UNIVERSITÀ DEL SALENTO
Facoltà di Ingegneria
Degree Course in Computer Engineering

Project Documentation
Internet of Things

Smart Building

Professor

Prof. Luigi Patrono

Tutor

Ing. Teodero Montanaro

Student(s)
Adriano GARCÍA-GIRALDA
MILENA
Ahsan Raza

Index

Chapter 1 – Introduction

Renewable energy is a very important subject that we as human beings are focusing on to reduce our carbon footprint. Planet Earth is in dire need to get rid of energy sources utilizing fossil fuels. In Europe, 37% of energy consumption in 2020 was from renewable sources. Among renewable energy resources like Wind, Solar, Biomass, Hydropower, etc., Solar energy is one of the resources that are readily available to end consumers. Since the solar energy system is compact and easily installable as compared to other energy sources. It's comparatively lighter on the pocket in the long run and does not require much maintenance. Moreover, having a solar power backup will reduce the reliance on grid power, reducing the consumption of fossil fuels that the grid usually uses to produce electricity.

Generating green energy is not the only solution to reduce the carbon footprint. The development in the renewable energy field and an increasing number of new uses of electricity produced a need to modernize the energy management system at the consumer level. Energy consumption in the residential sector constitutes a large portion of the total energy consumption. With the increase in economic growth and more appliances including vehicles running on electrical power, the demand for energy is skyrocketing. Energy consumption in a house accounts for a significant portion of total energy consumption in both developed and developing countries. The buildings consume 43% of the total energy consumed and residential buildings consume two-thirds of that energy. So managing and consuming the energy produced at the residential level is a highly untapped area and the work is needed to be done.

In the case of solar energy generation in buildings, a consumer is oblivious to consumption habits. That results in the form of energy waste and money. So the need to develop an Energy Management System that not only records the energy consumption pattern of consumers but also makes predictions on the energy consumption. The objective is to reduce the energy consumption during peak hours by controlling the power-intensive loads by keeping in the account consumer preferences and comfort specifically at night when the power is being fed from the battery pack. This will result in the long life of batteries and evidently will save money since the battery pack put a significant cost on solar-powered systems. Managing the energy consumption smartly and efficiently will not only save money but also converge towards carbon neutrality by reacting rapidly to potential issues such as cleverly controlling the Heating, Ventilation, air purification, and Air Conditioning (HVAC). Energy consumption forecasting for appliances in the home has a great influence on energy management. The energy management system is able to determine the best energy assignment plan and a good compromise between energy generation and energy consumption. Implementing this smart system in a building is a basic unit of a broader architecture that can be extended to residential complexes and even to Smart City.

So in summary, the system will record the energy consumption pattern and data of battery pack health with IoT sensors and will make predictions of the energy consumption based on the data

collected from the user habits and weather forecast. It will suggest to the user the best time to plan the usage of appliances. Consumer comfort and priority will be kept into consideration while controlling the power-intensive loads to implement the intelligent model in order to be efficient in energy distribution among all the loads connected to a solar power system. The whole solution will generate the data which will be utilized to implement the system in a digital twin which will help us to fine tune the performance of the A.I. model.

Chapter 2 - Requirements analysis

Functional Requirements

- Priority 1 – To be implemented for sure. essential need to rethink scope if you can't have these. Use these to identify where to apply resources when problems/change occurs (best bang for the buck).
- Priority 2 – To be simulated.
- Priority 3 – Future works.flexible or useful can be added later, expand tolerances – negotiable or deliver in a later phase (if doing incremental development)

FR ID	Title	Description	Priority
1.1	Energy Utilization	The system has to collect information about the energy consumption within the house by an appliance	1
1.2	Temp. Sensing	The system has to collect information about the ambient in which it is installed	1
1.3	Energy Source	An energy source (preferably solar + batteries) will generate the power.	1/2
2.1	Actuator 1	A real actuator has to be controlled	1
2.1	Actuator 2 and more	Some others actuator has to be controlled (but simulated)	2 we need to discuss it
2.1	Solar power performance	The system will record the solar power generated at a given time and make decisions on its basis and enhance its performance.	2(simulated)
2.2	User habits	A.I model will keep track of user habits to optimize the energy consumption and enhance performance.	2
2.3	Digital Twin	A Digital twin will contain all the information about user habits,consumption,generated power.	1

2.4	AI model integrated within the Digital Twin	Some inferred information should be provided by the system through the elaboration of the data contained in the digital twin (e.g., suggestion on how to improve life of solar panels, or notifications for cleaning, etc)	1
2.5	Front-end application	A front-end application allows the consultation of the data contained /stored in the digital twin. It also allows to consult info elaborated at point 2.4	1
2.6	Data for external devices	The system will collect data from external services((e.g. weather forecast or current weather). in order to predict the energy produced	1

Chapter 3 Developments

In this chapter, the approach and design to develop the project is discussed.

The system have the following Sensors in the Smart Building project:

1. Temperature/Humidity sensor
2. Solar Inverter Data (Data regarding the generation of electricity and Battery Status)

The system has the following actuators that we manage on the basis of decisions made from the forecasting model or user can also override the controls on the basis of its comfort and we can also just push a suggestion to the user to use/ don't use certain appliances during a certain period.

1. Leds lights

The system can turn on / turn off the actuators according to energy consumption forecasting. Weather API for forecasting the energy consumption and forecasting module is deployed on Azure Virtual machine but can be deployed on Raspberry Pi.

Devices / module connected to Azure:

1. Raspberry Pi
2. Digital Twin Model

Devices / module connected to Raspberry PI:

1. Sensors
2. Actuators

Devices/ Module that are in publisher mode:

1. Sensors
2. Weather API and Prediction module

Modules that are in Subscriber mode:

1. Server
2. Actuator

Following is the architecture diagram of the components being used in the project.

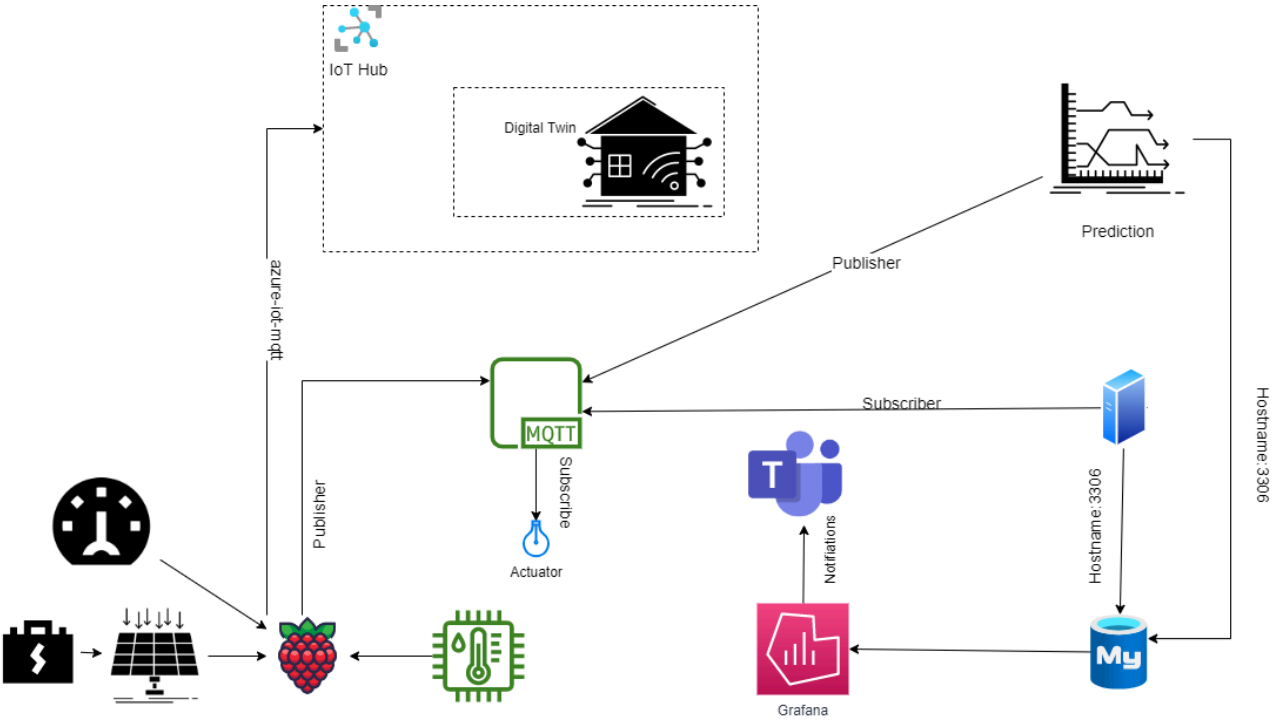

Figure 3.1 The architecture of Project

Chapter 4 State of the art

Ashwini et al (2013) presented a smart energy management system using Zigbee and IEEE 802.15.4. Zigbee is an IEEE 802.15.4 based suite for high-level communication protocol used to create personal area networks with small low-power digital radios. There are some disadvantages of Zigbee, as it has a low transmission rate. It cannot be used for outdoor communication as it has short-range limitations and is not secure at all. We will use MQTT which is a TCP protocol and reside in a network which is more secure.

Nupur Sinha et al (2015) developed a renewable energy management system for smart home for efficient energy generation, minimization cost, and environmentally friendly. Microcontroller and Zigbee were used to control and monitor energy consumption. The drawback was the energy consumption should be minimized by users manually and no implementation of A.I. In this project, the main focus was the development of an independent energy generation module for smart homes, offices, and industries.

Madhuri et al (2017) presented the design and implementation of an energy meter using an Arduino microcontroller which is used to measure the power consumed by any individual electrical appliance. The main intention of the proposed energy meter was to monitor the power consumption at the device level, upload it to the server, and establish remote control of any appliance. The energy monitoring system precisely calculates the power consumed by various electrical devices and displays it through a home energy monitoring website. Again this solution lacked the implementation of energy management in an efficient way and also the Artificial intelligent techniques to predict the energy consumption and minimize it.

Chapter 5 – Technologies

5.1 Machine Learning for Consumption Prediction

It is necessary to describe only technological aspects related to the solution adopted, highlighting which aspects of the technology have been used to carry out the work.

