IoT-Growhouse

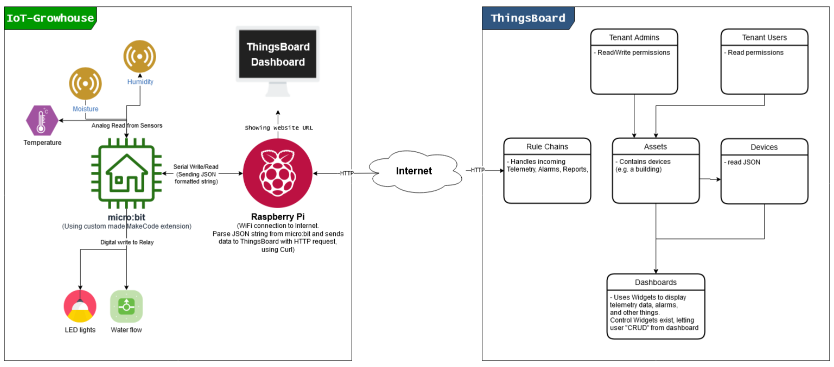
IT-Gården

Introduction

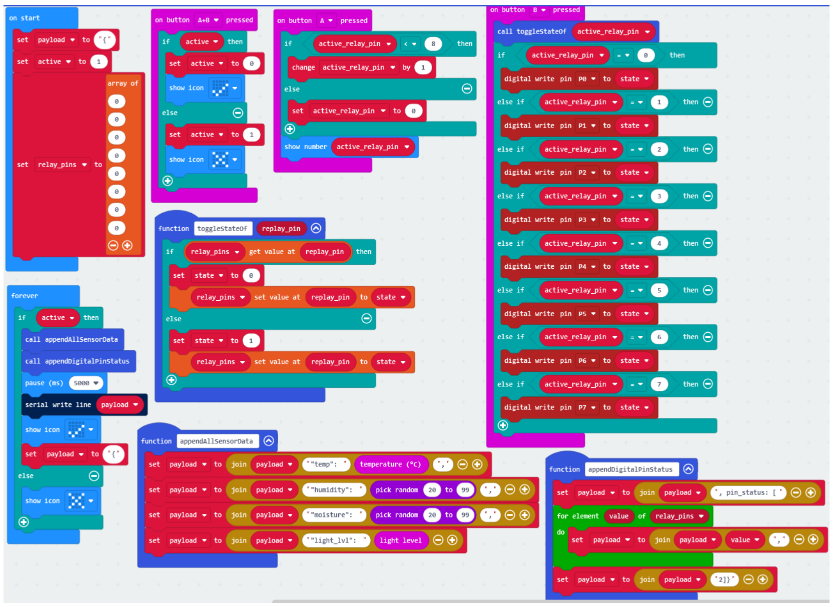
Here we give an overview of all devices and places used within the project.

General Overview

The system architecture, mapping all devices/places and their usage, looks the following:


IoT-Growhouse
Humidity
Moisture
Temperature
Analog Read from Sensors
ThingsBoard Dashboard
Showing website URL
micro:bit
(Using custom made MakeCode extension)
Serial Write/Read (Sending JSON formatted string)
Raspberry Pi
(WiFi connection to Internet. Parse JSON string from micro:bit and sends data to ThingsBoard with HTTP request, using Curl)
Digital write to Relay
LED lights
Water flow
HTTP
Internet
HTTP
ThingsBoard
Tenant Admins
- Read/Write permissions
Tenant Users
- Read permissions
Rule Chains
- Handles incoming Telemetry, Alarms, Reports,
Assets
- Contains devices (e.g. a building)
Devices
- read JSON
Dashboards
- Uses Widgets to display telemetry data, alarms, and other things. Control Widgets exist, letting user "CRUD" from dashboard

