.DATAINFLUX

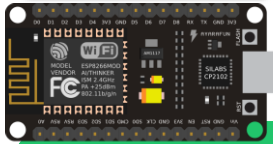
[AN IoT OVER .FLUIDITY CENTRALIZED MONITORING SENSOR SYSTEM]

.dataInflux

Technical Lead: Charalampos Kapolonaris
Software Tester: Vasilios Koutlas

What is .dataInflux ?

.dataInflux is an IoT centralized sensor monitoring system that utilizes .Fluidity's IP/VPN capabilities to build a flexible, yet robust data network.



The sensors are SONOFF devices, based on the ESP8266 chip.

- This chip allows wireless networking connectivity to the .dataInflux clients, that act as Access Points and collect the data locally.
- The default stock firmware is changed to an alternative firmware, known as **Tasmota**.

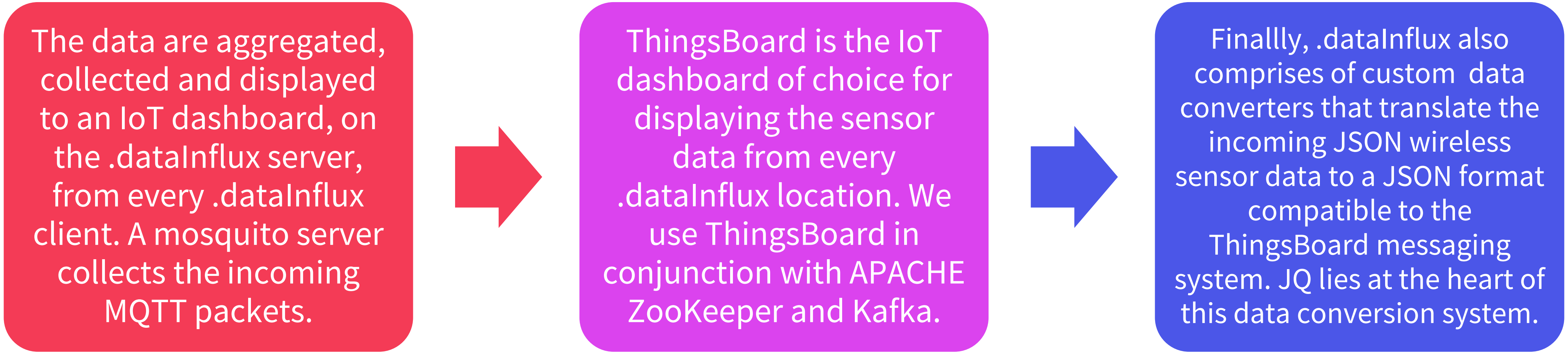
The .dataInflux clients are Raspberry Pis, with the additional role of also being .Fluidity clients. A secure IP/VPN connection is created to the .dataInflux server, over the ISP infrastructure, to securely transmit the data.

.dataInflux / Fluidity

What is .dataInflux ?

.dataInflux

ThingsBoard



The data are aggregated, collected and displayed to an IoT dashboard, on the .dataInflux server, from every .dataInflux client. A mosquito server collects the incoming MQTT packets.

ThingsBoard is the IoT dashboard of choice for displaying the sensor data from every .dataInflux location. We use ThingsBoard in conjunction with APACHE ZooKeeper and Kafka.

