



AN INFLUXDATA CASE STUDY

# How Worldsensing Uses InfluxDB to Make Cities Smart and Safe

**Albert Zaragoza**

Head of Engineering, Worldsensing

**Daniel Lazaro**

Software Developer, Worldsensing

**Fuad Mimoun**

Software Developer, Worldsensing



JANUARY 2019



## Company in brief

Worldsensing is a widely recognized global IoT pioneer. It provides customers with the tools to make the right operational decisions based on real-time intelligence. Its insights enable operators to understand the performance of distributed infrastructure, make predictions, improve efficiency and prevent disasters. Worldsensing uses wireless sensor technology and real-time software solutions to provide operational intelligence to operators and decision makers. This is what Worldsensing calls Connected Operational Intelligence.

With offices in Barcelona, London, and Los Angeles, Worldsensing is globally active and works with customers in more than 50 countries across 5 continents, reinventing the world we will live and work in, tomorrow.

## Case overview

Worldsensing needed to build a real-time platform that would integrate data from all its products to deliver *Connected Operational Intelligence* to cities and traditional industries and enable them to act in real time and predict anomalies.

Using InfluxDB, the Worldsensing engineering team built OneMind, an end-to-end IoT solution: for traffic flow management, smart parking, emergency & security response, and critical infrastructure monitoring. OneMind is an integrated real-time platform featuring data science and a variety of data sources including InfluxDB, a time series database to store a city's critical IoT time series data.

Worldsensing uses InfluxDB to collect and store sensor readings from traffic light controllers, smart parking meters, and other geotechnical sensors. The TICK Stack is also used for infrastructure monitoring as well as prototypes and innovation projects. Since OneMind requires real-time data analysis for distinct use cases, Worldsensing also tested the TICK Stack for generic time series data, alerts, aggregations, and anomaly detection. Through OneMind platform, Worldsensing is building solutions for its customers that make cities smarter and safer.



OneMind platform's metrics and analytics powered by the TICK Stack

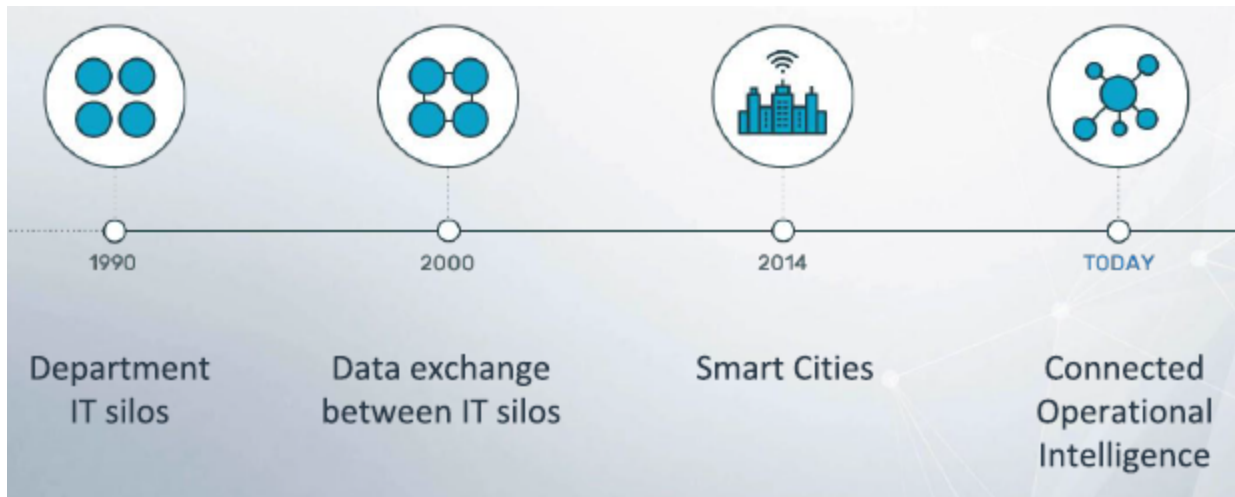
*"At Worldsensing, we build end-to-end solutions. We understand the pain point challenges that are present at each level of the IoT."*

*Albert Zaragoza, Head of Engineering, Worldsensing*

## The business problem

Worldsensing transforms operations in cities, mines, infrastructure, and construction sites with real-time intelligence – what it calls achieving Connected Operational Intelligence. As public and private entities digitize their operations, they need to rely on a data strategy to ensure all the necessary information is available. Most organizations – from smart cities to construction sites to private entities – have operated in silos and haven't shared any information. Worldsensing helps organizations transition to Connected Operational Intelligence by providing the data, efficiency and tools to optimize their operation and response capability.

### Shift from Data Silos to Connected Operational Intelligence



The best cities and construction sites are reinventing themselves by leveraging Connected Operational Intelligence to enable action based on real-time data and predict anomalies. Worldsensing achieves this for its customers through this five-step process.

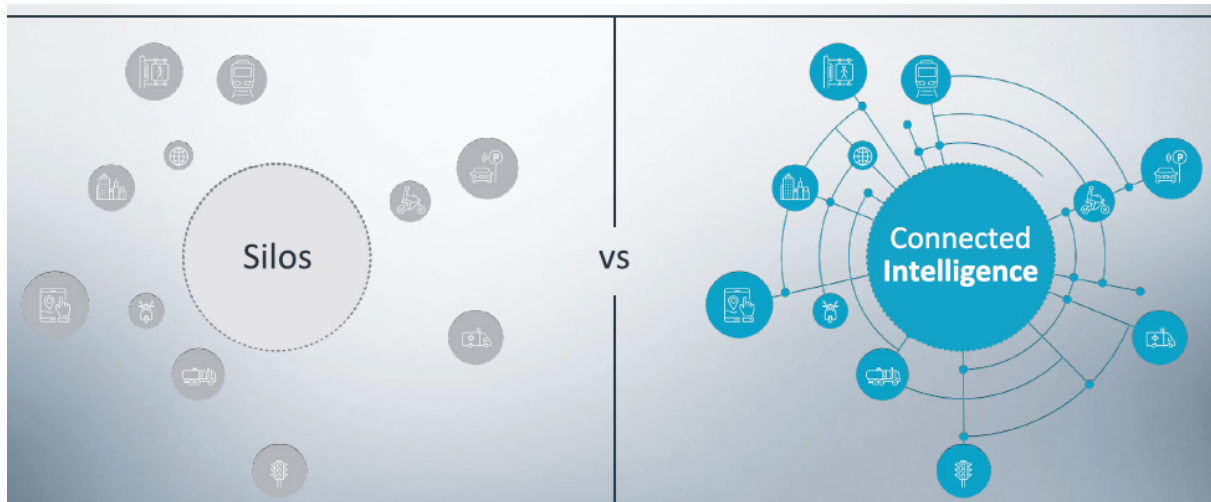
### Acting Based on Real-Time Data and Predicting Anomalies



The first two steps, digitizing and monitoring assets, are common in IoT. Yet to improve operations, organizations have to move into predicting, acting and engaging system operators and stakeholders. They need to have access to more real-time data to react faster. Once you've acted a few times, you are able to predict and engage with citizens.

### A New Way of Thinking



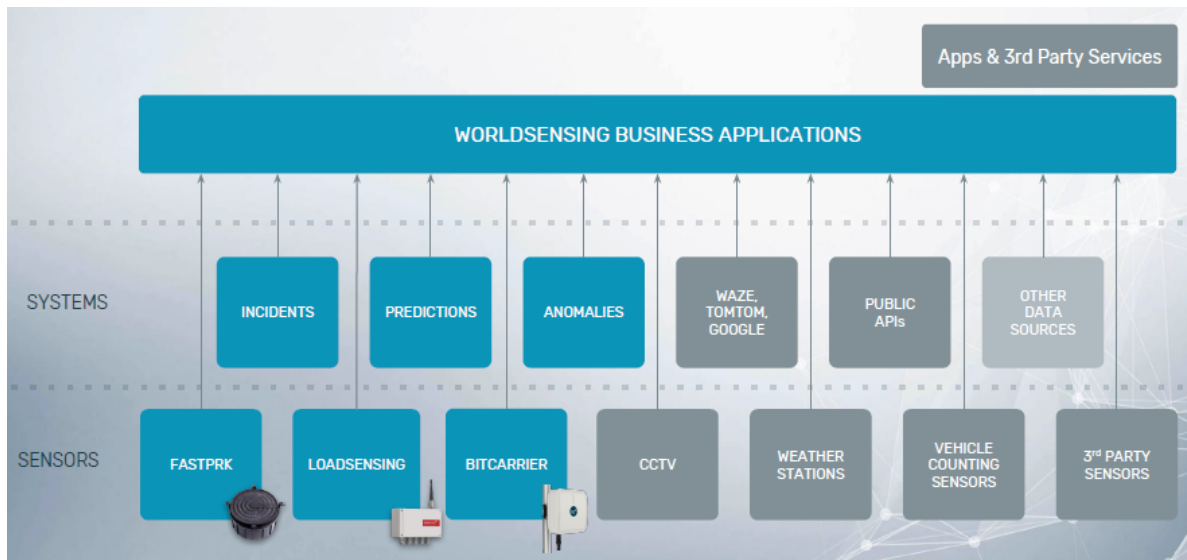


Yet not all industries and governments are prepared to leap into the new data strategies required to achieve Connected Operational Intelligence due to the complex architectures required. This is where the Worldsensing technology steps in to make cities smarter and safer by providing the tools and visibility for operators to make faster, more informed decisions.

