# Sensors and Linked Building Data

**SSoLDAC2023**

Alex Donkers

a.j.a.donkers@tue.nl

# Alex Donkers

PhD Candidate
Information Systems in the Built Environment
Eindhoven University of Technology
Semantic Digital Twins

W3C Linked Building Data Community Group

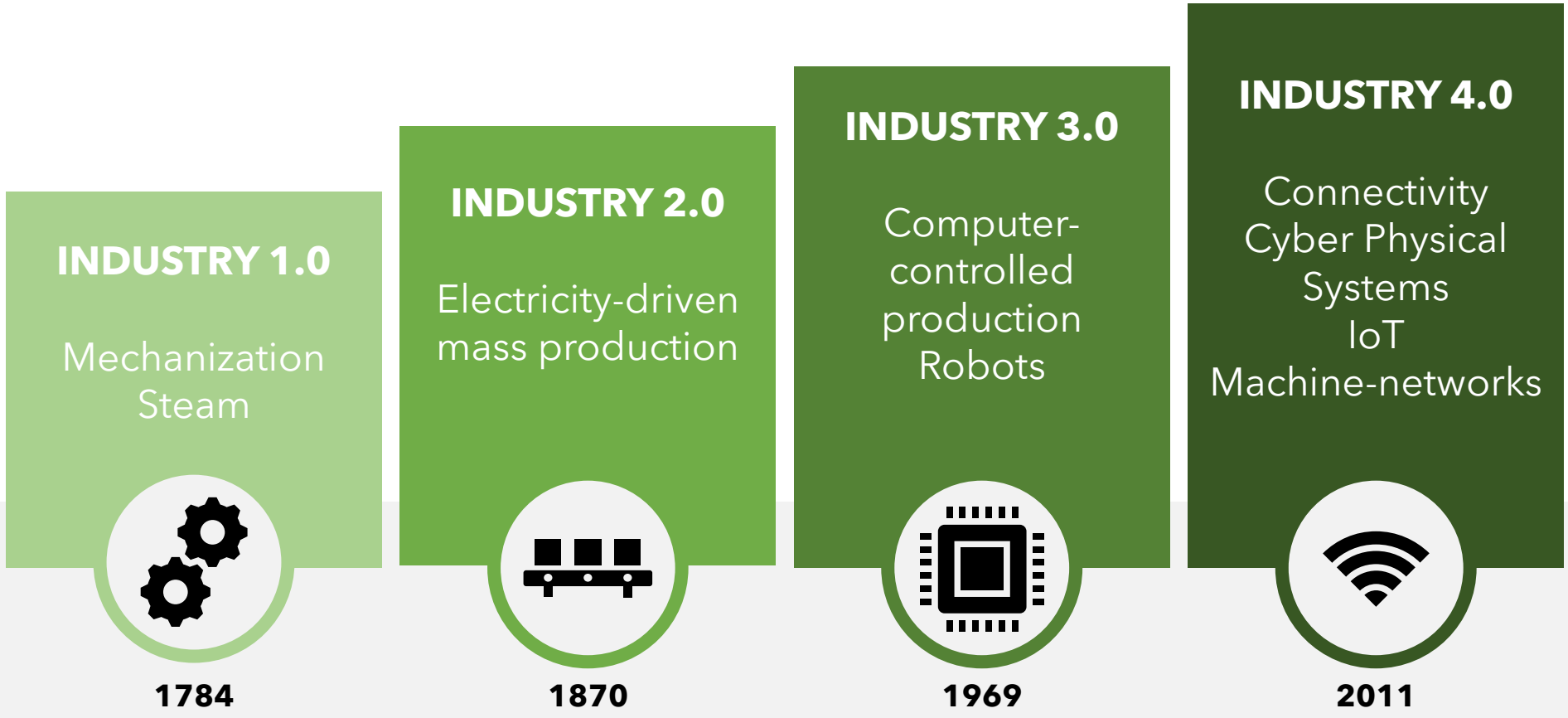
# Content

- What is a sensor? What is time-series data?
- How to semantically represent a sensor?
- How to integrate sensor data with building information?
- How can we use this in different lifecycle stages?
- Hands-on showcase

Part 1

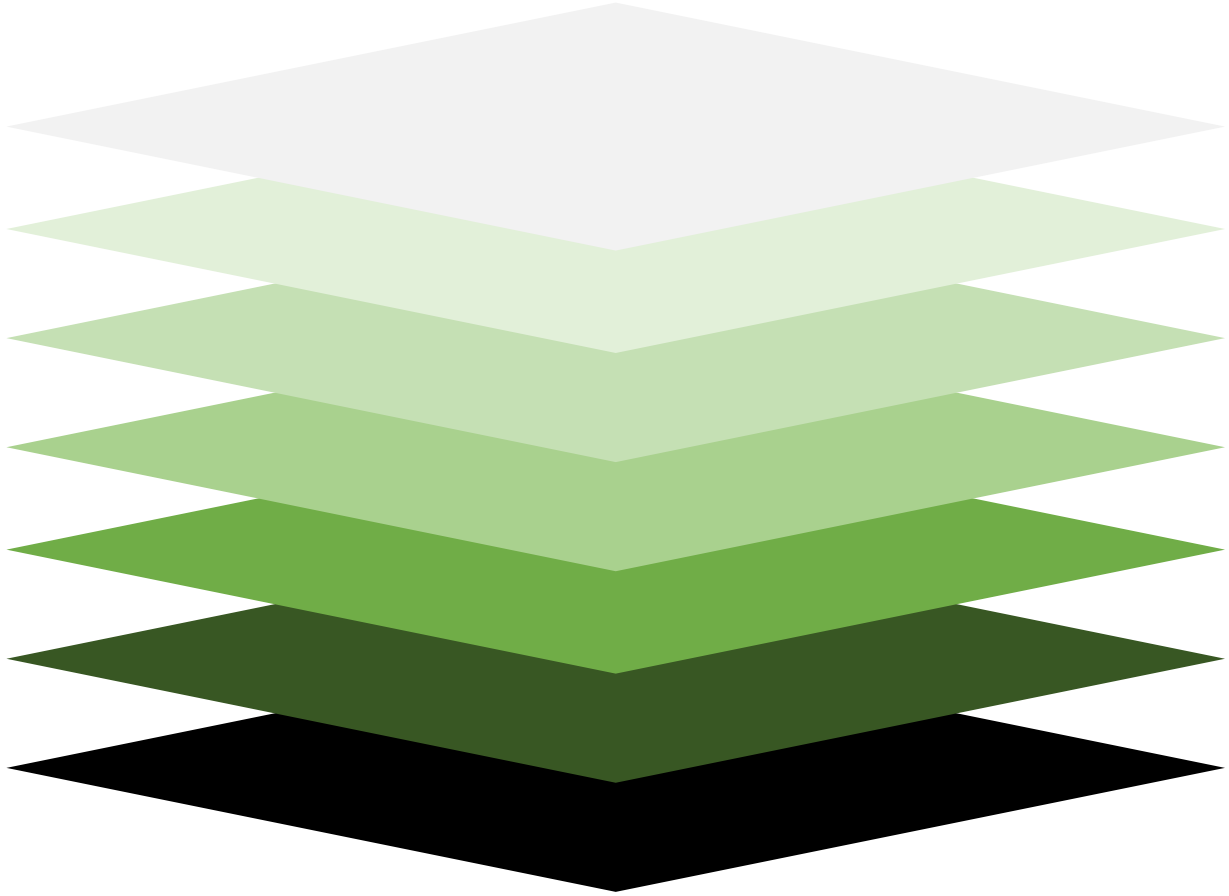
# Internet of Things

# Industry 4.0


INDUSTRY 1.0
Mechanization
Steam
1784
INDUSTRY 2.0
Electricity-driven
mass production
1870
INDUSTRY 3.0
Computer-
controlled
production
Robots
1969
INDUSTRY 4.0
Connectivity
Cyber Physical
Systems
IoT
Machine-networks
2011

# Internet of Things

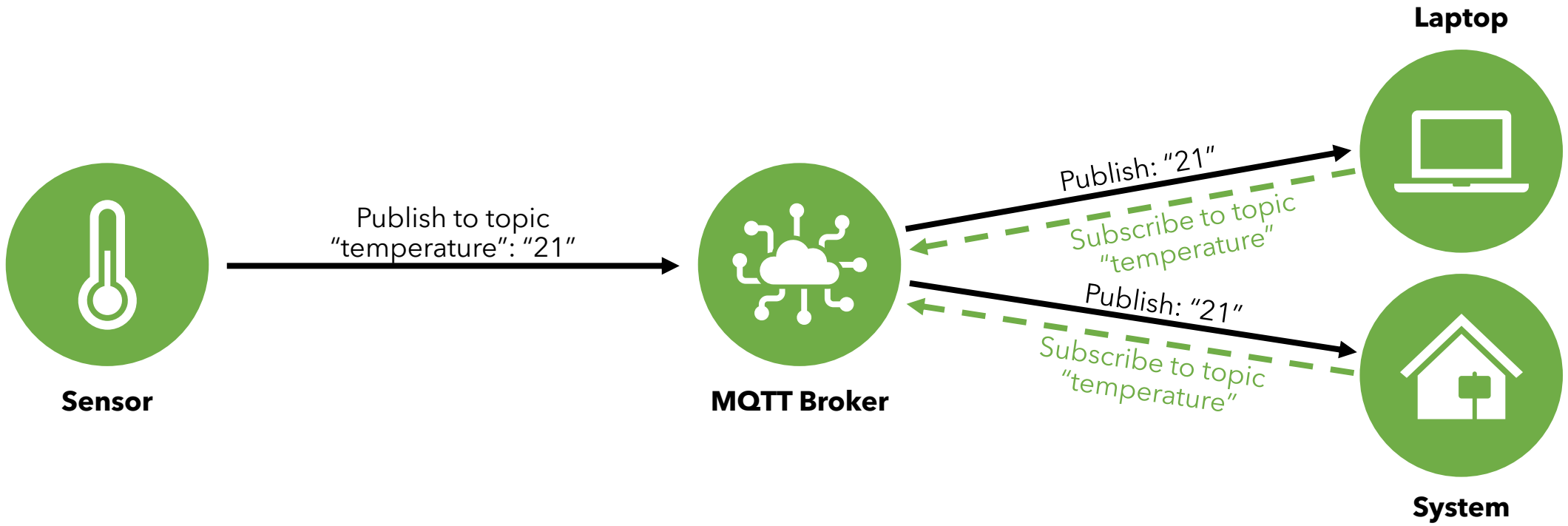
Processes
Applications
Data abstraction
Data accumulation
Edge computing
Connectivity
Appliances



