Häme University of Applied Sciences – HAMK

University Centre – Hämeenlinna

Smart Services Research Unit

IoT Platform Analysis for Educational Purposes

IoT-Ticket

Keywords: IoT Platform, Smart Campus, IoT-Ticket

Project developed under the exchange program *PROPICIE 14th ed.*, in cooperation between the Brazilian institution *Federal Institute of Santa Catarina – IFSC* and the Finnish institution *Häme University of Applied Sciences – HAMK.*

Advisors:

Joni Kukkamäki (HAMK)

Mario de Noronha Neto (IFSC)

Author:

Henrique Hilleshein

Hämeenlinna – FI, December 2018

1. TABLE OF CONTENTS

2. Introduction

By the need of a better insight and control of the environment around someone or something, the concept internet of things (IoT) was created. IoT is technology paradigm designed to connect machines and devices in a network, creating a global network of machines and devices capable of interacting with each other [1]. The devices can be sensors, RFID, computers or anything that gives useful critical and non-critical information about an environment. Using the information received from devices is possible to do predictions and take action when necessary.

The result of many connected devices sending data is a big amount of data that needs to me stored, processed and presented in an efficient and easily interpretable way [2]. To handle this big amount of data and exploit the IoT capability, an IoT platform can be used. An IoT platform is a tool that takes care of the storage, processing and presentation of data, so everything can be controlled and monitored in one place.

It is possible to find many IoT platforms, some of them are open source, like ThingsBoard, Kaa IoT Platform and WSo2. There are commercial platforms as well, such as IoT-Ticket, Microsoft Azure IoT, Amazon Web Services IoT.

A work about open source platforms was already made [3]. In this work the proposal is to evaluate a commercial IoT platform called IoT-Ticket. The idea is to assess the viability of the IoT-Ticket in some scenarios for future projects and also to be used as an educational tool to teach IoT to students.

3. Objectives

Main Objective:

Evaluate the viability to use IoT-Ticket as a IoT platform for educational and research purposes.

Specific Objectives:

- *Create and register devices to the platform*

- *Communicate to the platform using this IoT communication protocols: MQTT, HTTP, OPC-UA;*
- *Access external systems using the platform;*
- *Export the stored data using CSV or XLS;*
- *Deployment of IoT-Ticket in own server and cloud;*
- *Conduct data analytics using the platform;*
- *Configure the platform user management;*
- *Make a practical implementation testing the communication of the IoT platform and the protocols previously highlighted;*
- *Validation of real-time data visualization tools and widgets provided by the platform;*

4. Theoretical Approach

In this section it is explained IoT, IoT platform, IoT protocols in a succinct way. These concepts are needed to understand the proposal of this work.

4.1. IoT

IoT is a concept that sometimes is confused with M2M concept. The goal of IoT and M2M is communication between devices. It enables data collection, storage and exchange. The devices can make decisions based on the data and perform tasks with very minimal human intervention [4].

The main difference between IoT and M2M is the connectivity. M2M depends on point-to-point communication through a proprietary cellular or wired network. On other hand, IoT relies on IP network to communicate the device data to the cloud or middleware platform [4].

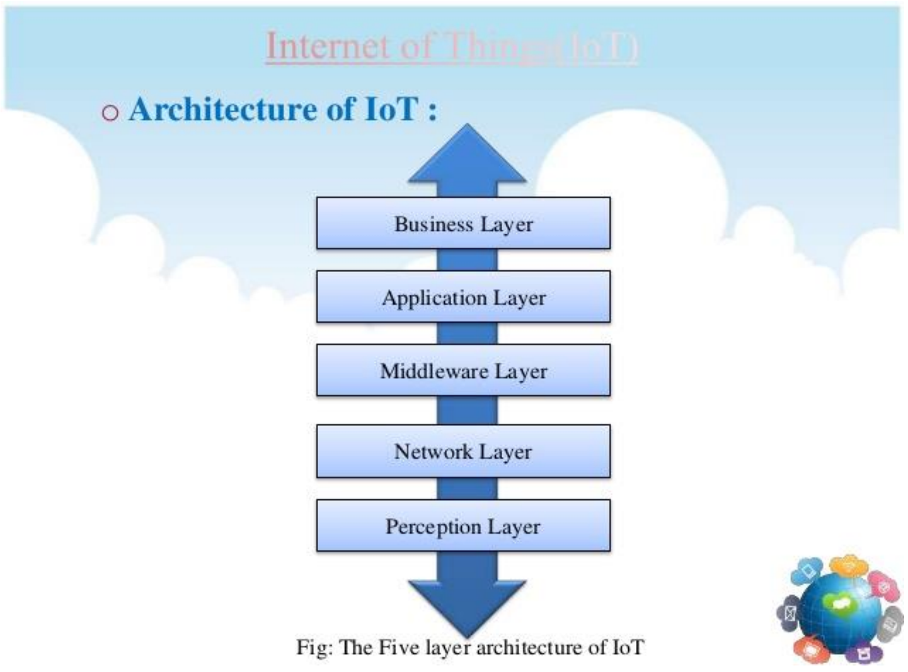
M2M is about connecting devices and IoT has the purpose of create a network of connected devices. IoT uses M2M connectivity, integrate web applications, and connects to the cloud [4]. M2M usually gives just raw data and it is used for point solution systems. IoT give a better insight of the devices and enables integration with big data, analytics and other applications.

The next subsections will give to the reader a macro view of IoT, using five-layered IoT Architecture concept and IoT applications examples.

4.1.1. Five-Layered IoT Architecture

The five layered IoT architecture it is a way to understand and explain IoT. It separates all that is needed to IoT works in five layers. The five layers can be seen in Figure 1.

Figure 1. IoT five-layered architecture



Source: www.slideshare.net/radhamahalle/a-brief-review-on-evolution-of-communication-technologies [Accessed 27 November 2018].

A brief introduction to the five layers of IoT system architecture is as follows [4]:

- **Perception Layer:** It is responsible for gathering data from the surroundings wielding sensors and actuators. It is done by end-node devices, such as sensors, microcontrollers, RFID, etc.

- **Network Layer:** This layer act as a bridge to transfer incoming digitized data from perception layer to the middleware layer. In short it is responsible for the transmission of the data. The transmission can be done using RFID, ZigBee, 6LowPAN, WiFi, 3G/4G, BLE etc. This layer can be associated to the physical and network layer of the TCP/IP model.

- **Middleware Layer:** This layer after identifying the names and address provides service to its requestors. It is with the help of this layer which IoT is capable of connecting with heterogeneous things. Middleware is about ubiquitous computing, services management, connection of heterogeneous things.

- **Application Layer:** The application layer is associated with delivering services to the customers. It incorporates protocols like CoAP, AMQP, DDS, MQTT, HTTP. These protocols are very used in IoT systems.

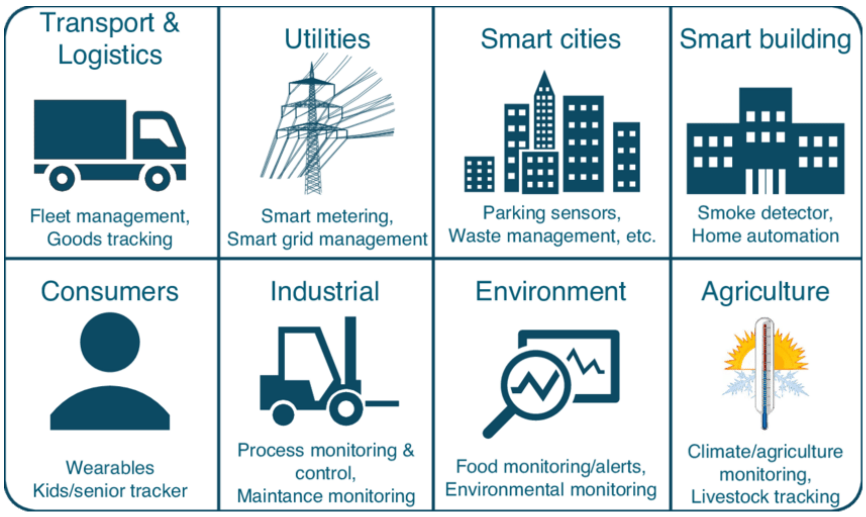
- **Business layer:** It incorporates certain graphs, flowcharts, and models based on the data acquired from the application layer. It is usually what end users really see in an IoT system. It is possible to see de data in a neat way and also to analyze and take action based on the data.

4.1.2. IoT Applications

IoT applications normally is based on gather data from devices status and sensors. The data is accessible for users, allowing the user to have an insight of the all devices and its respective data. Using the gathered information, the user can take action if needed or wanted. Some analytics can be done to the data, it enables the system to take action and process raw data. The processed data makes the system more efficient because the user does not need to separate useful from useless information, saves storage, and reduces the network throughput. The Figure 2 has some examples of IoT applications.

IoT enables an end user monitor and control devices using an appropriated device, such as a laptop or smartphone. The user can know in real time information about temperature, movement, water level or any information about an environment that the user wants to know. A smart fridge is a good example, the user can see what is available on the fridge, change temperature and predicts when will need to buy more food just using a smartphone. A whole house can be controlled and monitored using IoT, such as watch cameras, control garage door, control windows and so on.

Figure 2. IoT Applications

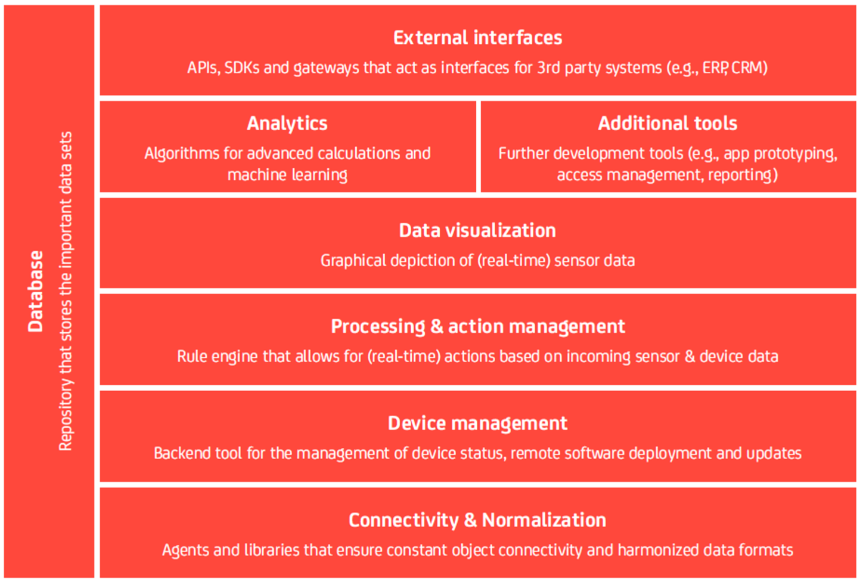

Transport & Logistics
Fleet management, Goods tracking
Utilities
Smart metering, Smart grid management
Smart cities
Parking sensors, Waste management, etc.
Smart building
Smoke detector, Home automation
Consumers
Wearables Kids/senior tracker
Industrial
Process monitoring & control, Maintance monitoring
Environment
Food monitoring/alerts, Environmental monitoring
Agriculture
Climate/agriculture monitoring, Livestock tracking

Source: www.researchgate.net/figure/Massive-IoT-applications-enabled-by-LPWANs_fig1_309617380 [Accessed 27 November 2018].

4.2. IoT Platforms

To be considered an IoT platform, a system must have the eight blocks that are shown in Figure 3 [6].

Figure 3. IoT Platform Blocks


External interfaces
APIs, SDKs and gateways that act as interfaces for 3rd party systems (e.g., ERP, CRM)
Analytics
Algorithms for advanced calculations and machine learning
Additional tools
Further development tools (e.g., app prototyping, access management, reporting)
Data visualization
Graphical depiction of (real-time) sensor data
Processing & action management
Rule engine that allows for (real-time) actions based on incoming sensor & device data
Device management
Backend tool for the management of device status, remote software deployment and updates
Connectivity & Normalization
Agents and libraries that ensure constant object connectivity and harmonized data formats
Database
Repository that stores the important data sets

Source: [6]

Next, there is a brief explanation to each block:

- **Connectivity & Normalization:** This block enables the devices(things) to communicate to each other. It represents all the layers of IoT mentioned before, but business layer.
- **Device Management:** It is a tool for the management of device status, remote software deployment and updates. When the system has just a few devices connected, it's not a big problem to access each device to see its status or to update the device, but in a scenario where hundreds or thousands of devices are connected, it's not practical. So a device management system is needed to efficiently manage the devices.
- **Processing & Action Management:** All the data received need to be processed and relayed to the destination concerned in that data. Actions can be done based on the data. Events can be created to provide better information to the user, or alarms can be triggered when something is out of place.
- **Data Visualization:** The data visualization is about graphs, charts, 3D models, etc. It's how the data is shown to the user. It has an important aspect, because to a lot of people it's the only part to be seen in a system that uses IoT. To many business people the important thing is pretty self-explanatory graphs and charts when gathering data using IoT.
- **Analytics:** In the current society a big amount of data is created each day, the human being is not able to analyze all the data. That is why machine learning and

AI are growing so much in the last years. The user has a better data visualization experience when it is provided just useful information. With a better insight of the devices and its data, correct actions can be done faster by the user or even the system can perform an action by itself.

- **Additional Tools:** It is the extra features that the IoT platform can provide. It gives value to the IoT platform. Some examples of extra features are reports, mobile app prototype, access management, etc.
- **External Interfaces:** It is the way that the platform can communicate with 3rd part systems. It can be for example Windows AD, CRM. Companies usually have useful systems, or legacy systems running and would like to integrate the IoT platform to those systems. So to have a way to communicate with those system is very important. The communication can be done via APIs, SDKs or gateways, for example.
- **Database:** It is how all the data is stored. It is the only block of an IoT platform that interact with all other blocks.

4.2.1. IoT-Ticket

IoT-Ticket is a proprietary IoT platform developed by Wapice Ltd. The IoT-ticket web interface enables user to monitor and control connected devices, also analyze and visualize the gathered data and generate reports. It is possible using widgets and logics that allows users to interact with their devices [7].

This platform supports different kind of communications interfaces, allowing different types of devices to be connected to the system. Such interfaces include RestAPI, OPC UA and MQTT [7].

Some extra specifications can be found in the Table 1.

