# Knowledge Graphs for IoT Platform Digital Twins based on SDF

Alejandro Jarabo Peñas , Dr. Bin Xiao.
In collaboration with other team members

Ericsson Research, IoT & Cyber-Physical Systems Team
Ericsson Nikola Tesla, ETK Research

2022-12-12

# Agenda

**Background and Research Question**

**Results Overview**

**Key Enablers**

**Deep Dive:**

- Test Environment
- Prototype
- Devices Similarity / Knowledge Graph Integration
- Demo?

**Future Work**

**Q&A**

# Background & Focus

**Knowledge Graph:** represents, manages, and reasons with graph-structured knowledge.

**- Knowledge Representation:**

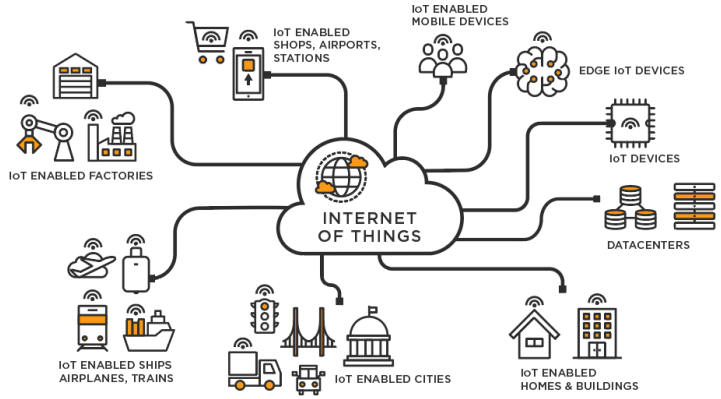
Knowledge Base – Facts + Rules

**- Knowledge Reasoning:**

Non-explicitly stated facts inference

**IoT Platform:**

platform where connected devices implementing sensors/actuators exchange data



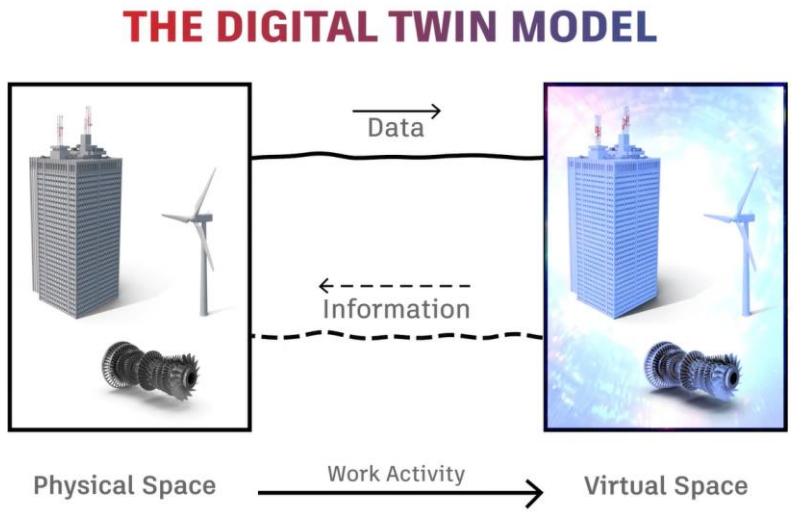

**Digital Twin:** digital model that accurately represents physical assets.

**vs. Emulations:**

- Built **automatically** relying on **real-time** data
- **Bi-directional information flow**: insights from digital copy shared with physical assets

**Physical Assets ->** Real-world objects of interest connected to DT system

**Digital Twin ->** Knowledge Graph presenting devices and their relations



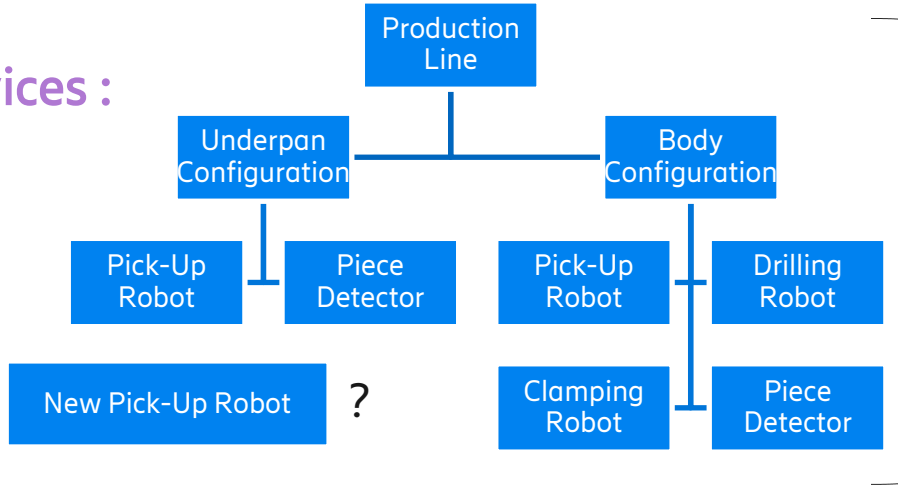
# Research Question

## Knowledge handler for Digital Twin IoT device platform that enables:

- **Storing** knowledge in a knowledge base?
- **Querying** knowledge from a knowledge base?
- Conduct **reasoning** via a reasoning engine?

