# IoT-Dashboard with ThingsBoard and mioty

Create your own IoT-Dashboard to display your sensor values in a simple and clear way.

## Components and supplies

 AVA gateway

 WEPTECH Indoor Humidity- & Temperature Sensor

## Apps and online services

www.thingsboard.io

## About this project

### The Story

In this project you will learn how to create your own IoT-Dashboard to display your sensor values in a simple and clear way. The topic is explained using a temperature sensor, but it is also possible with all other sensors.

### What you need

- ThingsBoard cloud maker account. This is free for the first few months. It is also possible to do an on-premise installation for free.
- mioty base station to receive your messages. e.g. AVA gateway from Weptech.
- External or internal service and application center with MQTT output. This is already included in the AVA gateway.
- mioty sensor e.g. temperature sensor.
- Any linux based pc to interconnect the mioty MQTT output with ThingsBoard. This can run directly on the mioty base station, if you have the login information.

## Preparation

Please get familiar with ThingsBoard by following the getting started guide. This mioty guide will take a similar approach with MQTT and `mosquitto-clients`.
https://thingsboard.io/docs/getting-started-guides/helloworld-pe/?connectdevice=mqtt-linux

In a mioty system, each sensor needs to be registered in the application center. Please also use the correct blueprint so that the payload is automatically parsed. If you haven't done this before, you can take a look at the "base station setup and connecting end-point" guide.

## MQTT reception of mioty telegrams

As a first step we collect the mioty telegrams via MQTT from the mioty application center. This is done with a simple python script. You can find it in the code section of this guide. Paho is used as python client. It can be installed with

```
pip install paho-mqtt
```

After inserting the correct server ip (e.g. localhost or ava.fritz.box) the script can be started with

```
python3 mqtt_client_example.py
```

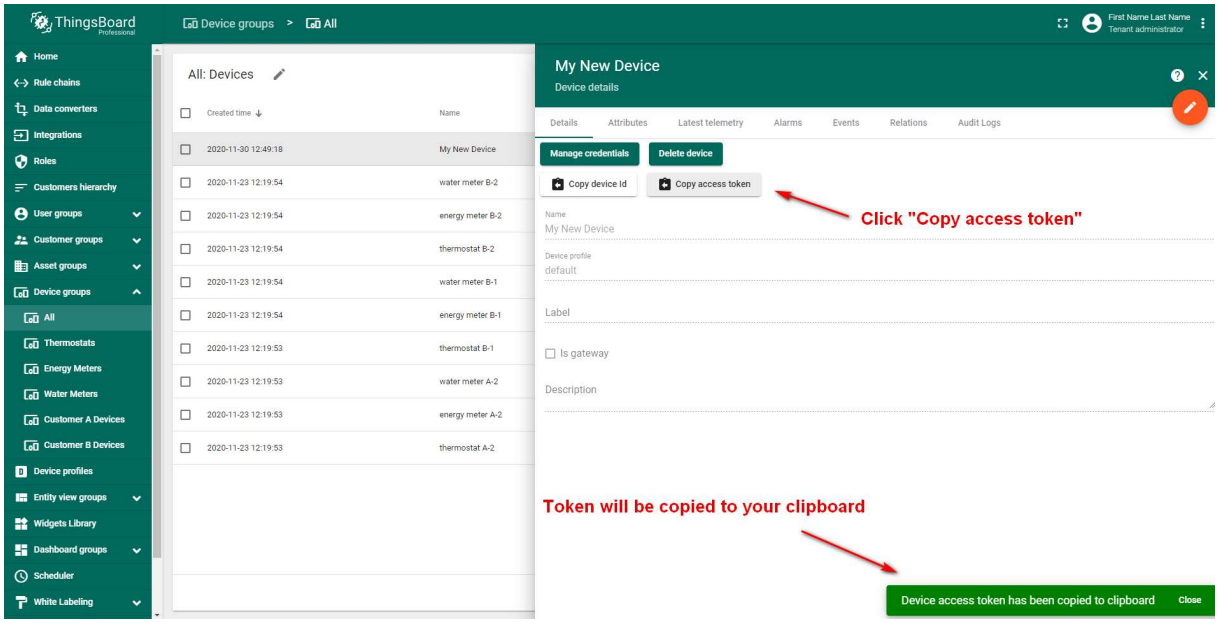
The output should look similar to this:

```
received topic: mioty/00-00-00-00-00-00-00-00/70-b3-d5-67-70-11-00-89/uplink
   b'{"baseStations":[{"bsEui":8121069422560411673,"rssi":-95.06800842285156,"rxTime":1642167
405003916373,"snr":17.526023864746094}],"cnt":51968,"components":{"battery":{"unit":"%","valu
e":100},"humidity":{"unit":"%rel","value":60},"pressure":{"unit":"hPa","value":1000.5},"tempe
rature":{"unit":"\xc2\xb0C","value":9.450000000000045}},"data":[5,100,11,10,60,39,21],"format
":0,"meta":{"name":"Weather","vendor":"Fraunhofer Mioty Demonstrator"},"typeEui":812106919331
7515269}'
   baseStations:
      bsEui: 0x70b3d59cd0000019 ,rssi:  -95.1 dBm, snr: 17.5 dB
   Endpoint Type:
      typeEui: 0x70b3d567700f0005
      name : Weather
      vendor : Fraunhofer Mioty Demonstrator
   Endpoint Data:
      raw: [5, 100, 11, 10, 60, 39, 21]
      battery : 100 %
      humidity : 60 %rel
      pressure : 1000.5 hPa
      temperature : 9.450000000000045 °C
```

Troubleshooting:
If only raw endpoint data is displayed, please make sure that the Blueprint is configured correctly.

## Understanding mioty eui and ThingsBoard tokens

In the mioty ecosystem every device has its own eui. It is an official IEEE-EUI-64 and therefore unique. On the ThingsBoard every device has an id, but the access token is more important . New telemetry values can be uploaded with this token.




## Publishing messages to ThingsBoard

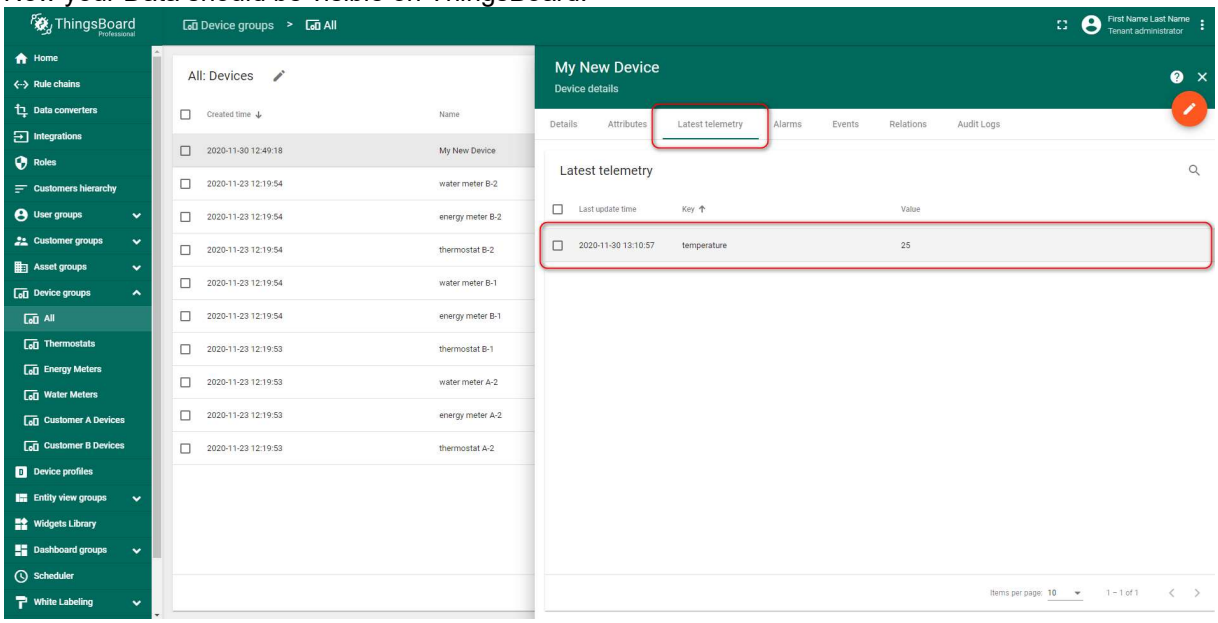
The second step is now to push the mioty telegrams into the ThingsBoard with the second python script in the code section. Please insert in the top section your euiTokenPair. The eui can be found on your device or in the received topic message. If everything works correct it should look like:

```
received topic: mioty/00-00-00-00-00-00-00-00/70-b3-d5-67-70-11-00-89/uplink
   b'{"baseStations":[{"bsEui":8121069422560411673,"rssi":-87.33042907714844,"rxTime":1642229
204032726279,"snr":22.361068725585938}],"cnt":52036,"components":{"battery":{"unit":"%","valu
e":100},"humidity":{"unit":"%rel","value":61},"pressure":{"unit":"hPa","value":998.2},"temper
ature":{"unit":"\xc2\xb0C","value":4.150000000000034}},"data":[5,100,10,213,61,38,254],"forma
t":0,"meta":{"name":"Weather","vendor":"Fraunhofer Mioty Demonstrator"},"typeEui":81210691933
17515269}'
Publish to thingserver
mosquitto_pub -d -q 1 -h "mqtt.thingsboard.cloud" -p "1883" -t "v1/devices/me/telemetry" -u "
NIA4efm4rrxSIqv    d" -m {"battery":100,"humidity":61,"pressure":998.2,"temperature":4.150000
000000034}
Client mosq-Li1yc0dXFc1YudaJcH sending CONNECT
Client mosq-Li1yc0dXFc1YudaJcH received CONNACK (0)
Client mosq-Li1yc0dXFc1YudaJcH sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ..
. (70 bytes))
Client mosq-Li1yc0dXFc1YudaJcH received PUBACK (Mid: 1, RC:0)
Client mosq-Li1yc0dXFc1YudaJcH sending DISCONNECT
```

After receiving a MQTT message from the mioty application centerthe script searches for a valid euiTokenPair and parses all fields in the blueprint and assembles the correct format for ThingsBoard telemetry data string. After this the program mosquito is called and it executes a single publish. This includes a CONNECT operation followed by a CONNACK from ThingsBoard. Afterwards the message gets PUBLISHed and ThingsBoard acknowledges it with PUBACK. Then the connection is closed.

If you have a mioty device that transmits a message every few minutes, it is totally fine to establish a new connection every time. If you have a very active device, you could consider rewriting the code to have a stable connection to ThingsBoard.

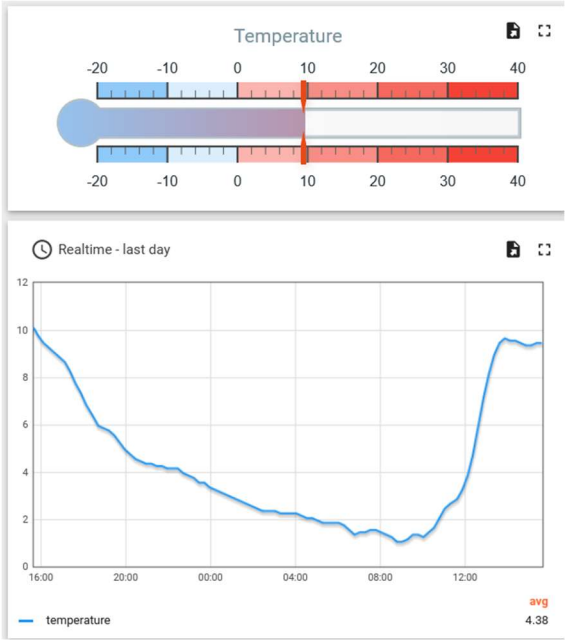
Now your Data should be visible on ThingsBoard.




Troubleshooting:
If you get now “Publish to ThingsServer” line. Please make sure the eui in the euiTokenPair has the same formatting as in the first line.
If you receive now CONNACK the access is granted over the access token from ThingsServer. Please check your Token.