User management	Device management	Report	Integration	Security	Analytics	Support for visualization	DB
Yes	Yes	Reports can be created in the platform and sent when it's asked or triggered	Rest APIs	Link Encryption (SSL) and basic authentication	Web-based Analytics package and IoT-Ticket R interface	Yes	ODBC, MySQL, MSSQL, and Oracle

Table 1. Extra Information IoT-Ticket

It was asked to the company(Wapice) if the software has any kind of limit, such as a limit of users, devices or messages per second. The company says that the limit is the hardware. The platform can be hosted by Wapice or by the user (own server, cloud etc.).

The RestAPI can be used to register devices, get devices information, write data and read data. To use the RestAPI it is needed an authentication. The authentication account is the same used on the web interface and the user need to have permission to use the API. All the requests are based on the enterprise which the user is associated. For example, when a device is being registered, the device will be registered on the same enterprise than the user is part of.

The MQTT has a API that enables data exchanging between IoT-Ticket and the devices. It is possible to control a device, such as give actions, update de firmware, change configurations using just the platform. This interface has a lot of advantages over RestAPI(HTTP) because of device management and the better data exchanging.

OPC-UA will not be used in this work.

4.3. IoT protocols

In this section is explained briefly the IoT protocols used in this work. IoT protocols are application protocols that are commonly used for data exchange in IoT systems.

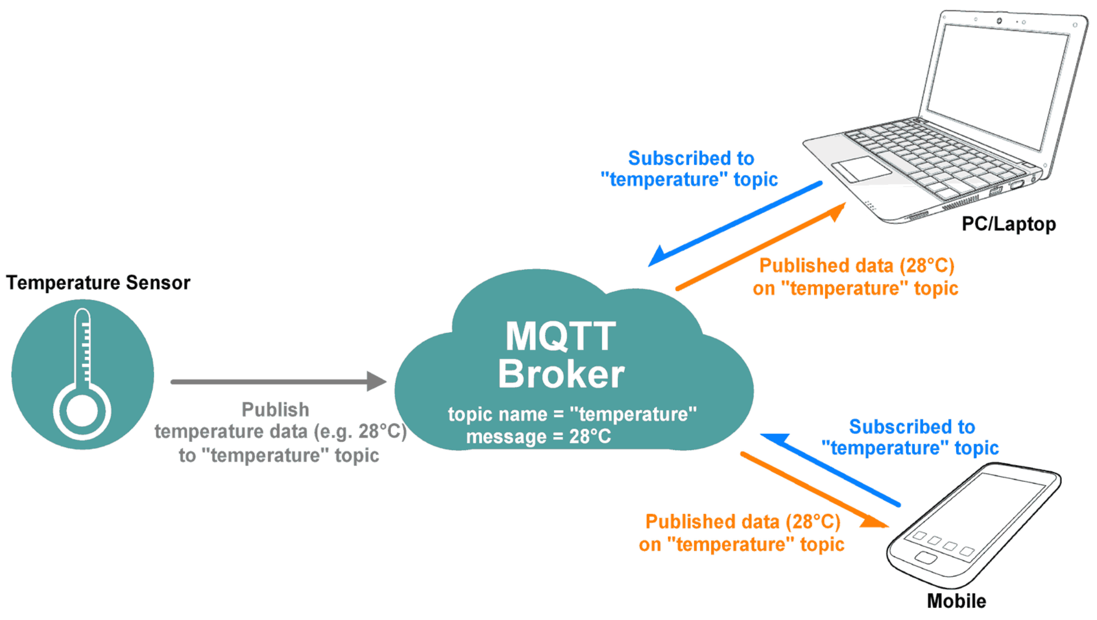
4.3.1. MQTT

Banks and Gupta described MQTT as “MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.” [8]

MQTT uses publish-subscribe pattern to transfer the data between the clients. Thus MQTT decouples the publisher from the subscriber, client connections are always handled by a broker. The broker knows to which client relay the data using topics. Clients that want to receive a specific data must subscribe to the respective topic, so always when a client publish to that topic the subscribed client will receive the update [9]. In short, the broker filter and deliver published data to the clients interested in that data(subscribers).

The Figure 4 shows the idea of MQTT. The laptop and the mobile subscribed to the topic “temperature” and after that they start receiving any data published to that topic.

Figure 4. MQTT Protocol


Subscribed to "temperature" topic
PC/Laptop
Published data (28°C) on "temperature" topic
Temperature Sensor
MQTT Broker
topic name = "temperature"
message = 28°C
Publish temperature data (e.g. 28°C) to "temperature" topic
Subscribed to "temperature" topic
Published data (28°C) on "temperature" topic
Mobile

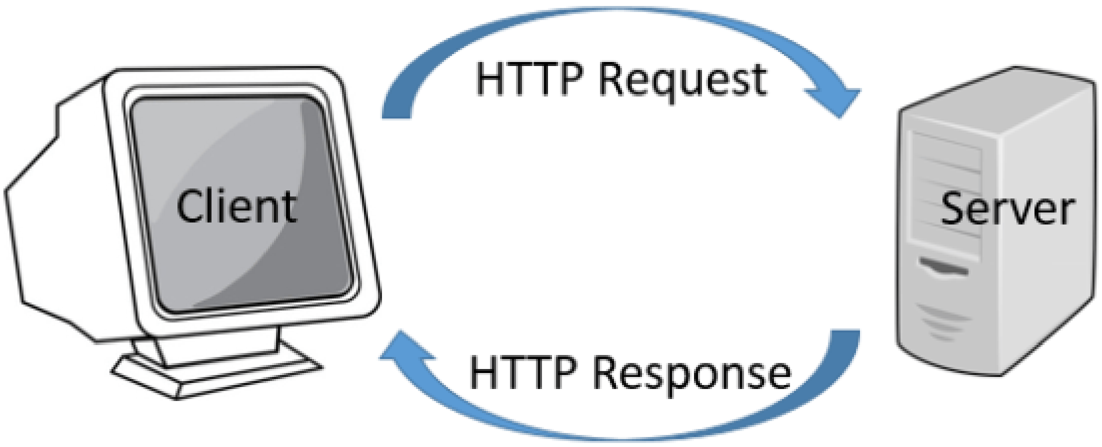
Source: www.electronicwings.com/nodemcu/nodemcu-mqtt-client-with-esplorer-ide [Accessed 27 November 2018].

4.3.2. HTTP

According to the RFC “The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers”. [10]

HTTP follows a classical client-server model, where the client makes a request and wait until it receives a response [11]. An example of HTTP is the Figure 5.

Figure 5. HTTP Protocol



Source: blogs.innovationm.com/http-protocol/
[Accessed 27 November 2018].

4.4. SSL/TLS

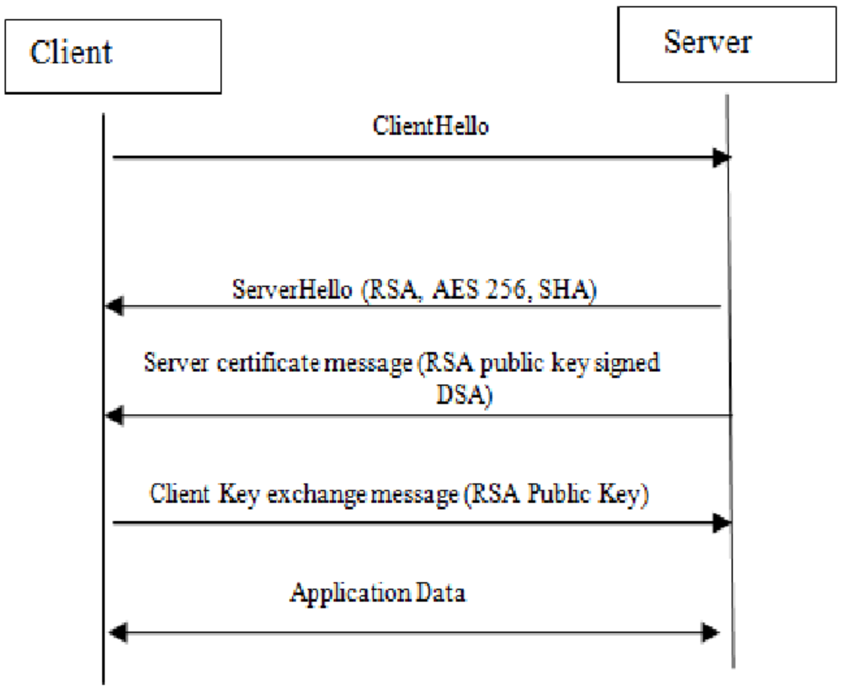
Security is one of the biggest issues when talking about IoT. It is getting harder to have privacy. A lot of companies think IoT as a good asset, but they are afraid of security flaws. The security is responsible for a lot of companies not get a step forward using IoT.

SSL it is a way to improve the security of the system. It allows to keep an internet connection secure and safeguarding any sensitive data that is being sent between two systems. It certifies that the client/server is who it alleged say to be and encrypt the data transferred between the two systems. It avoids criminals from reading or modifying the data being transferred [12].

This secure connection use SSL certificates. The SSL certificates are used to certify that the system is who it alleges to be [12]. The certificate is issued by a certification authority(CA) and it is associated to the system domain. Using the domain and consulting the CA the 3rd party system can know if the connection would be secure(trustful) or not.

Depending on the type of connection, just one side need a certificate or both sides need to have a certificate and guarantee they are who they allege to be. Using SSL, the system does what is called SSL handshake, it is the process the systems prove who they are (exchange of certificates) and also they decide the cryptography used. After the handshake, the systems start exchanging encrypted data. Figure 6 shows an example of a handshake between two systems (client and server) where just the server needs to prove its identity. The last transaction of the figure is application data that it is the data encrypted being exchanged. The application can use MQTT or HTTP to transfer the data for example, but when they are used in a secure connection they are called respectively MQTTS and HTTPS.

Figure 6. HTTP Protocol


Client
Server
ClientHello
ServerHello (RSA, AES 256, SHA)
Server certificate message (RSA public key signed DSA)
Client Key exchange message (RSA Public Key)
Application Data

Source: www.researchgate.net/figure/SSL-handshake-algorithm_fig1_323335493 [Accessed 03 December 2018].

It is important to mention that MQTTS is different of MQTT-S or MQTT-SN. They are a modified MQTT to fit better in scenarios that have low bandwidth and need to have low power consumption.

5. Development

In this section, the step-by-step of the development to accomplish the objectives of this project are shown. This section will be separated in these topics:

- Creating and registering devices on the platform
- Creating dashboards and reports
- Alarms and events
- User management
- Fetching data from external API using REST
- Exchanging data with device using REST
- Exchanging data with device using MQTT
- Exporting data via CSV files
- Extra Information

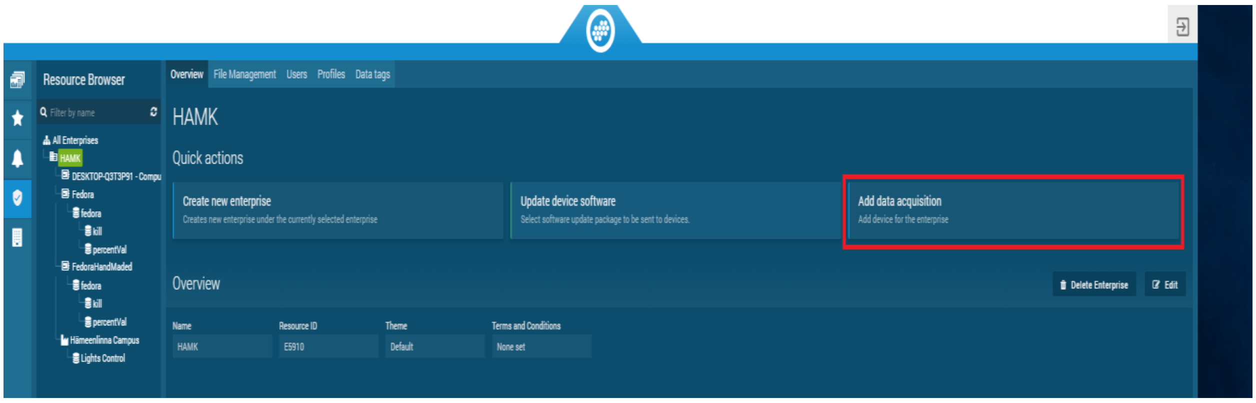
5.1. Creating and Registering Devices on the Platform

The most important thing of IoT is to be able to gather data, so creating and registering devices that will create the data to the platform is very important. There are two ways to create/register a device on IoT-Ticket platform. The device can be created via User Interface(UI) or using RestAPI. It is not possible to create a device using MQTT.

5.1.1. Creating Device by UI

In “Management Mode”, click on the enterprise that is wanted to register/create a device, this will display the “Overview” page of the Enterprise. In the overview of the interface, click on “Add data acquisition”. If the button “Add data acquisition” does not exist, it means that the user is not allowed to do this action. Clicking on “Add data acquisition” is displayed “Add a device” interface. Fill out the options and click on create. When this document was created the only “Communication method” was MQTT, it may be because this method is the only one available on the platform or because the version that is used in this work. Even though the communication method is MQTT the RestAPI can be used to the device.

Figure 7. Creating Device by UI – Step 1



HAMK

Quick actions

Create new enterprise
Creates new enterprise under the currently selected enterprise

Update device software
Select software update package to be sent to devices.

