VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

MONITOROVÁNÍ TEPLOTY MENŠÍCH OBJEKTŮ

MONITORING OF TEMPERATURE FOR SMALL BUILDINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAKUB HANDZUŠ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VOJTĚCH MRÁZEK**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Handzuš Jakub**
Obor: Informační technologie
Téma: **Monitorování teploty menších objektů**
Monitoring of Temperature for Small Buildings
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s problematikou zpracování dat ze senzorů z internetu věcí (IoT) a platformami (např. ESP8266, ESP32 apod.) umožňující připojení takovýchto senzorů. Zaměřte se zejména na možnosti efektivního ukládání dat do vhodného úložiště.
2. Navrhněte systém, který bude na serveru zpracovávat data z čidel připojených ke zvolené IoT platformě. Celý systém navrhněte tak, aby data byla ukládána efektivně s ohledem na typické operace prováděné nad těmito daty.
3. Zpracujte studii na výše uvedené téma.
4. Navržený systém implementujte. Tento systém bude sledovat teplotu minimálně ve třech místech.
5. Vyhodnoťte parametry navrženého řešení a diskutujte možnosti pokračování projektu.

Literatura:

- H. Cai, B. Xu, L. Jiang and A. V. Vasilakos, "IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges," in *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75-87, Feb. 2017.
- Dále dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Mrázek Vojtěch, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Cieľom tejto práce je navrhnúť a implementovať IoT systém pre sledovanie teploty ovzdušia menších objektov, napr. domácností. Nakoľko má byť daný systém finančne dostupný širšej verejnosti, je nutné, aby bol schopný plnej funkcionality aj pri nízkych obstarávacích a prevádzkových nákladoch – na základe tejto požiadavky je nutné vykonať analýzu dostupných alternatív pre prevádzku systémov. Pri výbere vhodnej technológie pre ukladanie získaných dát je potrebné brať ohľad na operácie, ktoré budú nad dátami najčastejšie vykonávané – z tohto dôvodu je vykonaná sada experimentov nad niekoľkými druhmi databázových systémov. Na základe poznatkov zozbieraných počas analýzy a experimentovania sa ako optimálne riešenie javí kombinácia generickej databázy so službami webových hostiteľských serverov. Vo výslednom systéme teda senzor odosiela získané dáta na serverovú časť s databázou, pričom spracované dáta sú následne interpretované za pomoci vizualizácií na klientskej strane.

Abstract

The aim of this thesis is to design and implement an IoT system for monitoring the air temperature of smaller objects, e.g households. As the system is to be financially available to the wider public, it needs to be fully functional even at low procurement and operating costs – based on this requirement, it is necessary to analyze available alternatives for operating the systems. When selecting the appropriate technology for storing the acquired data, it is necessary to take into account the operations most frequently performed on the data – for this reason, a set of experiments is carried out on several types of database systems. Based on the findings gathered during analysis and experimentation, the optimal solution appears to be the combination of a generic database with web host services. In the resulting system, the sensor sends the acquired data to the server with a database, whilst the processed data is subsequently interpreted by the client-side visualizations.

Kľúčové slová

Internet vecí, ESP8266, monitorovaie teploty, databáza pre IoT

Keywords

Internet of Things, ESP8266, temperature monitoring, IoT database

Citácia

HANDZUŠ, Jakub. *Monitorování teploty menších objektů.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vojtěch Mrázek

Monitorování teploty menších objektů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vojtěcha Mrázka. Ďalšie informácie mi poskytol pán Ing. Peter Hanzlik. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jakub Handzuš
10. mája 2018

Poďakovanie

Rád by som poďakoval vedúcemu práce, pánovi Ing. Vojtěchu Mrázkovi, za poskytnuté konzultácie a podporu pri práci. Ďalej ďakujem svojim rodičom a priateľke za podporu a motiváciu, jednak pri vypracovaní práce, ako aj počas celej doby štúdia. V neposlednom rade by som chcel poďakovať aj pánovi Ing. Petrovi Hanzlikovi za odbornú osobnú prednášku ohľadom vytvárania IoT systému v praxi.

Obsah

Kapitola 1

Úvod

V posledných rokoch sa technológia pre monitorovanie menších objektov – domácností – stáva pre bežného človeka čoraz dostupnejšou. Jednou z možností je buď siahnuť po komerčných produktoch renomovaných spoločností, alebo sa vydať cestou vytvorenia vlastného systému. Výsledok tejto práce je určený pre jednotlivcov, ktorých snahou je vytvoriť si vlastný systém, ale zároveň nemajú čas alebo dostatočné vedomosti pre vytvorenie takéhoto systému. Väčšinou sa jedná o domácich počítačových nadšencov, ktorí sa snažia obohatiť domácnosť o inteligentné prvky, pričom ich rozhodujúcim faktorom je práve cenová dostupnosť. Z dôvodu zachovania čo najväčšej jednoduchosti je vytvorený systém zameraný iba na monitorovanie teploty, čo je jedna z najsledovanejších meraných veličín v domácnostiach.

Existuje množstvo dôvodov, prečo je práve teplota predmetom merania objektov. Jednou z najčastejších príčin merania je potreba detekcie prekročenia istej prahovej hodnoty. Zvyčajne toto prekročenie slúži ako spúšťač pre ďalšie akcie. Pre zistenie aktuálnej teploty objektu, prípadne teploty objektu nameranej v istom konkrétnom čase, z akéhokoľvek miesta mimo objektu, je nevyhnutné, aby senzor, vykonávajúci meranie, bol pripojený do internetovej siete – jedná sa teda o zariadenie, ktoré je súčasťou Internetu vecí.

Pri vytváraní systému pre monitorovanie je potrebné sa zamerať nielen na zariadenie vykonávajúce zber dát, ale aj na systém, do ktorého budú dáta následne zasielané. Existuje mnoho alternatív pre návrh tohto systému, avšak väčšina ponúkaných riešení je pomerne finančne náročná. Preto je navrhovaný a následne aj implementovaný systém zameraný na poskytnutie platformy pre zber a vizualizáciu dát pri čo najnižších finančných nákladoch, a tým cielený na skupinu domácich počítačových nadšencov. Keďže je spomínaná cieľová skupina v tejto oblasti pomerne aktívna, pri poskytnutí systému ako otvorený projekt je umožnené užívateľom upravovať zdrojové súbory a tým zjednodušiť vývoj systému pre ich potrebu.

Štruktúra dokumentu Kapitola 2 je zameraná na senzorovú časť, v ktorej sú zdokumentované spôsoby merania teploty, hardvérové platformy umožňujúce zber a zasielanie dát a nakoniec sú popísané komunikačné technológie, s ktorými je možné zasielať dáta na server. V kapitole 3 je priestor venovaný porovnaniu dostupných databázových systémov vhodných pre ukladanie senzorových dát, a zároveň porovnaniu systémov schopných poskytovať potrebné prostriedky pre ich fungovanie. Na základe nadobudnutých vedomostí a vykonaných testov je v kapitole 4 popísaný detailný návrh vytváraného systému. Následne sú v kapitole 5 popísané využité technológie a zaujímavosti z implementácie systému. Kapitola 6 popisuje výsledný systém a jeho testovanie.

Kapitola 2

Senzory pre IoT

Internet vecí je zložený z celého radu rôznych zariadení. Do tejto kategórie môžeme zaradiť aj zariadenie, ktoré monitoruje akýkoľvek objekt a zároveň odosiela namerané dáta pomocou komunikačného protokolu na server. Pri vytváraní takéhoto zariadenia je potrebné v prvom rade špecifikovať merané veličiny, a na základe toho vybrať vhodné senzory. Samotný senzor väčšinou nedokáže komunikovať so serverom cez internet, a preto musí byť rozšírený o výpočtovú jednotku, ktorá spracováva dáta zo senzorov (analógových a digitálnych). Jednotka, v prípade potreby odosiela namerané dáta s využitím daného komunikačného protokolu. Všeobecná schéma zariadenia umožňujúca danú činnosť je znázornená na obrázku 2.1. Cieľom tejto kapitoly je zdokumentovať typy senzorov, vhodné hardvérové platformy, technológie a protokoly, umožňujúce odosielanie dát.

