SIN – Inteligentní systémy
Implementace meteostanice s autonomním řízením

Bc. Petr Stehlík <xstehl14@stud.fit.vutbr.cz>

Bc. Matej Vido <xvidom00@stud.fit.vutbr.cz>

1 Úvod

Cílem projektu je navrhnout a implementovat meteostanici využívající MQTT protokol pro zasílání naměřených hodnot serveru. Server implementuje databázi a webové grafické uživatelské rozhraní (GUI), analyzuje historické hodnoty a na jejich základě řídí akce aktuátorů.

Meteostanice měří teplotu, vlhkost a tlak vzduchu; vlhkost v květináči a světelnou intenzitu. Navržené aktuátory jsou ovládání žaluzií, klimatizace, topení a zavlažování rostlin. Vše je autonomně ovládáno na základě předem stanovených pravidel.

GUI zobrazuje aktuální a historické naměřené hodnoty jednotlivých senzorů, trendy a naposledy vykonané akce aktuátorů společně s krátkodobou předpovědí počasí získanou z volně dostupných zdrojů.

2 Nástroje pro monitoring a řízení

Na trhu je velké množství dostupných nástrojů pro monitoring a řízení a to i na poli chytrých domácností. Je zde mnoho komerčních a uzavřených systémů pro automatizaci domácnosti. V současnosti nejznámější a pravděpodobně i nejrozšířenější je Apple Home[1], který využívá protokolu Homekit také od společnosti Apple. Na trhu existuje pro Apple Home velké množství produktů a neustále se jejich počet zvyšuje. Existuje ale i mnoho open-source nástrojů pro řízení domácnosti. Jejich nejrozšířenější zastupitele jsou zde v krátkosti popsány.

2.1 Grafana

Grafana (`https://grafana.com/`) je vizualizační a analytický nástroj pro data zachycená v čase. Grafana samotná není primárně určena pro monitoring a řízení chytrých domácností, ale je natolik upravitelná, že existují konfigurace, které toto užití zpřístupňují. Je dostupná s mnoha rozšířeními a zdroji dat. Uživatel si vytvoří dashboard, který si následně nakonfiguruje a seskládá z dostupných modulů. Tyto moduly mimo dalších funkcionalit umožňují zobrazit čárový graf, jednotlivé hodnoty, trendy hodnot a s pomocí modulů i různé přepínače.

2.2 Domoticz

Domoticz (`https://domoticz.com/`) je kompletním open-source systémem pro domácí automatizaci. Tento nástroj dovoluje monitorovat, řídit a konfigurovat mnoho různých zařízení od různých výrobců. Disponuje i automatickým učením senzorů a aktuátorů. Rozhraní pro uživatele je vytvořeno jako webová stránka dostupná na stroji s lokální instalací Domoticz.

2.3 OpenHAB

OpenHAB (`https://www.openhab.org/`) je dalším velmi známým zástupcem open-source nástrojem pro domácí automatizaci. Podporuje velké množství platforem, výrobců a zařízení. Dokáže integrovat mnoho systémů do jednoho centrálního řešení, které lze ovládat mobilní aplikací, webových rozhraním nebo nativní desktopovou aplikací. Pro automatizaci disponuje rozsáhlým systémem pro tvorbu komplexních pravidel.

[1] `https://www.apple.com/lae/ios/home/`

2.4 Home Assistant

Home Assistant (`https://home-assistant.io/`) je primárně navrhován pro užívání na Raspberry Pi. Jako předchozí nástroje také podporuje široké množství výrobců a zařízení. Také disponuje konfigurovatelným webových rozhraním s přehledy zařízení a kontrolou aktuátorů.

2.5 BeeeOn

BeeeOn (`https://beeeon.org/`) je systém pro domácí automatizaci vyvíjený na FIT VUT v Brně. Tento systém je primárně vyvíjen jako bezpečná domácí brána pro různé IoT zařízení. Oproti předchozím systémům je nutno pro plnou funkcionalitu systému vlastnit domácí bránu, kterou lze propojit s několika výrobci a jejich zařízeními. BeeeOn disponuje Android mobilní aplikací pro ovládání domácnosti a prototypem webového rozhraní.

2.6 Vlastní řešení

Kromě uvedených kompletních řešení pro monitoring a analýzu řídících systémů lze vytvořit vlastní řešení z dostupných open-source knihoven, frameworků a nástrojů. Toto řešení jsme ve finále zvolili vzhledem k náročnosti instalace a konfigurace ostatních systémů. Pro naše řešení jsme zvolili kombinaci Angular pro klientskou část webové aplikace a Flask (Python) s SQLite3 pro backend. Pro distribuci zpráv od senzorů k serveru a mezi aktuátory a serverem MQTT broker Mosquitto.