## The technical problem

For over 10 years, Worldsensing was a hardware-based company manufacturing parking, infrastructure, and traffic sensors. They realized that data was key to building the predictions needed to empower cities to work more efficiently and recently augmented their hardware products with a software suite. They provide an offering that is both on-prem and via a SaaS, and both versions of the product have a microservices architecture where components run on dockers and independent modules.

### End-to-End Connected Operational Intelligence



As shown above, Worldsensing solution includes incident, prediction, and anomaly components. On top of that data, to help customers scale and better understand their data, Worldsensing builds business applications that integrate time series data from any third-party sensors – such as CCTV, weather stations, and vehicle counting sensors – and from other systems. Any API or any data in any format can be integrated. For example, Worldsensing has been:

- Monitoring mobility, parking, security, and traffic in 60 cities
- Collecting data from 10,000 sensors in 200 construction sites and 100 critical infrastructures
- Supporting decision-making in 6 major construction sites
- Ensuring that 50 mines are safe and maintained

Worldsensing needed to integrate all third-party sensor data together with the data from its sensors to build solutions for cities and infrastructures all over the world. This led to the need for a real-time platform capable of Data Science (to enable predictions) and of integrating various data sources including a time series database to collect and store sensor readings.

## Integrated platform for three products

The real-time data platform that Worldsensing sought to build – called OneMind – had to integrate data from the company’s three products: loadsensing, fastprk, and bitcarrier. Below is an overview of each, followed by an overview of OneMind platform.

## 1- Loadsensing

Worldsensing has traditional customers that were managing their infrastructures — bridges, railway stations, dams, mines — with cabled sensors. So Worldsensing developed loadsensing, a solution that is wireless, long range, and low-power monitoring system that helped revolutionize the Industrial IoT industry through its sensors.

### Using Industrial IoT Sensing Technology



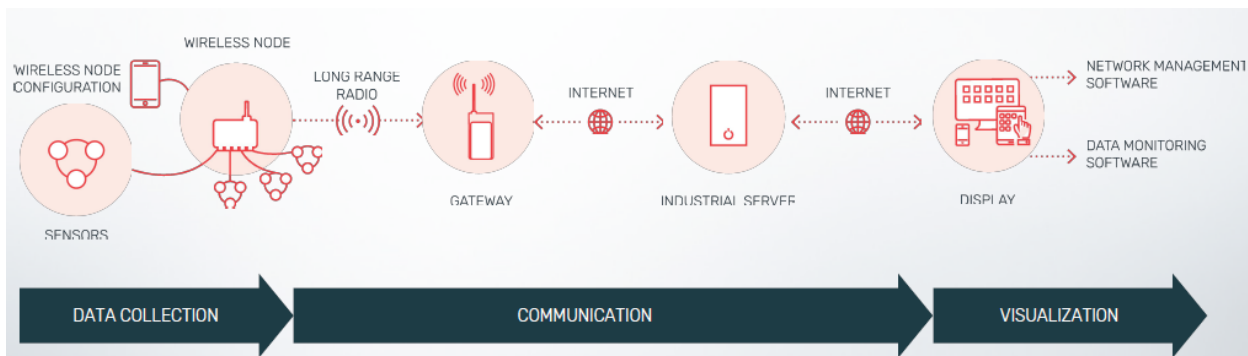
The data logger connects any third-party sensor and then helps customers communicate and build this data all up the stack so that they can detect infrastructural issues.

### Loadsensing System Components



The battery-powered, long-range, low-power devices are compatible with a wide range of geotechnical sensors. The system has a web-based software and mobile app that facilitate real-time data capture, analytics and alarm configuration.

### End-to-End System Architecture of loadsensing

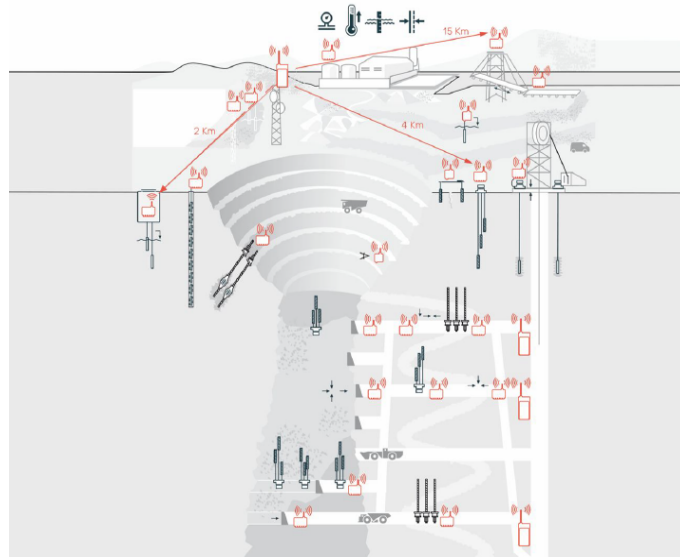


Here's how loadsensing helps cities and companies monitor their infrastructures over time:

- Third-party sensors measure different analog signals on infrastructure sites, and Worldsensing nodes connect directly to those sensors.
- Worldsensing sends the data over radio (LORA or SigFox) for this type of communication.
- Worldsensing put their software stack on their inhouse-built gateway that then communicates to the cloud, where Worldsensing have their monitoring software and network management for these sensors.

Below is a typical installation with their loadsensing sensors: a mining site where underground sensors monitoring landsite status are 200-300 meters deep. Outside the mines, in open air, many sensors with one gateway deliver data quickly to the cloud.

### How loadsensing Works in Mines and Dams

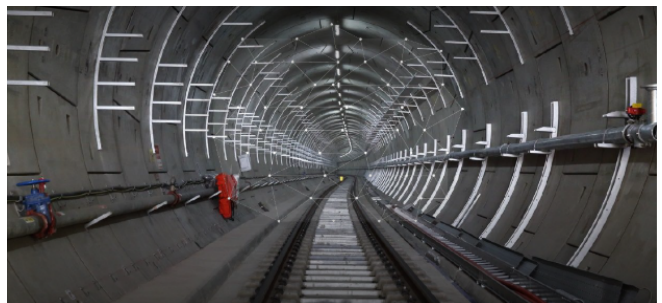


Worldsensing monitors sites such as the dam and tunnel shown below.

**Ponte Vecchio, Florence, Italy**



**Crossrail Tunnel, London, UK**



Worldsensing also monitors mine tailing dams (which pose environmental risks and make it critical to understand what's happening in real time) as well as rail network tracks.

**Mine Tailing Dams**



**Rail Network Tracks and Surroundings**



## 2- fastprk

Worldsensing builds fastprk sensors, which are wireless and low-power, to help cities understand outdoor parking usage. The digitized signal emitted enables real-time monitoring and helps cities instantly such as in payment enforcement for illegal parking; prediction of parking behavior; and engagement with citizens through apps and other systems.



fastprk, as all Worldsensing products, has an API enabling building applications on top of it.

