

IBM Watson – T0 – MKR1000 – Getting started

Commissioning task 0

1. An IoT Platform and a microcontroller

An IoT Platform is a development environment where services for system administrators and for end users are developed. The services can be for example management of the device fleet or getting alerts based on analysed data. In this example we are using the IBM Cloud Watson IoT as an IoT platform. Earlier it was marketed as IBM Bluemix Watson IoT.

<https://developer.ibm.com/iotplatform/>

For collecting sensor data an Arduino MKR1000 board will be used.

<https://www.arduino.cc/en/Main/Products>

By following these instructions you will get an overview on developing with an IoT Platform.

If you aim for working with the IoT Platform it would be best if you do similar tasks yourself already while reading these instruction for the first time.

2. MKR1000 code

The MQTT protocol used in IBM Watson is based on a standard. For the Arduino boards for this protocol there are several class libraries. Open the Arduino IDE, pull down menu Sketch, Include Library, Manage Libraries. Search with a word MQTT.

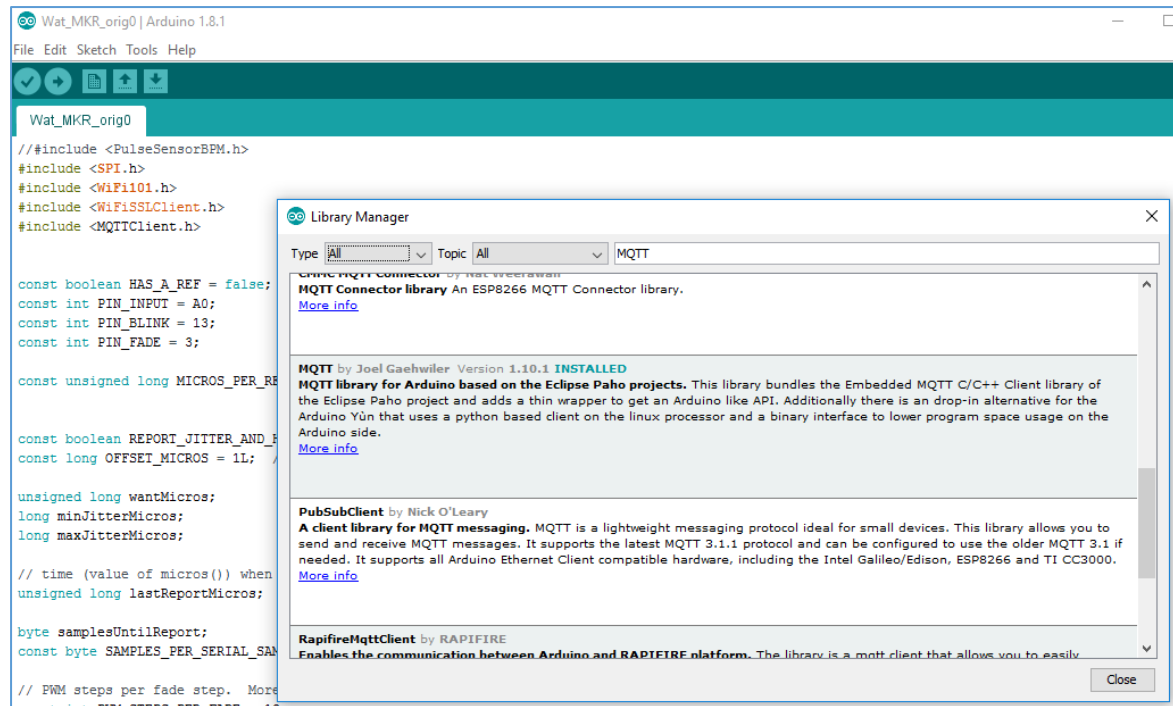


Fig. 2.1 MQTT class library by Joel Gähwiler.

Install the library published by Mr. Joel Gähwiler.

This code is based on examples published at the IBM Bluemix Watson IoT web pages.

After hard work on testing the code got its final form.

```

-----
/*
MKR1000 connecting to IBM Watson IoT Platform

Based on documentation and "recipes" on IBM Bluemix
https://www.ibm.com/cloud-computing/bluemix/watson
Timo Karppinen 19.2.2017

Modified for testing SPI microphone board Digilent PmodMIC3
Please connect
MKR1000 - PmodMIC3
GND - 5 GND
Vcc - 6 Vcc
9 SCK - 4 SCK
10 MISO - 3 MISO
1 - 1 SS

a sound indicator LED
6 - 220 ohm - LED or the onboard LED
Timo Karppinen 13.9.2017

Added calculated sensor data
Loop timing redone
14.11.2017
*/

#include <SPI.h>
#include <WiFi101.h>
#include <WiFiSSLClient.h>
#include <MQTTClient.h> // The Gähwiler mqtt library

// WLAN
char ssid[] = "Moto_Z2_TK"; // your network SSID (name)
char pass[] = "xxxxxxxxxxxxx"; // your network password (use for WPA)

```

```

//char ssid[] = "HAMKWlan"; // your network SSID (name)
//char pass[] = "xxxxxxxxxxxxxxxxxxxx"; // your network password (use for WPA)

// IBM Watson
// Your organization and device needs to be registered in IBM Watson IoT Platform.
// Instruction for registering on page
// https://internetofthings.ibmcloud.com/#

//char *client_id = "d:<your Organization ID>:<your Device Type>:<your Device ID>";
char *client_id = "d:v8yyyyy:A_MKR1000:DF48";
char *user_id = "use-token-auth"; // telling that authentication will be done with token
char *authToken = "xxxxxxxxxxx"; // Your IBM Watson Authentication Token

//char *ibm_hostname = "<your-org-id.messaging.internetofthings.ibmcloud.com>";
char *ibm_hostname = "v8yyyyy.messaging.internetofthings.ibmcloud.com";

// sensors and LEDs
const int ainputPin = A0;
const int soundLEDPin = 6; // must be a pin that supports PWM. 0...8 on MKR1000
// PModMIC3
const int mic3CS = 1; // chip select for MIC3 SPI communication
int sound12bit = 0; // 12 bit sound level value [ 0000 nnnn nnnn nnnn ] nnn.. = two's complement!
int soundByte1 = 0; // 8 bit data from mic board
int soundByte2 = 0; // 8 bit data from mic board
int sound32bit = 0; // in MKR1000 board SAMD21 processor the int is 32 bit two's complement
int sound8bit = 0;

const int numSamples = 100;
int sound8bitA[numSamples];
int sampleIndex = 0;
int soundSum = 0;

int soundLevel = 0; // 8 bit positive number from 0 to 255
int blinkState = 0;

/*use this class if you connect using SSL
 * WiFiSSLClient net;
 */
WiFiClient net;
MQTTClient MQTTC;

unsigned long lastSampleMillis = 0;
unsigned long previousWiFiBeginMillis = 0;
unsigned long lastWatsonMillis = 0;
unsigned long lastPrintMillis = 0;

void setup()
{
  pinMode(mic3CS, OUTPUT);
  digitalWrite(mic3CS, HIGH); // for not communicating with MIC3 at the moment
  Serial.begin(9600);
  delay(2000); // Wait for wifi unit to power up
  WiFi.begin(ssid, pass);
  delay(5000); // Wait for WiFi to connect
  Serial.println("Connected to WLAN");
  printWiFiStatus();

  /*
   client.begin("<Address Watson IOT>", 1883, net);
   Address Watson IOT: <WatsonIOTOrganizationID>.messaging.internetofthings.ibmcloud.com
   Example:
   client.begin("iqwckl.messaging.internetofthings.ibmcloud.com", 1883, net);
  */
  MQTTC.begin(ibm_hostname, 1883, net); // Cut for testing without Watson

  connect();

  SPI.begin();
  // Set up the I/O pins

  pinMode(mic3CS, OUTPUT);
  pinMode(soundLEDPin, OUTPUT);

  // Initializing the sound sample array to zero.
  for(int i = 0; i < numSamples; i++)
  {
    sound8bitA[i] = 0;
  }
}

void loop() {
  MQTTC.loop(); // Cut for testing without Watson

```

```

// opening and closing SPI communication for reading MIC3
if(millis() - lastSampleMillis > 1000/numSamples)
{
    lastSampleMillis = millis();
    SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));
    digitalWrite(mic3CS, LOW);

    soundByte1 = SPI.transfer(0x00);
    soundByte2 = SPI.transfer(0x00);

    digitalWrite(mic3CS, HIGH);
    SPI.endTransaction();

    soundByte1 = soundByte1 << 8;
    sound12bit = soundByte1 | soundByte2;
    sound32bit = sound12bit << 22; // 22 bits to the left to create 32 bit two's complement
    sound8bit = sound32bit / 16777216; // 2 exp24 = 16 777 216 means shifting 24 bits left without
    shifting the sign!

    soundSum = soundSum - sound8bitA[sampleIndex]; // subtract the oldest sample
    sound8bitA[sampleIndex] = sqrt(sound8bit * sound8bit); // reading the | latest sample |
    soundSum = soundSum + sound8bitA[sampleIndex]; //add the latest sample
    analogWrite(soundLEDPin, sound8bitA[sampleIndex]); // blink the LED with intensity = | sound sample |
    sampleIndex = sampleIndex + 1;
    if(sampleIndex >= numSamples)
    {
        sampleIndex = 0;
    }

    soundLevel = soundSum / numSamples;
}

// Print on serial monitor once in 1000 millisecond
if(millis() - lastPrintMillis > 1000)
{
    Serial.print("Sound32bit ");
    Serial.print(sound32bit);
    Serial.print(" Sound8bit ");
    Serial.print(sound8bit);
    Serial.print(" SoundLevel ");
    Serial.println(soundLevel);
    lastPrintMillis = millis();
}

// publish a message every 30 second.
if(millis() - lastWatsonMillis > 30000)
{
    Serial.println("Publishing to Watson...");
    if(!MQTTc.connected()) { // Cut for testing without Watson
        connect(); // Cut for testing without Watson
    } // Cut for testing without Watson
    lastWatsonMillis = millis();
    //Cut for testing without Watson
    MQTTc.publish("iot-2/evt/SoundTwo/fmt/json", "{\"Sound level sensors\": \"Sounds from field, too\", \"SoundMean\": \"\" + String(soundLevel) + \"\", \"SoundStreight\": \"\" + String(sound8bit) + \"\"}");

}

delay(1);

// end of loop
}

void connect()
{
    Serial.print("checking WLAN...");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print("."); // printing a dot every half second
        if ( millis() - previousWiFiBeginMillis > 5000) // reconnecting
        {
            previousWiFiBeginMillis = millis();
            WiFi.begin(ssid, pass);
            delay(5000); // Wait for WiFi to connect
            Serial.println("Connected to WLAN");
            printWiFiStatus();
        }
        delay(500);
    }
}
/*
Example:
MQTTc.connect("d:iqwk1:arduino:oxigenarbpn","use-token-auth","90wT2?a*1WAMVJStb1")

Documentation:
https://console.ng.bluemix.net/docs/services/IoT/iotplatform\_task.html#iotplatform\_task

```

```

*/

Serial.print("\nconnecting Watson with MQTT....");
// Cut for testing without Watson
while (!MQTTc.connect(client_id,user_id,authToken))
{
    Serial.print(".");
    delay(3000);
}
Serial.println("\nconnected!");
}

void messageReceived(String topic, String payload, char * bytes, unsigned int length) {
    Serial.print("incoming: ");
    Serial.print(topic);
    Serial.print(" - ");
    Serial.print(payload);
    Serial.println();
}

void printWiFiStatus() {
    // print the SSID of the network you're attached to:
    Serial.print("SSID: ");
    Serial.println(WiFi.SSID());

    // print your WiFi shield's IP address:
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);

    // print the received signal strength:
    long rssi = WiFi.RSSI();
    Serial.print("signal strength (RSSI):");
    Serial.print(rssi);
    Serial.println(" dBm");
}

```

Code sample 2.1 Application "A MKR1000 sound". Code version 4. The devices will be able to reconnect to WLAN network after getting outside network area and returning again to the network area. The timing of the program loop has been rewritten.

The code example works correctly only if the organization and device has been registered to IBM Cloud Watson IoT Platform service. Please note! If you copy the code from the pdf document it is better to copy it first to text editor (Windows Note) and only after that to the IDE. This procedure removes the un visible control symbols.

In this example an un secure connection is used. To enable this in IBM Cloud Watson IoT you need to select in the IoT Platform "Security", "Connection Security", "TSL Optional" .

Every device trying to connect to an IoT Platform needs to have an individual Client ID and Authentication Token. If the same credentials are used by several devices for trying to connect the IoT Platform the MQTT Broker will reject the device. This is the same procedure on all IoT Platforms and on all MQTT Brokers.

3. IBM Cloud

Please search with an Internet browser with words "IBM Bluemix Catalog". Most probably you will end up to page

<https://console.ng.bluemix.net/catalog/>

In the menu on left please select “Internet of Things”.

You can as well search with words “IBM Cloud Catalog”. The product name has been updated and as starting page you can use either the old “Bluemix” or the new “Cloud” page. Most probably you will now end up to page

<https://cloud.ibm.com/catalog>

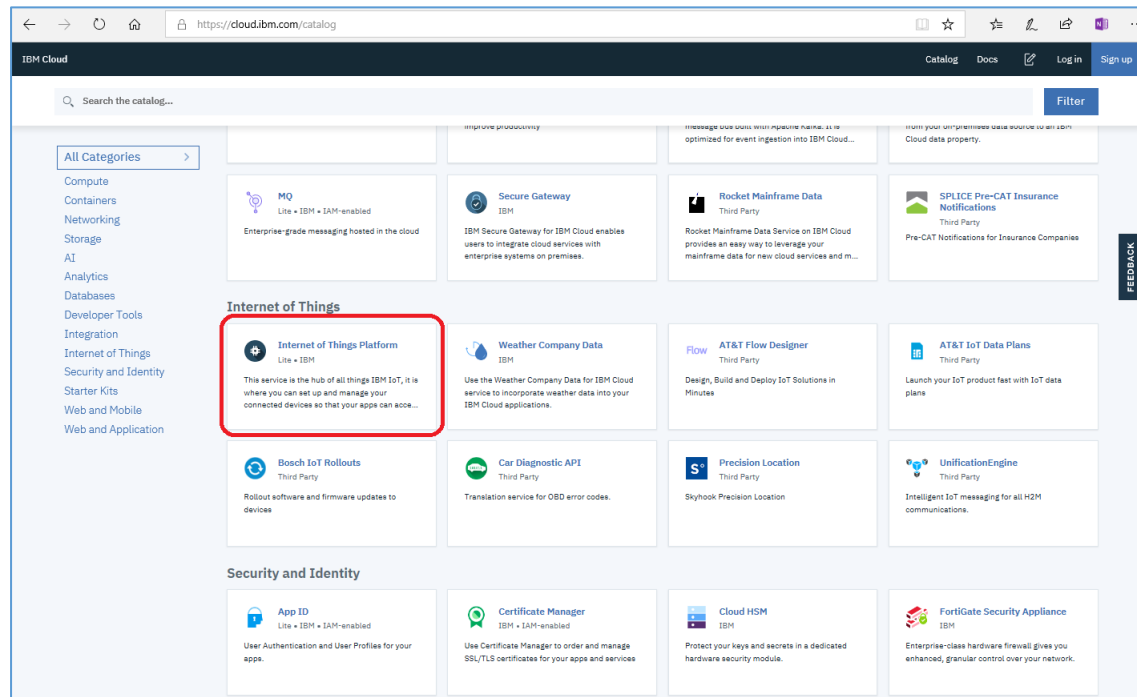


Fig 3.1 IBM Cloud, Internet of Things, Internet of Things Platform / <https://cloud.ibm.com/catalog> 3.6. 2019

Please select the Free as a payment option.

