Energy performance monitoring with Carlo Gavazzi UWP 3.0 and IoT HUBS

Release 3.0

CARLO GAVAZZI CONTROLS

February 28th 2019

Energy performance monitoring with Carlo Gavazzi UWP 3.0 and IoT HUBS

List of Contents

Abstract 2
Who should read this document 2
1. Why setting up an IoT based system based on Carlo Gavazzi solutions for monitoring of energy performances. 3
Abstract 3
Local solution: UWP 3.0 3
IoT solution: the 4-tier architecture 3
UWP 3.0 and Microsoft Azure IoT 4
UWP 3.0 and Amazon AWS IoT 4
2. How-To set up a Microsoft Azure IoT based system with Carlo Gavazzi solutions for monitoring energy performances. 5
The purpose of this "How to" 5
…and the necessary steps 6
From UWP 3.0 to an SQL database in 4 steps 7
1. Creating an IoT HUB into Microsoft Azure 7
2. Creating an SQL Server 8
3. Creating a Stream Analytics process 9
4. Cross transforming the table by using the PIVOT function 10
Connecting PowerBI and creating the dashboard in 3 steps 11
1. Opening PowerBI and connecting to the data source 11
2. Creating the Dashboard 12
3. Publishing the Dashboard 13
Dashboard in action: a real-world example 13
3. How-To set up an Amazon AWS IoTbased system with Carlo Gavazzi solutions for monitoring energy performances. 14
The purpose of this "How to" 14
Connecting to Amazon AWS IoT in 3 steps. 14
1. Registering into Amazon AWS IoT 14
2. Creating a THING via Console 14
3. Creating certificates 14
4. Setting policies 14
5. Link policies to certificates 15
6. Link Things to certificates 15
7. Create rules 15
8. Activating UWP 3.0 16
9. Checking the IoT HUB side 16
Disclaimer 17

Abstract

Systems Integrators, Energy Managers and Energy Service Companies can greatly benefit from CARLO GAVAZZI's UWP 3.0 datalogger, gateway and web-server. Nowadays the IoT paradigm delivers unlimited potential to convert raw data into valuable information. We're doing our part to make your job easier, quicker and more efficient, while providing you the unbridled power of the cloud.

Who should read this document

This document is meant for system integrators, product managers and CTOs of those companies which are working in the energy efficiency market and want to provide to their customers a scalable solution for energy performance monitoring.

Moreover, Energy Managers looking for energy monitoring solutions for their company will benefit by reading this document.

The IoT paradigm dramatically extends the capabilities of those Cloud based systems meant to interact with the real world: different data sources may be connected to a central hub and the relevant aggregated data may be analysed by different tools and linked to other data repositories; in other words, it is possible to move the focus from data to information. A monitoring system for multiple distributed installations, centralizing data in a common HUB and redistributing information to many systems can be built by putting together different hardware and software components without developing a solution from scratch.

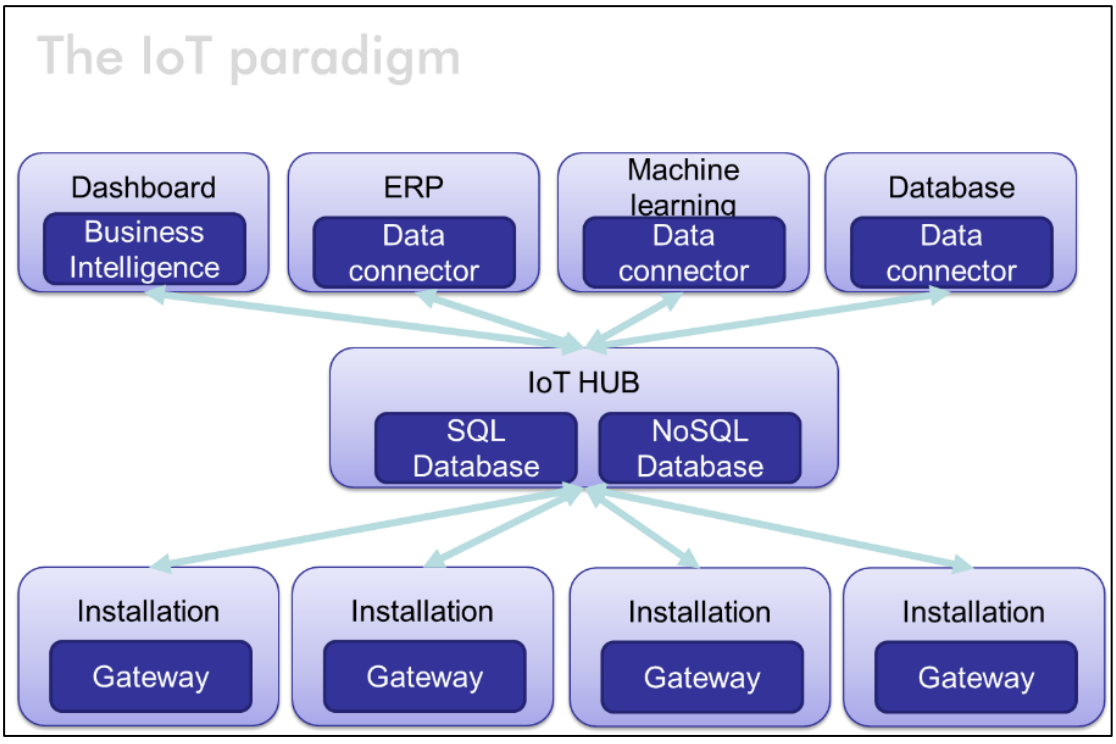


FIGURE 1: THE IOT PARADIGM FOR MONITORING DISTRIBUTED INSTALLATIONS

Basically, this document is split into 2 sections:

1. Section-1: explaining the overall solution to decision makers and energy managers
2. Section-2: going deeper into the details of How-To creating an online dashboard solution based on UWP 3.0

1. why setting up an IoT based system based on Carlo Gavazzi solutions for monitoring of energy performances.

