PANEURÓPSKA VYSOKÁ ŠKOLA

Fakulta informatiky

Semestrálna práca

Technológie inteligentného prostredia

PhDr. Ing. Mgr. Miroslav Reiter, DBA, MSC

Technológie inteligentného prostredia

Ing. Erich Stark, PhD.

22.04.2022

1. Charakteristika a cieľ projektu

V mojej semestrálne práci zaoberám Internetom vecí (IoT), ktoré mi umožňuje jednoduché monitorovanie a získavanie údajov zo senzorov cez internet. Potreba vysokokvalitných údajov sa zvyšuje do tej miery, že je potrebný systém monitorovania a získavania údajov v reálnom čase, ako napríklad inteligentné mesto alebo telediagnostika v medicínskych oblastiach. Preto je na vyriešenie týchto problémov potrebný vhodný komunikačný protokol. V poslednej dobe výskumníci vyvinuli množstvo komunikačných protokolov pre IoT, z ktorých každý má výhody a nevýhody. Táto semestrálna práca navrhuje využitie MQTT ako komunikačného protokolu, ktorý je jedným z dátových komunikačných protokolov pre IoT. V práca som vytvoril simulátor pre snímače teploty, vlhkosti a osvetlenia, pretože fyzikálne parametre sú často potrebné ako parametre stavu prostredia. Zber údajov prebiehal v reálnom čase a ukladal sa do databázy časových údajov InfluxDB. Prácu dopĺňa aj webové grafické používateľské rozhranie (Dashboard v Grafane) pre online monitorovanie. Výsledkom tejto práce je zlepšenie kvality a spoľahlivosti údajov pomocou protokolu MQTT.

Cieľom tohto projektu je naplno využiť potenciál internetu vecí, konkrétne protokolu MQTT s message brokerom Mosquitto, ktoré sa vo veľkej miere využívajú v internete vecí. Tento projekt využíva nasledujúce technológie

1. **MQTT** (Mosquitto): pre dátovú komunikáciu
2. **InfluxDB**: pre zber/ukladanie dát do databázového systému časových radov
3. **Node-Red**: vizuálne programovanie uzlov, spracovanie údajov a ich následne ukladanie do databázy
4. **Grafana**: na vizualizáciu údajov uložených v databáze v prehľadom dashboarde (Menovky a Čiarové grafy)

Všetky tieto technológie ďalej popisujem v semestrálnej práci.

2. Node.js (simulátor senzorov)

V tomto projekte sme použili Node.js na simuláciu rôznych hodnôt senzorov pomocou vybraných modulov. Hodnoty boli generované pseudonáhodne pomocou zabudované objektu Math a jeho funkcie random, pričom boli udržiavané v špecifickom rozsahu pre každý senzor a tiež s ohľadom na čas dňa. Berieme do úvahy, že tieto senzory sa nachádzajú v byte.

Napríklad počas dňa (06:00 až 19:59) boli všetky 3 hodnoty snímačov sú vybrané z nasledujúcich rozsahov:

1. **Teplota (Temperature)**: 18 – 21 °C
2. **Vlhkosť (Humidity)**: 40 – 50 %
3. **Intenzita svetla/svetelný tok (Light intensity/lumins)**: 0 – 350 lm

V noci (20:00 až 05:59) sa hodnoty senzorov vyberajú náhodne z týchto rozsahov

1. Teplota: 21 – 24 °C
2. Vlhkosť: 50 – 60 %
3. Intenzita svetla: 350 – 700 lm

V mojej aplikácií využívam balíčky/knižnice mqtt pre komunikáciu s brokerom v našom prípade mosquito, express pre routovanie aplikačných endpointov (URI) a reakcie na klientské požiadavky. Spracovanie cez get metódy pre teplo, vlhkosť a svetlo. Zároveň aplikácia na porte **5000** (NodeJS) "počúva" požiadavky zodpovedajú zadaným routes a metódam, a keď zistí zhodu, zavolá špecifikovanú funkciu spätného volania. V aplikácii ďalej vytváram senzory pre teplo, vlhkosť a svetlo. Každý senzor má vygenerované ID pomocou modulu uuid.

app.js

```
const uuid = require('uuid');
const mqtt = require('mqtt')

var express = require("express");
var app = express();
var delay = 0.1;

var tempUUID = uuid.v4();
var humUUID = uuid.v4();
var luminUUID = uuid.v4();
```

package.json

```
{
  "name": "simulator",
  "version": "1.0.0",
  "description": "Sensor simulator",
  "main": "app.js",
  "scripts": {},
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3",
    "mqtt": "^4.3.7",
    "node-red": "^0.19.6",
    "uuid": "^8.3.2"
  }
}
```

Celý zdrojový kód v JavaScript je okomentovaný vid ukážka z metódy get pre spracovania teploty. Spracovania dátumu/času na vhodný formát. Na tento účel použitá metóda replace s regulárnym výrazom, kde sa chceme zbaviť Tčka a nahradiť ho medzerou a chceme sa zbaviť milisekúnd a nahradzujeme ich ničím (prázdnym reťazcom). Rovnako overujem či riešiť prípade, kedy je deň alebo noc podľa rozsahu hodín získame cez getter na hodiny. Na konci si vytvoríme slovník objekt typu json a pošleme ho. Odosielame páry (kľúč + hodnota): IDčko zariadia a jeho UUID pre teplotu, potom senzor a teplotu, hodnotu a hodnotu teploty, na konci datum a anonymný objekt dát, z ktorého vyberieme len čas. Toto isté opakujeme aj pre vlhkosť a svetelný tok s prípadmi cez deň a cez noc.

```
app.get("/temp", (req, res, next) => {
    // Chcem získať čas z dátumu
    var time = new Date().getHours();
    // Prípad, keď je deň
    if (time > 5 && time < 20) {
        // Vygeneruj náhodnú hodnotu medzi 18 a 21
        var temp = Math.floor(Math.random() * (21 - 18 + 1)) + 18;
        var datetime = new Date().toISOString();
        // Formát dátumu a času na YYYY-MM-DDTHH:mm:ss.sssZ
        datetime = datetime.replace(/T/, ' ').replace(/\..+/, '');
        // Vytvor json
        var json = {'deviceId': tempUUID, 'sensor': 'temperature', 'value': temp, 'date': timestamp};
        // Posli json
        res.send(json);
    }
    else{
    // Prípad, keď je noc
        // Vygeneruj náhodnú hodnotu medzi 21 a 24
        var temp = Math.floor(Math.random() * (24 - 21 + 1)) + 21;
        // Vytvor json
        var json = {'deviceId': tempUUID, 'sensor': 'temperature', 'value': temp, 'date': new Date().getTime()};
        // Posli json
        res.send(json);
    }
});
```

3. MQTT

Po vygenerovaní pseudonáhodných hodnôt pre každý senzor publikujeme tieto hodnoty brokerovi MQTT. Pri publikovaní ešte z JSON objektu spravíme String. Broker MQTT, ktorý sme použili v tomto projekte je open source broker **mosquitto**, ktorý využíva publish/subscribe model. Na pripojenie MQTT využívame port **1883** na localhoste.

```
var client = mqtt.connect('mqtt://localhost:1883');
// Publikuj na MQTT opakovane každú 1 sekundu
client.publish('sensor/temp', JSON.stringify(tempMsg));
client.publish('sensor/hum', JSON.stringify(humMsg));
client.publish('sensor/lumin', JSON.stringify(luminMsg));
console.log("Published to MQTT");
```

Do Linux sa dá sa nainštalovať pomocou príkazu

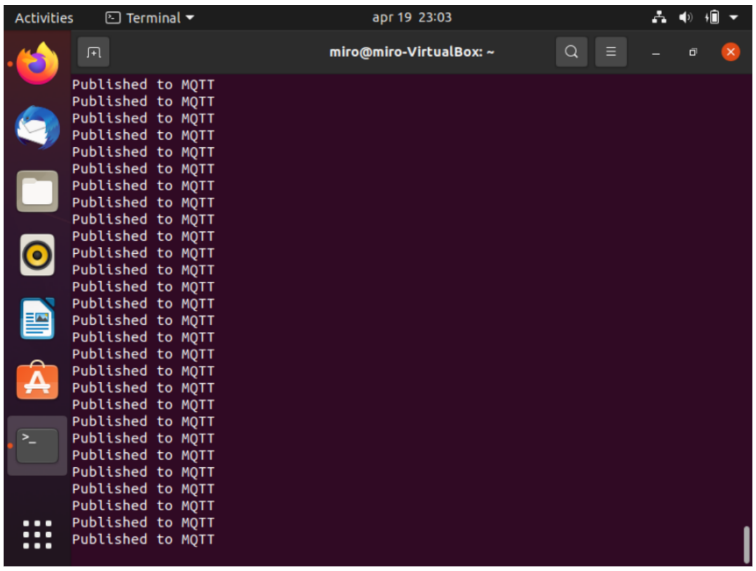
```
sudo apt install mosquitto
```

Po jeho nainštalovaní spustíme sprostredkovateľa MQTT pomocou tohto príkazu

```
mosquitto_sub -v -t „#“
```

V brokerovi MQTT sme použili zástupný znak, pretože pre každý senzor používame samostatné topics, ktoré sú nasledujúce:

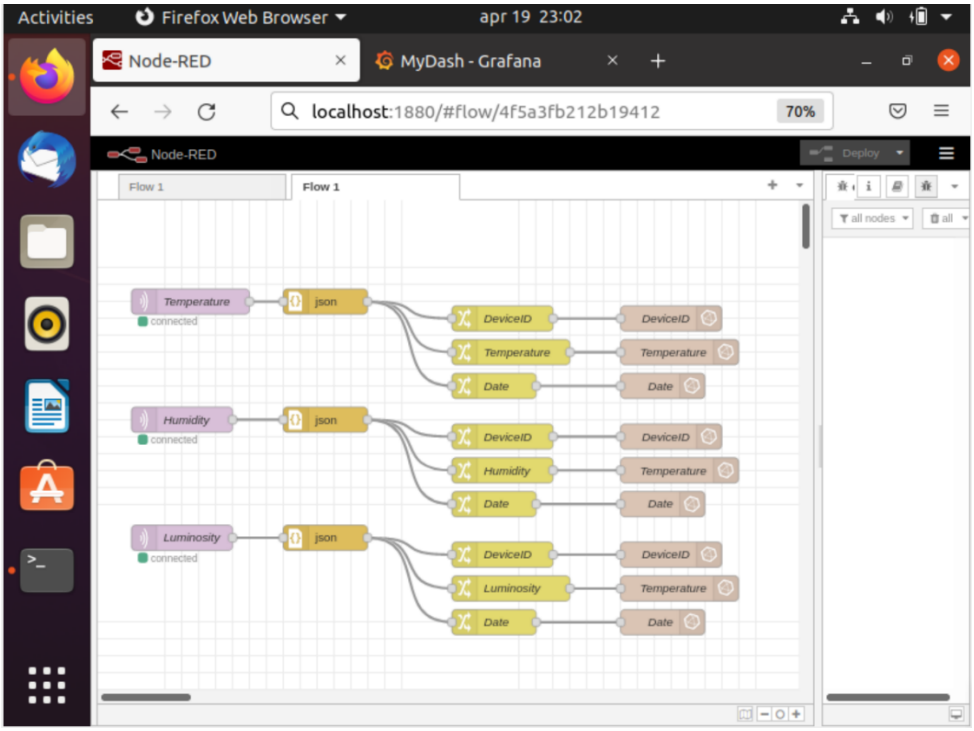
- sensor/temp/value
- sensor/temp/deviceid
- sensor/temp/date
- sensor/hum/value
- sensor/hum/deviceid
- sensor/hum/date
- sensor/lumin/value
- sensor/lumin/deviceid
- sensor/lumin/date



Zástupný znak („#“) v príkaze brokera MQTT v podstate akceptuje všetky topics a prepošle ich žiadajúcemu klientovi.