Pricing Plans			Monthly prices shown are for country or region: United States
PLAN	FEATURES	PRICING	
✓ Lite	Includes up to 20 registered devices, and a maximum of 100 MB of each data metric Maximum of 20 registered devices Maximum of 10 application bindings Maximum of 100 MB of each of data exchanged, data analyzed and edge data analyzed	Free	
The Lite service plan for Internet of Things Platform includes up to 20 registered devices, and a maximum of 100 MB each of data exchanged, data analyzed, and edge data analyzed per month.			
Standard	The Standard service plan for Internet of Things Platform includes your free tier of 100 MB each of data exchanged, data analyzed and edge data analyzed per month at no cost. Charge per MB of data exchanged (tiered by usage in MB) Charge per MB of data analyzed Charge per MB of edge data analyzed Multi-Tiered	Expand each section to view details	

Fig 3.2. A Free pricing plan. The offered amount of free resources will be increased every year / 2017

Fig 3.3. A free account will be available for 30 days. / 2017

Create the account and follow all instructions. Please note that now you are creating a Developer ID in the IBM Cloud. This is a different account than the one you already might have in IBM Documentation area. Later – although only within 30 days – you can change this account to an academic account. The academic accounts have more resources and are valid for longer periods. If we will need this your instructor will give you separate document of instructions.

Please note that the academic staff account and the student account needs to be renewed always within the last month of the license period. The pictures on the registering pages are from year 2017 but in 2019 the content was still the same with similar instructions.

Internet of Things Platform

Secure | <https://console.ng.bluemix.net/catalog/services/internet-of-things-platform/>

IBM Bluemix Catalog

The IBM Internet of Things service lets your apps communicate with and consume data collected by your connected devices, sensors, and gateways. Our recipes make it super easy to get devices connected to our Internet of Things cloud. Your apps can then use our real-time and REST APIs to communicate with your devices and consume the data you've set them up to collect.

Connect to:

Loading...

View Docs

AUTHOR IBM
PUBLISHED 02/13/2017
TYPE Service
LOCATION US South

Service name:

Create organization

1 2 3 4

Before you start using Bluemix, you need to set up your environment.

To start, name your first organization. Think of an org as a project or team that shares resources, such as apps, databases, and other services. Orgs exist in geographic regions, so decide where you'd like to put your first one.

United Kingdom HAMK_ICT_IoT Create

NEED SOME SUGGESTIONS? TRY THESE

timo.karppinen timo.karppinen@hamk.fi

LOG OUT | SUPPORT

Fig 3.4 The account is set to use the could services for example in the UK servers.

Summary

1 2 3 4

Good to Go!

You're up and running with your first org and space. Are you ready to get started with Bluemix?

Org name: HAMK_ICT_IoT

Space name: dev

I'm Ready

Fig 3.5 The account information

You might return automatically to page seen in Fig 3.2. Please select the blue button “Create” at the bottom of the page.

This leads you to a new page.

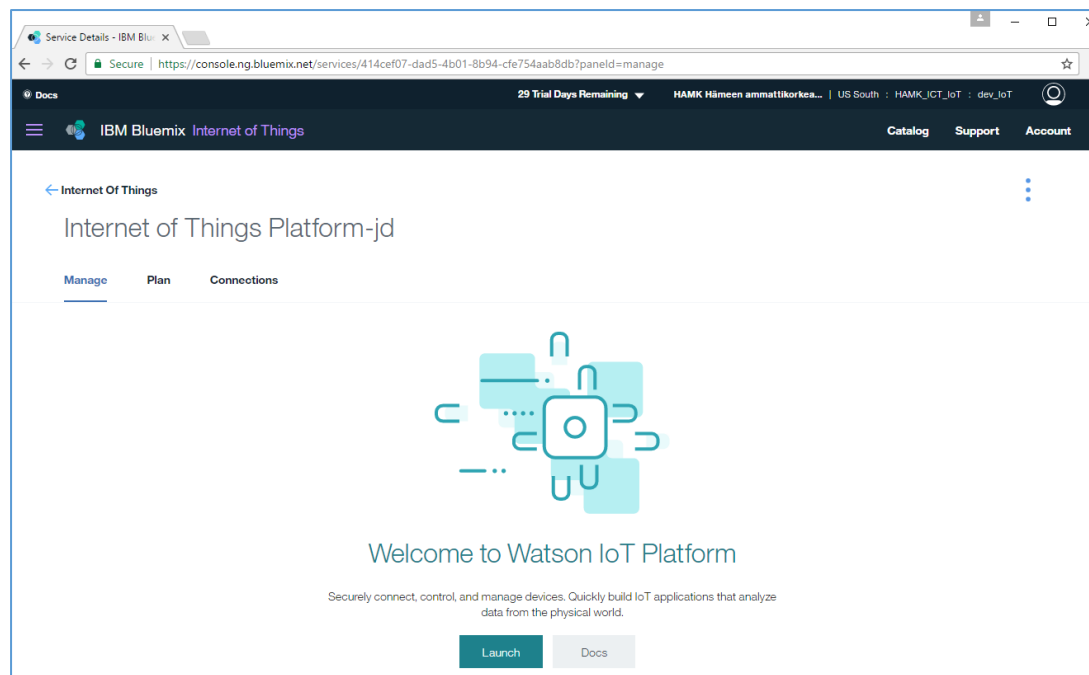


Fig 3.6 IoT Platform

Please select "Launch"

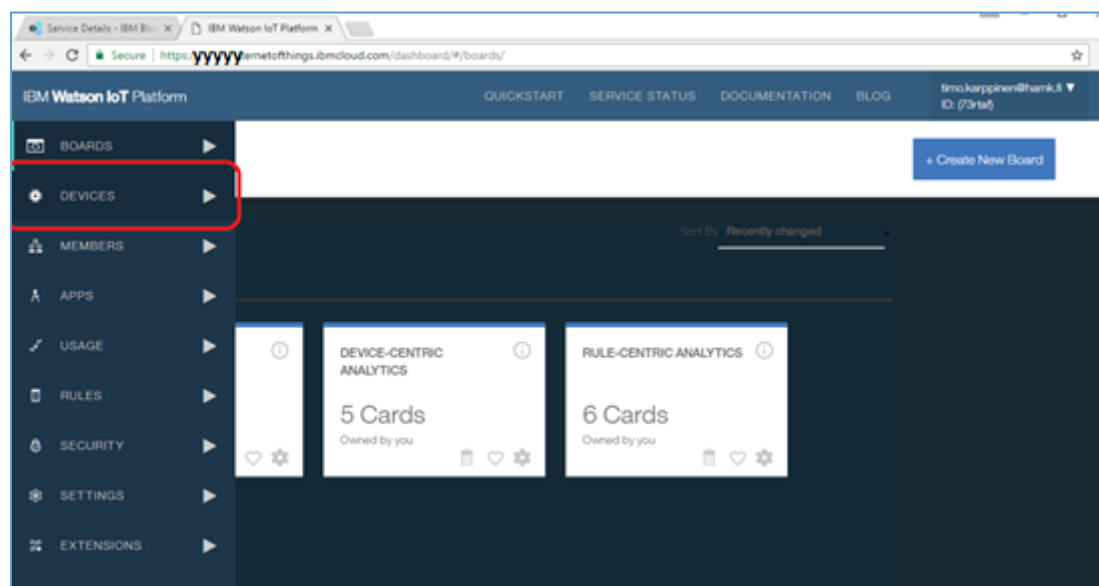


Fig 3.7 IoT platform

In the left please select "Devices" and further "Add new device".

Please fill in the parameters for your device. You always go further with button "Next".

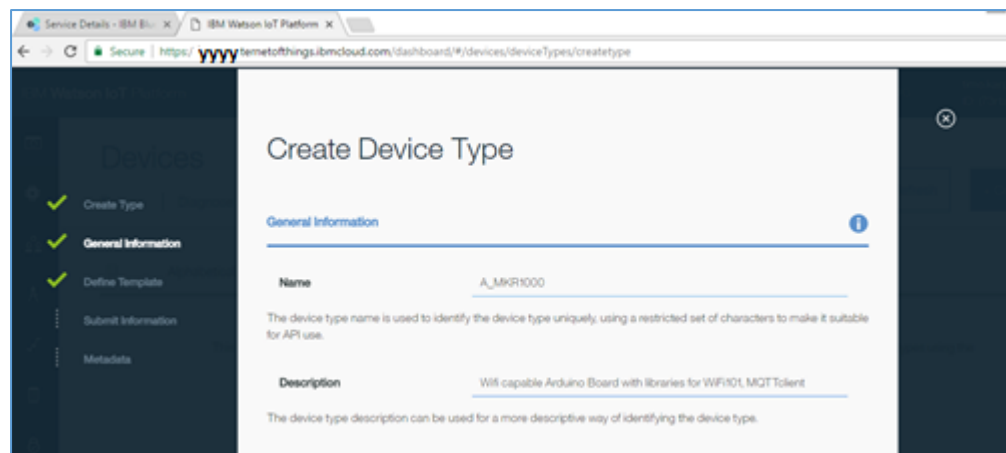


Fig 3.8 The type of the device is given in Device Type.

After filling in the parameters the new Device Type will be visible as option for Choose Device Type.

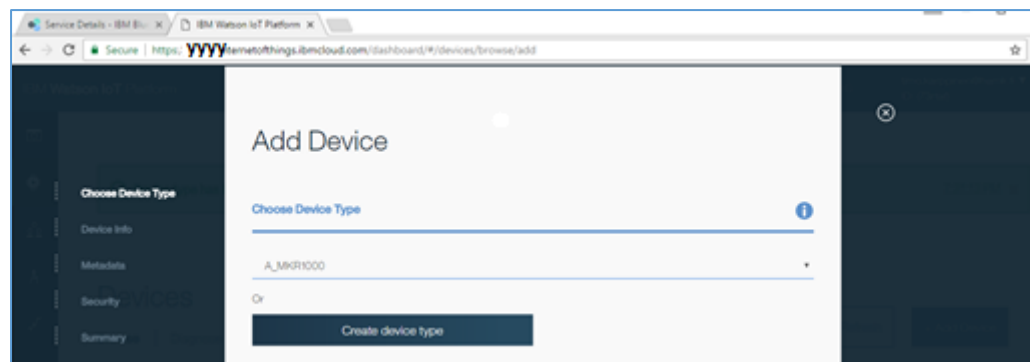


Fig 3.9. The Device Type created just earlier

Select Device and continue on filling in parameters. Please let the system automatically create the authentication token.

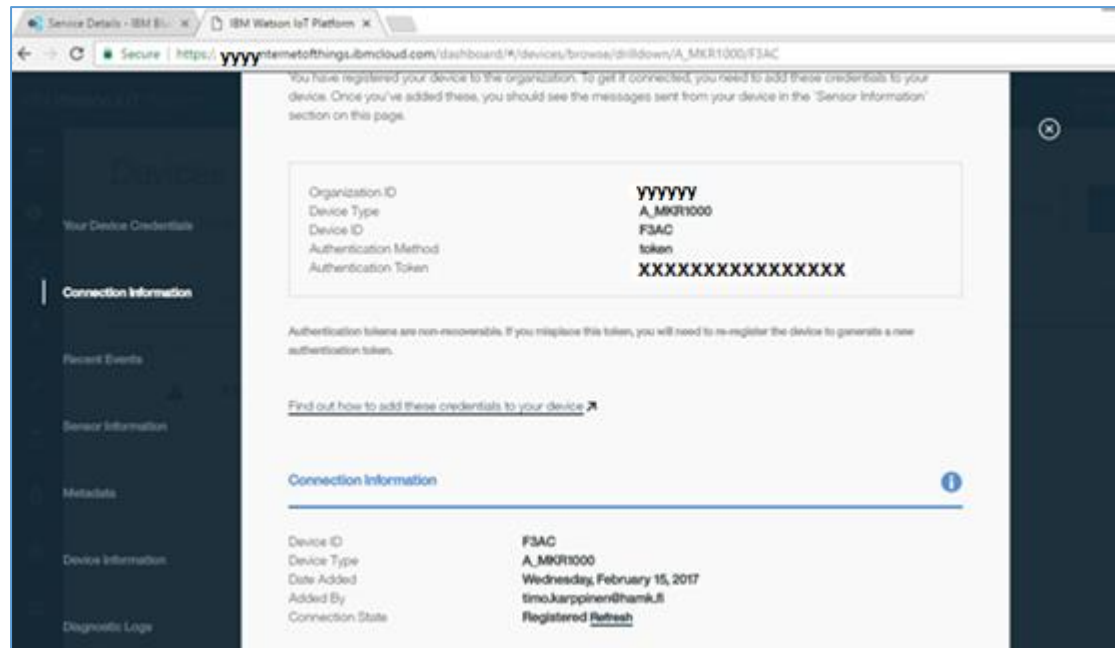


Fig 3.10 Authentication data

Please copy the authentication data from the view into a text file. These will be needed later.

More devices can be added when necessary. An Mqtt Client – it might be a microcontroller or even an application run on a windows computer – registers into IBM Cloud Watson IoT always as a device. Every client needs its own Device ID and Authentication Token.

With the free account In the IBM Cloud Watson the amount of resources might limit how many organizations you have. Please create all your devices to the same Organization ID.