Abstract

Energy performance monitoring is a matter of:

1) Installing and connecting energy meters or energy analysers to the target loads
2) Gathering data from meters and storing them locally
3) Re-transmitting data to remote systems for aggregating them, creating KPIs for benchmarking and reporting to decision makers and Energy Managers

Local solution: UWP 3.0

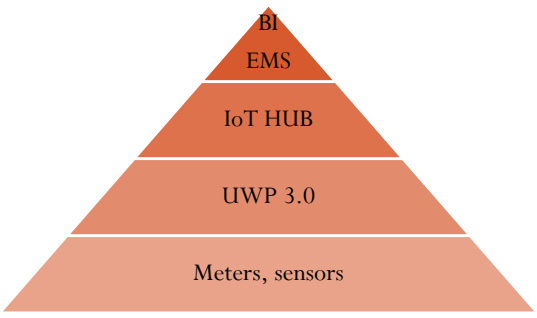
UWP 3.0 is a modular system that records, monitors and transmits analogue and digital signals from an industrial, commercial or residential installation with a specific focus on energy efficiency. The system includes a web server with a powerful and intuitive user interface to monitor data and set up the system. Data can be transmitted using various protocols (FTP, HTTP, Modbus TCP/IP) and via wired or wireless connection.

UWP 3.0 includes everything is needed to monitor a local installation, by gathering data from meters, logging them, displaying them via the web server and sending them remotely. Nonetheless, very often, it is necessary to:

1) Centralize data from multiple locations into a common repository
2) Integrate measured data with information from other sources
3) Analyse data with business intelligence tools

In this case the IoT paradigm provides a solution by connecting together different tiers of a common monitoring pyramid.

IoT solution: the 4-tier architecture



This is a 4-tier architecture, including meters, sensors, edge units (the UWP 3.0) and the Business Intelligence (BI) or Energy Management System (EMS) tiers.

The Edge unit (the UWP 3.0) gather data from the field and delivers them to the IoT HUB. Energy data management solutions developed in the cloud communicate with the IoT HUB and provide users with the needed functions to convert raw data into information.

UWP 3.0 and Microsoft Azure IoT

UWP 3.0 is Microsoft Azure Certified for IoT.

You can find more details about UWP 3.0 certification at the Microsoft link here: https://catalog.azureiotsolutions.com/details?title=UWP30RSEXXX&source=home-page

Variables gathered from the connected devices (meters, sensors and others) are converted into JSON format and sent to the Microsoft Azure IoT HUB with the necessary security and reliability.

By having data available on Microsoft Azure, users can leverage the powerful Microsoft tools for:

a) Integrating other data source data
b) Sharing information with other systems
c) Using the best Business Intelligence tools to dig into data

Please find more information about the Azure IoT solutions here: https://azure.microsoft.com/en-gb/suites/iot-suite/

UWP 3.0 and Amazon AWS IoT

UWP 3.0 is compatible with Amazon AWS IoT.

You can find more details about Amazon AWS IoT here: https://aws.amazon.com/iot/?nc1=f_ls

Variables gathered from the connected devices (meters, sensors and others) are converted into JSON format and sent via MQTT to the Amazon AWS IoT HUB with the necessary security and reliability.

By having data available on Amazon AWS, users can leverage the powerful Amazon tools for:

a) Integrating other data source data
b) Sharing information with other systems
c) Using the best Business Intelligence tools to dig into data

2. How-To set up a Microsoft Azure IoT based system with Carlo Gavazzi solutions for monitoring energy performances.

The purpose of this "How to" …

Integrating data pushed by UWP 3.0 into Microsoft Azure IoT, allows to put in place a monitoring system in a matter of hours without developing any software; by leveraging the powerful tools provided by Microsoft, it is possible to create relationships between UWP 3.0 data and information coming from different sources.

Besides, the use of Microsoft unlimited scalability options allows to match whatever use cases, from single site.

Even though once data are available into Microsoft Azure, there are dozens of different integration paths: the purpose of this document is to show how to analyse them with Microsoft PowerBI.

For further information about PowerBI, please check the official Microsoft PowerBI documentation: https://powerbi.microsoft.com/en-us/