# Various protocols

## MQTT

Message Queuing Telemetry Transport



# Various protocols

MQTT

HTTP

Websocket

DDS

…

# Time series data

```
S351,21,80,1630575705
S351,21,78,1630575706
S351,21,,1630575709
S349,21,80,480,1630575710
S351,22,80,1630575713
S351,22,81,1630575714
S351,22,81,1630575717
S349,22,78,492,1630575717
S351,21,80,1630575718
S351,21,78,1630575721
S351,22,77,1630575723
S349,23,74,1630575724
S351,22,,1630575727
S351,23,74,1630575729
S351,23,,1630575730
S349,,80,1630575730
S349,22,81,1630575732
S370,True,1630575733
S351,21,83,1630575735
S351,23,82,1630575736
S349,23,80,495,1630575738
S351,24,82,1630575738
S351,24,82,1630575739
S351,24,,1630575741
S349,,78,501,1630575742
S351,24,80,1630575742
S351,23,78,1630575744
S351,23,78,1630575744
S349,23,77,1630575749
S351,23,74,1630575760
S351,,73,1630575762
S351,23,,1630575763
S349,18,72,1630575766
S349,18,72,1630575768
S351,21,80,1630575769
S351,21,78,1630575772
S351,21,,1630575773
S349,21,80,480,1630575774
S351,22,80,1630575777
S370,True,1630575778
S351,22,81,1630575779
S349,22,78,492,1630575780
S351,22,80,1630575782
S351,22,78,1630575783
S351,23,77,1630575785
S349,25,74,1630575786
S351,25,,1630575788
S351,23,74,1630575789
S370,False,1630575793
S349,,80,1630575795
S349,22,81,502,1630575796
S351,21,80,1630575800
S351,21,78,1630575803
S351,21,,1630575804
S349,21,80,480,1630575805
S351,22,80,1630575805
S351,22,81,1630575808
S351,22,81,1630575810
S349,22,78,492,1630575812
S351,21,80,1630575813
S351,21,78,1630575815
S351,22,77,1630575816
S349,23,74,1630575824
S351,22,,1630575824
S351,23,74,1630575826
S351,23,,1630575827
S349,,80,1630575829
S349,22,81,1630575832
```

Volume
Velocity
Variety

# Compare this to IFC

```
DATA;
#1= IFCPERSON('CONNECT365\\1990jama','Undefined',$,$,$,$,$,$);
#2= IFCORGANIZATION($,'Trimble Solutions Corporation',$,$,$);
#3= IFCPERSONANDORGANIZATION(#1,#2,$);
#4= IFCAPPLICATION(#2,'2019i Service Pack 2','Tekla Structures','Multi material modeling');
#5= IFCOWNERHISTORY(#3,#4,$,.NOCHANGE.,$,$,$,1574669477);
#6= IFCCARTESIANPOINT((0.,0.,0.));
#7= IFCDIRECTION((1.,0.,0.));
#8= IFCDIRECTION((0.,1.,0.));
#9= IFCDIRECTION((0.,0.,1.));
#10= IFCAXIS2PLACEMENT3D(#6,#9,#7);
#11= IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,1.E-05,#10,$);
#12= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,*,*,#11,$,.MODEL_VIEW.,$);
#13= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Axis','Model',*,*,*,*,#11,$,.GRAPH_VIEW.,$);
#14= IFCGEOMETRICREPRESENTATIONSUBCONTEXT('FootPrint','Model',*,*,*,*,#11,$,.MODEL_VIEW.,$);
#15= IFCSIUNIT(*,.LENGTHUNIT.,.MILLI.,.METRE.);
#16= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(304.8),#15);
#17= IFCDIMENSIONALEXPONENTS(1,0,0,0,0,0,0);
#18= IFCCONVERSIONBASEDUNIT(#17,.LENGTHUNIT.,'FOOT',#16);
#19= IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#20= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.09290304),#19);
#21= IFCDIMENSIONALEXPONENTS(2,0,0,0,0,0,0);
#22= IFCCONVERSIONBASEDUNIT(#21,.AREAUNIT.,'SQUARE FOOT',#20);
#23= IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#24= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.028316846592),#23);
#25= IFCDIMENSIONALEXPONENTS(3,0,0,0,0,0,0);
#26= IFCCONVERSIONBASEDUNIT(#25,.VOLUMEUNIT.,'CUBIC FOOT',#24);
#27= IFCSIUNIT(*,.MASSUNIT.,.KILO.,.GRAM.);
#28= IFCSIUNIT(*,.TIMEUNIT.,$,.SECOND.);
#29= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#30= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.0174532925199433),#29);
#31= IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#32= IFCCONVERSIONBASEDUNIT(#31,.PLANEANGLEUNIT.,'DEGREE',#30);
#33= IFCSIUNIT(*,.SOLIDANGLEUNIT.,$,.STERADIAN.);
#34= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,.DEGREE_CELSIUS.);
#35= IFCSIUNIT(*,.LUMINOUSINTENSITYUNIT.,$,.LUMEN.);
#36= IFCUNITASSIGNMENT((#15,#19,#23,#27,#28,#29,#33,#34,#35));
#37= IFCPROJECT('3vQCgMnSj2$h1msSZB5TdK',#5,'Undefined',$,$,$,$,(#11),#36);
#38= IFCLOCALPLACEMENT($,#10);
#39= IFCSITE('1ijI259fXFn8wEPcpLaAC3',#5,'Undefined',$,$,#38,$,$,.ELEMENT.,$,$,0.,$,$);
```

# Different ways to consume the data

Directly, via these protocols
MQTT, Kafka, …

Indirectly, via (time-series) databases
InfluxDB
TimescaleDB
MongoDB
MySQL

# Common structure, formats, …? No

## InfluxDB Line protocol syntax

<measurement>[,<tag_key>=<tag_value>[,<tag_key>=<tag_value>]] <field_key>=<field_value>[,<field_key>=<field_value>] [<timestamp>]

## Example

materaTemperature,measuredBy=JohnDoe,location=cte temperature=“25" 1556813561098000000

# Common structure, formats, …? No

**MongoDB time series collection**

```
db.materaTemperature.insertMany( [
  {
    "metadata": { “measuredBy": “JohnDoe”, “location”: “cte”, "type": "temperature" },
    "timestamp": ISODate("2019-06-14T05:32:49.000Z"),
    "temperature": 12
  },
```

# Common structure, formats, …? No

**InfluxDB time series data querying**

```
data = from(bucket: “materaWeather")
    |> range(start: -1h)
    |> filter(fn: (r) => r._measurement == “materaTemperature"
and r._field == “temperature")
```

# Common structure, formats, …? No

**MongoDB time series data querying**

```
db.weather.findOne({
   "timestamp": ISODate("2021-05-18T00:00:00.000Z")
})
```

# This makes sense…

**Different databases are good in different things**

- × Querying
- × Storage
- × Semantics
- × Others (retention policies, mathematical expressions, …)

To do this, they need different data structures

# … but data integration is more complex

Part 2

# Semantics of sensors

# Sensor ontologies

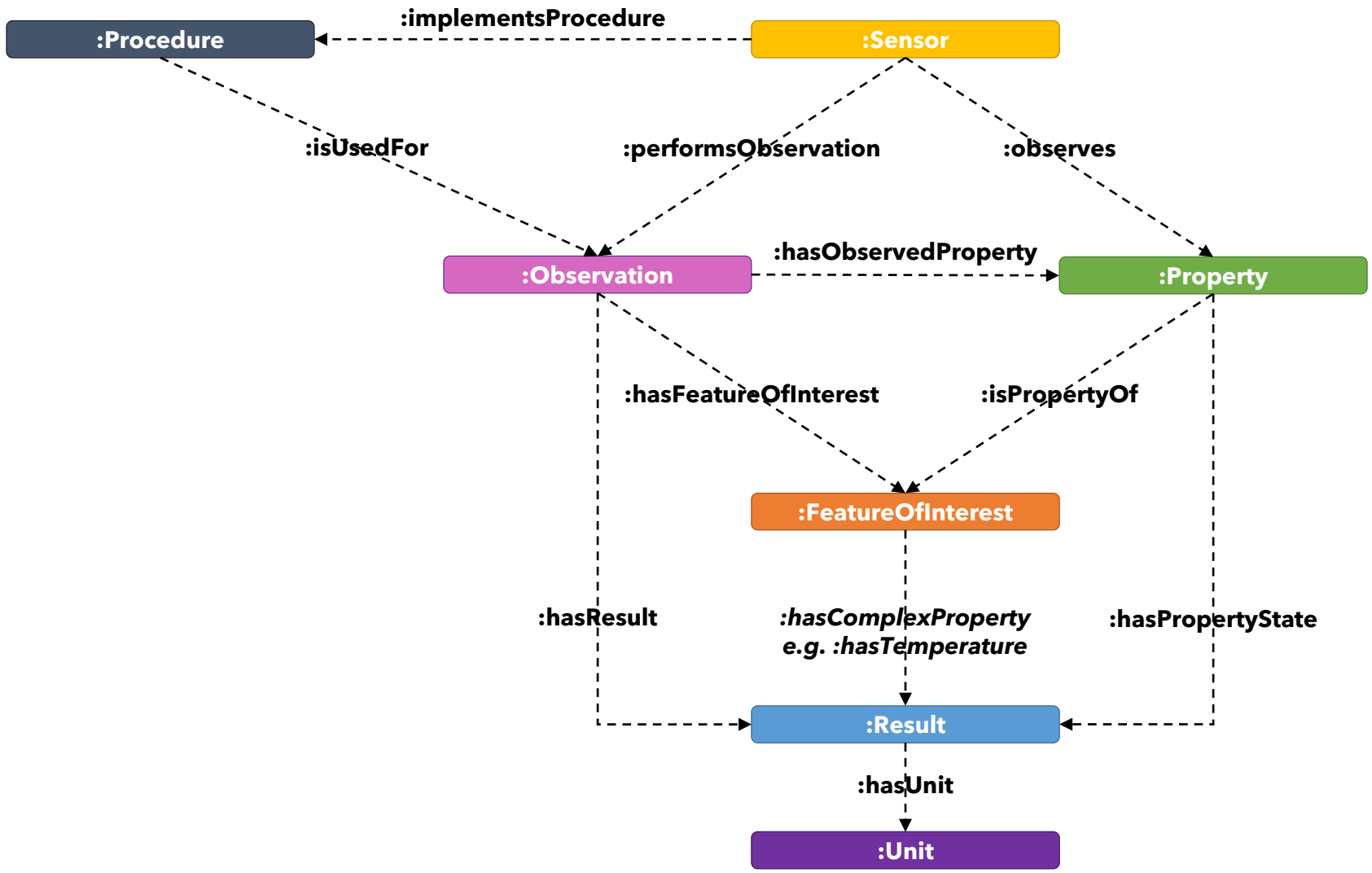
- Many authors created ontologies to represent sensors and actuators.

- SOSA/SSN
- BOP
- SAREF
- SEAS
- Brick
- And more in https://doi.org/10.3390/buildings12101522