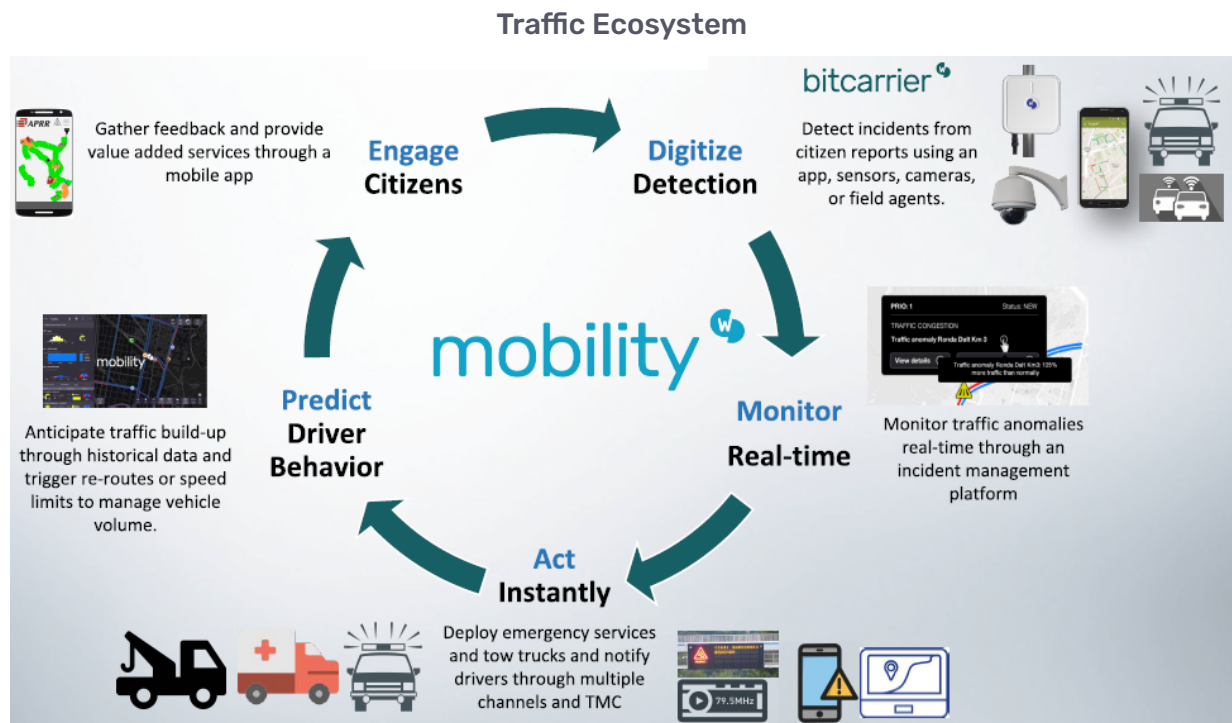
### How fastprk Works





### 3- Bitcarrier

Worldsensing builds bitcarrier sensors that track city traffic flow in real time. The sensors track Wi-Fi and Bluetooth signals from any car, as shown below.



## The solution

*“Overall, TICK Stack gives us this flexibility to build everything we need for real-time analysis.”*

*Albert Zaragoza, Head of Engineering*

### Why InfluxDB?

Worldsensing started thinking about time series databases years ago before InfluxDB became well-known. They first used relational databases. Yet upon being introduced to the TICK Stack by a team member who had previously used it, they adopted InfluxDB for infrastructure monitoring. Then they considered it for more use cases, and through trial and error adopted InfluxDB where it fit their specific use case. This approach resulted in how they deploy the TICK Stack today in OneMind.

Given its diverse products profiled above, Worldsensing has — in a given city — many systems and a large number of sensors capturing data from various sites. OneMind platform digitizes and unifies all the information so it can be presented to operators in smart city control centers to track performance, detect problems, and enable action.

**OneMind**



## YOUR SMART CITY APPLICATION

### OneMind<sup>®</sup> Core Technology



OneMind requires real-time data analysis for distinct use cases. Very particular to Worldsensing's architecture infrastructure is that most of their deployments are done in-house, in the data centers of each city or company, that they deploy for. There are many legal and other reasons why cities are interested in having the data as close as possible to themselves and also to make use of the data centers that they've built over time.

### An Integrated Solution for City Mobility Departments

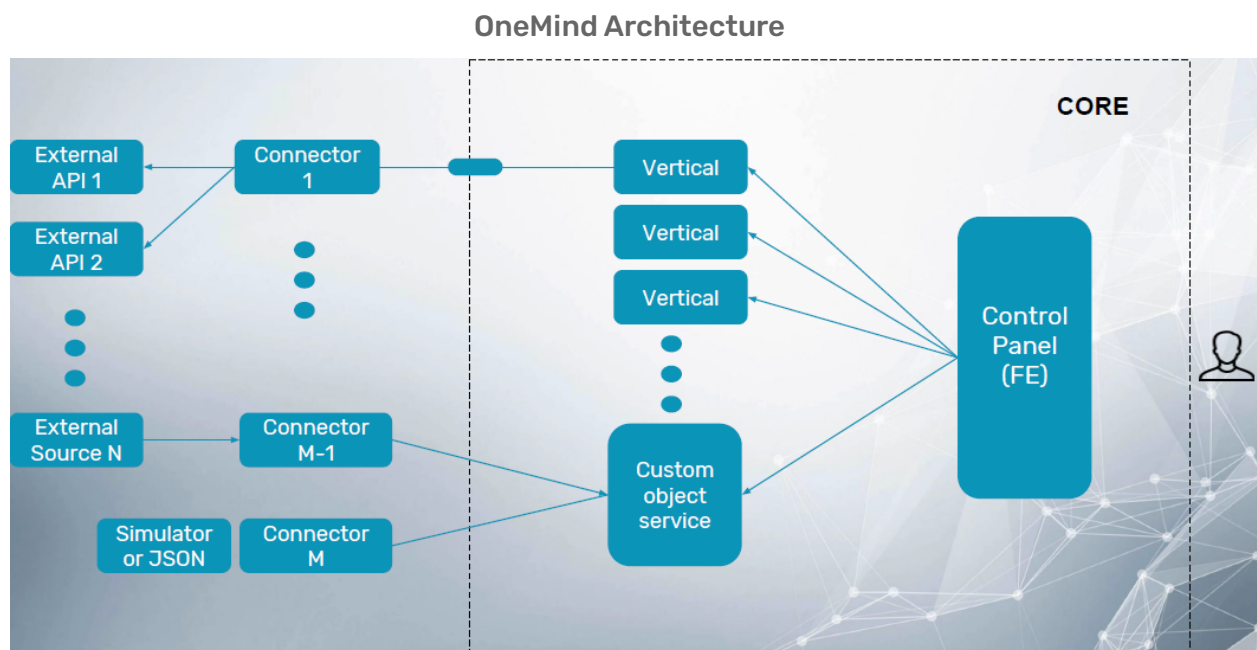


Deciding to leverage TICK Stack's flexibility, Worldsensing expanded its use to their metrics, analytics, and infrastructure's microservices, as shown in the below "Advanced Use Cases" section of this case study.

## Technical architecture

*"When we have in-house scenarios that we have to implement the tools inside of a data center, it's very useful to have a tool like TICK that can help us as well with the metrics and what's happening inside of the infrastructure."*

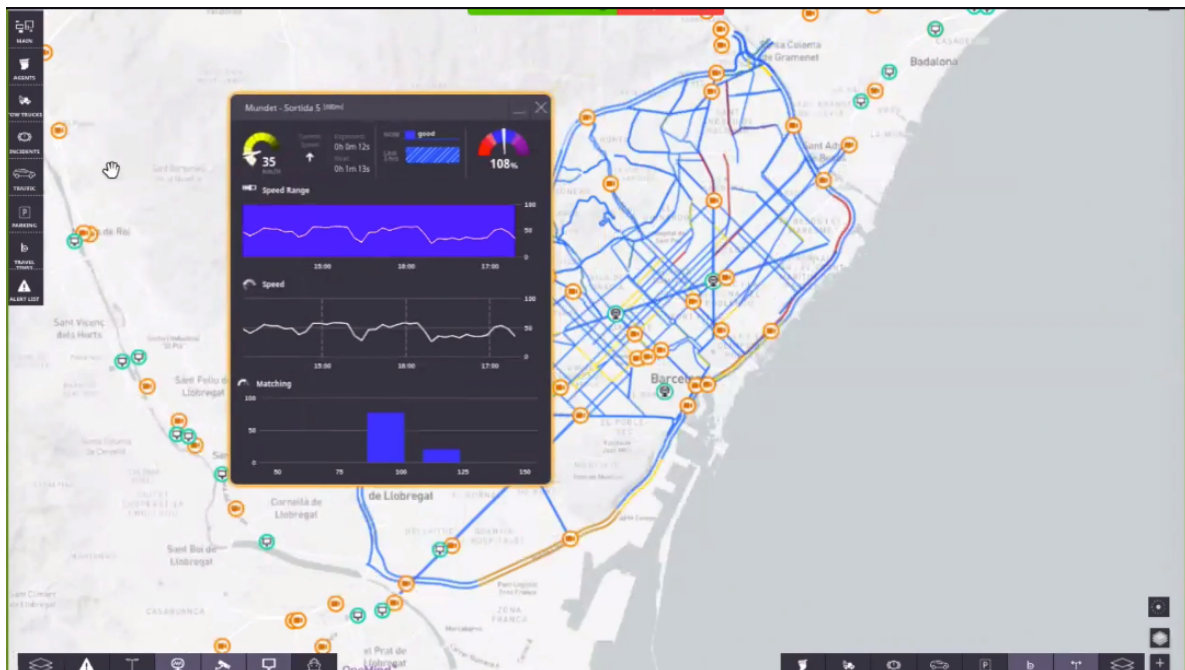
**Albert Zaragoza**, Head of Engineering