Built-in Knowledge Graph Functionalities

- Integrate new unforeseen devices :




Main contribution

# Results Overview

## Explore knowledge-storage methods for DT

- Based on **TypeDB** knowledge graph
- Domain-adapted ontology or **schema definition**
- **Data initialization**: initial devices and their connections

## Setup IoT-based testbed for knowledge ingestion

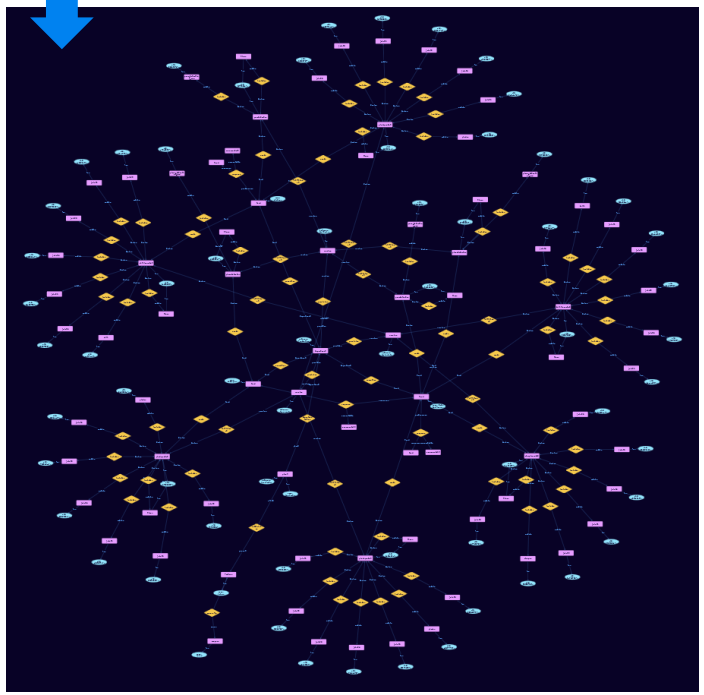
- IoT devices in a **simplified automobile production line**
- Device classes semantic description with **SDF**: modules, attributes...
- **Task-dependent** data generation

## Device data integration into Knowledge Graph

- Device **class definition** in schema
- **Real-time update** of device attributes
- Integration of **unforeseen devices**

Physical assets

Knowledge Handler



Knowledge Graph based Digital Twin

# Enabler 1: TypeDB Knowledge Graph - Schema

**Ontological** description – **Knowledge Graph backbone**

Domain-specific terminology defining objects and how they relate:

- **Types:**

- **Entity:** domain objects
- **Relationship:** n-ary connections based on **roles**
- **Attributes:** values owned by entities/relationships

TypeQL Query

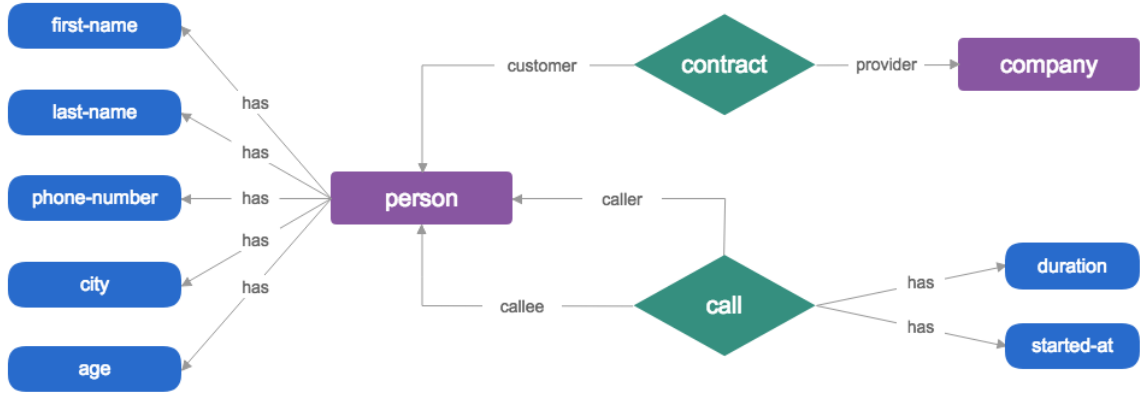
```
define
contract sub relation, relates provider, relates customer;

call sub relation, relates caller, relates callee,
owns started-at, owns duration;

company sub entity, plays contract:provider,
owns name;

person sub entity, plays contract:customer, plays call:caller, plays call:callee,
owns name, owns phone-number, owns city, owns age, owns is-customer;
```