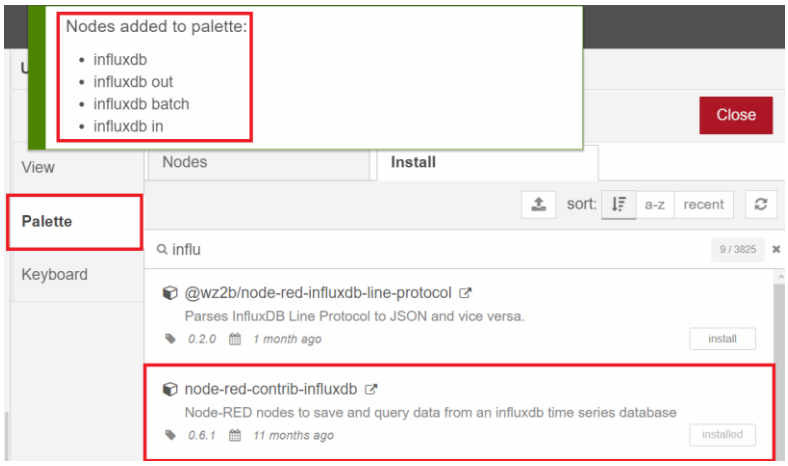
4. Node-RED

Node-RED je programátorsky nástroj slúžiaci na vizuálne programovanie spolu s na prepojením na hardvérových zariadení, rozhraní API a online služby. Pomocou uzlov (nodes) definujem jednotlivé kroky ako nám „tečú“ dáta. Node-Red som teda použil na prijatie údajov senzora od brokera MQTT a ich spracovanie pred ich odoslaním na uloženie do databázy InfluxDB. Node-Red mi beží na porte 1880. Tu vidíme diagram toku Node-Red.

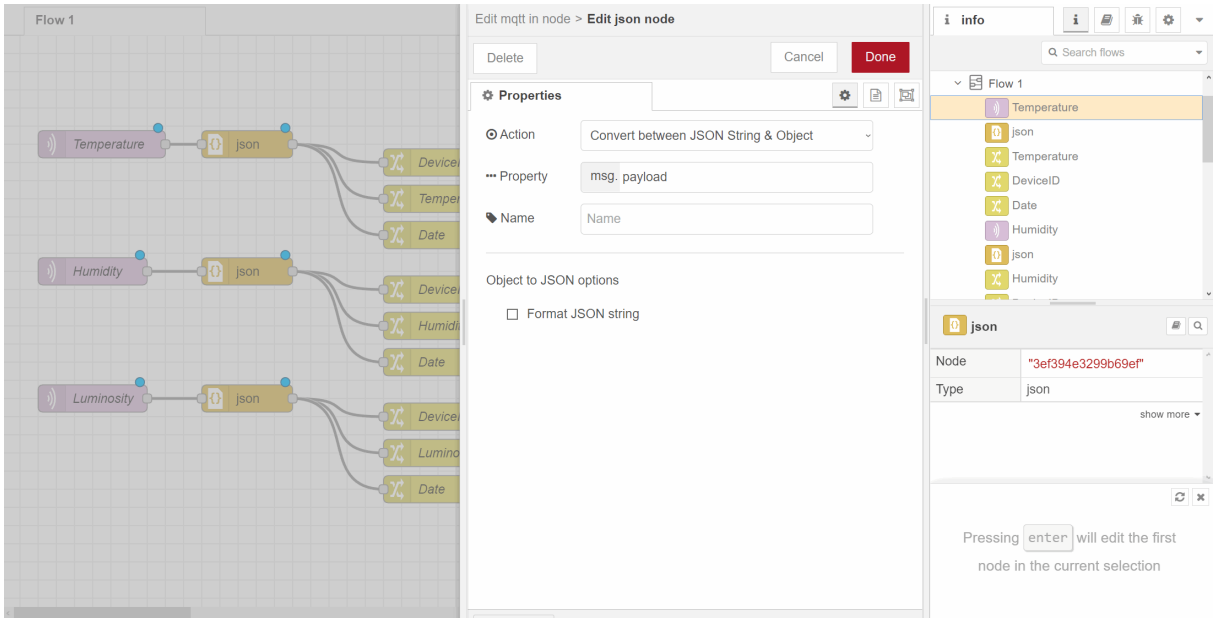


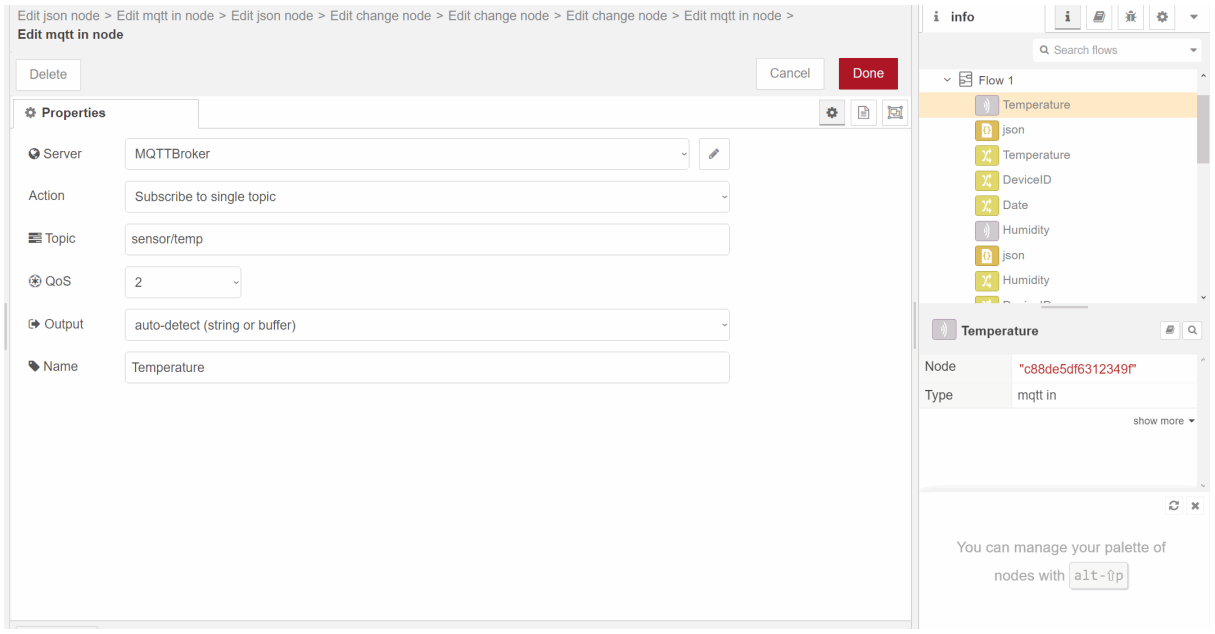
Môj flow tvoria uzly (nodes):

1. **mqtt in** (kategória network)
2. **json** (kategória parser)
3. **change** (kategória function)
4. **influxdb out** (kategória storage) - tie som musel doinštalovať cez paletu vid. nasledovný obrázok



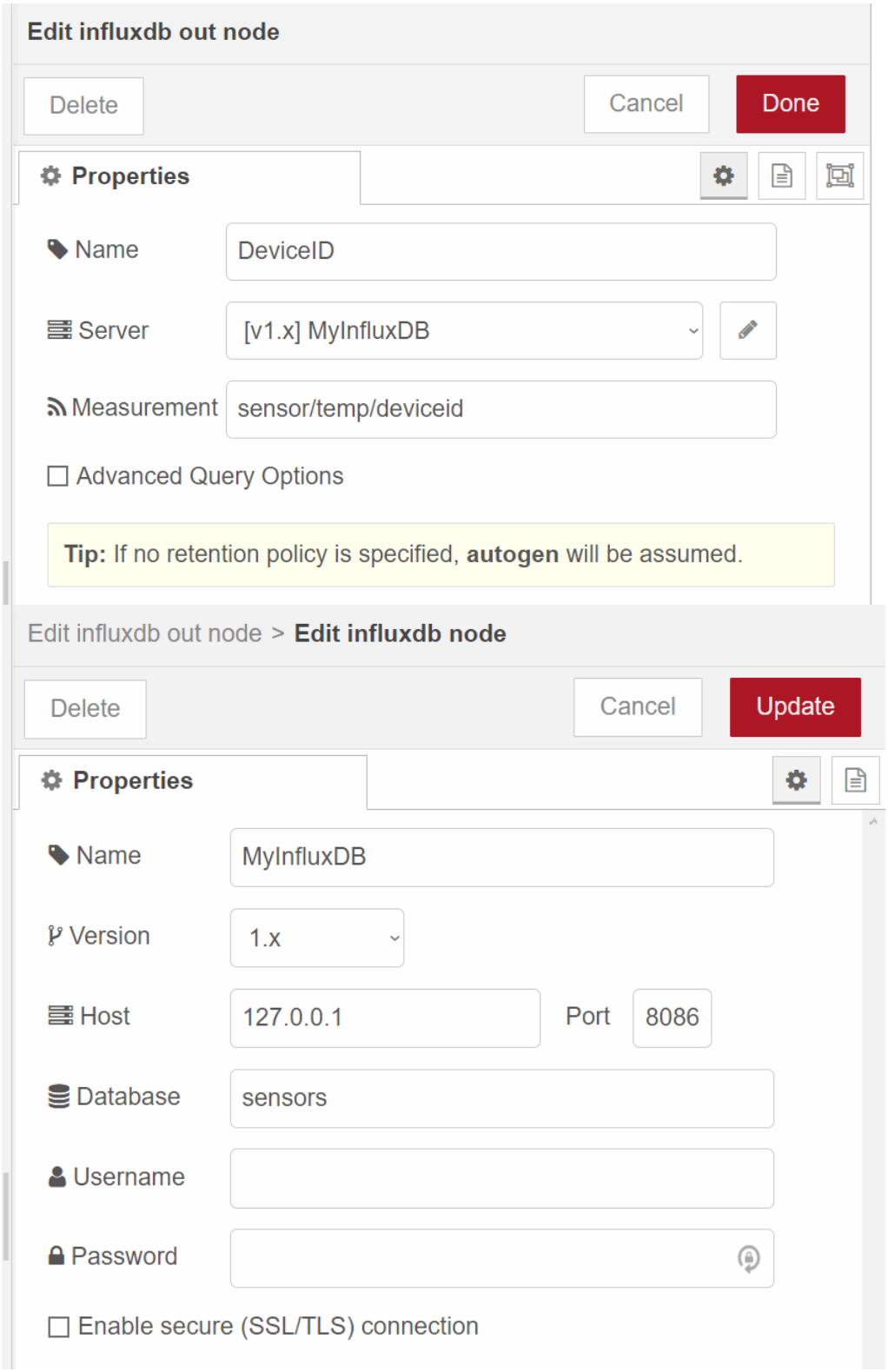
Flow funguje tak, že pomocou mqtt in uzla sa prihlásime na odber vybraných topics s Quality of Service nastavenou na úroveň 2 vid. druhý obrázok. To znamená, že správa je doručená len raz. QoS má najvyššiu úroveň.





Používame uzol json, pretože Node.js odosiela údaje brokerovi MQTT vo formáte JSON. Extrahujeme Json, aby sme získali DeviceID, hodnotu senzora a dátum a posielame ich do influxDB v príslušných meraniach (measurements).

Výstupné storage uzly pre ukladanie do influxDB s hodnotami generovanými simulátorom senzorov. Na obrázku príklad merania zo senzoru teploty do databázy sensors, ktorá nám opäť beží lokálne na porte 8086.



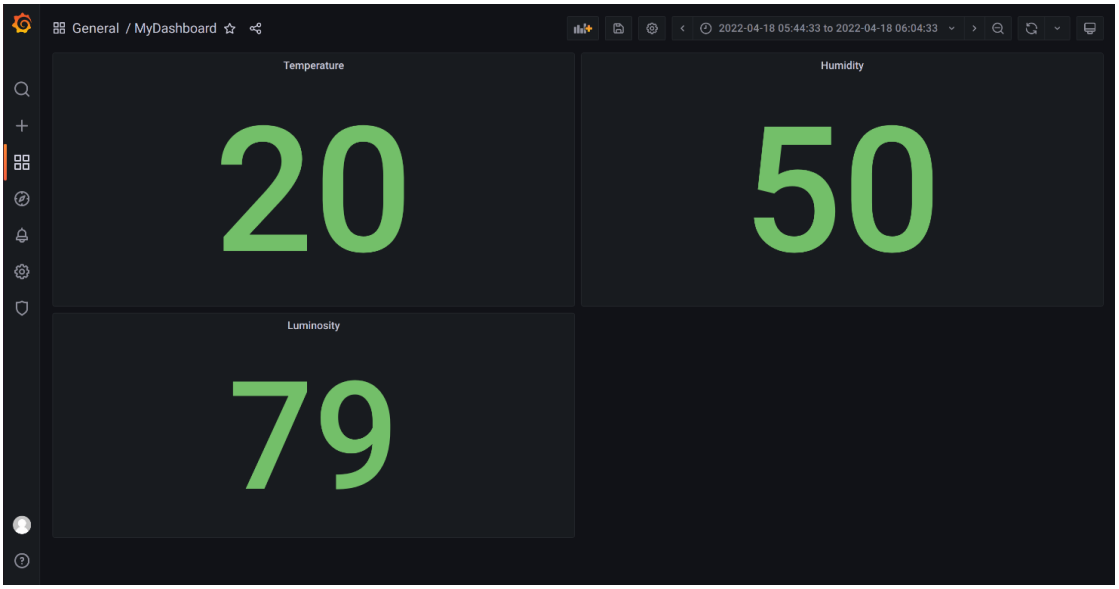
5. InfluxDB

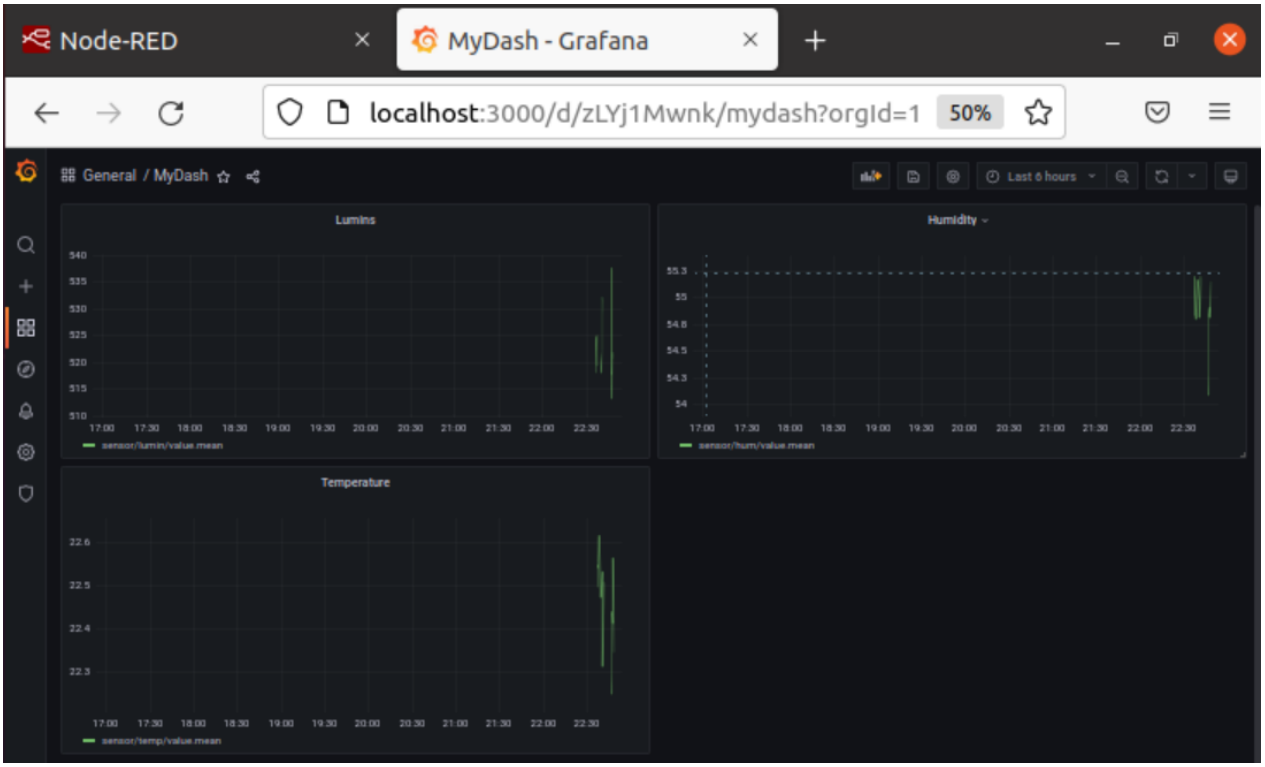
Na ukladanie údajov som použil InfluxDB. Mohol by sme použiť aj MySQL alebo nejaký iný rozšírenejší databázový systém, ale použil som InfluxDB, pretože je to lepšia voľba pre údaje z časových radov. InfluxDB je open-source databáza časových radov a priamo pracuje s časovou značkou (datetime/timestamps). InfluxDB nevyžaduje vytváranie stĺpcov tabuliek a ich dátových typov. Stačí len vytvoriť tabuľku a to je všetko. Databáza automaticky vytvorí stĺpce, keď dostane údaje. Ukážka z pripojenia na databázu InfluxDB na porte 8086 s výpisom všetkých meraní pre aktuálne aktívnu databázu:

```
Connected to http://localhost:8086 version 1.6.4
InfluxDB shell version: 1.6.4
> use sensors
Using database sensors
> show measurements
name: measurements
name
----
sensor/hum/date
sensor/hum/deviceid
sensor/hum/value
sensor/lumin/date
sensor/lumin/deviceid
sensor/lumin/value
sensor/temp/date
sensor/temp/deviceid
sensor/temp/value
>
```

