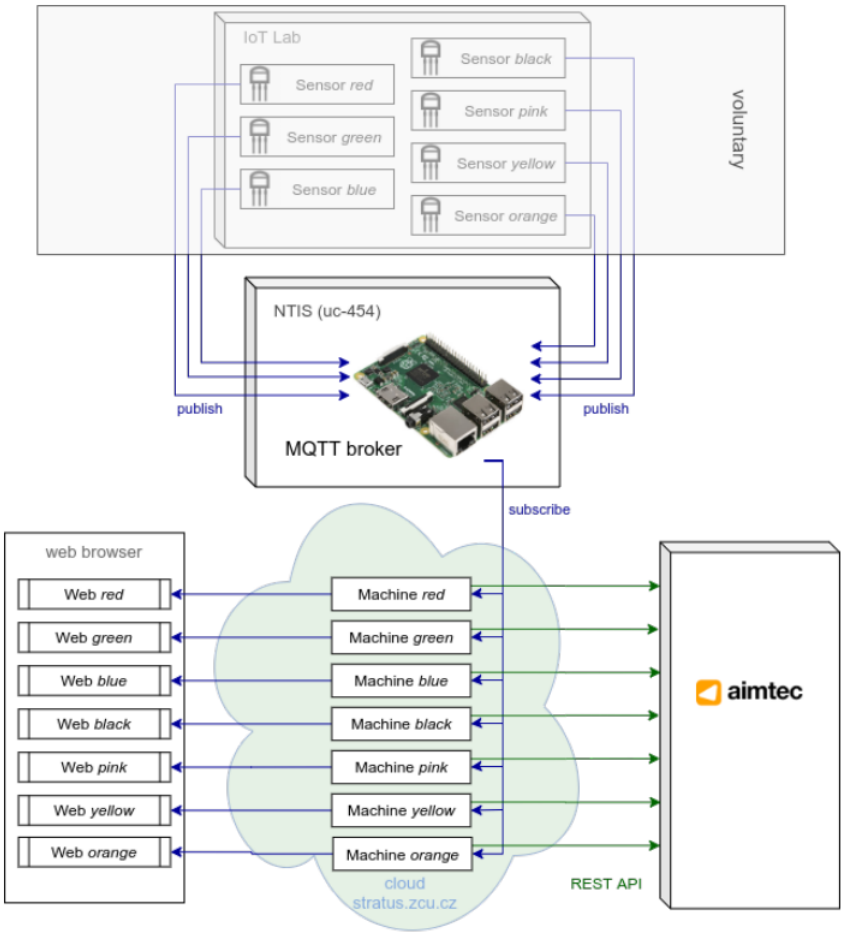
Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Internetové Technologie
Dokumentace k projektu

Václav Javorek, Ondřej Valach, Jiří Švamberg

TEAM BLACK

30. května 2020



Obsah

1 Cíl projektu

Vytvořit funkční program, který subscribuje na MQTT broker, přijímá datové balíčky z teplotních senzorů, odesílá data konkrétního senzoru do rest API a obsluhuje webserver poskytující webovou stránku. Na webu bude přehledně zobrazen stav a základní statistika všech senzorů.

2 Použité nástroje a technologie

- Overleaf - kolaborace při vytváření dokumentace
- Visual studio Code/2019 - úpravy kódu
- jazyk HTML - vizalizace dat a vzhled stránky
- jazyk Python - funkcionalita stránky a získání dat ze senzoru
- jazyk JavaScript - dynamická funkčnost webové stránky
- framework Tornado - přenos dat z Pythonové částí do zobrazení v HTML
- framework Bootstrap - grafické rozhraní webové stránky
- stratus.zcu.cz - virtuálního stroj
- Postman - testování příchozích dat a volání Aimtec Rest API
- WinSCP - připojení přes ssh vytvořeným přístupem
- Aimtec REST API
- Knihovna Requests - komunikace protokolem http
- Tmux – zajištění nepřetržitého běhu programu na virtuálním stroji

3 Čistá instalace a spuštění produktu v krocích

1. Na server s předinstalovaným Pythonem 3.7 a programem tmux nahrajte obsah složky *setup* z GitHub repozitáře ITE_black.
2. Na serveru vytvořte session pomocí programu tmux, ve které poběží program nepřetržitě:

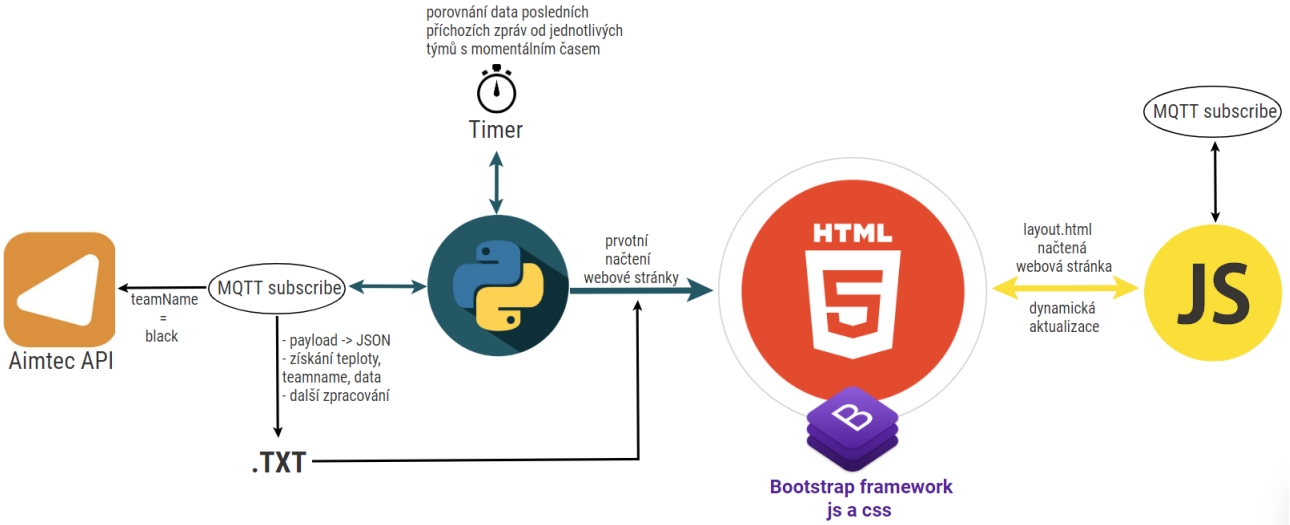
   ```
   tmux new -s 'nazev_session'
   ```

3. Spusťte skript `main.py` ve vytvořené session:

   ```
   python3 main.py
   ```

4. Na portu **8889** IP adresy vašeho serveru je nyní možné otevřít webovou stránku s údaji z MQTT brokeru.

4 Programová funkčnost



4.1 Hlavní program pro sběr a export dat

Hlavní program **main.py** se skládá ze tří separátních threadů (což umožňuje modul *threading*). V každém z nich běží nekonečná smyčka, která vykonává jednu ze základních funkcí nutných pro správný chod produktu.

4.1.1 1. thread - mqtt subscriber

První thread nejprve subscribne na MQTT broker pod zadaným username a heslem. Dále už jede v loopu a přijímá zprávy z jednotlivých senzorů.

Pokaždé, když přijde message, program načte její obsah (payload) do formátu *JSON*. Z něj si vytáhne klíčové informace, jako je team id senzoru, aktuální teplota a časové údaje. Pokud byla zpráva poslána z "našeho"senzoru barvy *black*, pošle data do rest API (viz sekce 4.2).

Dále program vypočítá základní statistiky, jako je maximální, minimální a průměrná denní teplota. Všechny hodnoty jsou uloženy do souboru **save_(barva).txt**. Probíhá také kontrola toho, zda poslední message přišla ve stejný den, jako ta předposlední. V případě, že tomu tak není, promaže program všechna uložená data a začne ukládání dat nového dne, aby byly denní statistiky správně podle zadání projektu.

Pozn.: Ukládání do textového souboru jsme zvolili pro jednoduchost a robustnost tohoto řešení v rámci dané úlohy. Pro potřebu větších objemů dat, jiného typu dat nebo složitější statistiku by bylo vhodnější použít nějakou z databázových technologií. Nebyl by však problém k tomuto přístupu v případě dalšího vývoje projektu přejít.

4.1.2 2. thread - webserver

Tento thread rozbíhá webserver pomocí Python frameworku Tornado. Je spuštěna smyčka, která běží a čeká na to, až uživatel bude chtít načíst stránku. Po zadání adresy v prohlížeči se spustí metoda MainHandler (více v sekci 4.4.1).

