

# RUNTIME VERIFICATION FOR SPATIO-TEMPORAL PROPERTIES OVER IOT NETWORKS

Second partial release of the  
project

DEGREE IN COMPUTER  
ENGINEERING

Oihana Garcia Anakabe

2021/2022

# Index

---



Introduction



Development



Problems and solutions



Work done until now



Conclusions

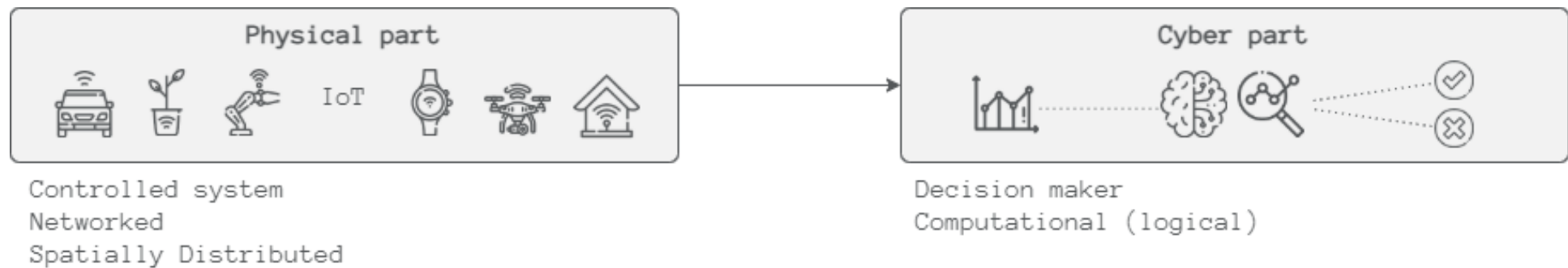
# Introduction

- Description
- Objectives

# Description

---

- CPS (Cyber Physical Systems) is one of the systems that can exploit an IoT infrastructure
  - IoT is a field that is growing fast (2030 → 25,44 billion worldwide connected devices)
  - In CPS physical systems are monitored and/or controlled by a computational core
    - Monitoring is an activity related to the wider category of Runtime Verification (RV)
  - The increasing numbers of IoT devices and intelligent systems made CPS influence society



# Objectives

---

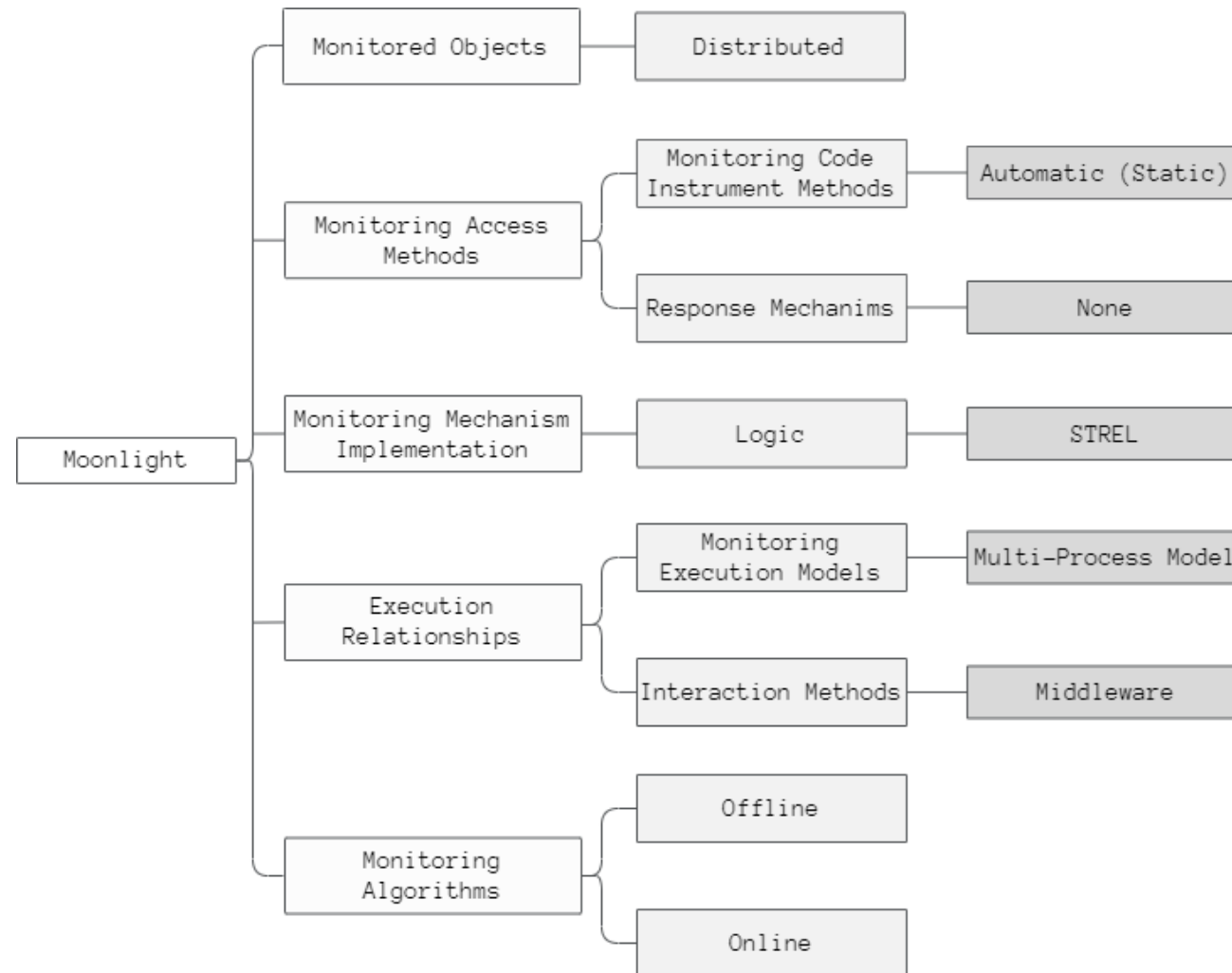
This project focuses precisely on the challenges when doing monitoring on CPS over IoT.