Example schema defined for phone calls KG



# Enabler 1: TypeDB Knowledge Graph – Data

## Entities instances and their relations

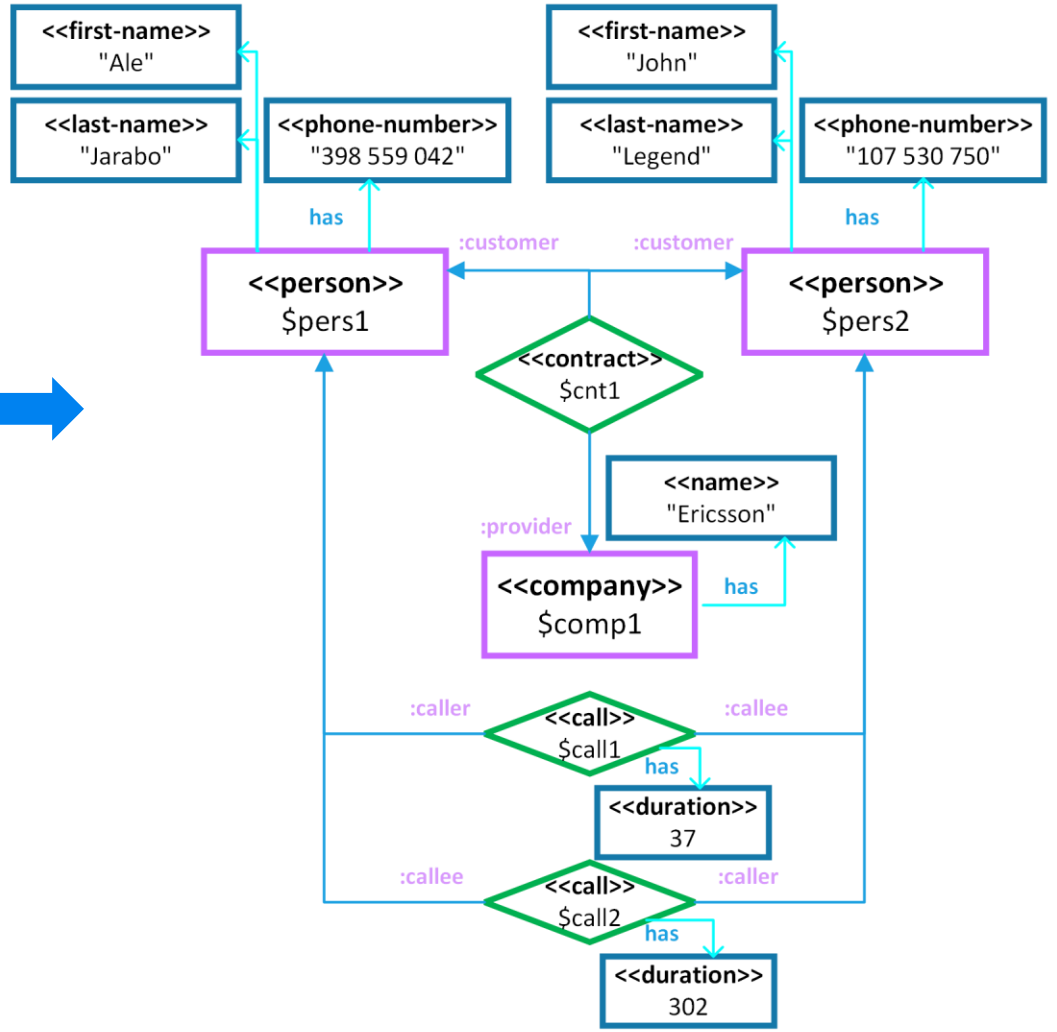
TypeQL Query

```
insert
$comp1 isa company, has name "Ericsson";
$cnt1 (provider: $comp1, customer: $pers1, customer: $pers2) isa contract;

$pers1 isa person,
has first-name "Ale", has last-name "Jarabo", has phone-number "398 559 423“;
$pers2 isa person,
has first-name "John", has last-name "Legend", has phone-number "107 530 750";

$call1 (caller: $pers1, callee: $pers2) isa call, has duration 37;
$call2 (caller: $pers2, callee: $pers1) isa call, has duration 302;
```

Example data inserted into phone calls KG



# Enabler 2: SDF – Semantic Definition Format

## JSON description of IoT Device Class (modules/attributes/value types)

```
{
    "namespace": {"eri": "https://ericsson.com/models"}, "defaultNamespace": "eri",
    "info": {
        "title": "Air Quality", "version": "2022-12-12",
        "copyright": "Copyright 2022 Ericsson. All rights reserved.",
        "license": "BSD-3-Clause"
    },
    "sdfThing": {"AirQuality": {"description": "Monitors air quality through a set of sensors",
      "sdfProperty": {"uuid": {"sdfRef": "#/sdfData/uuid"}},
      "sdfObject": {
           "temp_sensor": {"description": "Measures environmental temperature.",
                "sdfProperty": {
                    "temperature": {"description": "Temperature value",
                        "type": "number",
                        "unit": "Cel"
                    }
                }
           },
           "humidity_sensor": {"description": "Measures environmental humidity.",
                  ...
           },
           "pressure_sensor": {"description": "Measures environmental pressure.",
                  ...
           },
           "air_quality_sensor": {"description": "Meas
                  ...
           }
}}}}
```