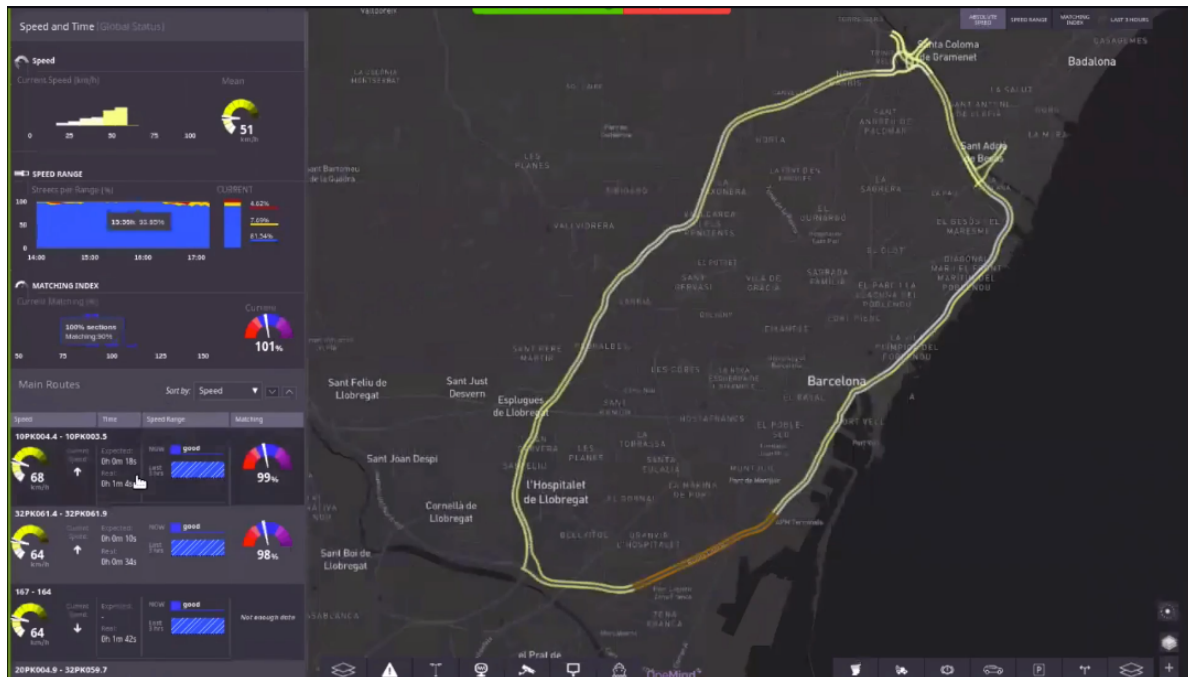
There are two types of data in OneMind: core verticals and custom data for each project. Connectors connect to each of the systems that generate or monitor the data and then insert the data into OneMind so that all data can be visualized together.

The main feature in OneMind's user interface is the map because most of the information is geo-localized. Operators want to know where things are happening. Different layers can be added to the map, and visualizations are customizable for each installation. For example, the map can show a city's traffic panel locations or track data from air quality controls, panels and cameras in real time. For the core verticals, WorldSensing can track traffic flow information, for example, and show its evolution over time as well as show computer KPIs and metrics.

### OneMind UI Screenshot



### Specialized View for Core Verticals in OneMind



The use of TICK Stack at Worldsensing progressed from infrastructure monitoring to software development. They first used TICK for monitoring Google cloud deployments; tens of docker containers per installation (they have a basic installation with docker compose and are now moving to Kubernetes for more scalable deployments); and machine and container resources. They visualized data with Chronograf.

Then they started using TICK for quick interfaces for prototypes and innovation projects where they needed to insert time series data in InfluxDB. They visualized data with Chronograf and Grafana. They also started implementing some simple processes, processing the data and triggering alerts with Kapacitor. Their data retention policies vary depending on each application of the TICK Stack.

## Advanced use cases of TICK in OneMind

Worldsensing tested using TICK Stack for parts of the core software — for the four advanced use cases in OneMind outlined below.

## 1. First use case: Generic time series data

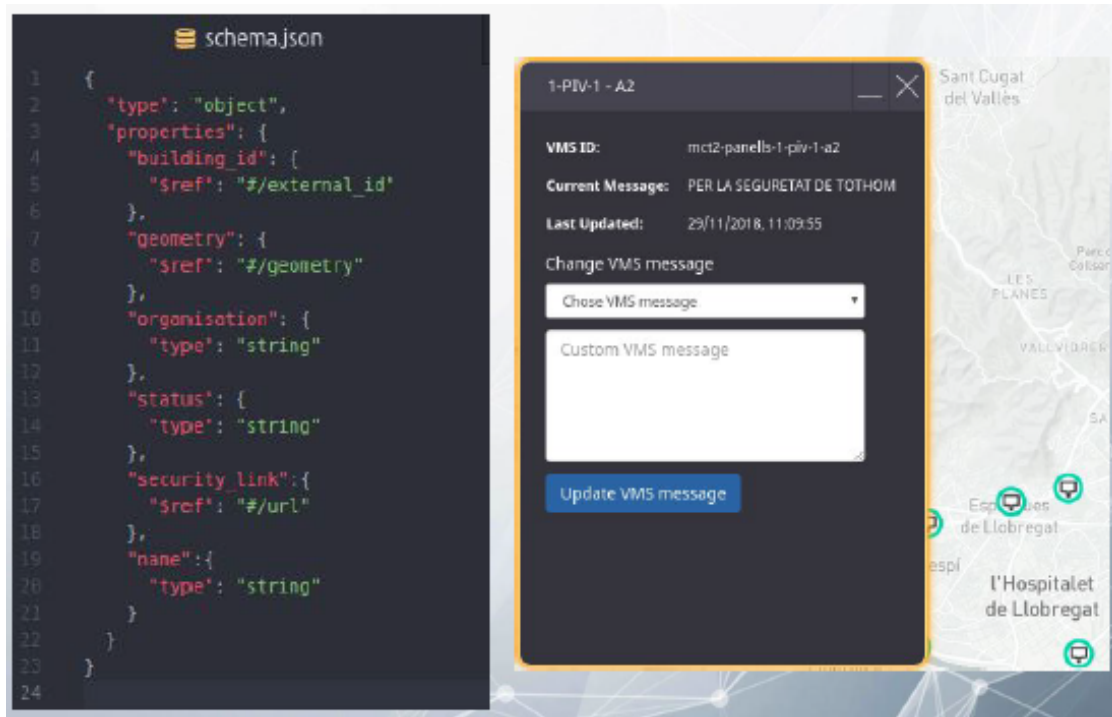
OneMind has two types of services: core verticals and custom data. The core verticals are fully independent microservices, each with functionality related to the specific domain (such as parking or city traffic service) and specific aggregations and visualizations. To enable these aggregations and visualizations, these microservices' APIs are fixed with mandatory fields. Worldsensing clients wanted to see more basic data, apart from the core verticals.

Worldsensing developed Custom Object Service (COS) – a service that allows them to insert static and generic data in OneMind in order to show geolocated data on their map and relate a popup visualization with current sensor data (such as tiltmeter inclination values in degrees, environment data, variable message panels, or CCTV cameras.) Worldsensing can locate this data on the map, and then the client can define which types of fields to see in the pop-up.