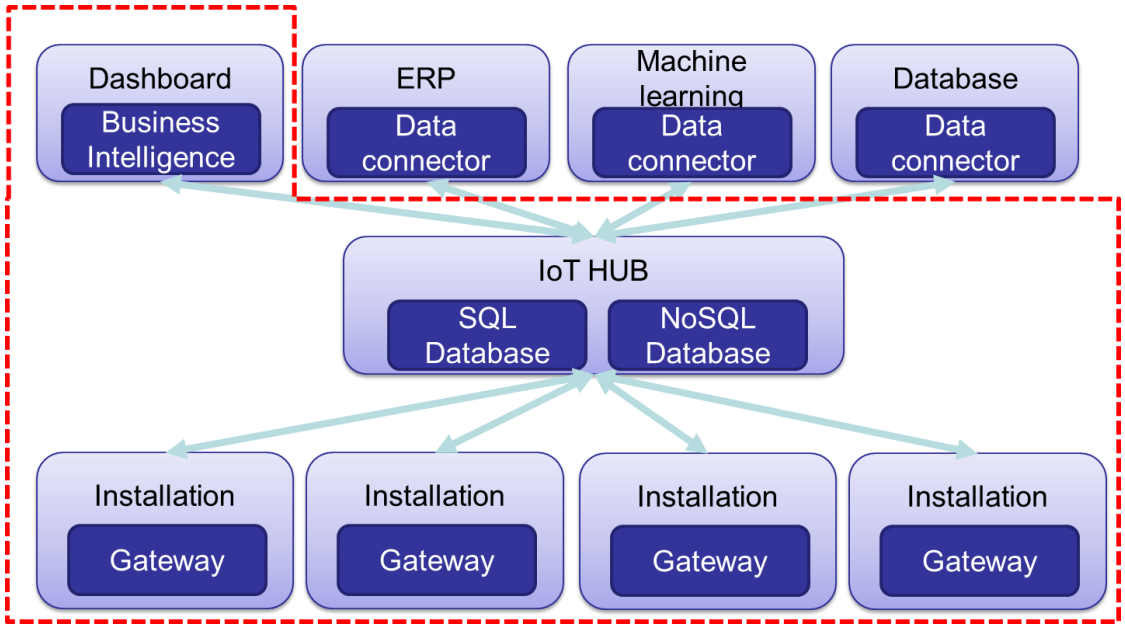


FIGURE 2: OUR CONTEXT, FROM INSTALLATIONS TO BUSINESS INTELLIGENCE

…and the necessary steps

The necessary steps are:

1) Log it into a proper Microsoft Azure account
2) Activate an IoT Hub from the relevant menu
3) Connect the target UWP 3.0 to the IoT HUB
4) Create an SQL Server database from the relevant menu
5) Create the target table(s) into the DB created at point -4-
6) Create a Stream Analytics item from the relevant menu, so as to connect the IoT HUB to the SQL Server
7) Use a Dashboard tool like PowerBI to connect to the SQL Server, create and publish the desired dashboards

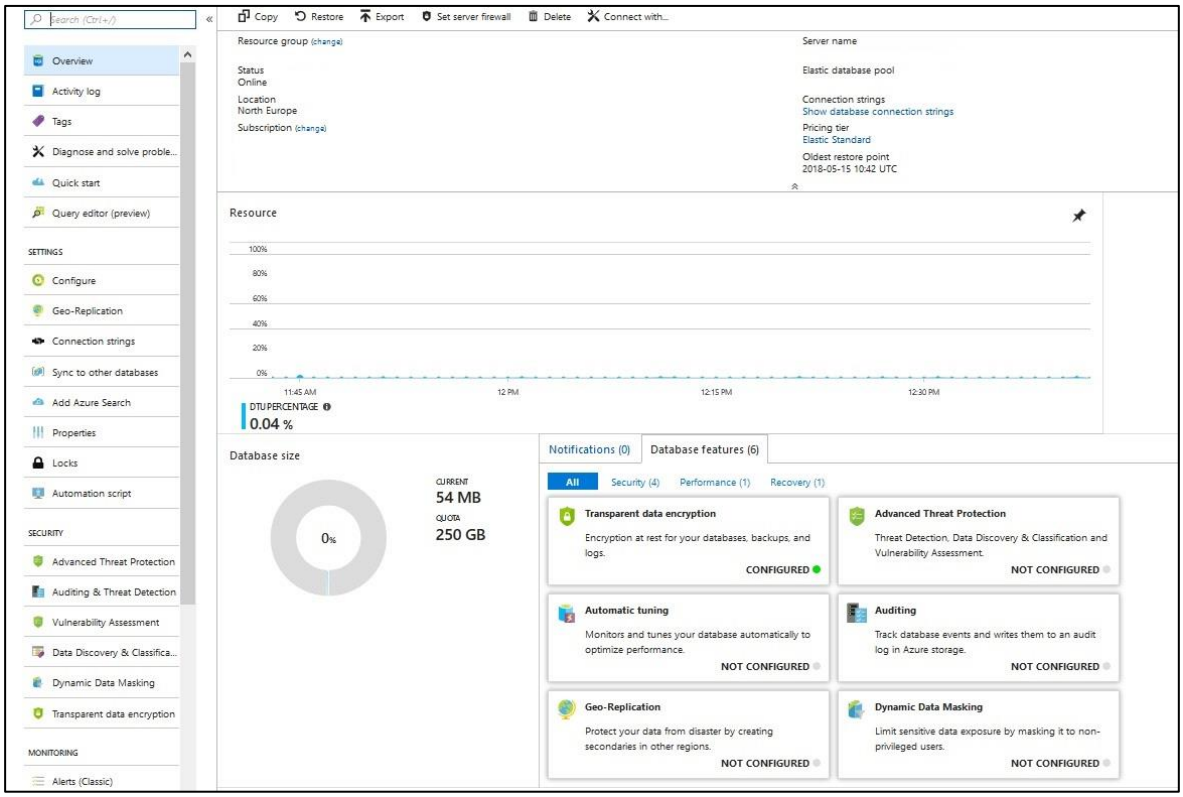


FIGURE 3: MICROSOFT AZURE CONTROL PANEL

We will see the relevant details in the next pages.

From UWP 3.0 to an SQL database in 4 steps

1. Creating an IoT HUB into Microsoft Azure

By using the Resources menu, it is possible to create the IoT HUB; an IoT HUB is necessary to create a single contact point on which the gateways can push data via MQTT; once the IoT HUB has been created, the relevant connection string must be set into the gateway so to create the link and start the communication.