- ✓ Feed MoonLight monitor with live data
- ✓ Implement a middleware that offers different services
- ✓ Establish the communication between the sensors and the monitor
- ✓ Show the results to the user

# Development

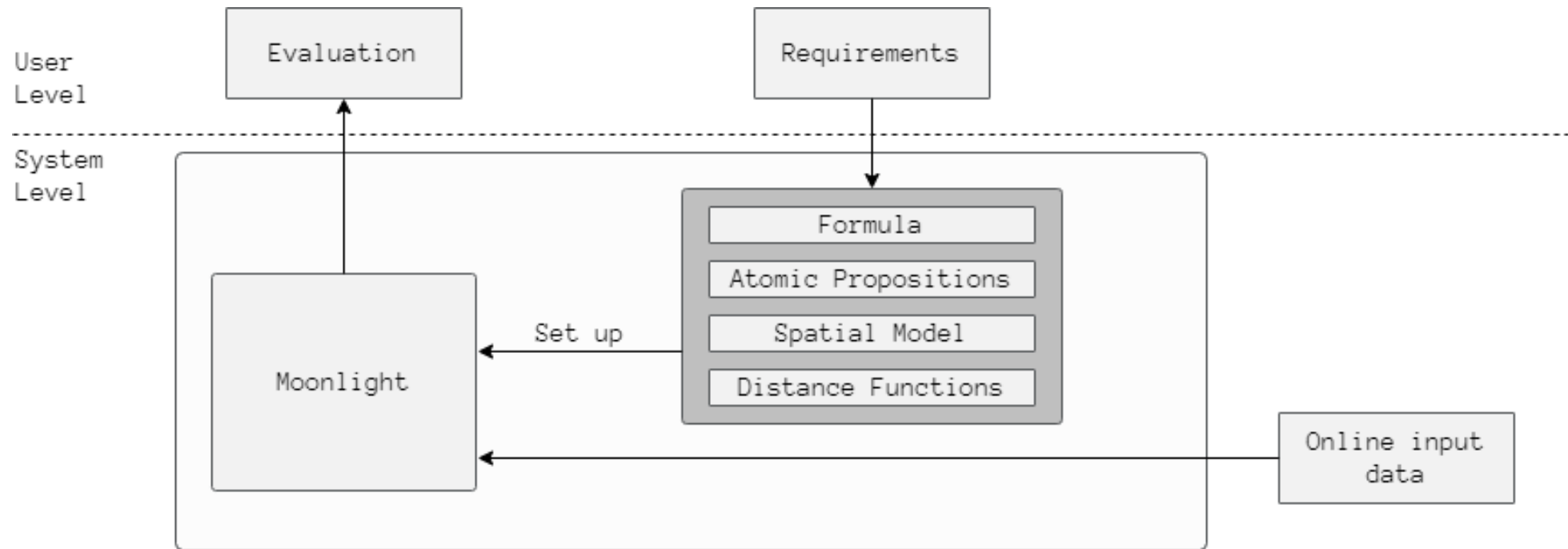
- Moonlight features
- Moonlight online monitor
- Middleware
- Services
- Service Oriented Architecture
- Physical system
- Use case