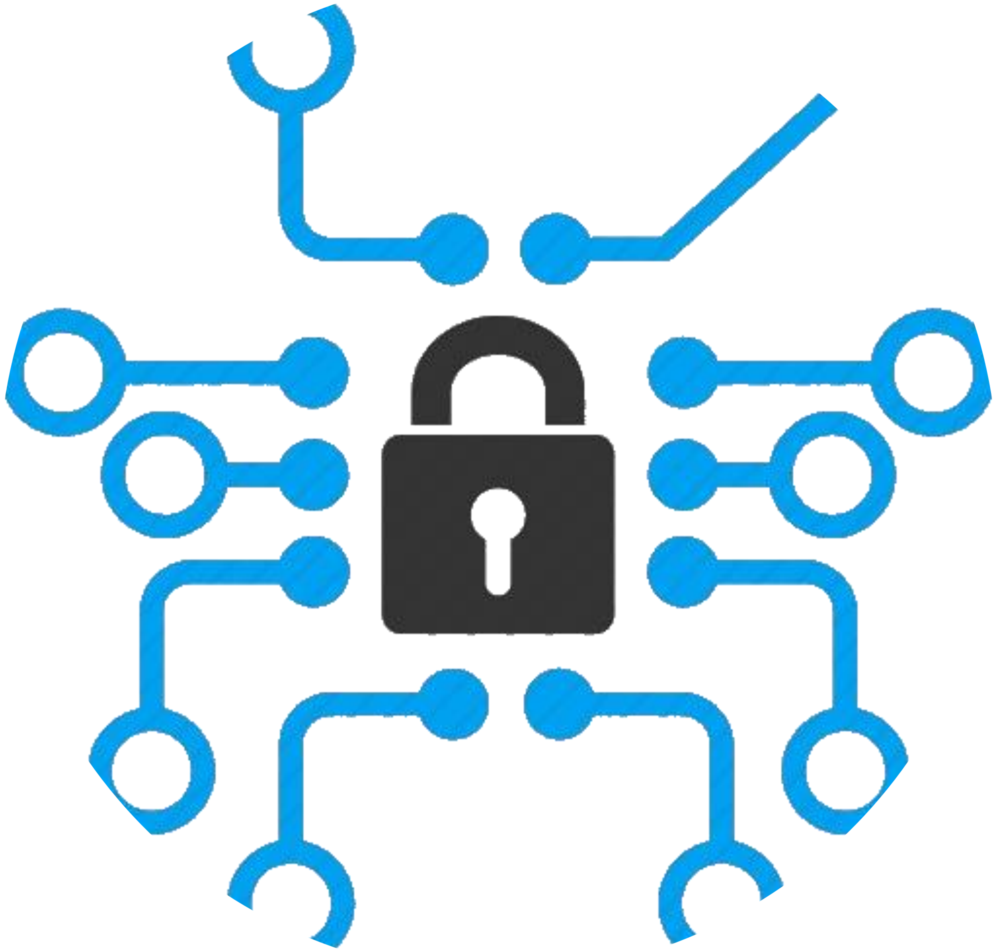
Finallly, .dataInflux also comprises of custom data converters that translate the incoming JSON wireless sensor data to a JSON format compatible to the ThingsBoard messaging system. JQ lies at the heart of this data conversion system.