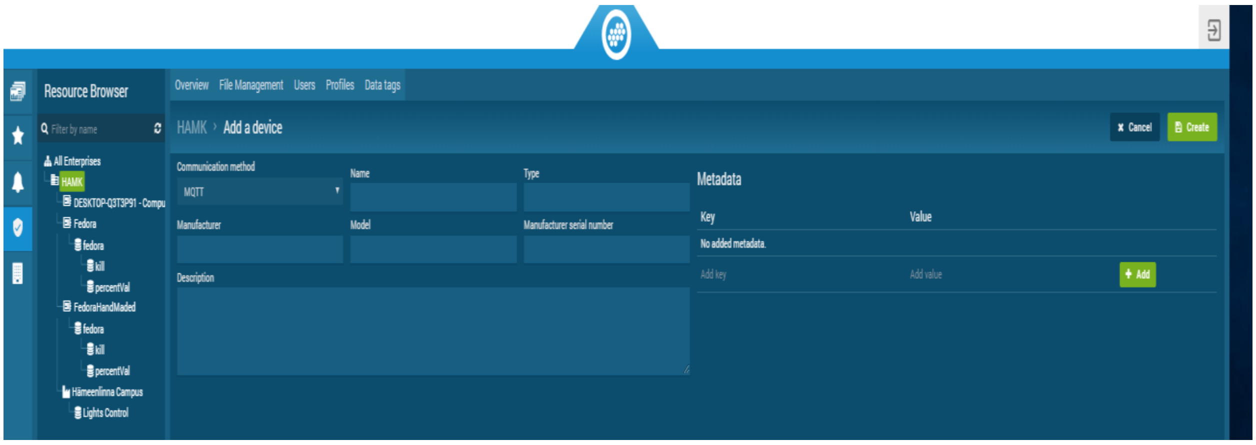
Add data acquisition
Add device for the enterprise

Overview

Name	Resource ID	Theme	Terms and Conditions
HAMK	E5910	Default	None set

Source: Author Screenshot

Figure 8. Creating Device by UI – Step 2



HAMK › Add a device

Communication method: MQTT
Name
Type
Manufacturer
Model
Manufacturer serial number
Description

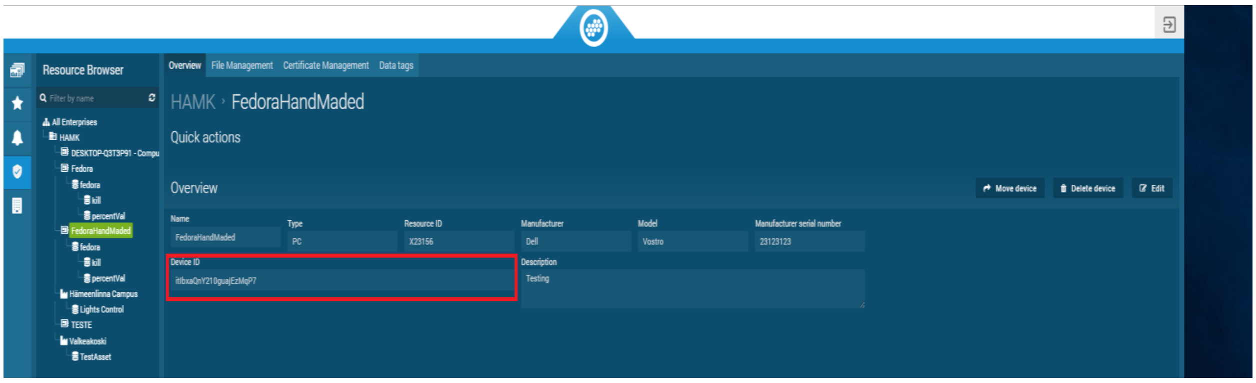
Metadata

Key	Value
No added metadata.	
Add key	Add value

Source: Author Screenshot

When a device is created, a device ID is generated and the real device can start being used. The device ID is how the platform identifies a device when receiving any data from any device. To find the device ID, in “Management Mode” select the device created in the tree. The location is shown in Figure 9.

Figure 9. Device ID



HAMK › FedoraHandMaded

Quick actions

Overview

Name	Type	Resource ID	Manufacturer	Model	Manufacturer serial number
FedoraHandMaded	PC	X23156	Dell	Vostro	23123123

Device ID: itlbxaQnY210guajEzMqP7

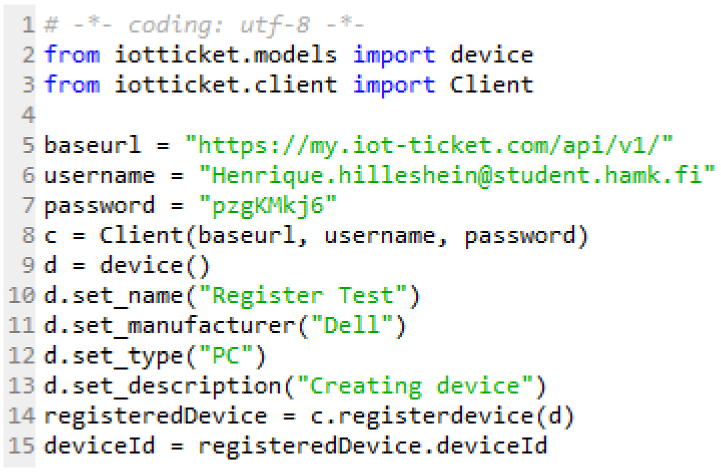
Description: Testing

Source: Author Screenshot

5.1.2. Creating Device by RestAPI

In this work, it was used a python API provided by IoT-Ticket to handle the RestAPI requests. This API helps the developer to create its own application. Figure 10 has an example code of registering a device. When using the IoT-Ticket the Wapice cloud, the URL used is https://my.iot-ticket.com/api/v1/. The username and password are the same that used to access the UI. In this example, the user connects to the IoT-Ticket API, create the device and in the end register the device. The outcome creating the device by RestAPI is the same than by the UI.

Figure 10. Example code to create device



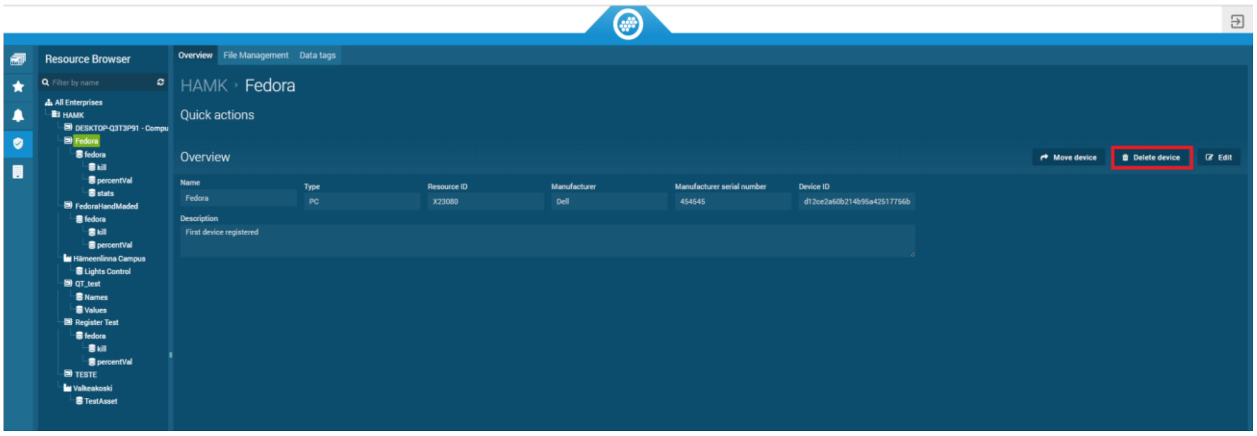
```python
# -*- coding: utf-8 -*-
from iotticket.models import device
from iotticket.client import Client

baseurl = "https://my.iot-ticket.com/api/v1/"
username = "Henrique.hillesheim@student.hamk.fi"
password = "pzgKMkj6"
c = Client(baseurl, username, password)
d = device()
d.set_name("Register Test")
d.set_manufacturer("Dell")
d.set_type("PC")
d.set_description("Creating device")
registeredDevice = c.registerdevice(d)
deviceId = registeredDevice.deviceId
```

Source: Author Code

5.1.3. Deleting Device

In the REST API documentation, there is no information about a request that allows to delete a device. The only way to delete a device is using the UI. In “Management Mode”, click on the device wanted to be deleted and after click on “Delete device” in the device “Overview” page. If the option to delete do not show up, it means that the user has no permission to delete the device. The button can be seen in Figure 11.

Figure 11. Deleting Device



HAMK › Fedora

Overview

Name	Type	Resource ID	Manufacturer	Manufacturer serial number	Device ID
Fedora	PC	X23080	Dell	454545	d12ce2a60b214b95a42517756b

Description: First device registered

Source: Author Screenshot

5.2. Creating dashboards and reports

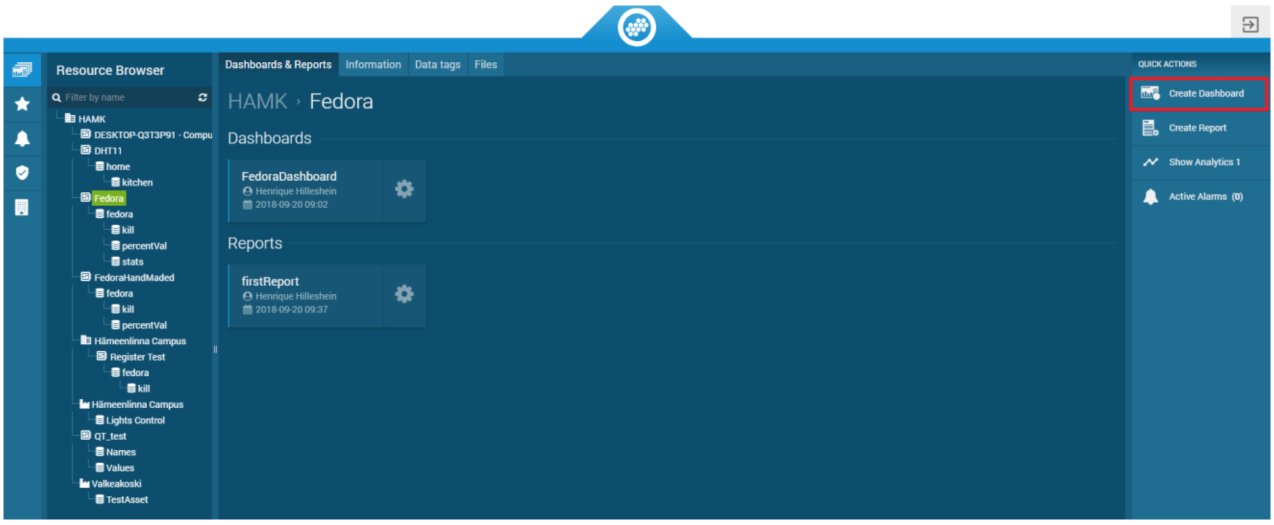
The dashboard and reports is the place where the user can have a good insight into the data coming from the devices.

5.2.1. Dashboard

In the dashboard, the user can show the data using maps, charts, gauges, tables and so on. It is possible to use buttons, checkbox, sliders, etc. to operate the IoT devices or the IoT platform. The user is free to display the data in any way it thinks fit the best and also how to control the IoT system.

To create a new dashboard, it is needed to click on “Create Dashboard” in the “Dashboard Browser”. After that it will be asked the name of the dashboard and if the user wants to use a template.

Figure 12. Creating Dashboard – Step 1



HAMK › Fedora

Dashboards

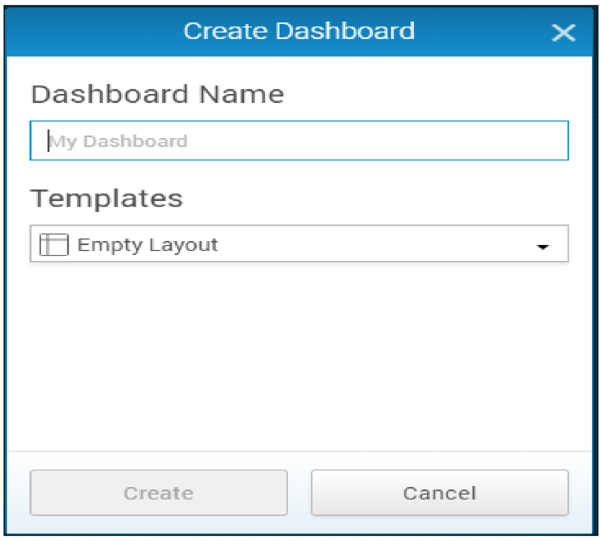
FedoraDashboard
Henrique Hillesheim
2018-09-20 09:02

Reports

firstReport
Henrique Hillesheim
2018-09-20 09:37

Source: Author Screenshot

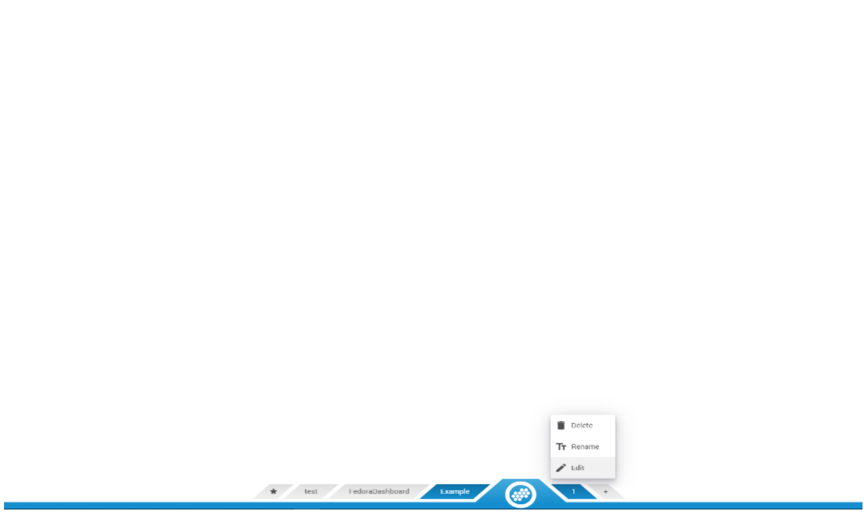
Figure 13. Creating Dashboard – Step 2



Source: Author Screenshot

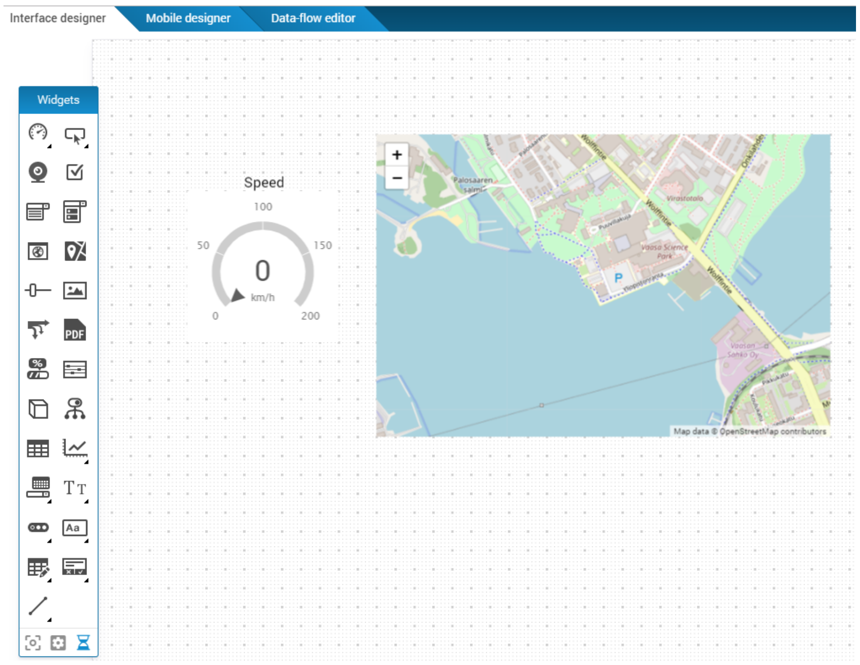
When the empty layout is selected as a template, a blank dashboard is created as can be seen in Figure 14. To edit the dashboard, right-click the number “1” and click on “edit”. It will open the “Interface designer” and show up a box called widgets as it can be seen in Figure 15. Using the widgets is how a dashboard show the data. Each widget has pros and cons, how well the data will fit the widget for the data visualization depends on the type of the data.