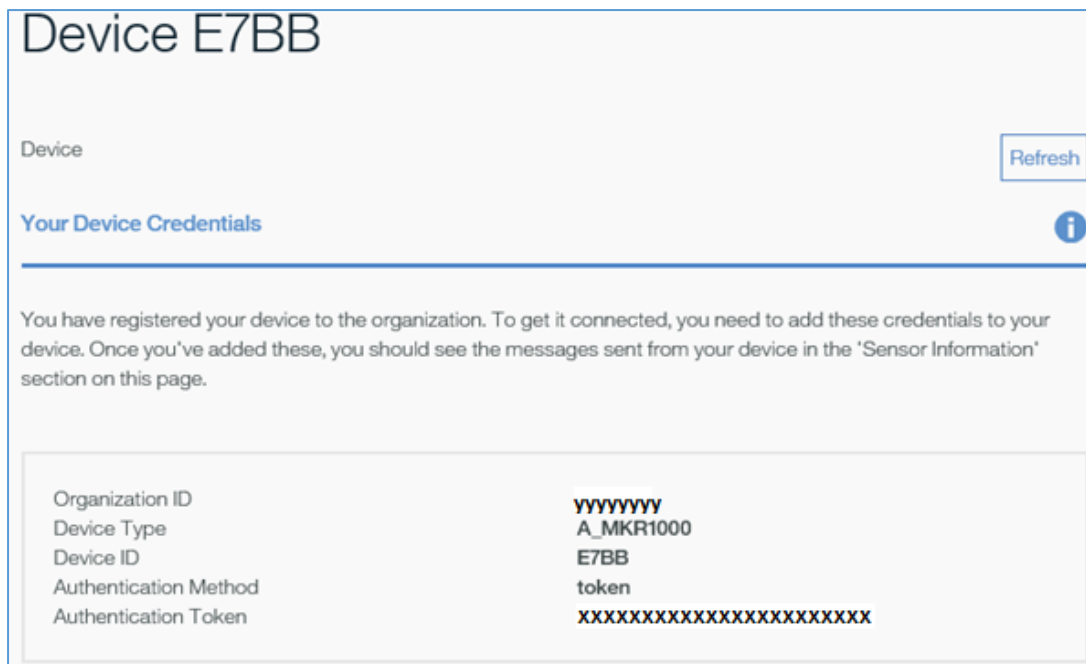


Fig 3.11. Authentication data for another organization and for another device.

Cloud Watson IoT security settings define as default the security with TLS. For these exercises it is easiest to create the connection without the TLS. In the IBM Cloud Watson IoT Dashboard we need to change the settings for security.

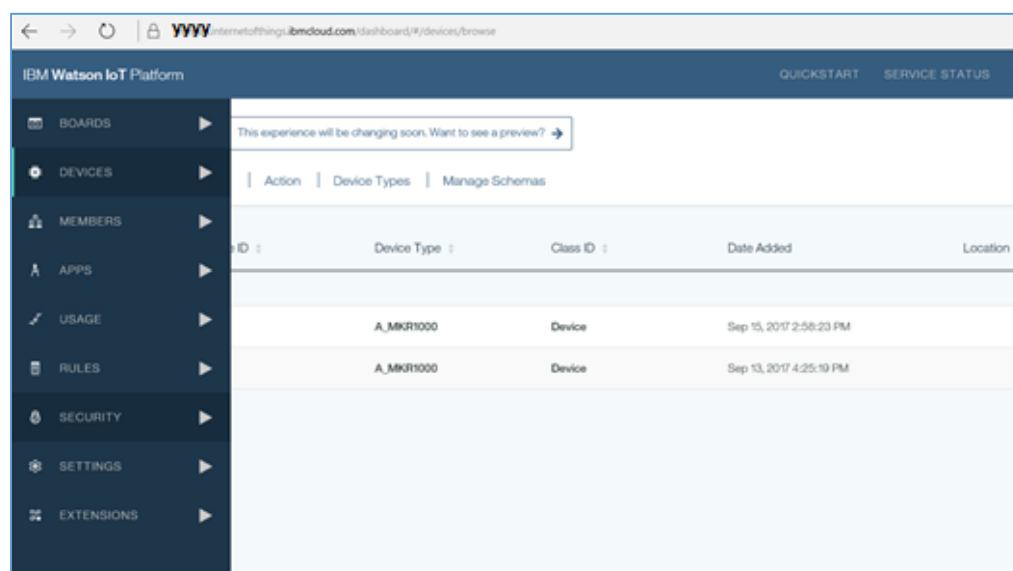


Fig 3.12 Changing security settings in the Security page.

On the menu on left please select "SECURITY". The following kind of page should open.

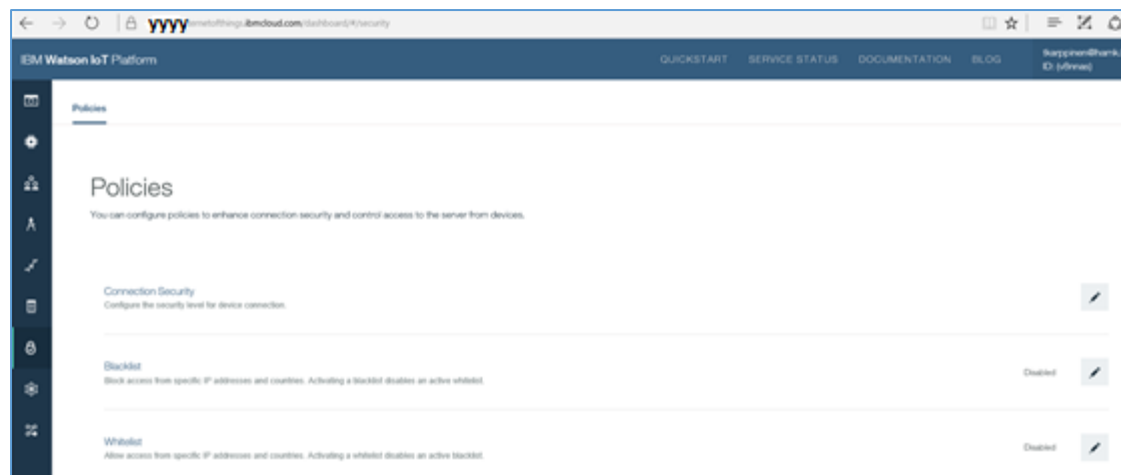


Fig 3.13 Policies, Connection Security

Please select Connection Security by clicking the pen symbol.



Kuva 3.14 TLS Optional

Please select "....CONNECTION SECURITY....", "TLS Optional"

4. Sensor data in the IBM Cloud Watson IoT

Now you can connect your iot device with sensors and microcontroller into the IBM Cloud Watson IoT service. Please study the code example "A MKR1000 sound" and create for yourself something similar. In the laboratory exercises there will be tutoring available for this.

When the connection has finally been successfully done the information sent by the device will be visible for registered users in the Watson IoT Platform in Dashboard, Devices.

The screenshot displays the IBM Watson IoT Dashboard for a specific device, F3AC. The interface is divided into a left sidebar with navigation options and a main content area. The main content area is titled 'Device F3AC' and includes a 'Refresh' button. Below the title, there is a 'Connection Information' section with details about the device ID, type, date added, and connection state. A 'Recent Events' table lists events with columns for Event, Format, and Time Received. A 'Sensor Information' table at the bottom shows data points with columns for Event, Datapoint, Value, and Time Received.

Device F3AC

Device ID: F3AC
Device Type: A_MKR1000
Date Added: Wednesday, February 15, 2017
Added By: timo.karpinen@hamk.fi
Connection State: Connected on Sunday, February 19, 2017 at 7:03:12 PM from 91.145.91.156 with an insecure connection [Refresh](#)

Recent Events

Event	Format	Time Received
tpm	json	Feb 19, 2017 7:07:42 PM
tpm	json	Feb 19, 2017 7:07:52 PM
tpm	json	Feb 19, 2017 7:08:02 PM
tpm	json	Feb 19, 2017 7:08:12 PM
tpm	json	Feb 19, 2017 7:08:22 PM
tpm	json	Feb 19, 2017 7:08:32 PM
tpm	json	Feb 19, 2017 7:08:42 PM
tpm	json	Feb 19, 2017 7:08:52 PM
tpm	json	Feb 19, 2017 7:09:02 PM
tpm	json	Feb 19, 2017 7:09:12 PM

Sensor Information

Event	Datapoint	Value	Time Received
tpm	name	Analog input value	Feb 19, 2017 7:09:12 PM
tpm	tpm	792	Feb 19, 2017 7:09:12 PM

Fig 4.1 The data sent by the device is visible on IBM Watson IoT, Dashboard, Device -page. / https://yyyyyy.internetofthings.ibmcloud.com/dashboard/#/devices/browse/drilldown/A_MKR1000/F3AC , 2017

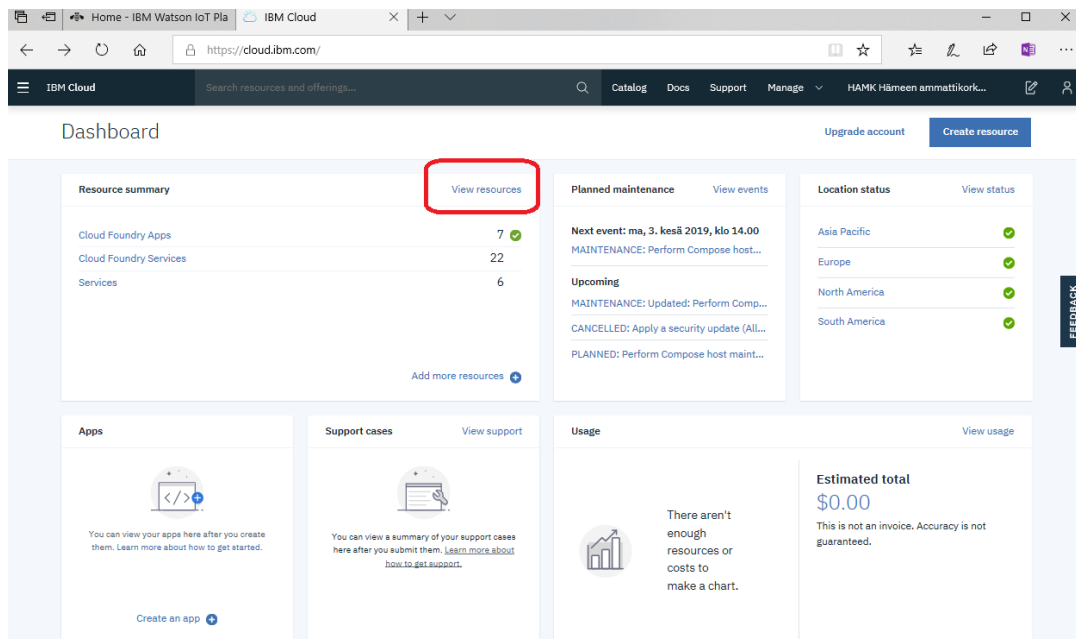


Fig 4.2 If you ended up in page like this when searching the Dashboard view please select the View Resources circled with red line in the picture above. /

<https://cloud.ibm.com/> 3.6.2019

When looking for the Dashboard view you might have ended up in a page seen in picture above. Please select

View Resources

Cloud Foundry Services, Internet of Things Platform ...

Launch

Now you need to get to page

<https://yyyyyyyy.internetofthings.ibmcloud.com/dashboard/> where the yyyyyyyy is the IoT platform identity you created earlier.

5. Data visible on a Dashboard view in IBM Cloud Watson

The data analysis can be started by looking at an example. Start the search with words IBM Developer. On page <https://developer.ibm.com/> you can select Topics, Technologies, IoT. On page <https://developer.ibm.com/technologies/iot/> there are dozens of examples. Different ways of using data produced by IoT is told in the article Making sense of IoT data <https://developer.ibm.com/tutorials/iot-lp301-iot-manage-data/> . Please study that one.

We will continue by analyzing data directly on the IoT Platform in so called Dashboard. The pictures seen below are from IBM Bluemix in the form those features opened on 27.9.2017. In 2019 the features are still the same. But there might of course be some differences in appearance.

We suppose that you have created your user account in the IBM Cloud Watson IoT service and you have logged in. We suppose as well that your device is sending measurement data into Watson IoT Platform. You are in the IoT Dashboard view. Please select on menu left the Boards.

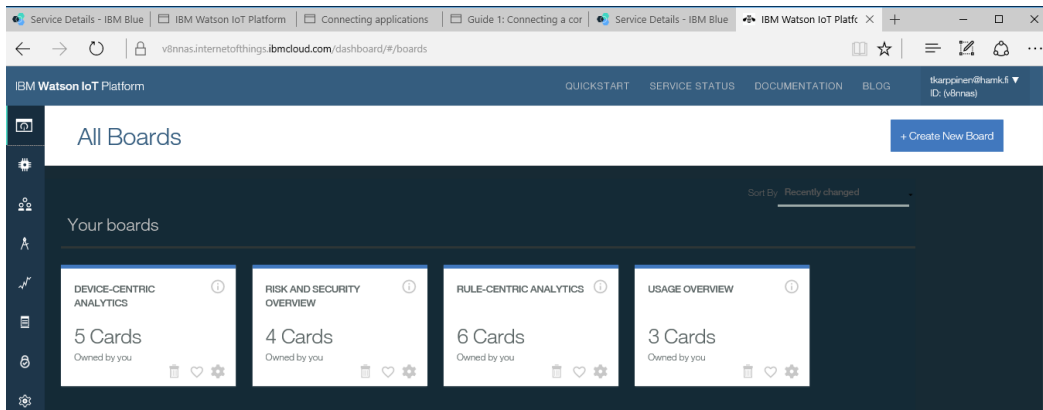


Fig. 5.1 Dashboard, Boards / 2017

In there you can select Device Centric Analytics.

If the Device Centric Analytics is not available as one of the default boards we will be able to create similar board view by ourselves.

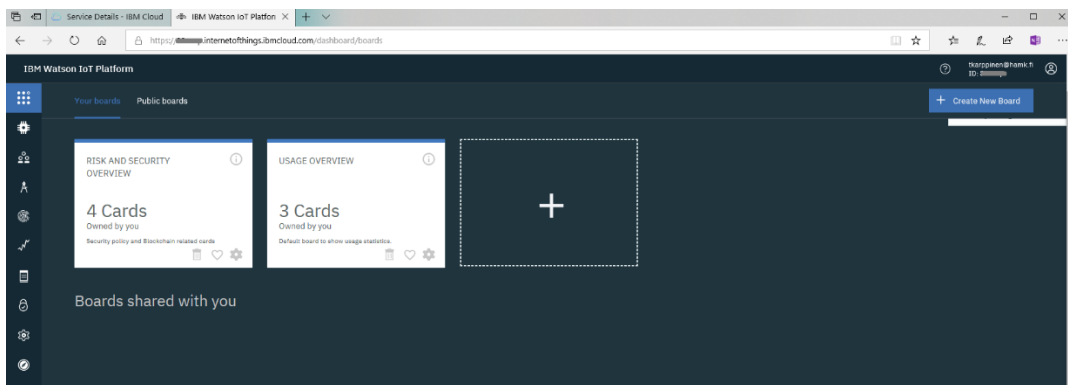


Fig 5.2 A new Board is added by clicking the + symbol / 2019

Please add in this view a new Board card. Please name it as Device Analytics.