# Moonlight features



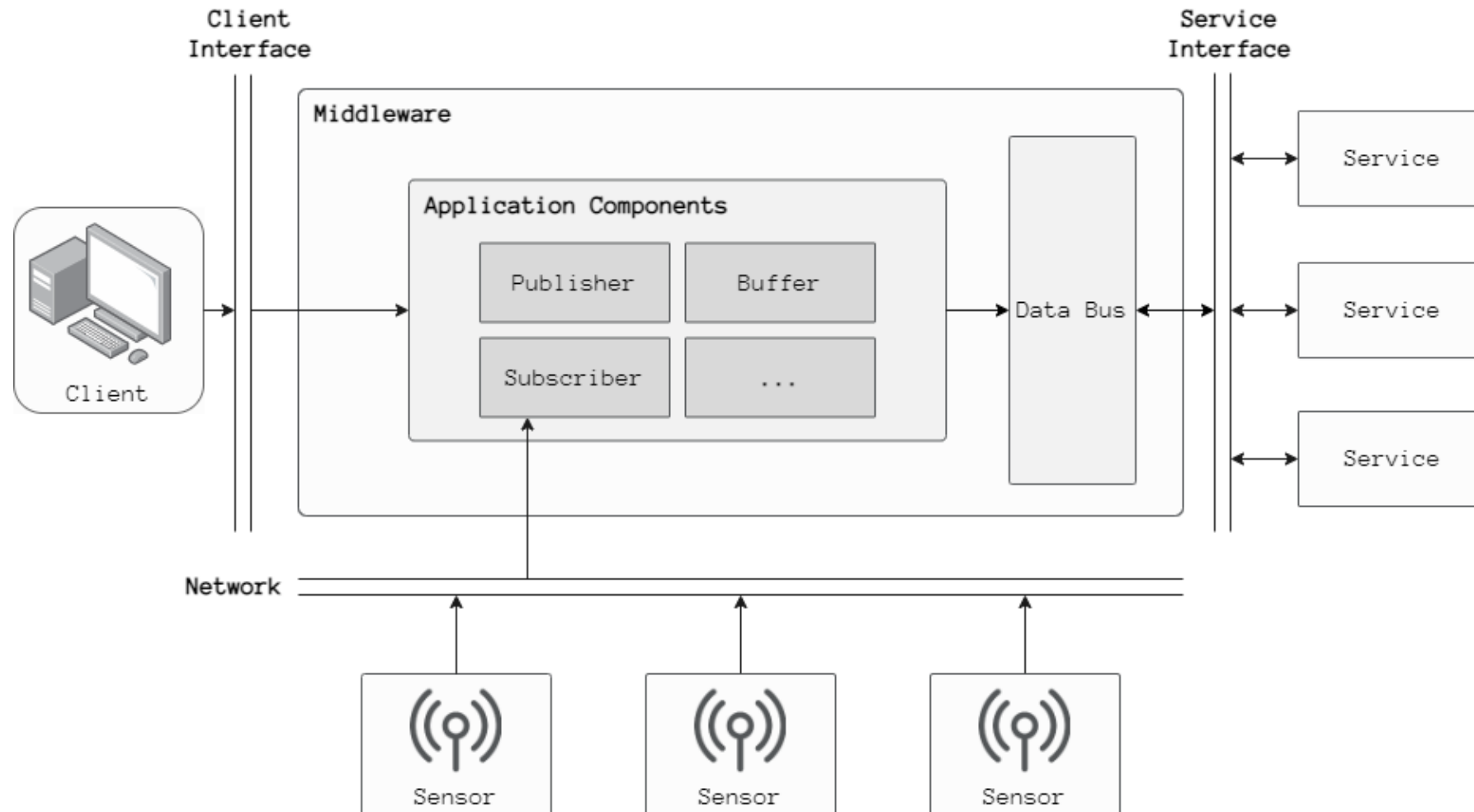
# Moonlight online monitor

---





# Middleware



# Services

---

- **Sensor service**

Receives the messages of the sensors.