Figure 14. Creating Dashboard – Step 3



Source: Author Screenshot

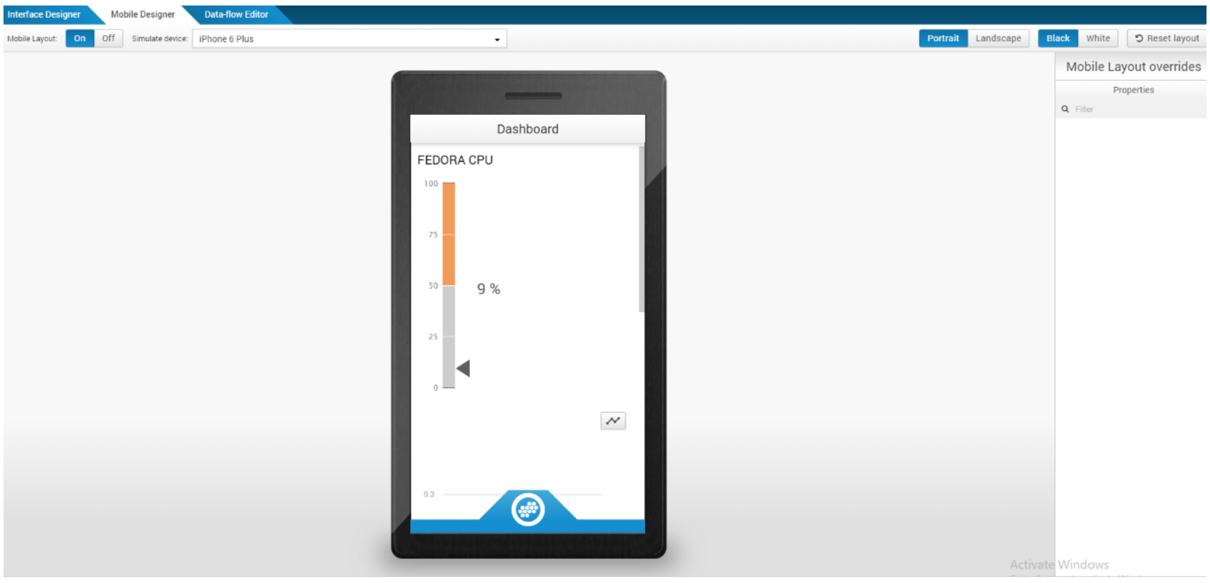
Figure 15. Interface Designer



Source: Author Screenshot

In “Mobile designer”, it can be created dashboards that fit mobile screens. As it can be seen in Figure 16.

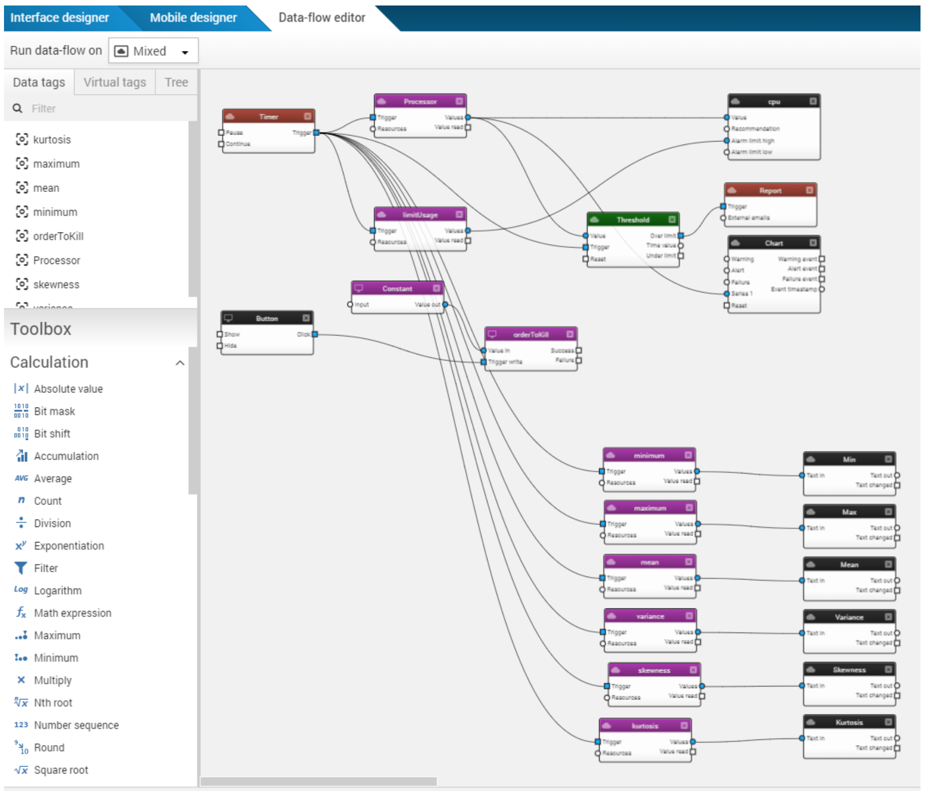
Figure 16. Mobile Designer



Source: Author Screenshot

In “Data-flow editor”, the user can create de logic behind the dashboard. It can be done some calculations, trigger reports, trigger alarms, create schedules and so on. It’s drag and drop programming as can be seen in Figure 17.

Figure 17. Data-flow Editor



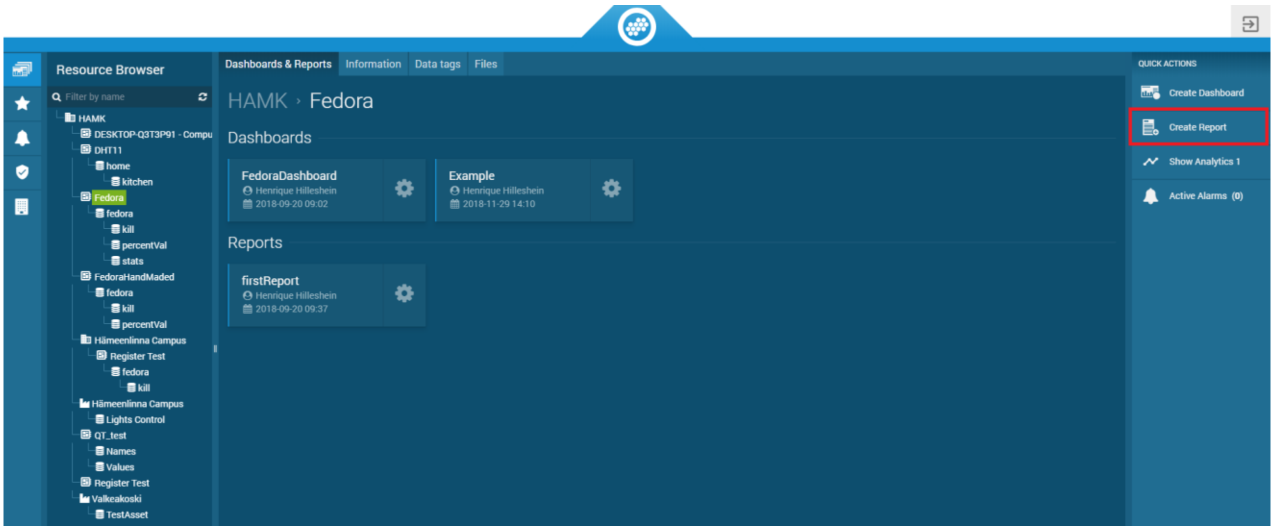
Source: Author Screenshot

5.2.2. Report

Report it is a way to export data from the IoT platform using the same widgets than a dashboard. The report is sent in PDF form to desirable emails. It can be useful to send the report to someone periodically or because of a critical thing.

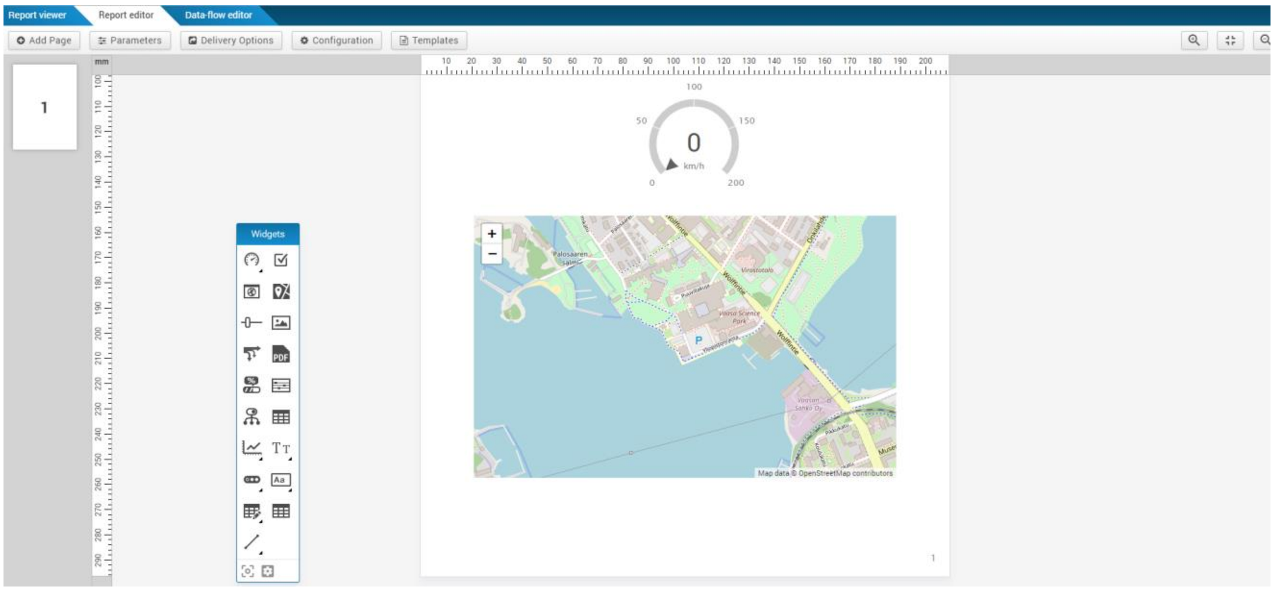
The way the report document is created is similar to the dashboard. In “Dashboard Browser”, click on “Create Report”. It will pop up the box “Create Report” where it asks for a name and a layout. When the report is created, the user can add the widgets and the logic behind just like when creating a dashboard, but rather than creating a dashboard the user will create a PDF that can be sent via email. In Figure 19 is shown the report editor.

Figure 18. Creating Report



Source: Author Screenshot

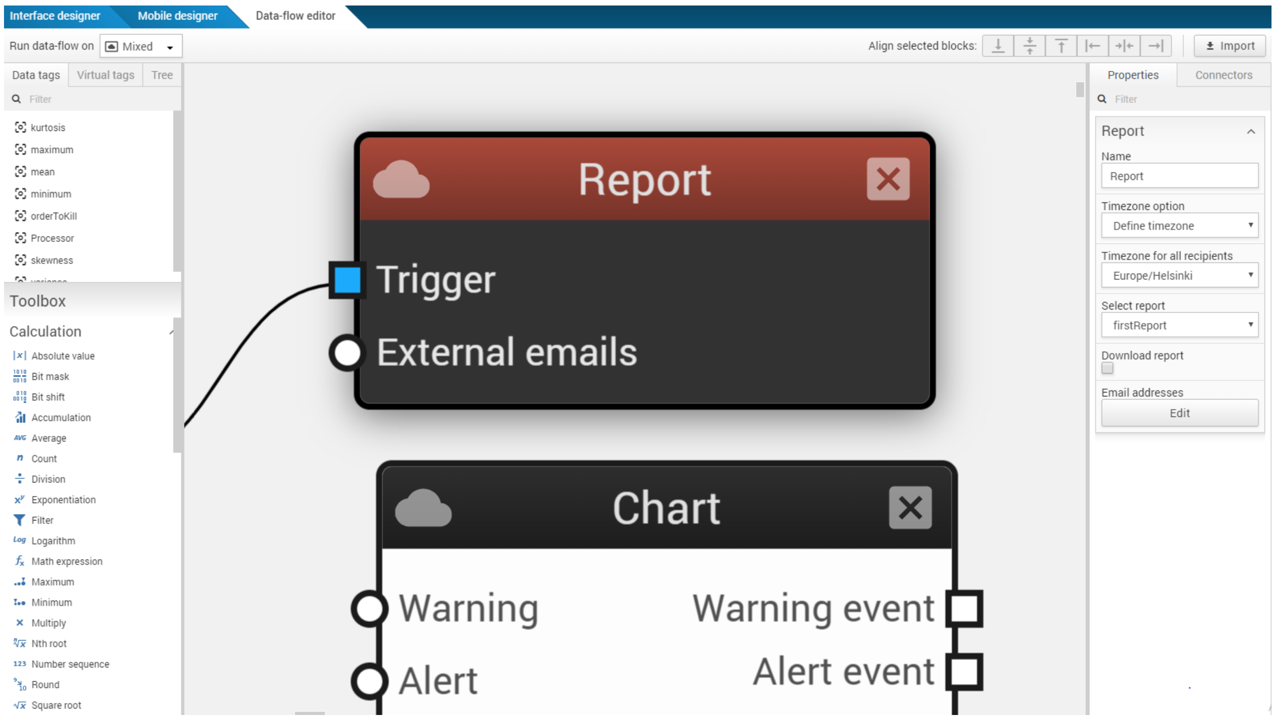
Figure 19. Report Editor



Source: Author Screenshot

To the report be sent it is used the dashboard. In the “Data-flow editor” of the dashboard editor, there is a block called “report”. This block its configured the destiny(email), the report used and the time zone. The block need to be triggered for something to send the report. The trigger can be a button, threshold or a scheduler. The Figure 20 shows the block.