2.1 Senzory

Priemyselné odvetvie už dlhší čas využíva rôzne typy senzorov, ale až príchod internetu vecí posunul využitie senzorov o úroveň vyššie. Za pomoci použitia rôznych typov senzorov dokáže IoT platforma dodať zariadeniam istý druh inteligencie, keďže zhromaždené dáta môžu byť analyzované a ďalej distribuované v sieti. Vo väčšine prípadov dodáva pridanú hodnotu systému až spomínaná analýza, kedy je možné vďaka veľkému objemu dát zo senzorov zistiť stav sledovaného objektu alebo predvídať udalosti. V IoT sieťach sa používajú senzory na meranie teploty, vlhkosti, tlaku, detekciu pohybu, akcelerácie, zemetrasení, požiaru, úniku škodlivých plynov, dymu a mnoho ďalších. V rámci zamerania práce budú v tejto kapitole priblížené technológie využívané na meranie teploty.

Obr. 2.1: Všeobecná schéma zariadenia pre zber a odosielanie dát. Zdroj: [10].

2.1.1 Teplotné senzory

Pre meranie teploty je možné použiť rôzne druhy snímačov pracujúcich s rozdielnymi technológiami, v závislosti od prostredia a od objektu, ktorého teplota je meraná. Prvá kategória snímačov meria teplotu bezdotykovo, v opačnom prípade sa jedná o meranie dotykové. Dotykové meranie sa ďalej rozdeľuje z hľadiska použitej technológie [12].

Termočlánky. Využívajú *Seebeckov* termoelektrický jav, ktorý zabezpečí vznik napätia. Takéto snímače teploty sú vyrobené z dvoch rôznych kovov, pričom sú spojené len na jednej strane. Výstupné termoelektrické napätie je úmerné teplote, avšak závislosť obvykle nie je lineárna. Pri meraní týmto spôsobom je možné dosiahnuť veľký rozsah meraných teplôt v závislosti od použitých materiálov, avšak kvôli malej úrovni výstupného napätia je tento spôsob náchylný na rušenie. Prevod na meranú teplotu je zložitý a časom môže byť ovplyvnený dôsledkom korózie kovov.

Odporové kovové senzory (RTD). Tieto senzory fungujú na princípe zmeny odporu priechodu elektrónov v kove. S rastúcou teplotou, atómy kryštálovej mriežky zvyšujú amplitúdu svojich kmitov, čím sa zvyšuje spomínaný odpor. Tieto senzory najčastejšie využívajú platinu, ktorá sa vyznačuje časovou stálosťou, vysokou teplotou tavenia a možnosťou dosiahnutia vysokej chemickej čistoty. Vďaka týmto vlastnostiam zvládne takýto senzor odmerať teploty od $-200°$ až do $800°$ s veľkou presnosťou.

Polovodičové teplotné senzory (termistory). Sú založené na podobnom princípe ako odporové kovové senzory, v ktorých sa mení odpor v závislosti od teploty. Na rozdiel od kovov je princíp vodivosti polovodičov odlišný, preto nadobúdajú tieto senzory iné vlastnosti. Pri teplote absolútnej nuly sú všetky elektróny viazané pevne ku svojím jadrám – tým pádom materiál nemôže viesť elektrický prúd. Pri zvyšovaní teploty bude koncentrácia nosičov náboja rásť a elektrický odpor sa bude znižovať. Termistory sa ďalej delia na dva typy – termistor NTC (negastor), ktorého odpor pri zahrievaní klesá, a termistor PTC (pozistor), ktorého odpor rastie. Medzi ich najväčšie nevýhody patrí veľká nelinearita, malá časová stálosť a malý merný teplotný rozsah [24].

V dnešnej dobe sa metóda bežne rozširuje o integrovaný obvod, ktorý vykonáva spracovanie nameraného signálu (linearizácia, korekcia, redukcia šumu a pod.). Niektoré senzory obsahujú A/D prevodník, ktorý prevádza teplotu do digitálnej podoby. Tieto dáta sú ďalej prenášané pomocou rôznych komunikačných protokolov.

2.2 Platformy

Dnešné technológie umožňujú zostrojiť finančne nenáročný, ale zároveň výkonný hardvér, plne postačujúci na zber a zasielanie primeraného množstva dát na centrálny kontrolér alebo bránu, ktorá poskytuje pripojenie do internetu. Z toho dôvodu je potrebné, aby platforma umožňovala komunikovať so zariadeniami, ktoré sú pripojené do internetu. Taktiež by mala ponúkať množinu vstupno/výstupných pinov, ktoré umožnia komunikáciu so senzorom.

V poslednej dobe vzniklo veľa spoločností zameraných na vývoj prototypovacích platforiem, ktoré uľahčujú vývojárom s minimálnymi znalosťami v oblastí hardvéru vytvárať zariadenia umožňujúce pripojenie do internetovej siete. V rámci tejto siete vznikli komunity ľudí, ktorí zverejňujú svoje riešenia rôznych projektov pomáhajúce práve začiatočníkom.

V nasledujúcich podkapitolách sú uvedené dostupné typy platforiem, ktoré je možné použiť pri vytváraní zariadenia IoT.

2.2.1 MCU

Trh ponúka veľký výber mikrokontrolérov, ktoré umožňujú zber senzorových dát. Pre jednoduchšie zariadenia, snímajúce a odosielajúce dáta niekoľkokrát denne, môže byť vhodné použitie 8-bitového mikrokontroléra. Avšak pre inteligentnejšie zariadenia, ktoré využívajú rádiovo-frekvenčnú (RF) komunikačnú vrstvu alebo sofistikovanejšie algoritmy, je 32-bitový mikrokontrolér vhodnejšia voľba. Niektoré 32-bitové mikrokontroléry (napríklad ARM Cortex M4/M7/M33) obsahujú špecializované jednotky pre spracovanie dát (FPU, DSP), vďaka ktorým je možné kvalitnejšie spracovávať dáta na jednotke, a tým znížiť veľkosť prenášaných dát. Väčšia výpočtová kapacita 32-bitových mikrokontrolérov umožňuje vykonať procesy rýchlejšie, čím sa dostanú skôr do úsporného režimu a tým šetria energiu. Tieto procesory obsahujú väčšiu kapacitu flash aj RAM pamäte, čím je možné implementovať celý sieťový zásobník bez nutnosti pridania ďalšieho pomocného procesora do systému.

Mikrokontroléry dokážu po pripojení vhodného rádio-frekvenčného rozširujúceho modulu (napr. ATA85200) využívať bezdrôtové siete založené na frekvencii pod 1 GHz alebo 2,4 GHz. Pri výbere bezdrôtovej technológie/protokolu je potrebné zvážiť energetickú náročnosť a cenu.

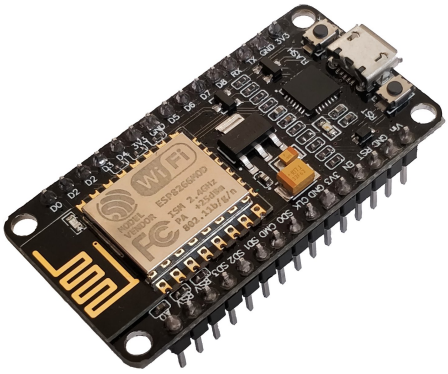
Programovanie mikrokontrolérov prebieha pomocou špecializovaných programovacích zariadení (ST-link, JTAG atd.), pričom program sa vytvára v jazykoch C/C++. Prípadné vyvíjanie programov určených práve pre tieto mikrokontroléry môže skomplikovať nedostupnosť voľne šíriteľných vývojových prostredí [10].