Communication using MQTT

- **Moonlight service**

Converts the raw data to be adequate for the monitor

Save the values in a buffer

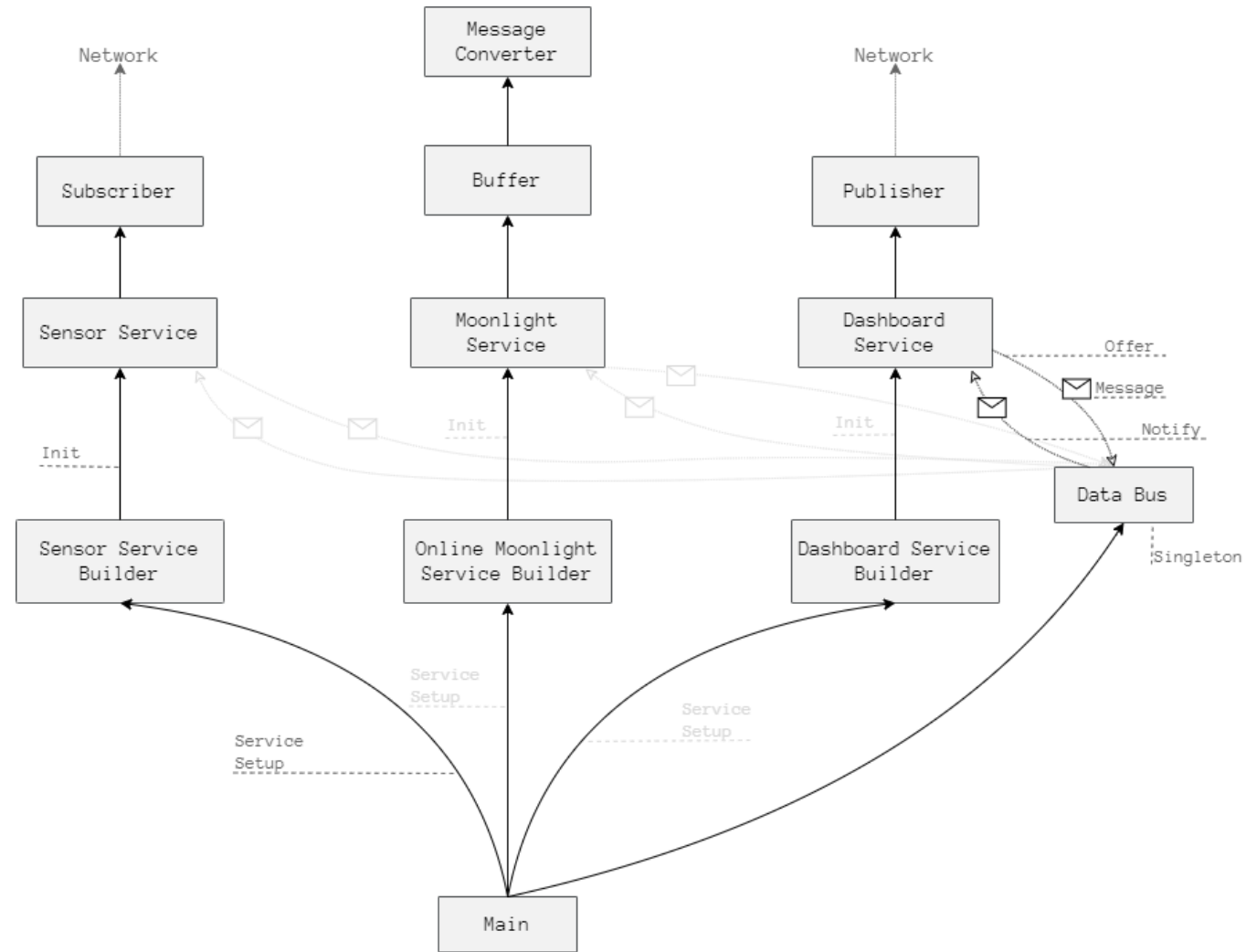
Monitor the input data

- **Thingsboard service**

Sends the system's information to the dashboard using MQTT

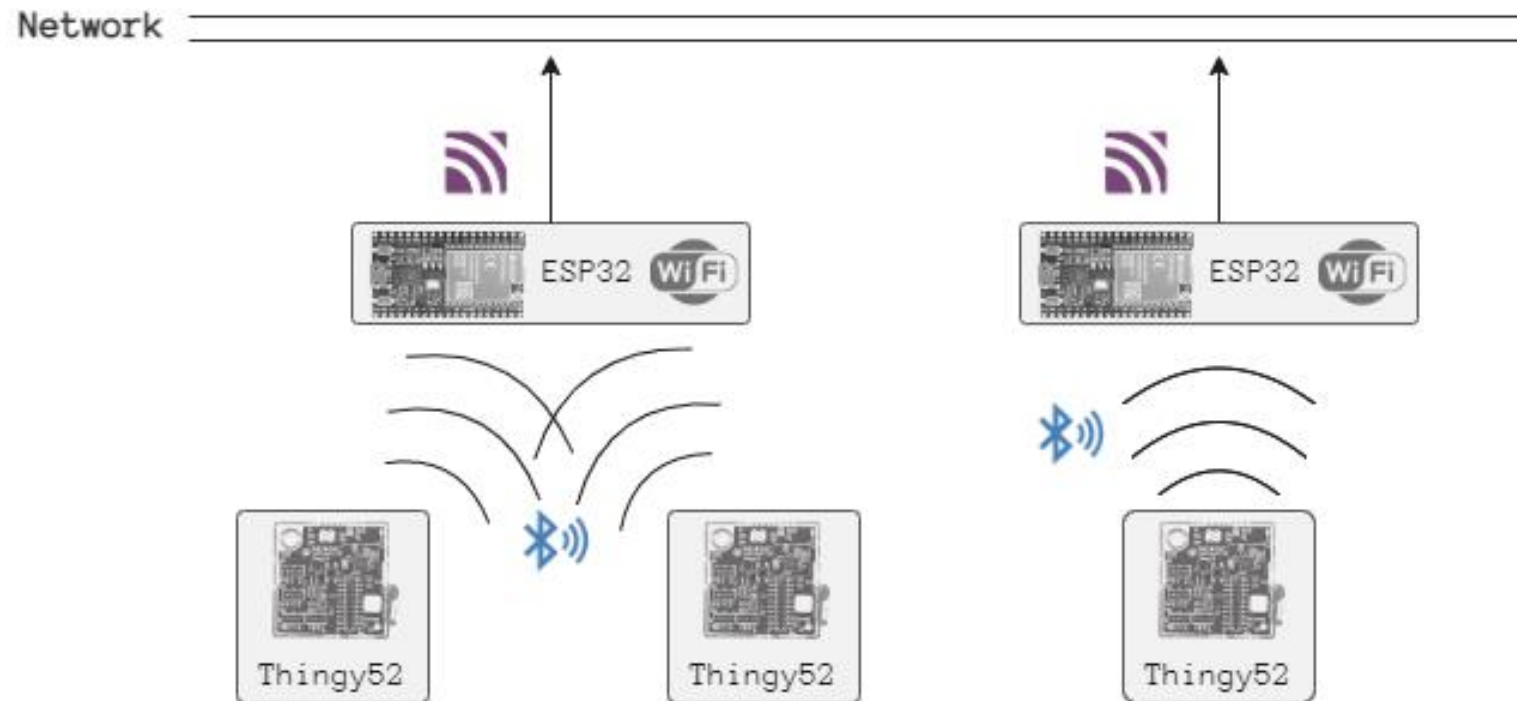
1. The sensors values