Figure 20. Report Block



Report

Name: Report

Timezone option: Define timezone

Timezone for all recipients: Europe/Helsinki

Select report: firstReport

Download report: ☐

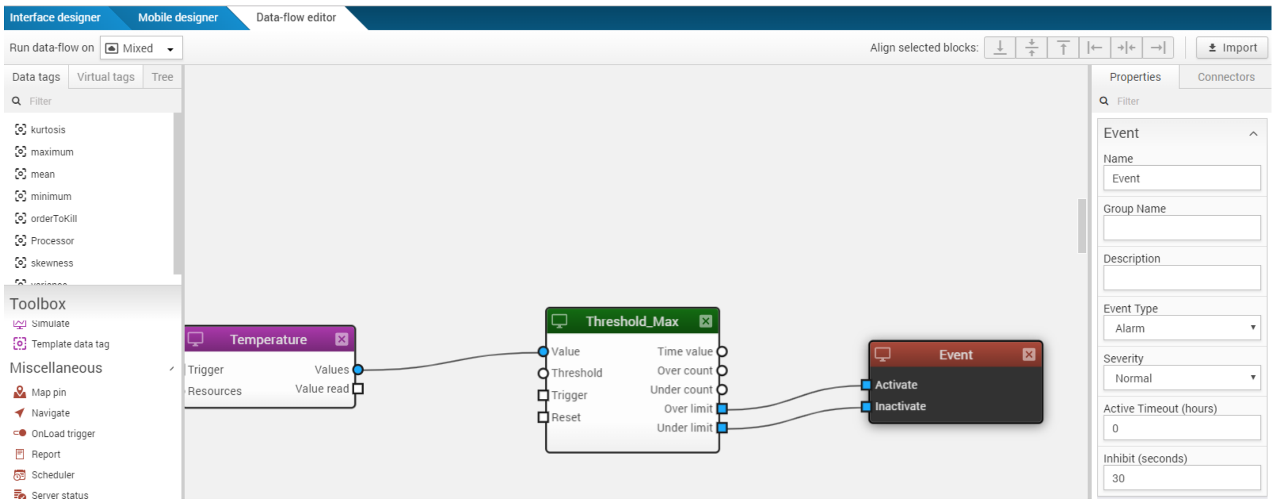
Email addresses: Edit

Source: Author Screenshot

5.3. Alarms and Events

Alarms and events are a feature used to warn the user that something happened, it can be something critical or not critical. If the system has hundreds of devices, it is hard to monitor everything because it would have a lot of dashboards and data that need to be analyzed. In this case, you could use events the could be triggered by a threshold for example. The Figure 21 has an example.

Figure 21. Creating Event



Event

Name: Event

Group Name:

Description:

Event Type: Alarm

Severity: Normal

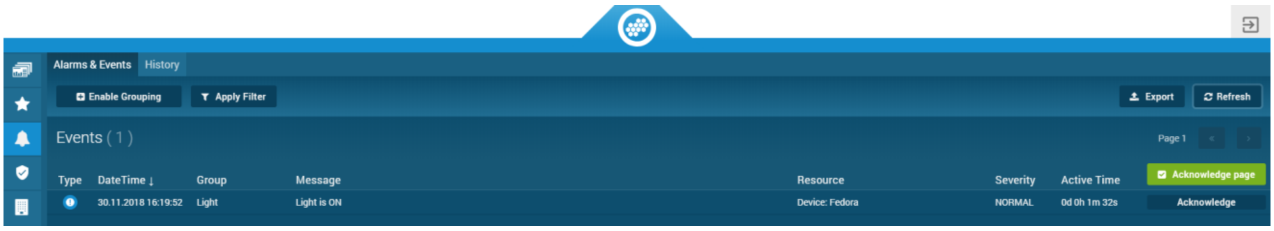
Active Timeout (hours): 0

Inhibit (seconds): 30

Source: Author Screenshot

Once the event triggered, the event shows up at Alarms & Events page as it can be seen in Figure 22. Every single event need to me acknowledged by a user to disappear or inactivated by a trigger. When acknowledging an event, the user need to put a comment. All the acknowledged events go to the history with the information of the event and what was commented when the event was acknowledged. The current events and acknowledged ones can be downloaded as a CSV file.

Figure 22. Event Page



Type	DateTime ↓	Group	Message	Resource	Severity	Active Time	
●	30.11.2018 16:19:52	Light	Light is ON	Device: Fedora	NORMAL	0d 0h 1m 32s	Acknowledge

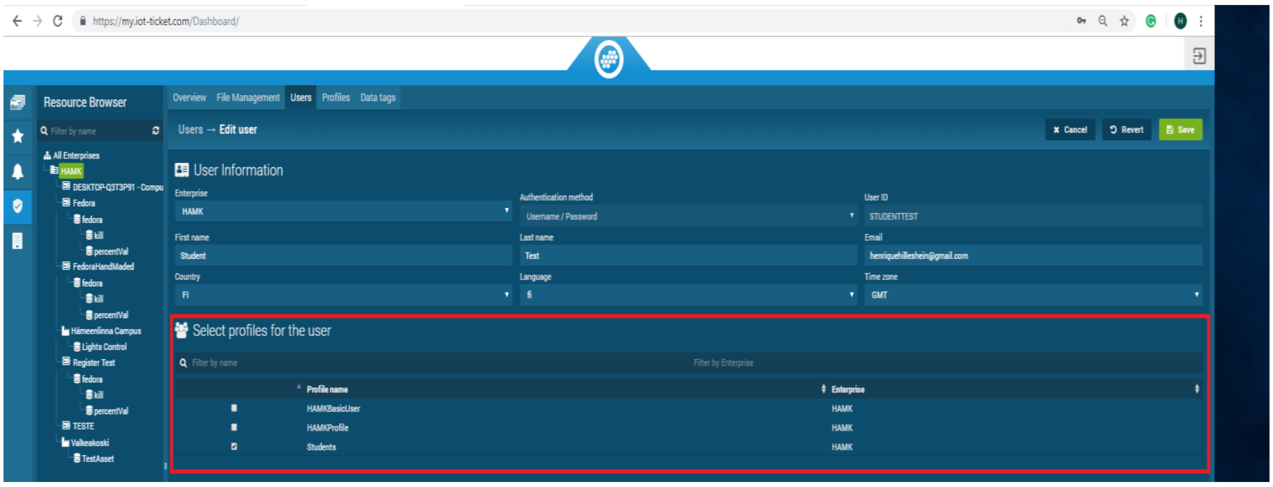
Source: Author Screenshot

5.4. User Management

5.4.1. Creating and Editing a User

User management has a big importance when using a IoT-Platform because there are users that cannot have all de access of the platform or they can have just access to specific things. The configuration of user access is done at “Management mode -> Users”. In this page, you can create or edit a user. The user has a configuration called “Select profiles for the user”, where it’s necessary to select which profiles the user should be part of (at least one profile need to be selected). The profile is like an access control template, users associated with the same profiles have the same permissions. The page used to edit a user can be seen in Figure 23.

Figure 23. Edit User Page



Users → Edit user

User Information

Enterprise	Authentication method	User ID
HAMK	Username / Password	STUDENTTEST

First name	Last name	Email
Student	Test	henriquehilleshein@gmail.com

Country	Language	Time zone
FI	fi	GMT

Select profiles for the user

	Profile name	Enterprise
☐	HAMKBasicUser	HAMK
☐	HAMKProfile	HAMK
✓	Students	HAMK

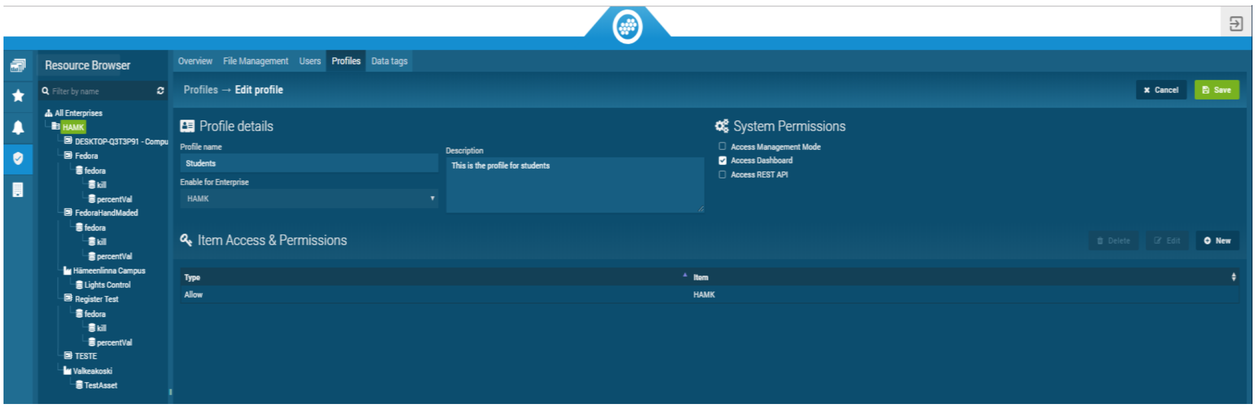
Source: Author Screenshot

When is create a new user, an email is sent to the user to confirm and verify the user.

5.4.2. Profiles

Profile creation and editing is done at “Management mode -> Profiles”. In this page you can set “System Permissions” and “Item Access & Permissions”, as can be seen in Figure 24.

Figure 24. Edit Profile Page



Profiles → Edit profile

Profile details

Profile name: Students

Enable for Enterprise: HAMK

Description: This is the profile for students

System Permissions

☐ Access Management Mode
✓ Access Dashboard
☐ Access REST API

Item Access & Permissions

Type	Item
Allow	HAMK

Source: Author Screenshot

In “System Permissions”, there are three options that need to be configured:

- Access Management Mode: Enable the Management Mode to the user.
- Access Dashboard: Enable the dashboard to the user.
- Access REST API: Enable user to register device and push/pull data from the server using the REST API

The configurations “Access Management Mode” and “Access Dashboard” have extra configuration, these extra configurations can be set for enterprise, device or data tag in “Item Access & Permissions”. So the user can have different permission to each enterprise/device/datatag.

If a user has not permission to Management mode and Dashboard, it will not be able to do login. The user receives the message: “You entered incorrect username or password”.

In “Access & Permissions” There are two access type that can be selected when creating Item Access. The types are Allow and Deny. When using Allow, it means that you are giving access of a device/enterprise/datatag to a user. The Deny type means that among of what was allowed, some access should be denied. Example: Allow all the access to an enterprise, but deny access to some devices.
The following are the available settings.

Dashboard

The dashboard has the configuration “Role”, where you need to specify what is the user role. It is necessary “Access Dashboard” activated to this option show up. These are the available rules:

- Viewer: This user can’t edit the dashboard and can just see the dashboards that are shared to the its profile.
- Operator: Can view and use the buttons of the dashboards shared to its profile.
- Engineer: This user can edit and view the dashboards shared to its profile.
- Manager: This user can see all dashboard and edit them

Management

These are the settings available when the “Access Management Mode” is enabled:

- Create new Enterprise
- Edit Enterprise
- Manage Enterprise Users and profiles
- Manage Sites
- Manage Enterprise templates and files
- Manage devices
- Manage assets and terminal in the site

Datanode Permissions

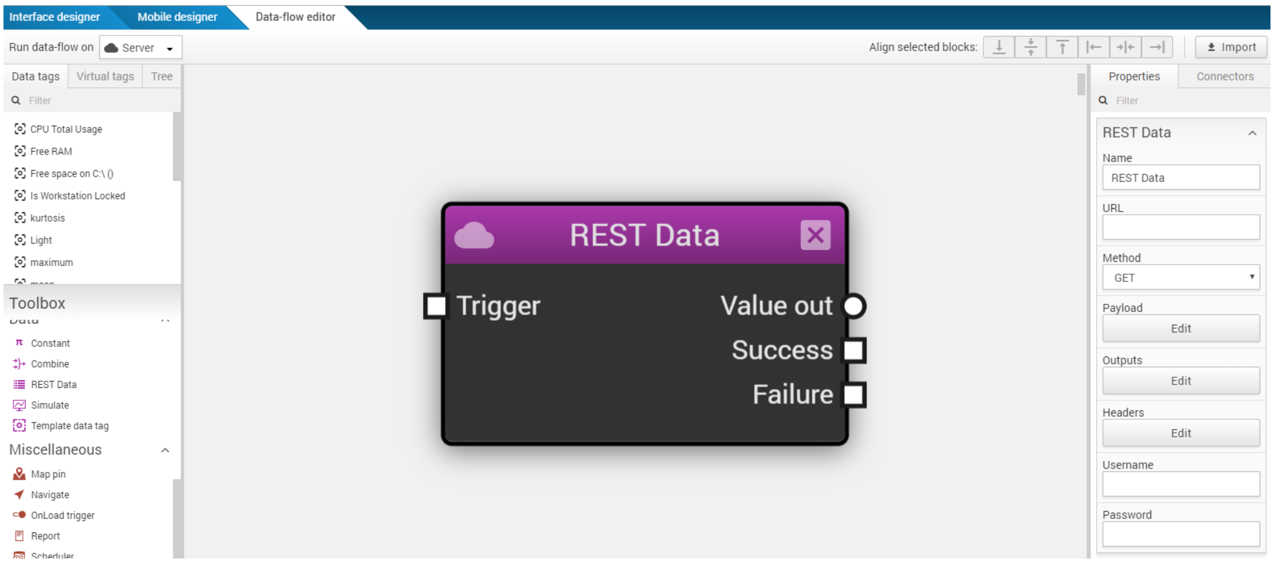
These settings are always available:

- Manage Datanode
- Write Datanode

5.5. Fetching Data from External API Using REST

It is possible to fetch data from an external API using REST. To do this the user need to use the “REST Data” block. In this block, it is set the URL, HTTP method, payload, header, username and password of the HTTP request. How the HTTP request need to be done depends on the 3rd party API. When the block is set with a request that suits the API, it can be set the output of the block if the payload format of the response is JSON or XML. The Figure 25 shows the REST Data block and the Figure 26 shows the configuration of the output.