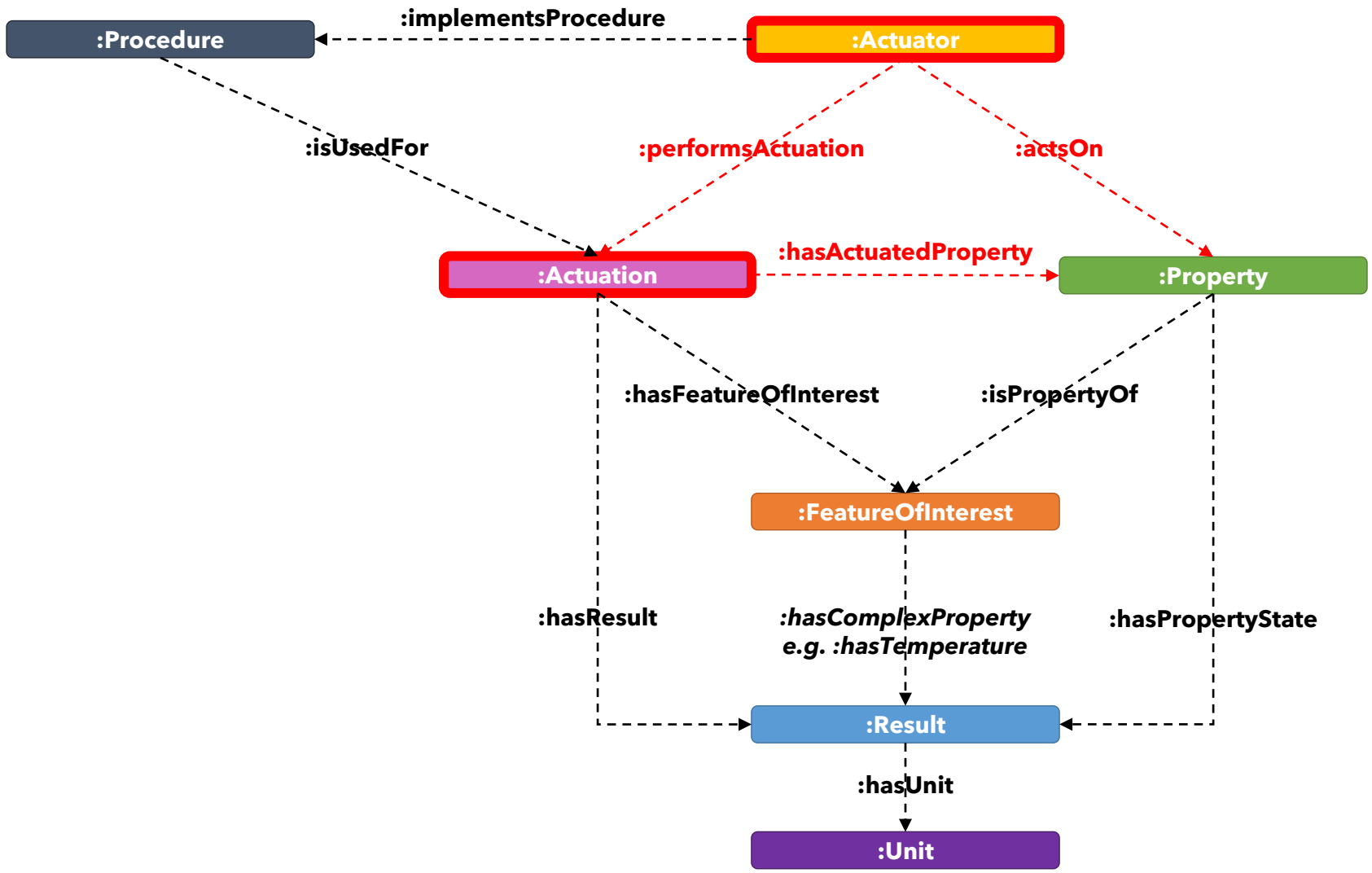
# Common semantic concepts

- :Sensor — Something that can measure values
- :FeatureOfInterest — A real-world phenomenon, a “thing”
- :Property — A measurable characteristic of a feature of interest
- :Observation — The act of observing the state of a property
- :Result — The outcome of the observation
- :Unit — The unit of measure
- :Procedure — A workflow, protocol or plan specifying how to perform an observation

# Ontology design pattern

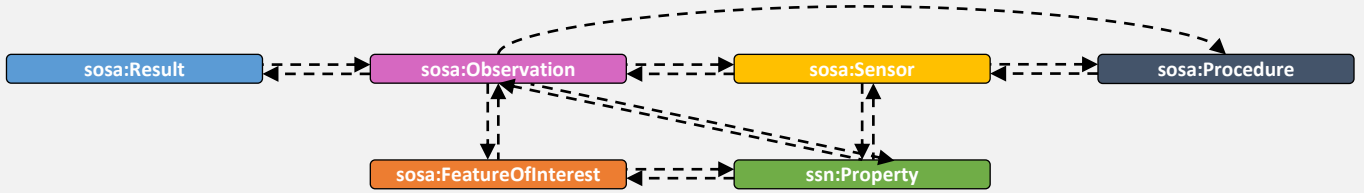


# Actuators follow a similar pattern

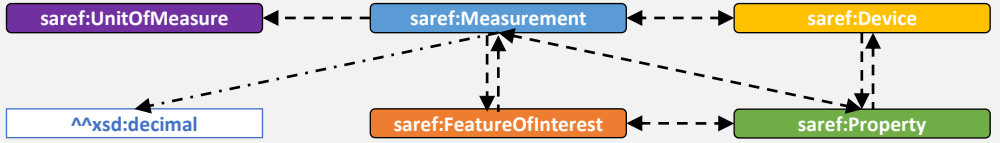


# Sensor ontologies

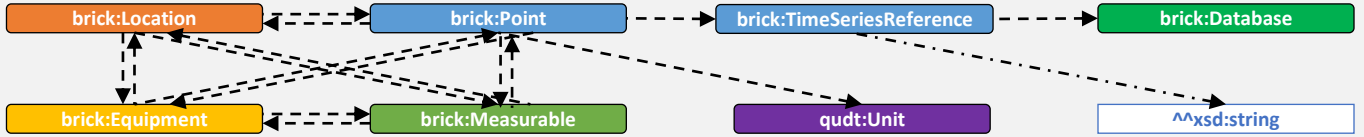
- SOSA/SSN



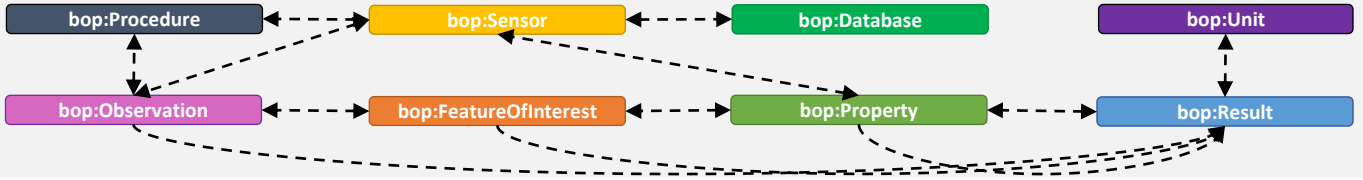

- SAREF




- Brick

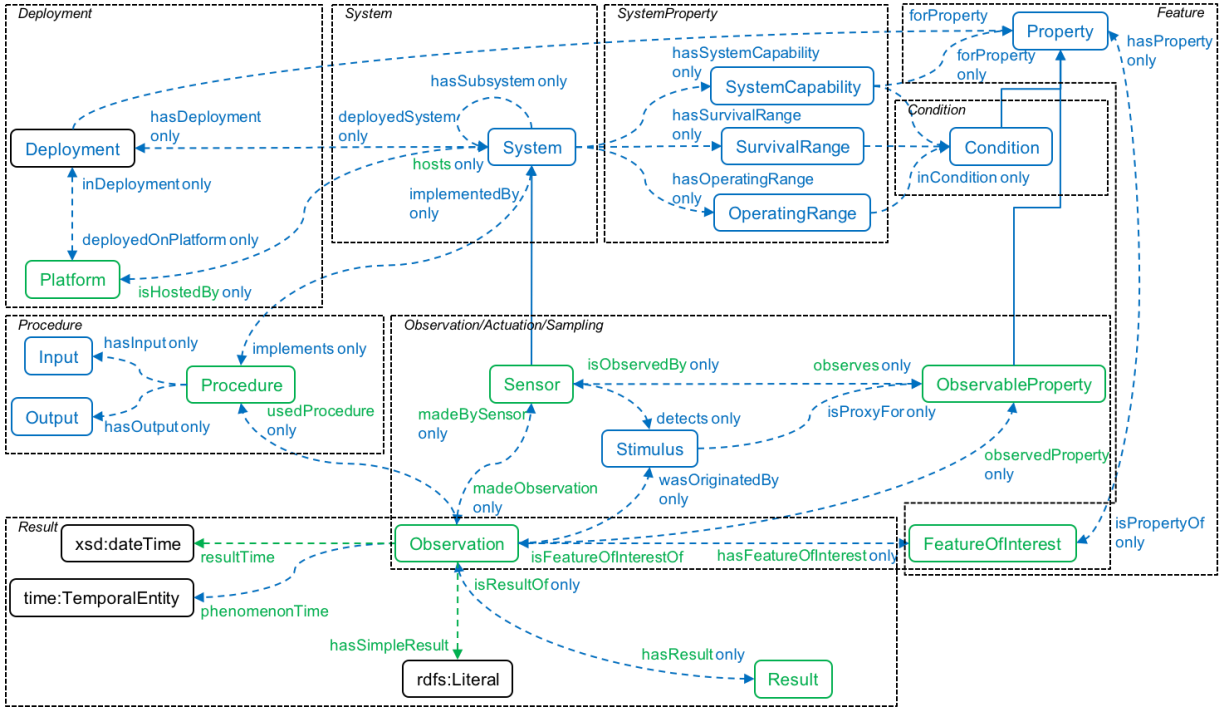



- BOP



# All these ontologies…

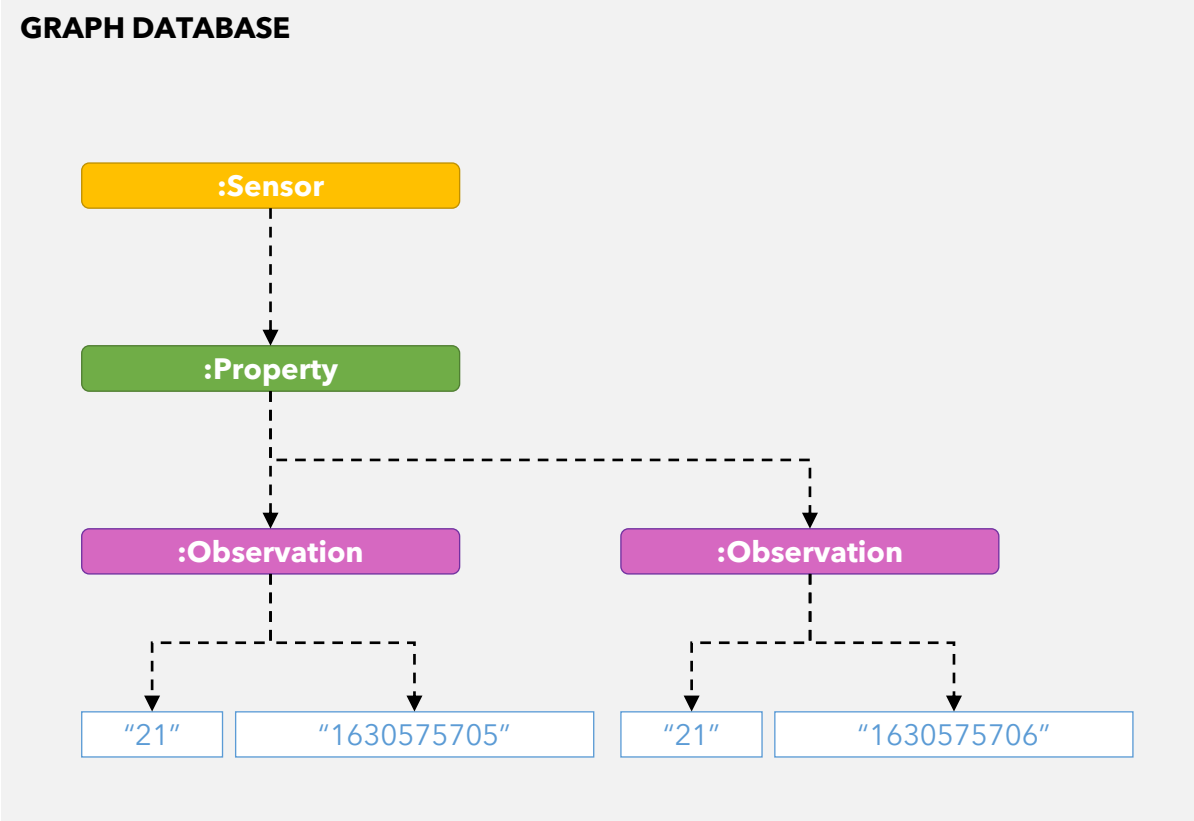
- Different perspectives, same structure
- W3C recommendation: **SOSA/SSN**
  - https://www.w3.org/TR/vocab-ssn/