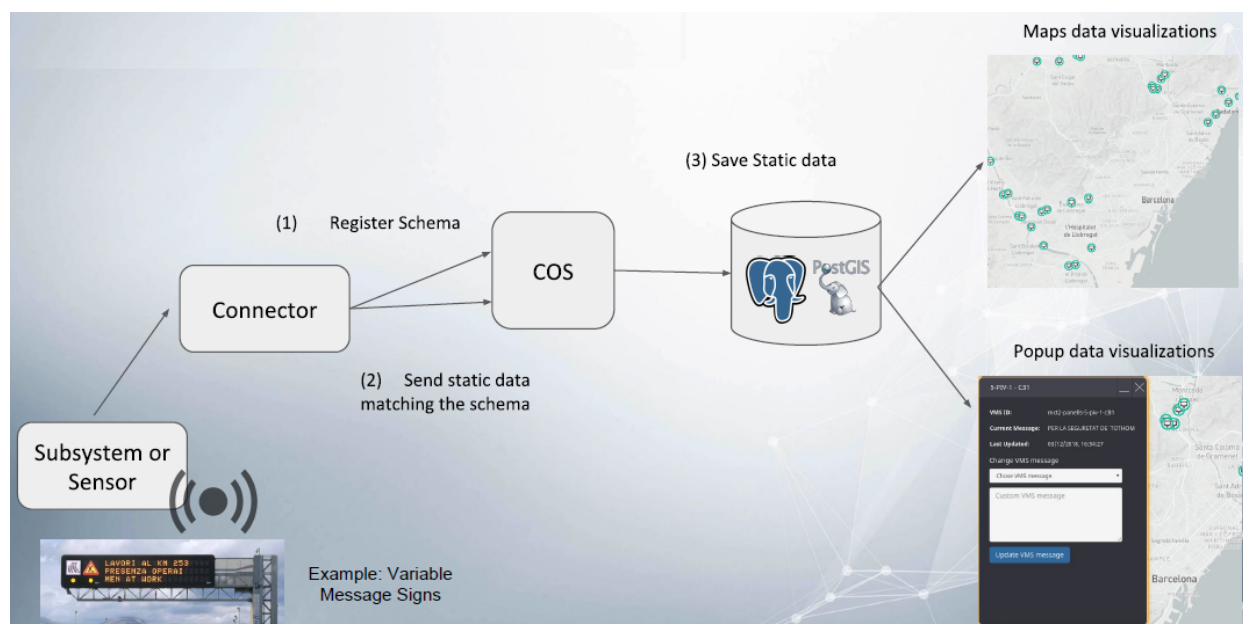
To enable that functionality, Worldsensing needed to define a connector between the sensor (or sub-system) and Custom Object Service. The connector is like a translator that allows these sub-systems to talk with Worldsensing's platform. The technical steps involved are:

1. Define the JSON schema for each data type
  - With custom types (dates, location, IDs...)
2. Insert data matching the schema
  - Current information, not historical data
  - Stored in PostgreSQL with JSON fields
3. Build custom visualization (for example, the locations or pop-up)

### **Defining a Connector Between the Sensor and COS**



## COS Inserting Static Data into OneMind

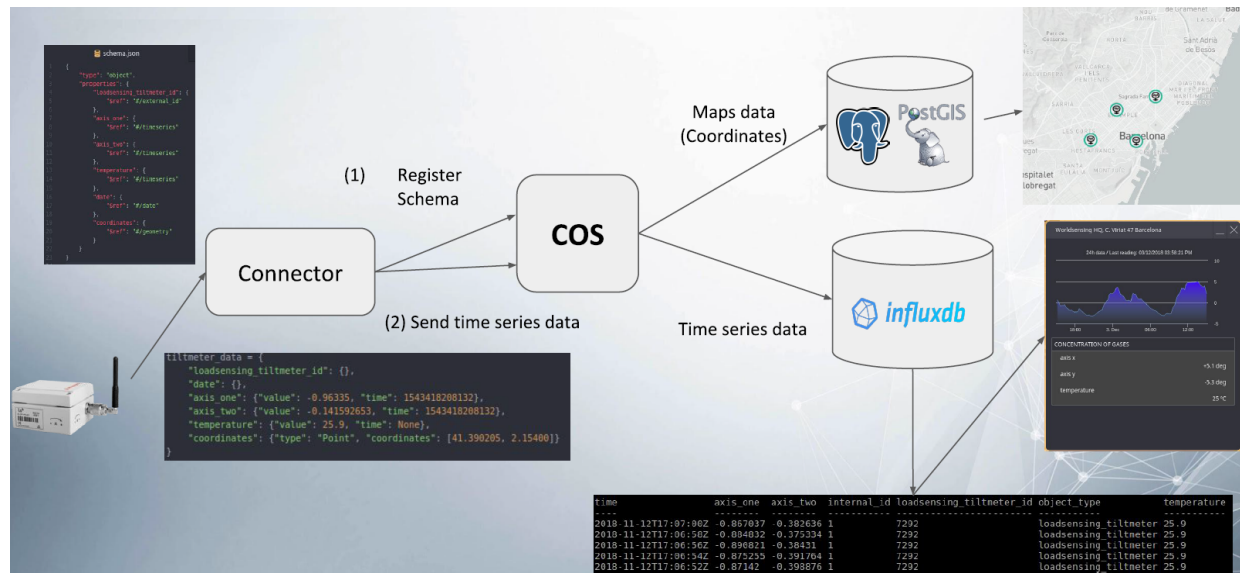


They also want to show generic time series data in their popups. For that, they integrated InfluxDB in their systems (adding time series data to COS by adding time series as a new custom data type), in an



architecture shown below. You can see how they store all the data in InfluxDB and visualize time series charts in their popups.

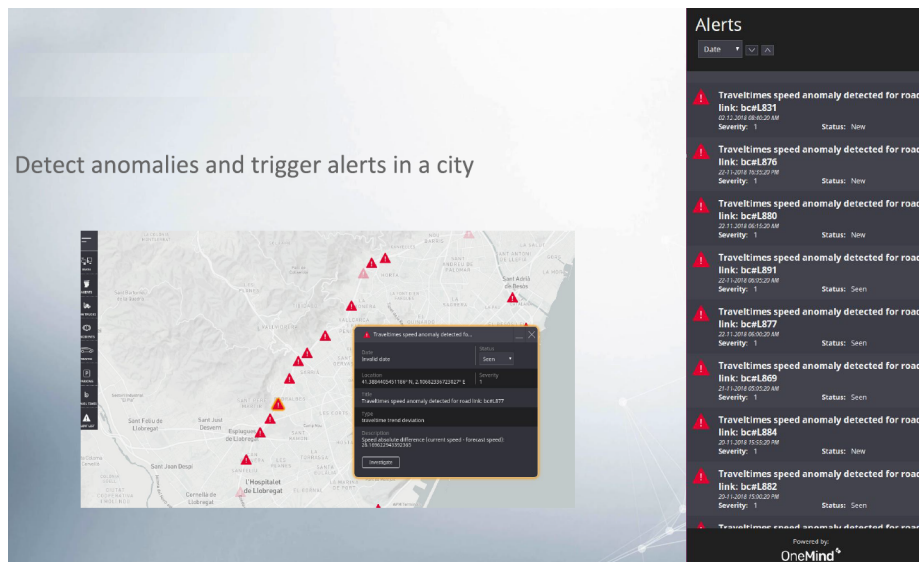
## Adding Time Series to COS



## 2- Second use case: Alerts

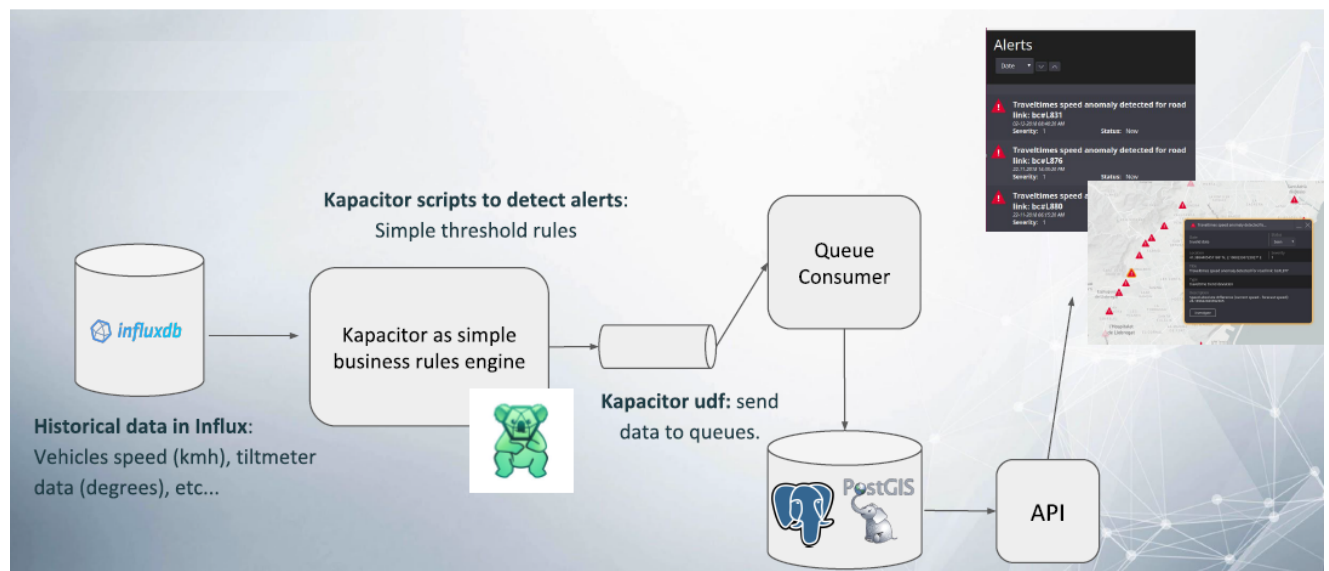
Saving historical data allowed Worldsensing to create alerting as a feature. They automatically detect anomalies using this data and trigger alerts based on thresholds. To configure thresholds, they used the task templates of Kapacitor.