4.1.3 3. thread - timer

Jednou z věcí, kterou je při běhu programu nutno kontrolovat, je stav jednotlivých teplotních čidel. Jedná se o stavy *online* a *offline*.

Stav *online* se nastavuje automaticky při každém příchodu zprávy od konkrétního senzoru. Stav *offline* naopak obstarává tato část kódu. Porovnává časové údaje posledních přijatých dat od jednotlivých senzorů s momentálním časem. Pokud tento rozdíl překročí **60 sekund**, zavolá metodu `setOffline` s parametrem jména senzoru.

4.2 Odesílání dat přes REST API do Aimtecu

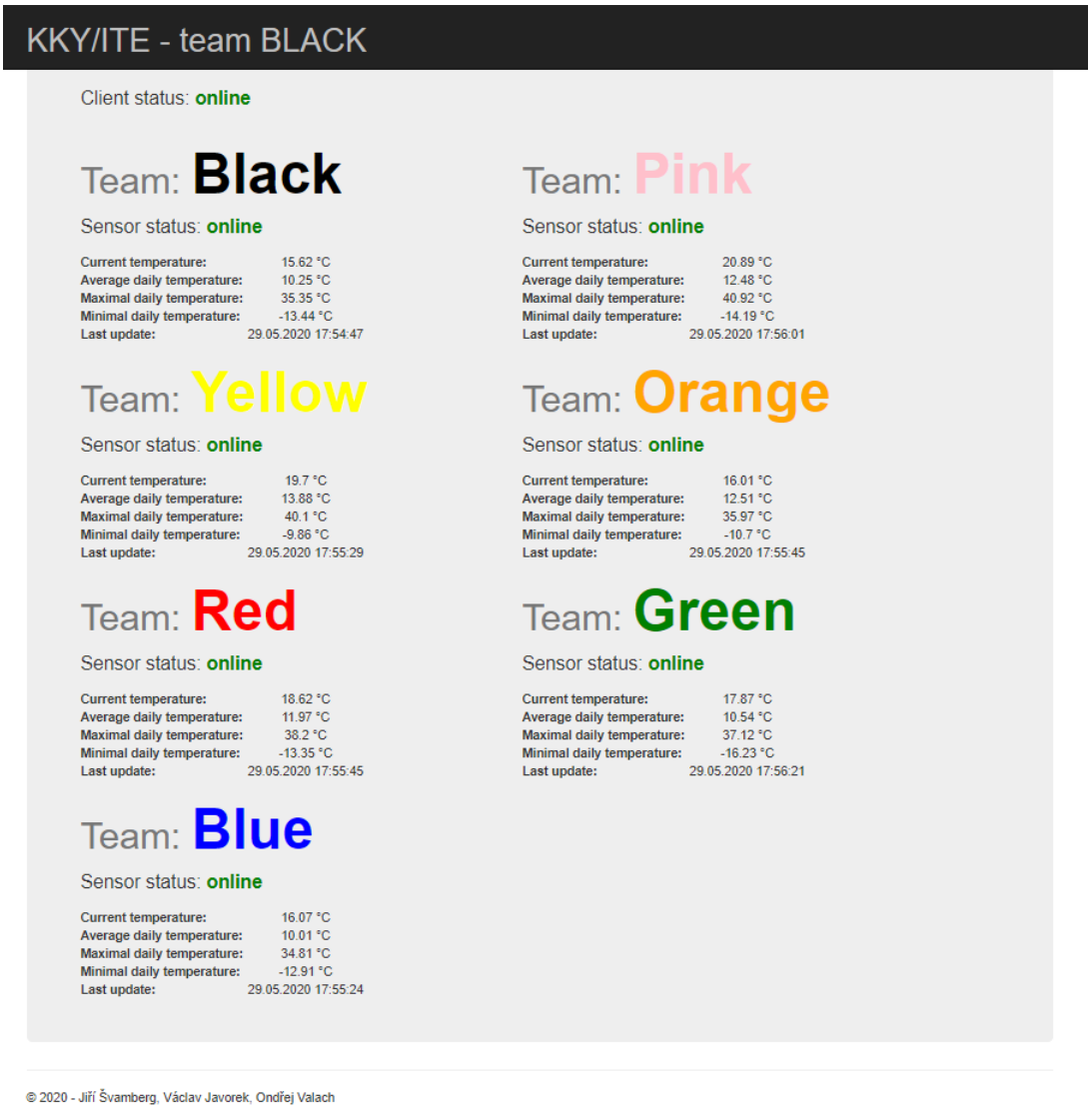
Jednou z hlavních částí této práce je průběžné odesílání dat, získaných ze senzoru týmu, užitím REST API od firmy Aimtec. Pomocí knihovny **requests** se *post* metodou `login` získá UUID týmu, následným zavoláním *get* `sensors` je získáno UUID senzoru týmu. *Post* metoda `measurements` dále vytváří záznam nového měření senzoru *black* a odesílá ho do Aimtecu. Následuje podmínka, která kontroluje, zda teplota nevykročila z mezí 0-25 °C. Pokud ano, *post* metodou `alerts` se odešle upozornění do firmy o příliš vysoké nebo naopak nízké teplotě.

