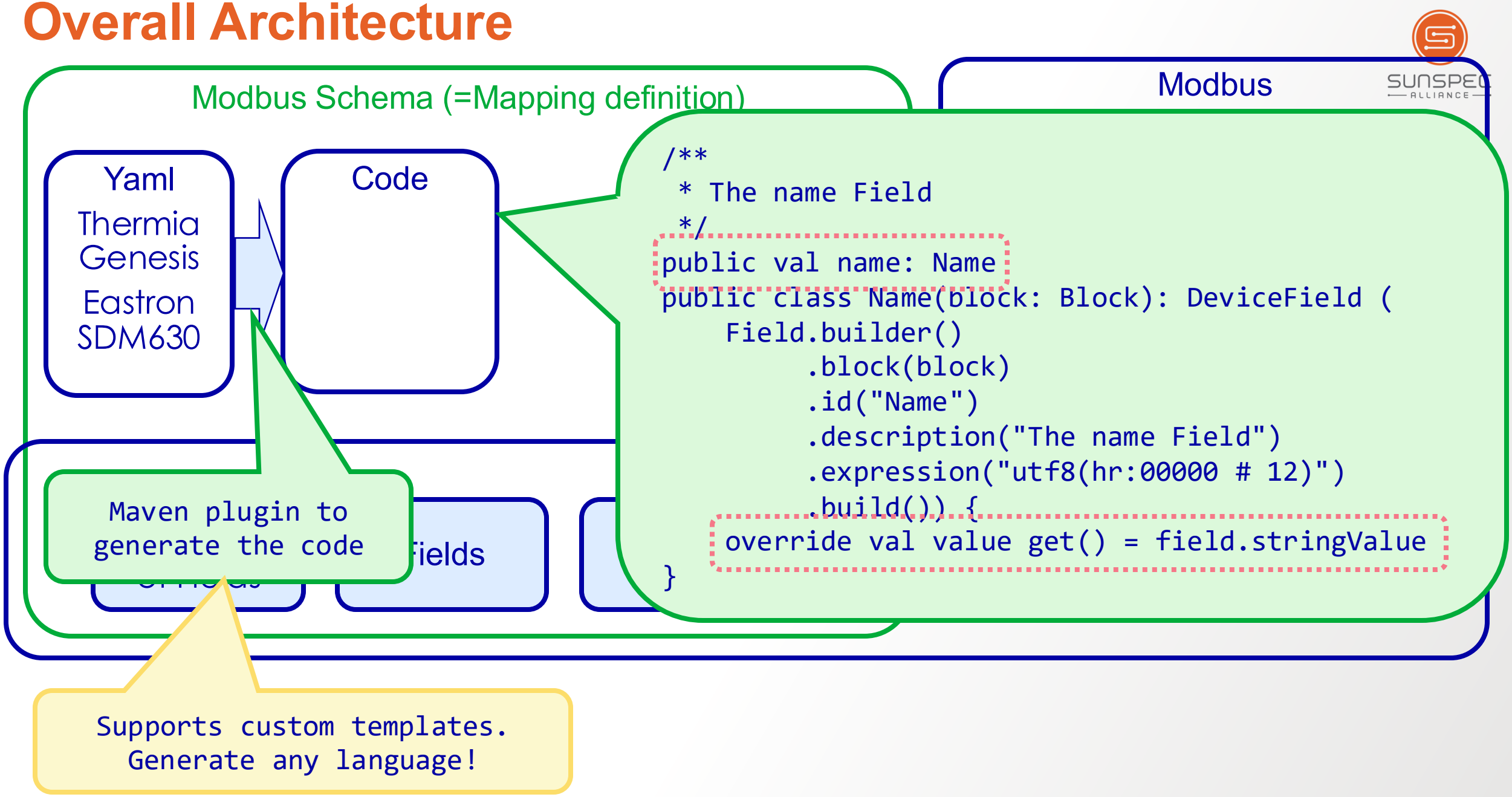
val block = Block.builder()
  .schemaDevice(schemaDevice)
  .id("Block 1")
  .description("The first block")
  .build()

Field.builder()
  .block(block)
  .id("Name")
  .description("The name Field")
  .expression("utf8(hr:00000 # 12)")
  .build()
```

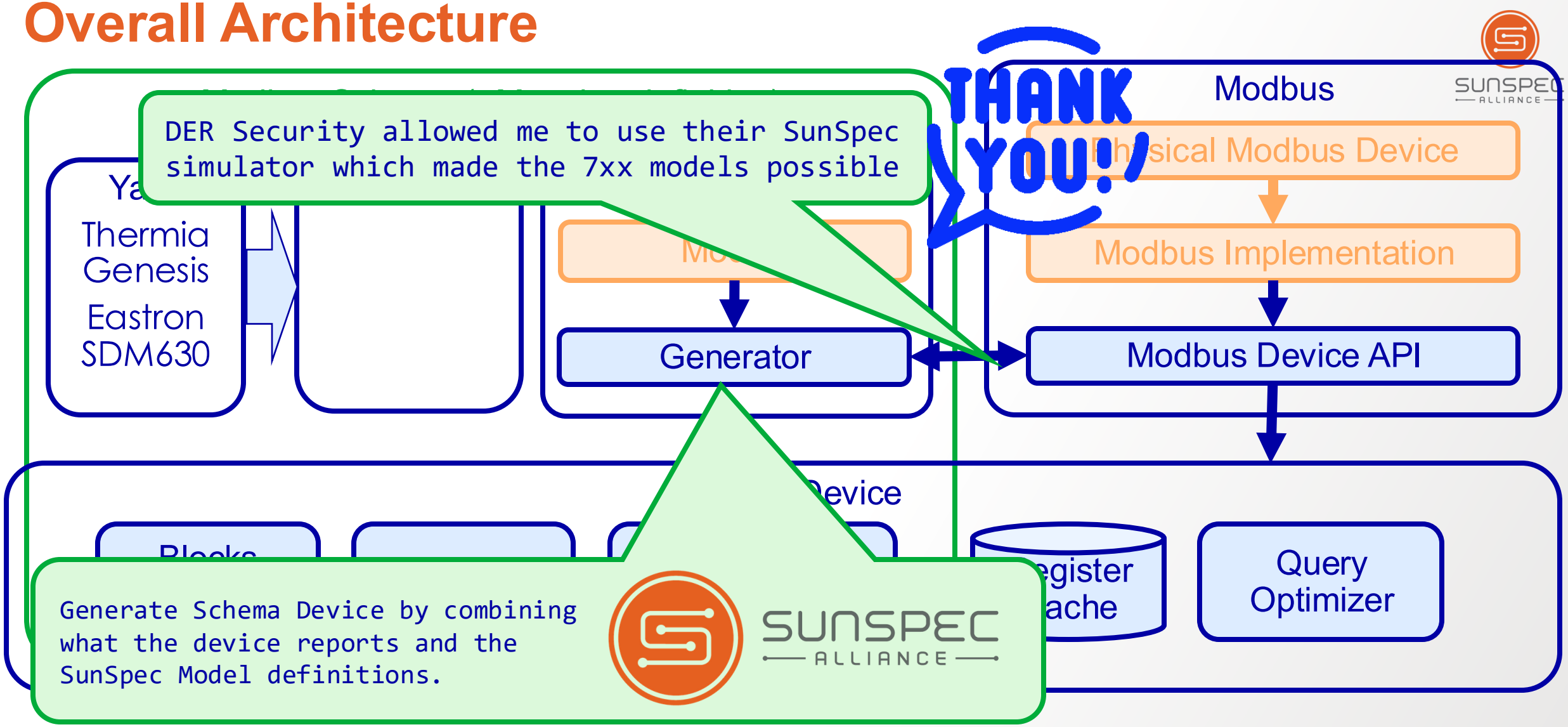
Overall Architecture



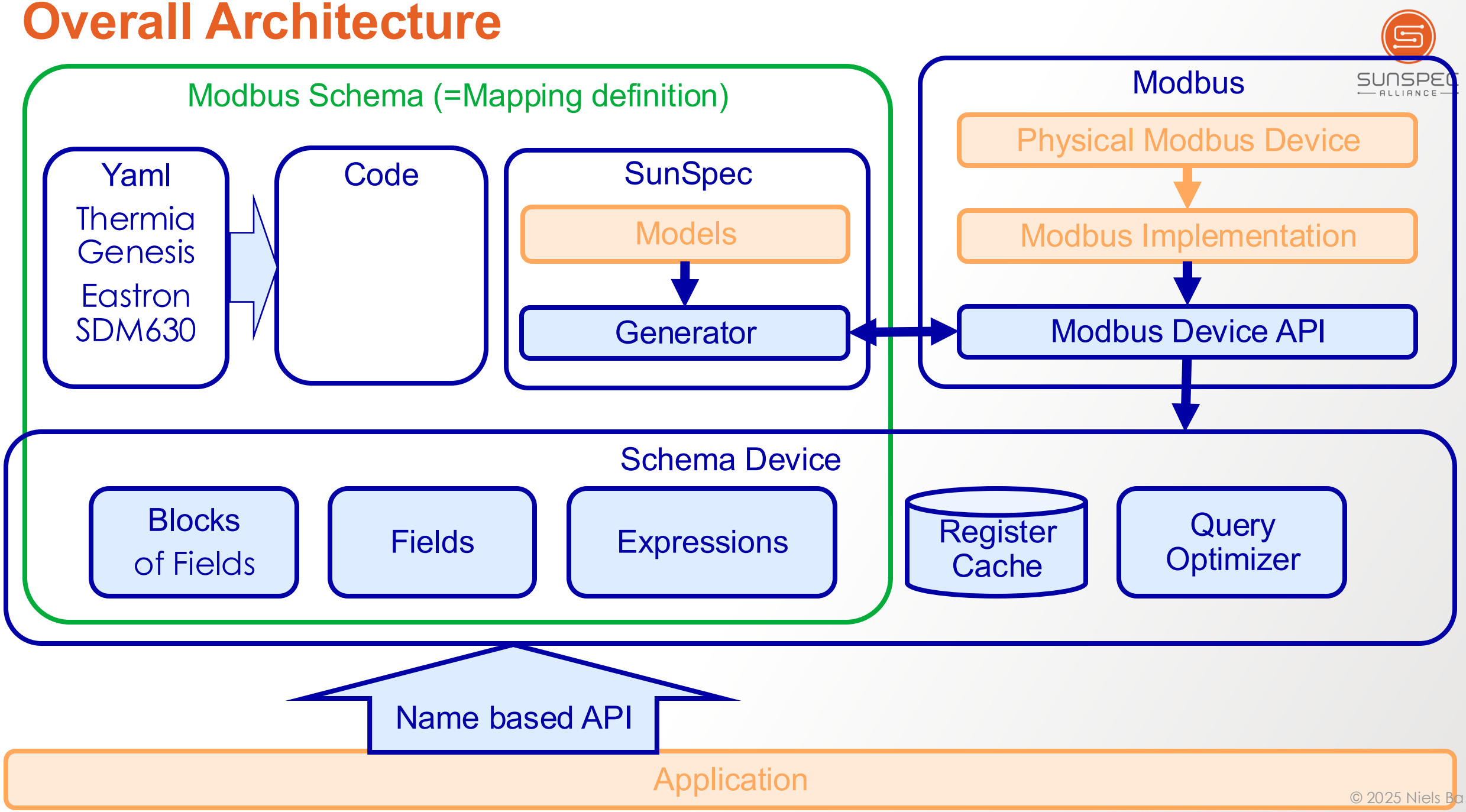
Overall Architecture



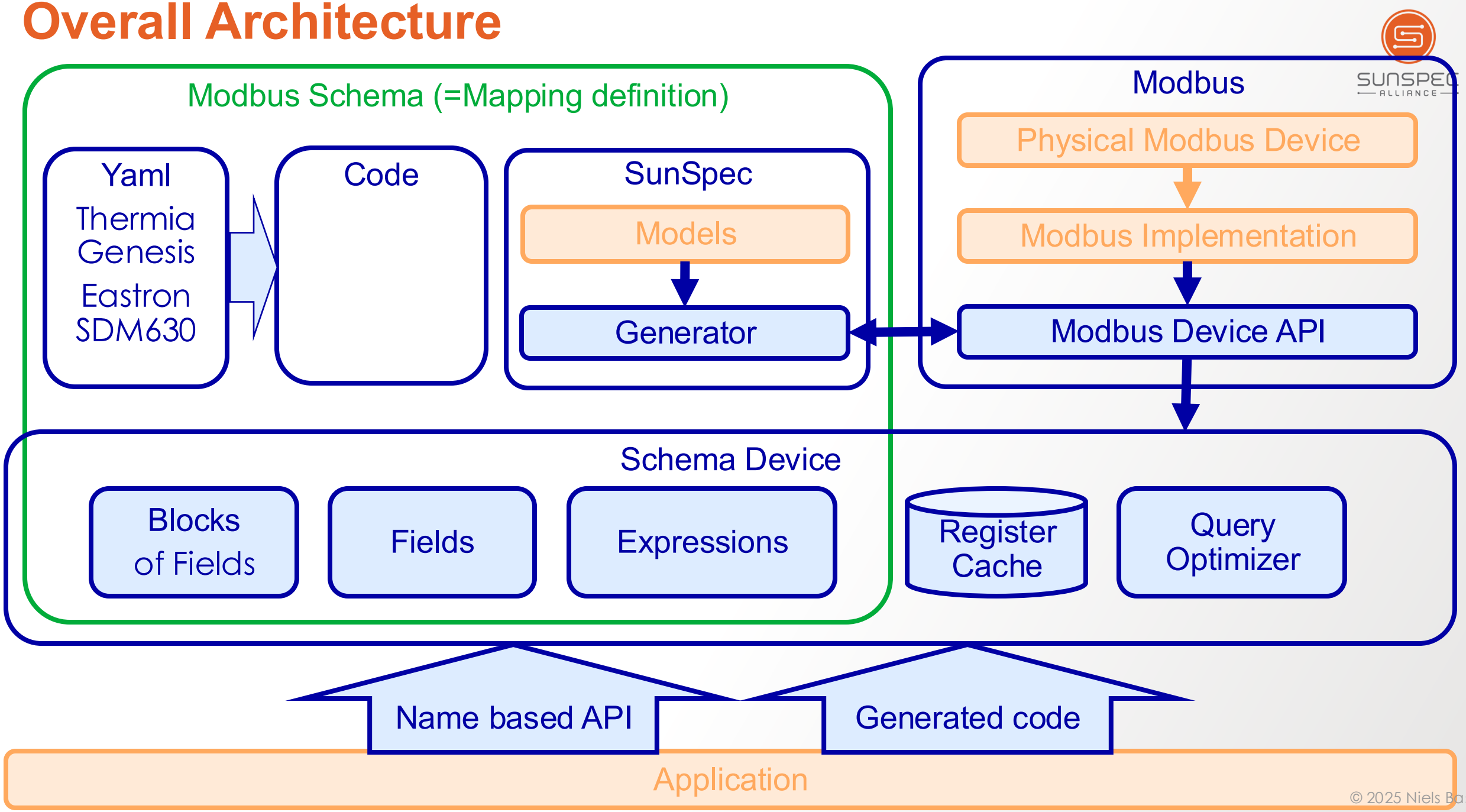
Overall Architecture



Overall Architecture



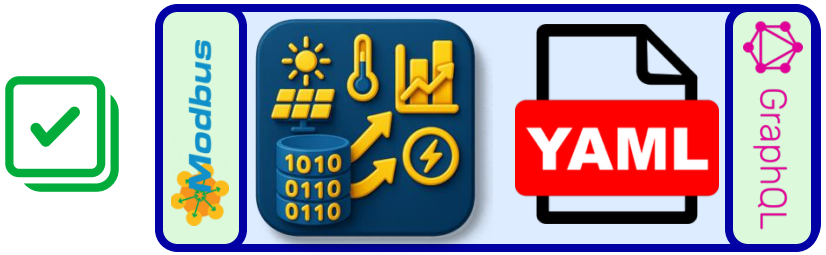
Overall Architecture



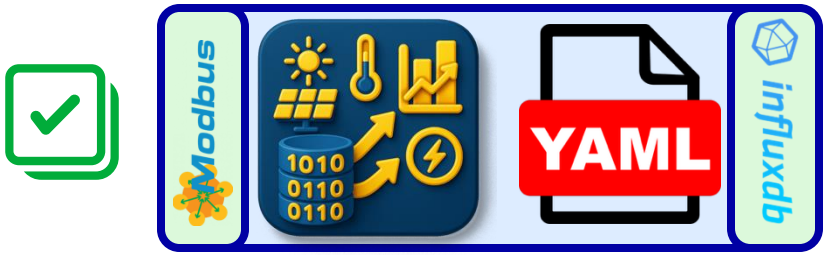
Making it easier to use

```
docker run
        -p8080:8080
        nielsbasjes/sunspec-graphql:latest
        --sunspec.host=sunspec.iot.basjes.nl
        --sunspec.port=502
        --sunspec.unit=126
```

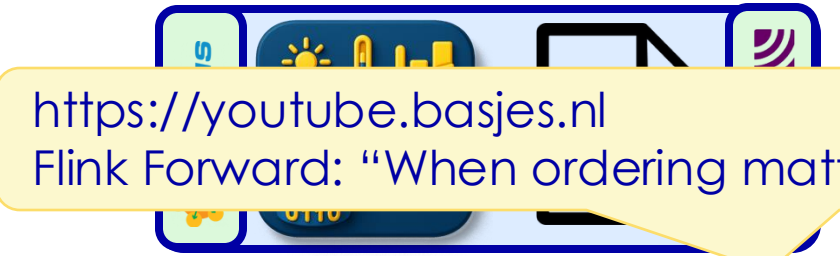
SUNSPEC ALLIANCE



docker

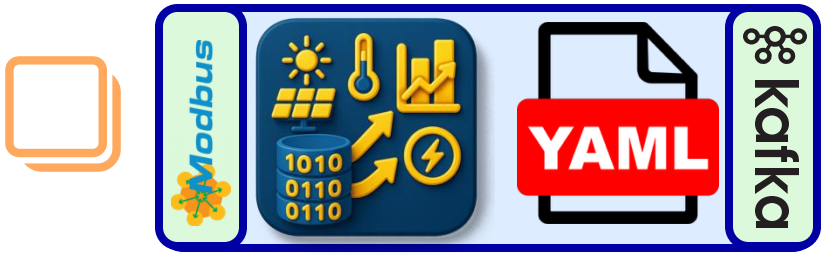