mosquitto

APACHE ZooKeeper™ APACHE kafka® A distributed streaming platform

./jq

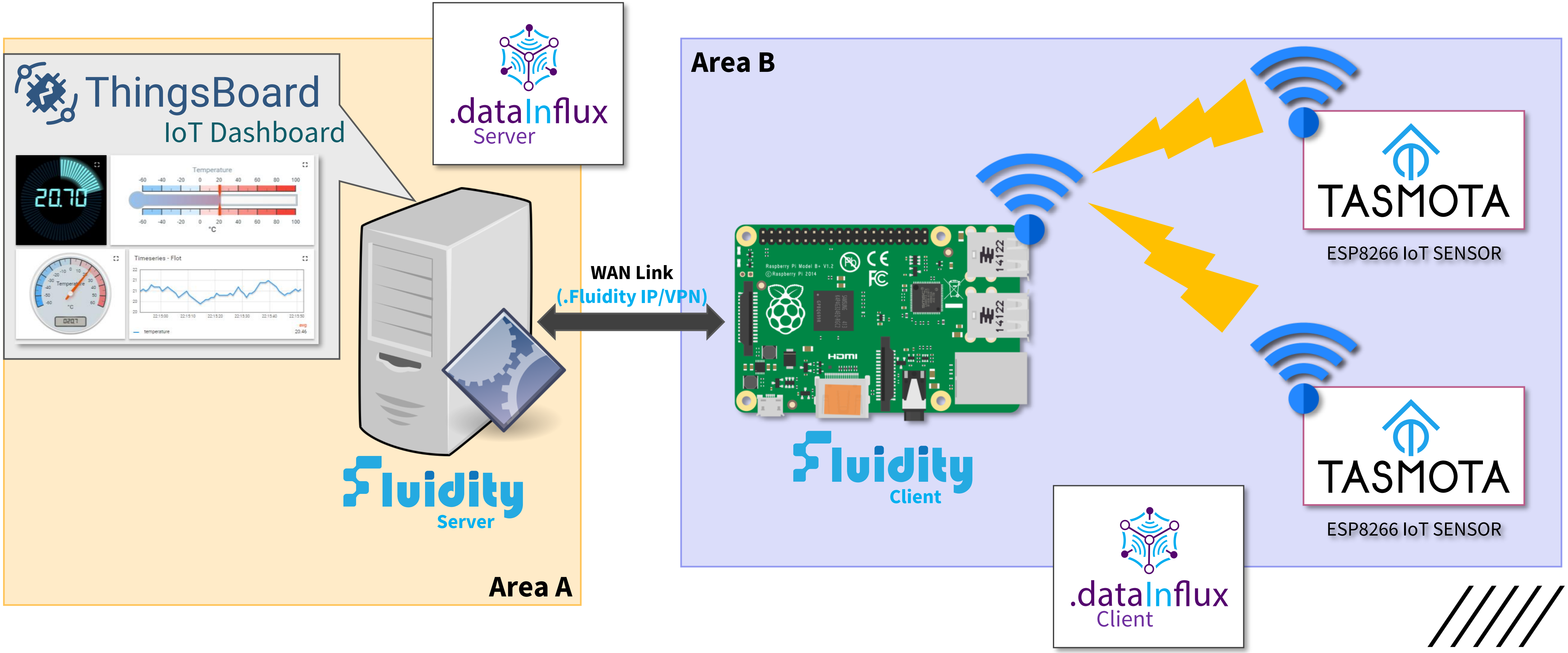
What is .dataInflux ?



.dataInflux, as opposed to “cloud computing”, champions the notions of “private networking” and “local hosting”.

We move to the opposite side of the spectrum, because we believe that maximum security, for mission critical IoT applications, can only be achieved in a closed IP/VPN environment.

.dataInflux / The Basic Concept



ThingsBoard
IoT Dashboard

.dataInflux
Server

WAN Link
(.Fluidity IP/VPN)