Rozšířeným frameworkem pro tvorbu klientské části webových aplikací je Angular[2]. Angular umožňuje snadnou integraci různých knihoven pro zobrazování grafů a tvorbu interaktivních single-page aplikací (SPA).

K implementaci serverové části webové aplikace je možné použít framework Flask[3] napsaný v jazyce Python. Flask umožňuje rychlou a snadnou tvorbu REST API na propojení serverové a klientské části webové aplikace. Výhodou použití jazyku Python pro tvorbu serverové části aplikace je, že obsahuje moduly pro obsluhu databázového systému a zprotředkování síťové komunikace s dalšími prvky celého řídícího systému.

Mosquitto[4] je volně dostupný broker pro MQTT protokol používaný v prostředí systémů IoT podporující různé platformy.

3 Architektura a implementace

Meteostanice obsahuje reálné senzory, z nichž jsou data přenášena na server. Server data ukládá do databáze, řídí činnost aktuátorů podle pravidel, které jsou také uloženy v databázi a obsluhuje webovou aplikaci, která poskytuje grafické uživatelské rozhraní k celému systému.

Činnost aktuátorů je simulována a pro účely testování byl také sestrojen modul pro simulování činnosti senzorů.

V části 3.1 je popsán hardware a způsob jeho zapojení. V následující části 3.2 je popsána implementace softwarových modulů systému.

3.1 Hardware

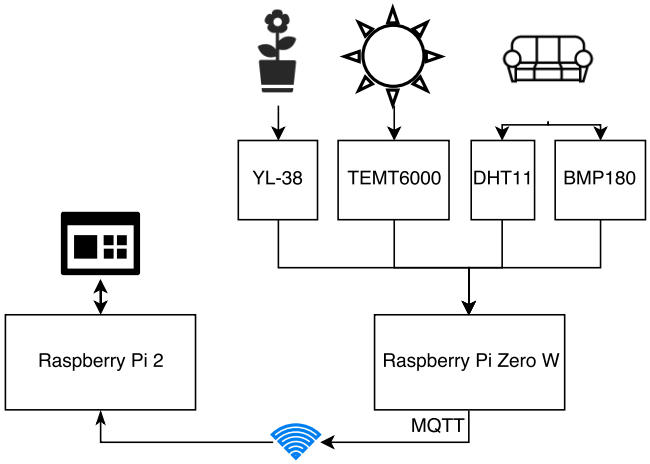
Základem pro hardware byly zvoleny Raspberry Pi 2 jako server a Raspberry Pi Zero W jako samotná meteostanice. Server realizován jako Raspberry Pi 2 zpřístupňuje uživateli minimalistické webové rozhraní a slouží jako MQTT broker pro meteostanici. Také jsou zde ukládány všechny naměřené hodnoty do SQLite3 databáze. Na obrázku **??** je znázornená schéma zapojenia senzorov a Raspberry Pi.

K Raspberry Pi Zero W jsou připojeny následující senzory:

[2] `https://angular.io/`
[3] `http://flask.pocoo.org/`
[4] `https://mosquitto.org/`

- DHT11 - digitální senzor pro měření teploty a vlhkosti vzduchu
- BMP180 - digitální senzor pro měření barometrického tlaku, teploty a nadmořské výšky
- TEMT6000 - analogový senzor pro měření intenzity okolního světla
- YL-69 společně s YL-38 - analogový senzor pro měření vlhkosti půdy

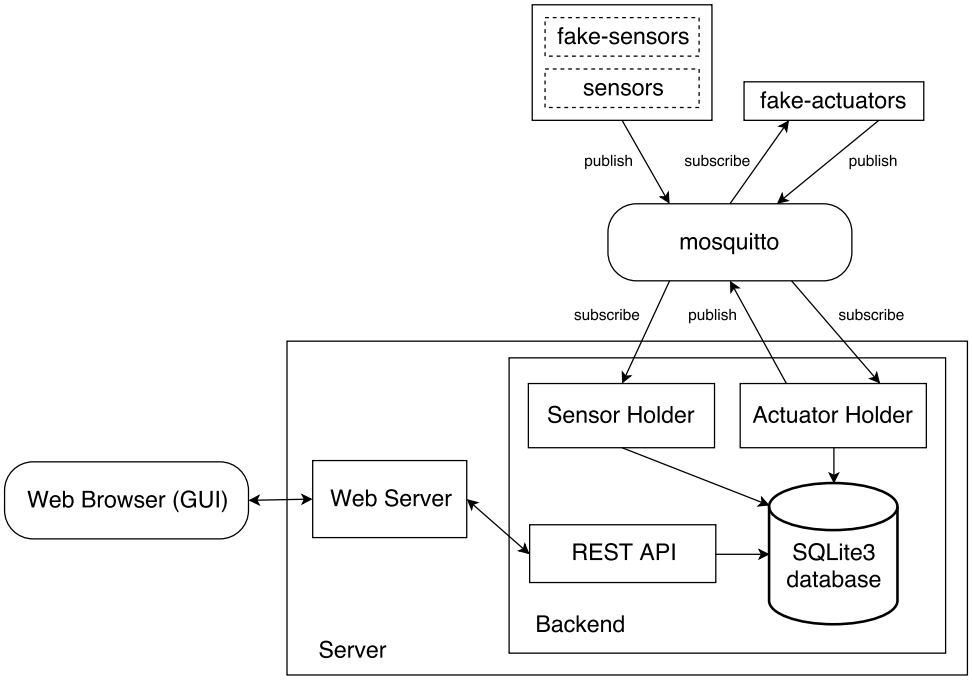
Pro konverzi analogových senzorů je použit AD převodník MCP3008.