2.2.2 ESP8266

Platforma ESP8266 zahrňuje všetky systémy založené na čipe ESP8266EX [13]. Tento čip bol vyvinutý čínskou spoločnosťou Espressif Systems a uvedený na trh v roku 2014. Krátko na to, boli spoločnosťou Ai-Thinker vyrábané rady modulov založených na tejto platforme. Čip tvoriaci jadro celého modulu je postavený na 32-bitovom RISC procesore s inštrukčnou sadou Tensilica Xtensa L106, a v základe je taktovaný na 80 MHz s možnosťou pretaktovania až na 160 MHz. Procesor má k dispozícii 64 KB inštrukčnej pamäti RAM, 96 KB dátovej pamäti RAM a až 16 MB externej flash pamäti, ktorá sa môže líšiť v závislosti od použitého modulu. Obsahuje 10-bitový analógovo-digitálny prevodník. Disponuje 16 GPIO portami, podporuje komunikáciu pomocou zberníc I2C, I2S, SPI, UART a umožňuje bezdrôtovú komunikáciu pomocou štandardu IEEE 802.11 b/g/n Wi-Fi. Zásluhou viac ako postačujúcich parametrov a nízkej ceny sa stal relatívne rýchlo obľúbený. Veľká základňa nadšencov prispela k uľahčeniu jeho používania.

Pôvodne bola primárnym účelom modulu ESP8266 realizácia TCP/IP komunikácie s rôznymi mikrokontrolérmi po pripojení pomocou zbernice SPI alebo UART. Po vydaní sady vývojových nástrojov (Software development kit – SDK) firmou Espressif Systems, bolo možné čip preprogramovať a potom ho využívať ako plnohodnotný mikrokontrolér. ESP čipy sú programovateľné pomocou UART linky. Vzhľadom k ich dostupnosti (napr. čipy FTDI, CH340) sú tieto čipy často integrované na vývojovej doske (ako je to u NodeMCU 1.0 viď. obrázok 2.2). K programovaniu zariadenia tak stačí rozhranie USB. Napísaný kód sa po kompilácii v počítači vypáli do flash pamäte MCU.

Existuje viacero možností pre vytváranie riadiacich programov pre tieto čipy, spomedzi ktorých si vývojár môže vybrať. Jednou z nich je možnosť využiť pôvodné SDK od



Obr. 2.2: NodeMCU 1.0 obsahujúci čip ESP-12F.

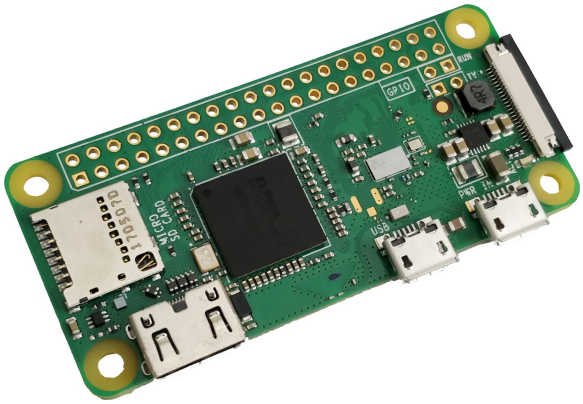
Espressif Systems. Pre vývojárov v jazyku Lua je vhodnejšie siahnuť po otvorenom projekte NodeMCU vytvoreného čínskou komunitou, ktorý umožňuje programovanie jednočipov v tomto skriptovacom jayzku. Platforma podporuje taktiež vykonávanie skriptov v jazyku Python za pomoci interpretu MicroPython. Pre začiatočníkov je najjednoduchšie využiť Arduino SDK a knižnicu pre čip ESP8266.

2.2.3 Raspberry Pi

Raspberry Pi je oproti predchádzajúcim platformám väčšinou výkonnejší a univerzálnejší. Jedná sa o nízkonákladový, jednodoskový počítač vo veľkosti kreditnej karty. Obsahuje výstup na monitor a môže byť ovládaný pomocou myši a klávesnice. Prvá verzia bola vyvinutá britskou nadáciou Raspberry Pi Foundation v roku 2012 so zameraním na výučbu programovania v jazykoch Scratch a Python, a taktiež lepšie porozumenie hardvéru. Dôraz sa kládol hlavne na podporu výučby základov informatiky na školách. Vďaka svojej malej veľkosti a dostupnej cene bol rýchlo prijatý komunitou ľudí, ktorá ho začala používať vo svojich projektoch. Na základe veľkej popularity boli neskôr vydané ďalšie, výkonnejšie modely.

Prvá verzia Raspberry Pi z roku 2012 vyšla v dvoch verziách – Model A a Model B. Jadrom obidvoch modelov je multimediálny procesor typu SoC (system-on-chip) od spoločnosti Broadcom postavený na 32-bitovej architektúre ARMv6. To znamená, že väčšina systémových komponentov, vrátane hlavného (CPU) a grafického procesora (GPU), spolu so zvukovým a komunikačným hardvérom, je integrovaná do jednej súčiastky. Jednojadrový procesor je taktovaný na 700 MHz, pričom mu sekunduje 256 MB pamäte RAM pre Model A a 512 MB pre Model B. Model B má na rozdiel od lacnejšej varianty 10/100 Mbit Ethernet a dva USB 2.0 porty. Pre zobrazovanie videovýstupu vo vysokom rozlíšení je dostupný HDMI port, pre kompatibilitu so staršími zobrazovacími zariadeniami sa využíva kompozitné video. Posledný výstup slúži pre pripojenie displeja pomocou štandardu DSI (Display Serial Interface), ktorý umožňuje pripojenie plochých displejov malých rozmerov. Vzhľadom na použitie jednoduchej architektúry s redukovanou inštrukčnou sadou je energeticky šetrnejší ako bežné stolné počítače, a aj pri plnej záťaži nemá spotrebu vyššiu ako 5 W, preto je možné ho napájať 5 V zdrojom s prúdom 1 A.

Základným operačným systémom je upravená linuxová distribúcia určená pre zariadenia typu Raspberry Pi. Novšie modely dovoľujú beh systému NetBSD, Android alebo Windows 10 IoT Core [22].



Obr. 2.3: Raspberry Pi Zero W.

V roku 2017 predstavila britská nadácia ďalší model, Raspberry Pi Zero W (obrázok 2.3), ktorý je oproti tomu klasickému omnoho menší, ale zároveň aj cenovo dostupnejší (pôvodná suma 10$). O beh systému sa stará rovnaký čip, aký je zabudovaný aj v prvej generácii, pričom takt sa zvýšil až na frekvenciu 1 GHz. Pribudla podpora pre nízko energetický bezdrôtový protokol Bluetooth, avšak chýba konektor pre Ethernetovú sieť. Model neobsahuje žiaden klasický USB port, iba dva mikro USB porty, z toho jeden je určený na napájanie a druhý podporuje technológiu OTG. So spotrebou presahujúcou tesne 1 W je zariadenie vhodné na zložitejšie IoT projekty, na ktoré nie je postačujúce ESP8266 zariadenie.

Raspberry Pi je možné využiť aj ako server, ktorý dokáže po pripojení senzora monitorovať hodnoty, ukladať ich do databázy a zobrazovať ich pomocou webovej stránky. V konečnom dôsledku sa jedná o malý počítač, avšak oproti klasickým počítačom je kvôli ARM procesoru značne obmedzený, z čoho vyplýva nekompatibilita klasických linuxových distribúcií a programov vyvíjaných pre x86 procesory.