Fluidity
Server

Area A

Area B

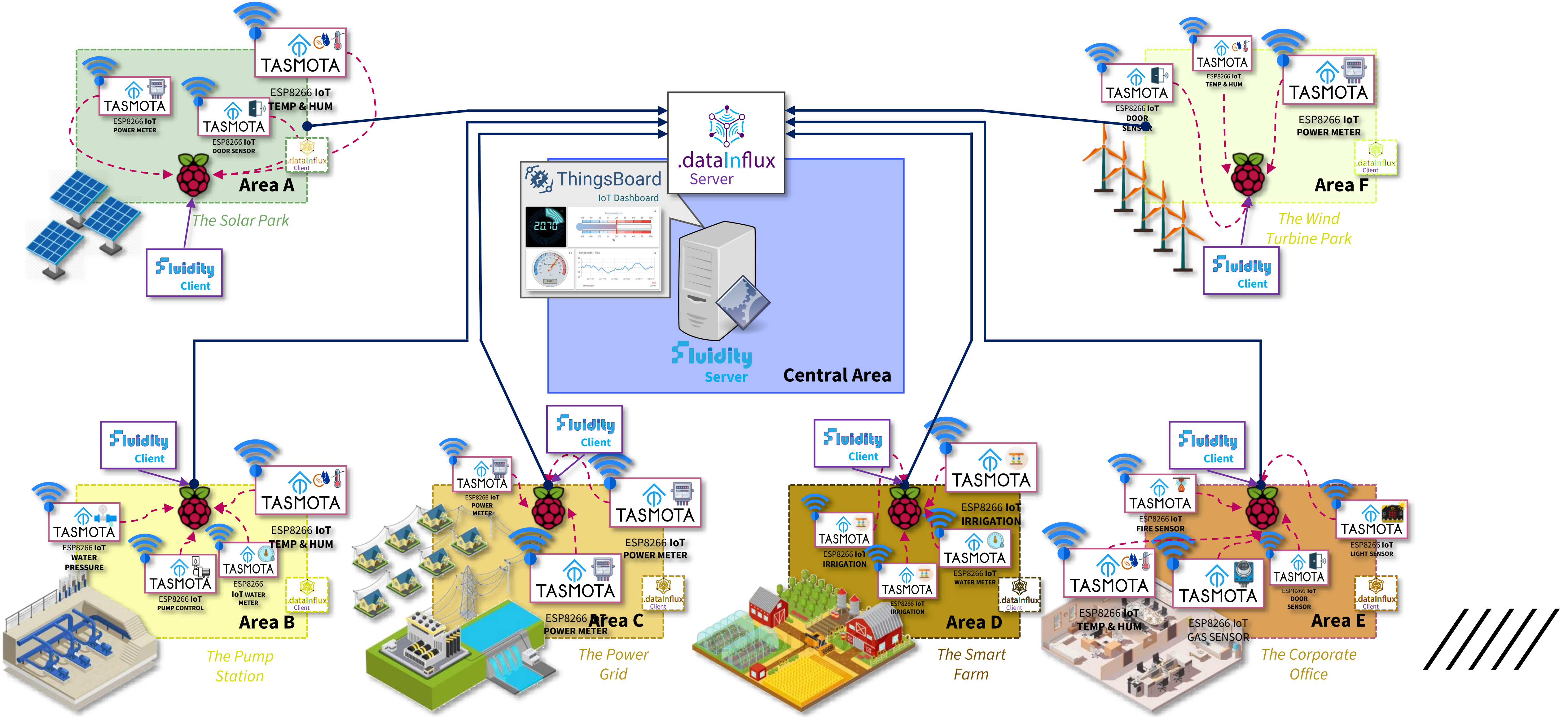
Fluidity
Client

TASMOTA
ESP8266 IoT SENSOR

TASMOTA
ESP8266 IoT SENSOR

.dataInflux
Client

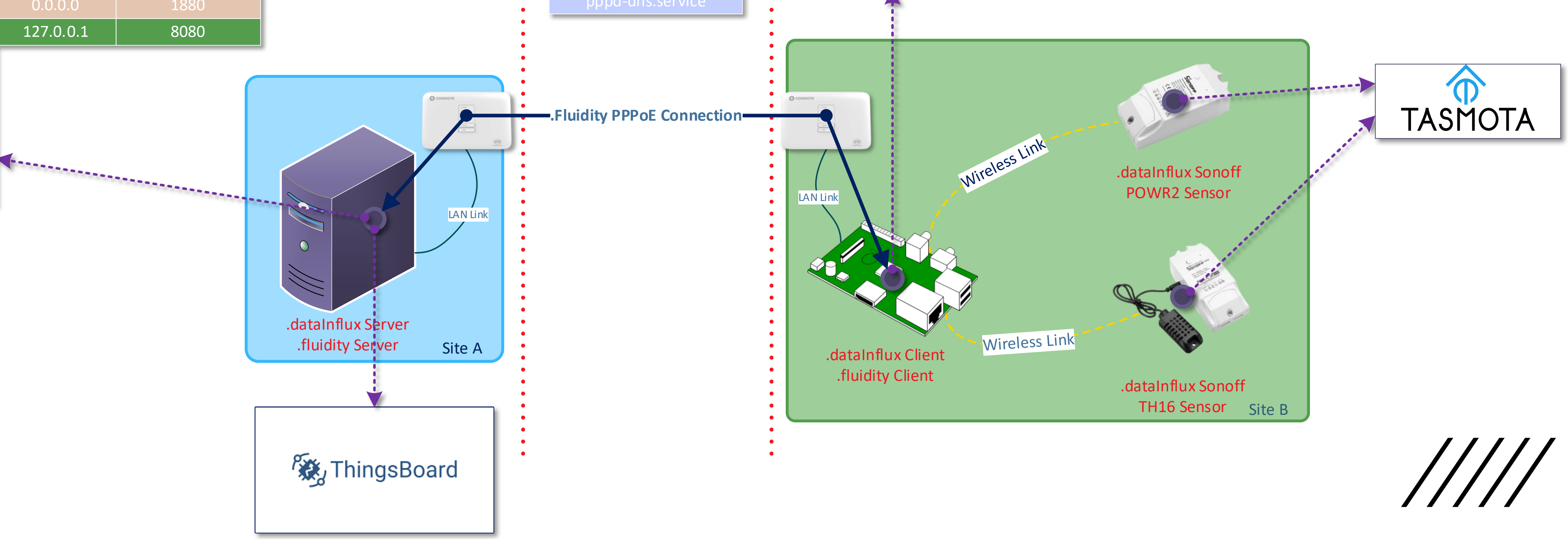
.dataInflux / The Extended Concept



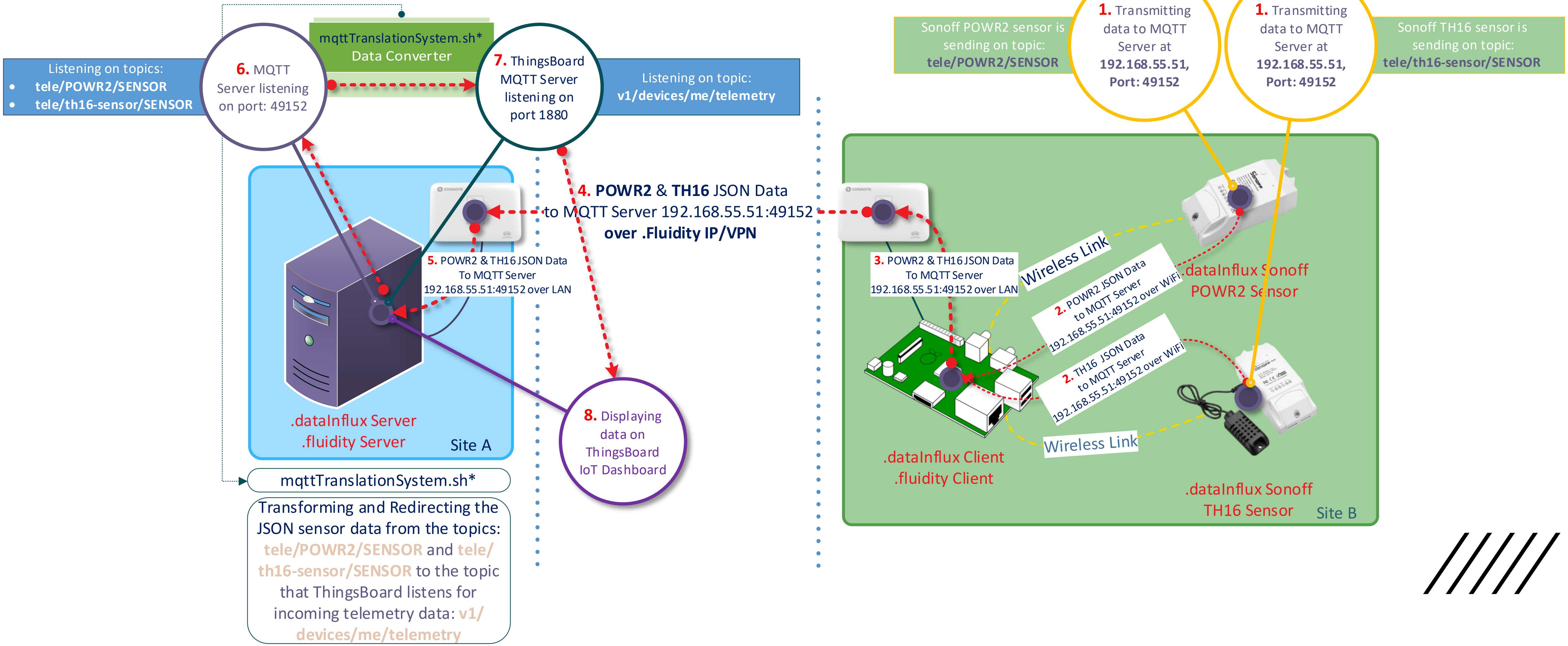
.dataInflux / The Engineering Specifics

List of services on .dataInflux client	
hostapd.service	SSID: dataInfluxNetwork PASS: x9dataInfluxx9
dnsmasq.service	Sensors' DHCP Range: 192.168.55.2 - 192.168.55.50
pppd-dns.service	

List of services on .dataInflux server	Active on IP address:	Port number	Specially assigned Lo interface for this service:
mosquitto.service	0.0.0.0	49152	192.168.55.51
mosquitto.service	0.0.0.0	1880	
thingsboard.service	127.0.0.1	8080	
postgresql.service			
zookeeper.service			
kafka.service			
mqttTranslationSystem.service			
pppd-dns.service			



.dataInflux / The MQTT messaging system from start to finish (Steps 1. to 8.)



Listening on topics:
- tele/POWR2/SENSOR
- tele/th16-sensor/SENSOR