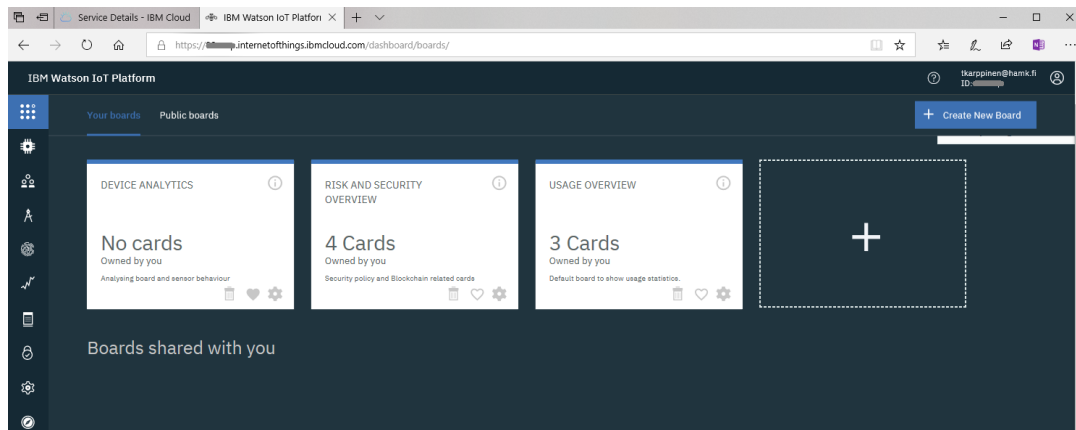


Fig 5.3 The new board view has been created. / 2019

Please select this new board and further Create Card.

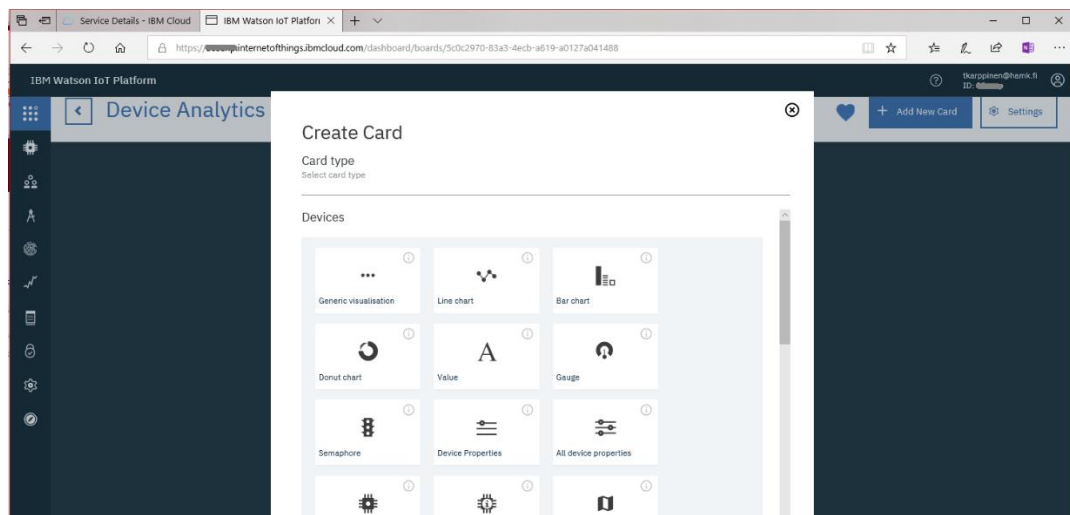


Fig 5.4 Card types / 2019

From the available cards please select Line chart.

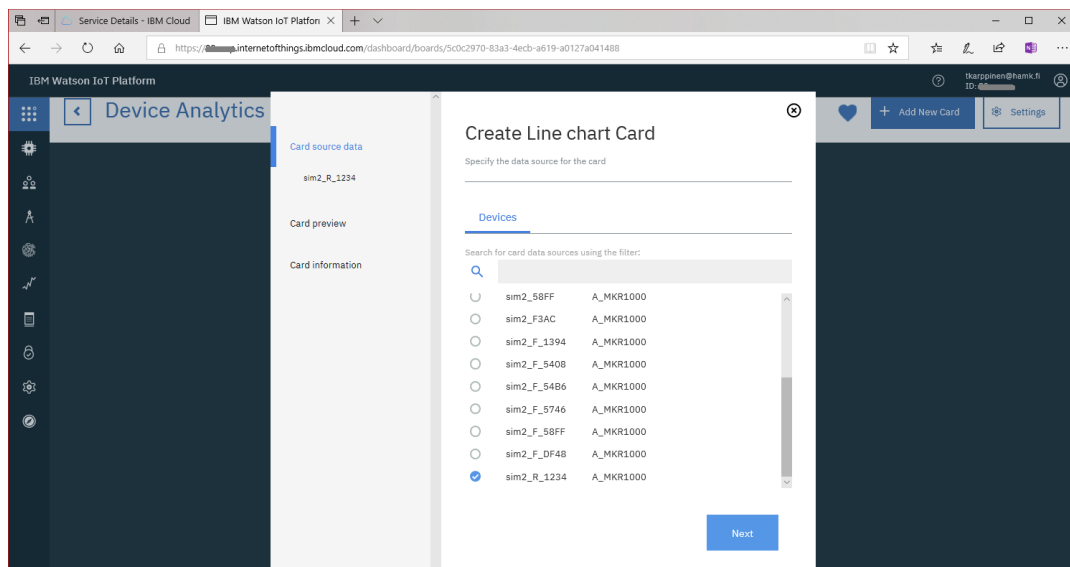


Fig 5.5 Adding a device in the Line chart card. / 2019

Please select from which one of your devices you would like to have the measurements visible.

Select the Connect new dataset. If your device is at the moment sending measurement data compatible with the authentication method on the IoT Platform will the IoT Platform be able to suggest for you the correct parameters.

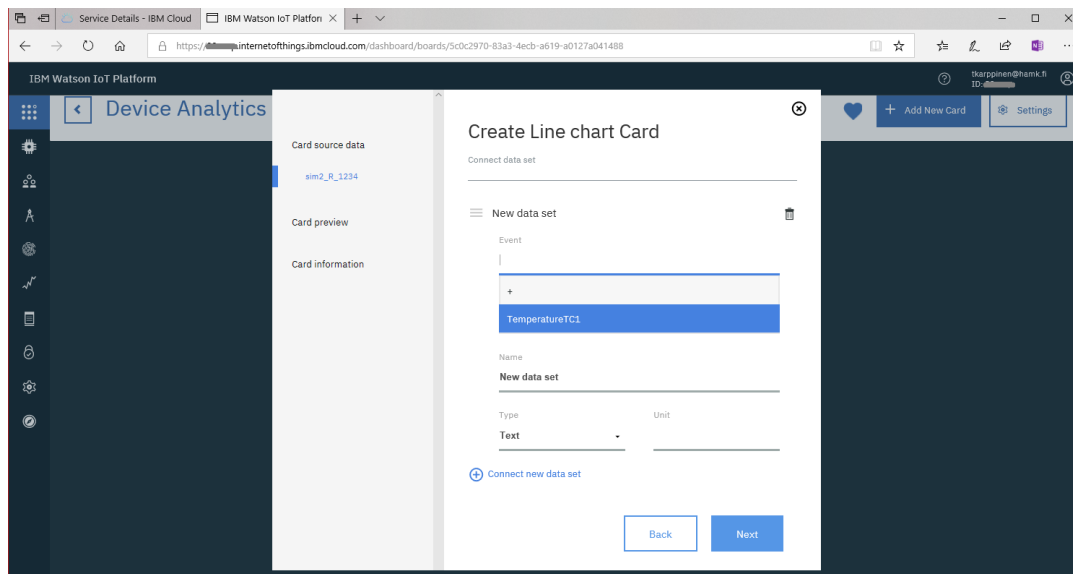


Fig 5.6 The Event type is taken from the last mqtt message your device has sent to the platform. / 2019

New Data Set event type is taken as it has been defined on creating the code for producing the mqtt message. The mqtt message compatible with Watson IoT contains the Even type.

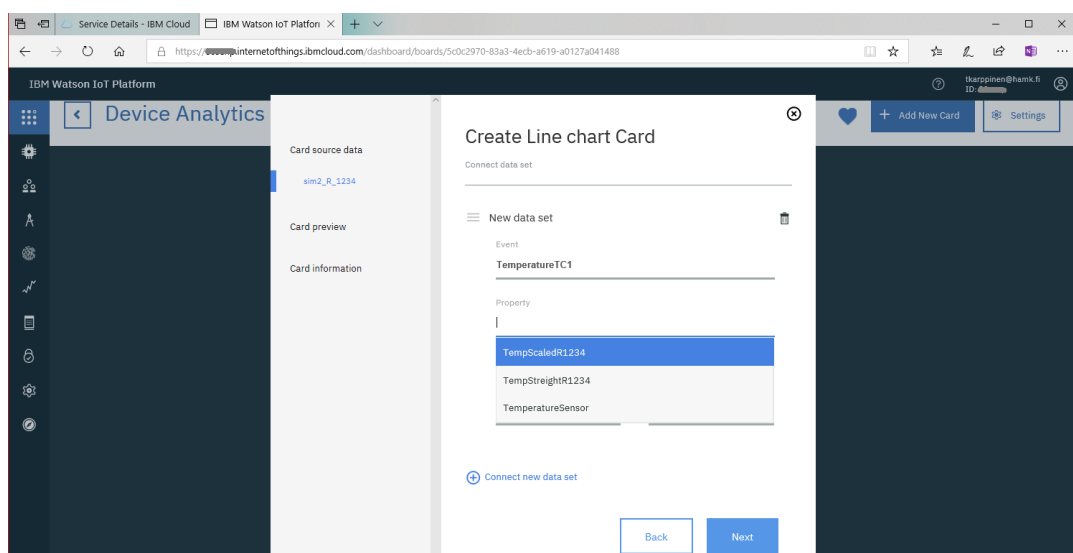


Fig 5.7 Property-types are taken from the last message your device has sent. / 2019

Please select a numeric data. That will be very well visible on a Line chart.

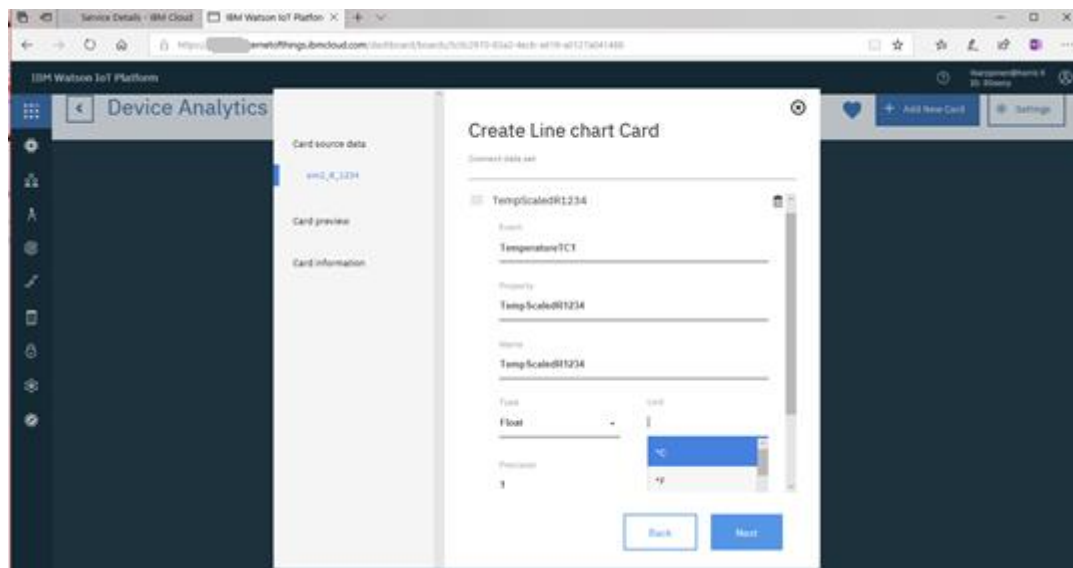


Fig. 5.8 The type and unit for the data / 2019

Please continue by defining correct type and unit for the data.

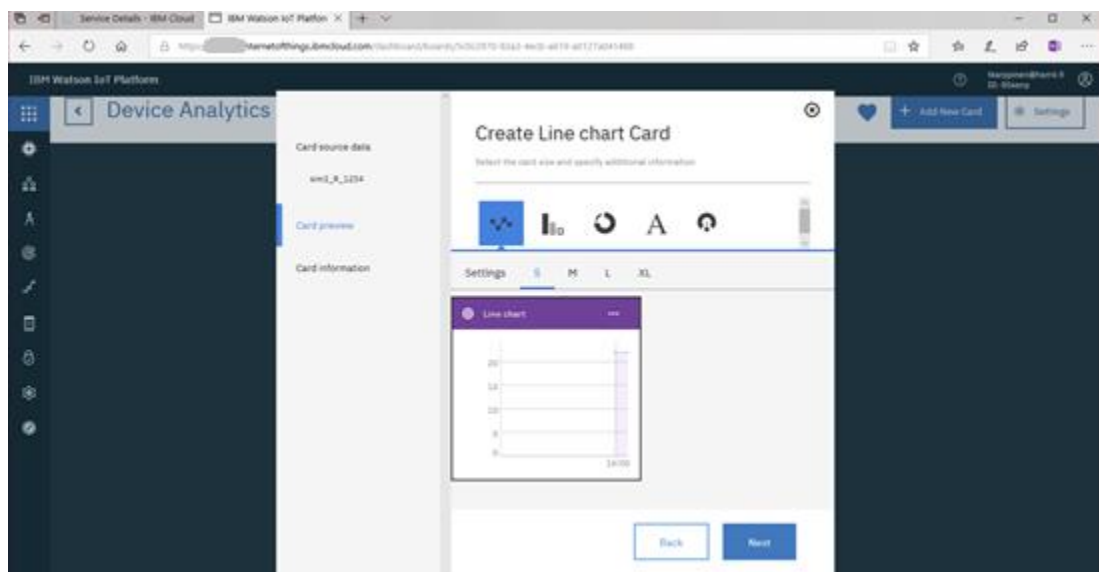


Fig 5.9 The measurement data will be visible / 2019

Now the real time measurement data should be visible!

If you had the board type Device Centric Analytics available as one of the default boards there should be a few card types as option. You can use one of the optional cards or you can create a new card as described above.