Obrázek 1: Schéma zapojení hardware.

3.2 Software

Na obrázku 2 je znázorněné propojení jednotlivých modulů a způsob jejich komunikace.



Obrázek 2: Schéma propojení software modulů.

Interakci s uživatelem zprostředkovává webová aplikace, která se skládá z grafického uživatelského rozhraní zobrazovaného ve webovém prohlížeči a serverové části.

Webové grafické uživatelské rozhraní je naimplementované ve frameworku Angular. Webová aplikace dále využívá knihovnu Dygraphs pro zobrazování čárových grafů pro jednotlivé naměřené metriky. GUI získává data pomocí REST API ze serverové části aplikace. Při prvním načtení se získají historická data za posledních 24 hodin a dále si GUI získává nová data každých 5 sekund.

K implementaci REST API je použit framework Flask. Serverová část webové aplikace (backend) přijímá data ze senzorů a ukládá je do databáze. Na základě dat ze senzorů a nastavených pravidel určuje činnost a řídí aktuátorů.

REST API umožňuje:

- získat poslední vzorek dat ze senzorů včetně časové známky spolu s trendem vypočítaným pro každou měřenou metriku
- získat informace o aktuálním stavu aktuátorů a pravidlech, kterými jsou řízeny
- nastavit nové prahové hodnoty jednotlivým aktuátorům

Výpočet trendů probíhá na základě lineární aproximace z určitého počtu posledních vzorků dané veličiny. Pro každou veličinu je tak určena funkce ve tvaru:

$$y = ax + b$$

Koeficient a udává jako prudce funkce klesá nebo stoupá. Koeficient b udává posunutí funkce po ose x. Toto posunutí nemá na změnu veličiny vliv a proto může být z dalších výpočtů vypuštěno. Do vztahu:

$$y = ax$$

je potom za x dosazený počet vzorků, z kterých sa počítá trend. Hodnota y udáva o kolik sa daná veličina za posledních x vzorků změnila.

Pro každou veličinu byly empirickým pozorováním určené hodnoty prahů. Pokud je absolutní hodnota změny veličiny větší než práh, tak podle znaménka změny veličiny je určen stoupající nebo klesající trend, jinak je určen trend rovnoměrného průběhu.

V databázi SQLite3 sa nachází 3 tabulky:

- `weather_data` - data získaná ze senzorů
- `actuators` - informace o stavu aktuátorů
- `thresholds` - informace o pravidlech řídících činnost aktuátorů

Data ze senzorů získává, agreguje a následně odesílá backendu modul `sensors`. Pro testovací účely byl připraven i modul `fake-sensors`, který simuluje činnost modulu `sensors` a odesílá umělá data ve stejném formátu.

Modul `fake-actuators` simuluje činnost aktuátorů. Z backendu přijímá příkazy na změnu stavu aktuátorů, které simuluje. Po provedení činnosti aktuátorů je odeslána informace o stavu, ve kterém se aktuátor nachází backendu.

Následuje seznam simulovaných aktuátorů:

- klimatizace - řízená na základě údajů o teplotě
- topení - řízené na základě údajů o teplotě
- žaluzie - řízené na základě údajů o intenzitě světla
- zavlažování rostlin - řízené na základě údajů o vlhkosti půdy

Komunikace mezi backendem, senzory a aktuátory probíhá skrz protokol MQTT. Jako broker byl zvolen nástroj Mosquitto.

MQTT topic pro zprávy zasílané senzory má tvar `home/ws/sensor/<jméno senzoru>` a obsah dané zprávy ve formátu `<unix časové razítko>;<hodnota>`.

MQTT topic pro zprávy zasílané aktuátory má tvar `home/ws/actuator/<jméno senzoru>` a obsah dané zprávy ve stejném formátu jako u senzorů.

Aktuátory přijímají MQTT zprávy od backendu pro nastavení stavu s MQTT topic ve tvaru: `home/ws/actuator/<jméno senzoru>/state` a obsahem zprávy je pouze nově nastavovaná hodnota v podobě hodnoty 0 nebo 1.

4 Výsledky

5 Závěr