Micro:bit code



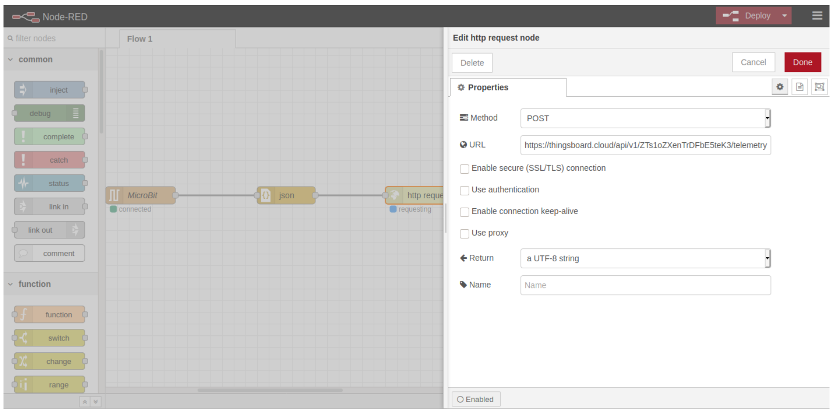
```
on start
    set payload to "{"
    set active to 1
    set relay_pins to array of 0 0 0 0 0 0 0 0

forever
    if active then
        call appendAllSensorData
        call appendDigitalPinStatus
        pause (ms) 5000
        serial write line payload
        show icon
        set payload to "{"
    else
        show icon

on button A+B pressed
    if active then
        set active to 0
        show icon
    else
        set active to 1
        show icon

on button A pressed
    if active_relay_pin < 8 then
        change active_relay_pin by 1
    else
        set active_relay_pin to 0
    show number active_relay_pin

function toggleStateOf replay_pin
    if relay_pins get value at replay_pin then
        set state to 0
        relay_pins set value at replay_pin to state
    else
        set state to 1
        relay_pins set value at replay_pin to state

function appendAllSensorData
    set payload to join payload ""temp": " temperature (°C) ","
    set payload to join payload ""humidity": " pick random 20 to 99 ","
    set payload to join payload ""moisture": " pick random 20 to 99 ","
    set payload to join payload ""light_lvl": " light level

on button B pressed
    call toggleStateOf active_relay_pin
    if active_relay_pin = 0 then
        digital write pin P0 to state
    else if active_relay_pin = 1 then
        digital write pin P1 to state
    else if active_relay_pin = 2 then
        digital write pin P2 to state
    else if active_relay_pin = 3 then
        digital write pin P3 to state
    else if active_relay_pin = 4 then
        digital write pin P4 to state
    else if active_relay_pin = 5 then
        digital write pin P5 to state
    else if active_relay_pin = 6 then
        digital write pin P6 to state
    else if active_relay_pin = 7 then
        digital write pin P7 to state

function appendDigitalPinStatus
    set payload to join payload ", pin_status: ["
    for element value of relay_pins
    do
        set payload to join payload value ","
    set payload to join payload "2]}"
```

Raspberry Pi

This device **is optional**, and can be skipped if setting up the the micro:bit to send POST requests through a ESP8226 module. ThingsBoard also have many other integration options, that can be used to connect devices.

It is connected to WiFi and running a Node-RED server:

1. Receive string from microbit
2. Parse string to JSON
3. Send JSON to ThingsBoard, using HTTP POST request
    - URL (includes device token): https://thingsboard.cloud/api/v1/ZTs1oZXenTrDFbE5teK3/telemetry