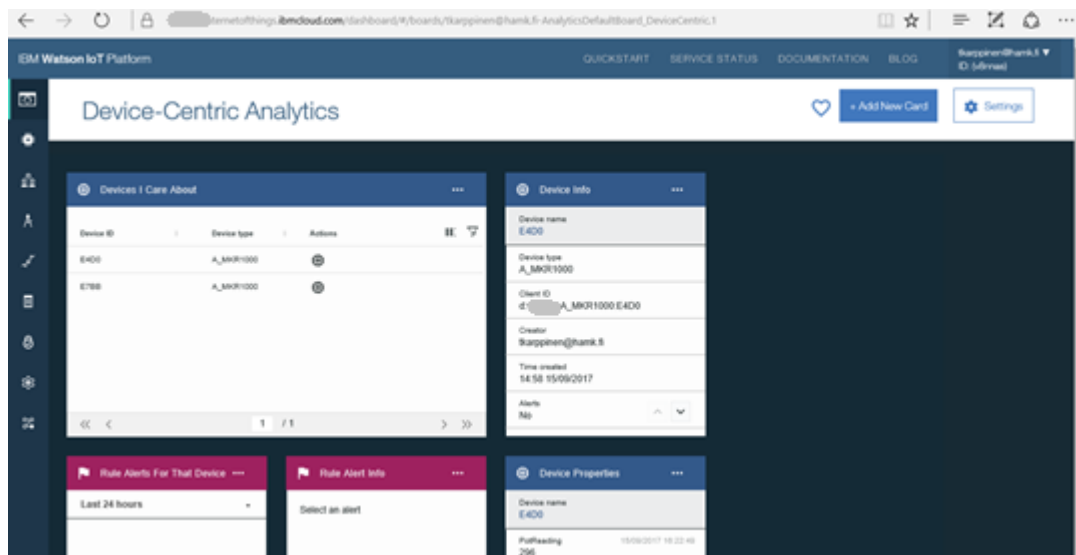


Fig. 5.10 Device Centric Analytics – one of the default boards. / 2017

Please select in this view your own device. Please select on top right Add New Card.

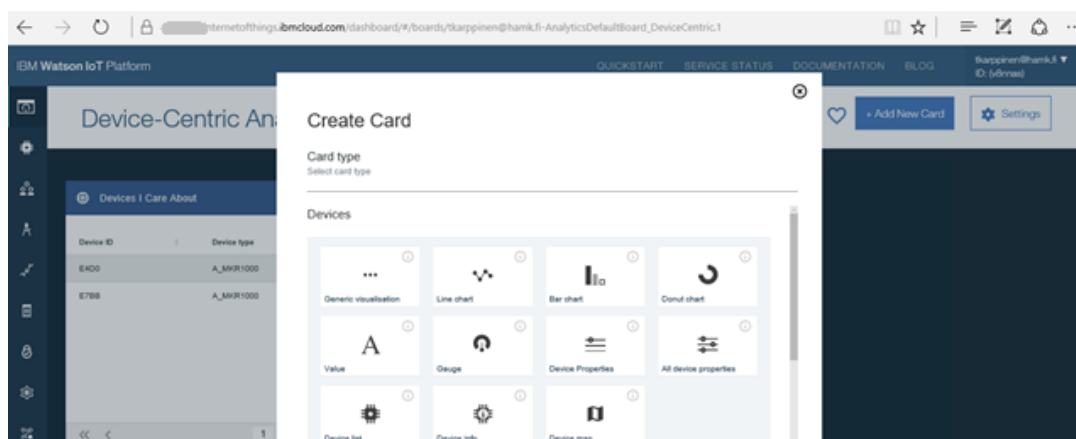


Fig 5.11 Card types / 2017

Please select Line chart.

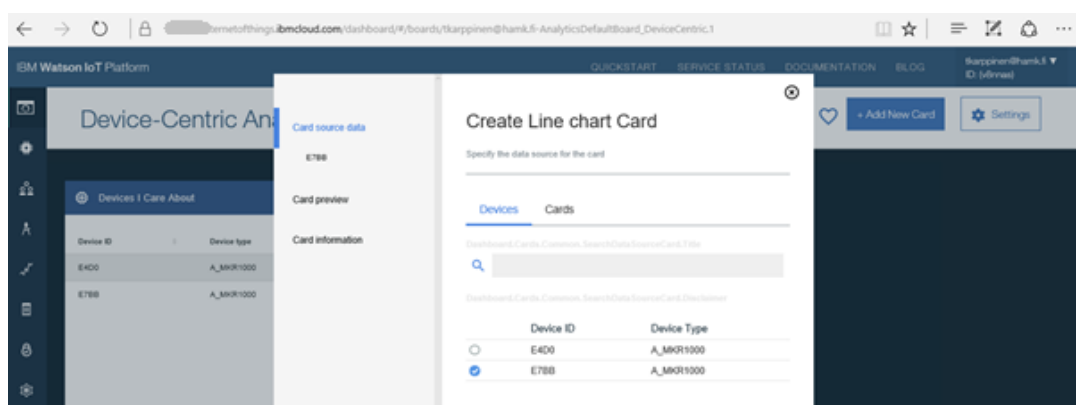


Fig 5.12 Selecting the device we would like to monitor. / 2017

Please select from which device you would like to collect data.

Next you will get a view where you will define which data and on which scale you would like to present. The system knows which Event, Property, Name.... parameters you have used earlier on publishing on the Watson IoT Dashboard. By clicking the corresponding line the previously used ones will be visible as drop down meny.

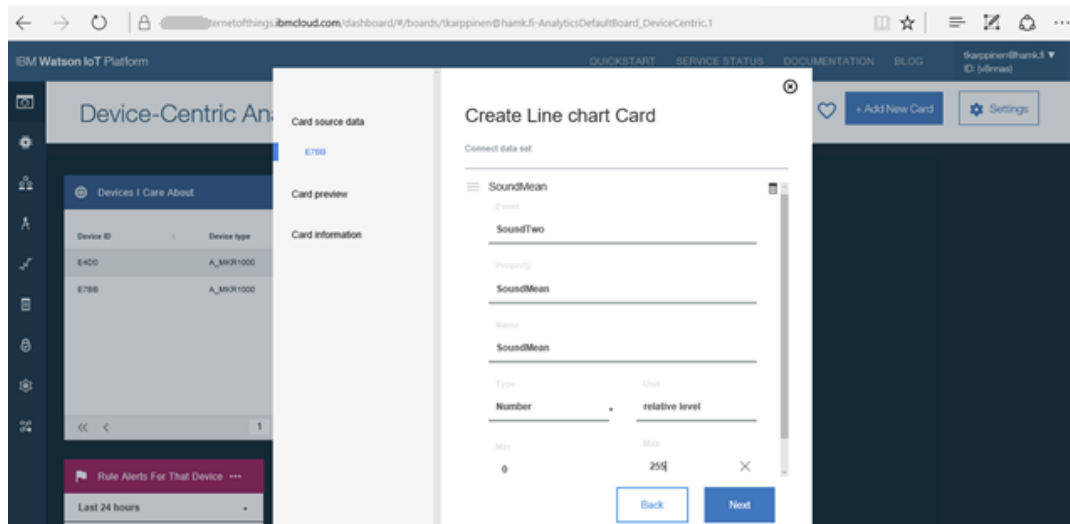


Fig 5.13 Defining the event to be monitored.

You can still create different presentation styles.

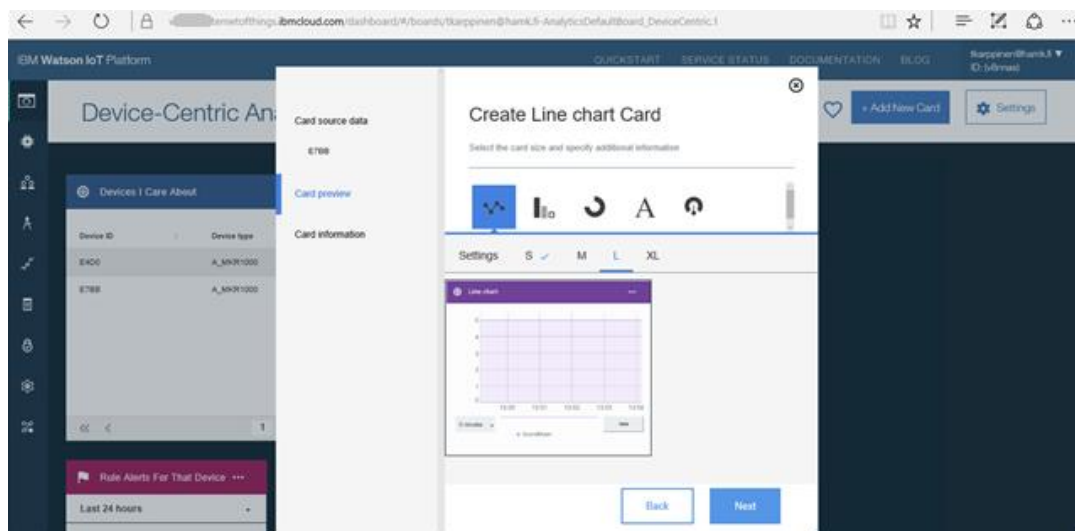


Fig. 5.14 Selecting a presentation style

Select Next and write a name for your chart. With Submit button you will create your presentation.

You do not need to worry about correct scale. After a while the Watson will adjust the horizontal and vertical scales according to your measurement data.

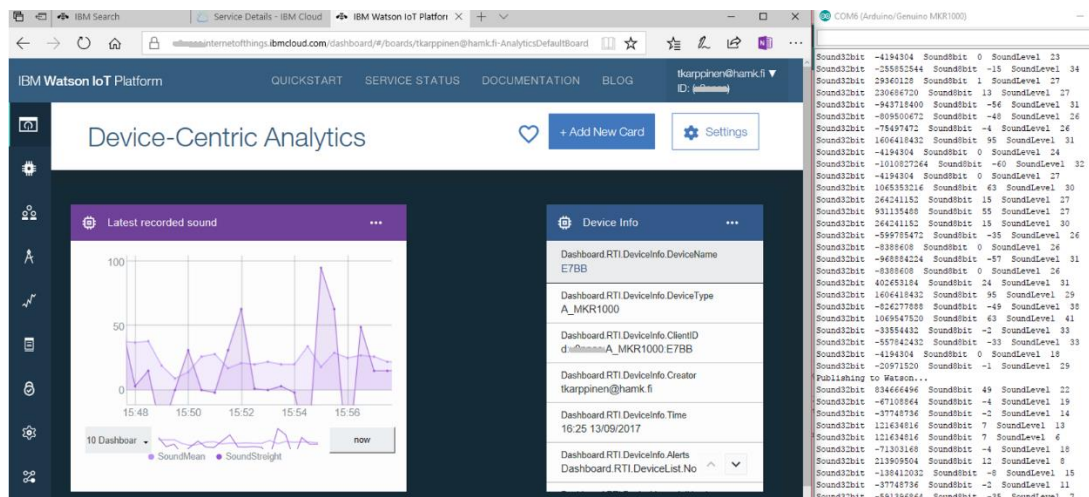


Fig 5.15 Measurement data is visible on the card. On picture right there is a view on the Arduino IDE Serial Monitor.

On picture above the same measurements are visible on the Watson IoT Dashboard and on the Arduino IDE Serial Monitor.

COMMISSIONING TASK 0, OPTIONAL PART

It is not absolutely necessary to complete the task explained in chapters 6 and 7 into one's own system. The commissioning task 1, 2, 3, ... can be completed without these.

The tasks in the chapters 6 and 7 will show how easy it will be to create simple analysis rules for the measurement data.

6. Datan analysis in the IBM Watson Dashboard

It is possible to create some simple data analysis in the Watson IoT Dashboard. These are for the monitoring and maintenance purposes for the system integrator. These are not for the final customer.

Please log in to the IBM Cloud account with your user credentials. We suppose this user already has measurement data flowing to the IoT Platform.

Browse to page Console Internet of Things

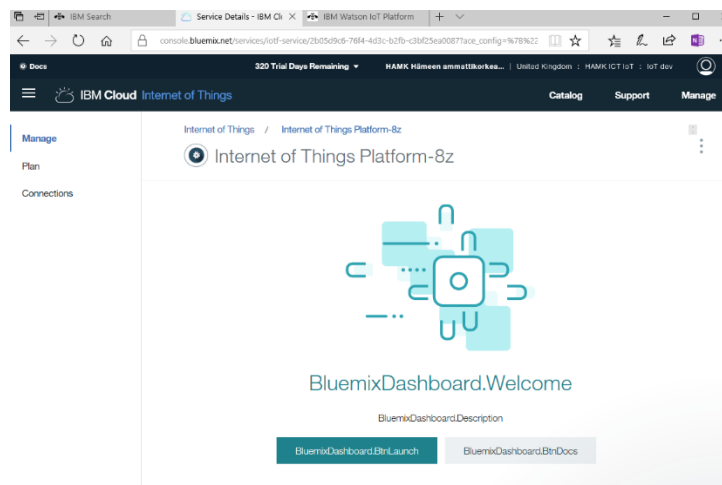


Fig. 6.1 The IoT Platform from previous exercises will be launched.

Click the button `BluemixDashboard.BtnLaunch`. It is possible that the name of the button is simply "Launch".

You will be moving to IBM Watson IoT Platform view. On the menu left please select Boards.

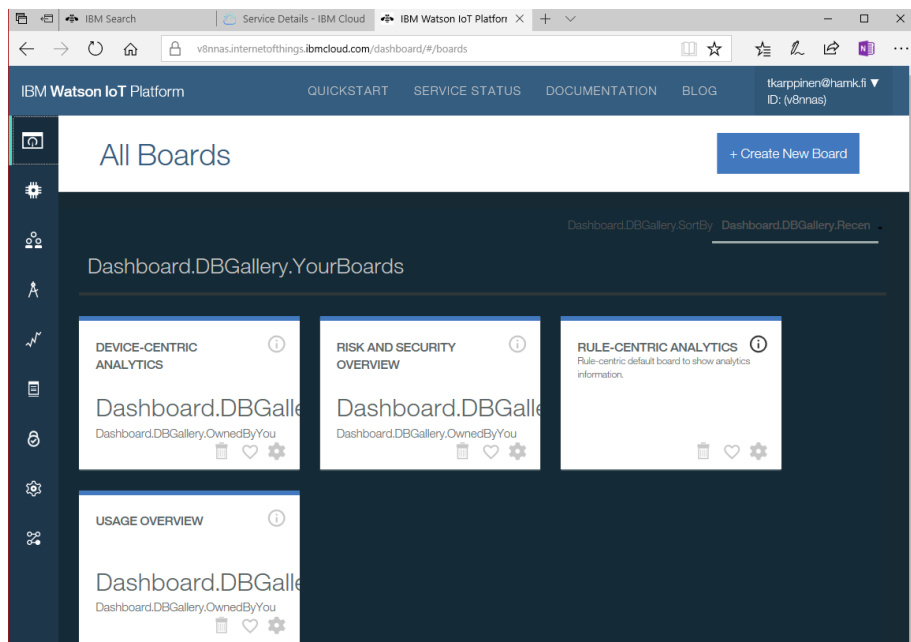


Fig. 6.2 Boards is selected on the menu left.

Please select RULE-CENTRIC ANALYTICS.

On the opening view please click Add New Card.