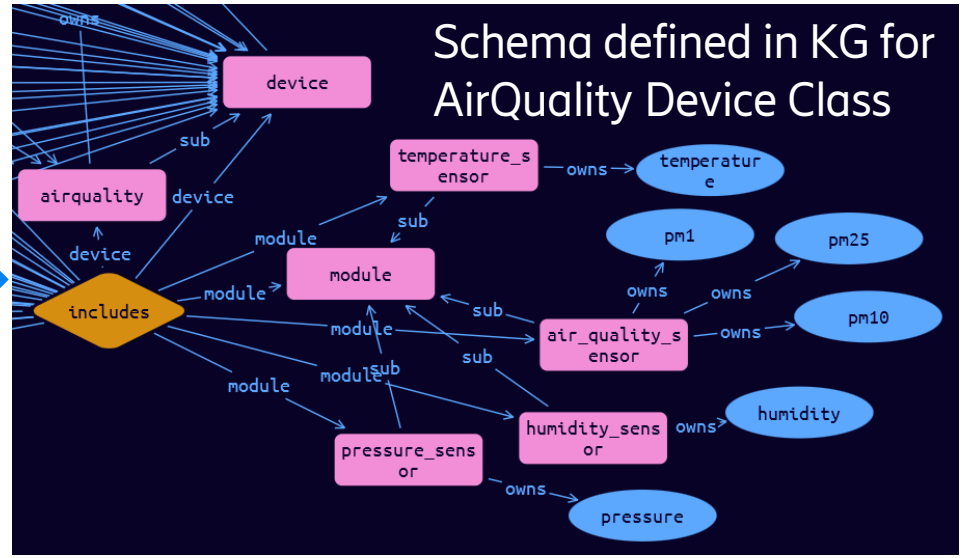
SDF AirQuality Device Class Description

```
define
temperature sub attribute, value double;
humidity sub attribute, value double;
pressure sub attribute, value double;
...
temp_sensor sub module, owns temperature;
humidity_sensor sub module, owns humidity;
pressure_sensor sub module, owns pressure;
...
```

Understand **devices** as **instances** of **classes**

**UUID property** for instance identification

Accessed **if class not yet defined** in schema



# Deep Dive

**Test Environment**: IoT devices in a car-manufacturing plant

**Prototype:**

- **Key Components** description
- **Data Flow**: procedural descriptions around data and knowledge graph