Edit http request node

Method: POST

URL: https://thingsboard.cloud/api/v1/ZTs1oZXenTrDFbE5teK3/telemetry

☐ Enable secure (SSL/TLS) connection

☐ Use authentication

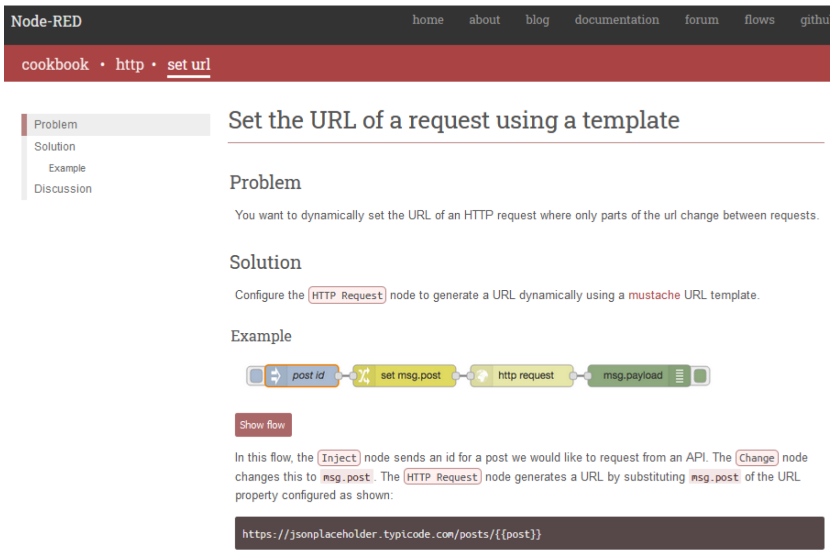
☐ Enable connection keep-alive

☐ Use proxy

Return: a UTF-8 string

Name: Name

We can add extra functionality to our Node-RED server, that makes it inject the device id given to it by our microbit into the request URL. By doing so we can now use our microbit that is connected to the Raspberry Pi as a gateway for other devices (most likely other microbits). These other microbits can use bluetooth (BT) to send in data to ThingsBoard, through our microbit gateway.



Set the URL of a request using a template

Problem

You want to dynamically set the URL of an HTTP request where only parts of the url change between requests.

Solution

Configure the `HTTP Request` node to generate a URL dynamically using a mustache URL template.

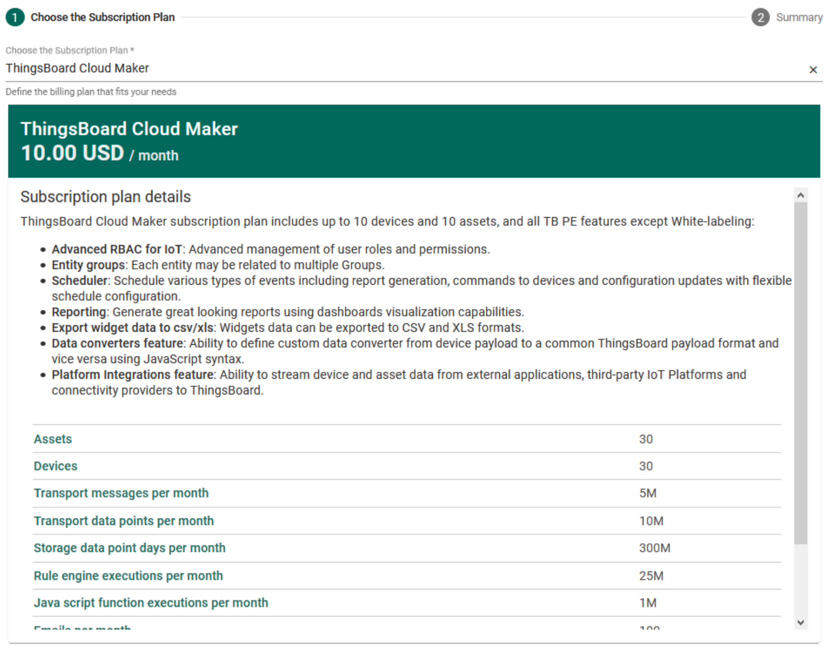
Example

In this flow, the `Inject` node sends an id for a post we would like to request from an API. The `Change` node changes this to `msg.post`. The `HTTP Request` node generates a URL by substituting `msg.post` of the URL property configured as shown:

```
https://jsonplaceholder.typicode.com/posts/{{post}}
```

ThingsBoard

It is possible to use ThingsBoard (TB)with the free Community Edition (CE) for our project, as we only need to use the HTTP API and not one of the available integrations that come with the Professional edition (PE). To simpliy and speed things up, we are using PE because it makes the ThingsBoard cloud installation available to us; saving us the job of setting up and configuring our own TB server.