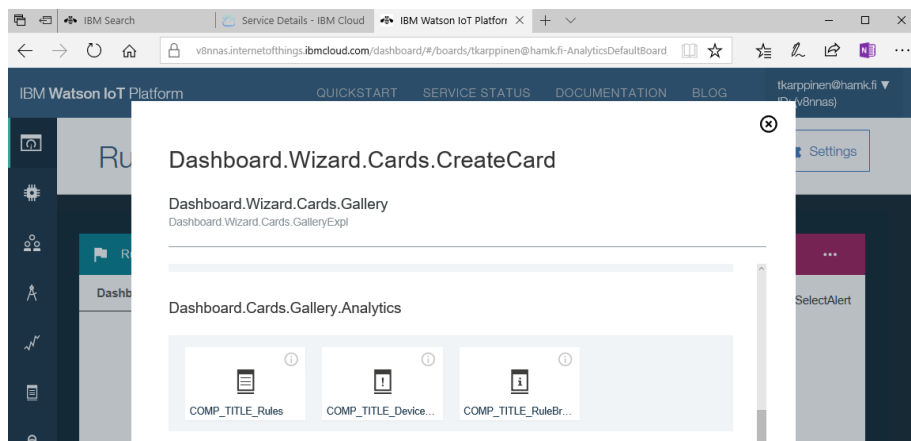
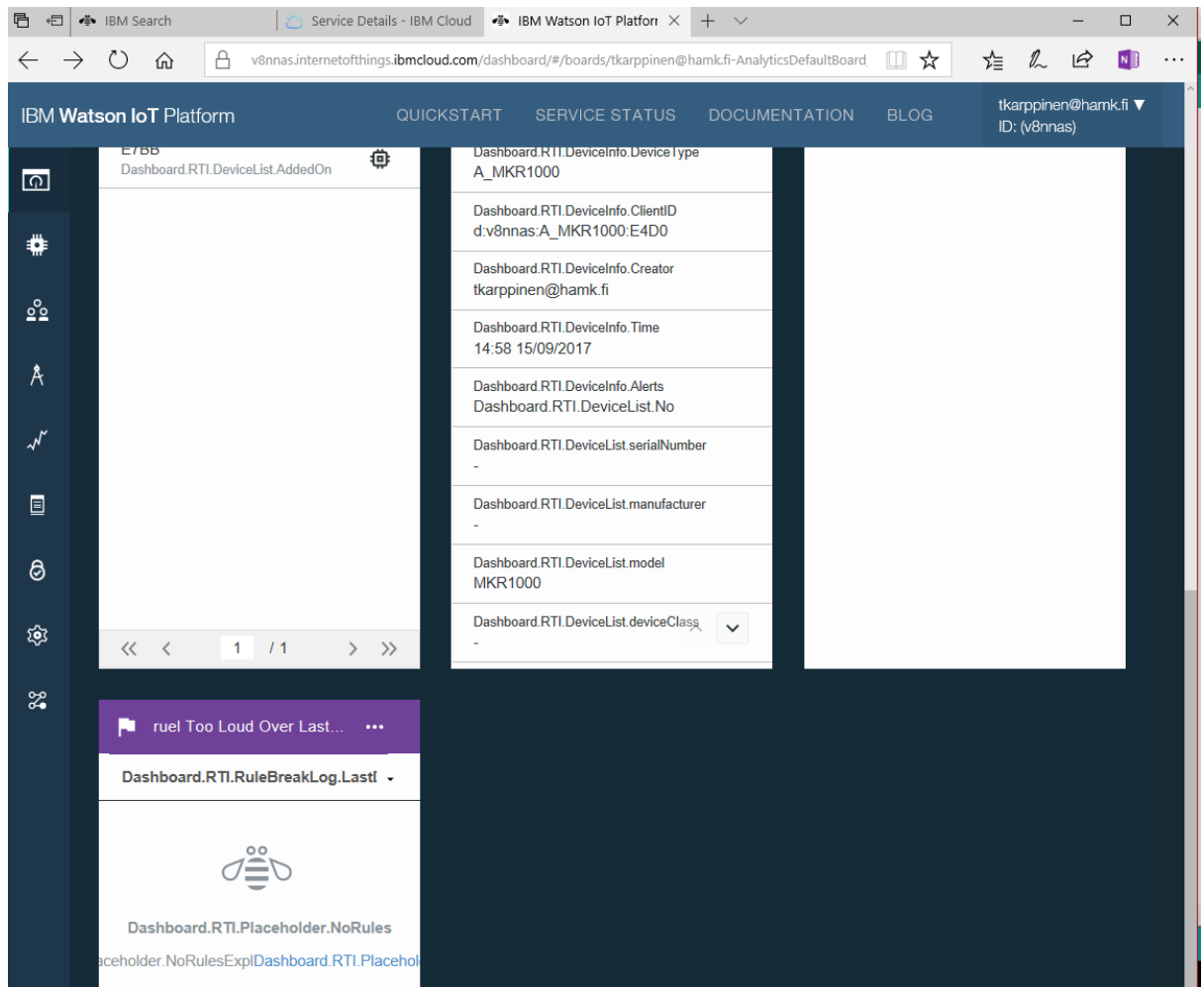


Fig. 6.3 New Card

Please select COMP_TITLE_RULES

Fill in the form. Start by giving a name for the rule: Too Loud Over Last Hour.

The rule will appear as a new rule card.



6.4 The new card can be seen on bottom in the rules view.

In the new rule card – the lowest one in the view – please click `Dashboard.RTI.Placeholder`.

In the opening view click the **+Create Cloud Rule**.

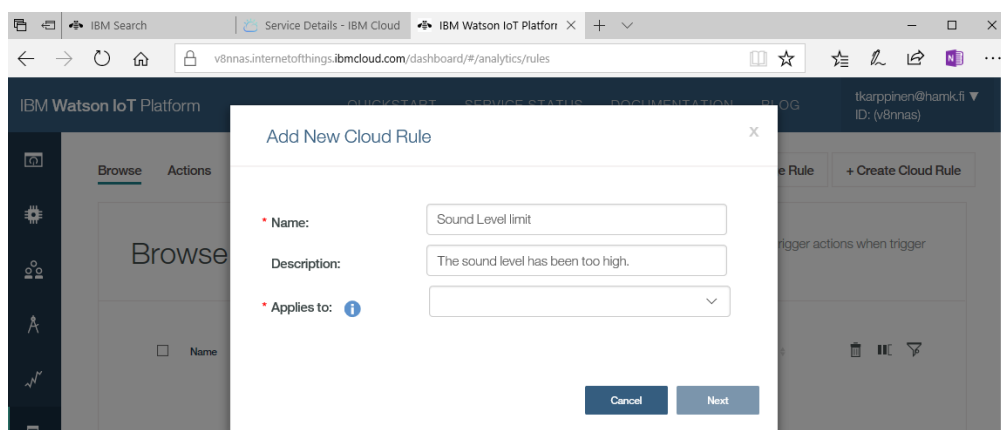


Fig. 6.5 A name is given for the rule.

Give a name for the rule. By clicking the icon “i” you will get instructions: “Go to Devices > Manage Schemas and add a schema for the device type.”

In the left frame please select Devices and on top of the opening page Devices.Menu: Schemas and further SchemaPage.button.AddSchema.

You will be creating a Schema for your device type – for example A_MKR1000.

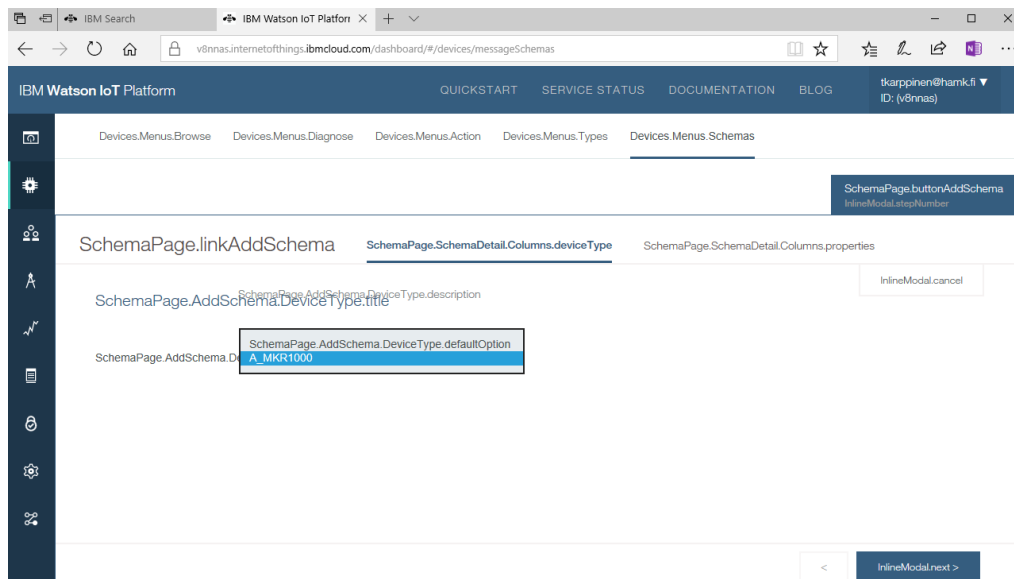


Fig. 6.6 A Schema is created.

Go further by clicking the button down right. It will take some time before the properties available on your IoT platform will be visible. The property can be for example SoundMean. Please select it.

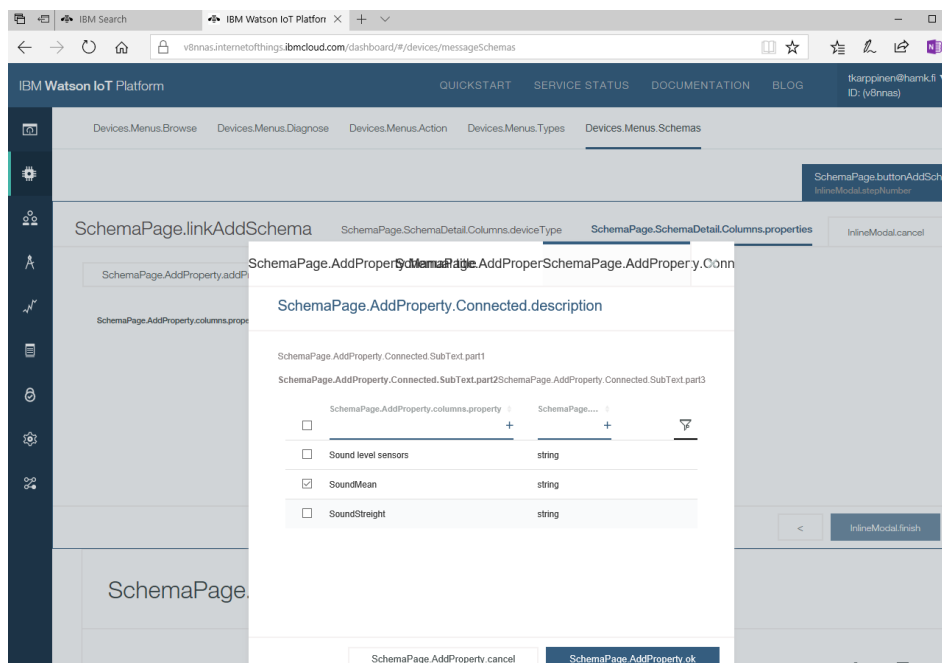


Fig. 6.7 Variables for the analysis will be selected.

Go forward with button SchemaPage.AddPropert.ok and further InlineModal.finish .

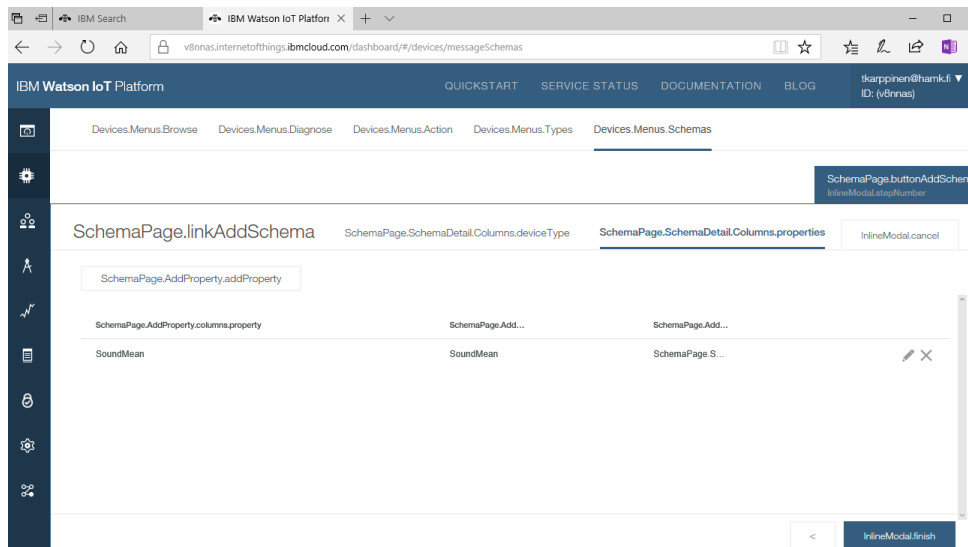


Fig 6.8 The Schema is ready

You have created a schema with properties:

`SchemaPage.SchemaDetailViewPropertis.columns.property = SoundMean`

`SchemaPage.SchemaDetailViewPropertis.columns.name = SoundMean`

`SchemaPage.SchemaDetailViewPropertis.columns.type =`
`SchemaPage.Schemadetail.typestring`

Now it is time to go back and create the rules. Please select on the left frame Boards and further RULE-CENTRIC ANALYTICS. Please select the rule card you created earlier. It can be for example "rule Too Loud Over Last Hour". Go further with `Dashboard.RTI:RulesBreakLog.LastHour` and on the middle of the card `Dasboard.RTI.Placeholder`.