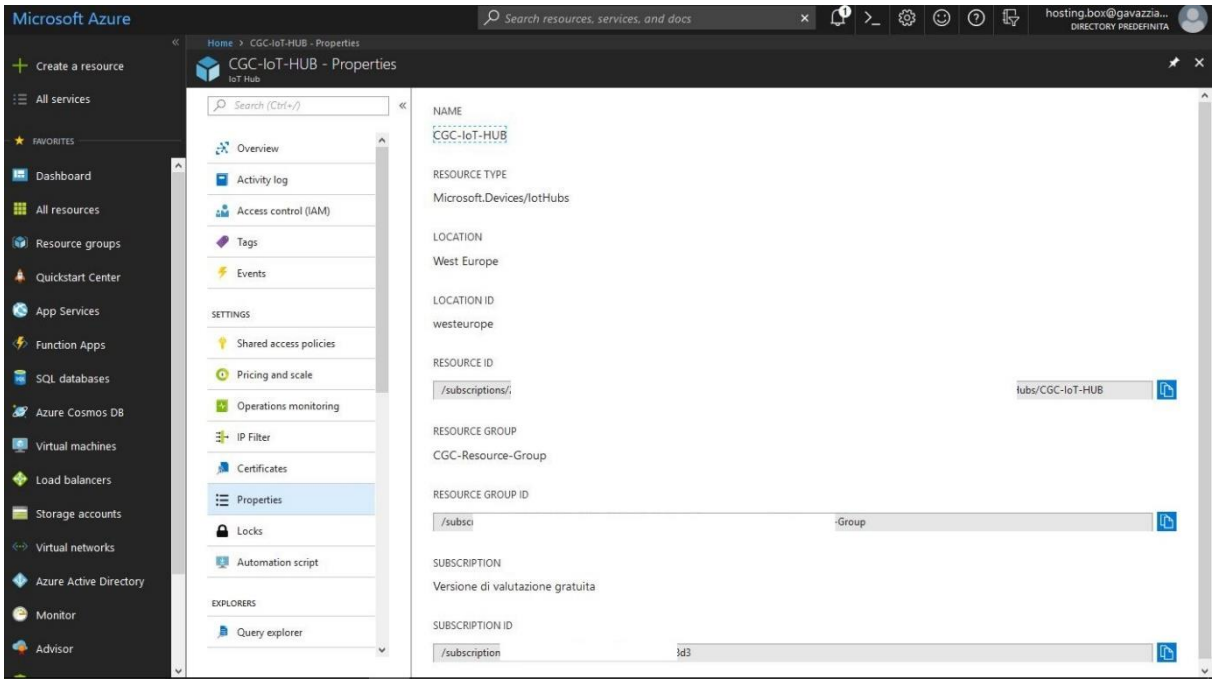


FIGURE 4: IOT HUB PARAMETERS AFTER THE SET-UP

For more information about the JSON format of data that UWP 3.0 pushes to internet, please check the following document: http://www.productselection.net/PDF/UK/UWP3.0_IoT_Protocols_R1.pdf

When data are available within Azure IoT Hub as JSON containers, the system integrators can move them to an SQL or an SQL database and use the available tools of MS Azure to display, convert, aggregate and analyse them according to the needs.

2. Creating an SQL Server

An SQL server is necessary to process historical data as a foundation for dashboards; IoT HUB data are unstructured (they have been delivered as JSON files) and can hardly be used for creating efficient queries. For these reasons, it is necessary to:

a) Create an SQL Server, by using the specific menu item in the Microsoft Azure control panel; the set-up depends on the available credit, and the application needs;
b) Create a table in the SQL Server, on which the data stream will be targeted;
c) Create a Pivot View, linked to the data table, meant to define a columns-based data structure which can be efficiently used by the dashboards.

The table may be created with a query like this one:

```
create table UWP30 (
varDatetime datetime,
GatewaySN char(50),
DeviceName char(50),
varName char(50),
varType char(50),
varEngUnit char(50),
varValue real
);
```

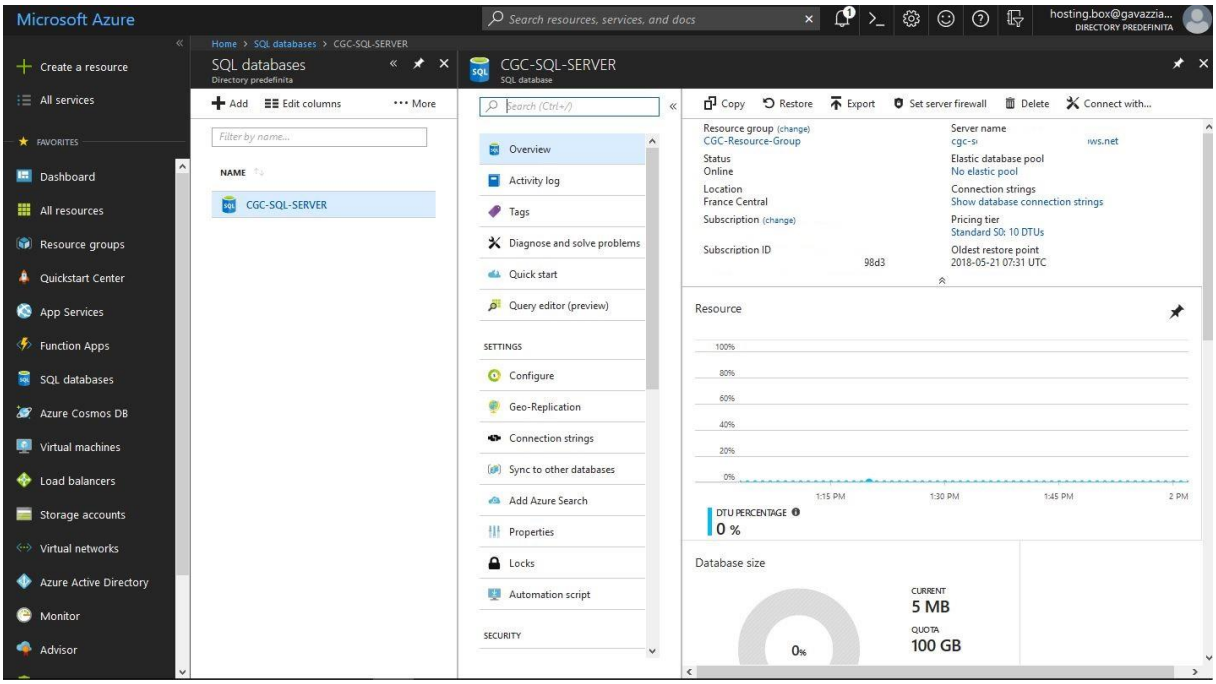


FIGURE 5:THE SQL SERVER HAS BEEN CORRECTLY CONFIGURED

3. Creating a Stream Analytics process

By using the Resources menu in Microsoft Azure Control Panel, a Streaming Analytics must be created to join:

a) INPUT: the IoT HUB
b) OUTPUT: the SQL Server's table

The link is possible via a specific function (please find here an example based on the UWP 3.0 payload):

```
SELECT
    cast(UDF.unixTimestamp(meter.ArrayValue.time) AS DATETIME) as
varDatetime,
    i.sn as GatewaySn,
    meter.ArrayValue.name as DeviceName,
    metrics.ArrayValue.var as VarName,
    metrics.ArrayValue.type as varType,
    metrics.ArrayValue.unit as varEngUnit,
    metrics.ArrayValue.value as varValue
INTO [SQL-OUTPUT]
FROM [IOT-INPUT] AS i
CROSS APPLY GetArrayElements(i.Meter) AS meter
CROSS APPLY GetArrayElements(meter.ArrayValue.data) AS metrics
where i.message is null;
```

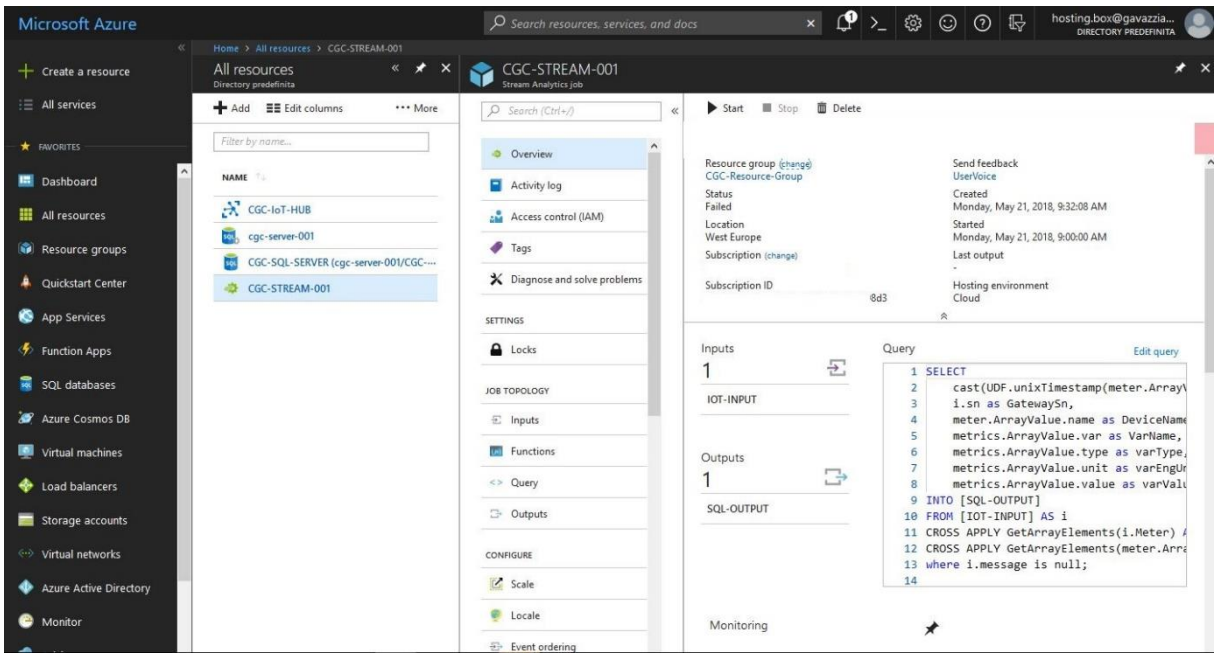


FIGURE 6: THE STREAM ANALYTICS CONFIGURATION PAGE

4. Cross transforming the table by using the PIVOT function

Usually it is necessary to cross-transform the Database table in which values are stored because a columns-based structure is more efficient than a row-based layout when dashboards are the targets.

In our case, the target is to extract energy data and transform the variable names (kWh, kvarh types) into columns.

This is the relevant query which can be implemented as a VIEW into the SQL Server database or used as the data-source query into the dashboard tool (i.e. Power BI, see later on).

```
SELECT *
FROM (SELECT
                    varDateTime,
                    GatewaySN,
                    DeviceName,
                    varName,
                    vartype,
                    varValue
        FROM   UWP30
          where varType =0

         ) as SourceTable
PIVOT
(
   Avg(varValue)
   FOR varName in
   ([kWhac],[kWhacn],[kWhl1],[kWhl2],[kWhl3],[kvarh],[kvarhn])
) AS AvgVarValue

ORDER BY varDateTime
```

Connecting PowerBI and creating the dashboard in 3 steps

PowerBI is a powerful dashboarding tool from Microsoft which allows user to build dashboards by connecting to a wide range of data sources, analyse them locally, sharing and/or publishing them on the web.

For further information please check here: https://powerbi.microsoft.com/en-us/desktop/

1. Opening PowerBI and connecting to the data source

By selecting NEW SOURCE, it is possible to set the connection parameters to the AZURE SQL SERVER and set the connection query implementing the PIVOT function (see the relevant point above).

By selecting IMPORT, it is possible to update data on demand, or set a scheduler; this is the less resource consuming option (pay care to this part because querying and downloading data has a cost).