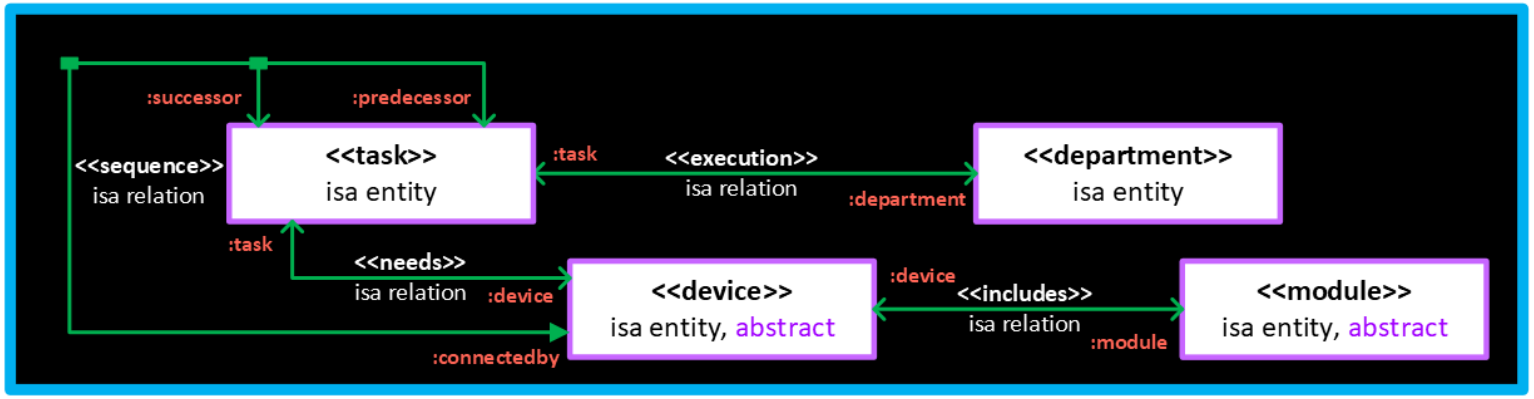
**Devices Similarity**

- **Class Similarity**
- **Behavioral Similarity**

# Environment – Car manufacturing plant divided in tasks

## Knowledge Graph Schema

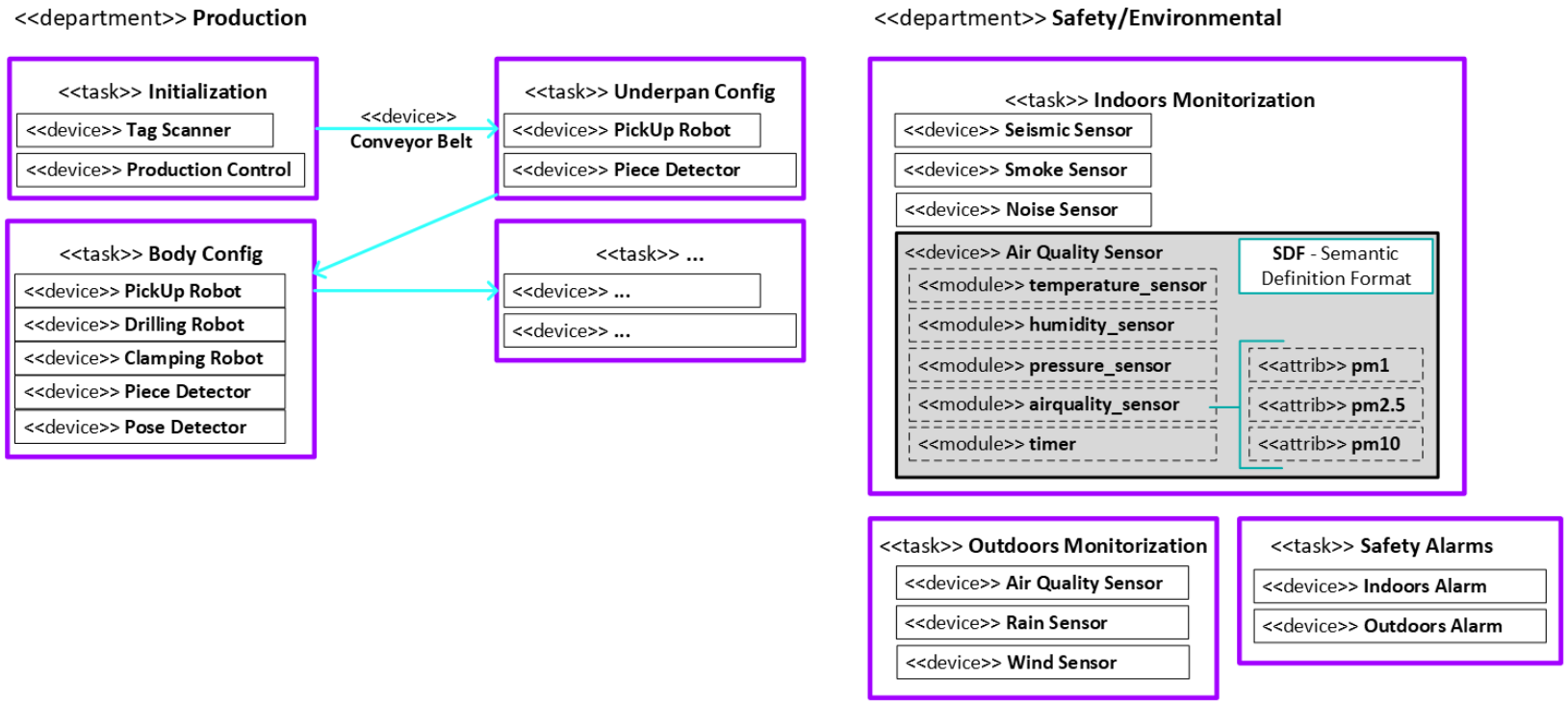
Domain-specific objects
and their relations