2.3 Komunikácia

Základnou vlastnosťou IoT zariadení je konektivita. Zariadenia môžu komunikovať medzi sebou (Machine-to-Machine), často však komunikujú priamo so vzdialeným serverom, pretože tieto zariadenia sú väčšinou jednoduché a dáta sa musia ukladať a spracovávať externe. Komunikácia s okolím môže byť s využitím káblového spojenia, často sa však volí bezdrôtová varianta, pretože bezdrôtové siete sú omnoho jednoduchšie na infraštruktúru v porovnaní s káblovými. V prípadoch, kedy neprichádza do úvahy káblové, je riešením bezdrôtové spojenie.

Bezdrôtové technológie môžu pre prenášanie dát využiť rôzne frekvenčné pásma, ktoré majú následne vplyv na ich vlastnosti. Protokoly využívajúce frekvencie pod 1 GHz majú väčší dosah oproti najrozšírenejšiemu pásmu 2,4 GHz, avšak na úkor rýchlosti. Pochopiteľne, pri použití vyšších frekvencií dosah klesá a rýchlosť naopak rastie.

Internet vecí kladie špecifické požiadavky na prenos dát po sieti, a preto vznikajú nové štandardy pokrývajúce tieto nároky. Medzi tieto požiadavky je možné zaradiť snahu o zníženie energetickej náročnosti na pripojené zariadenia pri súčasnom zvyšovaní dosahu sietí. Keďže zariadenia prenášajú malé a krátke správy, nie je potrebné sa zameriavať na rýchlosť dátového prenosu.

2.3.1 Ethernet

Ethernet je najpoužívanejšia fyzická vrstva v lokálnych sieťach (LAN) štandardizovaná inštitútom pre elektrotechnické a elektronické inžinierstvo (IEEE) pod štandardom IEEE 802.3. Tento štandard deklaruje, akým spôsobom majú zariadenia v sieti formátovať dáta pre prenos na iné sieťové zariadenia. Komunikácia prebieha po medenom kábli (tzv. krútená dvojlinka) alebo po optickom kábli (v minulosti sa používali aj koaxiálne káble). V závislosti od typu kábla sú dáta vysielané rýchlosťou od 1 Mbit/s až po 100 Gbit/s. Prvá komerčná verzia bola prvýkrát predstavená v roku 1980. Na jej vývoji sa podieľali firmy DEC, Intel a Xerox. Jej popularita akcelerovala na začiatku 90. rokov minulého storočia, kedy sa stala cenovo aj výkonnostne výhodným riešením sieťovej komunikácie, a tak vytlačila sériové porty. Až na priemyselné využitie, je táto technológia v posledných rokoch postupne nahrádzaná bezdrôtovým spôsobom komunikácie.

2.3.2 Wi-Fi

Wi-Fi je súbor štandardov založených na špecifikácii IEEE 802.11, ktoré umožňujú zariadeniam bezdrôtové pripojenie do lokálnej siete WLAN (Wireless Local Area Network). Na základe ISO vrstvového modelu môžu nad sieťami WLAN a LAN pracovať rovnaké protokoly. Bezdrôtové siete prinášajú množstvo výhod, medzi ktoré patrí jednoduchosť výstavby, ale taktiež prinášajú aj nové problémy s bezpečnosťou, rýchlosťou, stabilitou, rušením, a pod.

V roku 1997 publikoval medzinárodný štandardizačný inštitút IEEE špecifikáciu, ktorá sa neustále rozširuje. Štandardy sa líšia hlavne v prenosových rýchlostiach a vlnových frekvenciách, pričom každý nesie písmenkové označenie. Dnešný najrozšírenejší štandard IEEE 802.11n pracuje na frekvencii 2.4 GHz s maximálnou teoretickou rýchlostnou až 600 Mbit/s pri využití technológie 4X4 MIMO. MIMO technológia je založená na využití viacerých vysielacích a prijímajúcich antén, čím umožňuje navýšenie rýchlosti. Najnovší bežne dostupný štandard IEEE 802.11ac využíva 5 GHz pásmo, ktoré nie je natoľko rušené ako pásmo 2.4 GHz. Pri zvyšovaní frekvencie elektromagnetických vĺn narastá možná teoretická rýchlosť, avšak kvôli fyzikálnym vlastnostiam sa jeho dosah skracuje. Vlastnosti najpoužívanejších Wi-Fi štandardov sú zobrazené v tabuľke 2.1.

Wi-Fi siete majú hviezdicovú topológiu s prístupovým bodom (Access Point - AP) ako bránou do internetu, v dôsledku čoho je potrebné, aby každé zariadenie bolo v dosahu AP. Výstupný výkon je však dostatočne veľký na to, aby pokryl vo väčšine prípadov celú domácnosť. V situáciách, kedy je signál nedostatočný, je možné pridať ďalší prístupový bod.

Implementácia Wi-Fi a TCP/IP softvéru je pomerne rozsiahla a komplexná a kladie nároky ako na výpočtový výkon, tak aj na potrebnú pamäť. Z tohto dôvodu, nebolo až do nedávnej minulosti možné, pridávať Wi-Fi pripojenie do zariadení s malým výkonom a zvyklo sa preferovať použitie inej technológie. Avšak, nové zariadenia eliminujú väčšinu režijných nákladov, a tak umožňujú bezdrôtové pripojenie k internetu aj zariadeniam s nenáročnými mikrokontrolérmi [15].

V roku 2016 bola predstavená nová verzia protokolu IEEE 802.11ah, nazvaná *Wi-Fi HaLow*, ktorá sa zameriava na cenovo výhodné bezdrôtové siete veľkého formátu, vhodné práve pre internet vecí s podporou Machine-to-Machine komunikácie. Keďže sa jedná o relatívne novú verziu, nie je oficiálne známy dosah. Z dôvodu využitia frekvencie pod 1 GHz môže sieť nadobúdať dosah okolo 1 km – pri takýchto vzdialenostiach sa už nejedná o vnútorný dosah, ale o dosah v zastavanom území.

802.11 protokol	Rok vydania	Frekvencia (GHz)	Rýchlosť (Mbit/s)	Vnútorný dosah
b	1999	2.4	11	35m
g	2003	2.4	54	38m
n	2009	2.4/5	600	70m
ac	2013	5	3466	35m
ah	2016	0.9	8	1km

Tabuľka 2.1: Štandardy Wi-Fi.