We used python to create a model based on machine learning technique first of all we worked on the implementation of linear regression based on the data we got:

Raw Data per hour:

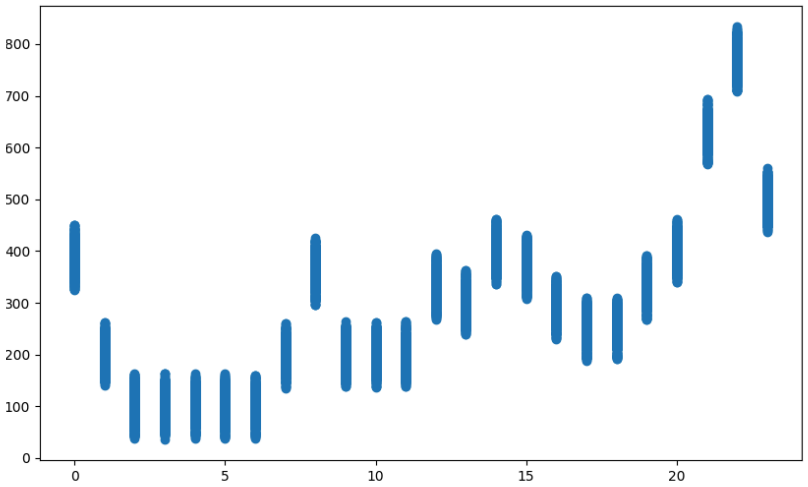

Figure 5.1.1 electric meter per hour all input data

We calculated the linear regression Figure 5.2 and obtained the following values as the best values trying with polynomial regression of degrees <= 100 the best values were obtained by degree =3 with and R^2=0.6317698409510453.

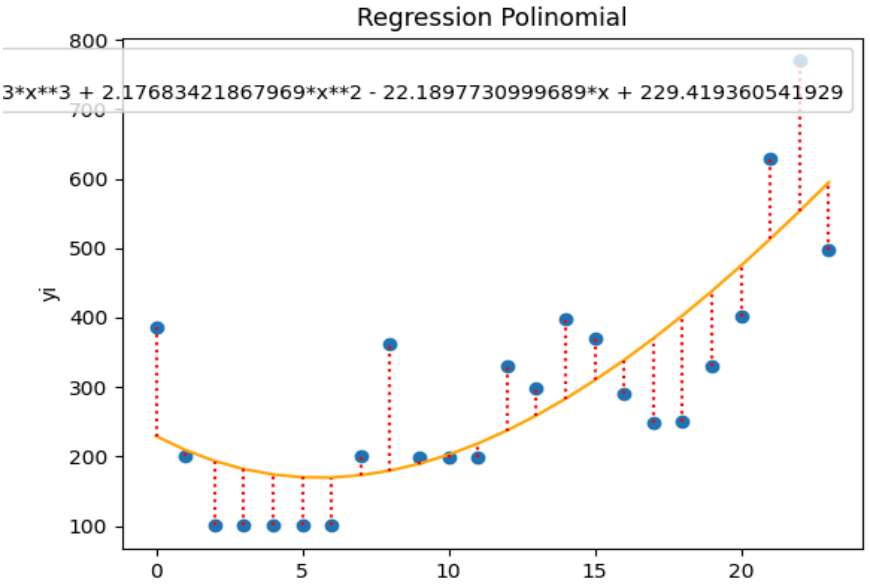

Figure 5.1.2 electric meter Polynomial regression

where Y its the consumption and
y=-0.0226643038529883*x**3 + 2.17683421867969*x**2 - 22.1897730999689*x + 229.419360541929
Because the model wasn't a good fit we used an averaged estimation with a point function in order to have functional data.

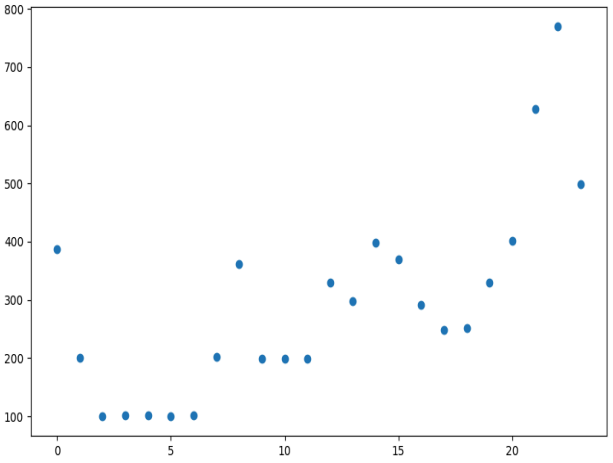

Figure 5.1.3 electric average dot function

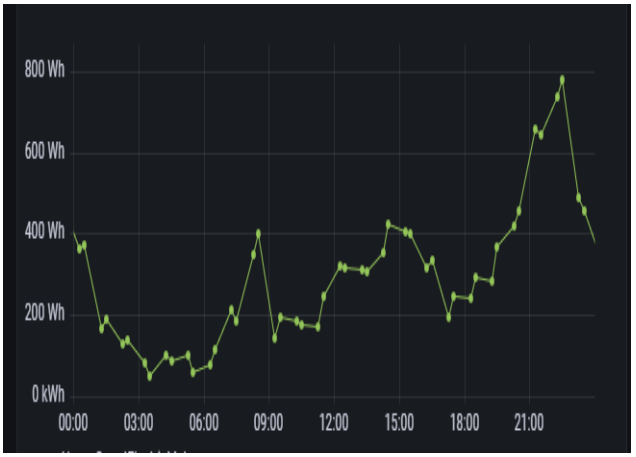

Figure 5.1.4 electric meter grafana from inputs

Why could we use this approach based on the average Figure 5.3 ? because the dispersion of the consumption is not high so we can actually obtain good results thanks to that as we see in the comparison with the actual consumption Figure 5.4.

5.2 Azure IoT Hub for Digital Twin

Azure provides its service named IoT Hub for connection, monitoring, and managing the IoT devices. We can create replicas of our IoT devices in Azure IoT hub and send the telemetry to it.

We can define message routes to other Azure services such as Digital Twin. Azure also provides service for creating Digital Twin of your IoT infrastructure. It's a digital representation of real-world things, places, business processes, and people. It enables us to gain insights that can help us drive better products, optimize operations and costs.
We are using the node package `azure-iot-device` to send the data from sensors attached to raspberry pi to the IoT hub. Once the telemetry data is received, the IoT hub shows the stats of the data in the web dashboard.

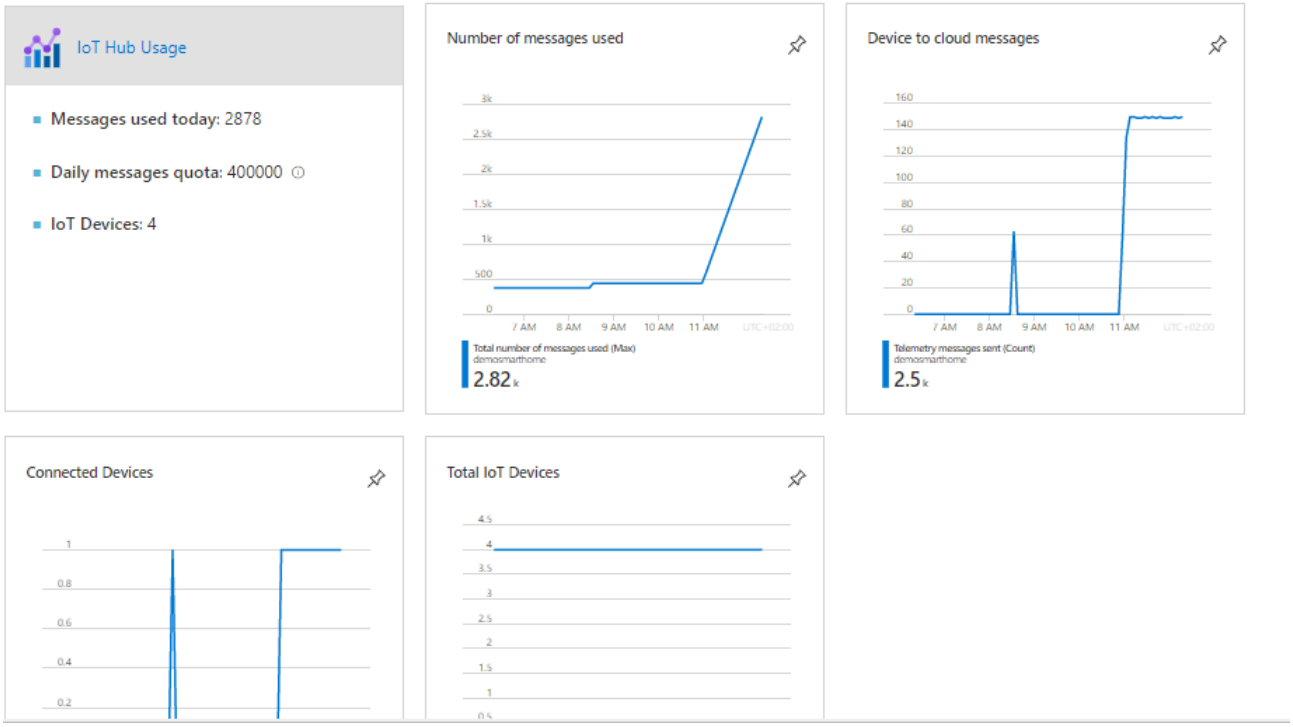

Figure 5.2.1 Iot Hub dashboard

5.3 Connection with Azure IoT hub using MQTT

We can connect to Azure IoT hub by providing an IoT hub *broker address*, *Shared Access Key* and *Username Password* and publish to certain topics. The issue arrives when we connect the IoT hub with Azure digital twin. The Digital Twin accepts the message body to be in json format, `Content-Type = application/json`. But the IoT hub sends the body of the data in the base64 encoded format to Azure digital twin if we use the node package `mqtt`.
We can resolve this issue by specifying `options` when publishing on the MQTT broker.

```
mqtt.Client#publish(topic, message, [options], [callback])
```

Publish a message to a topic

`options` is the options to publish with, including:

- `qos` QoS level, Number, default `0`

- `retain` retain flag, `Boolean`, default `false`
- `dup` mark as duplicate flag, `Boolean`, default `false`
- `properties`: MQTT 5.0 properties `object`
    - `payloadFormatIndicator`: Payload is UTF-8 Encoded Character Data or not `boolean`,
    - `messageExpiryInterval`: the lifetime of the Application Message in seconds `number`,
    - `topicAlias`: value that is used to identify the Topic instead of using the Topic Name `number`,
    - `responseTopic`: String which is used as the Topic Name for a response message `string`,
    - `correlationData`: used by the sender of the Request Message to identify which request the Response Message is for when it is received `binary`,
    - `userProperties`: The User Property is allowed to appear multiple times to represent multiple name, value pairs `object`,
    - `subscriptionIdentifier`: representing the identifier of the subscription `number`,
    - `contentType`: String describing the content of the Application Message `string`
- `cbStorePut` - `function ()`, fired when a message is put into `outgoingStore` if QoS is `1` or `2`.