+

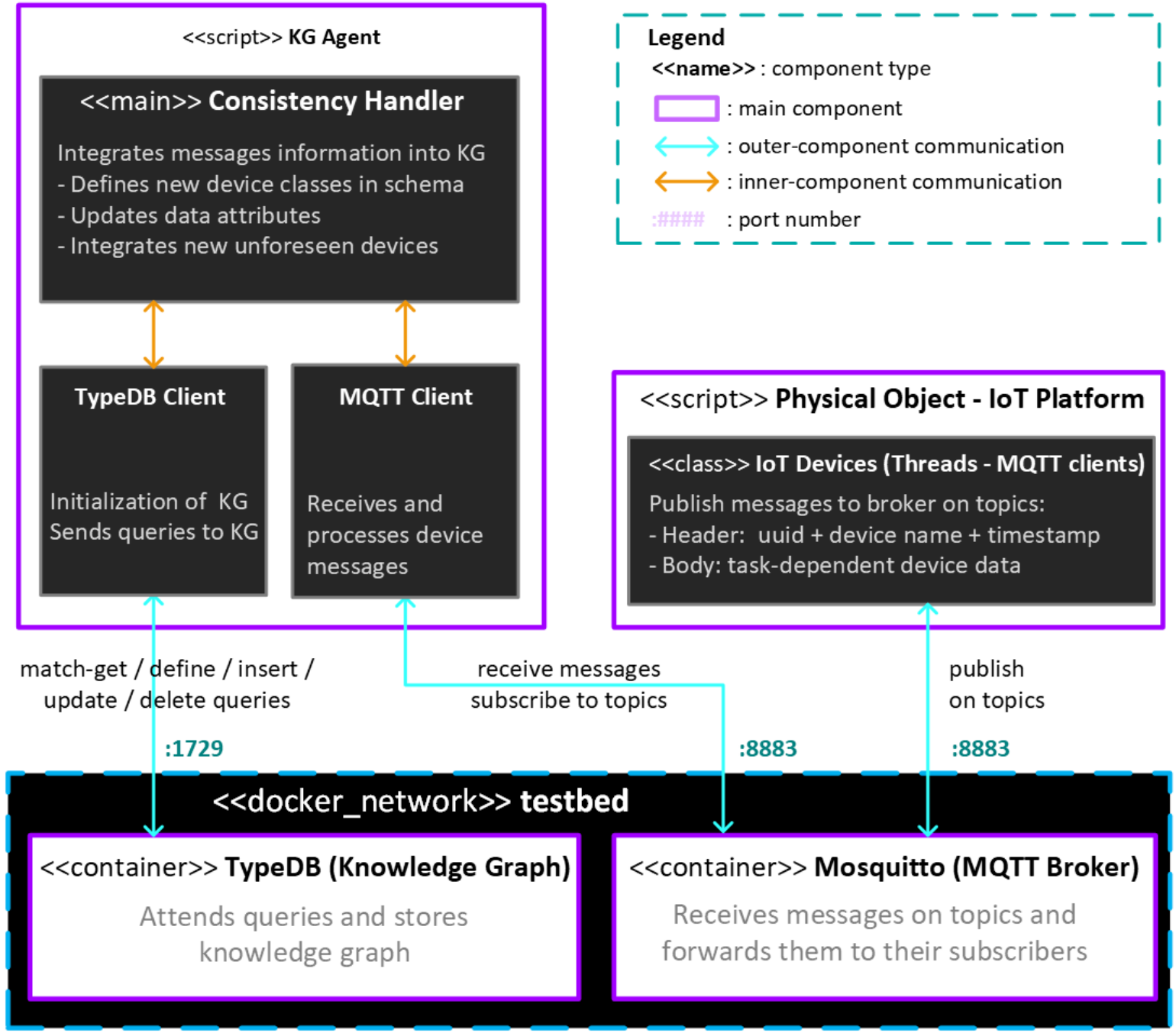
## Initialized IoT Platform

Initial tasks and their
devices adapted to schema



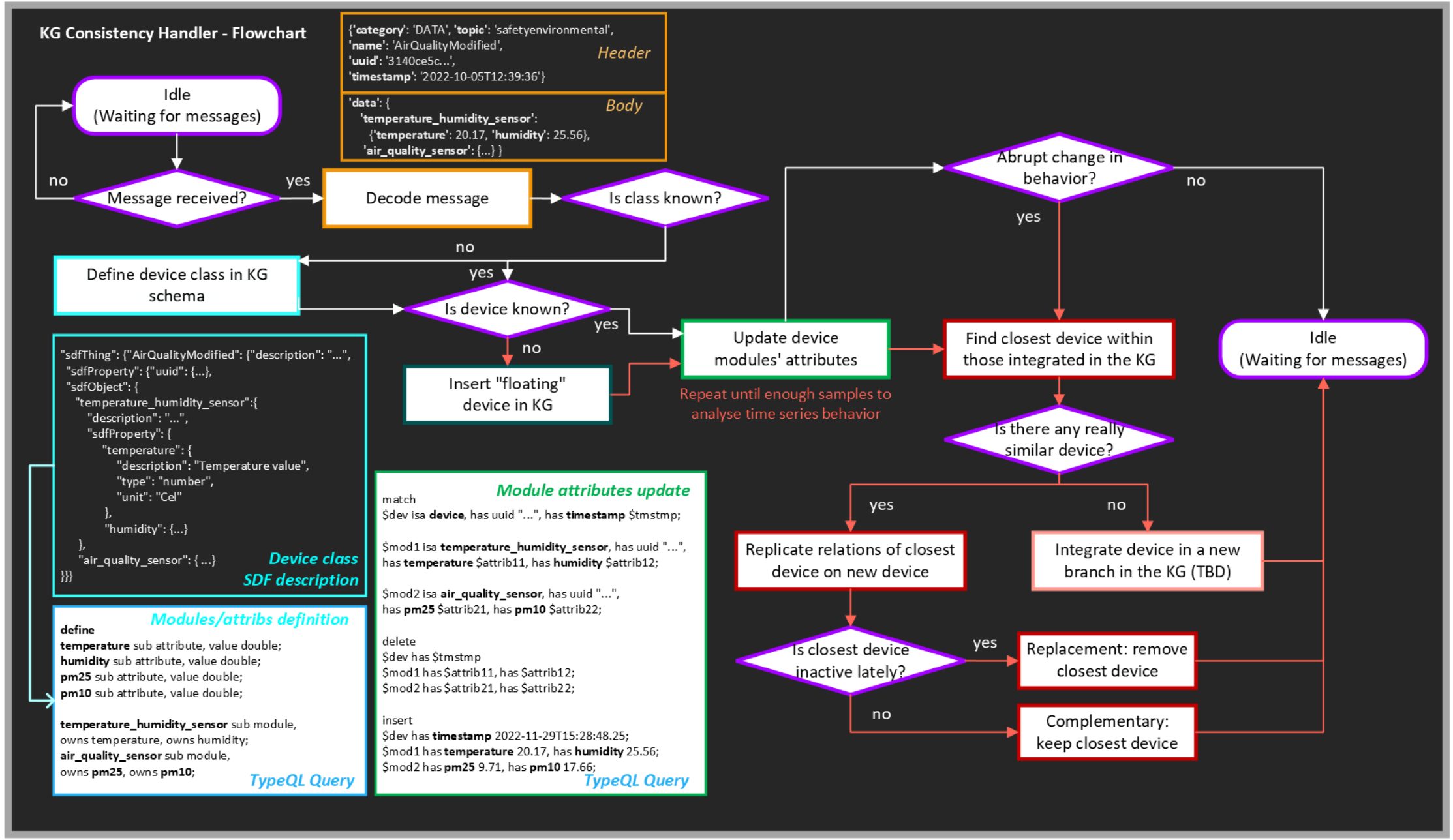
# Key Components

Description of solution components and their interaction



# Flow Chart

Procedural descriptions around data and KG



# Device Classes Similarity – SDF Relation Inference

SDF class descriptions turned into tables - eases analysis but adds redundancy

| sdfThing (Class) | Class Desc | sdfObject (Module) | Module Desc | sdfProperty (Attribute) | Attribute Description |
|---|---|---|---|---|---|
| **Air Quality Simplified** | Monitors air quality through a set of sensors | **Temperature Humidity Sensor** | Measures environmental temperature and humidity | **Temperature** | Temperature value |
| | | | | **Humidity** | Humidity value |
| | | **Air Quality Sensor** | Measures air pollutants. | **PM25** | PM2.5 value |
| | | | | **PM10** | PM10 value |