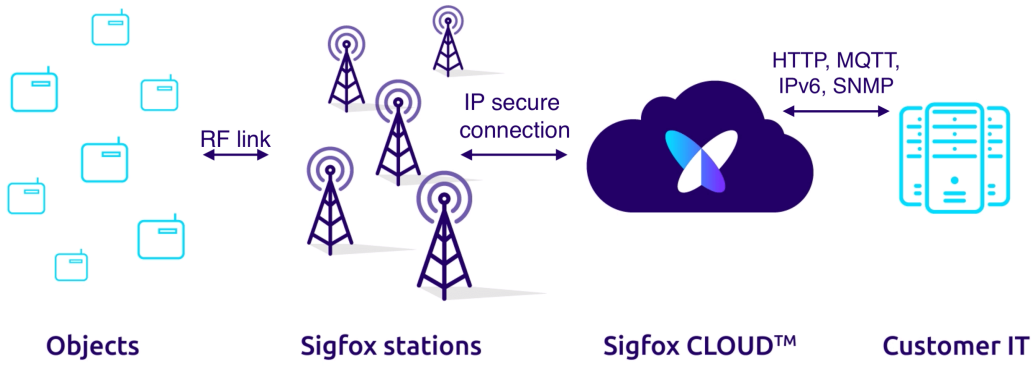
2.3.3 Sigfox

Francúzska spoločnosť Sigfox vyvinula v roku 2012 bezdrôtovú sieť určenú výhradne pre internet vecí. Pre danú sieť je charakteristický dosah, ktorý je v rádoch jednotiek až desiatok kilometrov a energetická nenáročnosť na pripojené zariadenia. Z tohto dôvodu sa Sigfox radí medzi LPWAN (Low Power Wide Area Network). Spoločnosť dosiahla tieto charakteristiky hlavne využitím veľmi úzkeho bezlicenčného pásma ISM, ktorého frekvencia sa líši v závislosti na kontinente. Zatiaľ čo sa v Európe šíri v pásme 868 MHz, v USA využíva kmitočty 902 MHz. Signál vysielaný v týchto pásmach má nielen dlhý dosah, ale je taktiež schopný voľne prechádzať cez pevné objekty s využitím minima energie. Z hľadiska topológie sa jedná o hviezdicový typ, pričom vyžaduje mobilného operátora, ktorý zaistí požadovanú prevádzku. V domácom Francúzsku si spoločnosť celoplošne vybudovala vlastnú sieť, ktorú zároveň aj spravuje. V ostatných krajinách spolupracuje s tradičnými mobilnými operátormi, ktorí nainštalujú Sigfox stanice na už vybudovanú sieť štandardných mobilných vysielačov.

Pôvodne bola sieť výlučne jednosmerná, pričom umožňovala iba odosielať dáta z koncových zariadení. Až v druhej polovici roku 2014 bola možnosť komunikácie rozšírená oboma smermi [11]. Z dôvodu čo najväčšej úspory energie koncového uzla, umožňuje táto sieť Sigfox zariadeniam odosielať správy o maximálnej veľkosti 12 bajtov. Celý prenášaný rámec spolu s dátami a réžiou sa tak vojde do objemu 26 bajtov. Pri prenosových rýchlostiach 100 až 600 b/s, ktoré sieť nadobúda, trvá zaslanie jednej správy v priemere dve sekundy. Prijímajúce správy sú obmedzené na maximálnu veľkosť 8 bajtov. V praxi táto veľkosť správ postačuje na odoslanie dvoch GPS súradníc s presnosťou na 3 m, šiestich hodnôt nameranej teploty v rozsahu od $-100°$ až do $200°$ s presnosťou na štyri tisíciny stupňa alebo 96 dvojstavových signálov. Spätná správa je dostatočne veľká na zmenu frekvencie zasielania správ alebo vypnutie/zapnutie niektorej z funkcie [20].

Pre zvýšenie kapacity siete a zvýšenie životnosti batérie limituje Sigfox počty voľne odoslaných a prijatých správ jedného zariadenia – odoslaných môže byť 140 a 4 prijaté. Po obdržaní správy Sigfox stanicou je správa odoslaná do Sigfox výpočtového centra už pomocou bežnej TCP/IP komunikácie. Tieto dáta nemajú v dátovom úložisku žiadnu pevne definovanú štruktúru a interpretácia závisí na príjemcovi. Príjemca sa k týmto dátam dostane cez softvérové rozhranie (API) zo svojej aplikácie alebo systému. Na obrázku 2.4 je možné vidieť graficky zobrazenú architektúru siete Sigfox spolu s prenosovými protokolmi.

Každé zariadenie v sieti Sigfox obsahuje jedinečný identifikátor, overovací a šifrovací kľúč. Vyššie spomínané údaje nadobudne zariadenie už pri výrobe. Pri odosielaní alebo prijímaní správ slúži jednoznačný identifikátor na adresovanie v sieti. Každá správa obsahuje kryptografický token vygenerovaný pomocou overovacieho kľúča, ktorý autentizuje odosie-



Obr. 2.4: Schéma architektúry siete Sigfox. Zdroj: [20].

lateľa. Token je založený na výpočtoch AES-128 a výsledok je skrátený na 2 až 5 bajtov. Za účelom uistenia sa, že nie sú generované žiadne kópie alebo duplicity, systém vkladá do správ sekvenčné číslo. Protokol taktiež zasiela každú správu 3-krát, zakaždým na inej vysielacej frekvencii [21].

2.3.4 LoRaWAN

LoRaWAN definuje komunikačný protokol a sieťovú architektúru založenú na fyzickej vrstve využívajúcu technológiu LoRa, ktorá je vyvíjaná LoRa Alianciou. Táto patentovaná technológia, vlastnená spoločnosťou Semtech, je založená na rádiovej modulácii využívanej pre vytváranie telekomunikačného spojenia na veľkú vzdialenosť. Z dôvodu niekoľko kilometrového dosahu patrí sieť LoRaWAN do kategórie LPWAN (Low-Power Wide-Area Networks). Okrem spomínaného dosahu, ktorý je v závislosti od terénu od 5 až do 10 km, je dôraz kladený na rýchle a cenovo výhodné rozšírenie [19].

Mnoho existujúcich sietí využíva architektúru Mash, ktorá umožňuje posielať dáta cez ostatné prvky v sieti, čím sa zvýši jej dosah, avšak tento spôsob kladie väčšie nároky na spotrebovanú energiu. Siete LoRaWAN využívajú hviezdicovú architektúru, ktorá je šetrnejšia na výdrž zariadenia – pri použití tohto typu architektúry je rovnako možné v krátkom čase pokryť veľké územia pomocou špecializovaných brán. Tieto brány využívajú nelicencované prenosové pásmo pod 1 GHz, konkrétne v Európe je to 868 MHz a 915 MHz v Spojených štátoch. Prenosová rýchlosť dát je od 0.3 Kb/s do 50 Kb/s v závislosti od použitého kanála.

Zariadenia v sieti LoRaWAN sú na základe komunikácie kategorizované do troch skupín, pričom vo všetkých troch skupinách môže byť komunikácia obojstranná. Prvá kategória, nazývaná taktiež kategóriou A, zahrňuje zariadenia zamerané na maximálne šetrenie energie z batérie. Tieto zariadenia odosielajú dáta v pomerne veľkých časových intervaloch, pričom príjem dát je obmedzený na vopred stanovené intervaly. Po odoslaní správy, zariadenie obyčajne čaká určitú dobu na prichádzajúcu správu. Zariadenia v kategórii B sú takisto napájané z batérie, avšak, na rozdiel od kategórie A, neuprednostňujú maximálne šetrenie. Z tohto dôvodu môžu zariadenia zasielať každú správu viackrát, aby minimalizovali možnosť straty dát. Zariadenia tejto kategórie dokážu prijímať dáta počas dlhších časových intervalov, prípadne môžu byť vyvolané pomocou tzv. *pingu* zo servera. Posledná kategória (typ C), zahrňuje zariadenia napájané z elektrickej siete, a preto nie je potrebné

venovať pozornosť spotrebe energie. Zariadenia sú tak celý čas v nepretržitom vyčkávacom režime a zachytávajú príchodzie správy.

Architektúra siete LoRaWAN je podobná architektúre siete Sigfox. Obe obsahujú rovnaké prvky: koncové zariadenia umožňujúce obojsmernú komunikáciu; bránu, ktorá spája koncové zariadenia so serverom; server inicializujúci komunikáciu smerom na zariadenia, ktorý zároveň ukladá dáta zo zariadení. Klienti taktiež pristupujú k svojím dátam pomocou API. Každé koncové zariadenie v sieti LoRaWAN je jednoznačne identifikované 64-bitovým identifikátorom, ktorý je priradený zariadeniu už v priebehu výroby. Adresovanie v sieti prebieha pomocou 32-bitového identifikátora zloženého zo 7-bitového čísla siete a 25-bitového čísla zariadenia, ktoré dostane pridelené v rámci aktivačnej procedúry.