2. Monitor results

# Service Oriented Architecture



# Physical system

---



# Use case

---

## Office use case

The environmental conditions must be adequate:

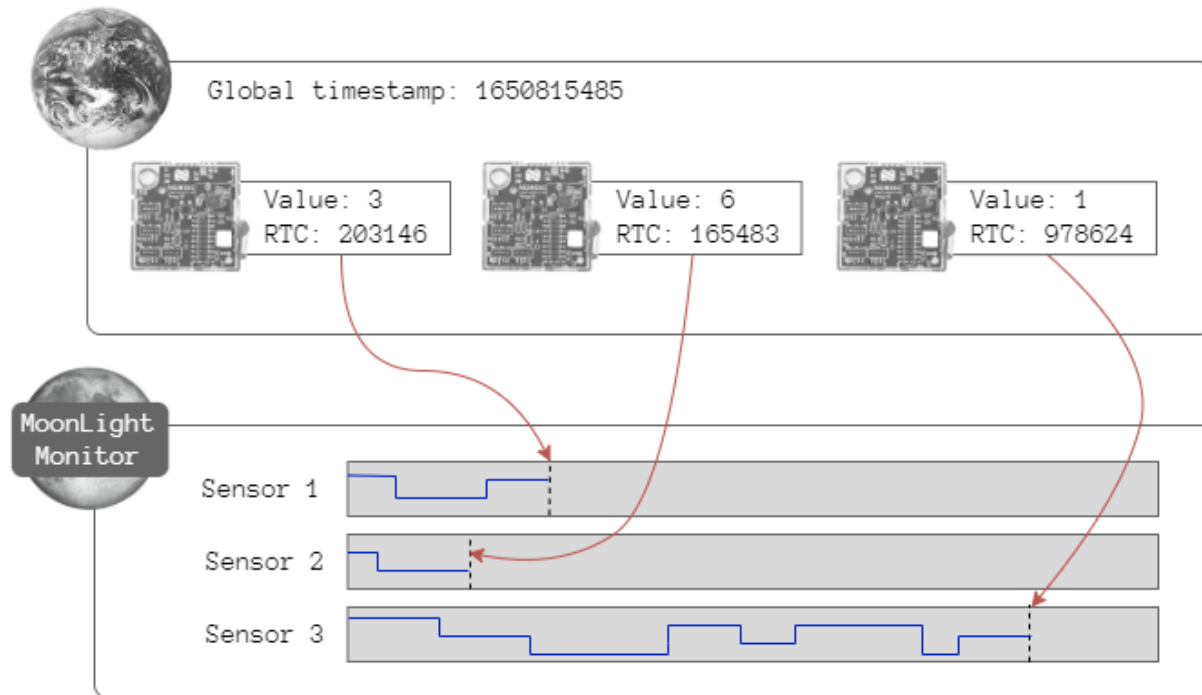
$$\begin{aligned} &\text{Temperature} < 30^{\circ}\text{C} \\ &\quad \wedge \\ &(\text{CO}_2 > 600) \rightarrow F_{[0,1500]}(\text{CO}_2 < 600) \end{aligned}$$