Kotlin Script



Kotlin Script

https://youtube.basjes.nl
Flink Forward: “When ordering matters”



Kotlin Script

GraphQL

model1: Model1

[Model 1]: All SunSpec con

model11: Model11

[Model 11]: Include to supp

model12: Model12

[Model 12]: Include to supp

model101: Model101

[Model 101]: Include this m

model120: Model120

[Model 120]: Inverter Contr

model121: Model121

[Model 121]: Inverter Contr

model122: Model122

[Model 122]: Inverter Contr

model123: Model123

[Model 123]: Immediate Inverter Controls

Model101

[Model 101]: Include this mode
inverter monitoring

Fields

acCurrent: Float

AC Current. (A)

acCurrentPhaseA: Float

Phase A Current. (A)

acCurrentPhaseB: Float

Phase B Current. (A)

acCurrentPhaseC: Float

Phase C Current. (A)

acVoltagePhaseAB: Float

Phase Voltage AB. (V)

```
subscription {
  deviceData(intervalMs:1000, maxAgeMs:500)
    totalUpdateDurationMs
    requestTimestamp
    dataTimestamp
    model1 {
      manufacturer
      model
      version
    }
    model101 {
      acPower
      acEnergy
      acCurrent
      acPowerFactor
      acApparentPower
      acReactivePower
      cabinetTemperature
    }
    model160 {
      module_0_DCCurrent
      module_0_DCPower
      module_0_DCVoltage
      module_0_InputID

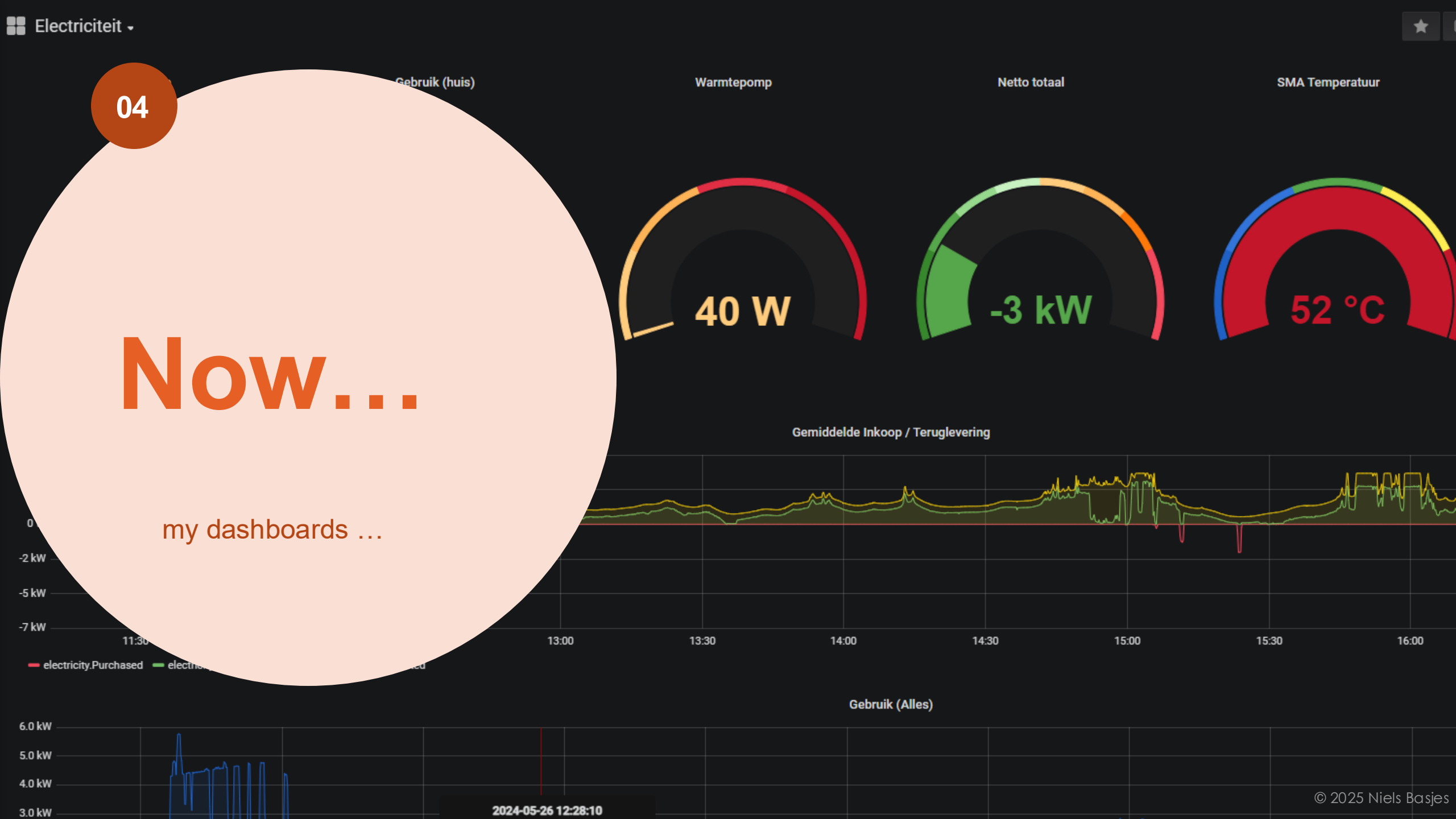
      module_1_DCCurrent
      module_1_DCPower
      module_1_DCVoltage
      module_1_InputID
    }
  }
}
```


```
{
  "data": {
    "deviceData": {
      "totalUpdateDurationMs": 289,
      "requestTimestamp": "2025-07-23T08:42:14.32
      "dataTimestamp": "2025-07-23T08:42:14.609Z"
      "model1": {
        "manufacturer": "SMA",
        "model": "SB3.6-1AV-41",
        "version": "4.01.15.R"
      },
      "model101": {
        "acPower": 690,
        "acEnergy": 31871890,
        "acCurrent": 3,