LoRaWAN ponúka dve vrstvy zabezpečenia – jednu na úrovni siete a druhú na aplikačnej úrovni. Zabezpečenie na úrovni siete zaisťuje, že dáta neboli počas zasielania modifikované. Pre každú správu je vygenerovaný integritný kód správy pomocou 128-bitového AES kľúča. Zabezpečenie aplikačnej vrstvy poskytuje ochranu prenášaných údajov až do koncovej aplikácie za použitia 128-bitového AES kľúča. [16]

Kapitola 3

Spracovanie a ukladanie dát

Jedným z cieľov tejto práce je zoznámenie sa s problematikou spracovania dát v oblasti IoT senzorov. Pre väčšinu týchto senzorov je často špecifická produkcia veľkého množstva dát v pomerne krátkych časových intervaloch – je preto potrebné riešiť aj ich efektívne uskladnenie. Pre spracovávanie malého množstva dát je možné vytvoriť vlastný ukladací systém založený na súborovom systéme. Ďalšou alternatívou je využiť špecializovaný databázový sýstém, ktorý slúži výhradne na efektívne ukladanie, modifikáciu a manipuláciu s perzistentnými dátami. Tento systému pozostáva z úložiska dát (databáza, do ktorej sú ukladané štrukturované dáta) a systému riadenia bázy dát, ktorý tvorí rozhranie medzi aplikačnými programami a uloženými dátami [14]. Oproti súborovému systému je databázový systém robustnejší, ale zároveň prináša so sebou niekoľko výhod, ako napríklad:

- Relačné databázy ukladajú dáta do tabuliek, ktoré môžu byť navzájom poprepájané.
- Väčšina databázových systémov umožňuje klient-server komunikáciu pomocou štandardizovaného jazyka, čím poskytujú väčšiu flexibilitu.
- Vo väčšine databáz je pre rozšírenie funkcionality taktiež možné využiť databázové objekty, akými sú uložené procedúry, funkcie, triggery, pohľady, atď.

Súborový systém dokáže, na rozdiel od databázového, ukladať dáta efektívnejšie – z dôvodu absencie optimalizácií však pri väčšom množstve dát hrozia problémy s nedostatočnou rýchlosťou prístupu k nim. Keďže práve databázové systémy riešia rôzne optimalizačné metódy, ani pri väčších množstvách dát systém netrpí zásadným znižovaním výkonu.

3.1 Databázové systémy

Predmetom tejto kapitoly je popis dostupných databázových systémov. V prvom rade je však potrebné ozrejmiť niektoré termíny vyskytujúce sa v tejto kapitole.

Database engine. Formát úložiska dát, tvorí jadro databázového sýstému spolu so základnými obslužnými programami tohto jadra.

Databázové servery. Špecializované uzly siete zamerané na databázové operácie a funkcie systémov riadenia bázy dát. Tieto servery vykonávajú základné operácie s bázou dát, ako napríklad operácie zadávania, modifikácie, vyhľadávania dát, triedenia informácií alebo

indexácie. Ďalej poskytujú prístup k dátam pre ľubovoľné aplikačné procesy, súbežný prístup viacerých klientov, centrálnu správu, archiváciu a ochranu prístupu k informáciám. V rámci komunikácie klientov s databázovým serverom je potrebné dodržiavať štandardizované rozhranie. Najčastejšie používaný štandard je relačný jazyk **SQL** (*Structured Query Language*), ktorý umožňuje výber informácií (select) a rovnako aj aktualizáciu dát (insert, update, delete).

Transakcia. Logická jednotka programu tvorená z postupnosti operácií, ktoré čítajú alebo modifikujú dáta v databáze. Často je využívaný model transakcie, ktorý spĺňa vlastnosti označené skratkou **ACID**, ktorá je tvorená začiatočnými písmenami vlastností v anglickom jazyku.

- Atomickosť (Atomicity) transakcie zaručuje vykonanie transakcie ako celok – vykonanie všetkých čiastkových operácií alebo žiadnej z nich.
- Konzistencia (Consistency) databázy zostane po vykonaní transakcie zachovaná.
- Izolovanosť (Isolation) transakcie znamená to, keď súbežne prebiehajúce transakcie vykonávajú operácie nad dátami v databáze izolovane.
- Trvanlivosť (Durability) transakcie zaručuje po jej úspešnom vykonaní zmeny s trvalým charakterom [23].

3.1.1 MySQL

Relačná databáza MySQL sa stala jednou z najpopulárnejších databázových systémov vďaka jej bezplatnému použitiu a taktiež otvorenosti zdrojových kódov. U väčšiny webových hostiteľských služieb je dostupná už v základných balíčkoch.

Prvá verzia databázového systému MySQL bola predstavená v roku 1995. Pôvodnou myšlienkou bolo vytvoriť rýchlejšiu a jednoduchšiu databázu, ktorá by mala rovnaké aplikačné rozhranie ako konkurenčná databáza mSQL. V porovnaní s proprietárne licencovanou mSQL je MySQL licencovaná pod slobodnou licenciou GPL, čím umožňuje databázu využívať, upravovať a ďalej šíriť. Spoločnosť, ktorá stála za MySQL, bola predaná spoločnosti Sun Microsystems v roku 2008 za 1 miliardu USD. O dva roky neskôr spoločnosť Oracle kúpila Sun Microsystems za 7,4 miliardy USD[1]. Spoločnosť Oracle stojí za produkciou vlastnej rovnomennej databázy, ktorá si momentálne drží prvenstvo v rebríčku najpoužívanejších databázových systémov. Na rozdiel od MySQL je však databáza Oracle licencovaná a spoplatnená. Potom, čo došlo k prevzatiu Sun spoločnosťou Oracle, nespokojnosť bývalých hlavných vývojárov MySQL viedla k založeniu komunitnej vetvy (fork) pod názvom MariaDB. Hlavným dôvodom vytvorenia predmetnej vetvy bolo udržanie licencie slobodného softvéru GPL, zlepšenie kvality a zrýchlenie vývoja [9].

3.1.2 Porovanie MySQL a MariaDB

Na úvod je potrebné uviesť, že ide o veľmi podobné databázy, keďže MariaDB je postavená na otvorenom zdrojovom kóde MySQL, na ktorom začali vývojári ďalej budovať. Akonáhle bola pôvodná databáza obohatená o nové vlastnosti alebo bezpečnostné záplaty, vývojári zmeny aplikovali aj do komunitnej vetvy.

[1] http://www.oracle.com/us/corporate/press/018363

Pri inštalácii MariaDB je možné si povšimnúť zmeny v názvoch inštalačných balíčkov, avšak následne po inštalácii sa názvy nainštalovaných súborov zhodujú a dokonca sú uložené na rovnakej lokácii. V dôsledku toho nie je možné jednoducho nainštalovať súbežne obe databázy.

Používanie databázy MariaDB je úplne rovnaké ako pri MySQL. Obe podporujú rovnaký jazyk SQL, rovnaké príkazy a konfiguračné súbory. Databázy využívajú rovnaké porty, sokety, klientské programové rozhranie. V prípade, že aplikácia využíva MySQL, bez úprav zvládne používať aj komunitnú vetvu tejto databázy. Po vydaní verzie 10.0 databázového systému MariaDB boli zmeny natoľko výrazné, že došlo k odkloneniu od úplnej kompatibility medzi oboma databázami [9].