Figure 25. REST Data



REST Data

Name: REST Data

URL:

Method: GET

Payload: Edit

Outputs: Edit

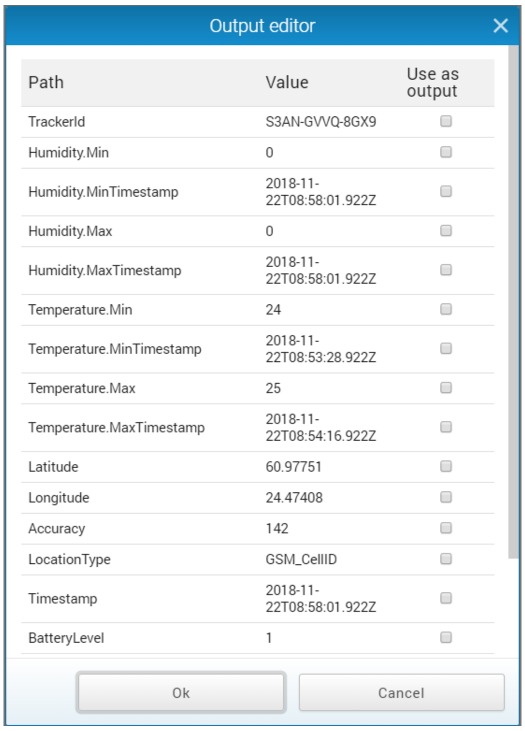
Headers: Edit

Username:

Password:

Source: Author Screenshot

Figure 26. REST Data Block Output



Path	Value	Use as output
TrackerId	S3AN-GVVQ-8GX9	☐
Humidity.Min	0	☐
Humidity.MinTimestamp	2018-11-22T08:58:01.922Z	☐
Humidity.Max	0	☐
Humidity.MaxTimestamp	2018-11-22T08:58:01.922Z	☐
Temperature.Min	24	☐
Temperature.MinTimestamp	2018-11-22T08:53:28.922Z	☐
Temperature.Max	25	☐
Temperature.MaxTimestamp	2018-11-22T08:54:16.922Z	☐
Latitude	60.97751	☐
Longitude	24.47408	☐
Accuracy	142	☐
LocationType	GSM_CellID	☐
Timestamp	2018-11-22T08:58:01.922Z	☐
BatteryLevel	1	☐

Source: Author Screenshot

When the payload or the header needs to be changed, the user can create more inputs for the REST Data block, so the request can use variables that are out of the REST block. To do this the user needs to use the special character “#” when editing the payload or the header. For example, “#password” (without quotes) would create the input “password”, the value of “#password” will be the value of the input created.

To test this REST block, the Yepzon and Trusted API were used.

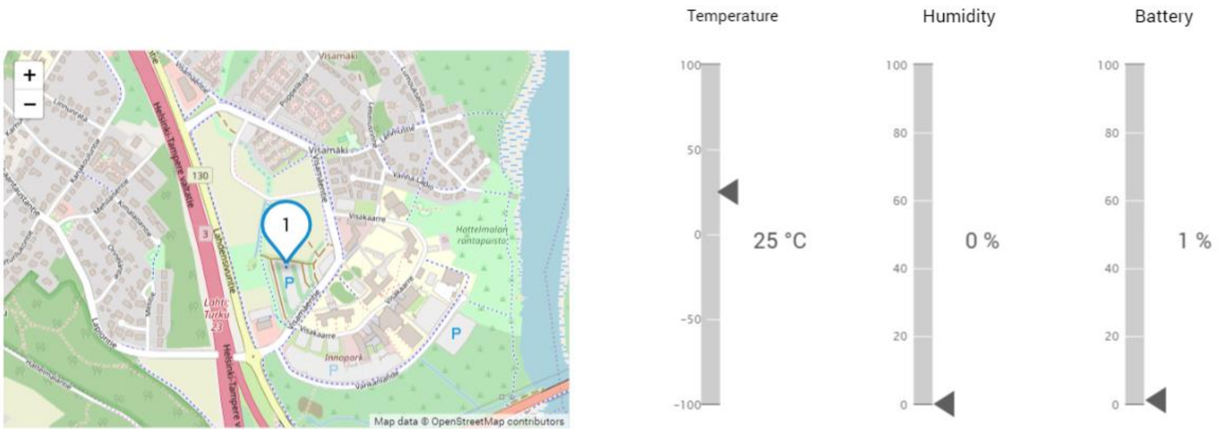
5.5.1. Yepzon

We used the Yepzon asset (Figure 27) to do this test, this device has temperature, humidity and accelerometer sensors and also positioning method as GPS and GSM triangulation. Yepzon has its own IoT platform and the Yepzon asset send the data directly to Yepzon cloud. To be able to use the data externally Yepzon has an API the enables fetch the data using HTTP REST. According to Yepzon API, doing a request using HTTP method GET and URL https://coco.yepzon.com/resource/coco/SERIAL/data/latest, the response is the latest sensor data and location in JSON format. “SERIAL” needs to be replaced by the Yepzon asset serial. Figure 28 and Figure 29 show the dashboard and logic used to test IoT-Ticket with Yepzon API.

Figure 27. Yepzon Asset

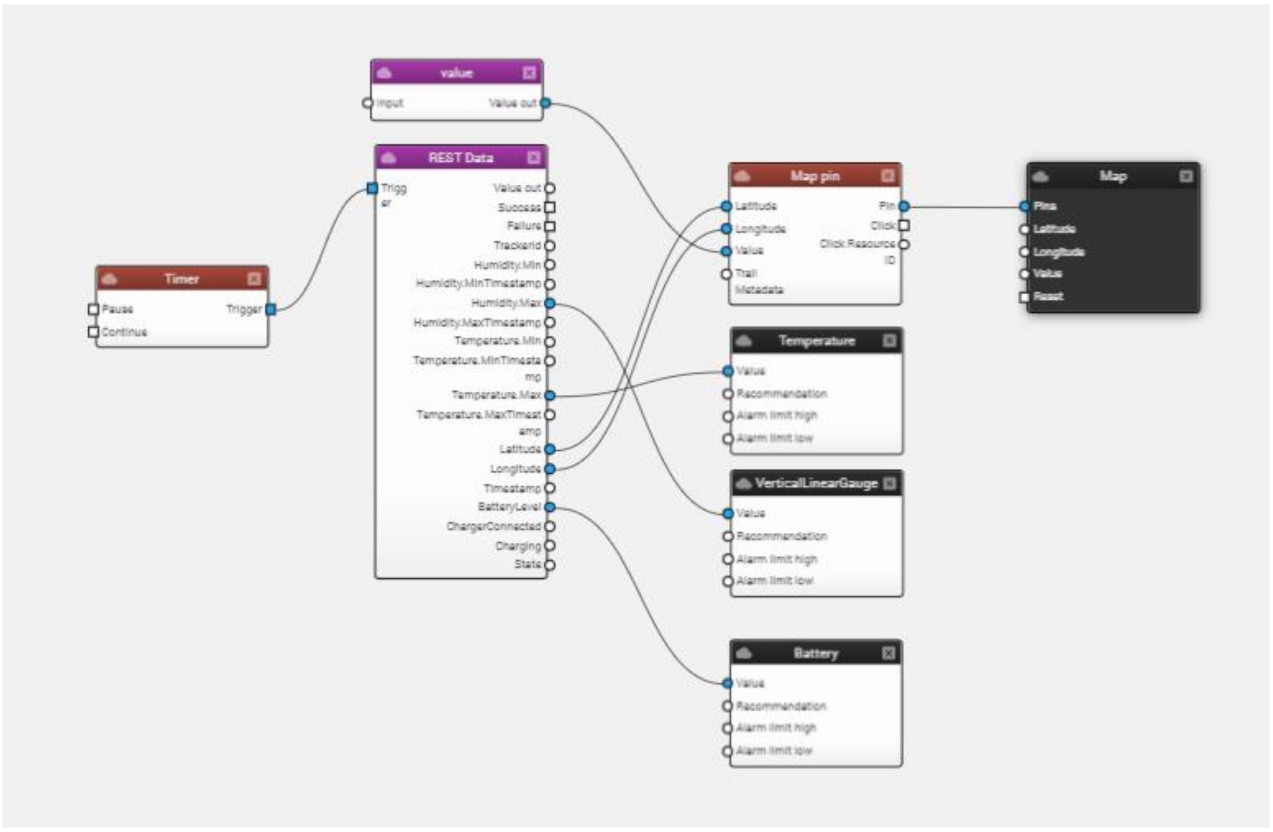
Source: yepzon.com/us/product/asset/
[Accessed 01 December 2018].

Figure 28. Yepzon Dashboard Example



Source: Author Screenshot

Figure 29. Yepzon Data-flow Example



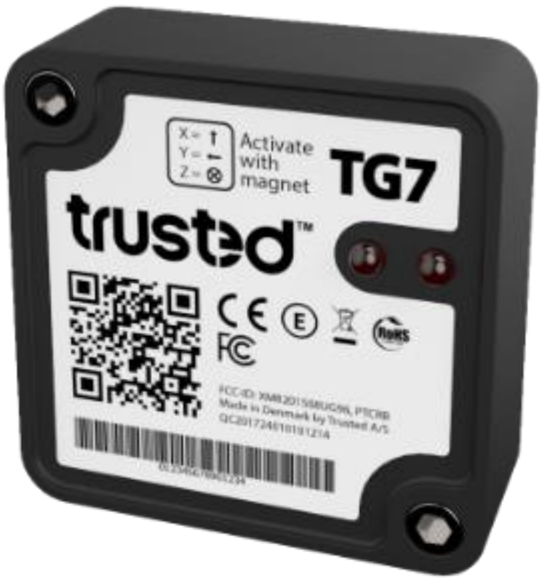
Source: Author Screenshot

The result of the test shows that there is no issue in the communication between IoT-Ticket and Yepzon.

5.5.2. Trusted

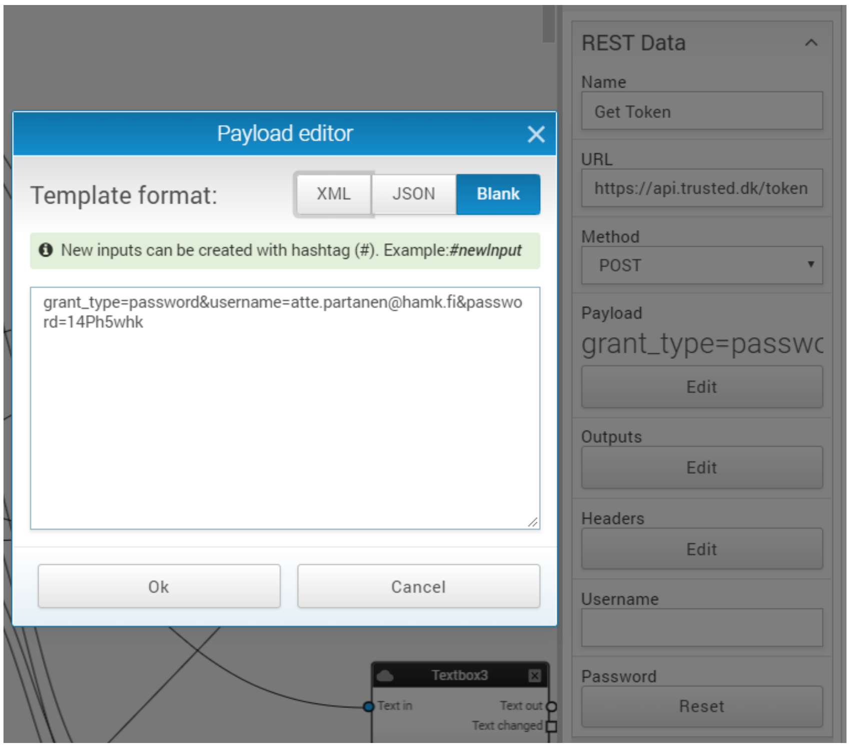
The device used is the TG7, it can be seen in Figure 30. Trusted has the sensors of Position, Motion, Tilt, Temperature, Light, Utilization,3-axis Shock. As Yepzon, the Trusted device sends the data directly to the Trusted cloud and also has a HTTP REST API to communicate with 3rd party applications. To request data from Trusted API the requester needs to have a token. The token is obtained doing a HTTP POST request to “https://api.trusted.dk/token” inserting the username and password that give access to trusted API in the payload. The Figure 31 shows the payload used.

Figure 30. Trusted Device



Source: tecuso.com/produkt/trusted-tg7/
[Accessed 01 December 2018].

Figure 31. Token REST Request


Payload editor

Template format: XML JSON **Blank**

New inputs can be created with hashtag (#). Example:*#newInput*

grant_type=password&username=atte.partanen@hamk.fi&password=14Ph5whk

Ok Cancel

REST Data

Name: Get Token

URL: https://api.trusted.dk/token

Method: POST

Payload: grant_type=passwc

Edit

Outputs: Edit

Headers: Edit

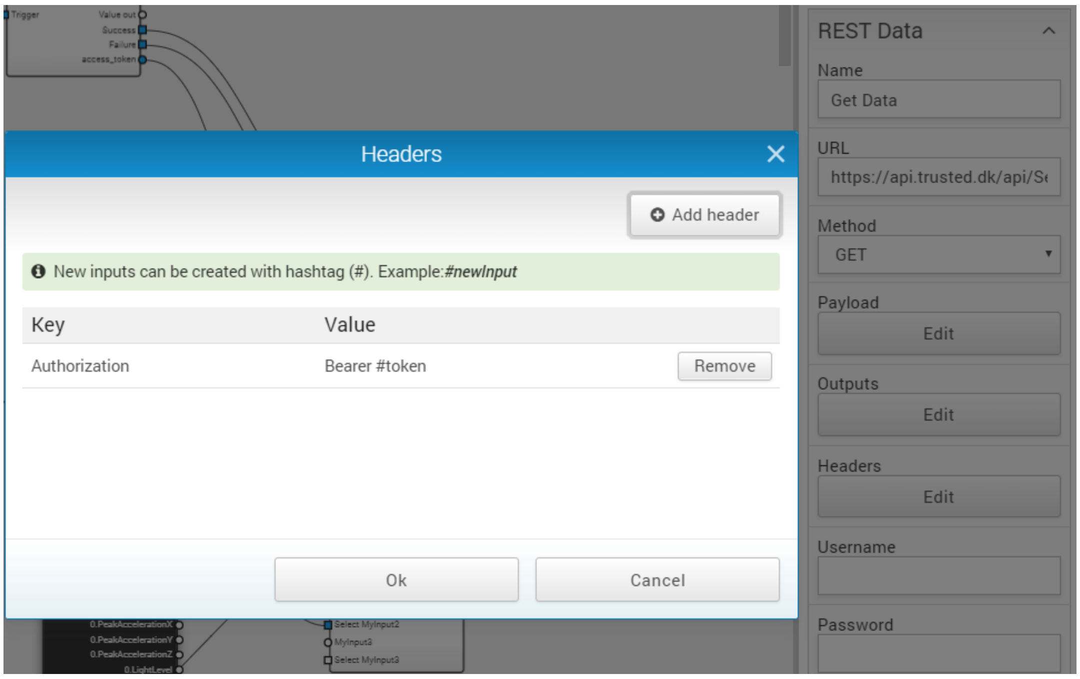
Username:

Password: Reset

Source: Author Screenshot

The response of the request for the token is a JSON that has the token value. Using the token, the user can start requesting for the data to the Trusted API. Just like Yepzon example, the test done with Trusted used always just the last data sent from the Trusted device. To accomplish that it was used the HTTP request GET with URL "https://api.trusted.dk/api/SensorData/Get?SerialNumber=865851039617741&Count=1" to receive temperature data and "https://api.trusted.dk/api/Positions/Get?SerialNumber=865851039617741&Count=1" to receive location data, where the "SerialNumber" is the device serial number and "Count" is the number of samples from the newest to the oldest sample. Every request need to use the token in the header of the request as can be seen in the Figure 32. It is being used the special character "#", so the input called "token" is created. The "token" input is connected to the output of the request of the token.