# Problems and solutions

- Sensors time synchronization
- Missing values and imprecise signals
- Other problems

# Sensors time synchronization

## PROBLEM



## SOLUTION

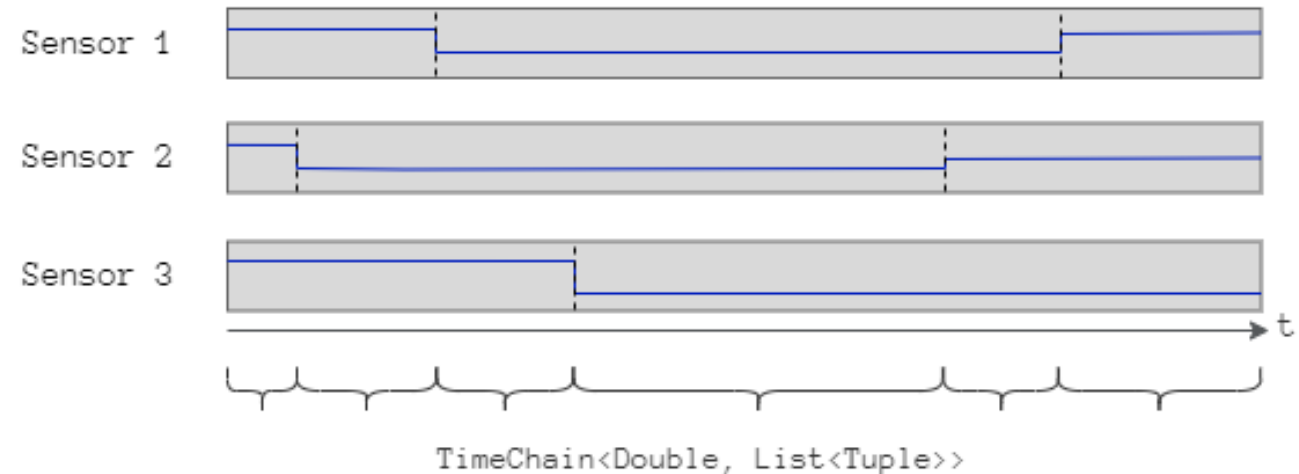
Time List
203146
165483
97624

# Missing values and imprecise signals

## PROBLEM

- Not all the sensors send the data at the same time
- There are gaps between each sensor's messages
- The time interval between messages can change

## SOLUTION





# ESP problems

---

## PROBLEM

- The ESP is not stable
  - Sometimes it doesn't connect to the MQTT
  - Sometimes the connections are abruptly interrupted

- It does not have a long memory

- Disconnecting and connecting multiple times to the Thingy52s consume a lot of memory and it crushes

————→ Set permanent connections

- It can't handle more than 3 BLE connections at the same time

————→ Program each ESP to connect to less than 4 Thingys

# Other problems

---

- MoonlightRecord (a Moonlight java class) had some problems:

Escape formula + online monitor + MoonlightRecord = infinite loop

MoonlightRecord null values didn't throw errors, wrong error handling

Solution: Notify this error to Ennio → He created Tuple a class that works in the same way but without problems