        "acPowerFactor": -0.99,
        "acApparentPower": 710,
        "acReactivePower": 100,
        "cabinetTemperature": 44
      },
      "model160": {
        "module_0_DCCurrent": 1.5,
        "module_0_DCPower": 460,
        "module_0_DCVoltage": 309,
        "module_0_InputID": 1,
        "module_1_DCCurrent": 1.4000000000000001,
        "module_1_DCPower": 320,
        "module_1_DCVoltage": 224,
        "module_1_InputID": 2
      }
    }
  }
}
```

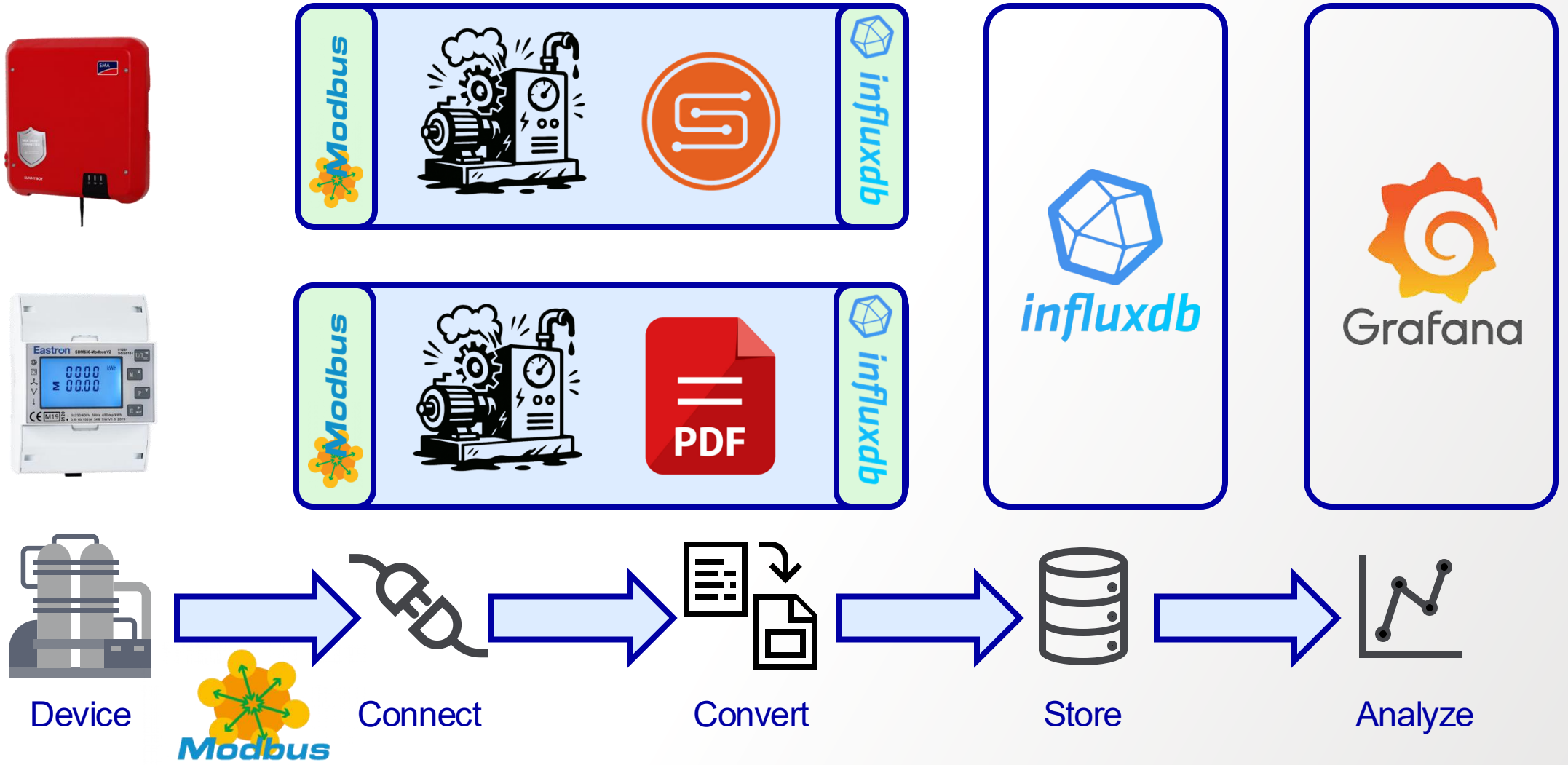
04

Now…

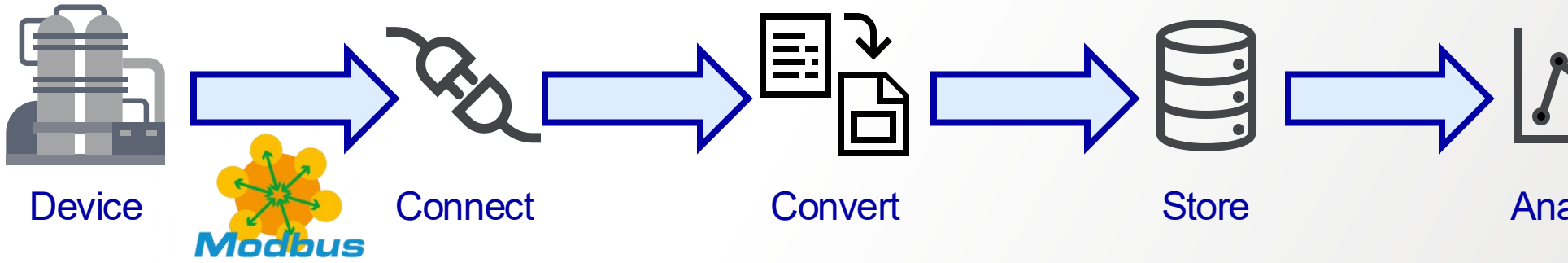
my dashboards …



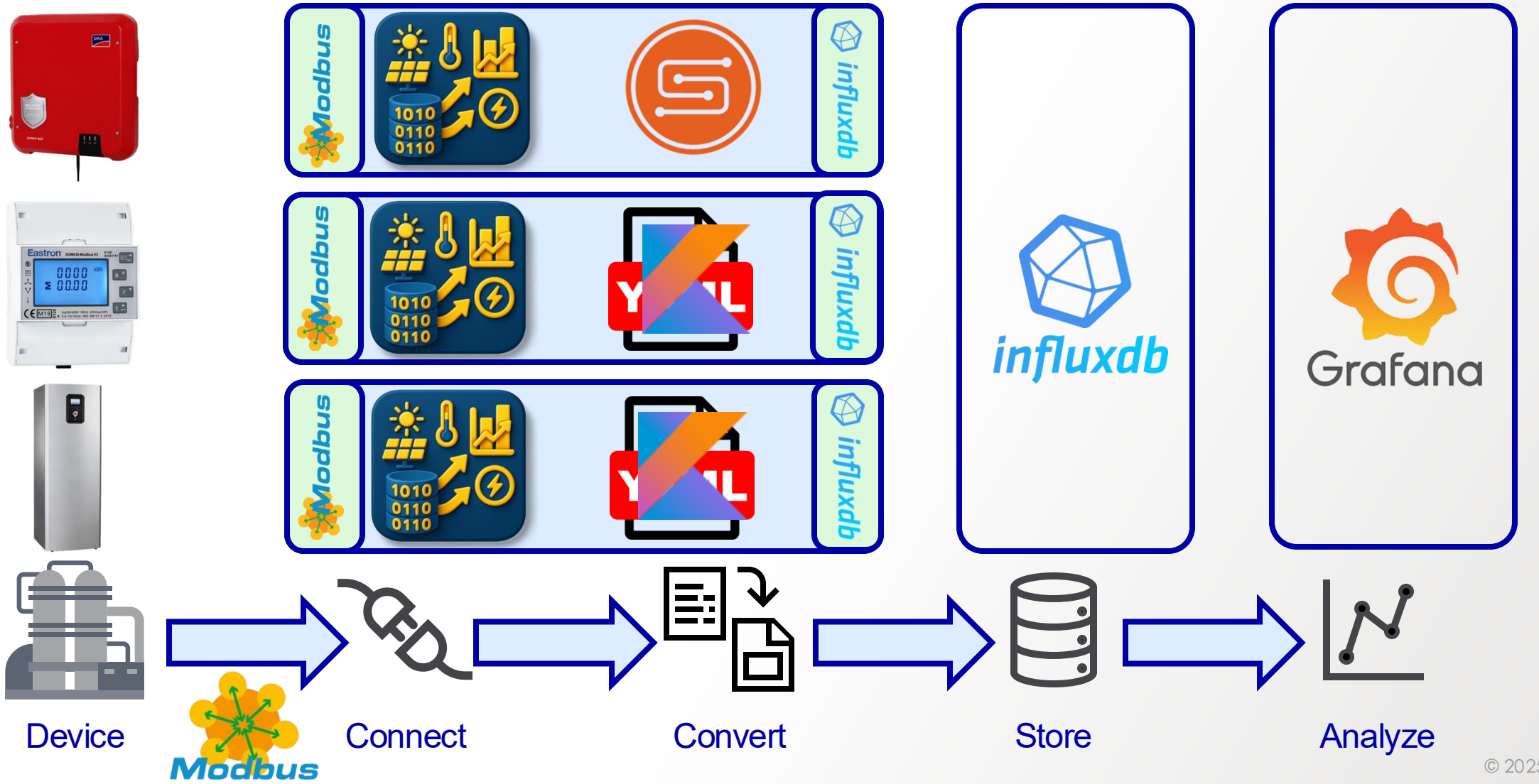
What I had (2019-2025)



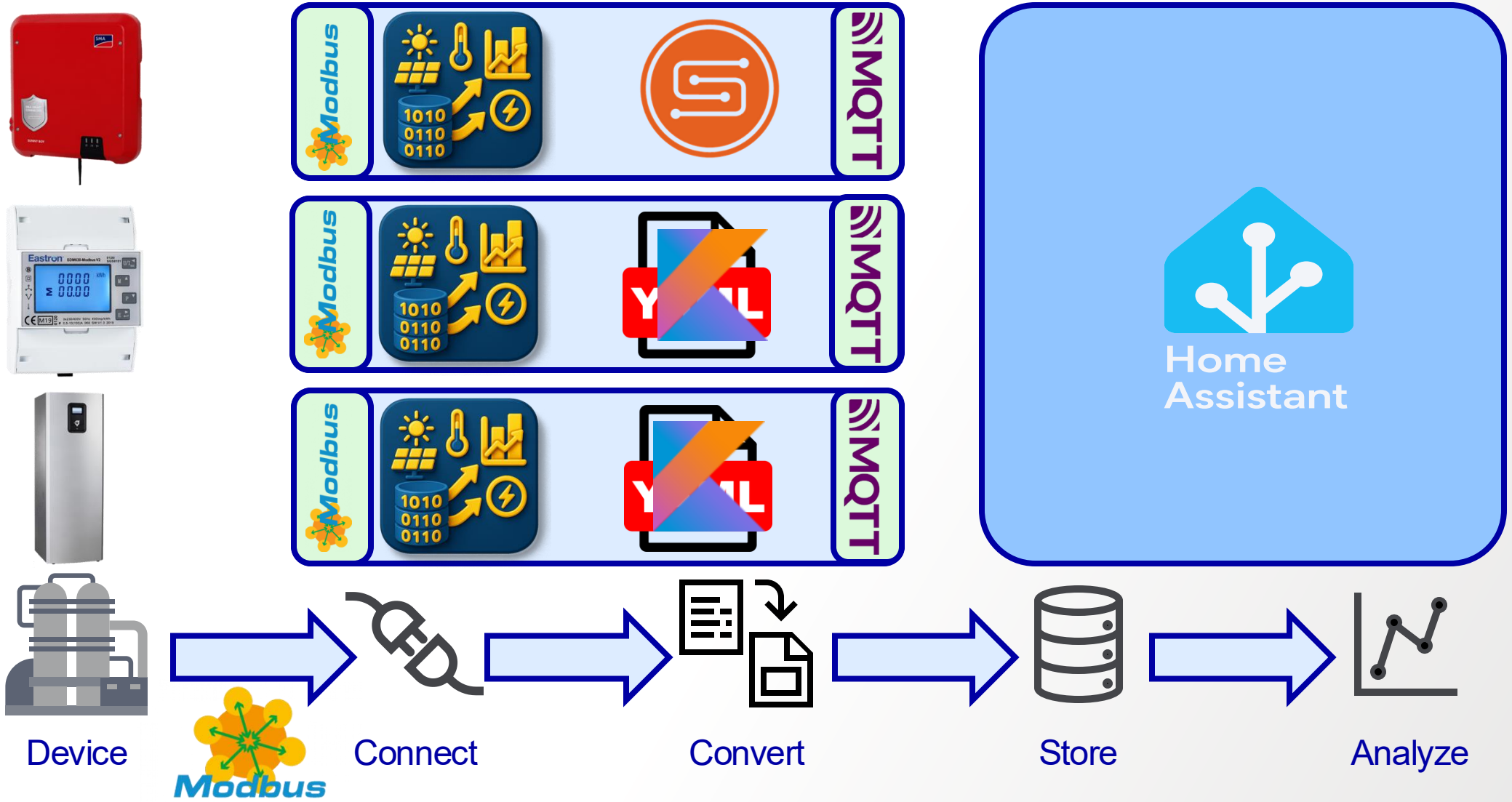
What I have (2025-)



What I have (2025-)



What I have (2025-)



Warmteppomp status

Passive Cooling

Buiten temperatuur
22.35 °C

Binnen temperatuur
21.60 °C

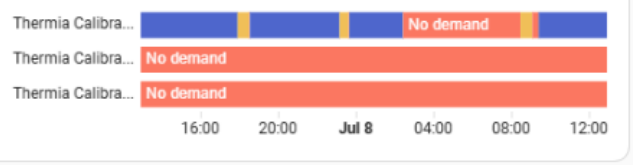
Warm water
55.01 °C