Napriek značným podobnostiam oboch databáz má MariaDB vďaka komunite nové vlastnosti, vylepšenia výkonu, lepšie testovanie a skoršie (transparenté) opravy chýb. Podporuje viac úložných formátov ako napríklad OQGRAPH na spracovanie zložitých grafov, SphinxSE úložný formát kompatibilný s Sphinx vyhľadávacím formátom úložiska alebo formát úložiska dát Cassandra, ktorý dovoľuje pripojenie k bežiacemu clusteru NoSQL databázy Cassandra a prácu s dátami pomocou SQL. Formát Spider umožní škálovať server pomocou tzv. shardingu, teda rozdelí dáta do viacerých uzlov, z ktorých každý obsahuje vlastnú časť dát. Zaujímavou vlastnostnou môže byť podpora dynamických stĺpcov, ktoré umožnia uložiť viaceré hodnoty do jedného stĺpca. MariaDB ponúka model rolí, kde môže administrátor priradiť roliam rôzne práva a následne zaradiť užívateľov do týchto rolí [4].

3.1.3 Databázové formáty úložisk

V tejto časti budú popísané najpoužívanejšie formáty úložisk databáz MariaDB a MySQL.

InnoDB je univerzálny, v základe predvolený, databázový formát používaný oboma systémami. Databázové jadro vyvažuje vysokú spoľahlivosť a vysoký výkon pri dodržaní ACID modelu. Formát je založený na transakčnom spracovaní, podporuje príkazy COMMIT a ROLLBACK. V prípade nečakaného výpadku je systém schopný úplného zotavenia, čím zabezpečuje integritu. InnoDB podporuje fulltextové indexy, zámky riadkov tabuľky a konzistentné čítanie pre viacužívateľský prístup. Ďalej podporuje cudzie kľúče, čo v prostredí relačných databáz znamená integritné obmedzenie tvorené zo stĺpca odkazujúcej tabuľky, so stĺpcom odkazovanej tabuľky.

MyISAM bol až do verzie 5.1 východzí formát úložiska databázového systému MySQL, neskôr nahradený formátom InnoDB. Keďže sa jedná o jeden z najstarších úložných formátov, momentálne už nie je aktívne vyvíjaný, ale aj naďalej je stále dostupný, ako aj v MySQL, tak aj v systéme MariaDB. MyISAM je náhrada za staršie jadro ISAM (Indexed Sequential Access Method). Ľahký a zároveň vysoko výkonný databázový formát nepodporuje transakčné spracovanie ani cudzie kľúče. V prípade potreby je ľahko prenositeľný medzi systémami. Na rozdiel od InnoDB MyISAM používa zámky na úrovní celých tabuliek, nie iba riadkov, ale aj napriek tomu dokáže súbežne vykonávať príkaz SELECT počas vkladania hodnôt do tabuľky. V tabuľkách je možné vytvárať indexy na prvých 500 znakoch stĺpcov TEXT a BLOB[2] a podporované sú aj fulltextové indexy [6].

Aria bol prvýkrát predstavený v roku 2007 s cieľom vytvoriť východzí tabuľkový formát pre transakčné aj netransakčné spracovanie pre systémy MariaDB a MySQL. Momentálne

[2] binary large format - typ stĺpca slúžiaci na uloženie binárnych dát

ide o alternatívu k formátu MyISAM, ktorého vývoj bol pozastavený. Oproti MyISAM má Aria zabezpečené dáta a indexy v prípade pádu systému, podporuje viacnásobné súbežné vkladanie do rovnakej tabuľky, v budúcnosti sa počíta s transakčným spracovaním a podporou ACID. Veľkou výhodou je možnosť zopakovania akcie z logovacieho súboru. Z toho dôvodu je postačujúce vytvárať zálohy iba z tohto súboru.

3.1.4 NoSQL databázy

V dnešnej dobe sa stávajú čoraz viac populárne NoSQL databázové systémy. Pojem NoSQL vznikol už na konci 90. rokov, kedy označoval relačné databázové systémy, ktoré nepoužívali relačný jazyk SQL. Od roku 2009 sa tento pojem začal znovu používať, ale tentokrát označuje nerelačné databázové systémy. Oproti relačným databázam, ktoré ukladajú dáta do tabuliek, sú NoSQL databázy postavené na mierne odlišnom princípe. Väčšinou sa jedná o úložisko typu kľúč-hodnota, na ktorom sú postavené rôzne typy modelov.

Databázy založené na jednoduchom modeli typu **kľúč-hodnota** patria k veľmi efektívnym. Dáta sú zvyčajne jedného z dátových typov prevzatých z programovacích jazykov, prípadne objektu a pozostávajú z dvoch častí – reťazca reprezentujúceho kľúč a zo skutočných dát, označovaných ako hodnota, čím vytvárajú spomínaný pár. Tento spôsob ukladania umožňuje ukladať neštrukturované dáta, keďže tieto databázy nemajú striktne definovanú schému dát. Model vychádza z tabuľky s rozptýlenými položkami, kde je jedinečný kľúč použitý ako index ukazujúci na hodnotu. Aj vďaka tomu sú dotazy nad týmto typom databáz rýchlejšie ako u relačných databáz. Na druhej strane, trpia absenciou ad-hoc dotazovacích a analytických funkcií, ako aj funkciou spojenia (*JOIN*) a agregačných operácií. NoSQL databázy preferujú vysokú škálovateľnosť, flexibilitu a rýchle vyhľadávanie na úkor udržania konzistencie dát. Medzi najpoužívanejšie databázy tohto typu sa radia databáza Redis, ktorá primárne uchováva dáta v pamäti, a taktiež databáza *DynamoDB* od spoločnosti Amazon, ktorá je využívaná hlavne v *AWS* (Amazon Web Services).

Dokumentovo orientované databázy ukladajú dáta v dokumentoch, ktoré sú reprezentované pomocou štandardných formátov ako *JSON*, *XML*, *PDF* atď. Keďže NoSQL databázy nemajú pevne danú schému, štruktúra dokumentov môže byť pre každý dokument jedinečná. Dokumenty sú adresované pomocou jedinečného kľúča, ktorý máva podobu jednoduchého reťazca alebo reťazca obsahujúceho cestu k dokumentu. Najznámejšou dokumentovo orientovanou databázou je *MongoDB*, ktorá dokáže zabezpečiť konzistenciu, odolnosť voči poruchám, perzistentnosť a ponúka agregačné funkcie, ad-hoc dotazy, indexovanie a automatické rozloženie zátaže naprieč viacerými systémami. *MongoDB* ukladá dáta v dokumentoch typu *BSON* (binárny *JSON*), ktorý pozostáva z usporiadaného zoznamu prvkov obsahujúcich názov, dátový typ a hodnotu.

Stĺpcové databázy sú podobné relačným databázam, keďže princíp riadkov a stĺpcov je stále zachovaný. Pred vkladaním dát je nutné definovať druhy stĺpcov, čím je dopredu známa štruktúra databázy. Dáta sú tak ukladané ako úseky stĺpcov. Namiesto ukladania údajov po riadkoch pre rýchly prístup sú údaje organizované pre rýchle stĺpcové operácie. Preto tento model využívajú hlavne analytické systémy. Tento model vo veľkej miere využíva Google s vlastnou databázou *Big Table*, ktorá poskytuje dáta aplikáciám ako Gmail, YouTube či Google Earth.

Grafové databázy ukladajú dáta vo forme grafu, ktorý sa skladá z uzlov a hrán. Uzol predstavuje objekt, ktorý môže nadobúdať vlastnosti a hrana definuje vzťahy medzi uzlami. Hrany sú reprezentované v databáze tak, že každý uzol obsahuje ukazovatele na jeho susedné uzly, čím je možný rýchly pohyb po grafe. Grafové databázy nemajú schému a ich hlavný dôraz je kladený na rýchle prepojenie dát. Dotazy nad databázou predstavujú prechádzanie grafu. Ako jediný NoSQL model dokáže zabezpečiť *ACID* vlastnosti. Tento typ databáz je využívaný hlavne v rámci sociálnych sietí, softvérov zameraných na ponuku odporúčaní, vedomostných grafov a mnohých ďalších. Keďže tento typ databáz je vhodný iba v rámci určitých špecifických prípadov, nie je tak rozšírený ako ostatné NoSQL databázy. Najznámejšia grafová databáza je Neo4j.