**Problems**

Compute string values distance? -> **Natural-Language Processing** analysis

Compare device classes with different # of properties? -> **Row-wise voting**

# Device Classes Similarity – SDF Relation Inference

## Compare device classes with different # of properties – **Row-wise voting**

**Algorithm**

Compare new class row to all other classes rows based on NLP

Class of closest row gets a vote

**Closest class = Class with most votes**

Integrate inference as sdfRelation into SDF desc

```
"sdfThing": {"AirQualitySimplified": {
  "description": "…", "sdfObject": {...},
  "sdfRelation": {
      "sameAs": {
          "relType": "exont:same-as",
          "target": "saref:AirQuality"
      }
  }
}
```

SDF AirQualitySimplified Device Class Description

New class (to be integrated)

| sdfThing (Class) | sdfObject (Module) | Module Desc | sdfProperty (Attribute) | Attribute Description |
|---|---|---|---|---|
| **Air Quality Simplified** | **temperature_ humidity_sensor** | Measures environmental temperature and humidity | **temperature** | Temperature value |

Already integrated classes

| | | | | |
|---|---|---|---|---|
| **Air Quality** | **temperature_sensor** | Measures environmental temperature. | **temperature** | Temperature value |
| | **…** | … | **…** | … |
| **Rain Sensor** | **rain_sensor** | Measures the amount of rain that is falling. | **cumulativedepth** | Cumulative rain fall depth. |
| **Pick Up Robot** | **joint** | Joint position and orientation. | **x_position** | Joint position (X) |
| | **…** | … | **…** | … |

# Device Entities - Behavioral Similarity

Compare device entities – **Row-wise voting** based on **time series pattern matching**