As we can determine from the documentation the options property is introduced in version 5 of MQTT.

```
const options = {
   protocolVersion: 5,
   properties: {
       payloadFormatIndicator: 1,
       contentType: "application/json"
   }
}
```

But this does not work as MQTT protocol version 5 is not supported by Azure IoT hub. MQTT version 4 was publishing the data to IoT hub in base64 format. So we have to change the node package from using mqtt to `azure-iot-device` which gives us the option to specify the Content-Type = application/json and Payload Format to be utf-8.

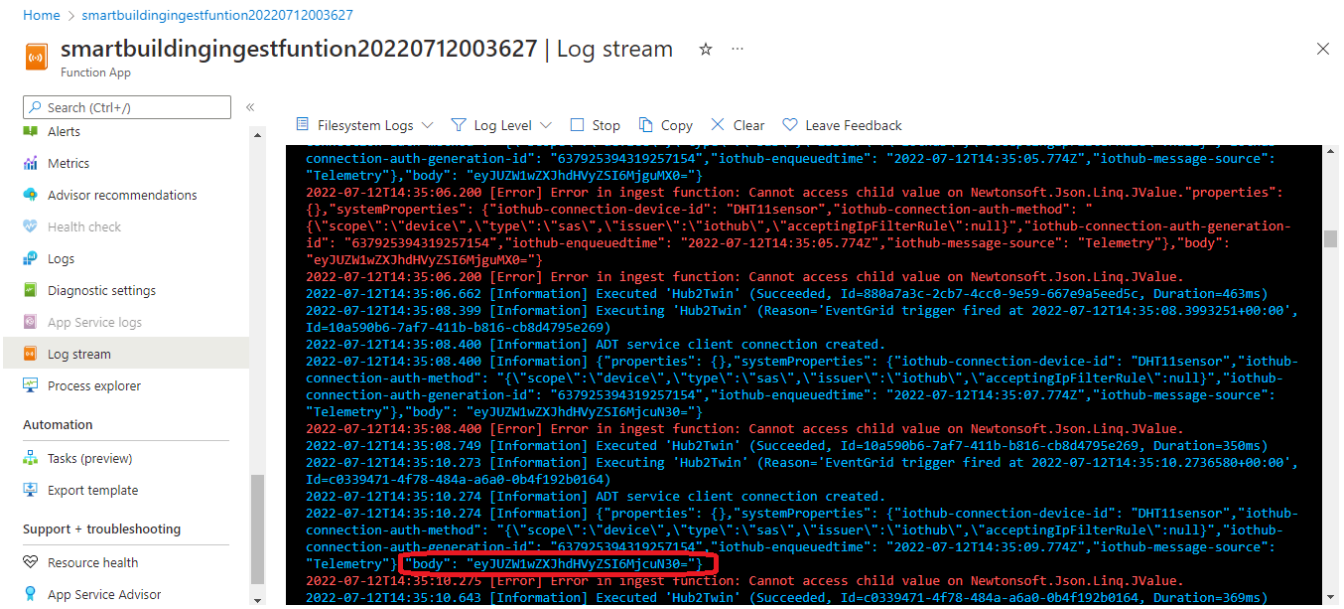

Figure 5.3.1 ingest function

So we are bound to use the Azure software development kit , Azure SDK for Node . This document is also very helpful for starters.

5.4 IoT Hub to digital twin

We have received data from our IoT devices to Azure IoT hub. We need to ingest this data to the digital twin model. Azure Digital Twins is driven with data from IoT devices and other sources. A common source for device data to use in Azure Digital Twins is IoT Hub.

The process for ingesting data into Azure Digital Twins is to set up an external compute resource, such as a function that's made by using Azure Functions. The function receives the data and uses the DigitalTwins APIs to set properties or fire telemetry events on digital twins accordingly.

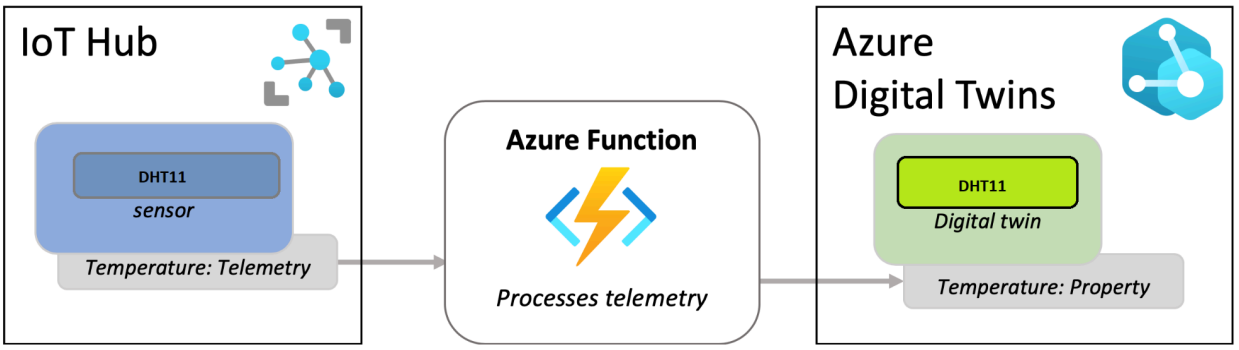

Figure 5.4.1 Iot Hub & Digital Twin connection

Chapter 6 – Proposed solution

The system will forecast the energy consumption based on the data collected from the user habits, sensors and weather forecast. The system will suggest the user's best time to plan the usage of appliances.

The system tries to minimize the energy consumption by forecasting the energy consumption and suggesting to the user which time is the best for a specific appliance to operate. For instance, which time is the best to plan the laundry when there is very little energy consumption.

The system will turn off lights when the user when the battery percentage is in bad values or the hours are bad.

The system will generate notification about the solar panel efficiency drops to advise the human that maybe needs some maintenance.

The system will upload the telemetry to a digital twin in order to be able to see in almost real time the state of the sensors in the building.

The system will also upload the information about telemetry and some graphs into a front end app.

The system will take information about sensors and decide which actuators are going to be executed

Chapter 7 – Functional validation

Describe how the application was validated from a functional point of view. To do this, you can describe a real use case, show any hardware used and describe how it is used in such a scenario, accompanying the description with appropriate screenshots.

7.1 Raspberry Pi

We have a Raspberry Pi and we Install Raspberry Pi OS using Raspberry Pi Imager in order to be able to access the raspberry pi. We recommend to follow https://elearning.unisalento.it/pluginfile.php/479119/mod_resource/content/2/Lesson%206_Embedded%20Systems_2021-2022.pdf where its explained how to connect it and enable the ssh conection to conect the device to our pc.

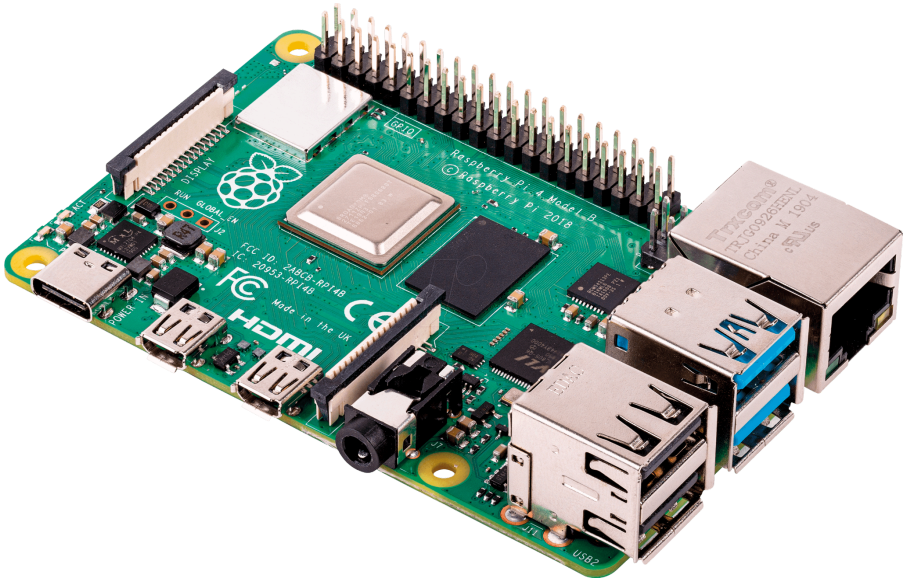

Figure 7.1.1 Raspberry pi 4

After this we will need to prepare our raspberry pi with the following circuit and devices connected:

DHT11 SENSOR: connected to GPIO 4

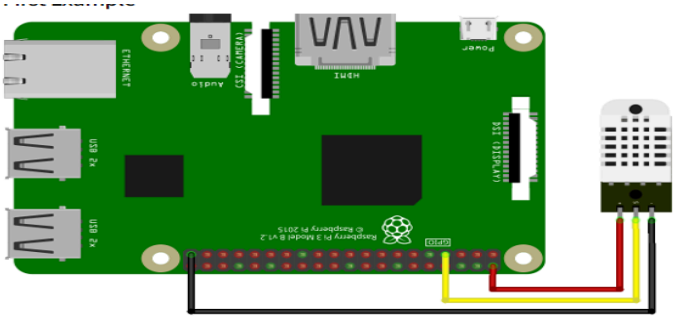

Figure 7.1.2 Raspberry pi 4 connection to DHT11 sensor

LED ACTUATOR: connected to GPIO 17

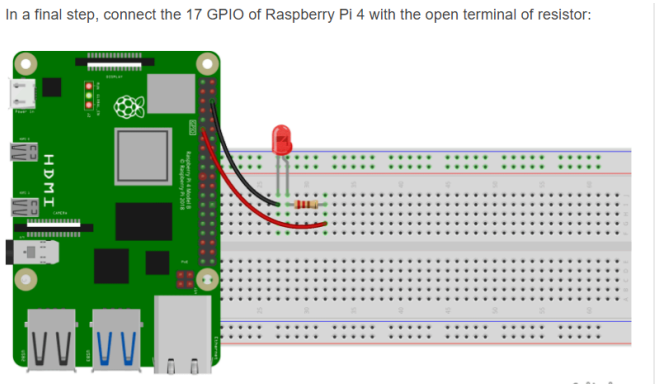

Figure 7.1.3 Raspberry pi 4 connected to a led

7.2 Sensors and Actuators activation:

So we go to our Raspberry Pi and obtain from our github repositories the code of sensor and actuators in order to execute them in the device:
Links:
sensors:https://github.com/UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-sensors-ahsan-adriano
actuator:https://github.com/UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-actuator-ahsan-adriano

In our pc we should run the sql and the forecasting modules
Links:
serverSQL:https://github.com/UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-server-ahsan-adriano
recomendation:https://github.com/UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-recommendation-ahsan-adriano

Sensors will be connected through mqtt and the broker, Also the sql module will be subscribed and save data for the forecasting system in our SQLAzure (All tables are generated by the code the first time we connect to mysql, we only need to create the DB).