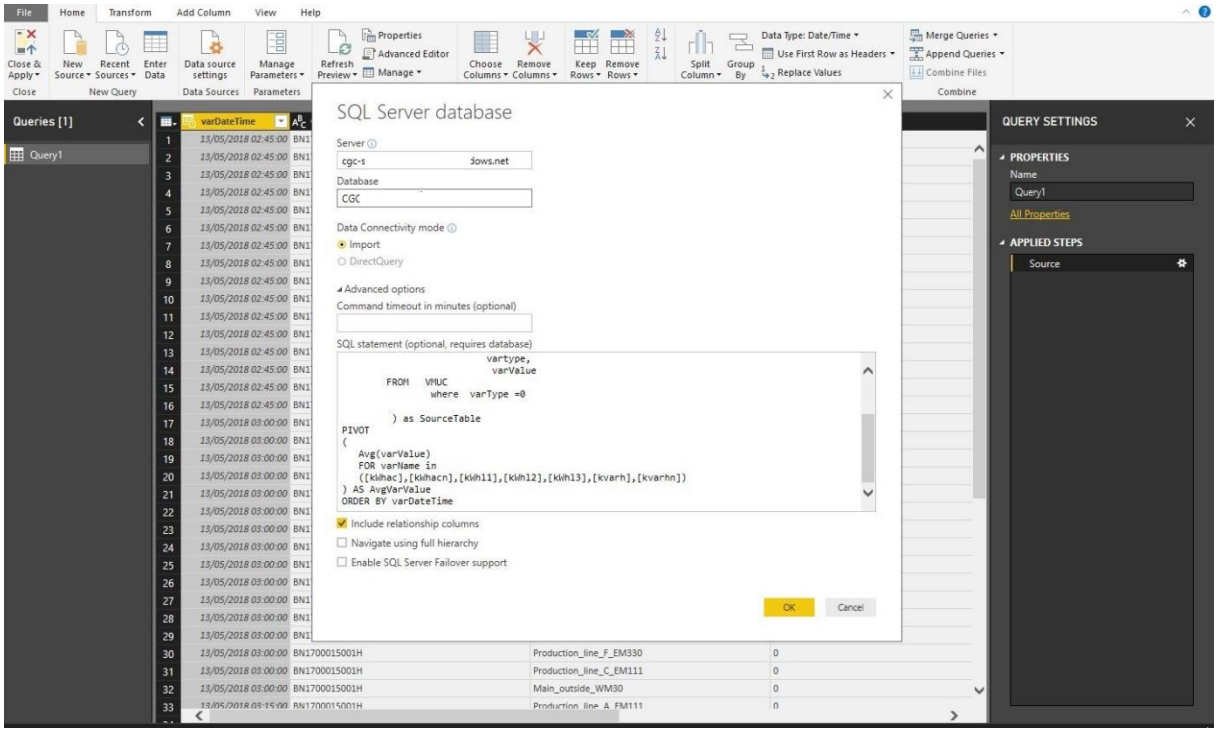


FIGURE 7: THE DATA SOURCE CONNECTION

2. Creating the Dashboard

By using the tools of PowerBI, it is possible to create dashboards with filters and sliders and, by selecting the preferred charts, it is also possible to join different datasources.

The description of PowerBI functions is beyond the scope of this document; the relevant documentation is available here: https://powerbi.microsoft.com/en-us/desktop/

For advanced users, PowerBI provides DAX embedded language: https://docs.microsoft.com/en-us/power-bi/guided-learning/introductiontodax?tutorial-step=1

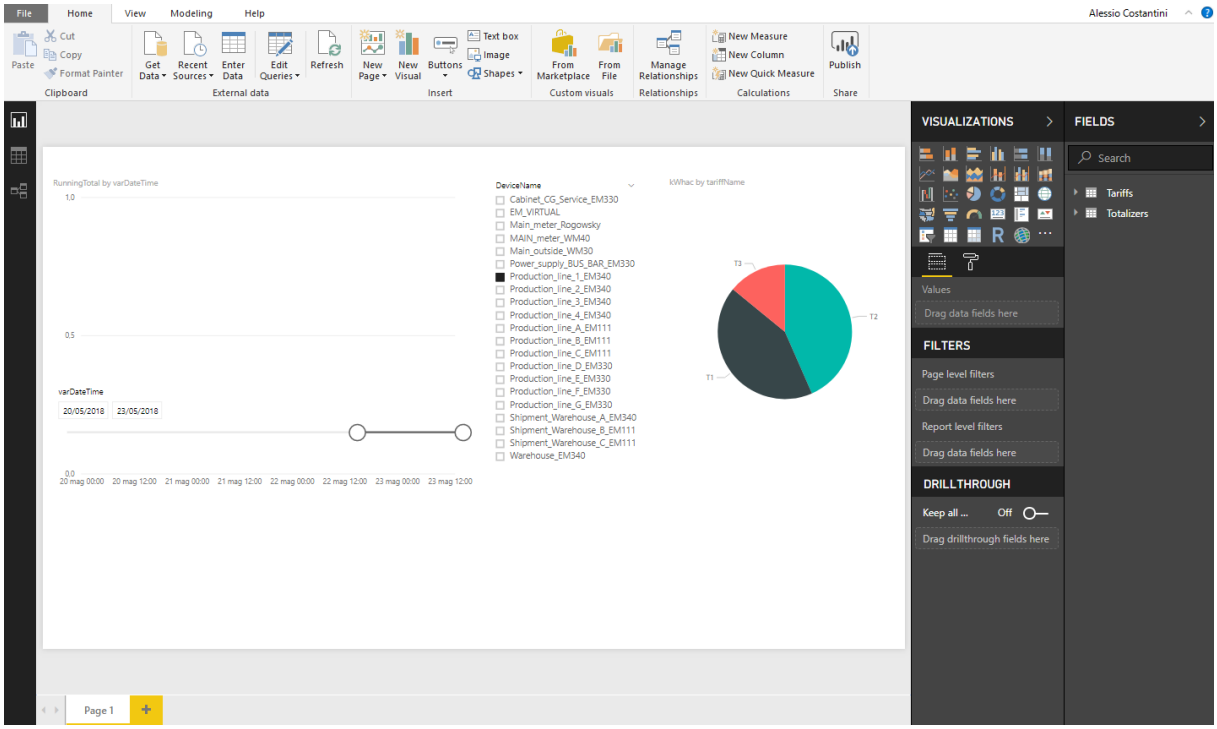


FIGURE 8: DASHBOARD EDITING

3. Publishing the Dashboard

By selecting PUBLISH, the user can publish the dashboard on the selected workspace.