Gewenste binnen temperatuur
20.0 °C

1e Prio Demand
Passive Cooling

2e Prio Demand
No demand

3e Prio Demand
No demand



Brine IN
17.71 °C

Brine OUT
18.76 °C

SunSpec

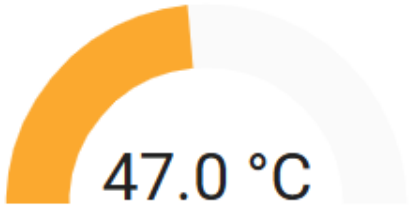
Live data from my SMA Inverter

SUNSPEC ALLIANCE

SMA Inverter

Manufacturer	Model	Version	Serial Number
SMA	SB3.6-1AV-41	4.01.15.R	3005067415

SMA SB3.6-1AV-41 Inverter - AC Power

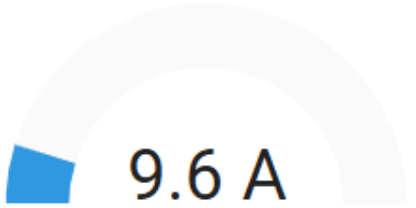


SMA SB3.6-1AV-41 Inverter - Cabinet Temperature

SMA SB3.6-1AV-41 Inverter MPPT - Module[0] - DC Power

SMA SB3.6-1AV-41 Inverter MPPT - Module[1] - DC Power

SMA SB3.6-1AV-41 Inverter - AC Voltage Phase AN



SMA SB3.6-1AV-41 Inverter - AC Current

Summary

- An open Modbus Mapping standard
  - Machine readable (Yaml)
  - That any vendor can create (no PDF)
    - Or generate (SunSpec)
- An open implementation in Kotlin
  - Integrates with InfluxDB, MQTT, GraphQL and more