# Integrating time series data

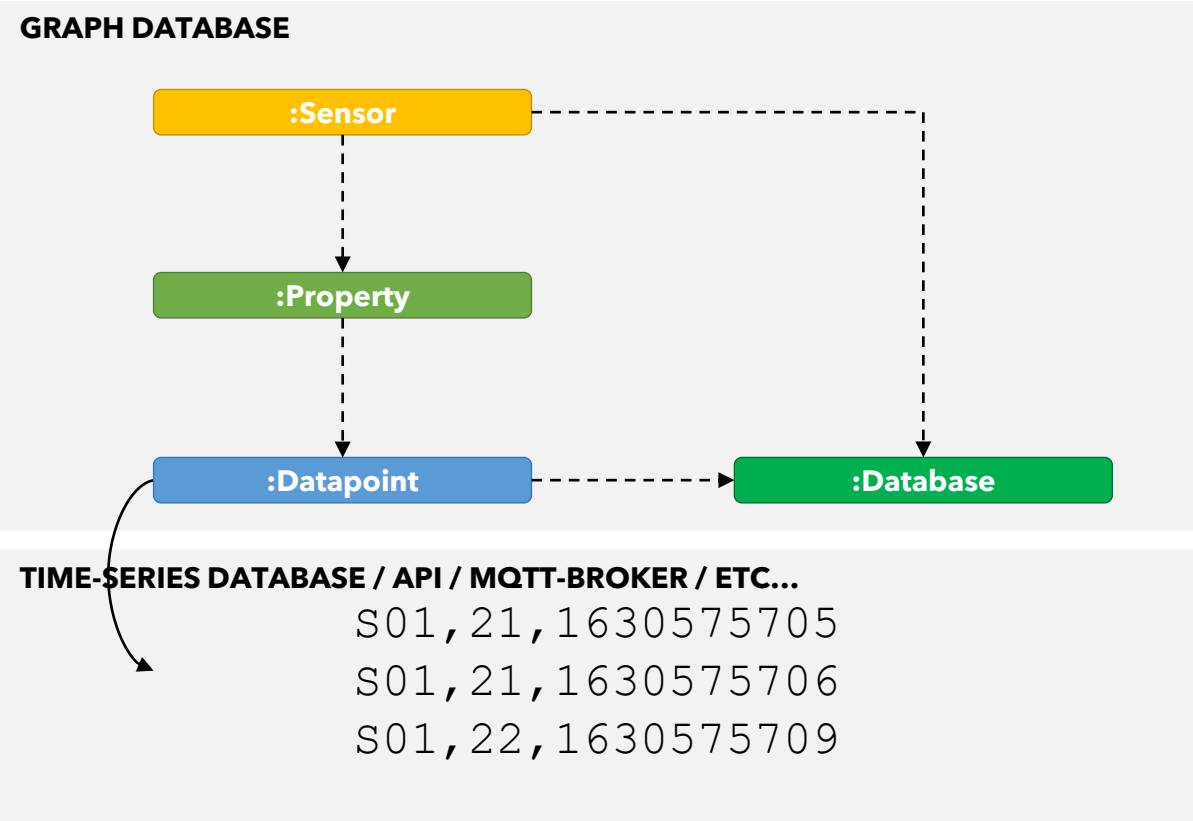
## Method 1 Time series data as RDF

- ✓ Query data directly using SPARQL
- ✓ Simpler software stack
- ✗ Graph explodes
- ✗ Data conversion




## Method 2 Link to an external database

- ✓ Best for storage
- ✓ No data conversion
- ✗ Two query languages



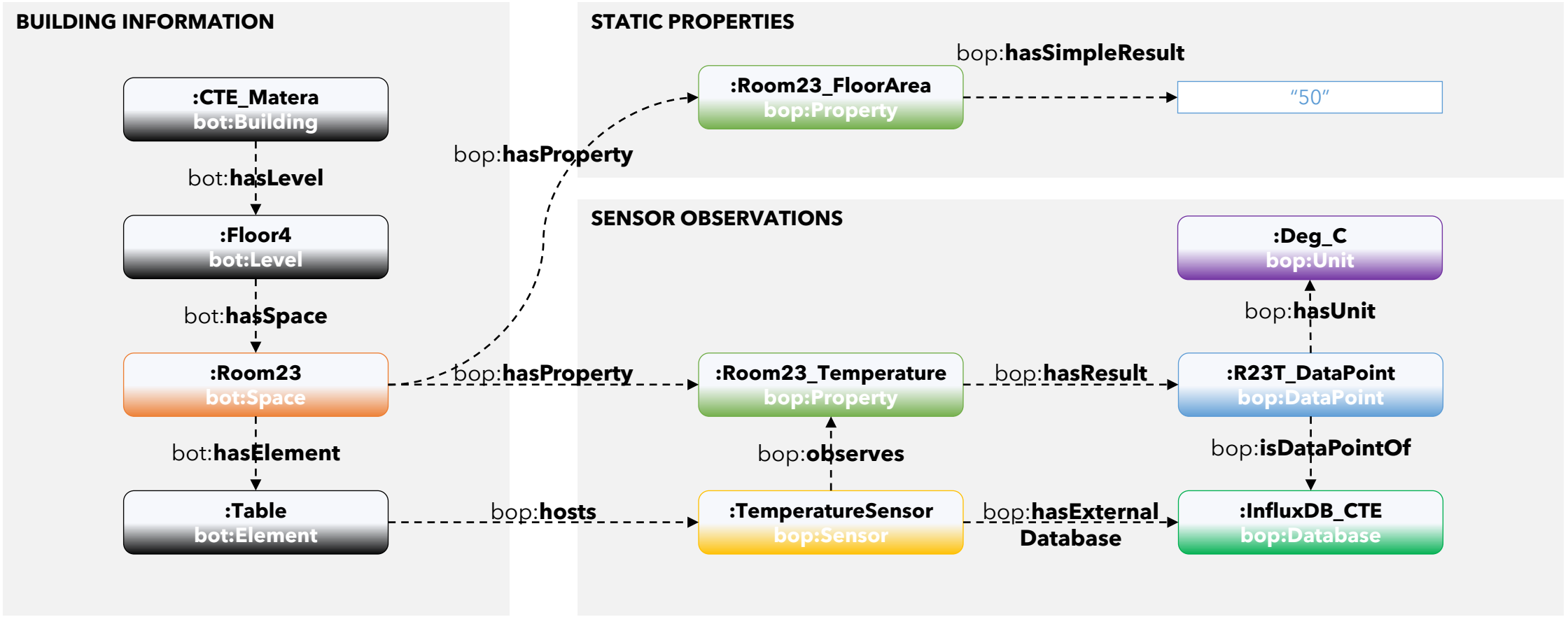
Part 3

# Sensors and Linked Building Data

# Let’s get active!

- Draw an RDF graph, representing **this room** and **a sensor in this room**, using LBD ontologies.
- Which queries could you now write?

# Linking BOT and Sensor metadata



# How to query?

SPARQL

```
PREFIX bot: <https://w3id.org/bot#>
PREFIX bop: <https://w3id.org/bop#>
SELECT * WHERE {
        :CTE_Matera bot:hasLevel ?level .
        ?level bot:hasSpace ?space .
        ?space bop:hasProperty ?property .
        ?property a quantitykind:Temperature .
        ?property bop:isObservedBy ?sensor .
        ?property bop:hasResult ?dataPoint .
        ?dataPoint bop:isPartOfDatabase ?database .
}
```

TSDB QUERY

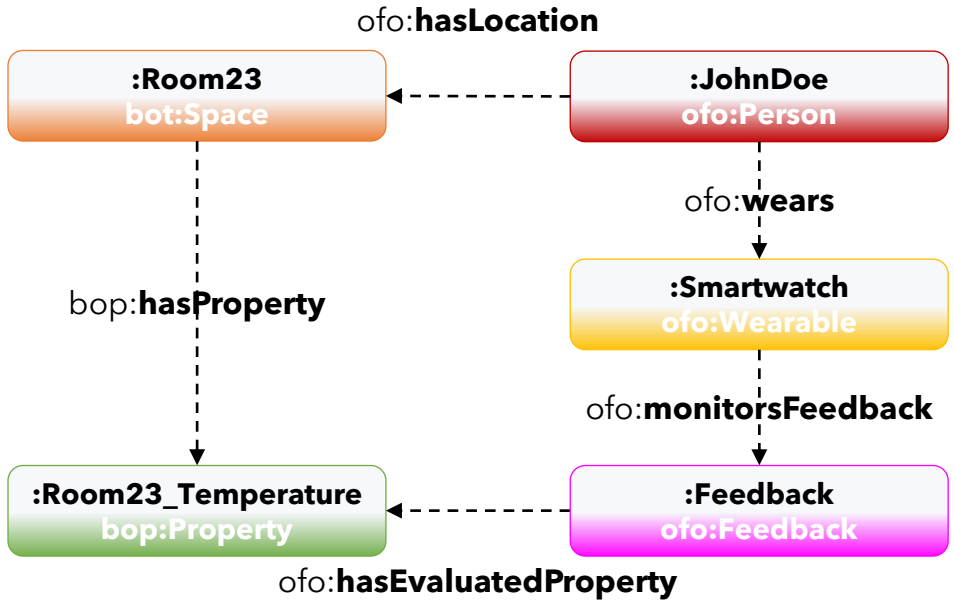
```
from(bucket: "?database")
|> range(start: v.timeRangeStart, stop:
    v.timRangeStop)
|> filter(fn: (r) => r["_measurement"] ==
    "?dataPoint")
```