## Detecting Anomalies and Triggering Alerts



Once they detect the alert, they send it to a queue (for which they implemented a UDFs from Kapacitor) and store it in Postgres database for later visualization in OneMind.

## Alerts Using User Defined Functions from Kapacitor



The threshold rule is configurable and can be altered depending on the use case.

## Alert Tasking and Defining the Threshold Value



```

1 dbrp "mb_staging_db"."autogen"
2
3 // axis_threshold
4 var option_threshold = 1.0
5 var option_item_type = 'tiltmeter'
6 var option_severity = 1
7 var option_rule_id = 'RULE#TILTMETER#1'
8 var option_rule_name = 'tiltmeter rule'
9
10 var rule = lambda: abs(float("axis_one")) > option_threshold OR abs(float("axis_two")) > option_threshold
11
12 var tiltmeter_alert = batch
13 |query('SELECT object_type, axis_one, axis_two, loadsensing_tiltmeter_id as item_id
14 FROM "mb_staging_db"."autogen"."measurements"')
15 .period(10s)
16 .every(2s)
17 .align()
18 .groupBy('item_id')
19 |where(lambda: "object_type" == 'loadsensing_tiltmeter') // COS
20 |where(rule)
21 |httpOut('alert')
22
23 tiltmeter_alert
24 @QueueSender()
25 .item_type(option_item_type)
26 .rule_id(option_rule_id)
27 .rule_name(option_rule_name)
28 .severity(option_severity)
29

```

```

1 alert_tiltmeter.yml
2
3 template-id: alert_tiltmeter
4 vars:
5   option_threshold:
6     type: float
7     value: 5.0

```

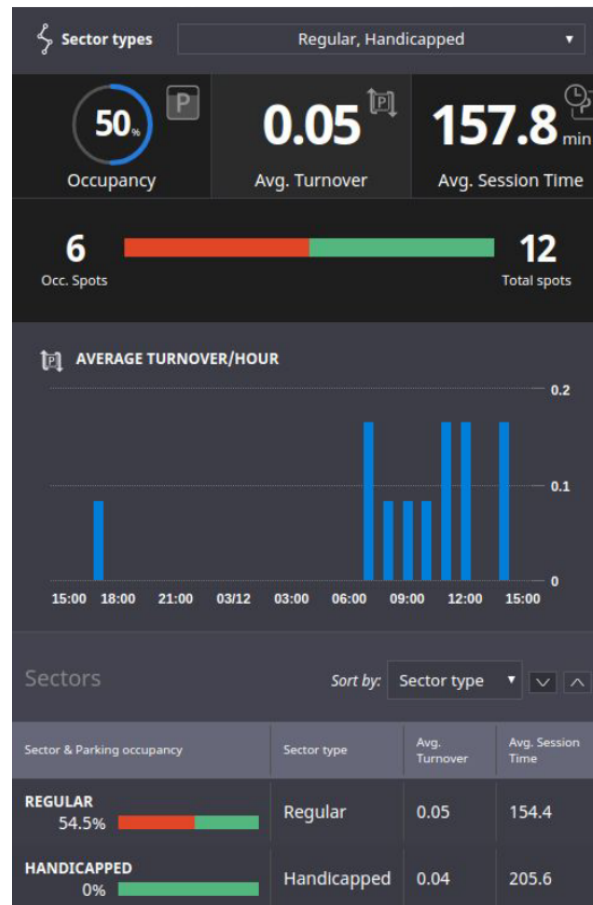
### 3- Third use case: Aggregations

Since one of WorldSensing's core verticals is parking, they wanted to integrate parking data into OneMind. They have the following parking KPIs:

- **Occupancy percentage** is how many parking spots are occupied.
- **Turnover** is how many cars park in a place for a time period.
- **Session time** is how long each car remains parked in a parking spot.

They wanted to aggregate data by area (Sector, District, City) and time (Hourly Average, Daily Average) to evaluate the parking situation and decided to compute that using the TICK Stack. Some capabilities were straightforward using InfluxDB, and some were not. Here are three such examples:

#### Aggregations for Parking Data



- **Historical data in InfluxDB - OK**  
They successfully stored both individual parking spot level occupations (occupied or free) and by sector (how many spots are free or occupied at some point for a given area).
- **Aggregation queries in InfluxDB - OK**  
They also found that querying on time on the aggregated data was easy, allowing the user to quickly get an hourly average of the parking turnover at a (aggregated) sector view.

```
SELECT sum(sum_session_time)/sum(turnover) as session_time
FROM parking_aggregated_sector_turnover
WHERE time >= {start} AND time <= {end}
GROUP BY time(1h) ORDER BY time ASC
```

- **Aggregations in Kapacitor - not OK**

Not all aggregation capabilities were that simple, and they found that some of the more complex aggregations were difficult to implement, and hard to debug. In the end, they chose to implement those aggregations with a set of cron jobs in Python and InfluxQL queries.

### The Script Resulting from Computing Session Time (not adopted since not consistently functional):

```
5 var durations = batch
6   |query('
7       SELECT elapsed("occupied", 1s)
8       FROM "set_db"."autogen".parking_set_name_sensor_occupation
9       ')
10  .period(48h)
11  .every(1m)
12  .align()
13  .groupBy('sensor_id', 'sector_id', 'sector_type')
14
15 batch
16 |query('
17     SELECT "occupied"
18     FROM "set_db"."autogen".parking_set_name_sensor_occupation
19     ')
20 .period(48h)
21 .every(1m)
22 .align()
23 .groupBy('sensor_id', 'sector_id', 'sector_type')
24 |join(durations)
25 .as('events', 'durations')
26 |where(lambda: "events.occupied" == FALSE)
27 |influxDBOut()
28 .database('set_db')
29 .measurement('parking_set_name_sensor_session_times')
30
31
32 batch
33 |query('
34     SELECT mean("duration") AS turnover
35     FROM "set_db"."autogen".parking_set_name_sensor_session_times
36     WHERE "occupied" = 0
37     ')
38 .period(1h)
39 .every(1m)
40 .align()
41 .groupBy(time(1h), 'sector_id', 'sector_type')
42 |influxDBOut()
43 .database('set_db')
44 .measurement('parking_set_name_sector_session_times')
45
46 batch
47 |query('
48     SELECT mean("duration") AS turnover
49     FROM "set_db"."autogen".parking_set_name_sensor_session_times
50     WHERE "occupied" = 0
51     ')
52 .period(24h)
53 .every(1m)
54 .align()
55 .groupBy(time(24h), 'sector_id', 'sector_type')
56 @QueueSender()
57 .type('parking_session_time')
58 .name('set_name')
```

Instead, they easily performed the aggregations using the power of InfluxDB from their Python script. The below script was adopted and includes only three queries:

- Computing the time that each occupation lasts with the `ELAPSED()` function of InfluxQL
- Filtering the ones set to "occupied" = FALSE (when a car left)