6. MQTT Server listening on port: 49152

mqttTranslationSystem.sh*
Data Converter

7. ThingsBoard MQTT Server listening on port 1880

Listening on topic:
v1/devices/me/telemetry

Sonoff POWR2 sensor is sending on topic:
tele/POWR2/SENSOR

1. Transmitting data to MQTT Server at **192.168.55.51, Port: 49152**

1. Transmitting data to MQTT Server at **192.168.55.51, Port: 49152**

Sonoff TH16 sensor is sending on topic:
tele/th16-sensor/SENSOR

4. POWR2 & **TH16** JSON Data to MQTT Server 192.168.55.51:49152 **over .Fluidity IP/VPN**

5. POWR2 & TH16 JSON Data To MQTT Server 192.168.55.51:49152 over LAN

3. POWR2 & TH16 JSON Data To MQTT Server 192.168.55.51:49152 over LAN

Wireless Link

2. POWR2 JSON Data to MQTT Server 192.168.55.51:49152 over WiFi

2. TH16 JSON Data to MQTT Server 192.168.55.51:49152 over WiFi

.dataInflux Sonoff POWR2 Sensor

.dataInflux Server
.fluidity Server

Site A

8. Displaying data on ThingsBoard IoT Dashboard

.dataInflux Client
.fluidity Client

Wireless Link

.dataInflux Sonoff TH16 Sensor

Site B

mqttTranslationSystem.sh*

Transforming and Redirecting the JSON sensor data from the topics: tele/POWR2/SENSOR and tele/th16-sensor/SENSOR to the topic that ThingsBoard listens for incoming telemetry data: v1/devices/me/telemetry

.dataInflux / The Data Converter

A .dataInflux Data Converter is the following BASH script:

```
(mosquitto_sub -p 49152 -u dataInflux -P influx -t tele/th16-sensor/SENSOR \
| jq -r -c -M --unbuffered '{ Temperature: .AM2301.Temperature, Humidity: .AM2301.Humidity, DewPoint: .AM2301.DewPoint }’ |
while IFS= read -r line
do
   mosquitto_pub -d -q 1 -h localhost -p 1883 -t "v1/devices/me/telemetry" -u ThingsBoard_Key -m $line
done) &
```

Some initial remarks on the script:

- The script considers specifically the SONOFF TH16 sensor and the JSON data that the sensor transmits.
- The script begins and ends with a parenthesis. This means that it executes in a subshell.
- The script ends with an ampersand (&). This means that it will be executed in the background.