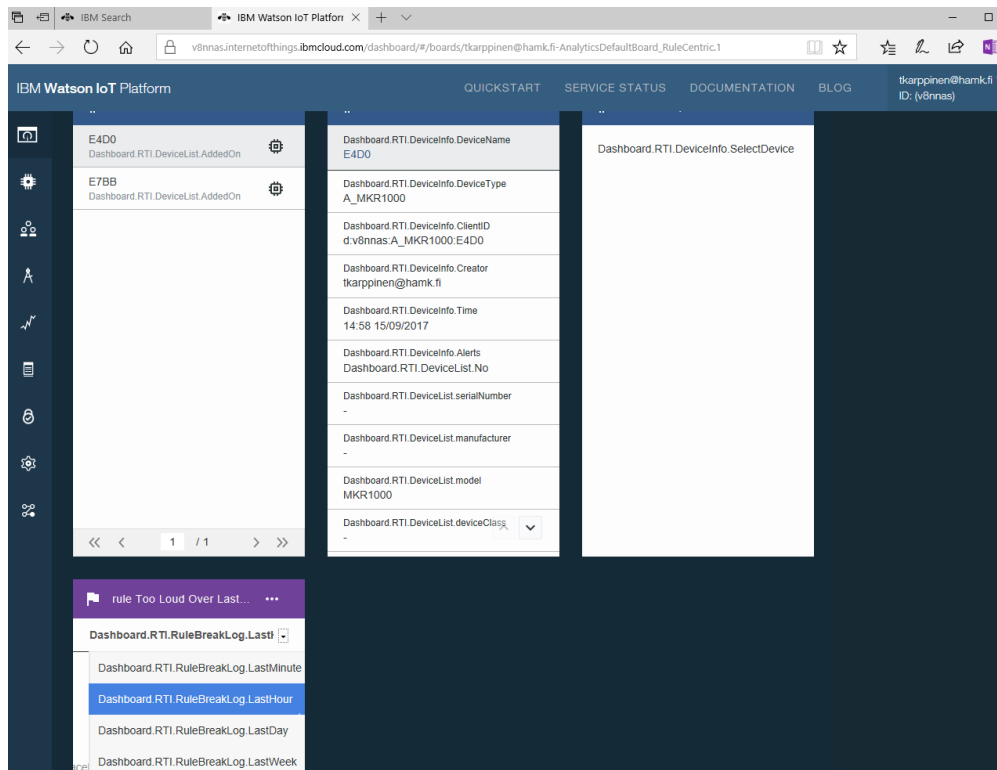


Fig. 6.9 The measurements from the last hour will be taken into analysis with rules.

You can get into this same view also by clicking on the left frame the RULES.

Now you can select again on top right +Create CloudRule.

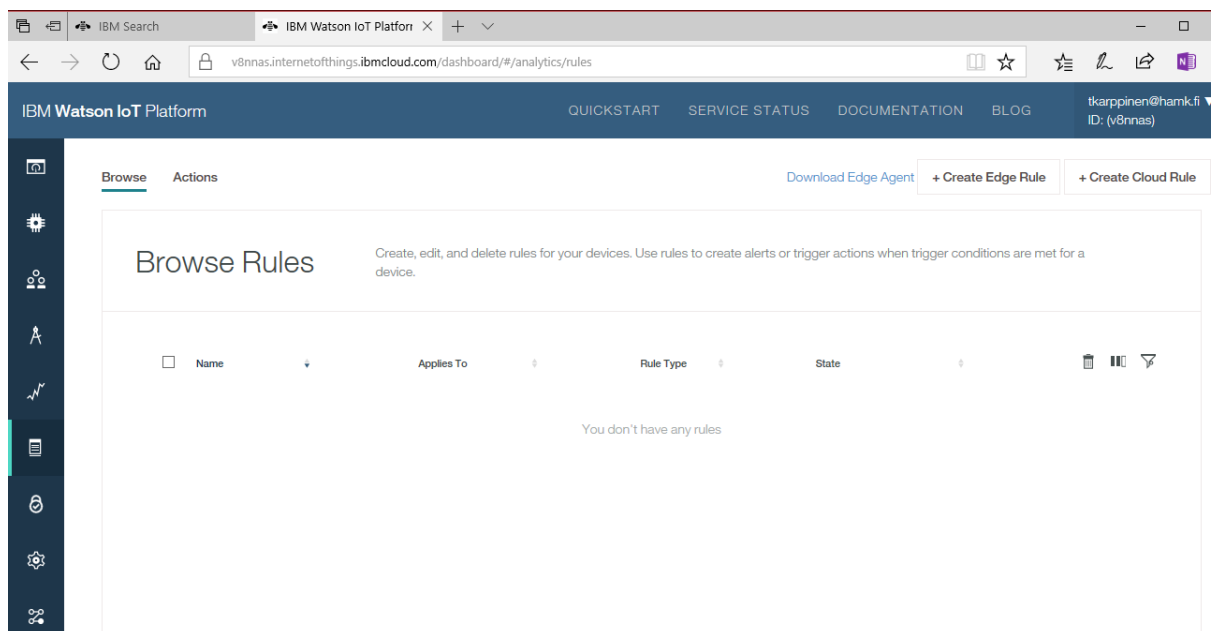


Fig. 6.10 A view where the rules will be created.

It is time to set a rule for your device.

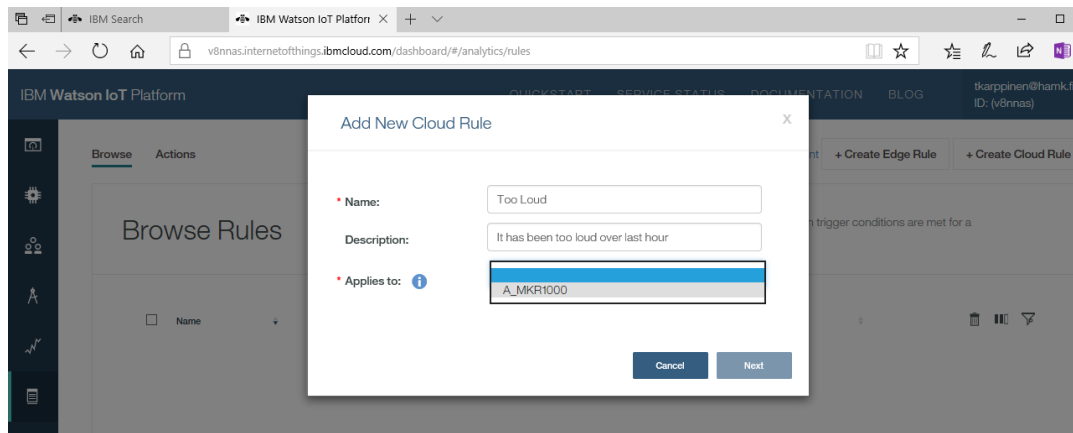


Fig. 6.11 Selecting a device for which the rule will be made.

Please select your device and click Next.

In the opening view you can start creating the rules.

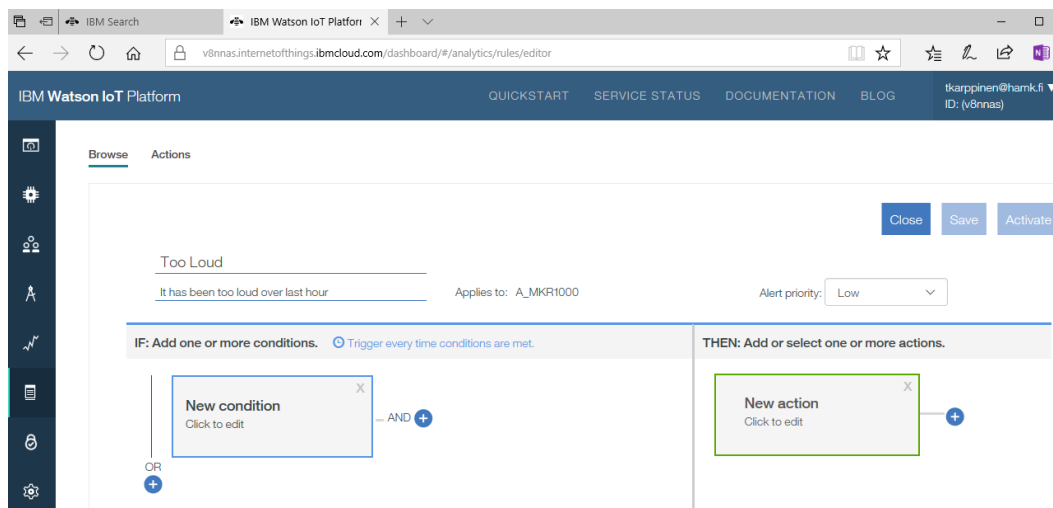
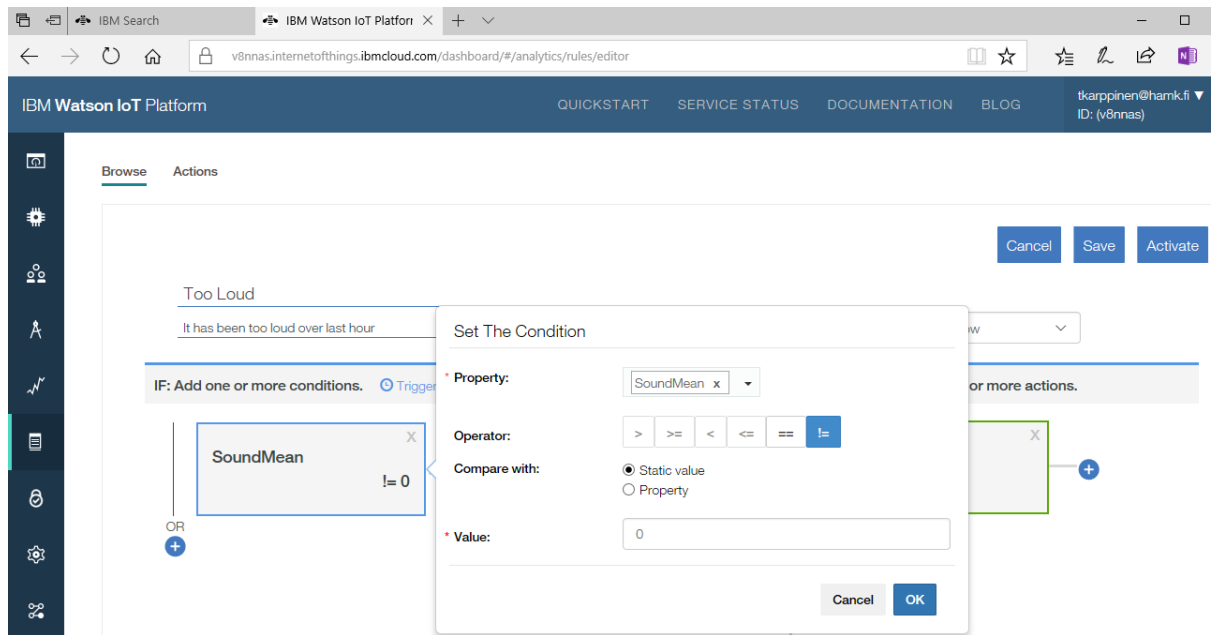


Fig. 6.12 Logic between rules.



6.13 Defining a rule.

Surprise, surprise – there are two rules available only: the == and !=. We have transmitted the mean value of the sound samples as a string. The only rules available for string type variable are the equal or the not equal. We can compare the measurement string with for example string 0 or string 1.

This happened because in the program code in the client device – the Arduino card – we had taken the floating point variable value inside a long message string.

There is a better code version here. With the following code the float type variable value inside the JSON string will be transmitted as an ascii code representing the floating point value.

```
String wpayload = "{\"d\":{\"TemperatureSensor\":\"TC1 \",\"TempScaledR1234\":\"" +
String(tempScaledF)+ ", \"TempStreightR1234\":\"" + String(temp14bit)+"\"}"};
```

```
MQTTc.publish("iot-2/evt/TemperatureTC1/fmt/json", wpayload);
```

This is the correct code and Watson IoT platform will be able to read the variable value as a floating point number.

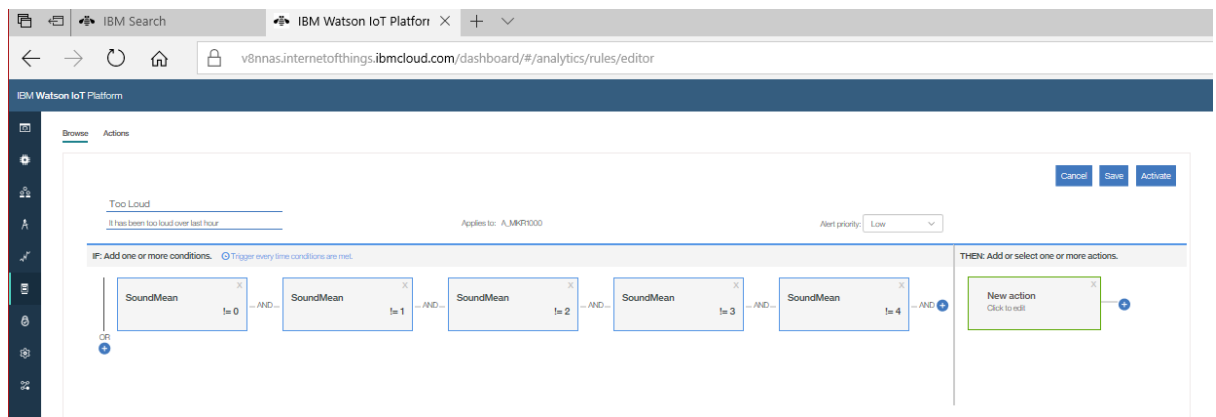


Fig. 6.14 A number of rules connected with logic rules.

The next step is to define what happens when the rules provide the state TRUE. Please click the New Action.

In the opening view please select Add action.

Please add the name, description and type: TooLoudAlert, sends an email, ActionPage.Modal.ActionType:email .

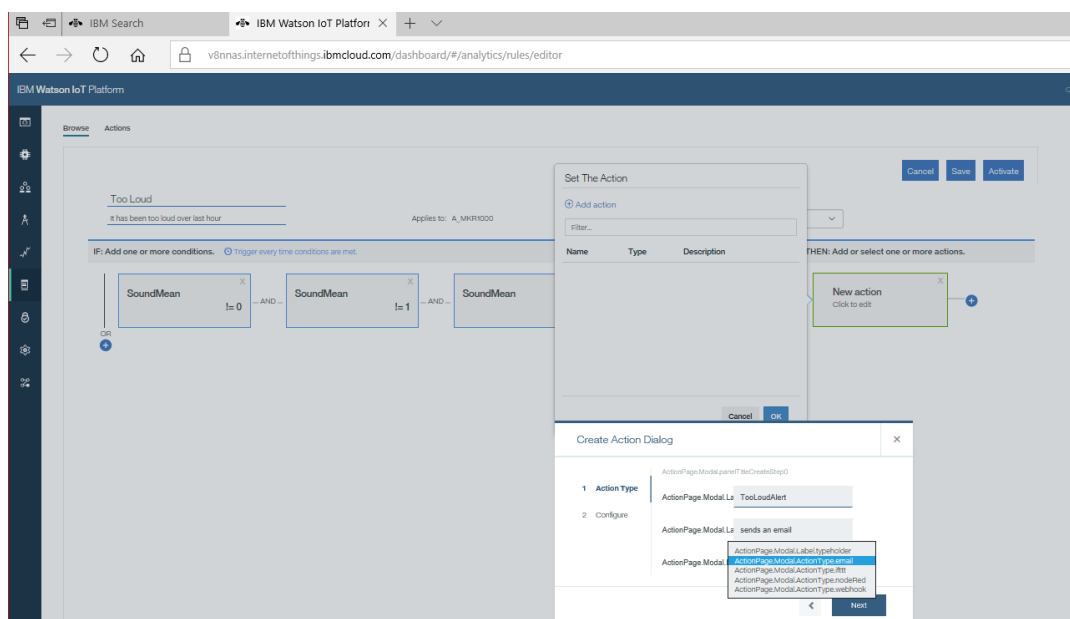


Fig. 6.15 A view where an action can be added.