Figure 32. Trusted Data Request



New inputs can be created with hashtag (#). Example:*#newInput*

Key	Value
Authorization	Bearer #token

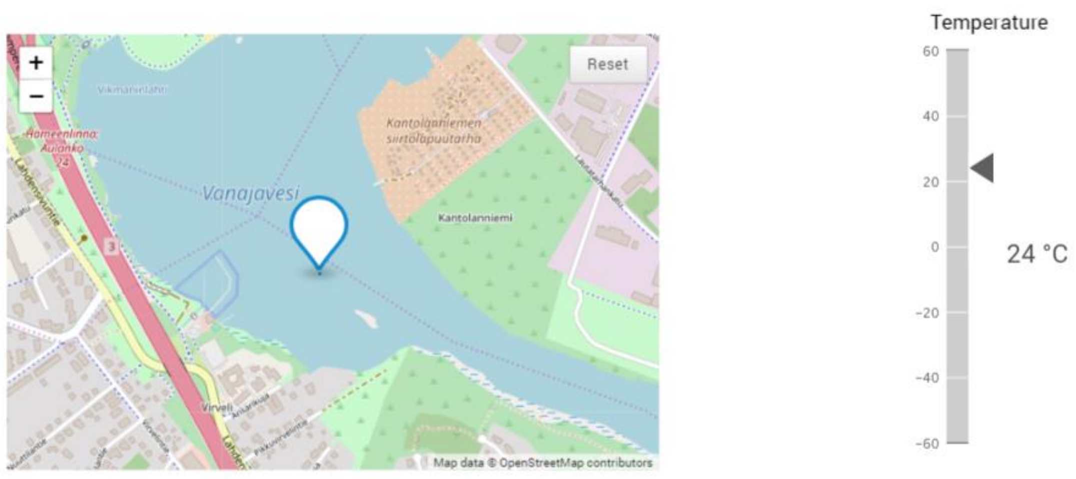
Name: Get Data
URL: https://api.trusted.dk/api/S
Method: GET
Payload
Outputs
Headers
Username
Password

Source: Author Screenshot

The IoT-Ticket worked with Trusted API as can be seen in Figure 33, but there are issues to do the communication to work. It is needed to create a request for token and another to fetch the data, thus it is needed at least 2 blocks to be able to fetch any data. The token expires so it need to be renewed periodically. When using the token as input of the block that fetch the data, the user cannot configure the output of the block because

IoT-Ticket says that there is no output that can be selected. The workaround for that is hardcode the token in the header of the request, but the user always will need to do this when wants to change the output of the request.

Figure 33. Trusted Dashboard Example



Source: Author Screenshot

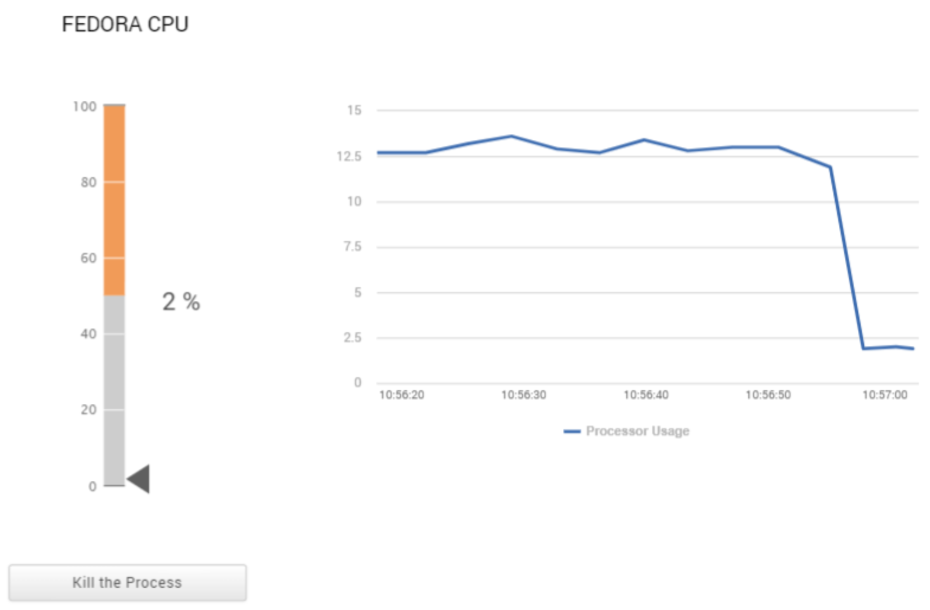
5.6. Exchanging Data with Device Using REST

IoT-Ticket has HTTPS REST API to communicate with devices and provide a ready to use API for a lot of programming languages. In this work, it will be used the Python one. It was used a laptop with Fedora operational system to test this feature. The Fedora sends periodically information about its processor usage and it is show in the dashboard. From platform the user is able to kill the process that is using de processor pressing a button.

Using REST there are two ways to send event/actions to an end device. It can be used polling, where the end device needs to always read a datatag from IoT-Ticket and take an action based on the value of the datatag. The second option is the end device have a HTTP REST API service to receive requests, so it could be used the REST Data block to send action to de device.

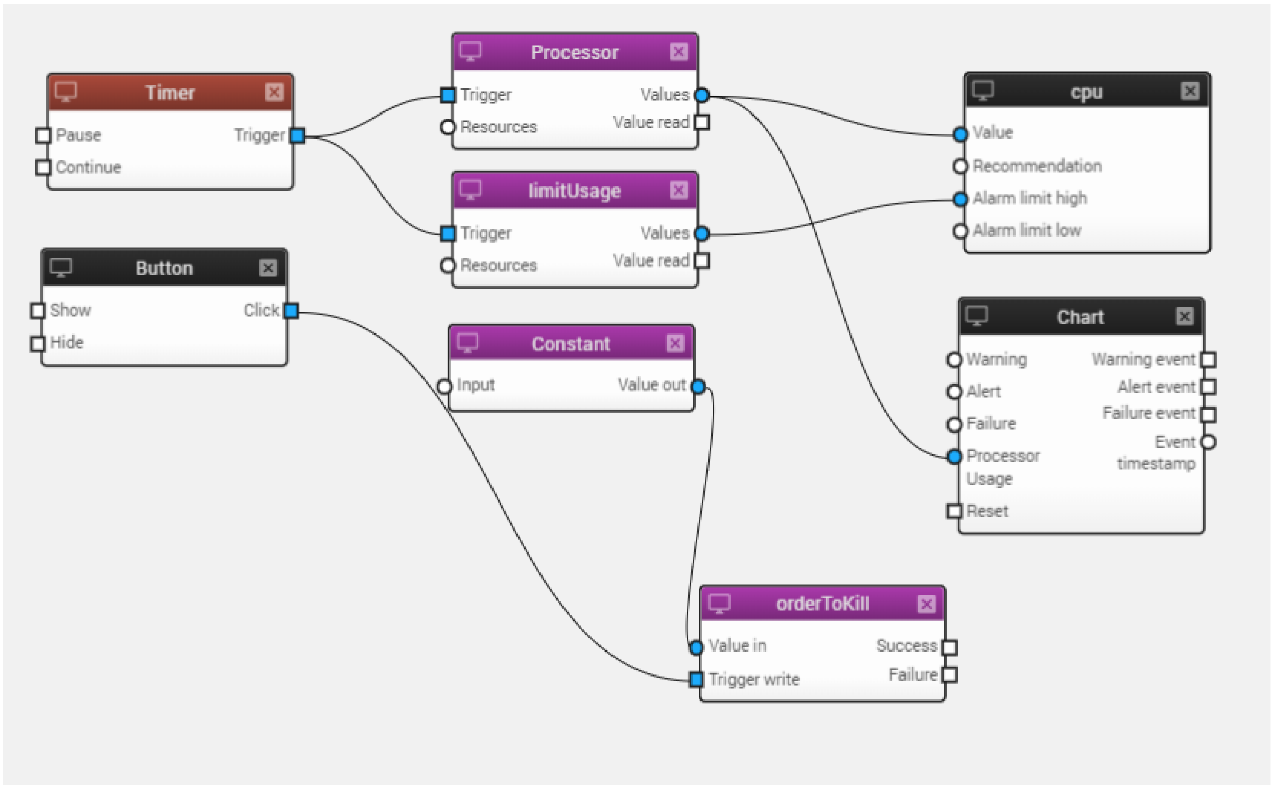
Figure 34 and Figure 35 has the dashboard example used. The system worked very well using the REST API communication and polling to send action.

Figure 34. Fedora Dashboard Example



Source: Author Screenshot

Figure 35. Fedora Dashboard Logic



Source: Author Screenshot

The example code is available on github.com/HenriqueHilleshein/IoT-Ticket_RESTAPI_Example.

5.7. Exchanging data with device using MQTT

MQTT is one of the most important application protocol because fit well with IoT applications. MQTT does not just help to deliver the data, it can be used for device management. IoT-Ticket enables device management using MQTT. The device can manage itself by requesting files from IoT-Ticket. The files may be any type of files, but the file is defined as CONFIGURATION, FIRMWARE, INFORMATION, CERTIFICATE and UNDEFINED. The user can create or update any of these files and the device will receive a notification about this file. Using the notification, the device can request to IoT-Ticket the respective file. The device also can create files itself and upload them to IoT-Ticket platform.

IoT-Ticket uses MQTTS where both sides (broker and device) need to have a certificate. In the device certificate, there is the device ID that IoT-Ticket use to authenticate the device. The device must be created in the IoT-Ticket UI or RestAPI and the certificate needs to be generated. The first certificate the user needs to download from the IoT-Ticket and manually upload to the device. Once the device has the MQTTS communication working with IoT-Ticket the device can request for new certificates when its own certificate is going to expire.

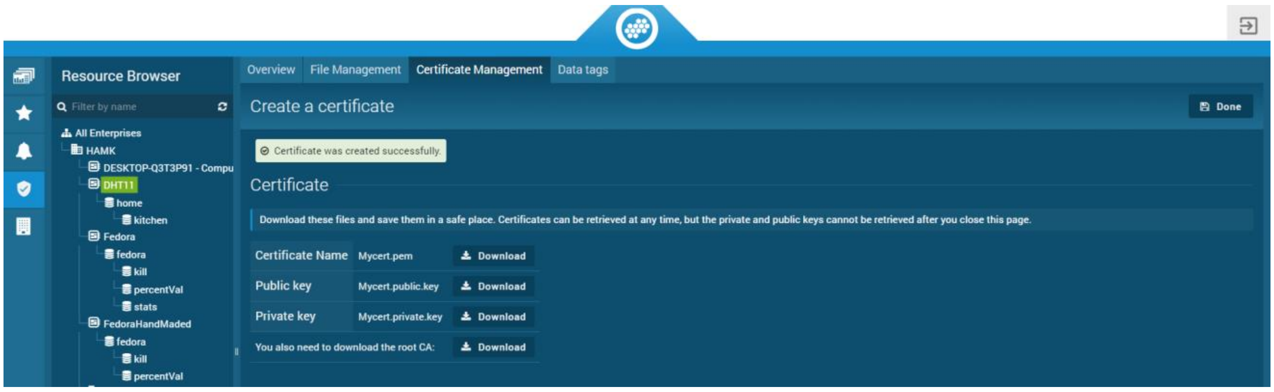
The MQTT of IoT-Ticket has an API that will be followed in this work. All the information given about MQTT of IoT-Ticket were fetched from the documentation, but the documentation was not publicly available while this document was written.

In the next sections, it will be shown how download the certificates and an example code to communicate with IoT-Ticket using MQTTS.

5.7.1. Downloading the Certificates

In “Management Mode”, click on the device that will communicate using MQTT and click on “Certificate Management”. In “Certificate Management” click on “New certificate”, it will ask for a name to the certificate, write the desirable name to the certificate and click on “Create”. It will show the keys and the certificates generate for download as it is shown in Figure 36. Those file will be used to allow the device to communicate to the IoT-Ticket broker.

Figure 36. Download of Certificate



Certificate was created successfully.

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

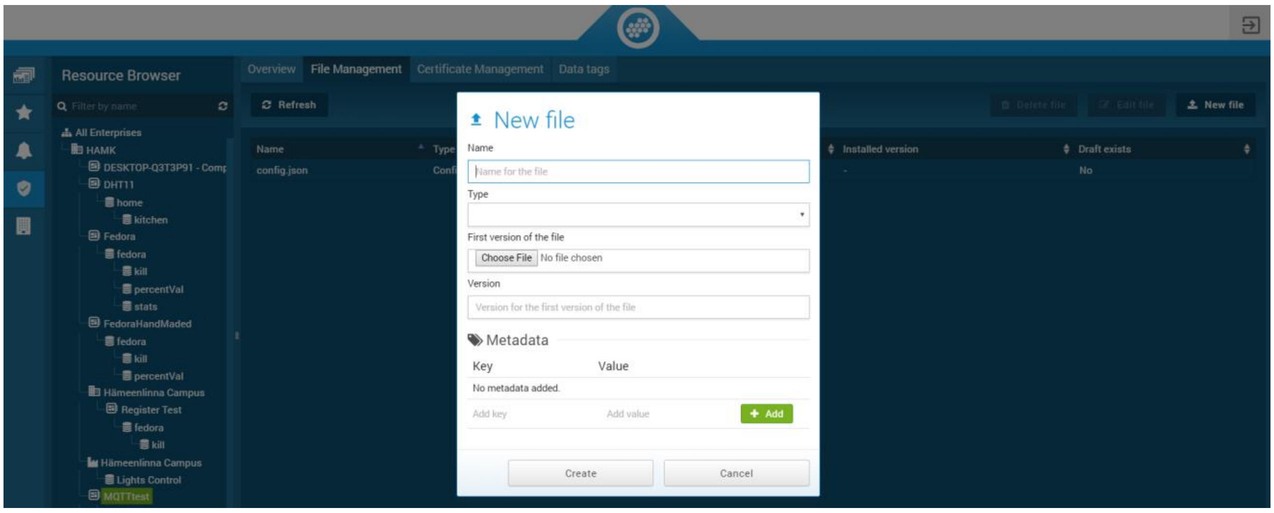
Certificate Name	Mycert.pem	Download
Public key	Mycert.public.key	Download
Private key	Mycert.private.key	Download
You also need to download the root CA:		Download

Source: Author Screenshot

5.7.2. File Management

The file management is the place where the user can upload, update or delete a file from IoT-Ticket using the UI. To do changes in the file management, in “Management Mode” select the device or enterprise desired and click on “File Management”. To create a new file, click on “New File” and a pop up to give information about the file will show up (Figure 37). The user needs to say the name, type and the version of the file and also upload the file. The type of file can be configuration, firmware or other in the version used to create this document. Once the file uploaded to IoT-Ticket, a device can ask for that file.