- I was having problems with Windows + Zephyr project

Solution: Use ubuntu

# Work done until now

- Results
- June prospect

# Results

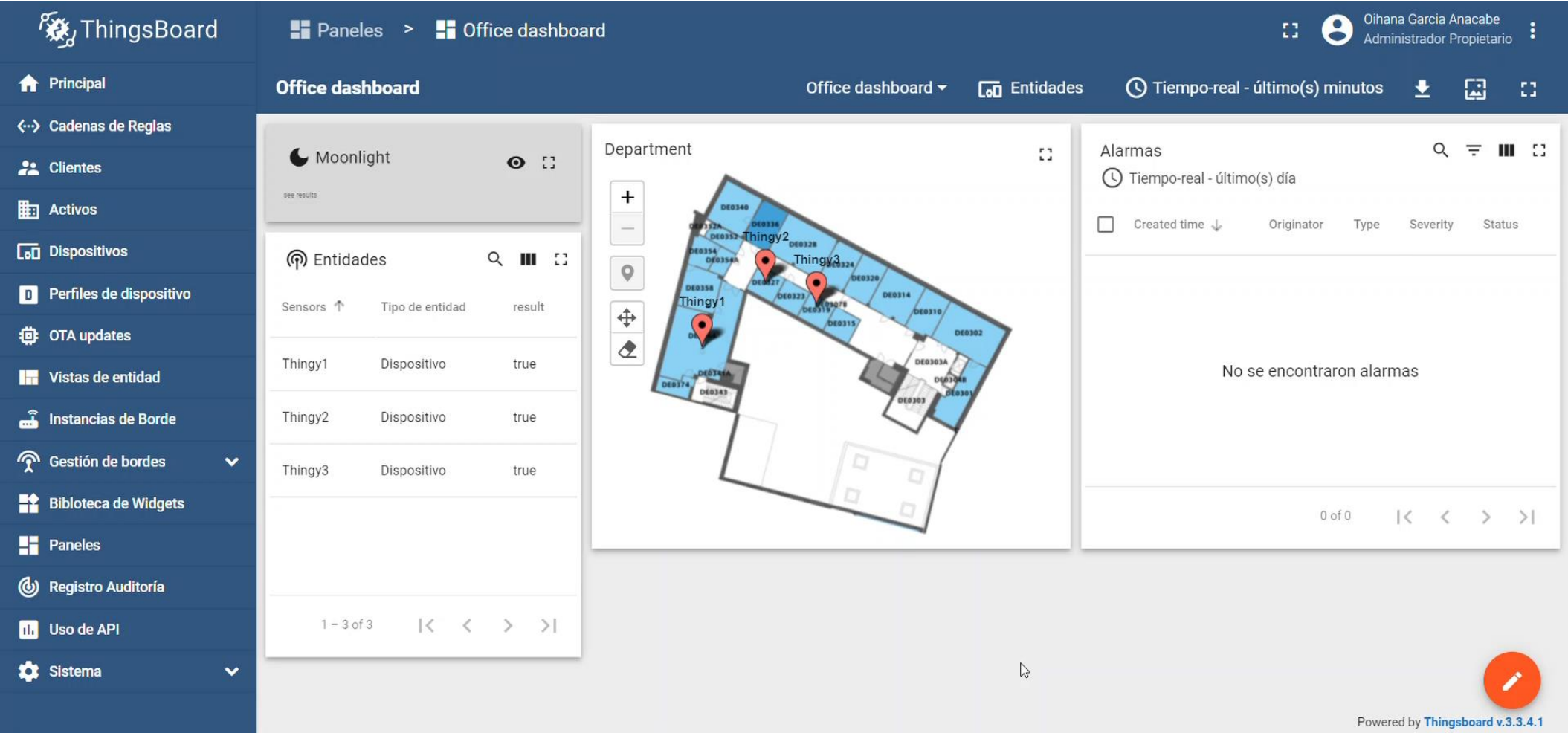
```
The characteristic value was: 369256;28.250000;48.000000;533;20
JSON -> { 'id': 000,'time': 369256,'temp': 28.250000,'hum': 48.000000,'co2': 533,'tvoc': 20}
-> 000, { 'id': 000,'time': 369256,'temp': 28.250000,'hum': 48.000000,'co2': 533,'tvoc': 20}
The characteristic value was: 395394;28.625000;48.000000;503;15
JSON -> { 'id': 001,'time': 395394,'temp': 28.625000,'hum': 48.000000,'co2': 503,'tvoc': 15}
-> 001, { 'id': 001,'time': 395394,'temp': 28.625000,'hum': 48.000000,'co2': 503,'tvoc': 15}
The characteristic value was: 445213;28.625000;48.000000;457;8
JSON -> { 'id': 002,'time': 445213,'temp': 28.625000,'hum': 48.000000,'co2': 457,'tvoc': 8}
-> 002, { 'id': 002,'time': 445213,'temp': 28.625000,'hum': 48.000000,'co2': 457,'tvoc': 8}
The characteristic value was: 369471;28.250000;49.000000;505;15
```

```
double distance = 7.0;
SpatialModel<Double> spatialModel = buildSpatialModel(size);
Formula formula = formula();
```