- Calculating the average of the above

### The Script Adopted: Comprising Only 3 Queries

```
INITIAL_AGGREGATION = ""
SELECT elapsed("occupied", 1s) as "duration"
INTO {sensor_occupations} FROM {sensor_occupations}
WHERE time >= {end} - {range} AND time < {end}
GROUP BY sensor_id, sector_id, sector_type
""

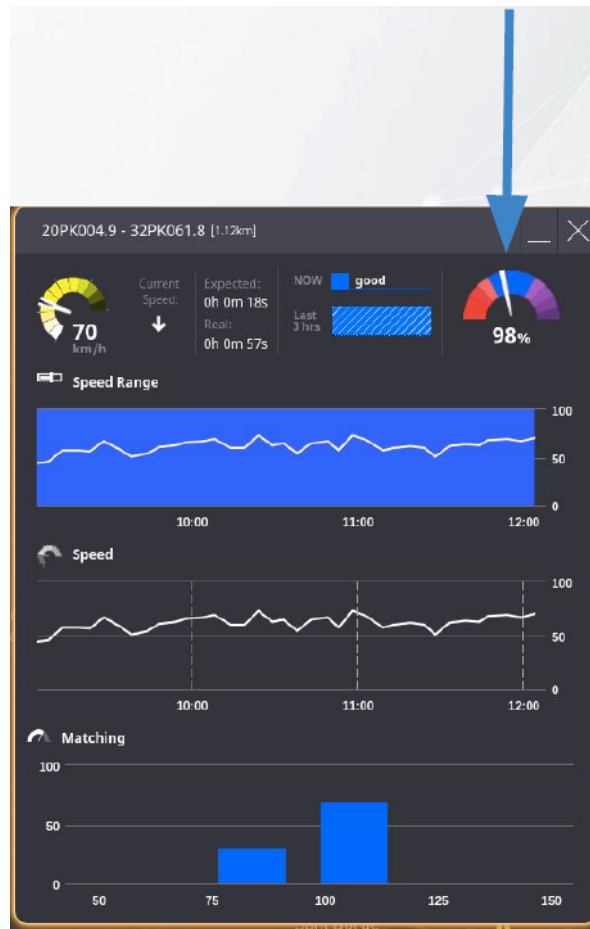
HOURLY_AGGREGATION = ""
SELECT mean("duration") as session_time,
sum("duration") as sum_session_time
INTO {sector_occupations}
FROM {sensor_occupations}
WHERE time >= {end} - {range} AND time < {end}
AND "occupied" = FALSE
GROUP BY sector_id, sector_type, time(1h)
""

DAILY_AGGREGATION = ""
SELECT mean("duration") as session_time
FROM {sensor_occupations}
WHERE time >= {end} - {range} AND time < {end}
AND "occupied" = FALSE
GROUP BY sector_id, sector_type
""
```

## 4- Fourth use case: Anomaly detection

Worldsensing wanted to inform operators whether a given city system is functioning normally. The indicator shown below is a percentage close to 100%, indicating proper performance. An indicator in the red area would mean problems potentially requiring corrective action. They can automate such anomaly detection using the previously outlined alert system.

## Detecting Anomalies in Traffic Data



For example, detecting anomalies in traffic data requires the below three steps:

### 1. Develop custom forecast algorithm based on historical data

They tried anomaly detection in Kapacitor (using the Holt-Winters prediction method, an algorithm used to forecast data points in a series, provided that the series is repetitive over some period). Yet this method is based on recent past values, while they were computing in this case historical data (values for the next day of the week, same day of the week, same hour, etc.).

They discarded implementing the forecast algorithm in Kapacitor because:

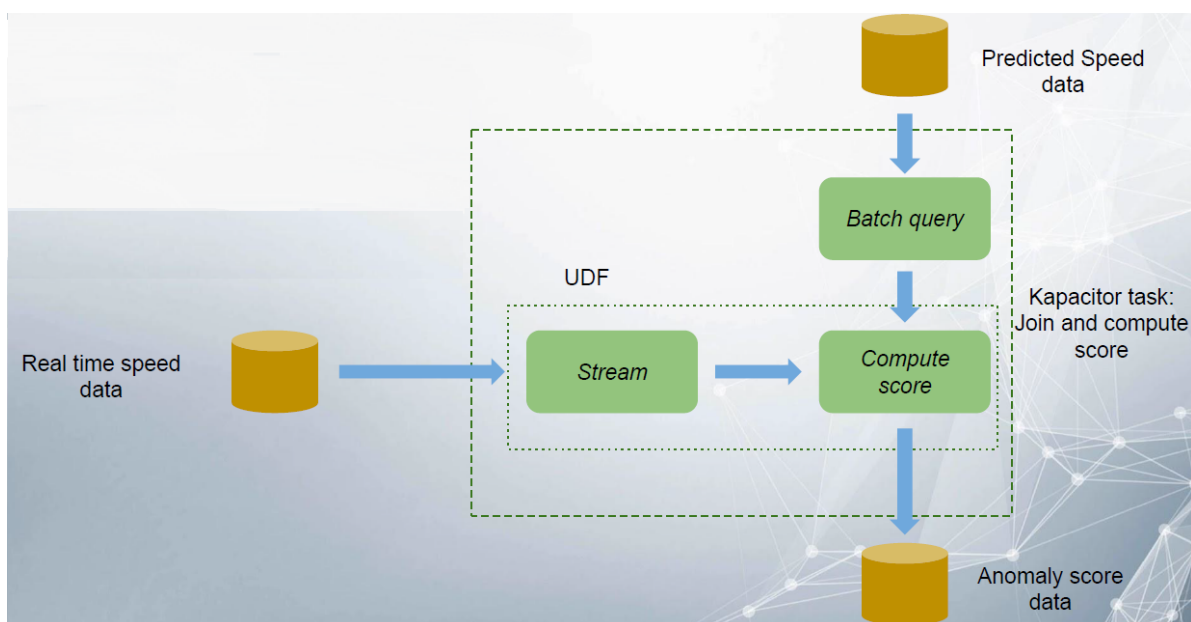
- It's based on historical data, so they can predict in batch values for the next day, week...
- It was too complex to put it as a UDF (since they wanted to use Pandas, scikit)

In the end, they decided to store real-time speed data in InfluxDB and then do a cron job that queries InfluxDB, performs a forecast, and stores it again in InfluxDB.



## 2. Compare real-time values to forecast (to compute trend deviation/anomaly score)

- Needed to read two data sources (real and predicted)
- Couldn't get them to join in a TICKscript (querying Kapacitor, you can join two data streams or two batch queries, but not one of each)
- Implemented as UDF that receives a stream and that queries InfluxDB from Python