Figure 37. New File



Name	Type	Installed version	Draft exists
config.json	Confi	-	No

Name: Name for the file
Type:
First version of the file: No file chosen
Version: Version for the first version of the file

Key	Value
No metadata added.	
Add key	Add value

Source: Author Screenshot

A file uploaded to IoT-Ticket platform can have a lot of versions. A right-click to the file show up the options to upload a new version, download a version, activate a version, edit version metadata and delete a version. Every single time that the active version changes, a notification is sent to the devices, so they can download the new version.

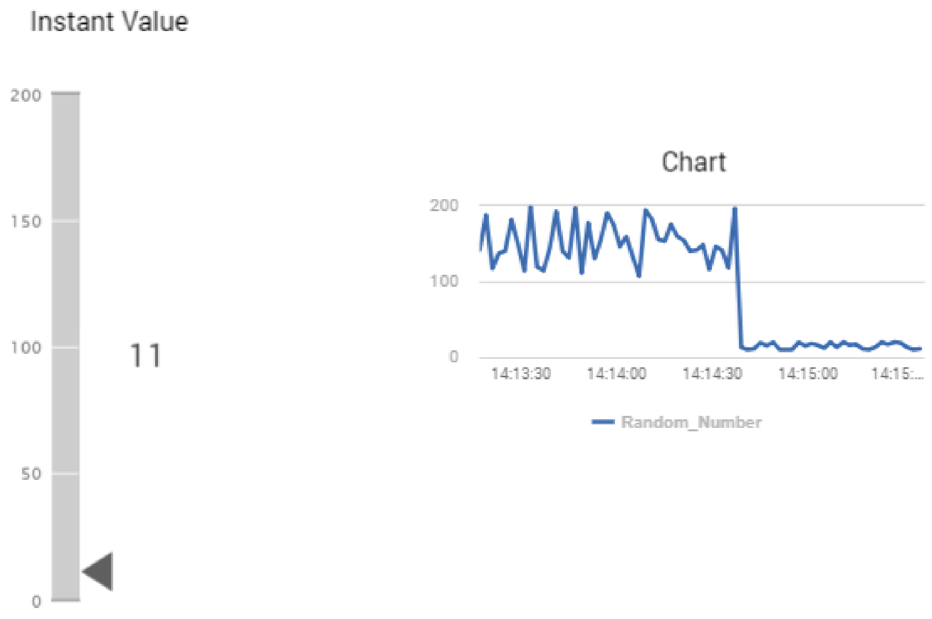
5.7.3. Testing MQTT

The MQTT was tested using JS. It was used the JS MQTT module to handle the SSL handshake and the MQTT protocol. The topics used and the payload of any message were done based on IoT-Ticket documentation. In this example, the end device (JS software) send periodically a random number(data) to IoT-Ticket and the IoT-Ticket can update the device configuration. The configuration is the min and max value that the random number can have. The dashboard of the example is in Figure 38.

In the example, the random number started being between 100 and 200 and after it was update to stay between 10 and 20 as it can be seen in Figure 38. The random number sampling rate was 0.5 numbers per second.

The MQTT using JS as expected. It is possible to receive data and control the devices using the IoT-Ticket platform.

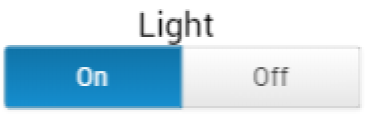
Figure 38. MQTT Example Dashboard

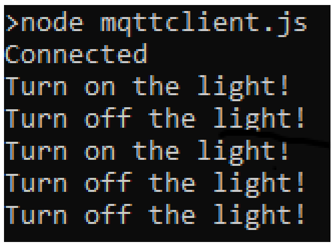


Source: Author Screenshot

It the same dashboard it was inserted a toggle button to simulate the IoT-Ticket sending commands to the device. The toggle and the output of the JS program can be seen in Figure 39.

Figure 39. Send Command to Device via MQTT



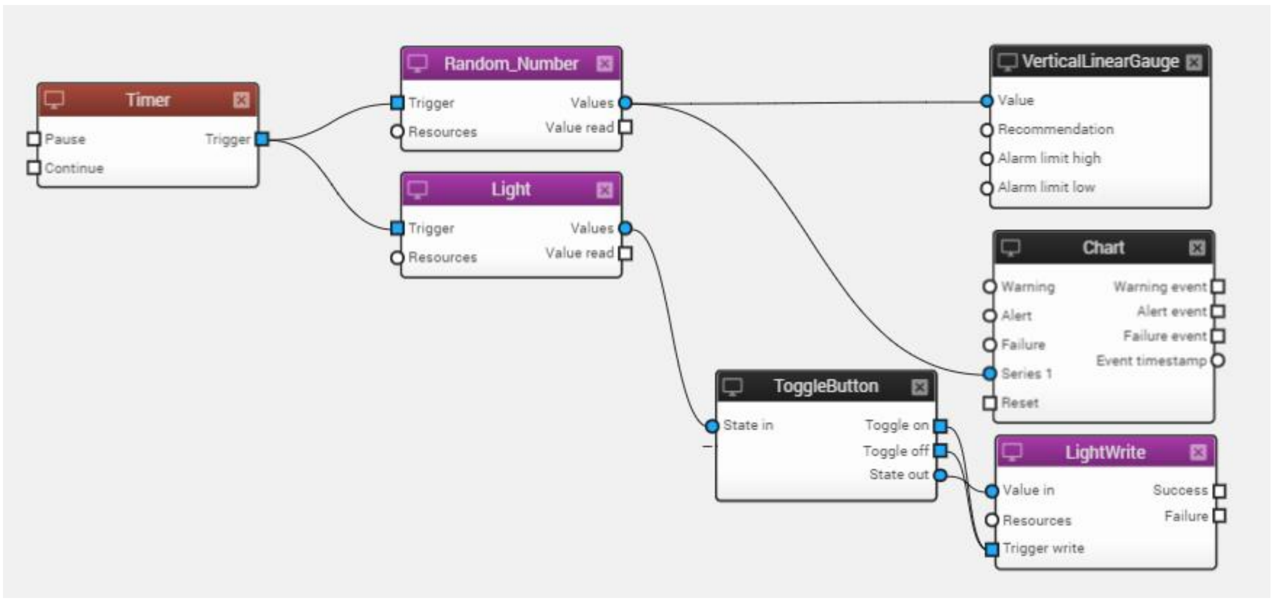


```
>node mqttclient.js
Connected
Turn on the light!
Turn off the light!
Turn on the light!
Turn off the light!
Turn off the light!
```

Source: Author Screenshot

The logic used in the dashboard is shown in Figure 40.

Figure 40. MQTT Dashboard Logic



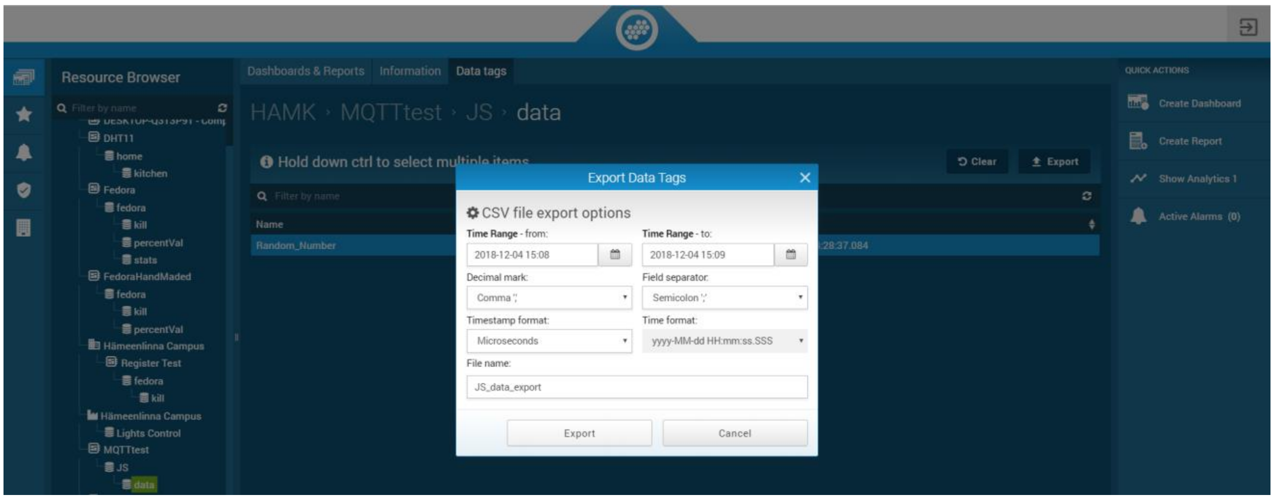
Source: Author Screenshot

The example code is available on:
github.com/HenriqueHilleshein/IoT-Ticket_MQTT_Example

5.8. Exporting Data Using CSV File

It is possible to export data using CSV File but just one data tag can be exported each time. To export the datatag, in “Dashboard Browser” go to the datatags tab where is the datatag that should be exported. Select the desired datatag and click on “Export”, it will open a pop up to do the download of the file as it is shown in Figure 38.

Figure 38. Export CSV



Export Data Tags

CSV file export options

Time Range - from: 2018-12-04 15:08

Time Range - to: 2018-12-04 15:09

Decimal mark: Comma ','

Field separator: Semicolon ';'

Timestamp format: Microseconds

Time format: yyyy-MM-dd HH:mm:ss.SSS

File name: JS_data_export

Export | Cancel

Source: Author Screenshot

5.9. Extra Information

It is not possible to do analytics using IoT-Ticket, some simple analytics can be done using the calculations blocks or JavaScript block. Those blocks are available in dashboard editor.

The IoT-Ticket platform can be hosted by the costumer on-premises or in its own cloud. The costumer can allocate the resources that it wants for the platform. The costumer hosting the platform make it easier to integrate the platform with 3rd party systems. So it would be easier to integrate the user management of the IoT-Ticket platform to a Windows AD for example.

6. Conclusion

The main objective of this work was to try the IoT-Ticket platform to see the viability to use the platform for educational purposes, such as teach students IoT. IoT-Ticket is an easy platform to manage and it has a good design, the platform does not have a good documentation and an online community though.

The platform has a good user management, a lot of widgets that can be used in the dashboard and reports, it has also a good alarm and event system. The device management using MQTT and files is a really good feature, mainly when the IoT system has a big amount of devices. The MQTT showed itself as a good way to do two-way communication of platform with devices.

The biggest challenges during this work were a bug when using the REST block, a feature of REST block that IoT-Ticked did not support, test MQTT using an embed system board as end device and documentation. The bug and the feature of the REST block were implemented and released in the new version of IoT-Ticket, enabling the work advance in regard to import data from 3rd party systems. The embed board used was a Wemos D1 Mini Pro. The problem with using the embed board was not to terminate the handshake, the handshake would be always aborted in the middle of the transaction and server logs were no accessible for debugging.

It was not found a good way to export data from IoT-Ticket, such as a CSV file. The user can just export one data tag each time.

IoT-Ticket looks like be a promising platform, but currently, it has some bugs and lack of the documentation. It would be wise to keep a close eye in the platform for some time to see how the platform will progress before buying it. Depending on the advance of IoT-Ticket it could be a really good platform and according to Wapice, they can do some custom features to their customer if it is needed.

Some tests and validations of IoT-Ticket platform are pending, it is needed to test the platform hosted on-premises and using an own cloud. When hosting the platform, the developer could try to integrate the platform with 3rd party systems, such as Windows AD. A functional implementation using embed systems (without OS) need to be made. A better way to export data from the platform need to be found or implemented, it would be an option to try to use the IoT-Ticket HTTP RestAPI to export the data.

7. Acknowledgments

I would like to thank everybody who helped me during the project and also HAMK and IFSC for giving me the opportunity to take part in this exchange program.

I want to thank especially Kristiina Ritari-Venäläinen and Joni Kukkamäki for helped me with anything I needed to settle in Hämeenlinna.

8. References

[1] LEE, In; LEE, Kyoochun. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, v. 58, n. 4, p. 431-440, 2015.

[2] GUBBI, Jayavardhana et al. Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, v. 29, n. 7, p. 1645-1660, 2013.

[3] ANJOS, Willian: BITENCOURT, Eduardo. IoT Centralization and Management Applying ThingsBoard Platform, 2018

[4] PINEDA, Nina. M2M VS IoT: Know the Difference. Available at: <www.peerbits.com/blog/difference-between-m2m-and-iot.html>. [Accessed 27 November 2018].

[5] KAUR, J.; KAUR, K. Internet of things: A review on technologies, architecture, challenges, applications, future trends. International Journal of Computer Network and Information Security(IJCNIS), v. 9, n. 4, p. 57–70, Apr 2017. ISSN 2074-9104.

[6] SCULLY, Padraig. 5 Things To Know About The IoT Platform Ecosystem. Available at: <iot-analytics.com/5-things-know-about-iot-platform>. [Accessed 27 November 2018].

[7] VÄLIMAA, Lauri. MQTT client implementation in IEC 61131-3 compatible programming environment. 2017.

[8] BANKS Andrew, GUPTA Rahul. MQTT Version 3.1.1. Available at: <docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>. [Accessed 28 November 2018].

[9] HiveMQ. MQTT Essentials Part 2: Publish & Subscribe. Available at: <www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe>. [Accessed 28 November 2018].

[10] FIELDING, R., GETTYS, J., Mogul, J., FRYSTYK, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, DOI 10.17487/RFC2068, January 1997, <www.rfc-editor.org/info/rfc2068>.

[11] MDN web docs. HTTP. [Online]. Available at: <developer.mozilla.org/en-US/docs/Web/HTTP>. [Accessed 28 November 2018].

[12] SYMANTEC. [Online]. Available at: <www.websecurity.symantec.com/security-topics/what-is-ssl-tls-https>. [Accessed 03 December 2018].