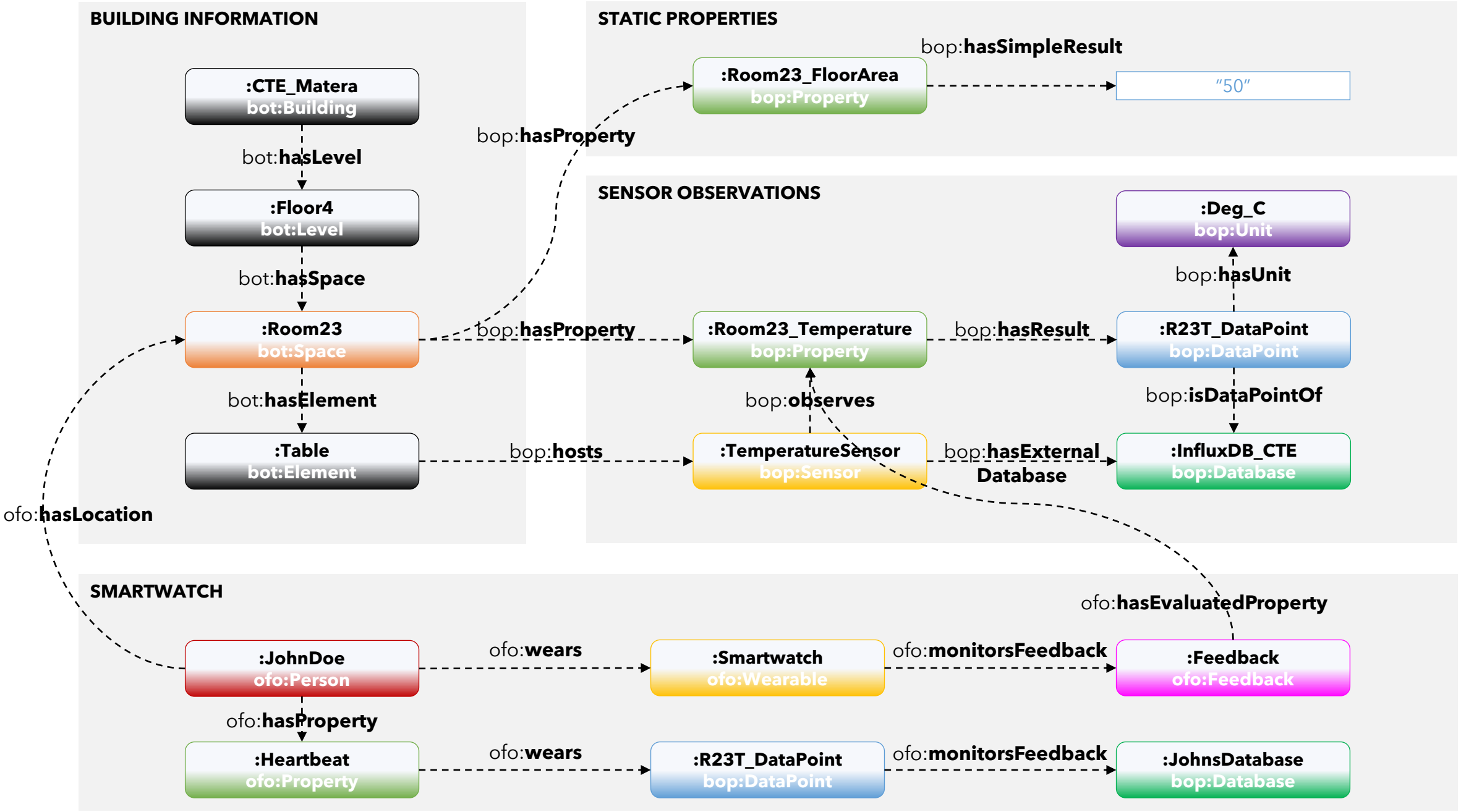
# Never stop drawing graphs…

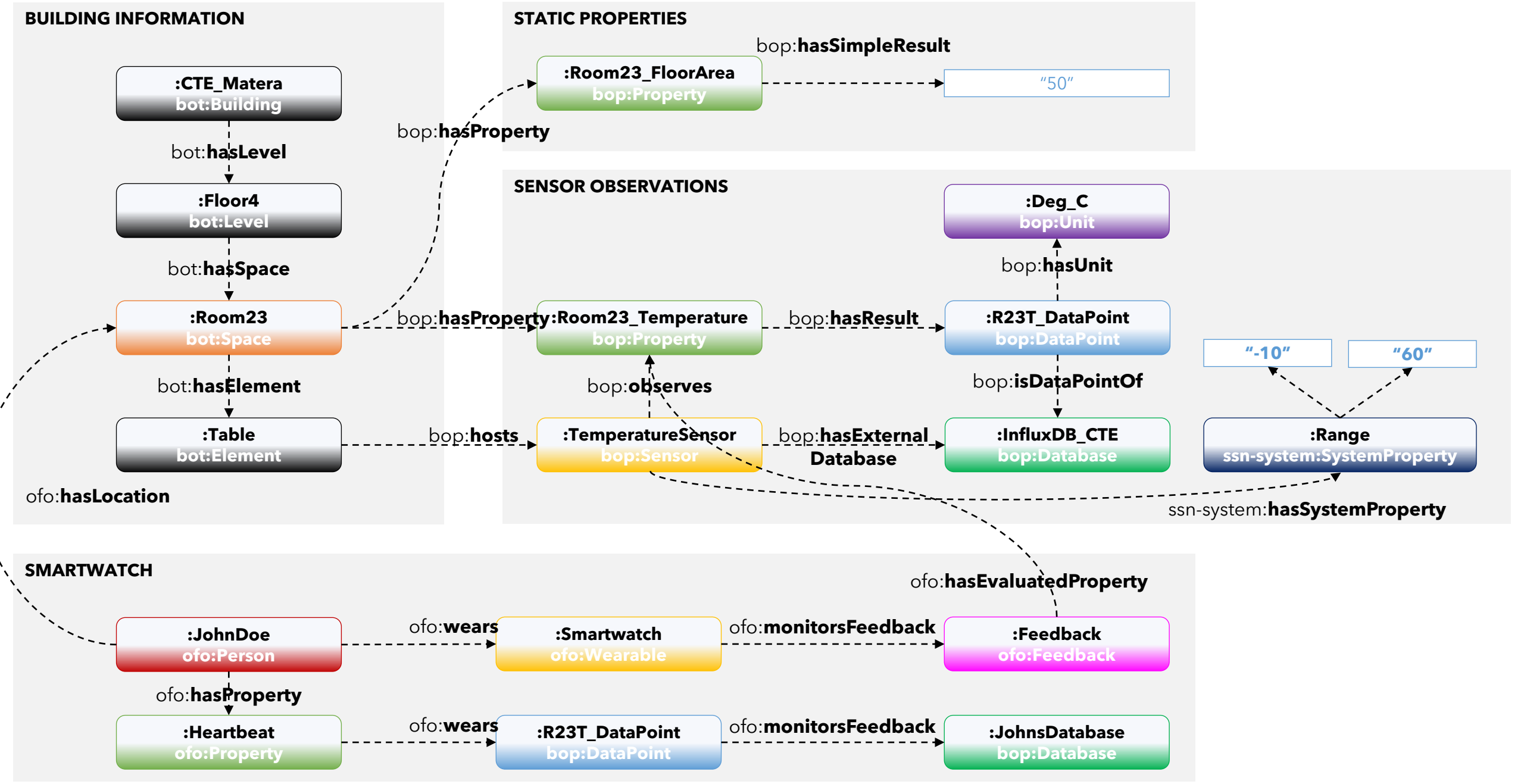
- What if there's multiple sensors in the same room?
- What if one sensor measures multiple properties?
- What if two sensors measure the same property?
- What if two sensors measure in different frequencies?
- Does the exact location of the sensor matter? How would you query this?

# Smartwatch

Measures **feedback** and has various health-related **sensors**





BUILDING INFORMATION
:CTE_Matera
bot:Building
bot:hasLevel
:Floor4
bot:Level
bot:hasSpace
:Room23
bot:Space
bot:hasElement
:Table
bot:Element
ofo:hasLocation
STATIC PROPERTIES
bop:hasSimpleResult
:Room23_FloorArea
bop:Property
"50"
bop:hasProperty
SENSOR OBSERVATIONS
:Deg_C
bop:Unit
bop:hasUnit
bop:hasProperty
:Room23_Temperature
bop:Property
bop:hasResult
:R23T_DataPoint
bop:DataPoint
bop:observes
bop:isDataPointOf
bop:hosts
:TemperatureSensor
bop:Sensor
bop:hasExternal Database
:InfluxDB_CTE
bop:Database
SMARTWATCH
ofo:hasEvaluatedProperty
:JohnDoe
ofo:Person
ofo:wears
:Smartwatch
ofo:Wearable
ofo:monitorsFeedback
:Feedback
ofo:Feedback
ofo:hasProperty
:Heartbeat
ofo:Property
ofo:wears
:R23T_DataPoint
bop:DataPoint
ofo:monitorsFeedback
:JohnsDatabase
bop:Database


BUILDING INFORMATION
:CTE_Matera
bot:Building
bot:hasLevel
:Floor4
bot:Level
bot:hasSpace
:Room23
bot:Space
bot:hasElement
:Table
bot:Element
ofo:hasLocation
STATIC PROPERTIES
bop:hasSimpleResult
:Room23_FloorArea
bop:Property
"50"
bop:hasProperty
SENSOR OBSERVATIONS
:Deg_C
bop:Unit
bop:hasUnit
bop:hasProperty
:Room23_Temperature
bop:Property
bop:hasResult
:R23T_DataPoint
bop:DataPoint
"-10"
"60"
bop:observes
bop:isDataPointOf
bop:hosts
:TemperatureSensor
bop:Sensor
bop:hasExternal Database
:InfluxDB_CTE
bop:Database
:Range
ssn-system:SystemProperty
ssn-system:hasSystemProperty
SMARTWATCH
ofo:hasEvaluatedProperty
:JohnDoe
ofo:Person
ofo:wears
:Smartwatch
ofo:Wearable
ofo:monitorsFeedback
:Feedback
ofo:Feedback
ofo:hasProperty
:Heartbeat
ofo:Property
ofo:wears
:R23T_DataPoint
bop:DataPoint
ofo:monitorsFeedback
:JohnsDatabase
bop:Database

SPARQL

```
PREFIX bot: <https://w3id.org/bot#>
PREFIX bop: <https://w3id.org/bop#>
SELECT * WHERE {
        :CTE_Matera bot:hasLevel ?level .
        ?level bot:hasSpace ?space .
        ?space bop:hasProperty ?property .
        ?property a quantitykind:Temperature .
        ?property bop:isObservedBy ?sensor .
        ?sensor ssn-system:hasSystemProperty ?range .
        ?range :hasValue ?minRange,?maxRange .
        ?property bop:hasResult ?dataPoint .
        ?dataPoint bop:isPartOfDatabase ?database .
}
```

TSDB QUERY

```
from(bucket: "?database")
|> range(start: v.timeRangeStart, stop:
    v.timRangeStop)
|> filter(fn: (r) => r["_measurement"] ==
    "?dataPoint")
|> filter(fn: (r) => r._value > ?minRange)
|> filter(fn: (r) => r._value < ?maxRange)
```