For obtaining the SAS Key you should install AzureIotExplorer with your IoT Hub devices created. Follow the figures 7.2.1 ,7.2.2 & 7.2.3

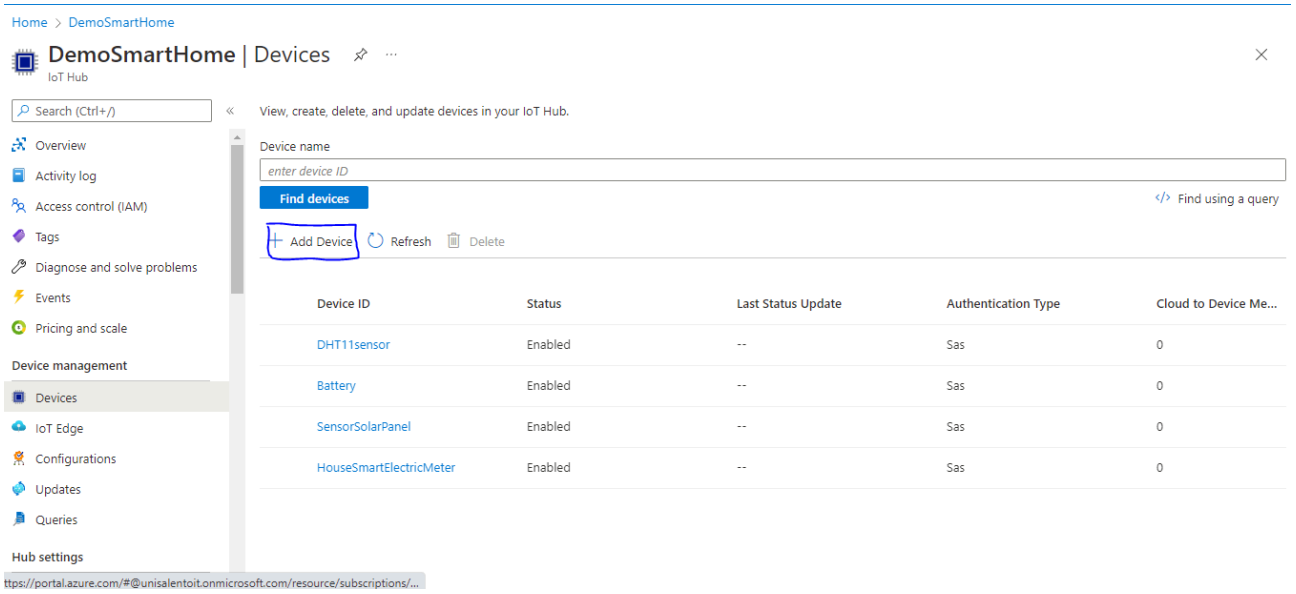

Figure 7.2.1 Iot Hub Device creation 1

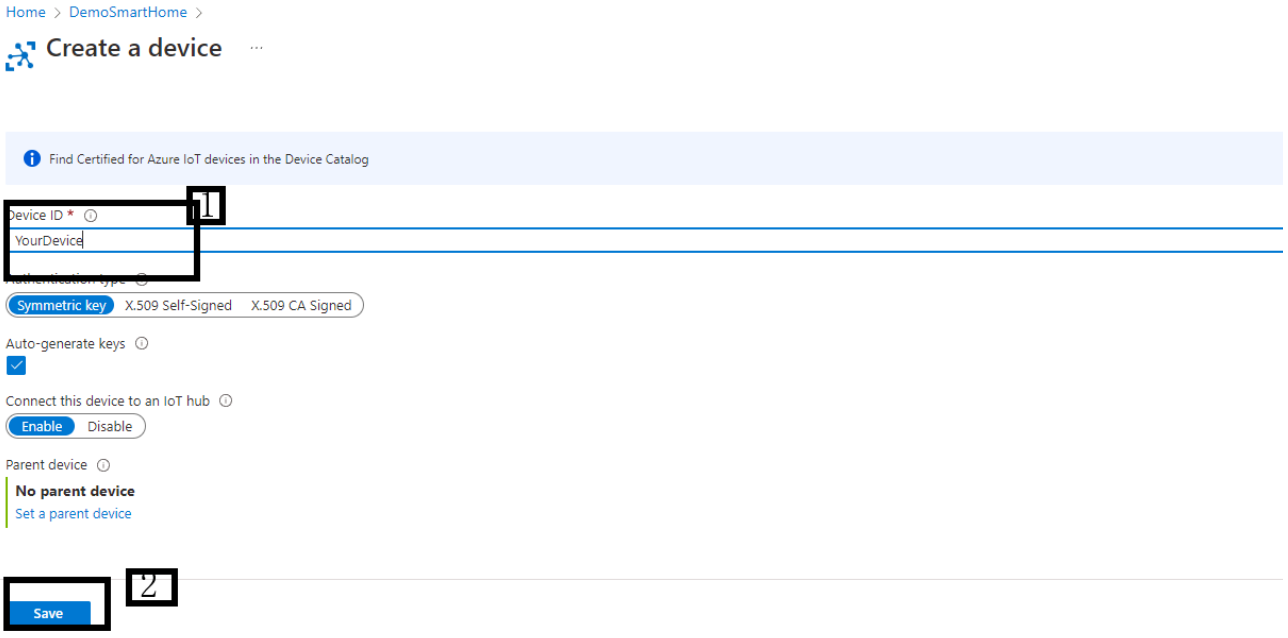

Figure 7.2.2 Iot Hub Device creation 2

Once generated, open AzureIotExplorer on your Desktop.

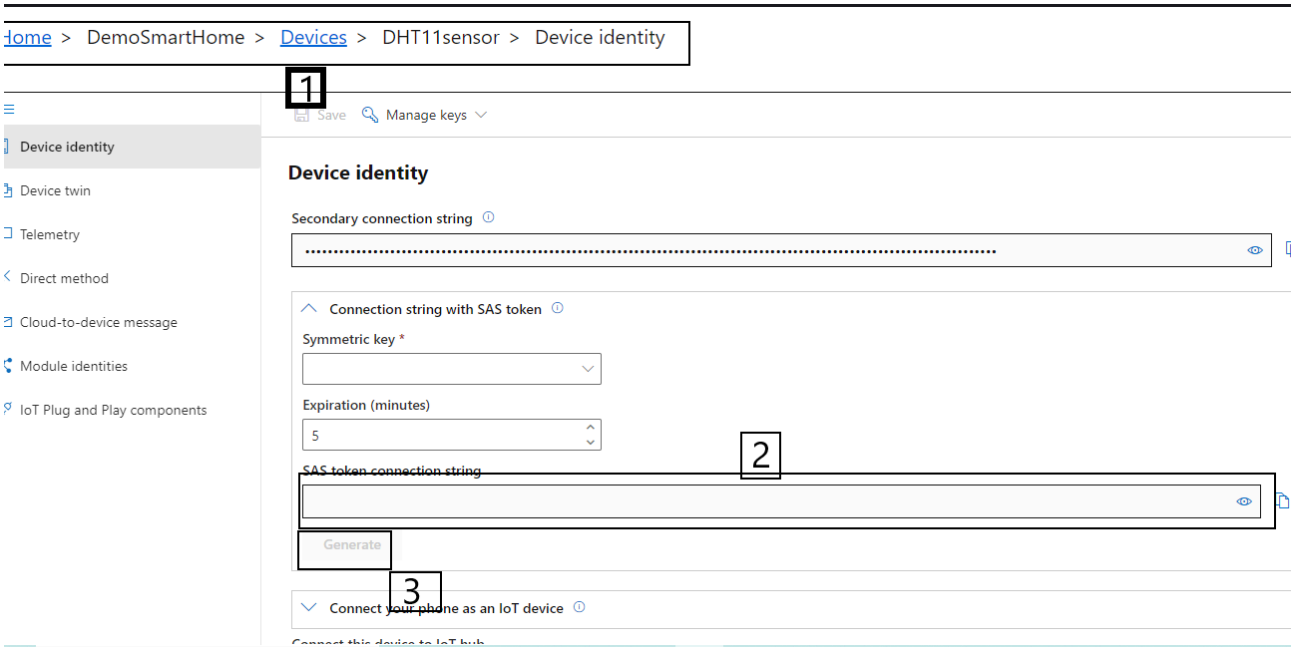

Figure 7.2.3 Azure IotExplorer

Head to your IotHub and select your device Figure 7.2.3 we recommend after step 2 in the expiration date you select a month or a larger date because can be annoying to have to change them each day/week
so we recommend expirations minutes of 44640.(Minutes of a 31 days month)

7.3 Grafana

Grafana is an open source analytics and interactive visualization web application. It provides various kinds of dashboard panels like Graphs, Score card, gauge, bar gauge, pie chart etc. It also has the feature to integrate with Time series databases such as prometheus.
We can deploy Grafana by two methods.

1. Azure Managed Instance of Grafana
2. Raspberry Pi
3. Linux / Windows server instance (on cloud or on premises)

To deploy Grafana, follow this document.
Once it's deployed, login and import the dashboard that we have created.

https://github.com/UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-grafana-ahsan-adriano

7.4 IoT Hub Telemetry

This is an example of an IoT Hub dashboard. One the telemetry data is being published to the IoT hub through the MQTT protocol, we could view the stats in the IoT Hub welcome page.