Choose the Subscription Plan *
ThingsBoard Cloud Maker
Define the billing plan that fits your needs

ThingsBoard Cloud Maker
10.00 USD / month

Subscription plan details

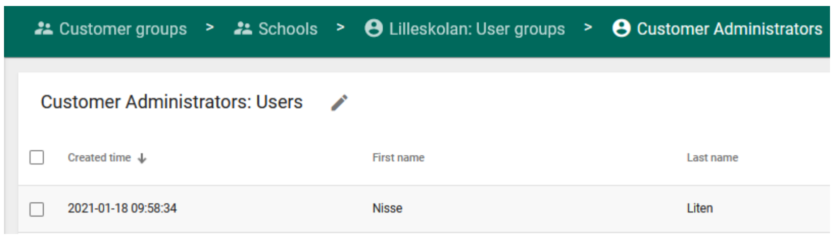
ThingsBoard Cloud Maker subscription plan includes up to 10 devices and 10 assets, and all TB PE features except White-labeling:

- **Advanced RBAC for IoT**: Advanced management of user roles and permissions.
- **Entity groups**: Each entity may be related to multiple Groups.
- **Scheduler**: Schedule various types of events including report generation, commands to devices and configuration updates with flexible schedule configuration.
- **Reporting**: Generate great looking reports using dashboards visualization capabilities.
- **Export widget data to csv/xls**: Widgets data can be exported to CSV and XLS formats.
- **Data converters feature**: Ability to define custom data converter from device payload to a common ThingsBoard payload format and vice versa using JavaScript syntax.
- **Platform Integrations feature**: Ability to stream device and asset data from external applications, third-party IoT Platforms and connectivity providers to ThingsBoard.

Assets	30
Devices	30
Transport messages per month	5M
Transport data points per month	10M
Storage data point days per month	300M
Rule engine executions per month	25M
Java script function executions per month	1M

User Administration

Documentation on how a system administration can create and handle users is found here. In our case we will create a Costumer for each school. Each Costumer have their own set of users, assets, devices, dashboards, and so on. We create on "Tenant Administrator" for each school, that have the authorization to create and do everything needed to fully utilize the ThingsBoard platform. Scripts can be used to automate the creation of costomers, and set up default things we wish them to have at start.


Customer groups > Schools > Lilleskolan: User groups > Customer Administrators

Customer Administrators: Users

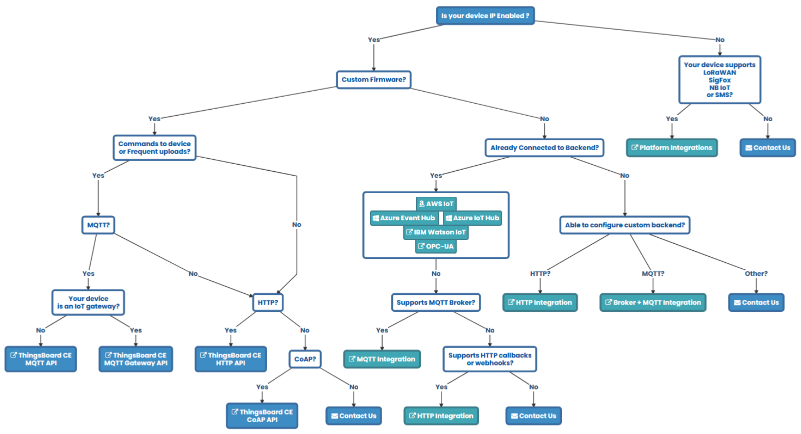
Created time ↓	First name	Last name
2021-01-18 09:58:34	Nisse	Liten

Dashboards can be created through the GUI or code. Here are some example python scripts that can be used to quickly set up the default Dashboard for our Growhouse.