4.3 Nasazení programu na cloud a zprovoznění webserveru

Před nasazením kódu na virtuální stroj bylo potřeba zajistit vlastní instanci (virtuální stroj) na *www.stratus.zcu.cz*. Na této instanci má uživatelské konto a složku každý z členů našeho týmu, pro práci s kódem a jeho nahrání byla vybrána uživatelská složka *ondra*.

Po zajištění tohoto pracovního prostoru se již dá s kódem pracovat jednoduše. Pomocí klienta SCP (konkrétně *WinSCP*) je všechen kód přenesen SSH protokolem do složky uživatele *ondra*. Dále po připojení přes příkazovou řádku zavoláním `ssh ondra@147.228.121.17` a vytvořením session v programu *tmux*, který byl doinstalován na linux systém vzdáleného stroje, lze příkazem `python3 main.py` spustit v této session hlavní program.

4.4 Vizualizace - webová stránka



4.4.1 Python

V případě, že uživatel bude chtít zobrazit data na naší webové stránce a zadá do vyhledávače odpovídající adresu, webový framework Pythonu Tornado spustí `MainHandler`. Ten si načte všechna potřebná data o každém týmu z aktuální verze textových souborů (např. *save_black.txt*) a uloží je do dictionary *data*. Handler následně zavolá metodu `render`, jejíž atributy jsou cesta ke spouštěnému HTML souboru (**index.html**), a všechna data, která jsou na stránce potřeba zobrazit.

4.4.2 HTML

V souboru **index.html** se nejprve načte **layout.html**, který zajišťuje, že všechny stránky budou mít stejný vzhled. V tomto HTML se volají veškeré JavaScriptové a CSS scripty, které jsou na stránce potřeba, včetně scriptů webového frameworku Bootstrap zajišťujícího uživatelsky příjemný vzhled stránky. Každá důležitá hodnota, kterou budeme chtít načítat a zpracovávat v JavaScriptu, musí mít své vlastní ID (např.: *status_black*).

4.4.3 JavaScript

JavaScriptová část funguje podobně jako hlavní program napsaný v Pythonu. Při načtení těla stránky se spustí metoda `MQTTConnection()`, která se pomocí `userName` a `password` přihlásí na broker a začne z něj přijímat zprávy.

Při přijetí zprávy se zavolá metoda `onMessageArrived(msg)`, kde `msg` je příchozí zpráva ve formátu JSON. Příkazem `document.getElementById('element_id').innerHTML` se podle `team_name` načtou hodnoty u daného týmu, aktualizují se a následně se pomocí stejného příkazu vykreslí na stránku. Stejně lze upravovat i styly jednotlivých elementů. V tomto případě je upravována barva aktuální teploty.

V případě, že je teplota mimo meze 0-25 °C, se text teploty zbarví červeně, tzn. přidá se pomocí `document.getElementById('element_id').style.color` na stránku k elementu s daným ID `style="color: red"`. V opačném případě se atribut `style` úplně odebere.

S příchozí zprávou se také ukončí a znovu spustí časovač, po jehož doběhnutí (60 sekund) se nastaví status daného senzoru na *offline*, text statusu se zbarví červeně a místo aktuální teploty vykreslí —. Po příchozí zprávě se opět status změní na *online* a zelenou barvu.

Pokud je problém s připojením k brokeru, nastaví se status klienta opět na *offline* a současně se opět změní barva textu na červenou.