Breaking up the script to its constituent parts:

```
mosquitto_sub -p 49152 -u dataInflux -P influx -t tele/th16-sensor/SENSOR
```

mosquitto_sub: Subscribe to the MQTT mosquito server:

- **-p** which runs on port **49152**.
- **-u** the server’s username is **dataInflux**.
- **-P** the server’s password is **influx**.
- **-t** and start listening to topic **tele/th16-sensor/SENSOR** for incoming MQTT messages.

.dataInflux / The Data Converter

.dataInflux / Fluidity

Click on the logo for more info ./jq

```
| jq -r -c -M --unbuffered '{ Temperature: .AM2301.Temperature,
Humidity: .AM2301.Humidity, DewPoint: .AM2301.DewPoint }'
```

(|) pipe the output from **`mosquitto_sub`** to the **`jq`** data converter.

According to its developers "**`jq`** is a lightweight and flexible command-line JSON processor" which "you can use it to slice and filter and map and transform structured data".

`jq:` Transform the JSON data:

- **`-r`** raw output.
- **`-c`** compact output.
- **`-M`** disable coloring on output.
- **`--unbuffered`** Flush the output after each JSON object is printed (empirically, this worked the best).
- Do the data transformation according to the following formula:

  ```
  '{ Temperature: .AM2301.Temperature, Humidity: .AM2301.Humidity,
  DewPoint: .AM2301.DewPoint }'
  ```

Hence, the incoming JSON data from the TH16 sensor:

From this:

```
{"Time":"2021-03-08T02:18:16","AM2301":{"Temperature":20.9,"Humidity":36.4,"DewPoint":5.4},"TempUnit":"C"}
```

Changes to this:

```
{"Temperature":20.9,"Humidity":36.4,"DewPoint":5.4}
```

.dataInflux / The Data Converter

```
|
while IFS= read -r line
do
   mosquitto_pub -d -q 1 -h localhost -p 1883 -t "v1/devices/me/telemetry" -u ThingsBoard_Key -m $line
done
```

(|) pipe the output from `jq` to a `while loop`.

```
while IFS= read -r line
do
   { command block }
done
```

`while` a new line is received from the previous command, repeatedly execute the internal command block. Store the result in variable `line`.

```
mosquitto_pub -d -q 1 -h localhost -p 1883 -t "v1/devices/me/telemetry" -u ThingsBoard_Key -m $line
```

`mosquitto_pub`: Publish the converted data payload from the TH16 sensor to the ThingsBoard MQTT server:

- **`-d`** enable the debug messages
- **`-q`** with quality of service **1**
- **`-h localhost`** publish the messages on localhost address **`(127.0.0.1)`**
- **`-p`** the server is active on port **`1883`**
- **`-t`** publish the messages on topic **`v1/devices/me/telemetry`**
- **`-u`** by using the ThingsBoards generated key **`ThingsBoard_Key`** for the specific sensor
- **`-m`** by sending the JSON data payload stored in variable **`line`** which is: `{"Temperature":20.9,"Humidity":36.4,"DewPoint":5.4}`

Appendix A / Network Planning and the Addressing Scheme

.dataInflux / .Fluidity over PPPoE EXAMPLE SYNOPSIS			
.Fluidity Devices	**External IP Address/External Interface**	**.Fluidity Device Password**	**List of Internal Interfaces on each .Fluidity Device**
.Fluidity Server	`static_ip_from_isp/ppp0`	`_dataInflux-S3rv3r-2020`	`Network 1: lo:1`
.Fluidity Client	`static_ip_from_isp/ppp0`	`_dataInflux_Cli3nt`	`Network 2: wlan0`

PPPoE VPN TUNNELING ADDRESING SCHEME					
Tunnel Connections	**Server Address**	**Client Address**	**Subnet Mask**	**Network Address**	**Broadcast Address**
Server - Client (VPN NETWORK)	192.168.53.1	192.168.53.2	255.255.255.252	192.168.53.0	192.168.53.3