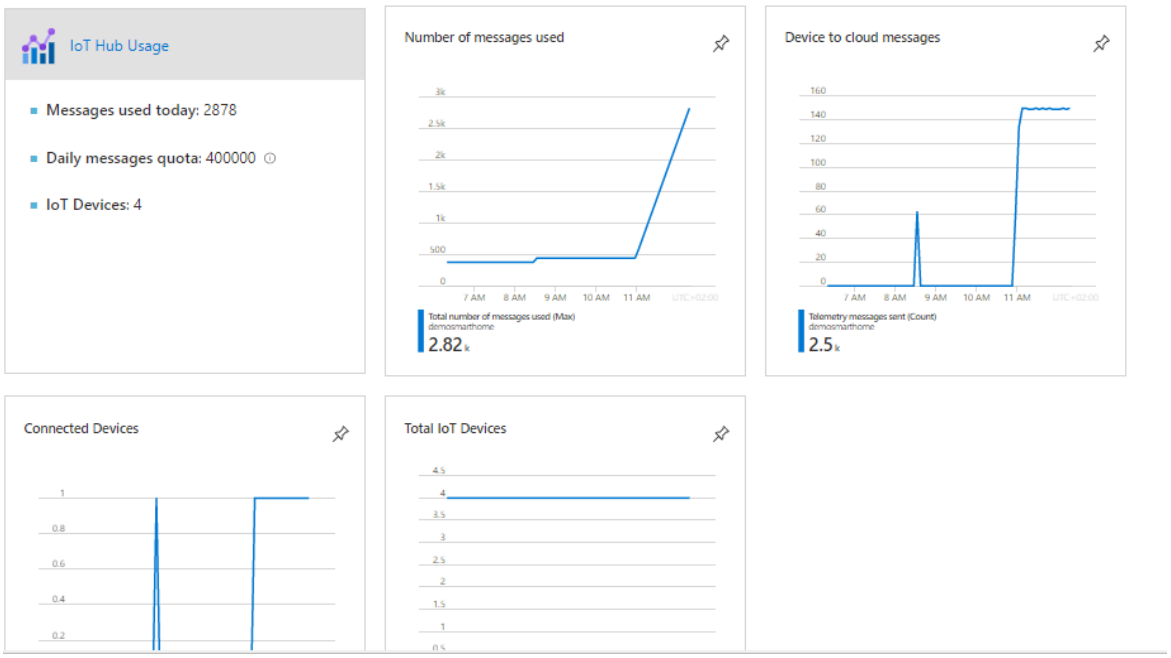

Figure 7.4.1 IotHub telemetry

We create the same devices in the IoT hub that we have in real life. For instance we have a temperature sensor that sends the telemetry data. We can create a device with the name of *DHT11sesnor* in IoT hub that will receive all the data that we are sending from the publisher, i.e. Raspberry Pi.

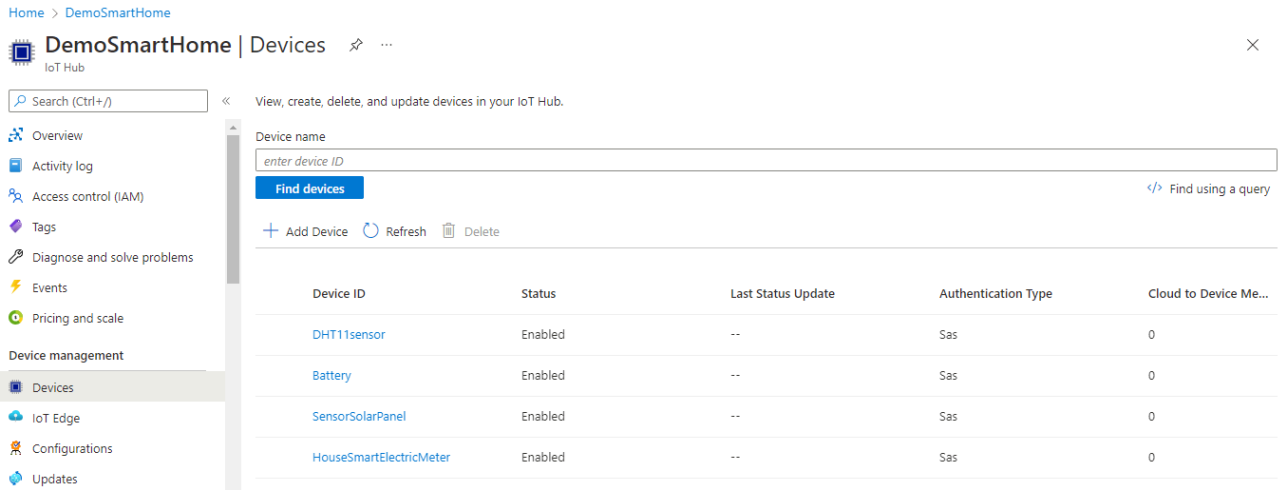

Figure 7.4.2 Devices

This is the view of devices we have created in our Azure IoT Hub.

We established the connection between IoT hub and Raspberry Pi by using a secure method that node package `azure-iot-device` uses. It uses Primary Connection String. We can find the Primary Connection String if we click on the device name in the IoT Hub dashboard. Save these connection strings somewhere,we are going to need them at a later stage.

7.5 Create Digital Twin

We need to create a digital twin instance in order to get the telemetry data from Azure IoT Hub to ingress the data to itself by using Azure Functions. We can follow this official documentation from Microsoft to create a digital twin instance on Azure IoT hub.

After creating the digital twin instance on Azure, we need to upload the DHT11sensor model to the instance we have just created, which describes the properties of a sensor and will be used later to create the actual twin.

```
{
    "@id": "dtmi:com:example:DHT11sensor;1",
    "@type": "Interface",

    "displayName": "DHT11sensor",
    "contents": [
        {
        "@type": "Property",
        "name": "temperature",
        "schema": "double"
        },
        {
        "@type": "Property",
        "name": "timestamp",
        "schema": "string"
        },
        {
            "@type": "Property",
            "name": "sensor",
            "schema": "string"
        }
    ],
    "@context": "dtmi:dtdl:context;2"
}
```

We can upload the model by using Azure Digital Twin Explorer. Simply navigate to https://explorer.digitaltwins.azure.net/

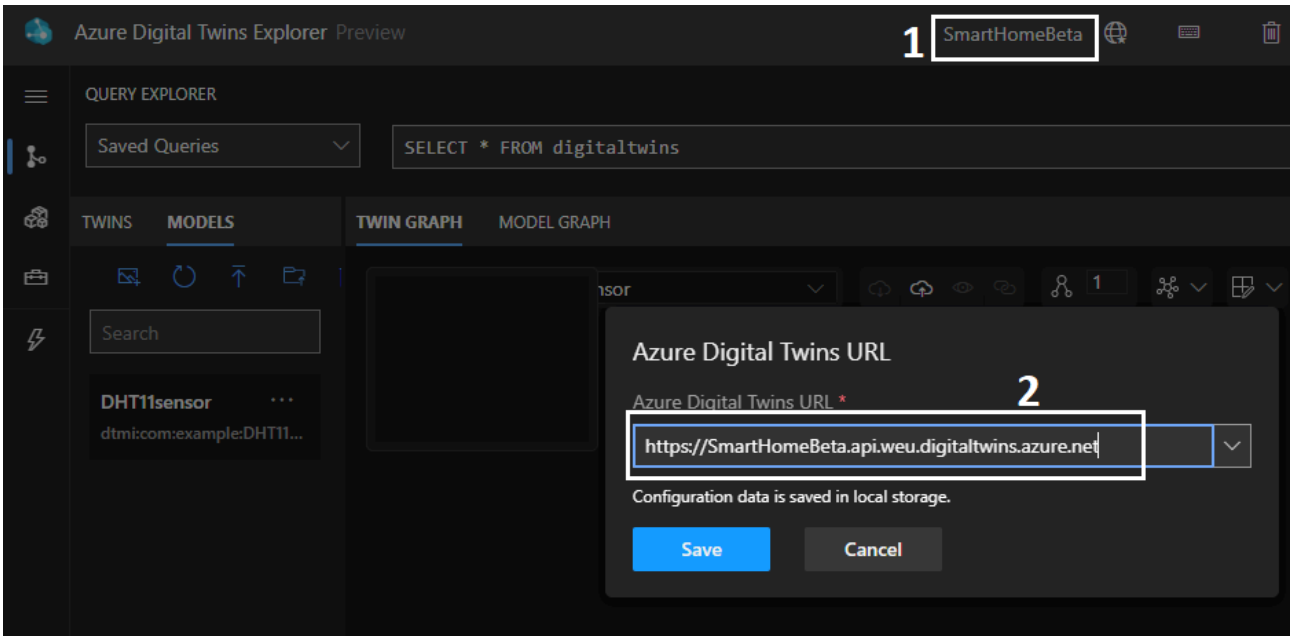

Figure 7.5.1 Digital Twin

If Digital Twin Explorer is not able to make connection with Azure Digital Twin instance, manually copy and paste the instance URL as instructed above.

Click on the Upload icon on the left side of the application and upload all four models. These models can be obtained in the github repository

Once the model is uploaded through Azure Digital Twin Explorer, we need to deploy an Azure function to ingest the telemetry data from IoT hub to Digital Twin. We can follow this well written document by Microsoft to deploy the function.

Once the function is deployed, we need to grant this function access to the Azure IoT hub without using credentials. For that purpose, we will use a system assigned managed identity. First navigate to the function instance on Azure and click on Identity. Then turn the `status On` and click on `Azure role assignments`.