Only compared to **related classes** entities

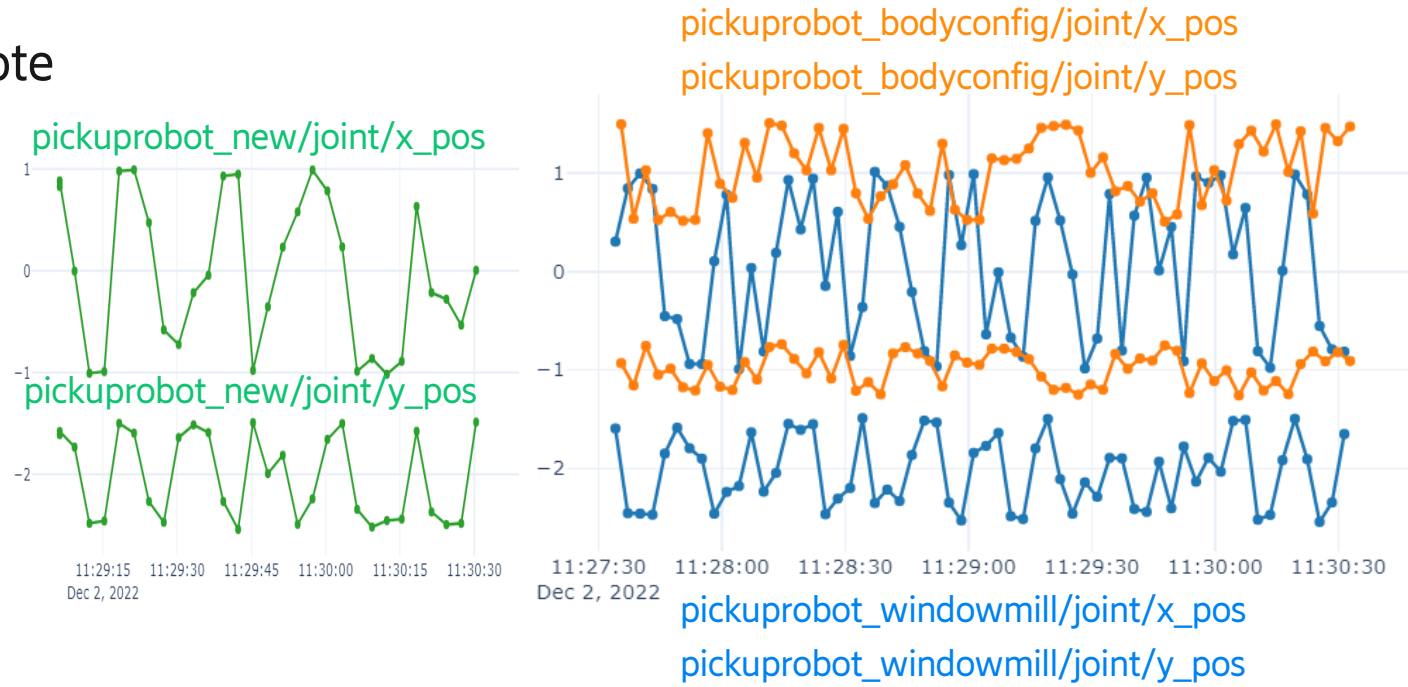
## Algorithm

Get time series window corresponding to new device attribute

Compare sliding window along all other devices attribute time series

Device (UUID) with closest match gets a vote

**Closest device = Device with most votes**

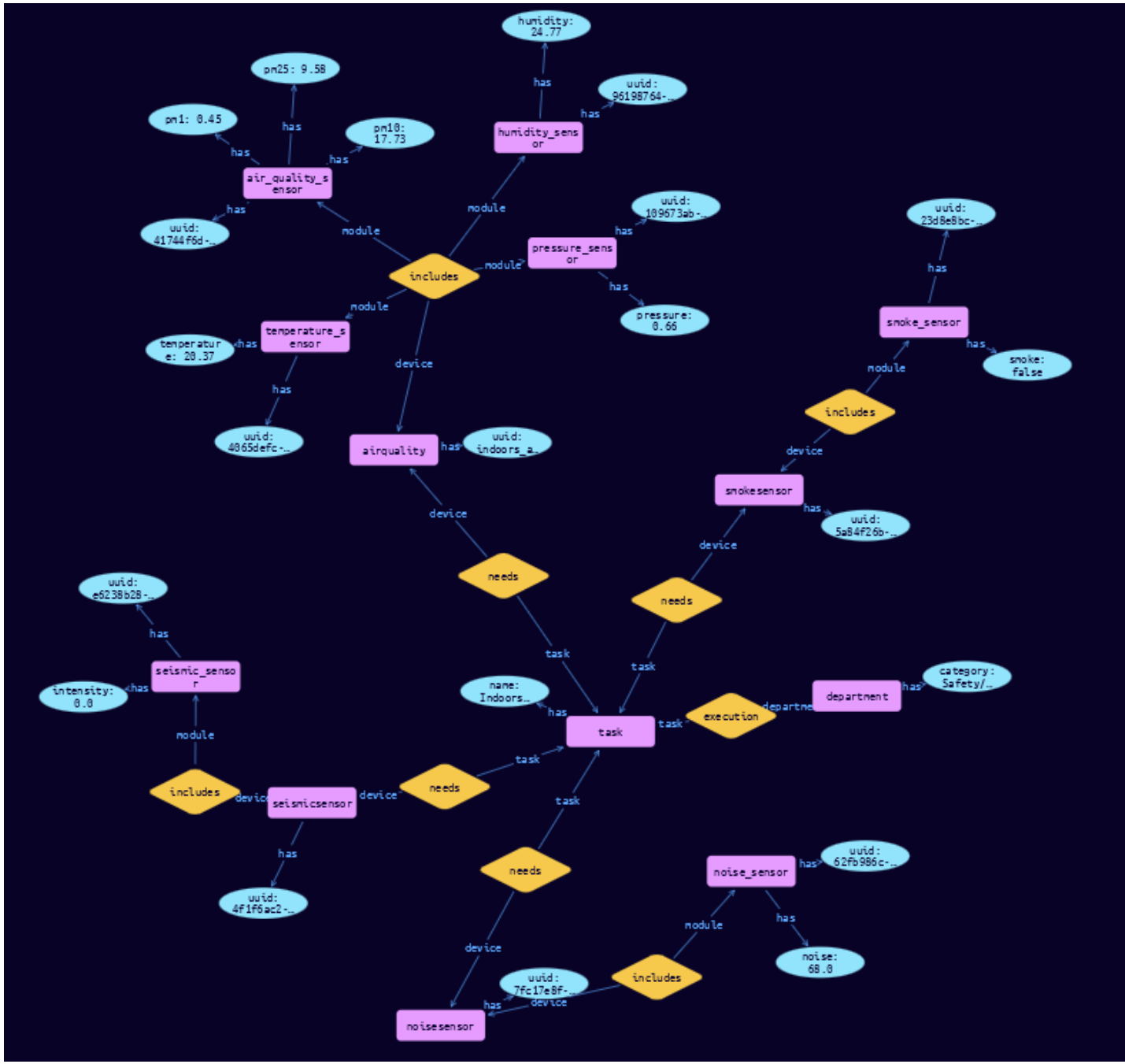


# Demo - Running Steps

1. **Knowledge Graph** initialization
   - Initial schema and data

2. **IoT Platform** emulation
   - Tasks and devices involved
   - Messages visualization
   - Device replacement situation

3. **Knowledge Handler**
   - Definition of device classes
   - Attributes update in real-time
   - Handle new devices



# Future Work

Automatic SDF generation from device specifications

Optimization of closest devices computation (class and behavioral distance)

Integration from scratch – No similar devices / classes found

Test with real data from an actual IoT platform

# Q&A

# Appendix

# OS Knowledge Graphs – Comparison & Choice

| | Storage | Querying | Reasoning | Other relevant characteristics |
|---|---|---|---|---|
| Neo4j Comm Ed (Property Graph) | Horizontal scaling | Cypher<br>• Pattern matching<br>• Navigation | Hierarchies<br>Increased capabilities if exported to RDF format | ACID transactions<br>Large community<br>Native Python support<br>GPL v3 |
| TypeDB (Hypergraph) | Horizontal scaling<br>Decentralized | TypeQL<br>• Pattern matching | Hierarchies<br>**Complex rules**<br>**Query enhancement**<br>**Validation** | Limited **ACID transactions**<br>**Native Python support**<br>GPL v3 |
| Apache Jena (RDF) | Horizontal scaling | SPARQL<br>• Pattern matching<br>• Navigation | Hierarchies<br>Complex rules<br>Forward/backward chaining<br>Validation… | ACID transactions<br>Apache License v2 |