- Generate a dashboard (from file) using a script

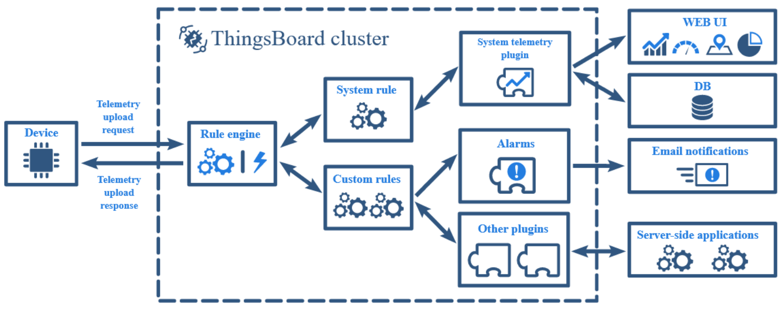
```
#       Copyright 2020. ThingsBoard
#   #
#       Licensed under the Apache License, Version 2.0 (the "License");
#       you may not use this file except in compliance with the License.
#       You may obtain a copy of the License at
#   #
#          http://www.apache.org/licenses/LICENSE-2.0
#  #
#      Unless required by applicable law or agreed to in writing, software
#      distributed under the License is distributed on an "AS IS" BASIS,
#      WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#      See the License for the specific language governing permissions and
#      limitations under the License.
#

import logging
from json import load
# Importing models and REST client class from Professional Edition version
from tb_rest_client.rest_client_pe import *
from tb_rest_client.rest import ApiException


logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s - %(levelname)s - %(module)s -
%(lineno)d - %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S')


# ThingsBoard REST API URL
url = "http://localhost:8080"

# Default Tenant Administrator credentials
username = "tenant@thingsboard.org"
password = "tenant"


# Creating the REST client object with context manager to get auto token refresh
with RestClientPE(base_url=url) as rest_client:
    try:
        # Auth with credentials
        rest_client.login(username=username, password=password)

        # Getting current user
        current_user = rest_client.get_user()

        # Creating Dashboard Group on the Tenant Level
        shared_dashboards_group = EntityGroup(name="Shared Dashboards", type="DASHBOAI
        shared_dashboards_group = rest_client.save_entity_group(shared_dashboards_grou

        # Loading Dashboard from file
        dashboard_json = None
        with open("watermeters.json", "r") as dashboard_file:
            dashboard_json = load(dashboard_file)
        dashboard = Dashboard(title=dashboard_json["title"], configuration=dashboard_j
        dashboard = rest_client.save_dashboard(dashboard)
        # Adding Dashboard to the Shared Dashboards Group
        rest_client.add_entities_to_entity_group(shared_dashboards_group.id, [dashboar

        # Creating Customer 1
        customer1 = Customer(title="Customer 1")
        customer1 = rest_client.save_customer(customer1)

        # Creating Device
        device = Device(name="WaterMeter1", type="waterMeter")
        device = rest_client.save_device(device)

        # Fetching automatically created "Customer Administrators" Group.
        customer1_administrators = rest_client.get_entity_group_info_by_owner_and_name

        # Creating Read-Only Role
        read_only_role = Role(name="Read-Only", permissions=['READ', 'READ_ATTRIBUTES
        read_only_role = rest_client.save_role(read_only_role)

        # Assigning Shared Dashboards to the Customer 1 Administrators
        tenant_id = current_user.tenant_id
        group_permission = GroupPermission(role_id=read_only_role.id,
                                           name="Read Only Permission",
                                           is_public=False,
                                           user_group_id=customer1_administrators.id,
                                           tenant_id=tenant_id,
                                           entity_group_id=shared_dashboards_group.id
                                           entity_group_type=shared_dashboards_group.t
        group_permission = rest_client.save_group_permission(group_permission)

        # Creating User for Customer 1 with default dashboard from Tenant "Shared Dash
        user_email = "user@thingsboard.org"
        user_password = "secret"
        additional_info = {
            "defaultDashboardId": dashboard.id.id,
            "defaultDashboardFullscreen": False
        }
        user = User(authority="CUSTOMER_USER",
                    customer_id=customer1.id,
                    email=user_email,
                    additional_info=additional_info)
        user = rest_client.save_user(user, send_activation_mail=False)
        rest_client.activate_user(user.id, user_password)

        rest_client.add_entities_to_entity_group(customer1_administrators.id, [user.id

    except ApiException as e:
        logging.exception(e)
```