## Create your own IoT-Dashboard

ThingsBoard offers many widgets to create fancy dashboards. If you have followed the getting started guide from ThingsBoard you should already have a first dashboard. Here is an example:




## Code

MQTT reception of mioty telegrams

```
#!/usr/bin/python3
from __future__ import print_function
import paho.mqtt.client as mqtt         # pip install paho-mqtt
import ssl
import json

# mqtt configuration
server = "localhost" #"ava.fritz.box"
port = 1883
path = 'mioty/+/+/uplink'

def on_message(client, userdata, message):
    print("received topic:", message.topic)
    print("  ", message.payload)
    fields = json.loads(message.payload.decode("utf-8"))
    print("   baseStations:")
    for i in fields['baseStations']:
        print ("      bsEui:", hex(i["bsEui"]), ",rssi: ",round(i["rssi"],1),
"dBm, snr:", round(i["snr"],1), "dB")

    if(fields['typeEui'] != 0):
        print("   Endpoint Type:")
        print("      typeEui:",hex(fields['typeEui']))
    if(isinstance(fields['meta'], dict)):
        for x in  fields['meta']:
            print("     ", x, ":", fields['meta'][x])

    print("   Endpoint Data:")
    print("      raw:", fields['data'])
    valueDict = fields['components']
    if(isinstance(valueDict, dict)):
        for x in valueDict:
            print("     ",x,":", valueDict[x]["value"],valueDict[x]["unit"])

    print("\n\n")


def on_connect(client,userdata,flag,rc):
```

```
    print("connect")
    client.subscribe(path)


client = mqtt.Client()
client.on_message = on_message
client.on_connect = on_connect
client.connect(server, port, 60)

try:
    client.loop_forever()

except KeyboardInterrupt:
    client.loop_stop()
    client.disconnect()
```

Publishing messages to ThingsBoard

```
#!/usr/bin/python3
from __future__ import print_function
import paho.mqtt.client as mqtt         # pip install paho-mqtt
import ssl
import json
import os

# Put in your Mioty-Eui and Thingsboard Token
# Format: [["Eui1","Token1"],["Eui2","Token2"],["Eui3","Token3"]]
euiTokenPair = [
    ["70-b3-d5-67-70-11-xx-xx","xxxxxxxxxxxxxxxxxxx"],
    ["70-b3-d5-67-70-11-xx-xx","xxxxxxxxxxxxxxxxxxx"]
]

# mqtt configuration
miotyServer = "localhost" #"ava.fritz.box"
miotyPort = 1883
path = 'mioty/+/+/uplink'

def on_message(client, userdata, message):
    #-- new mioty telegram received
    print("received topic:", message.topic)
    print("  ", message.payload)
    fields = json.loads(message.payload.decode("utf-8"))

    #-- search for correct Eui Token Pair
    for sens in euiTokenPair:
        sensorEui = sens[0]
        sensorToken = sens[1]
        if(message.topic.find(sensorEui)>=0):
            print("Publish to thingserver")

            #-- Assamble Data String
            first = True
            valueDict = fields['components']
            msg = ""
            for x in valueDict:
                if(first == False):
                    msg += ","
                msg += "\"%s\":%s"%(x,str(valueDict[x]["value"]))
                first=False

            #Publish Data to thingsboard via mosquitto
            mosCmd = "\
mosquitto_pub -d -q 1 \
-h \"mqtt.thingsboard.cloud\" -p \"1883\" \
-t \"v1/devices/me/telemetry\" -u \"%s\" -m {%s}"%(sensorToken,msg)
            print(mosCmd)
            os.system(mosCmd)
```

```
            print("\n\n")

def on_connect(client,userdata,flag,rc):
    print("connect")
    miotyClient.subscribe(path)

miotyClient = mqtt.Client("mioty")
miotyClient.on_message = on_message
miotyClient.on_connect = on_connect
miotyClient.connect(miotyServer, miotyPort, 60)

try:
    miotyClient.loop_forever()
except KeyboardInterrupt:
    miotyClient.loop_stop()
    miotyClient.disconnect()
```