6. Grafana

Grafanu som použil na vizualizáciu údajov zo senzorov. Grafana je veľmi praktický nástroj na vizualizáciu rôznych druhov údajov pomocou tabuliek, grafov, tabuliek, meradiel atď. Na vizualizáciu teploty, vlhkosti a intenzity svetla sme použili widgety grafov. Vytvoril som jednoduchú prehľadovú obrazovku (dashboard) pre zobrazenie aktuálnych údajov zo simulovaných senzorov. Grafana mi beží na localhoste na porte 3000.





7. Ako spustiť/rozbehať projekt

Najprv je potrebné si nainštalovať VirtualBox + ideálne aj extension pack. Stiahnuť si Linux image a pripojiť si ho do svojej virtuálky. Po inštalácii a nastavení všetkého (Linux, IDE, Node.JS + príslušné balíčky) môžeme projekt spustiť podľa nasledovných krokov:

1) Otvorím terminál a zadám tento príkaz na spustenie broker MQTT na pozadí

```
mosquitto_sub -v -t „#“ &
```

2) Potom zadaj príkaz na spustenie príslušného Node.js s našou aplikáciou

```
node app.js
```

3) Zadaním týchto príkazov sa uistím, že služba influxDB a služba Grafana bežia

```
sudo systemctl enable influxdb
sudo systemctl start influxdb
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

4) Potom otvorím prehliadač a zadám adresu URL „localhost:1886“, aby som spustil stránku s node-red a nasadil flow. Ten stačí naimportovať z príslušných odovzdaných súborov (node-red-flows.json). Treba doinštalovať cez palette aj podporu pre node storage influxdb out.

5) Na koniec otvorím ďalší tab v prehliadači a zadám URL localhost:3000 na spustenie grafana dashboardu.

Záver

Mám skúsenosť s časovo orientovanými databázami MachBase a TimescaleDB, keď som robil rigoróznu prácu. InfluxDB bol teda pre mňa novinkou. V JavaScripte programujem a rovnako mám skúsenosť s vizuálnym programovaním, ale PLC zariadení (Ladder a FBD) či MIT Inventor, ale Node-red som videl prvý krát v živote.

Čo sa týka IoT tak myslím, že nasledovný obrázok s mojimi hračkami je celkom samovysvetľujúci. Mám niekoľko kitov Raspberry PI 4 aj s klávesnicami, Joy-Pi a Grove Base Kit s rôznymi senzormi. Aj som chcel urobiť napojenie na reálny senzor teploty a vlhkosti vzduchu vid. tá biela krabička na Raspberry PI je Xiaomi senzor, ktorý by bol na to úplne ideálny, ale pre nedostaok času som spravil, len to čo bolo v zadaní. Už tak zrejme stratím body za neskoršie odovzdanie. Osobne radšej preferujem programovanie a automatizáciu IoT v jazyku Python (MicroPython).

K dispozícii sú kompletné zdrojové kódy vrátane package.json, node-red-flows.json a Grafana-Dashboard.json. Rovnako sú k dispozícii JSON vygenerované súbory zo simulácie, alebo po spustení aplikácie v priečinku outputs. Moduly Node.js teda neodovzdávam, keďže obsahujú cca 5000 súborov a 45 MB.