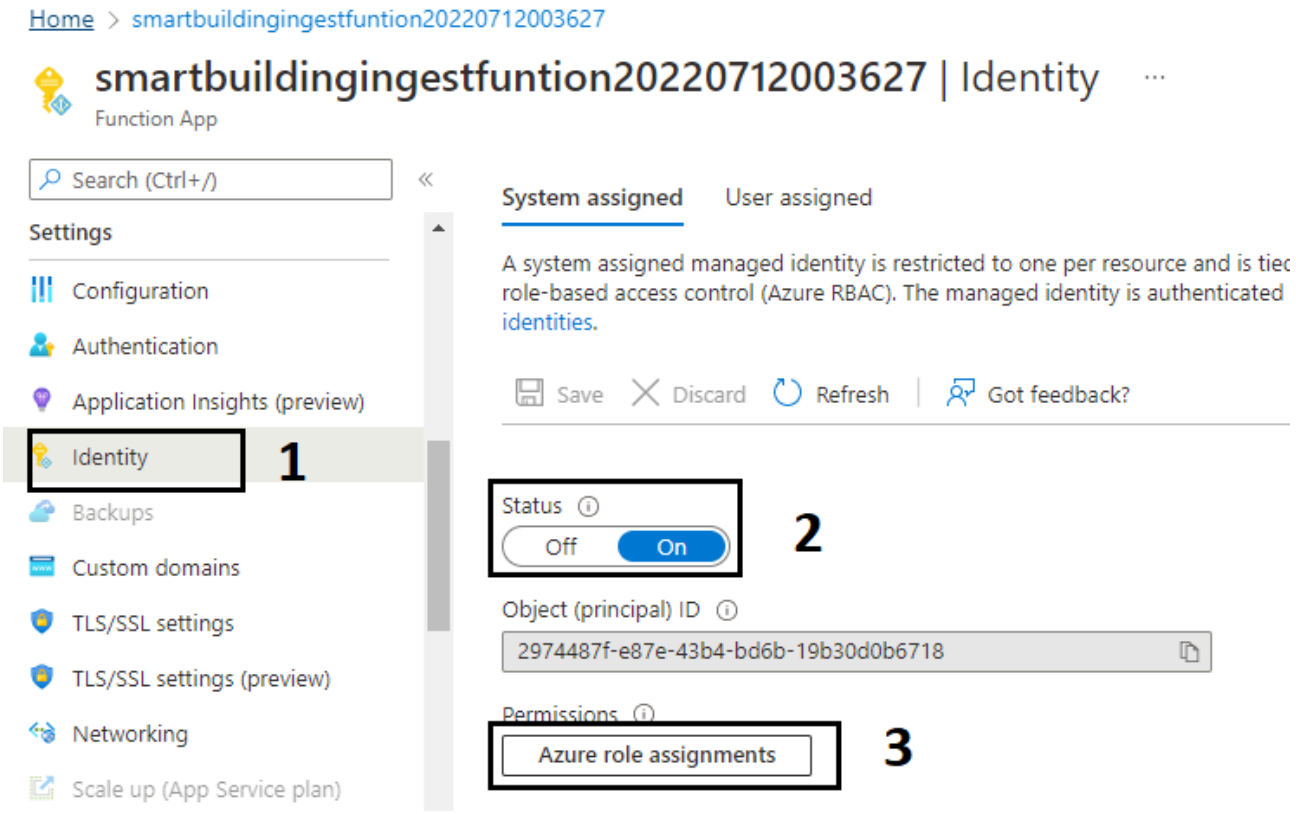

Figure 7.5.2 Ingest Function setup

A new interface will be opened. select the options as it is indicated in the following steps.

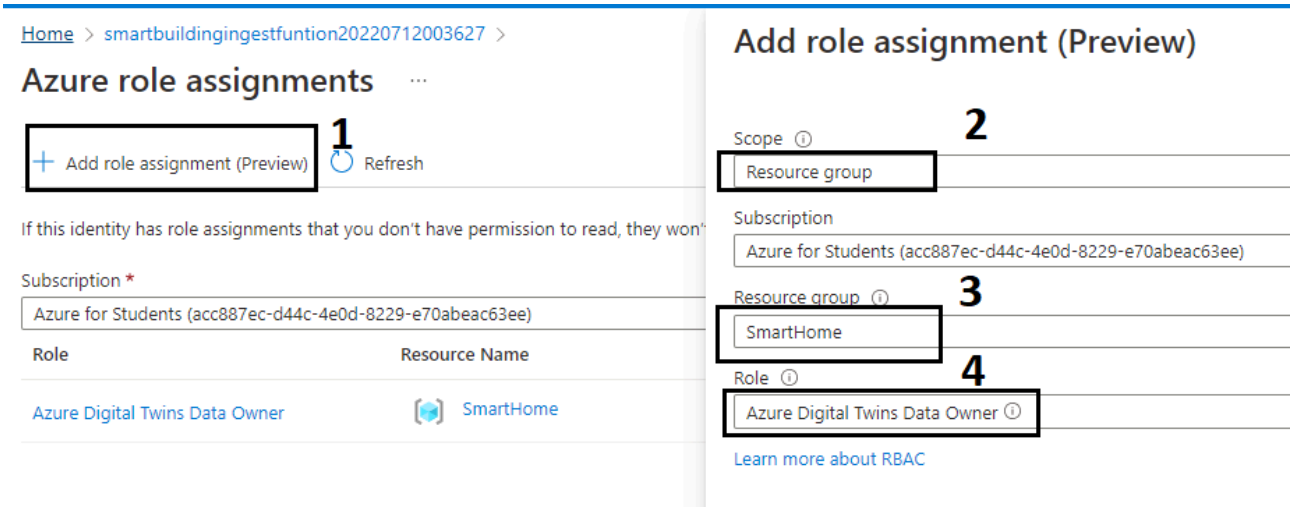

Figure 7.5.3 Ingest Function assign roles

Save the configurations by clicking on save.
Now we need to set the environment variable in order to make accessible our Digital Twin instance to the function we have just deployed. To do that, first navigate to the Azure Digital Twin instance web page on Azure and copy the hostname of the D.T. instance.

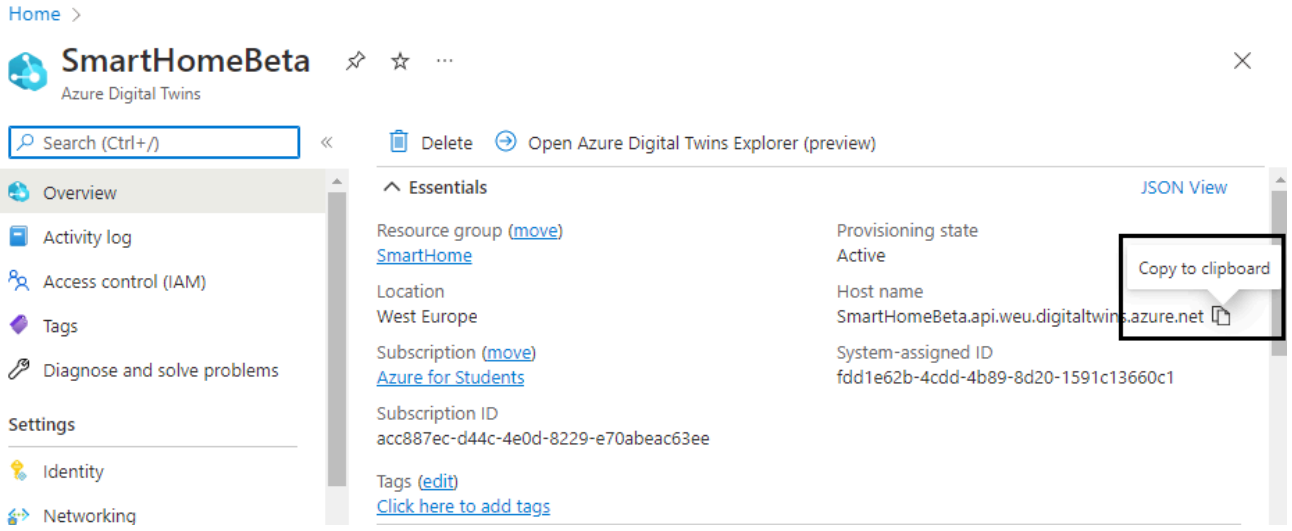

Figure 7.5.4 Azure digital twin obtain Host path

Now navigate to the function instance again and from the menu on the left side, select `configuration` and then click on +`New application settings`.

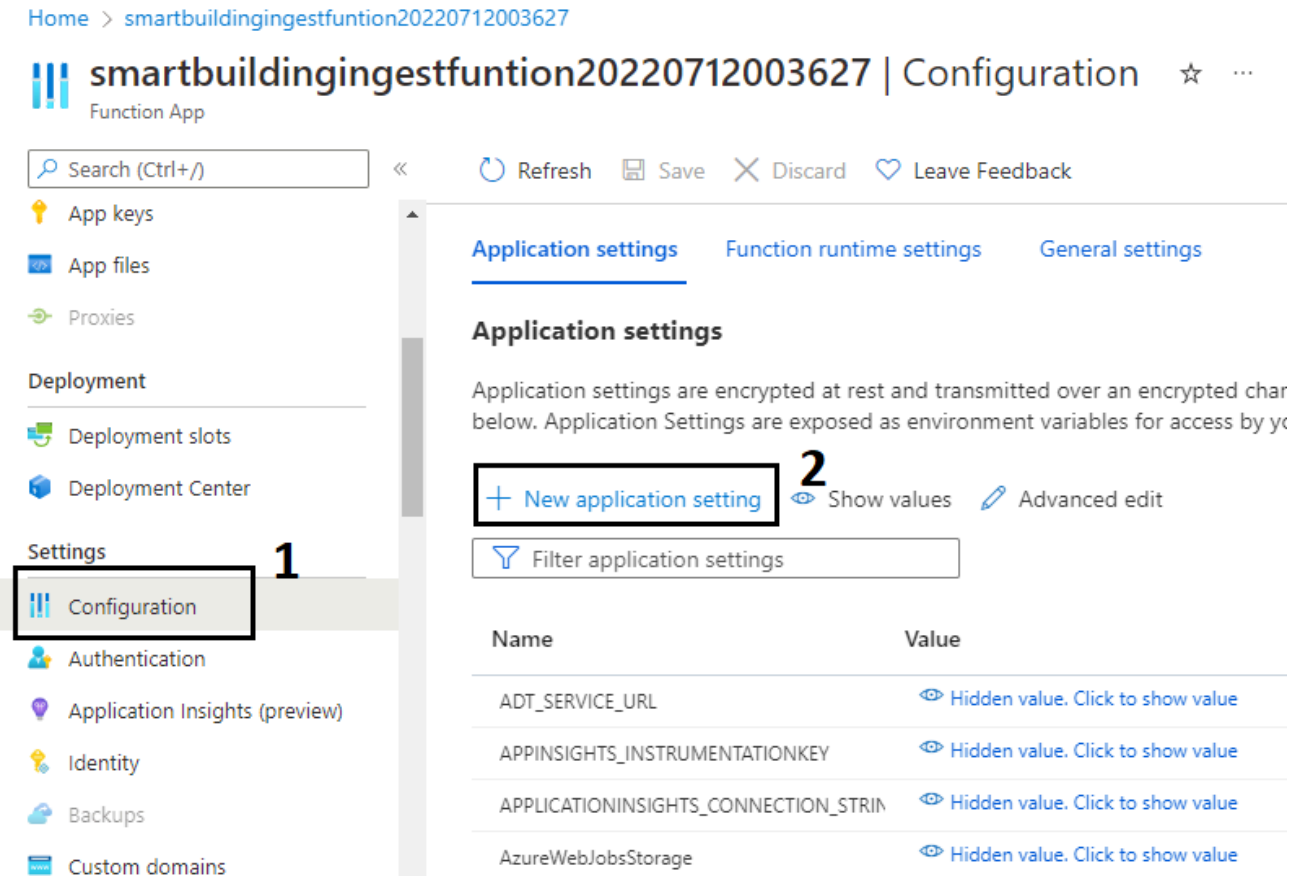

Figure 7.5.5 Ingest Function configuration

Update the fields in the following manner and save the configuration twice.