Devices Attached and the Networking Addressing Scheme in Detail	
Network 1	
CIDR Network Address	192.168.54.0/24
(mosquitto service) lo:1 on .dataInflux Server	192.168.54.1
Broadcast Address	192.168.54.255
Network 2	
CIDR Network Address	192.168.55.0/24
.dataInflux DHCP wireless sensors network range	192.168.55.1 - 192.168.55.50
Broadcast Address	192.168.55.255

○ Appendix B / Data Converter – The mosquito server and the **mqttTranslationSystem** Unit File

Configuring the mosquitto service

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Copy - paste the text below:

```
allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 49152
```

```
sudo mosquitto_passwd -c /etc/mosquitto/pwfile dataInflux
pass: influx
```

Configuring the mqttTranslationSystem service

```
sudo nano /etc/systemd/system/mqttTranslationSystem.service
```

Copy - paste the text below:

```
[Unit]
Description=.dataInflux MQTT messaging translation process from Tasmota Sonoff
devices to ThingsBoard telemetry system
Requires=zookeeper.service

[Service]
Type=forking
ExecStart=/home/datainflux/mqttTranslationSystem.sh start
PIDFile=/run/mqttTranslationSystem.pid

[Install]
WantedBy=multi-user.target
```

Appendix B / Data Converter – **mqttTranslationSystem** BASH script

The mqttTranslationSystem script

```
sudo nano /home/pi/mqttTranslationSystem.sh
```

Copy - paste the text below:

```
#!/bin/bash

(mosquitto_sub -p 49152 -u dataInflux -P influx -t tele/th16-sensor/SENSOR \
| jq -r -c -M --unbuffered '{ Temperature: .AM2301.Temperature, Humidity: .AM2301.Humidity, DewPoint: .AM2301.DewPoint }' |
while IFS= read -r line
do
   mosquitto_pub -d -q 1 -h localhost -p 1883 -t "v1/devices/me/telemetry" -u ThingsBoard_Generated_Key -m $line
done) &

(mosquitto_sub -p 49152 -u dataInflux -P influx -t tele/POWR2/SENSOR \
| jq -r -c -M --unbuffered '{ Total: .ENERGY.Total, Yesterday: .ENERGY.Yesterday, Today: .ENERGY.Today, Period: .ENERGY.Period, Power:
.ENERGY.Power, ApparentPower: .ENERGY.ApparentPower, ReactivePower: .ENERGY.ReactivePower, Factor: .ENERGY.Factor, Voltage:
.ENERGY.Voltage, Current: .ENERGY.Current }' |
while IFS= read -r line
do
   mosquitto_pub -d -q 1 -h localhost -p 1883 -t "v1/devices/me/telemetry" -u ThingsBoard_Generated_Key
-m $line
done) &

# Write pid of the child to the pidfile:
echo "$!" >/run/mqttTranslationSystem.pid
exit
```

WHO WE ARE

ckapolonaris@c-ts.gr

LinkedIn

Charalampos Kapolonaris is a Telecoms Electronics Engineer and is currently employed by the OTE Group of Companies (HTO) as a B2B Technician.

Charalampos believes in the transformative power that Open Source Software brings to modern society.

Based on the challenges that modern telecommunications organizations are currently facing, Charalampos drew inspiration for this project.

These can be summarized as follows:

- The convergence of the legacy technologies to the new unified telecommunications IP infrastructure.
- The need for innovation in a continuously changing technological environment.

But most of all:

- safeguarding the telecommunications privacy and data security in systems that service critical subsystems of the corporate infrastructure.

vkoutlas@c-ts.gr

Vasileios Koutlas holds a degree in Electrical Engineering and adheres to the idea of Ethical and White Hat Hacking.

He has broad experience in the use of Linux OS and participated in many projects, mainly related to IP and VoIP networks.

Being highly sensitive in matters concerning data security and data confidentiality, he enthusiastically accepted his participation in this project.

He believes that his involvement in the Open Source Community may result in the development of safer methods for data transfer.

He has been working in OTE Group of Companies (HTO) occupying various positions in the organization, currently occupying the position of CX Technicians Supervisor of Filed Technical Operations in Xanthi Greece.

THANK YOU
FOR YOUR
ATTENTION