# Data cleaning using explicit range

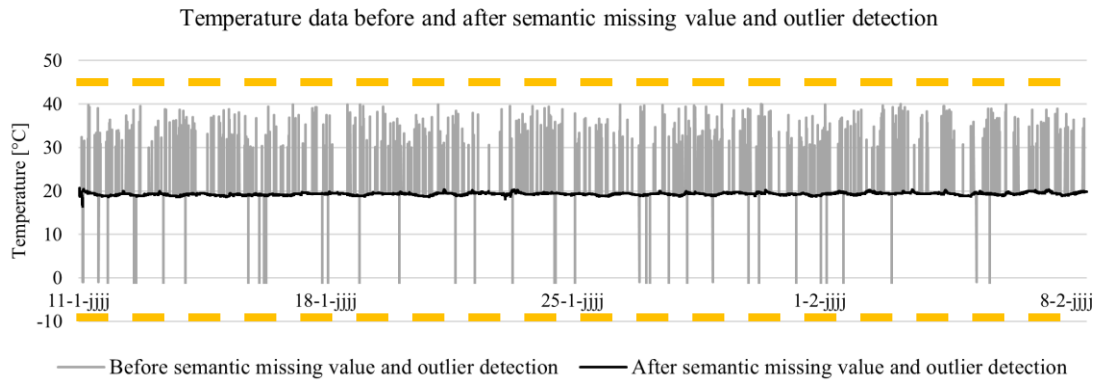
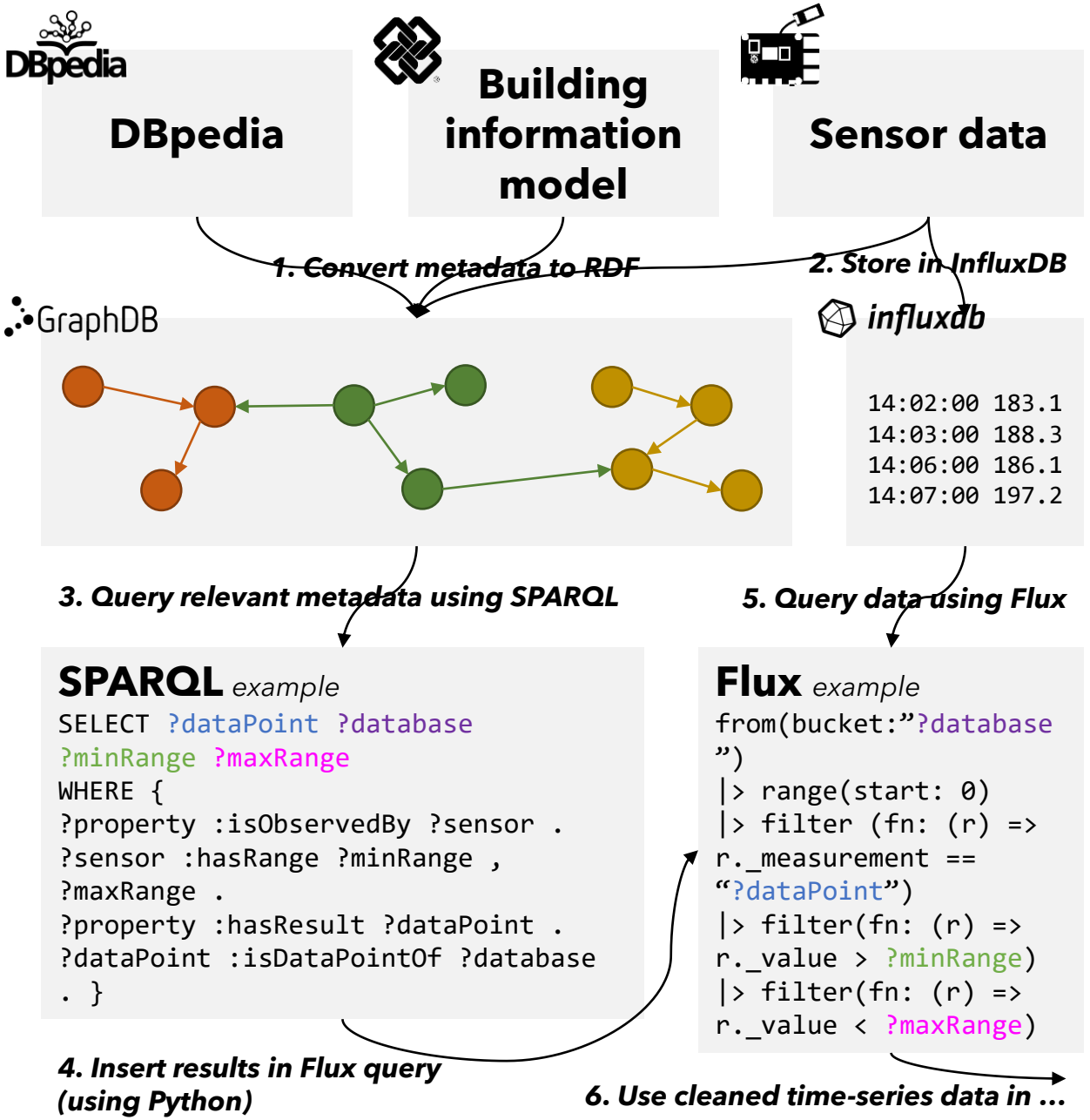



*Figure 3: Temperature data before and after the missing value and outlier detection*

```
PREFIX OFH:
<http://github.com/AlexDonkers/OpenFamilyHome#>
PREFIX bop: <https://w3id.org/bop#>
PREFIX ssn-system:
<http://www.w3.org/ns/ssn/systems/>
SELECT * WHERE {
OFH:Kitchen bop:hasProperty ?property .
?property a quantitykind:Temperature .
?property bop:isObservedBy ?sensor .
?sensor ssn-system:hasSystemProperty ?range ,
?frequency .
?sensor bop:hasNullValueRepresentation
?nullValueRepresentation .
?range bop:hasSimpleMinimum |
bop:hasSimpleMaximum ?rangeValue .
?frequency a ssn-system:Frequency .
?frequency bop:hasSimpleResult ?frequencyValue .
}
```

# Add non-sensor data

Allows you to link other data (e.g. DBpedia) with your sensor data, even if there's no direct relationship in the RDF graph.



# Data cleaning using exp


*Figure 3: Temperature data before and after the missing value and outlier detection*

## Flux

```
from(bucket: "?database")
|> range(start: v.timeRangeStart, stop:
v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] ==
"?dataPoint")
|> filter(fn: (r) => r._value > ?minRange)
|> filter(fn: (r) => r._value < ?maxRange)
```

## SPARQL

```
PREFIX OFH:
<http://github.com/AlexDonkers/OpenFamilyHome
#>
PREFIX bot: <https://w3id.org/bot#>
PREFIX ssn-system:
<http://www.w3.org/ns/ssn/systems/>
PREFIX bop: <https://w3id.org/bop#>
PREFIX quantitykind:
<http://qudt.org/vocab/quantitykind/>
PREFIX dbo: <https://dbpedia.org/ontology/>
PREFIX dbr: <https://dbpedia.org/resource/>
INSERT {
?sensor ssn-system:hasSystemProperty
OFH:CustomRange .
OFH:CustomRange rdf:type ssn-system:Range,
bop:Property , ssn-system:CustomRange.
OFH:CustomRange bop:hasSimpleMinimum "10" .
OFH:CustomRange bop:hasSimpleMaximum "30" .
} WHERE {
?property bop:isObservedBy ?sensor .
?property a quantitykind:Temperature .
?sensor bop:isHostedBy ?host .
?zone bot:containsElement ?host .
?building bot:hasSpace ?zone .
?building a dbr:House .
?site bot:hasBuilding ?building .
?site dbo:location dbr:Netherlands .
}
```

Part 4

# Use-cases

# LBD viz

# ATLAS TU/E

1bRZPDoXj3GePwfwldDbEK

## Options +

## Queries –

Find | SPARQL | Graph locator

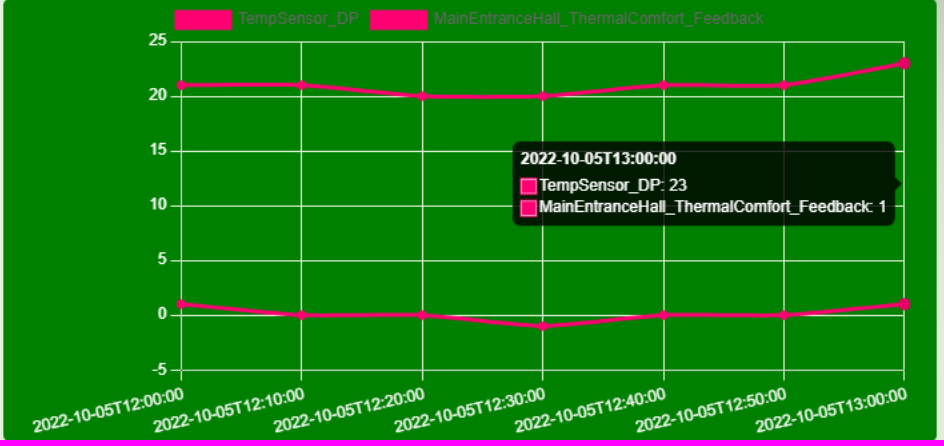
```
PREFIX ofo: <https://w3id.org/ofo#>
PREFIX Atlas: <https://www.github.com/AlexDonkers/Atlas#>
PREFIX prov: <http://www.w3.org/ns/prov#>
select ?Feedback ?Value ?Time ?FeatureOfInterest where {
    ?Feedback ofo:hasFeatureOfInterest Atlas:MainEntranceHall .
    ?Feedback ofo:hasFeatureOfInterest ?FeatureOfInterest .
    ?Feedback ofo:hasFeedbackResult ?Result .
    ?Result ofo:hasValue ?Value .
    ?Result prov:generatedAtTime ?Time .
}
```

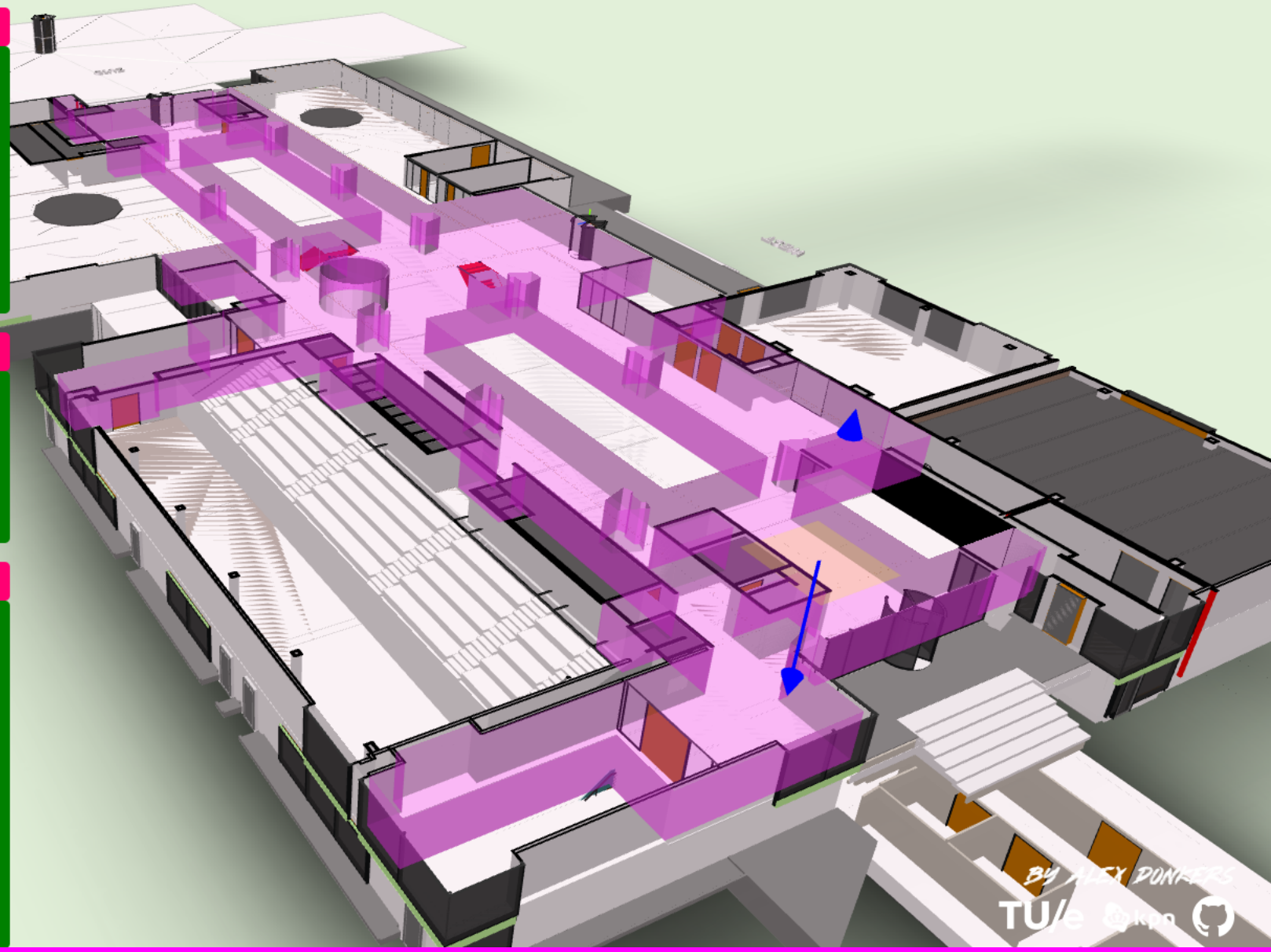
Run query!