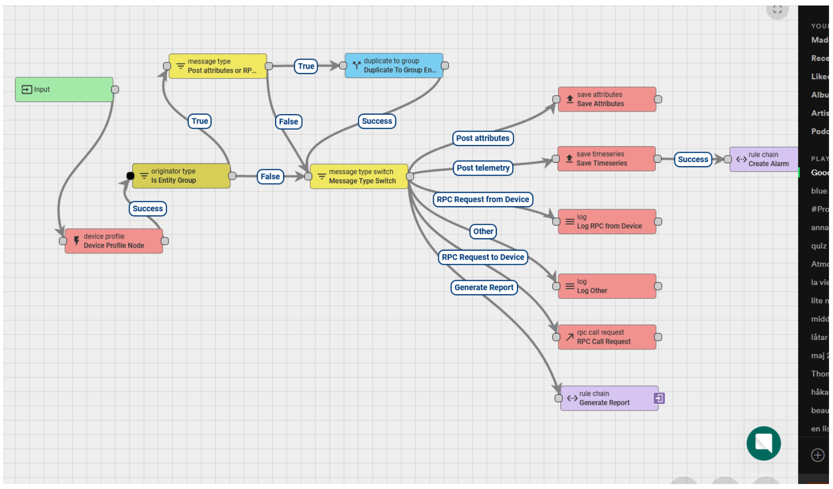
Connection options



Data handling & Rule Chain

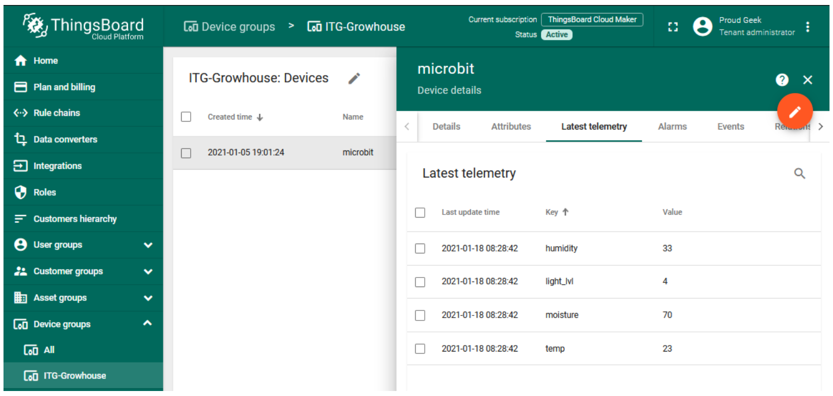
Telemetry data processing





Device

We have created one device, that represents the microbit mounted on our IoT-Growhouse. We send all data from the growhouse using a single HTTP POST request. When the request has gone through the Root rule chain, where data from devices gets saved, we can see it on our device. Then we can connect and display values of each key (in the JSON data the device receives) in Dashboard Widgets.

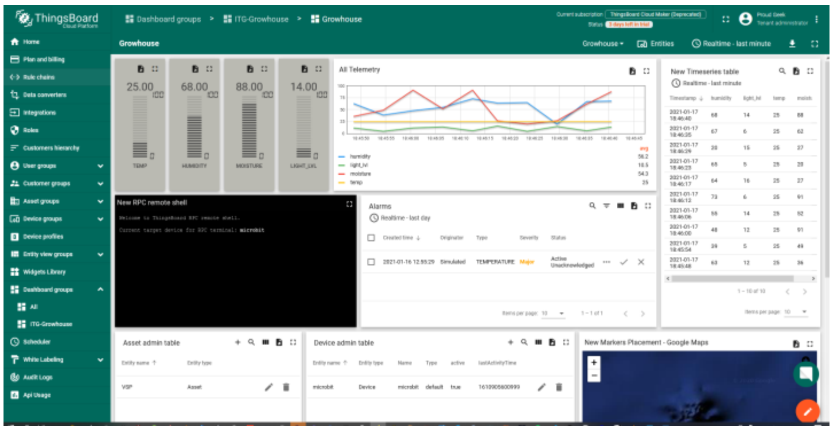


Created time	Name
2021-01-05 19:01:24	microbit

Last update time	Key	Value
2021-01-18 08:28:42	humidity	33
2021-01-18 08:28:42	light_lvl	4
2021-01-18 08:28:42	moisture	70
2021-01-18 08:28:42	temp	23

Dashboard

Here is an example dashboard, showing sensor data received from our growhouse:



Downlink

- TODO: We are currently not doing this

MakeCode (micro:bit)

- **Relay Labels**
  - 0:
  - 1:
- **Pins**

- 0:
- 1:
- 2:
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:
- 10:
- 11:
- 12:
- 13:
- 14:
- 15: