Západočeská Univerzita v Plzni

Fakulta Aplikovaných Věd

Kybernetika a Řídící Technika

Semestrální projekt - tým Red

Sběr a export teplotních dat, zprovoznění webserveru a webové stránky

Dalibor Máj
Hynek Moudrý
David Preibisch
Bohdan Yeremenko

KKY/ITE 2019/2020

Obsah

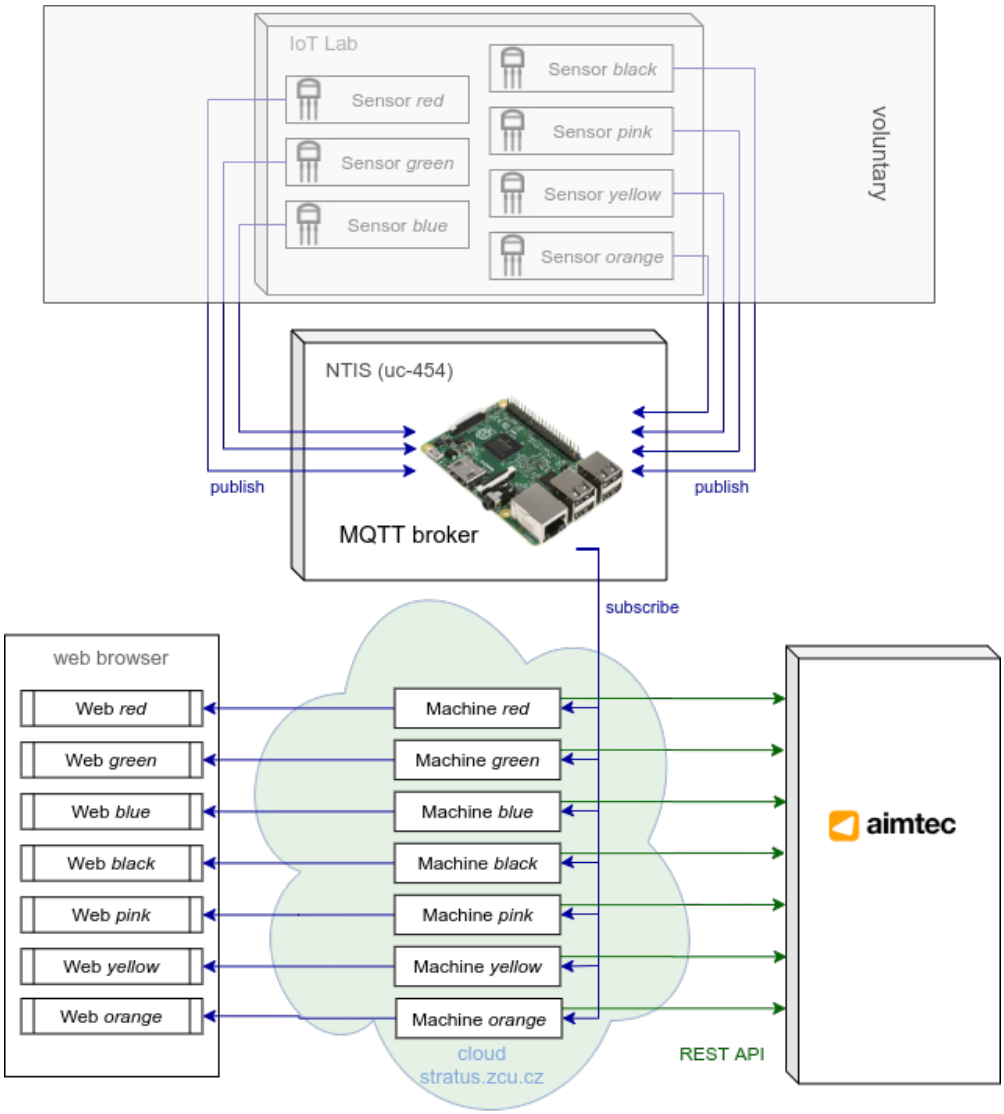
1 Úvod

Dokumentace stručně popisuje řešení semestrálního projektu, který se zabývá čtením, zpracováním a exportem dat z teplotních senzorů.

V první části bude popsáno, jakým způsobem se data z čidel získávají a následně zpracovávají, aby měl uživatel obecný přehled o tom, jakým přístupem jsou data ze senzorů zpracovávána a poté odesílána na zákazníkův server.

Dále bude popsáno zprovoznění a funkce webserveru, který slouží primárně pro poskytování a aktualizaci webové stránky. Ta je následně popsána v další sekci.

Poslední část dokumentace obsahuje návod k obsluze programu, tedy spouštění a ukončování.



Obr. 1: Náhled na zadaný projekt

2 Příjem a export dat

2.1 Zpracování dat z teplotních čidel

V této dokumentaci bude uvažováno pouze zpracování **uměle vygenerovaných dat** příslušných jednotlivým týmům. Každý tým reprezentuje jedno teplotní čidlo. Jména týmů jsou black, blue, green, orange, pink, red a yellow.

Formát dat je například:


```
{"source": "fake",
"team_name": "blue",
"created_on": "2020-03-24T15:26:05.336974",
"temperature": 25.72965126684654}
```


Tato data jsou odesílána s periodou 60 vteřin mikropočítačem Raspberry Pi, následně jsou tato data odebírána (subscribe) programem napsaným v jazyce Python pomocí protokolu MQTT.
V tomto programu je nastavena *IP adresa mikropočítače*, dále *topic*, kde jsou data z čidel a *přístupové údaje*, aby bylo možné tato data odebírat pomocí protokolu MQTT z brokeru běžícího na Raspberry Pi.

Takto přijatá data jsou nejprve zpracována tak, aby byla zajištěna maximální robustnost ve smyslu odolnosti vůči chybám v přijatých datech. Data se ukládají lokálně do JSON souborů, abychom je při případném výpadku programu neztratili a při opětovném spuštění znovu načetli. Následně se ze zpracovaných (opravených) dat vytváří základní statistiky, které jsou zapisovány do souboru *data.json*. Ten obsahuje základní statistiky, vytvořené z přijatých dat příslušných každému týmu (čidlu). Je nutné podotknout, že data a statistiky v JSON souborech jsou pouze z příslušného dne, čehož dosahujeme filtrováním dat při výpočtu statistik.

Po vypočítání statistik týmu se modulem *websockets* připojíme k běžícímu webserveru (skript *tornado_server_v1.3.py*) a pošleme zprávu v následujícím formátu:


```
broadcast {"blue": {"prumerna": "9.4", "maximalni": "34.8", "minimalni":
"-12.9", "posledni": "15.0", "cas": "2020-05-29T16:24:15", "online": true}}
```


Klíčovým slovem *broadcast* říkáme webserveru, že chceme zprávu odeslat všem připojeným klientům.

Dále pošleme na servery AIMTECu záznam o teplotě a jako poslední pošleme webserveru zprávu (*server <bool>*) o úspěšnosti záznamu, která určuje stav serveru AIMTEC.

2.2 Export dat

K exportu dat využíváme poskytované rozhraní REST API, které funguje na základě protokolu HTTP a jeho základních požadavků (GET, POST, PUT, DELETE).

Rozhraní obsahuje šestnáct vlastních požadavků, z nichž budeme potřebovat pouze čtyři (*login, createMeasurement, createAlert, getSensors*). Obsluhu těchto požadavků má na starost třída *RestAPI* v souboru *rest_api.py*.

Požadavek *login* se volá při spuštění skriptu *mqtt_subscriber.py* a dále pokaždé, když přijde zpráva od senzoru našeho týmu a jeho úspěšnost nám udává stav serveru AIMTECu.

Pokud je *login* úspěšný, pokusíme se vytvořit záznam o teplotě požadavkem *createMeasurement* a rovnou zkontrolujeme, jestli se teplota nenachází mimo meze, které zjišťujeme požadavkem *getSensors*. Pokud se teplota pohybuje mimo zjištěné meze, vytvoříme alert požadavkem *createAlert*, který se volá pouze poprvé v za sebou jdoucí sekvenci několika "extrémních" teplot.

3 Webserver

Pro zprovoznění webserveru používáme framework Tornado, který se stará o komunikaci mezi vzdáleným klientem a virtuálním počítačem, na kterém webserver běží. K odesílání dat z webserveru klientům využíváme protokolu WebSockets.

Vzdálený klient se přes prohlížeč připojí na webserver adresou 147.228.121.62:8881, webserver detekuje tento GET požadavek a pošle klientovi soubor *index.html*. Soubor vyžaduje k běhu i javascript *my_script.js* a stylizaci *style.css*. Javascript si při spuštění vytvoří websocket spojení s webserverem a po otevření spojení si zažádá o všechny statistiky ze souboru *data.json* a o status serveru AIMTECu. O tyto informace si zažádá odesláním websocket zpráv "request" a "req_server". Při otevření websocket spojení si webserver toto spojení (instance Handleru) uloží do globální proměnné, aby bylo možné odesílat zprávy všem klientům.

Webserver také využívá modulu *asyncio*, aby byla zajištěna co nejspolehlivější komunikace při vysokém počtu připojených klientů. Dále webserver pravidelně kontroluje stav senzorů jednotlivých týmů tak, že každou minutu zjistí čas poslední aktualizace a porovná ho s jeho předchozí hodnotou (před minutou). Pokud se časy rovnají, nedošlo v posledních minutě ke změně a senzor je tedy offline. Skutečná detekce kdy je senzor offline se na webové stránce může projevit v rozsahu od jedné do dvou minut. Tento rozsah je dán časovou odchylkou mezi pravidelným kontrolováním a aktualizací dat.
Informace o stavu senzorů webserver broadcastuje ve formátu:

```
statistics {"black": {"online": true}, "blue": {"online": false}...}
```

4 Webová stránka

Webová stránka obsahuje menu pro výběr týmu, jehož statistiky chceme zobrazit. Zvolený tým je kontrastně zvýrazněn pro dobrou viditelnost ve špatných podmínkách. Stav senzorů je reprezentován zeleným indikátorem vedle názvu týmu, pokud indikátor zmizí, stav je offline.

Statistiky jsou zobrazeny uprostřed stránky v přehledných blocích. Je zde zobrazena poslední, průměrná, maximální a minimální denní teplota a čas záznamu poslední teploty. Stav serveru AIMTECu je zobrazen slovně v pravém horním rohu.

Stránka je dynamicky aktualizována, není tedy třeba ji obnovovat pro nejnovější informace.

5 Rozběhnutí a ukončení programů

V této části bude popsáno, jakým způsobem bylo docíleno, aby skripty běžely nepřetržitě.

Skripty *mqtt_subscribe_v1.7.py* a *tornado_server_v1.3.py* jsou nahrány na virtuálním počítači pomocí cloudové služby ZČU.
Ze složky, kam byly tyto skripty nahrány jsou pro jejich spuštění nutné dva terminálové příkazy:

```
nohup python3 mqtt_subscriber_v1.7.py &
```

```
nohup python3 tornado_server_v1.3.py &
```

kde příkazem *nohup* říkáme, že nechceme aby se skript ukončil po přerušení ssh připojení a ampersandem na konci skript spustíme asynchronně v pozadí.

Po každém z těchto dvou příkazů se v konzoli vypíše PID procesu. Pro ukončení skriptů tato čísla budeme potřebovat a to konkrétně v příkazu:

```
kill PID
```

Pokud bychom si PID nezapsali, můžeme je vyhledat příkazem:

```
ps -ef
```

V každém skriptu je obsažen modul *logging* pro logování aktivit a chyb, které jsou zapisovány do logovacích souborů ve složce logs.