It is necessary to subscribe to PowerBI to publish dashboards. Afterwards the project is uploaded to the selected PowerBI account and then it can be:

a) Used by the account owner
b) Shared with other PowerBI accounts
c) Published on a website

The full description of PowerBI web capabilities is beyond the scope of this document; please check here for further information: https://powerbi.microsoft.com/en-us/desktop/

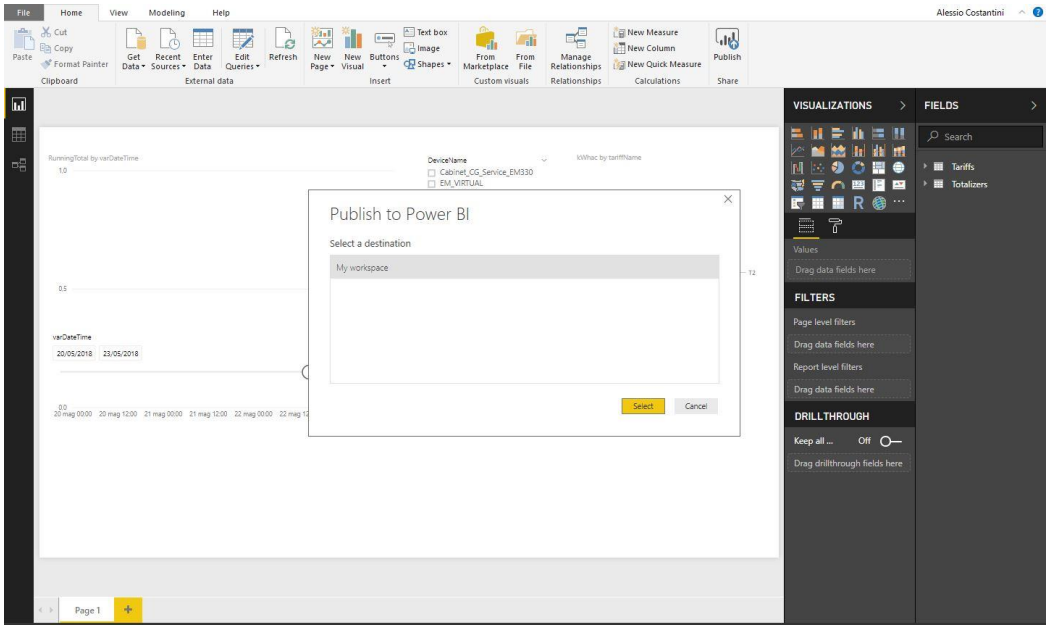


FIGURE 9: PUBLISHING A DASHBOARD

Dashboard in action: a real-world example

By opening the following link, it is possible to see a dashboard connected to a real installation powered by UWP 3.0 and using the above tools to publish the relevant data:
https://app.powerbi.com/view?r=eyJrIjoiNzM2N2FiNTQtNzBiZS00NTM5LWE2ZGEtYWQwMWE3MGI4MDM4IiwidCI6IjkwMjNiZWM1LWZiYjMtNDUzMi1iNzMwLWFhZWI3NmQ1OTBjYSIsImMiOjl9

3. How-To set up an Amazon AWS IoTbased system with Carlo Gavazzi solutions for monitoring energy performances.

The purpose of this "How to" …

Integrating data pushed by UWP 3.0 into Amazon AWS IoT, allows to put in place a monitoring system in a matter of hours without developing any software; by leveraging the powerful tools provided by Amazon AWS, it is possible to create relationships between UWP 3.0 and any other system connected to the same cloud infrastructure.

Connecting to Amazon AWS IoT in 3 steps.

1. Registering into Amazon AWS IoT

It is necessary to have valid Amazon AWS IoT credentials and log in into https://console.aws.amazon.com/iot/home

2. Creating a THING via Console

The following steps are:

1) Opening the **Console** (IoT Core) and click on **Manage** on the left pane
2) Selecting **Things**
3) On the main screen a box will appear: click on the **Register a thing** button
4) Click on the **Create a single thing** button
5) Type a name for the thing and click on **Next** to create it

3. Creating certificates

Communication between a device and the AWS IoT happens through X.509 individual certificates, and AWS can create them on its own.

1) Click on **Secure** on the left pane
2) Select **Certificates**
3) On the main screen a box will appear; click on the **Create a certificate** button
4) Select the **One-click certificate** creation option by clicking on Create certificate
5) A new **Certificate created! page** will open, where you can download the certificate and the relevant keys
6) KEEP THE FILES IN A SAFE PLACE! If deleted, the Private and Public Keys can't be downloaded anymore, and a new certificate shall be created.
7) Click on **A root CA for AWS IoT Download** to open a new tab
8) Download the Amazon Root CA 1 certificate by right-clicking on the relevant link and selecting Save Link as…
9) Click on **Activate** to activate the device's certificate

4. Setting policies

As certificates are used to instantiate the communication layer between the device and the AWS IoT, policies are used by AWS IoT to allow a certain set of operations for devices to interact with itself.

1) Click on **Secure** on the left pane
2) Select **Policies**
3) On the main screen a box will appear: click on the **Create a policy** button

4) Type a name for the policy and, in the Add statements tab, type the IoT action and the associated Resource ARN:
    a. Essential actions required for a proper communication are iot:Connect, iot:Subscribe and iot:Publish
    b. To get the Resource ARN, open Manage on the left pane and click on Things. Select your thing by clicking on it, and select the text in the dark box within the Resource ARN tab.
    c. Normally it is like: arn:aws:iot:eu-central-1:178870925190:thing/UWP3_IoT
    d. To allow every aspect of the AWS IoT account, simply change it into arn:aws:iot:eu-central-1:178870925190:*
5) Check the **Allow** box and click on **Add** statement to create a policy statement for any of the aforementioned actions
6) Click on **Create** to create the policy

5. Link policies to certificates

For policies to work, they shall be linked to certificates:

1) Click on **Secure** on the left pane
2) Select **Certificates**
3) Click on the **three dots** in the upper right corner of your certificate and choose **Attach policy**
4) In the Attach policies to certificate(s) tab, select the checkbox next to your policy and click on **Attach**

6. Link Things to certificates

Also **things** may need their certificates to be attached, in order to be able to create policies based on thing certificates.

1) Click on **Secure** on the left pane
2) Select **Certificates**
3) Click on the **three dots** in the upper right corner of your certificate and choose **Attach thing**
4) In the Attach things to certificate(s) tab, select the checkbox next to your thing and click on **Attach**

7. Create rules

It might be useful, but not mandatory, to create a rule to trigger an action every time a packet is published to AWS IoT Core.

1) Open the **Amazon SNS Console**
2) Click on **Topics** in the left pane and click the **Create new topic** button
3) Type a topic name and a display name, then click on **Create topic**
4) Check the topic's checkbox on the left, open the Actions menu and click on **Subscribe to topic**
5) In the Protocol and Endpoint fields, select the protocol by which you want to be notified (i.e. EMAIL and your email address)
6) Click on **Create subscription**, so the newly created topic is now to be used in a custom rule:
7) Open the **AWS IoT Core Console** and click on **Act** in the left pane
8) On the main screen a box will appear: click on the **Create a rule** button
9) Type a name and a short description for the rule
10) Scroll down to **Rule query statement** and insert the query you want to be performed
    a. For example, to get all message information from the uwp30rsexxx/BQ3210005010N topic, the query will be SELECT * FROM 'uwp30rsexxx/BQ3210005010N' (NO SEMICOLON NEEDED)
11) In the **Set one or more actions**, click on **Add** action

12) In the **Select an action** page, select the action you want to be performed (i.e. Send a message as an SNS push notification) and click on Configure action
13) Select the SNS topic created some steps earlier from the SNS target combo box
14) Assign a role to grant AWS IoT access to the resource topic
    a. Click on **Create a new role**
    b. Type the role name and click on **Create a new role** again
    c. Select the role from the combo box and click on **Update role**
15) Click on **Add action** and then on **Create rule**

8. Activating UWP 3.0

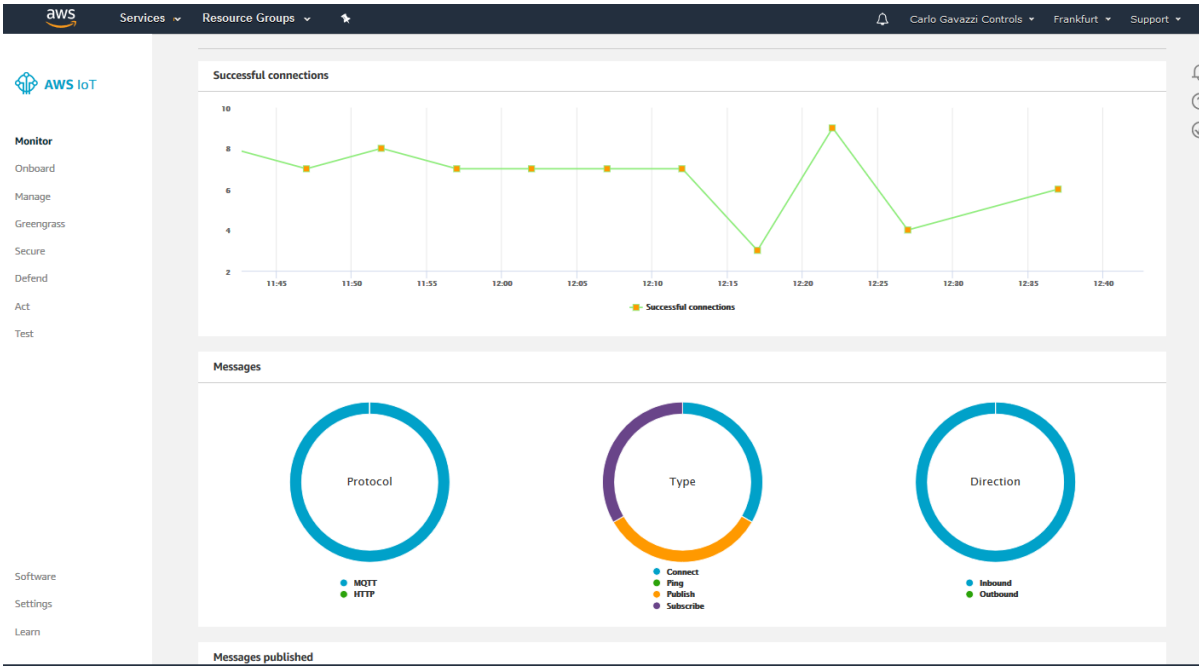
Now it's time to set-up UWP 3.0 for activating MQTT data push:

1) open the **Manage Things** menu and access the desired device
2) Copy the **Rest API endpoint** in the Interact page
3) Login to UWP3.0's Web UI with a valid user profile
4) Paste the **Rest API endpoint**
5) Set the default **Client ID** and the topic where to subscribe
6) Upload the **Device Certificate** and the **Private Key**
7) Click the **Save** button
8) Click the **Test Connection** button

9. Checking the IoT HUB side

In the AWS IoT HUB console:

1) Select **Monitor**
2) Check the system for expected behaviour



Disclaimer