  - Usable by tech savvy home users
- Anyone can reimplement it in any other language.
  - Python, Go, Rust, C#, …
- A business friendly license
  - Apache 2.0

Thank you!

https://modbus.basjes.nl

Peter Grace

Secure SunSpec Modbus-TCP ► MQTT ► Cloud/Home Assistant.

https://github.com/PeterGrace/sunspec_rs/ (Rust)

https://github.com/PeterGrace/sunspec_gateway (MQTT Gateway)

Who's this random guy?

- In my day job, I'm a "DevOps Engineer"
- Observability topics are my bread and butter
- When I'm not at my day job, one of my hobbies is continuing to do my day job.
- I like metrics, I like data, I like to know and prove things.
- I am an avid HomeAssistant user and wanted solar data in HomeAssistant

I have a lot of solar

System Specs

- On Paper:
  - 60x 350W solaria deck-mounted panels
  - 3x 9.6kW inverters
  - 3x 18kWh battery cabinets
- Specifications are cool, but what about actual production data?

Please stop making poor-performing mobile apps

- My solar equipment manufacturer only provides data to end users via a mobile app, which provides occasionally useful info when it is not malfunctioning
- Self-organized user groups online agree that its practically useless
- My solar provider claimed they were SunSpec certified – could I use that?

SunSpec Modbus compatibility: Claimed.

- Reality: the equipment did in fact provide a sunspec interface, gated behind a firewall inside the inverter unit. Consumers aren't supposed to be able to access it.
  - I think SunSpec says that I actually should be able to, though?

- In some cases, end users may need to void their warranty. Manufacturers: please don't make end-users do that!

SunSpec Modbus TCP: Now what?

- I have access to the sunspec data now, and happen to have the model files since they helpfully put them on the inverter's processing unit as well (thank you!)