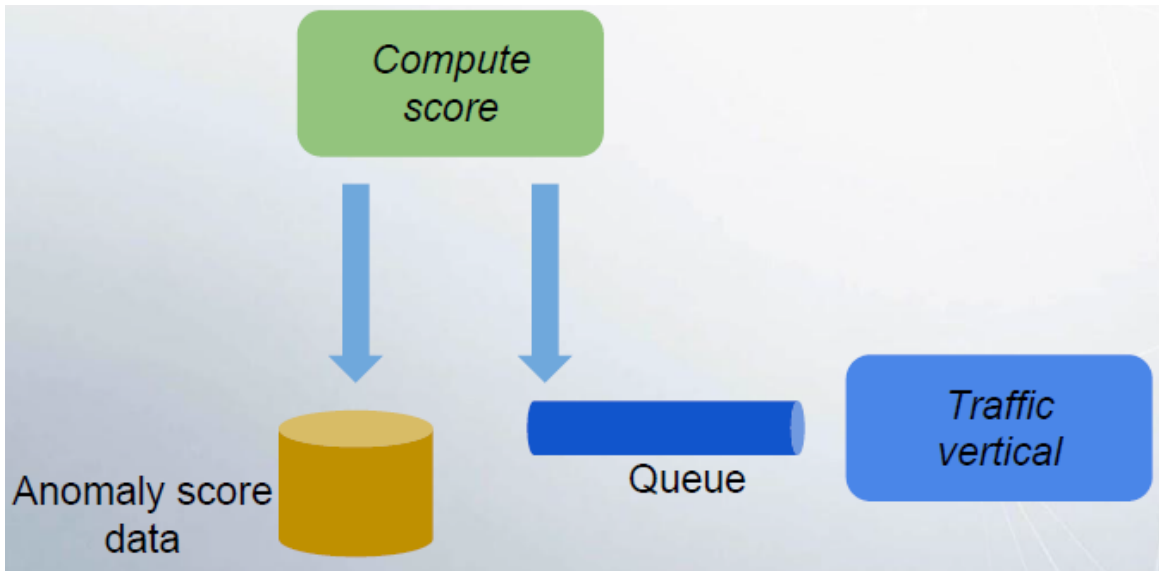


## 3. Compute "anomaly score"

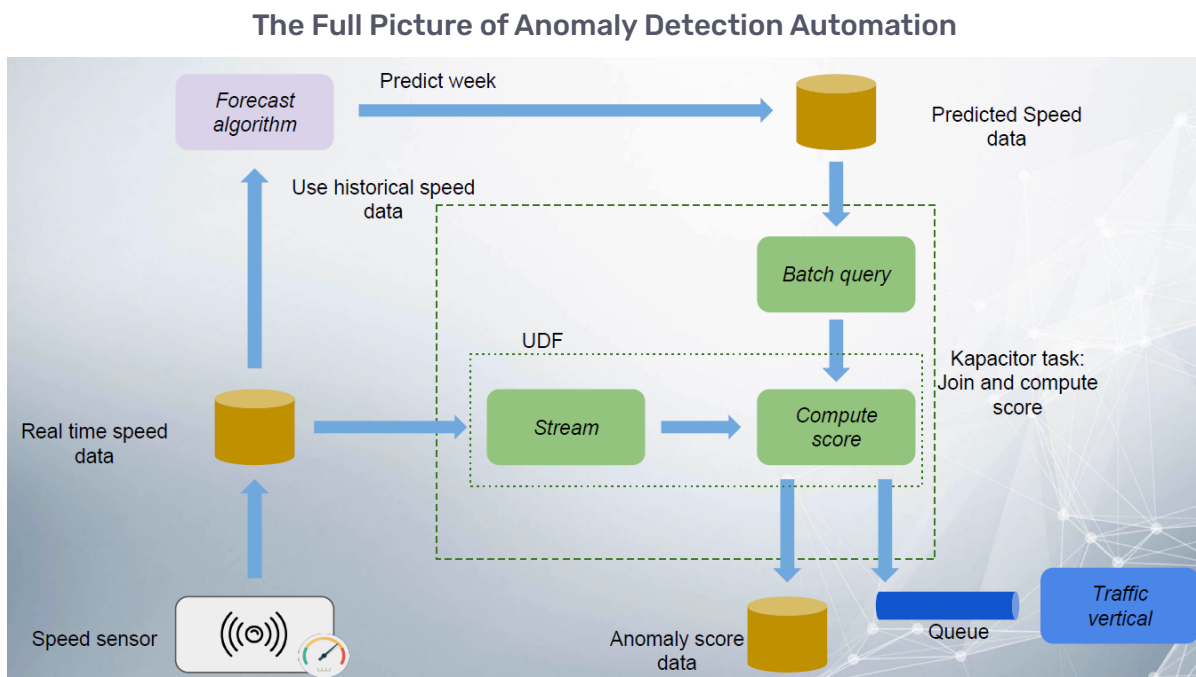
Once the comparison is completed, it allows them to compute the anomaly score data and store it in InfluxDB which can be done easily from Kapacitor. One of Kapacitor's strengths is that you can easily implement a UDF and put values into a queue so you can build whatever output you want for the system very easily.

They send computed values to a queue:

- Same UDF as for alerts
- Can be taken by the Traffic Vertical or other services in the system



The three anomaly detection automation steps outlined above are brought together in this chart:





## What's next for Worldsensing?

They are still working on the alerts functionality. They played with Flux, InfluxData's new scripting and query language, and intend to keep an eye on it because it looks very promising in terms of resolving the shortcomings they encountered in Kapacitor for some of their particular use cases.

## Results

*"The amount of data you collect, how precise you want it to be, when you want to expire it. These are things to consider when building out your solutions because the TICK Stack offers the flexibility to make those changes."*

***Albert Zaragoza, Head of Engineering***

After trying InfluxDB and Kapacitor in several application-level use cases, Worldsensing concluded:

### Pros:

- InfluxDB is great for time series queries.
- Kapacitor is good for simple triggers (alerts, etc.) and simple processing of data streams.
- UDF's provide a lot of flexibility.

### Cons:

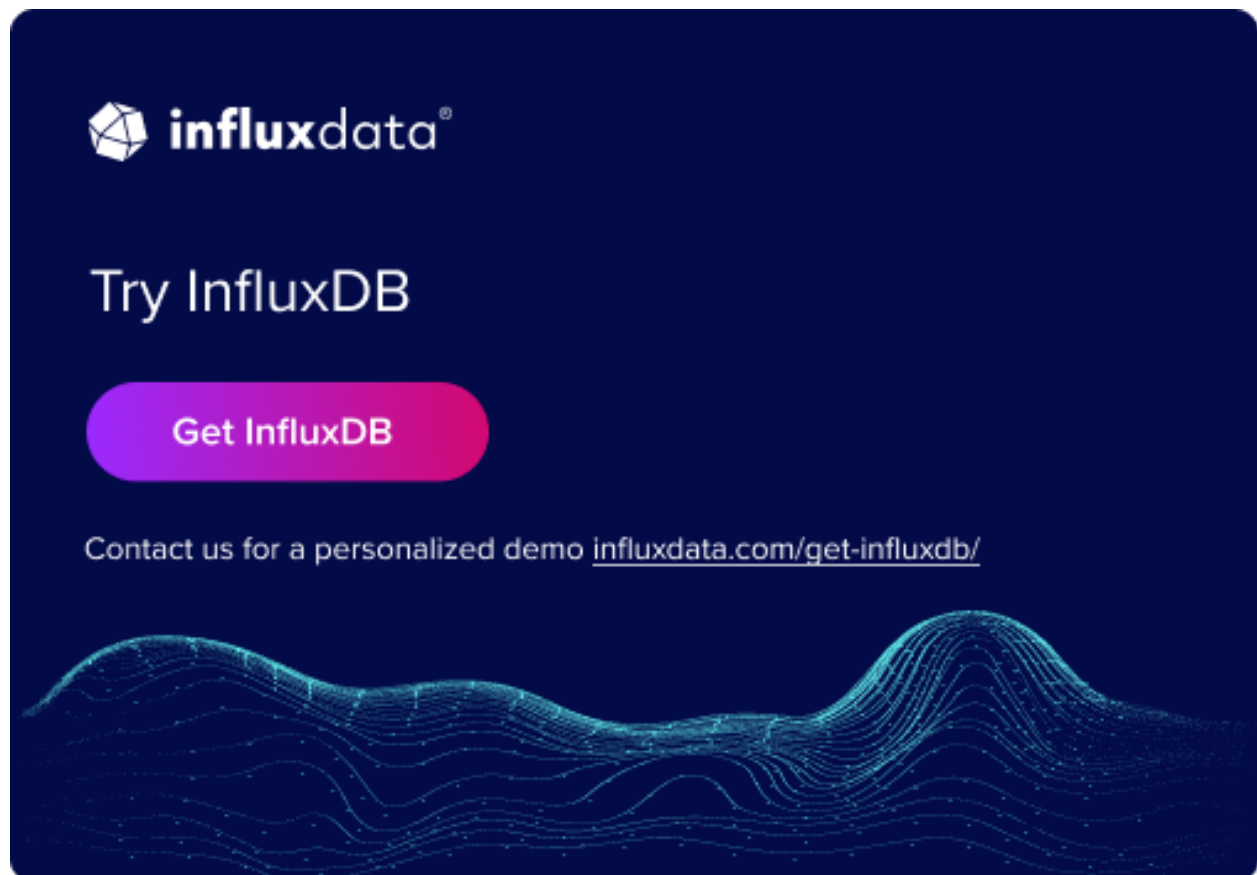
- Kapacitor is hard to test and debug, particularly with long and complex TICKscripts.
- Using Kapacitor as a scheduler to query InfluxDB and process in Python isn't worth it.

Using the TICK Stack for time series storage, processing, management, and visualization, Worldsensing is providing its customers with the data they need to optimize their operations and is thereby fulfilling its mission of delivering Connected Operational Intelligence.



# About InfluxData

InfluxData is the creator of InfluxDB, the leading time series platform. We empower developers and organizations, such as Cisco, IBM, Lego, Siemens, and Tesla, to build transformative IoT, analytics and monitoring applications. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by sensors, applications and computer infrastructure. Easy to start and scale, InfluxDB gives developers time to focus on the features and functionalities that give their apps a competitive edge. InfluxData is headquartered in San Francisco, with a workforce distributed throughout the U.S. and across Europe. For more information, visit [influxdata.com](https://influxdata.com) and follow us [@InfluxDB](https://twitter.com/InfluxDB).

A promotional banner for InfluxData with a dark blue background. At the top left is the InfluxData logo, which consists of a white geometric cube icon followed by the text "influxdata" in white lowercase letters with a registered trademark symbol. Below the logo, the text "Try InfluxDB" is written in a large, white, sans-serif font. Underneath this is a prominent, rounded rectangular button with a pink-to-purple gradient, containing the white text "Get InfluxDB". Below the button, the text "Contact us for a personalized demo" is followed by the URL "influxdata.com/get-influxdb/" in white. The bottom of the banner features a decorative graphic of glowing, wavy lines in shades of blue and green, resembling a data visualization or a stylized landscape.

548 Market St. PMB 77953, San Francisco, CA 94104