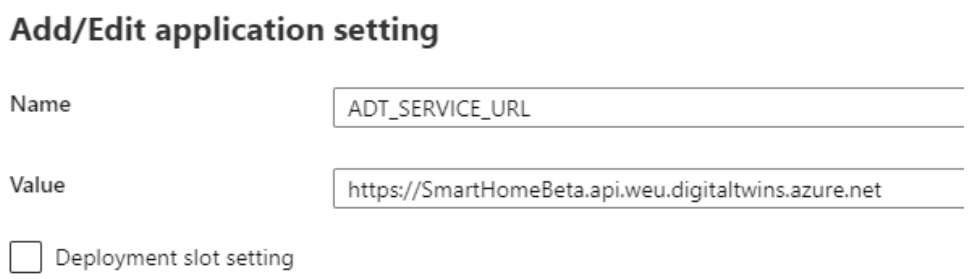

Figure 7.5.6 Adding host path

Now we need to configure the IoT hub to send event messages coming from Raspberry Pi to the function app. To do that first navigate to the IoT hub instance and click on Events and then click on Event Subscription.

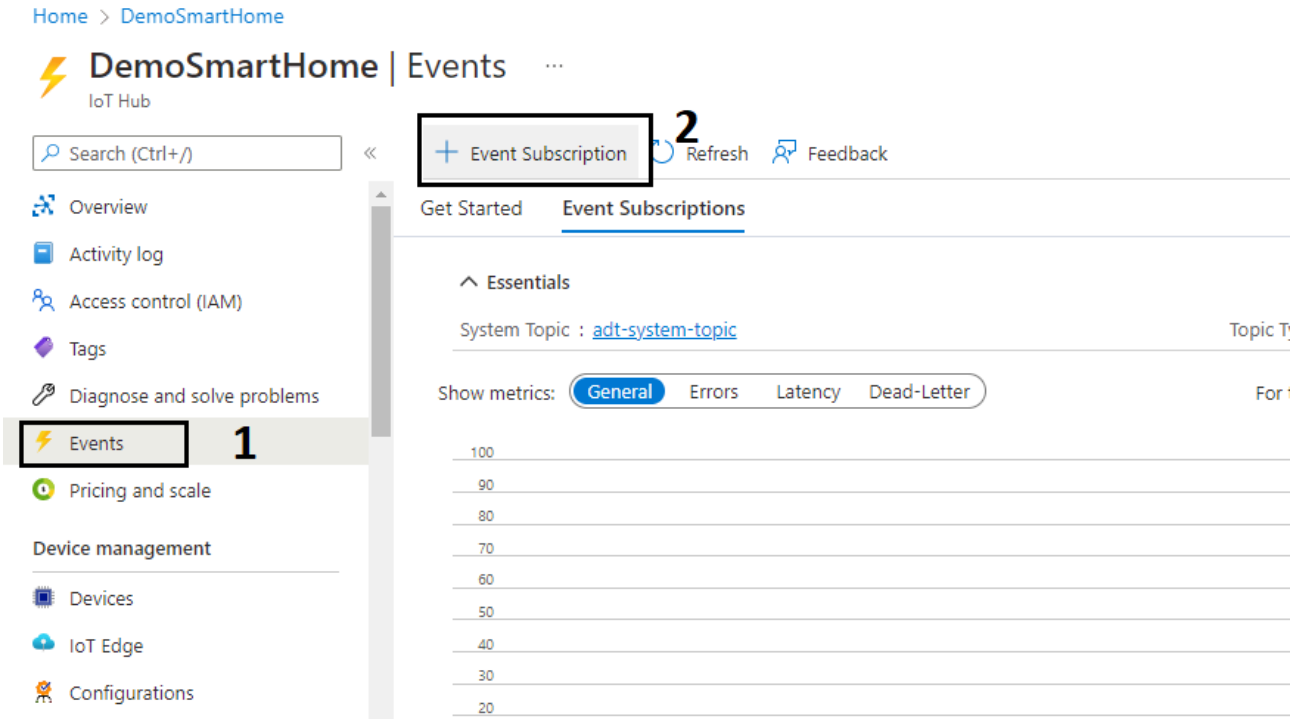

Figure 7.5.7 Events Subscription

Put the Event Subscription Details and Event Topic as shown below.

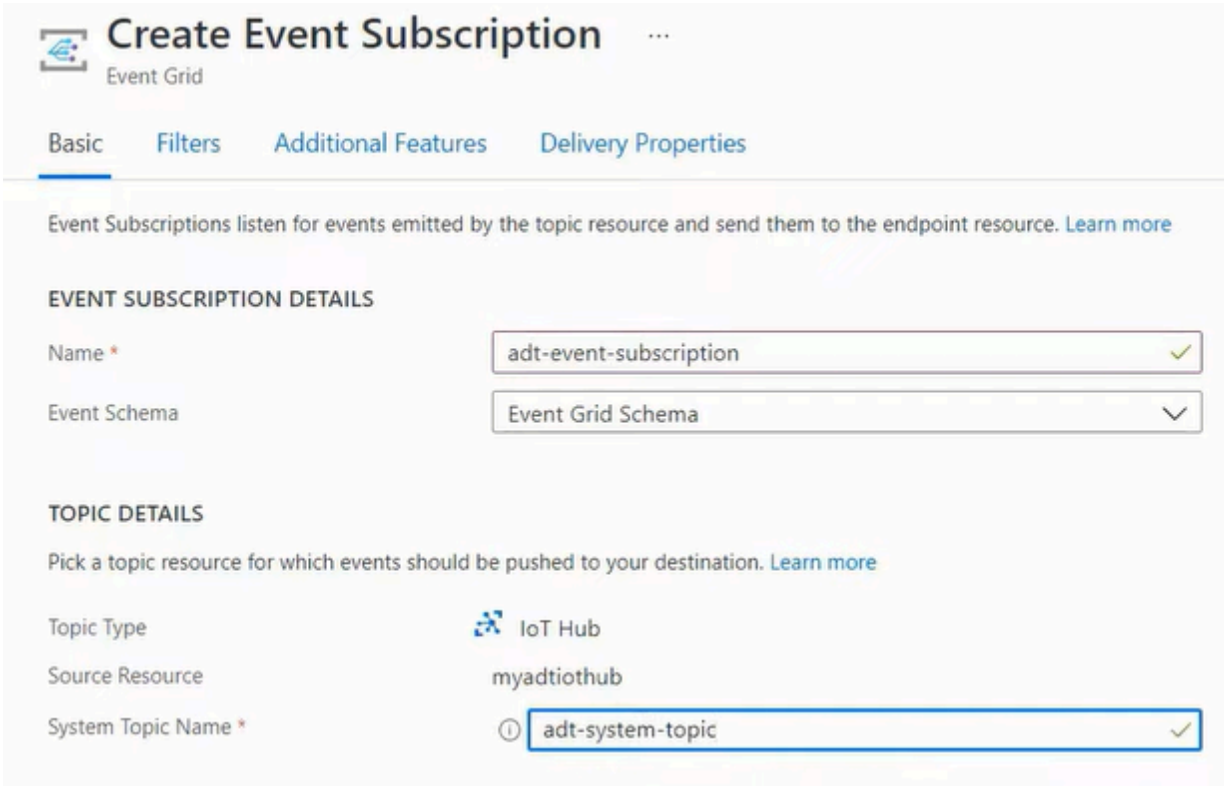

Figure 7.5.8 Events Subscription creation

In Events Type, select Device Telemetry only.

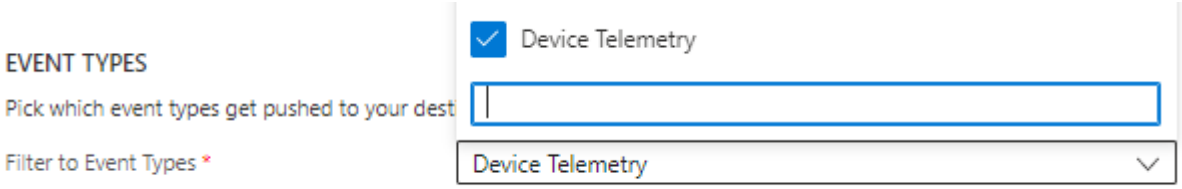

Figure 7.5.9 Events Subscription select telemetry

For Endpoint Details, select Azure function

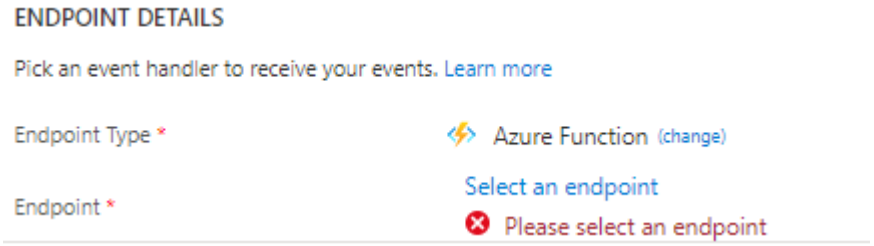

Figure 7.5.10 Events Subscription select endpoints

Now click on Select an endpoint, and select the function app we have deployed earlier.