- My options at that point: write a pysunspec2-based application, or write a completely new library
- I chose the 'write a new library path' since I was recently excited about the Rust programming language.

sunspec_rs

- Utilizes *tokio-modbus* Rust crate to communicate with SunSpec-compatible device via modbus-rtu or modbus-tcp
- Reads model json files and builds a map of available points, so you can request a specific point.
- Handles all the annoying bitmath and binary to values translations.

- Now that I have a library, how do I get the data into HomeAssistant?

sunspec_gateway

- An app that allows an end-user to configure the datapoints that are relevant to them and query them at an interval that makes sense.
- Uses sunspec_rs to do the heavy modbus lifting
- Sends data to MQTT and will, by default, create the required HomeAssistant discovery messages in the homeassistant topic in MQTT.
- Voila: instant data in homeassistant!

What does this look like?

- Show some live data!
  - Pete's actual instance
  - Testing instance
- Lets add some data points!
  - Find, add, validate
  - Configure presentation of data
- Even better: prometheus->grafana

Things I'd like to improve

- Config files are a very "devopsy" thing, not user friendly
  - Perhaps convert to storing config in existing sqlite database
  - That'd allow changing the config in the web interface
- I'd love feature enhancement requests!

What's helped me along the way?

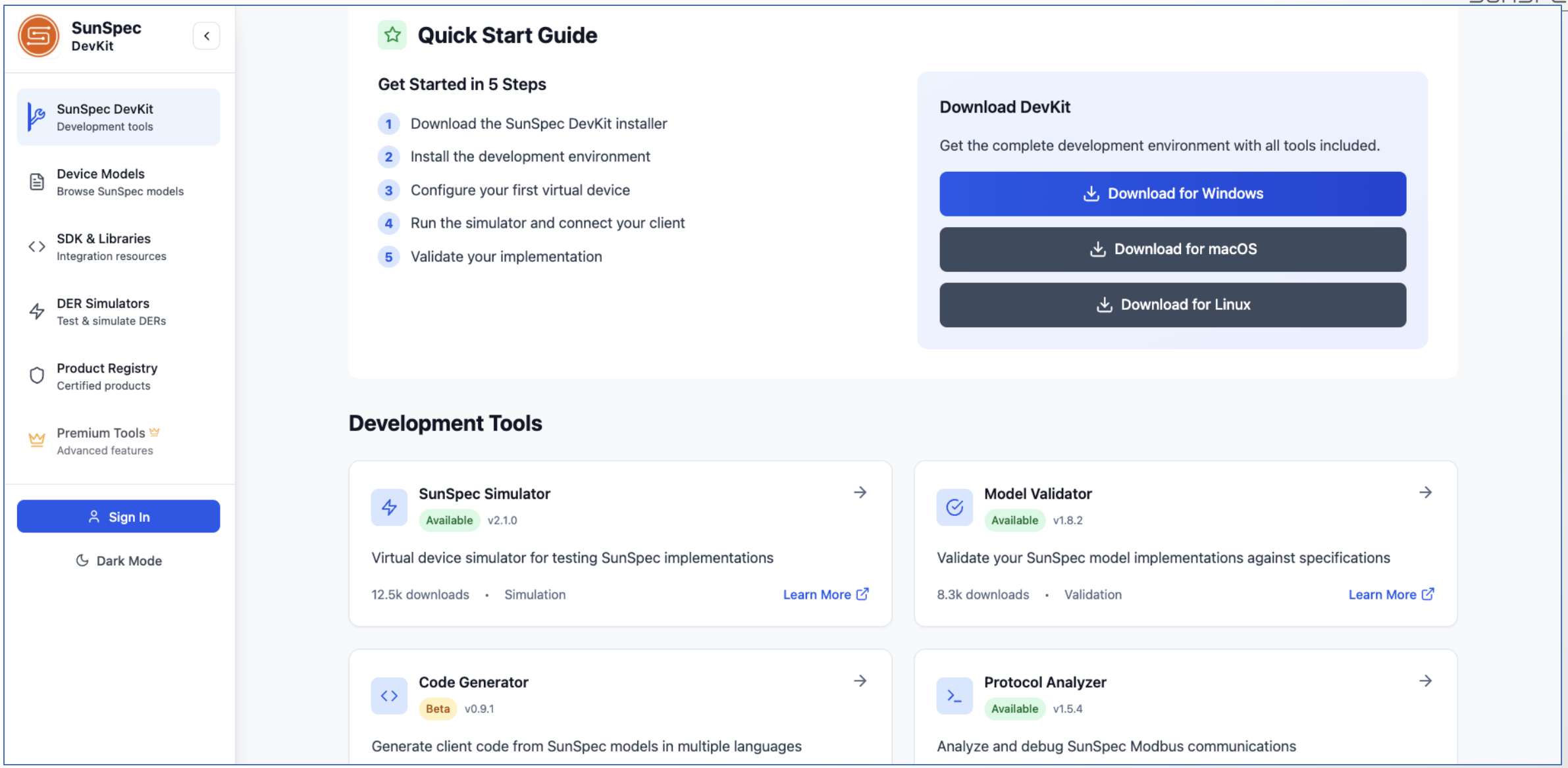
- Special thanks to the SunSpec Alliance for publishing the specification in an open-enough way that I could utilize it!
  - Also, thank you for publishing the models data publicly!

- A big thank-you to DER Security for recently allowing me to utilize their simulator to help test my tool – I solved two major bugs in less than 3 hours after getting the simulator!

More info

- Repos (both permissively-licensed MIT)
  - github.com/PeterGrace/sunspec_rs
  - github.com/PeterGrace/sunspec_gateway
- Docs (work in progress!)
  - https://sunspec-gateway.netlify.app/
- Contact me?
  - software@gfd.dev

SunSpec DevKit : Launching in Fall 2025



Collection of Existing SunSpec Repos

- PySunspec2 : https://github.com/sunspec/pysunspec2
- SunSpec DER Device Models: https://github.com/sunspec/models

SunSpec DevKit

Express Interest

https://forms.gle/qc6U2YnG2aHbK3rP8

Thank you!