It might not be a very good idea to send an email every time the mean sound level goes over a limit value. Anyway we will use this as the first example. And this action might be useful for example in estate management. The house lord will get an alert of party noise.

Click Next and fill in:

ActionPage.Modal.Label.tospecialPeople [the email address]

ActionPage.Modal.Label.cc noon

select the ...includeData.

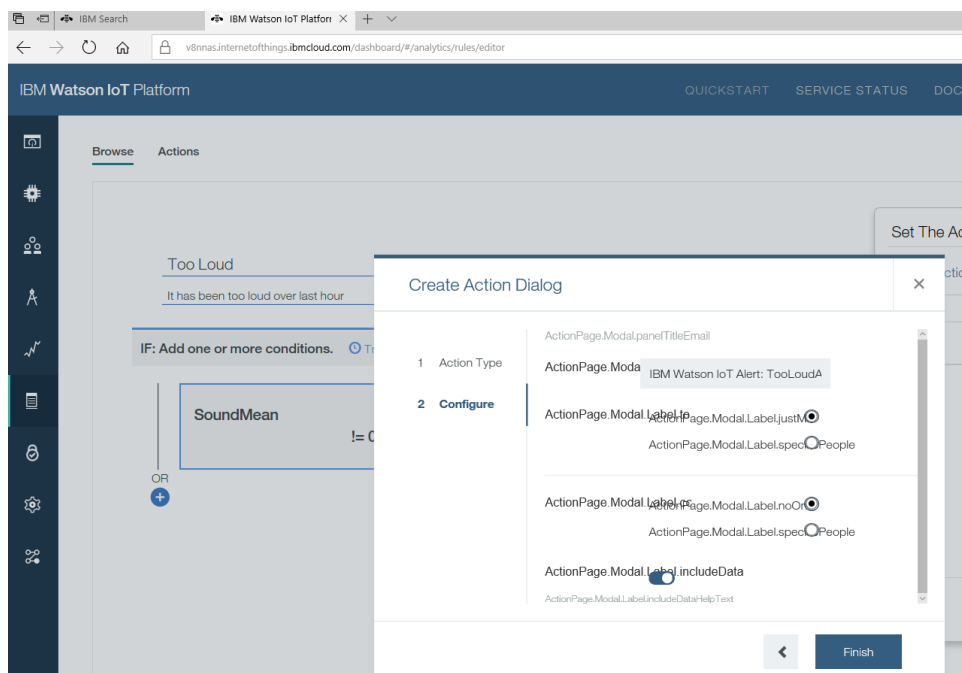


Fig. 6.16 The action

The task can be completed by clicking the Finish.

7. Adding metadata into the measurement values

7.1. Two different types of IoT applications

Let's divide the IoT applications into two categories.

A. One-off application

The application can be more or less unique. An example of this can be a production line in a factory. There can be hundreds of measurement points from where values will be transmitted to the IoT platform. These can be different than those used for daily routines in the automation system controlling the process. Or these can be exactly the same as in the automation system. Typically all these measurement points will be exactly the same in exactly the same application for the next couple of years.

In this kind of application it will be possible to code the sensor device to send both the measurement value and the metadata to the IoT platform. The metadata is for example a description of the measurement point or location of the measurement point.

B. An application where devices are manufactured in large series

We can think that the product is part of a product family aimed at consumer market. Manufacturer of sports accessories can for example install a sensor device into every

manufactured running shoe. The user will be allured to register her/his shoes into a database by offering her/him some analysis services where the exercise progress can be monitored. By registering the consumer opens a direct connection between her/him and the company. The connection can be used for example for marketing purposes.

We don't bother the user with details of registering her/his sensor device into the IoT platform. The user simply gives the product serial number and some personal information when registering to the database.

The running shoe – the sensor device – will always start uploading data into the IoT platform when the running shoe is used. The IoT platform can match the serial number information and the information given by the user. The user and the shoe will get an identity.

The identity and the measurement data can be analysed in the IoT platform with simple rules. The rules can be created for sending correct marketing information at correct time. For example it will be good time to send an advertisement of all weather shoes if the user has been running with conventional shoes in a heavy rain or snow fall. The advertisement can be published on web page banners.

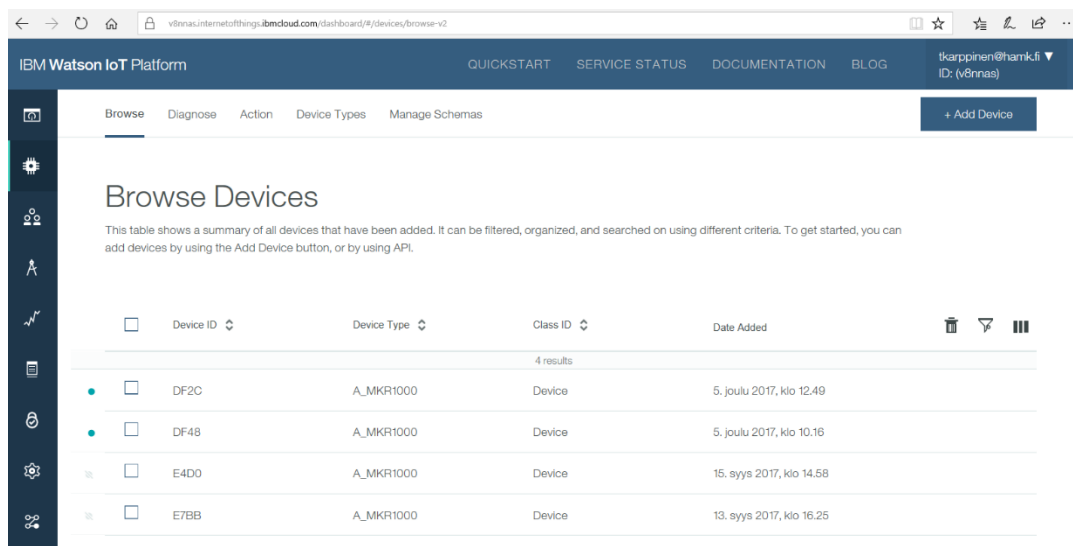
Most of the IoT application are already of this type. An this will be more and more common in the future in all kinds of products.

7.2. Adding metadata in an IoT platform

Again we will use the IBM Cloud Watson IoT as an example of typical IoT platform.

We suppose that the application is of type B. But we don't think this is a consumer market application where the matching of metadata and user data would be completely automated.

In this example the metadata matched with a specific IoT device is transferred to the IoT platform as JSON formatted string.



Device ID	Device Type	Class ID	Date Added
DF2C	A_MKR1000	Device	5. joulu 2017, klo 12.49
DF48	A_MKR1000	Device	5. joulu 2017, klo 10.16
E4D0	A_MKR1000	Device	15. syys 2017, klo 14.58
E7BB	A_MKR1000	Device	13. syys 2017, klo 16.25

Fig. 7.1 Selecting a device with which the metadata is connected.

We will select the device from our device fleet. The metadata will be specific for exactly this device.

In this example all devices are similar and have exactly the same program code. For each device there is an unique identity – the Device ID - in the IoT platform. The ID could include the sensor device serial number. In this example we have written the four last hexadecimal symbols of the device MAC address. Let's select the device DF48.

Click in the view seen above the row for the device and further click Device Information.

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'QUICKSTART', 'SERVICE STATUS', 'DOCUMENTATION', and 'BLOG'. The user is logged in as 'tkarppinen@hamk.fi' with ID 'v8nnas'. The main content area displays a table of devices. The table has columns for 'Device ID', 'Device Type', 'Class ID', and 'Date Added'. There are 4 results shown. The device 'DF48' is selected, and its 'Device Information' view is open. This view shows fields for 'Serial Number', 'Model', 'Description', 'Hardware Version', 'Manufacturer', 'Device Class', 'Firmware Version', and 'Descriptive Location'. The 'View Metadata' button is visible in the top right of the information panel.

Device ID	Device Type	Class ID	Date Added
DF2C	A_MKR1000	Device	5. joulu 2017, klo 12.49
DF48	A_MKR1000	Device	5. joulu 2017, klo 10.16

Device Information	
Serial Number	--
Model	MKR1000
Description	with sensors
Hardware Version	--
Manufacturer	--
Device Class	--
Firmware Version	--
Descriptive Location	--

Fig. 7.2 Some information for the device was already written when creating the device in the IoT platform.

In the opening view click the pencil symbol and further click the edit Metadata.

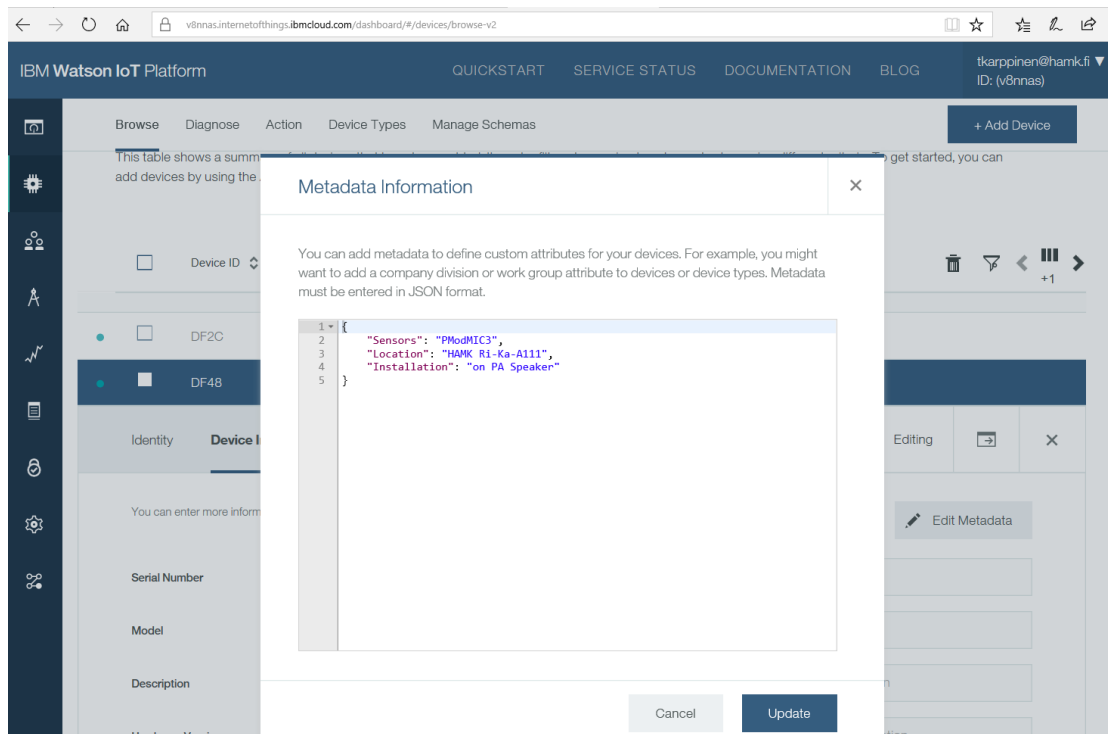


Fig 7.3 Editing the metadata for one of our devices.

Please write the metadata in JSON format. Save it by clicking first Update and immediately after that Save.

Please create a new user interface view. It is a “board”. The instructions for creating a new board can be found in the previous chapter.

Bring in measurement data from at least two devices. When creating the board you will be selecting the device and the measurement data from that device. In the field Name please write a name which will differentiate that device from any other device. In our example we will use four last symbols from the device WLAN unit MAC address.

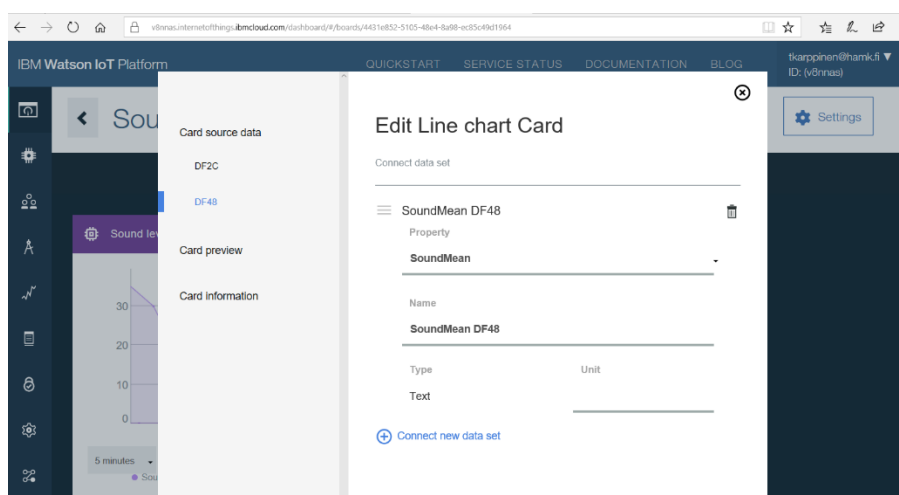


Fig. 7.4 The sensor data will be visible as it has been defined in the JSON formatted sensor device message.

Select the correct sensor data.

In this way we will get sensor data from several devices and we will be able to see from which device each data is coming from.

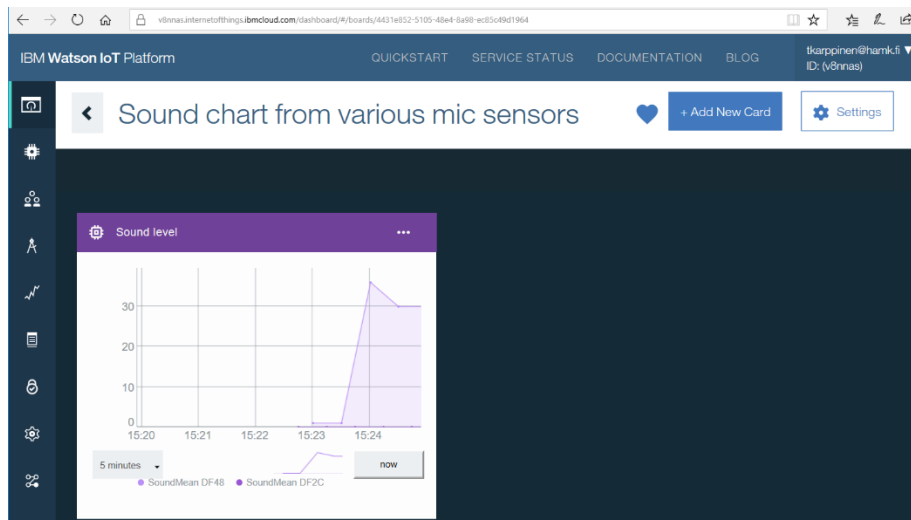


Fig. 7.5 In the user view in the board there are sensor values from two different sensor devices.

If you would like to see the metadata connected to each device you need to go back to the device view. Please select on left frame the second symbol from top, the symbol for devices.

Please select the device and further Device Information, View Metadata.