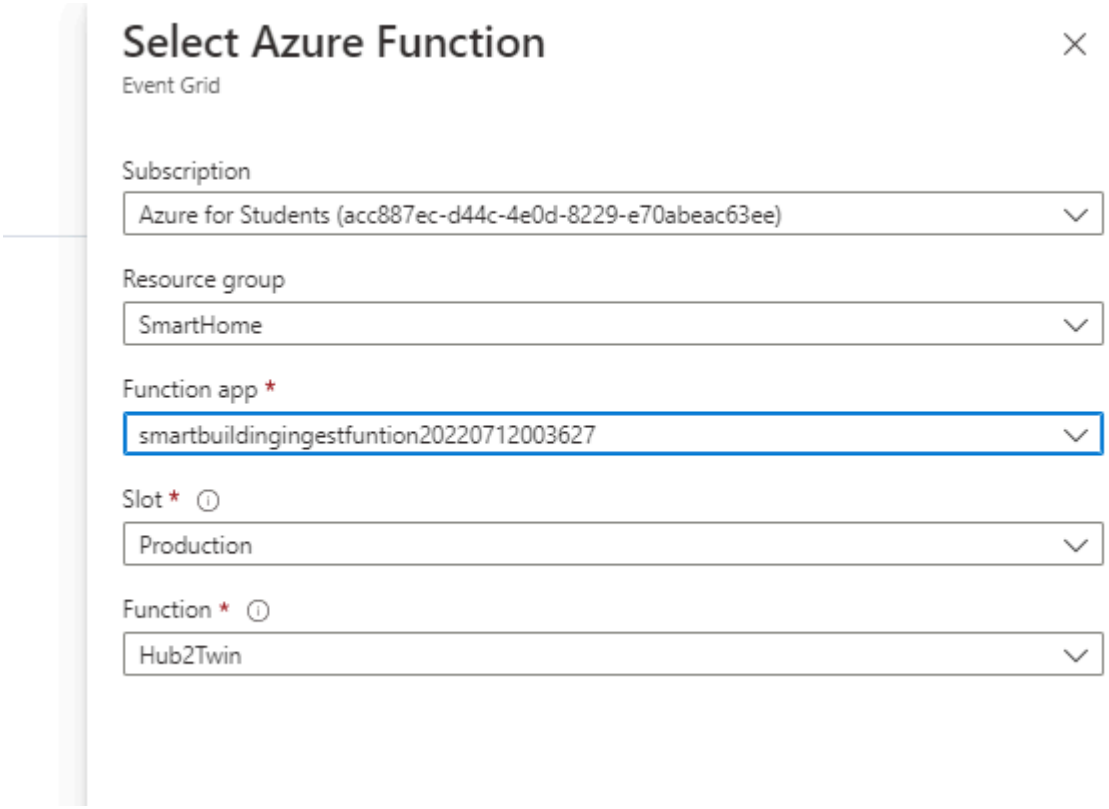

Figure 7.5.11 Setting events function

click on save to save the configuration.
We have successfully created the event subscription for Azure digital twin from IoT Hub.
Now we need to start Raspberry Pi to send the sensor data to IoT Hub which will push the events to the digital twin instance.

7.6 Publish to MQTT and Azure IoT Hub:

We need the Primary Connection Strings obtained for each device from IoT Hub and update these strings in the code we will clone from the github classroom repositories.
First we need to clone the code from github. This is the link to the git repository.

For sensors:

git@github.com:UniSalento-IDALab-IoTCourse-2021-2022/wot-project-sensors-ahsan-adriano.git

For Server:

git@github.com:UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-server-ahsan-adriano.git

For Actuator:

git@github.com:UniSalento-IDALab-IoTCourse-2021-2022/wot-project-actuator-ahsan-adriano.git

We have 4 sensors so we need to update the Primary Connection String in 4 modules of every sensor in the `sensors/` directory.
Once it's done, run the `server` component first and then start the individual sensor and then actuator.

Now clone the repository on either Raspberry Pi or on a virtual machine where you want to run the prediction of the energy consumption.

git@github.com:UniSalento-IDALab-IoTCourse-2021-2022/smartbuilding-recommendation-ahsan-adriano.git

7.7 Digital Twin Explorer:

We now navigate through the digital twin we have just created and verify whether the digital twin is receiving telemetry data from our Raspberry Pi or not. We need to navigate to the Digital Twin explorer again. Now click on the run query, we should be able to see the sensors as Digital Twin in the dashboard. In our example, we have a temperature sensor available as a digital twin and it updates the values after the interval set in Raspberry Pi Sensor application code.

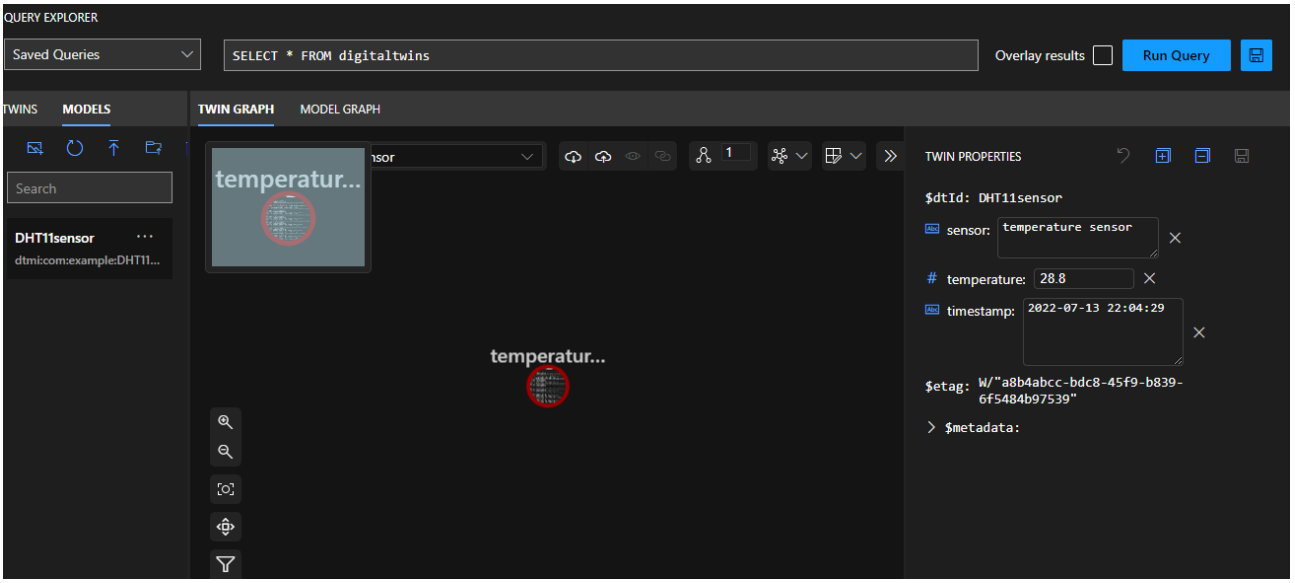

Figure 7.7.1 Digital Twin Query

We may also run different queries like, if we have more than one temperature sensors as digital twin, we can query to get only the sensors placed in certain rooms where the temperature is greater than 20c

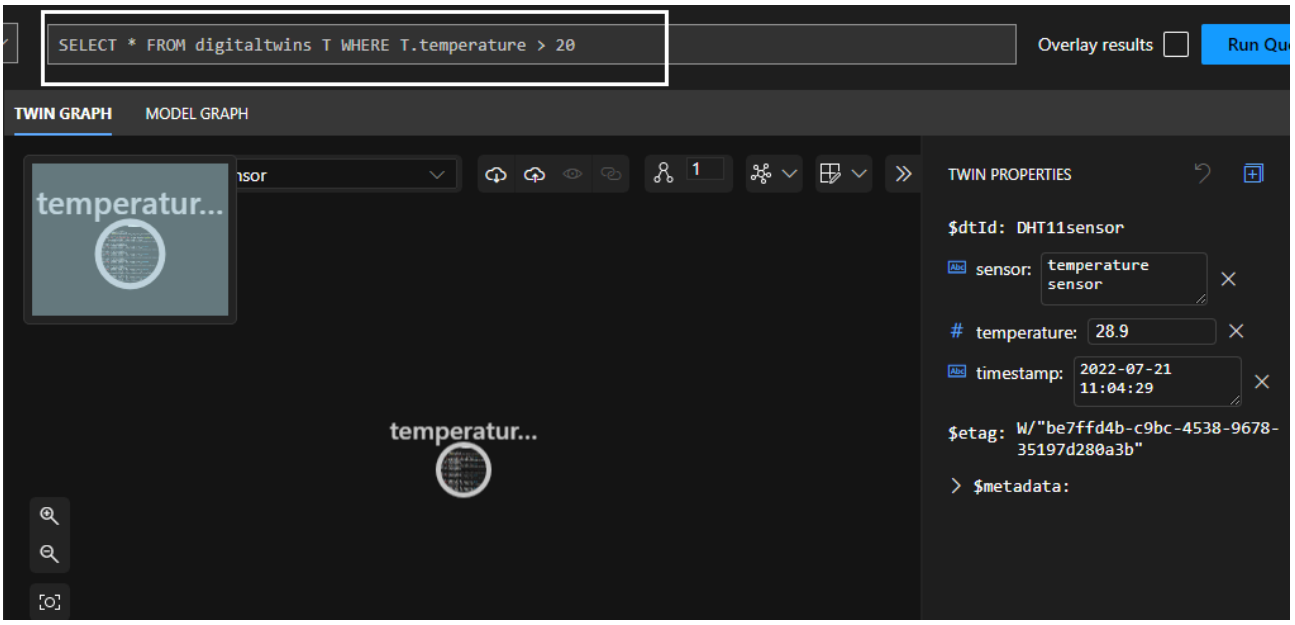

Figure 7.7.1 Digital Twin Query syntax

Chapter 8 – Conclusions and future works

Summarize the goal, highlighting how this has actually been achieved, and mention any detail that may improve this work.

So we accomplished the main goal of this project that was to create a functional SmartHome system and also the Digital twin.
Supporting and building a state of the art that can be updated and changed according to the necessities .

The only goals we didn't accomplish is to integrate in a satisfactory way an IA model in the forecasting part due to the limitations on time and quality data we have, and to find a dataset to integrate the recognition of the user habits.

So the project can be improved in those aspects and also in the building of a 3d model for the Digital Twin part.
We can package the code in Docker containers and ship it with one single docker compose file which will help us deploy all the code in Raspberry Pi with one line of command.