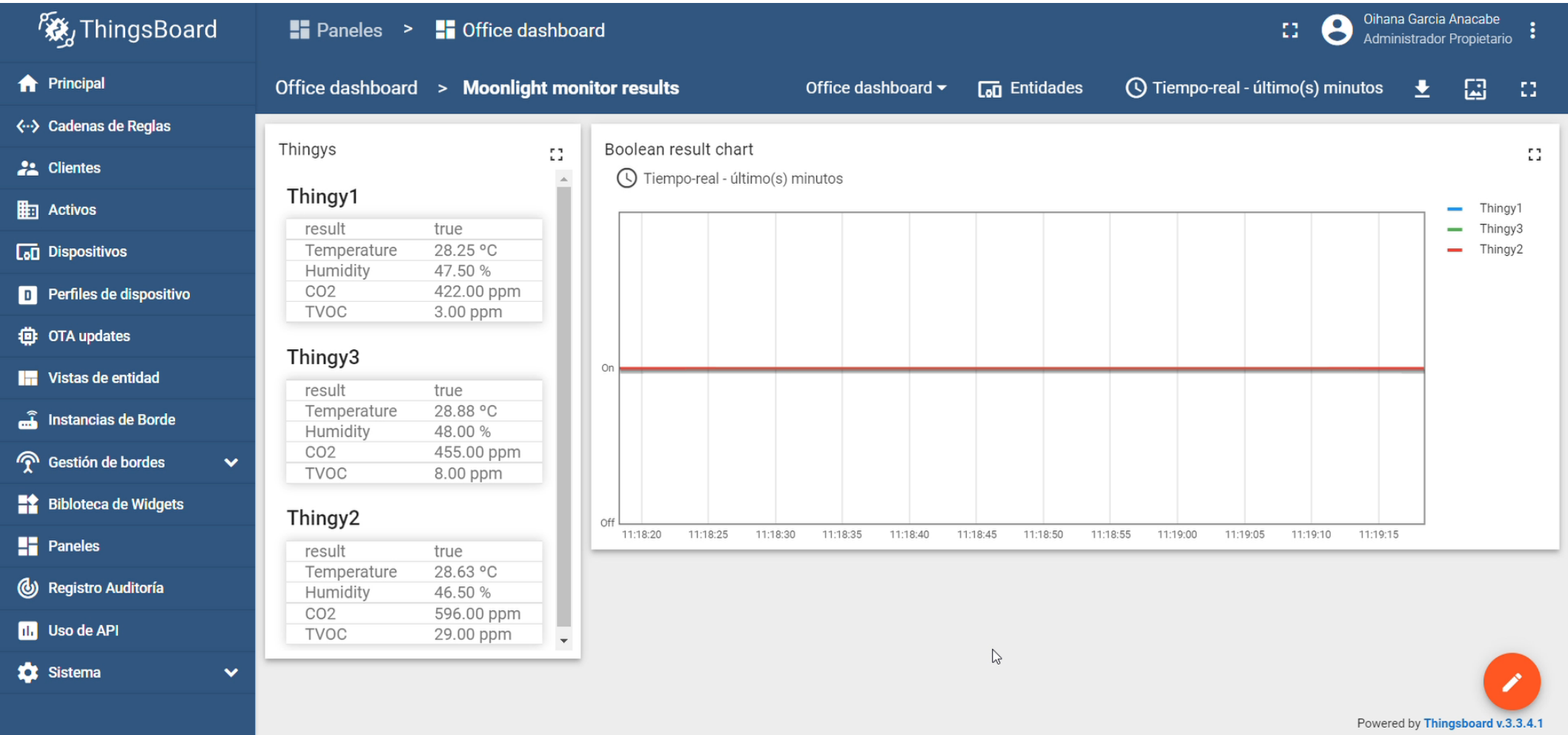
Segment(start=247788.0, value=[Interval: [false, true], Interval: [false, true], Interval: [false, true], Interval: [false, true], Interval: [false, true]])

```
MESSAGE: { 'id': 003, 'time': 247099, 'temp': 30.000000, 'hum': 44.000000, 'co2': 523, 'tvoc': 18}
MESSAGE: { 'id': 001, 'time': 247740, 'temp': 30.750000, 'hum': 41.000000, 'co2': 457, 'tvoc': 8}
MESSAGE: { 'id': 002, 'time': 247564, 'temp': 31.250000, 'hum': 40.000000, 'co2': 618, 'tvoc': 33}
MESSAGE: { 'id': 005, 'time': 246981, 'temp': 30.375000, 'hum': 41.500000, 'co2': 877, 'tvoc': 72}
MESSAGE: { 'id': 000, 'time': 248244, 'temp': 30.625000, 'hum': 40.500000, 'co2': 574, 'tvoc': 26}
```

# Results



# Results



# June prospect

---

- Client
  - Let the client customize the monitor's features
- Add another use case
  - Wiener Linien use case

# Conclusions

- Technical conclusions
- Methodological conclusions



# Technical conclusions

---

The proposed objectives are reached:

Moonlight monitor is fed with live data

The middleware is implemented

The communication from the sensors, to the monitor and to the dashboard works

# Methodological conclusions

---

Moonlight monitor has been implemented in real-life systems. Based on the use cases, Moonlight has been proven to be a suitable monitor for CPS.

- Currently the available tools for monitoring formal specifications are restricted to temporal properties
- It is a usable tool that can handle spatio-temporal properties in an online way

Thank you  
Eskerrik asko