## Results –

| FeatureOfInterest | Value | Time | Feedback |
|---|---|---|---|
| MainEntranceHall | Like | 2022-10-05T12:00:00.000 | MainEntranceHall_ThermalComfort_Feedback |
| MainEntranceHall | Dislike | 2022-10-05T12:30:00.000 | MainEntranceHall_ThermalComfort_Feedback |
| MainEntranceHall | Like | 2022-10-05T13:00:00.000 | MainEntranceHall_ThermalComfort_Feedback |

## Sensor data –



# LBD viz

## OFH

2yXAO5SFr65OutxTxWsGLr

| | Now | Total |
|---|---|---|
| Energy Socket | 3.166 | 0.828 |
| P1Meter | 1467 | 11759.12 |
| Water | 0 | 7.867 |

**Options** –

Tools | Settings | Authorization | Energy

Load IFC | Default IFC

**Queries** –

Find | SPARQL | Graph locator

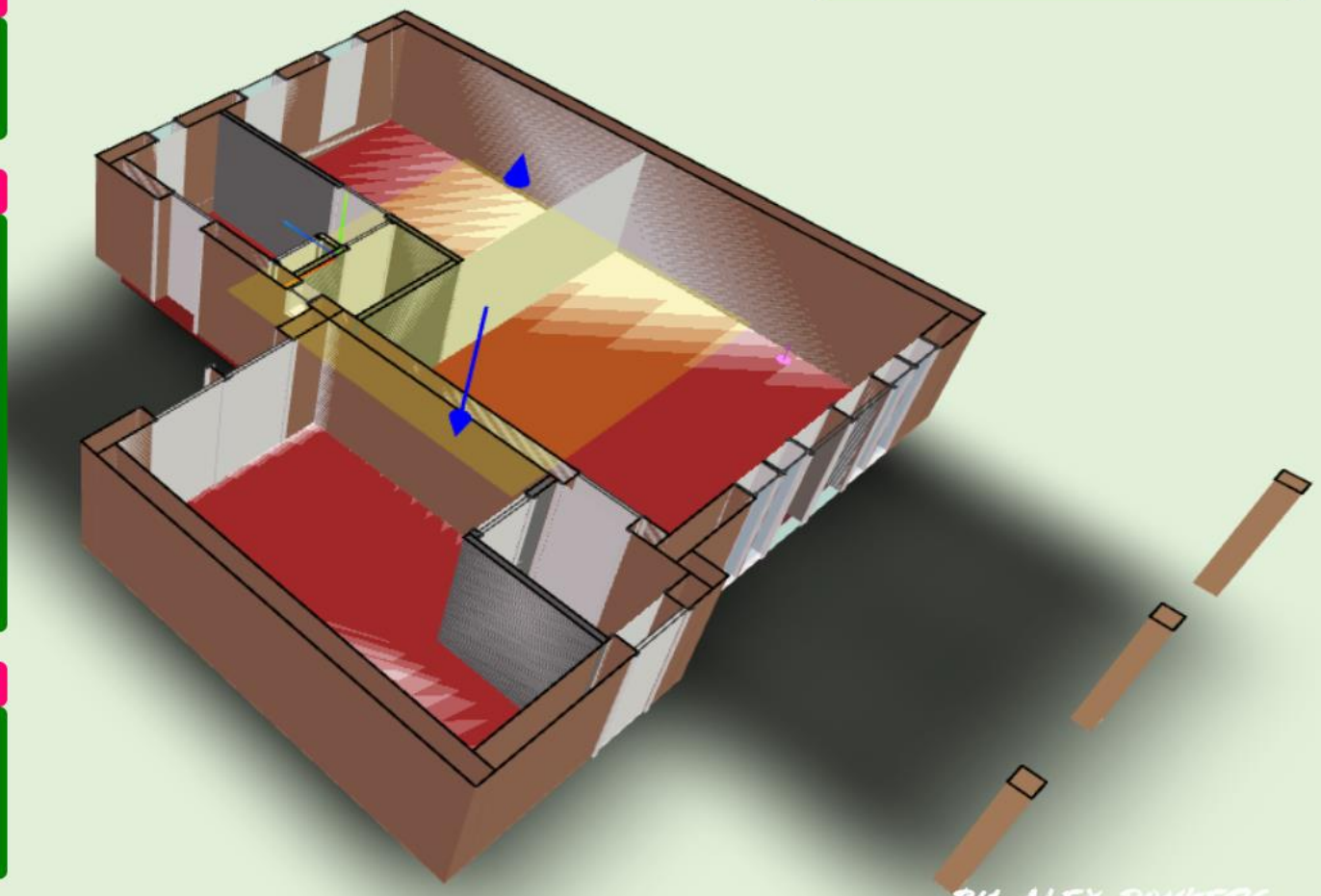
```
PREFIX ofo:     <https://w3id.org/ofo#>
PREFIX prov:    <http://www.w3.org/ns/prov#>

SELECT ?Person ?Property ?Value ?Time
WHERE {
  ?Person ofo:givesFeedback ?Feedback .
  ?Feedback ofo:hasResult / ofo:hasValue ?Value .
  ?Feedback ofo:hasResult / prov:generatedAtTime ?Time .
  ?Feedback ofo:hasEvaluatedProperty ?Property .
}
LIMIT 5
```

Run query!

**Results** –

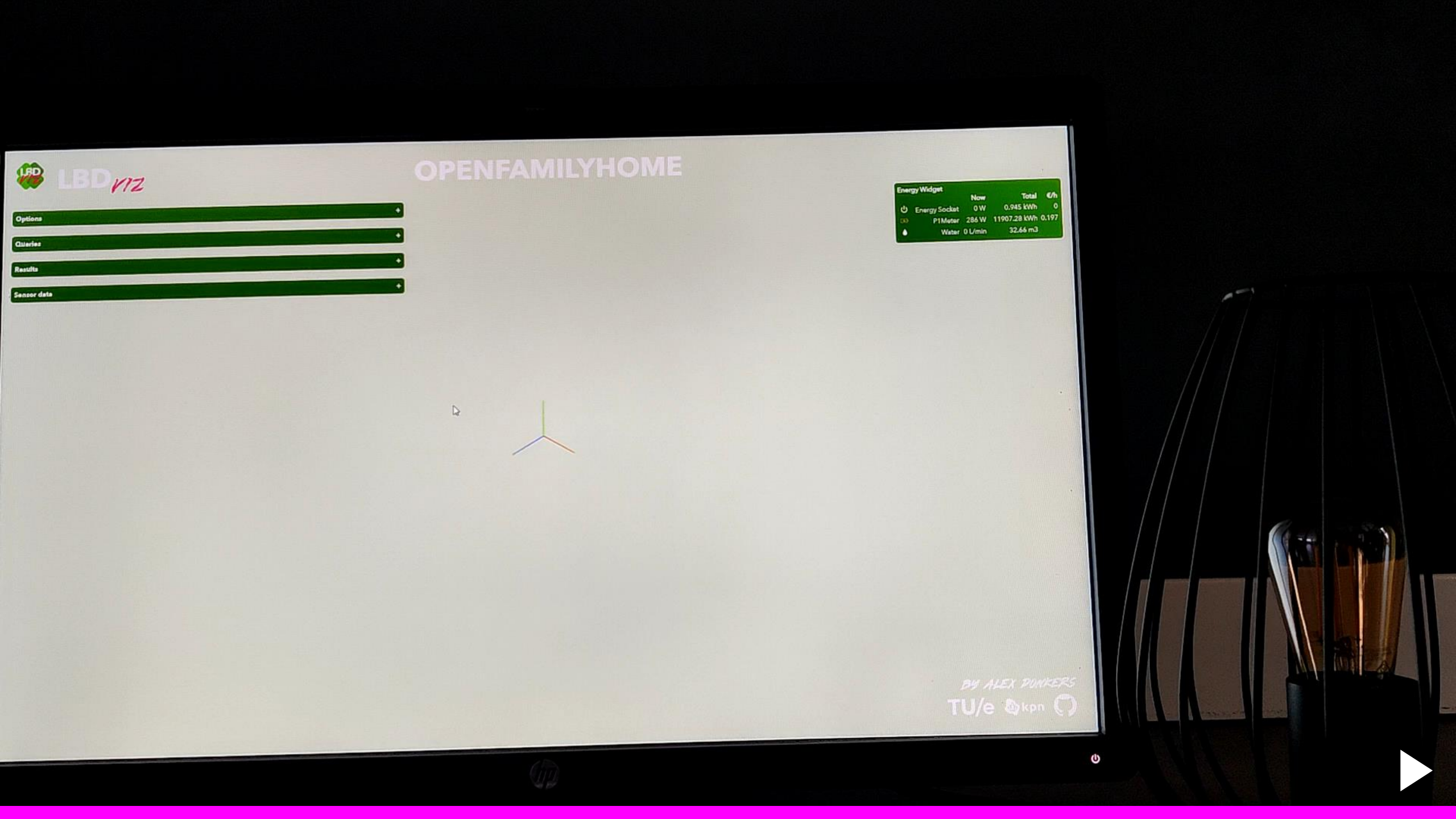
| Property | Value | Time | Person |
|---|---|---|---|
| KitchenIlluminance | Like | 2023-02-25T17:00:00+01:00 | JohnDoe |
| KitchenIlluminance | Dislike | 2023-02-25T17:30:00+01:00 | JohnDoe |
| KitchenIlluminance | Dislike | 2023-02-25T18:00:00+01:00 | JohnDoe |
| KitchenIlluminance | Like | 2023-02-25T18:30:00+01:00 | JohnDoe |
| KitchenIlluminance | Like | 2023-02-25T19:00:00+01:00 | JohnDoe |

**Sensor data** +

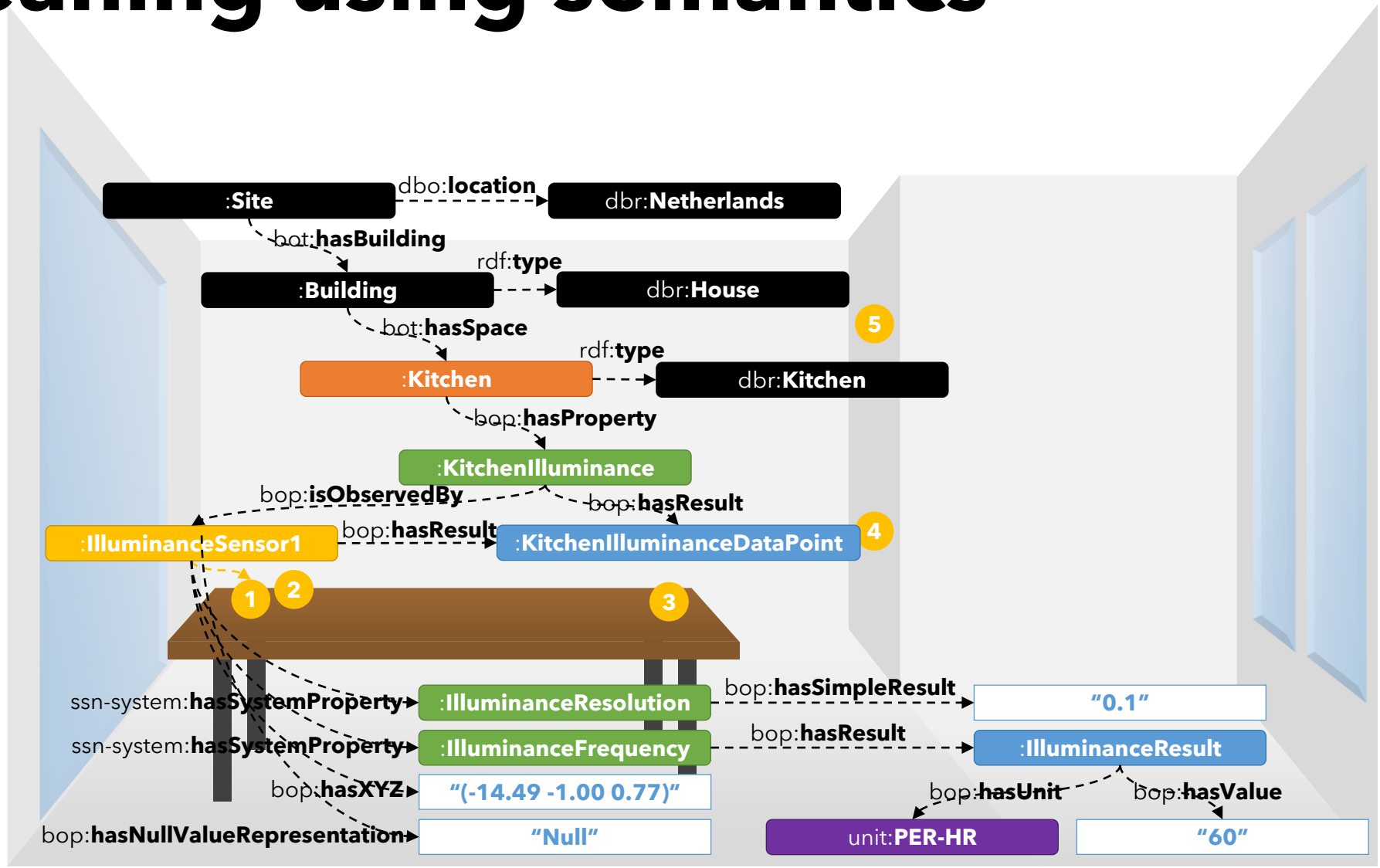


BY ALEX DONKERS

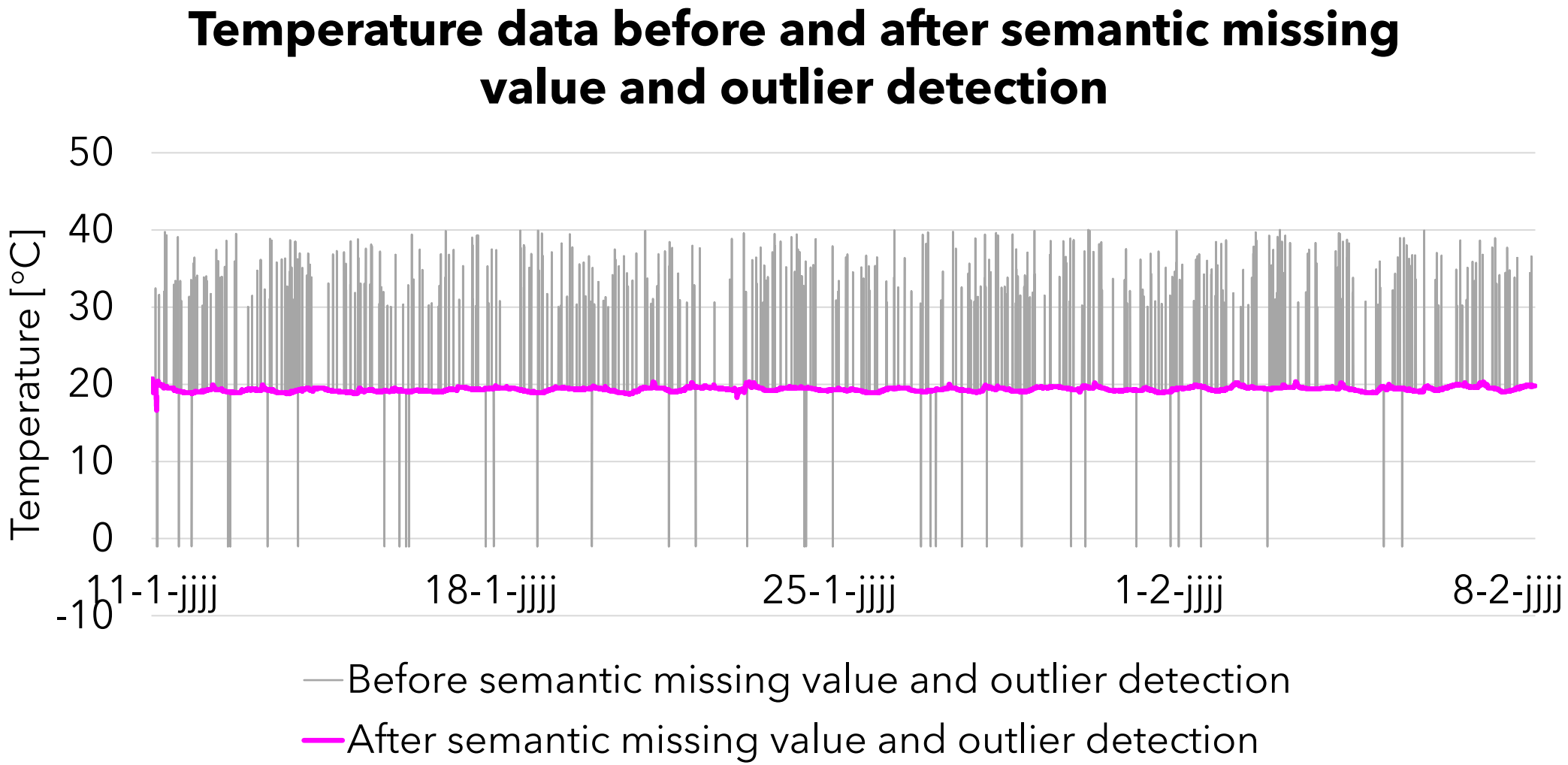
TU/e kpn


LBD VIZ
OPENFAMILYHOME
Options
Queries
Results
Sensor data
Energy Widget
Now
Total
Energy Socket 0 W 0.945 kWh 0
P1Meter 286 W 11907.28 kWh 0.197
Water 0 L/min 32.66 m3
BY ALEX DONKERS
TU/e
kpn

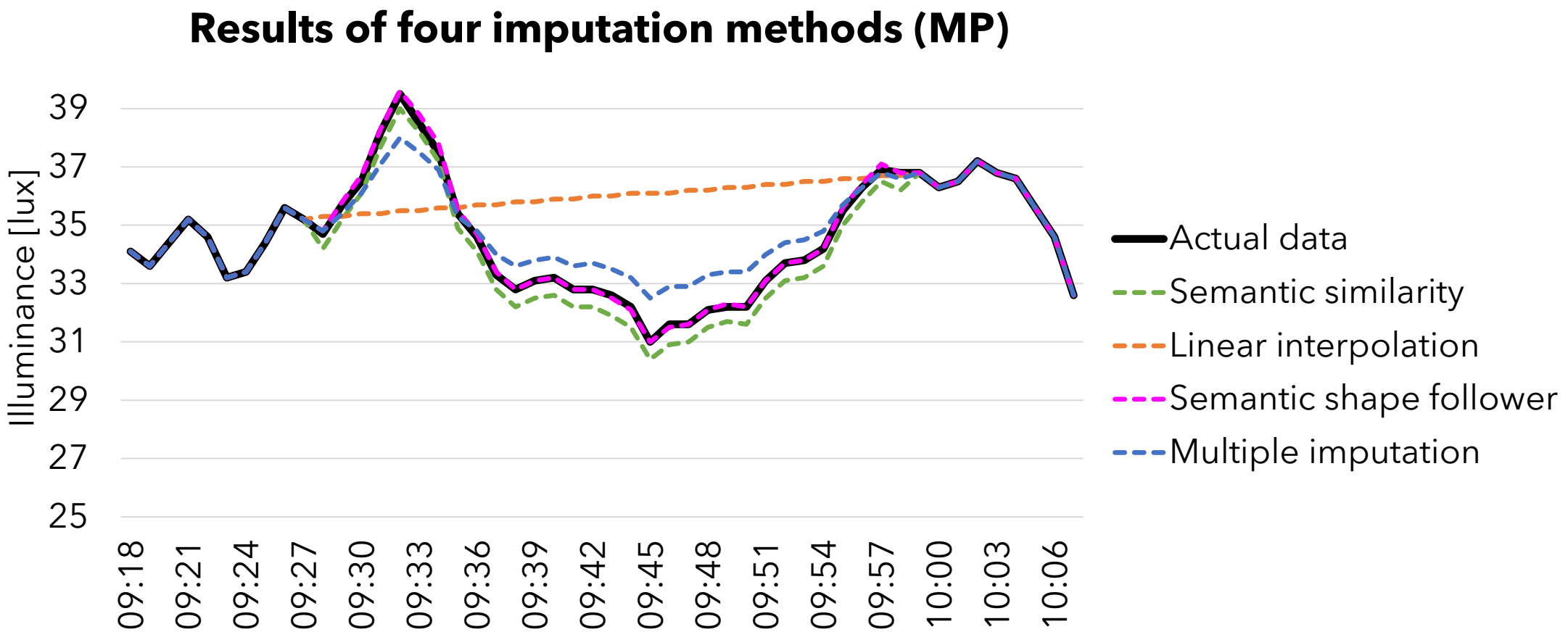
# Data cleaning using semantics



# Data cleaning using semantics



# Data cleaning using semantics



Part 5

# Hands-on showcase

**GEOMETRY**
GraphDB

```
sparql.setQuery("""PREFIX props: <https://w3id.org/props#>
PREFIX quantitykind: <http://qudt.org/vocab/quantitykind/>
PREFIX bot: <https://w3id.org/bot#>
PREFIX bop: <https://w3id.org/bop#>
PREFIX beo: <http://pi.pauwel.be/voc/buildingelement#>
PREFIX : <https://research.tue.nl/nl/persons/alex-ja-donkers#>

SELECT ?room ?wall ?interface ?interfaceWidthValue

WHERE {
?room bop:hasSimplePropertyState "Bedroom" ;
      bot:adjacentElement ?wall .
?wall a beo:Wall .
?interface bot:interfaceOf ?room, ?wall ;
           bop:hasProperty ?width .
?width a quantitykind:Width ;
       bop:hasPropertyState / bop:hasValue ?interfaceWidthValue
} """)
```

# TEMPERATURE
GraphDB

```
graph_url = "http://localhost:7200/repositories/OpenSmartHomeRepos
itory"
sparql = SPARQLWrapper(graph_url)
sparql.setQuery("""PREFIX props: <https://w3id.org/props#>
PREFIX quantitykind: <http://qudt.org/vocab/quantitykind/>
PREFIX bot: <https://w3id.org/bot#>
PREFIX bop: <https://w3id.org/bop#>
PREFIX : <https://research.tue.nl/nl/persons/alex-ja-donkers#>

SELECT ?property ?datapoint ?database

WHERE {
?room bop:hasSimplePropertyState "Bedroom" ;
        bot:containsElement ?sensor .
?sensor bop:observes ?property .
?property a quantitykind:Temperature ;
         bop:hasPropertyState ?datapoint .
?datapoint bop:isDataPointOf ?database .
} """)
```

**TEMPERATURE**
InfluxDB

```
client = InfluxDBClient(host='localhost', port=8086)

client.switch_database(database)

resultset = client.query(

'SELECT ‘ + property + ’ FROM ’ + datapoint + ‘ WHERE time <= ‘ +
str(maxTime) + ‘ AND time > ‘ + str(minTime) + ‘ ORDER BY time
DESC LIMIT 1’)

value = list((resultset.get_points(measurement=datapoint)))

temperature = value[0]['value']
```

# I hope you now understand

- What a sensor is and why time series data is different from other data
- How we can semantically represent a sensor
- How we can integrate sensor data and building information
- How we can query this data
- How we can use this integrated data

# Sensors and Linked Building Data

**SSoLDAC2023**

Alex Donkers

a.j.a.donkers@tue.nl