Objektovo orientované databázové systémy sú najmenej využívané spomedzi vyššie spomenutých. Pri týchto databázach sú dáta ukladané a reprezentované ako objekt, ktorý je podobný objektom z objektovo orientovaných jazykov (OOP). Objekty ponúkajú všetky funkcie a vlastnosti z OOP ako zapúzdrenie dát, polymorfizmus a dedičnosť. Tieto databázy sú využívané v aplikáciach, ktoré zahŕňajú komplexné vzťahy objektov alebo pri meniacich sa štruktúrach objektov.

V porovnaní s relačnými databázami, ktoré využívajú jazyk SQL, NoSQL databázy nemajú jednotný dotazovací jazyk. Pri dotazovaní sa vo väčšine prípadov nepoužíva jazyk SQL, ale rozhodujúca časť databázových systémov ponúka vlastné dotazovacie spôsoby/jazyky. Preto je pre užívateľov migrácia z jedného databázového systému do druhého náročná, rovnako ako aj následná údržba. Nie všetky NoSQL databázy dokážu zabezpečiť *ACID* vlastnosti. Na druhej strane, NoSQL databázy ponúkajú široké možnosti výberu databázového modelu a jednoduché škálovanie. NoSQL databázy dokážu poskytnúť veľkú mieru flexibility, rýchlejšie a efektívnejšie databázové operácie [18].

3.2 Vzdialené riešenie

V sekcii 3.1 boli spomenuté databázové systémy, ktoré slúžia na ukladanie dát. Takéto systémy môžu byť prevádzkované na dvoch typov systémov - lokálne alebo vzdialené. Každý zo systémov ma svoje špecifické vlastnosti, ktoré budú rozobrané v následujúcich častiach.

3.2.1 Vzdialené výpočtové centrá

Riešením ako prevádzkovať systém využívajúci databázový server je použitie vzdialeného výpočtového centra – cloud computing, čím sa označuje spôsob dodávania výpočtových služieb akými sú servery, úložiská, databázy a iný softvér, cez internet. Poskytovatelia centier ponúkajú tieto služby zákazníkom za poplatok, ktorý sa odvíja od ich používania. Zákazníkom tak odpadajú problémy s nákupom, inštaláciou a správou hardvéru a softvéru. Zároveň výpočtové centrá ponúkajú škálovateľnosť zdrojov podľa potreby.

Existujú 3 hlavné kategórie služieb ponúkaných poskytovateľmi vzdalených výpočtových centier. Tieto kategórie služieb sú postavené v hierarchickej závislosti (jedna na druhej) a tvoria tak zásobník ponúkaných služieb výpočtových centier.

Infraštruktúra ako služba (IaaS – Infrastructure as a Service) ponúka zákazníkom výpočtové zdroje, ktoré môžu byť využité na rôzne účely. Táto architektúra je využívaná v prípadoch, keď zákazník nemá záujem o nákup a správu vlastných fyzických serverov.

Prostriedky sa ponúkajú ako samostatné servisné komponenty, ktoré sú prenajímané na určitú dobu. Poskytovatelia služieb zabezpečujú samotnú infraštruktúru, zatiaľ čo zákazníci nakupujú, inštalujú, konfigurujú a spravujú svoj vlastný softvér – operačné systémy, middleware[3] a aplikácie [17]. Medzi typické oblasti využívajúce IaaS patria:

- **Hostovanie webov.** Ktoré môže byť lacnejšie, než klasické prevádzkovanie na webových hostiteľských službách.
- **Webové aplikácie.** IaaS poskytuje všetky potrebné prostriedky pre podporu webových aplikácií – úložisko, webové a aplikačné servery, sieťové prostriedky. Organizácie prevádzkujúce svoje aplikácie vedia infraštruktúru flexibilne škálovať podľa potreby aplikácie.
- **Úložisko, zálohovanie a obnovenie.** V prípade využitia infraštruktúry ako služby je možné sa vyhnúť počiatočným nákladom a zložitej správe úložiska. IaaS je z toho dôvodu výhodnejší pri postupnom raste potrieb úložiska.
- **Prostredie HPC[4].** Rieši komplexné problémy ako sú simulácie vesmíru či rozkladu proteínov za použitia superpočítačov, gridovej architektúry alebo na výpočtových clusteroch.
- **Analýza veľkého objemu dát.** Pri skúmaní veľkého množstva dát (big data) je možné získať potenciálne užitočné vzorce, trendy a asociácie. Avšak na túto analýzu je potrebné veľké množstvo výkonu spolu s veľkou úložnou kapacitou, ktoré môže IaaS poskytnúť za akceptujúcich podmienok.

Platforma ako služba (PaaS – Platform as a Service) je ďalším modelom poskytovania aplikácií. Podobne ako IaaS zahrnuje infraštruktúru, ktorú ďalej obohacuje o všetky potrebné nástroje nutné k vytváraniu aplikácií. Model PaaS je navrhnutý tak, aby podporoval celý životný cyklus webovej aplikácie, medzi ktoré patrí: návrh, vývoj, testovanie, implementácia, správa a aktualizácie. Okrem týchto služieb ďalej ponúka integráciu s webovými službami, integráciu databáz, správu verzií, podporu tímovej spolupráce a mnoho iných. Týmto je možné uľahčiť vývoj aplikácie, a tým sa vyhnúť nákupu a spravovaniu softwarových licencií, podpornej aplikačnej infraštruktúry, nástrojov pre vývoj a ďalších prostriedkov. Okrem vyvíjanej aplikácie spravuje všetko ostatné poskytovateľ služby. V minulosti bola nevýhodou tejto koncepcie chýbajúca schopnosť systémov rôznych poskytovateľov vzájomne spolupracovať, z čoho vyplýval problematický prenos celého systému k inému poskytovateľovi [8]. Dnes je možné tento problém vyriešiť použitím linuxových kontajnerov[5].

Softvér ako služba (SaaS – Software as a Service) je ďalším modelom. Zákazníkovi sú poskytované úplné softwarové riešenia ponúkané ako internetová služba. Preto nie je úlohou zákazníka spravovať software a zabezpečovať jeho podporu, avšak musí akceptovať všetky prípadné zmeny vykonané poskytovateľom služieb. Ten okrem opráv a aktualizácií softvéru spravuje aj hardvér a udržiava infraštruktúru. Na rozdiel od bežného softvéru si zákazník za službu platí pravidelne, čím sa stáva tento koncept pre dodávateľov atraktívnejší, keďže im poskytuje trvalý zdroj príjmov.

Typicky ponúkaným softvérom sú webové e-mailové služby, ktoré sú zväčša dostupné bezplatne, avšak to platí iba pre osobné použitie. Práve preto organizácie využívajú služby

[3] je špecializovaný softvér poskytujúci aplikáciám služby, ktoré sú nad rámec operačného systému
[4] high-performance computing
[5] technológia umožňujúca zabaliť a odizolovať aplikácie spolu s ich prostredím potrebným na prevádzku

SaaS a prenajímajú si kancelárske aplikácie, akými sú e-mail, nástroje pre spoluprácu, videokonferencie, kalendáre, obchodné aplikácie, správa dokumentov a mnohé ďalšie. Organizácia so špecifickými požiadavkami nemusí nájsť vhodnú aplikáciu, ktorá by bola dostupná v tomto modeli. Koncepcia SaaS čelí ďalším problémom, ktoré prinášajú aplikácie typu open-source. Tie si môžu organizácie nainštalovať na vlastný hardvér, ktorý je čoraz lacnejší a výkonnejší [8].

3.2.2 Špecializované služby pre IoT

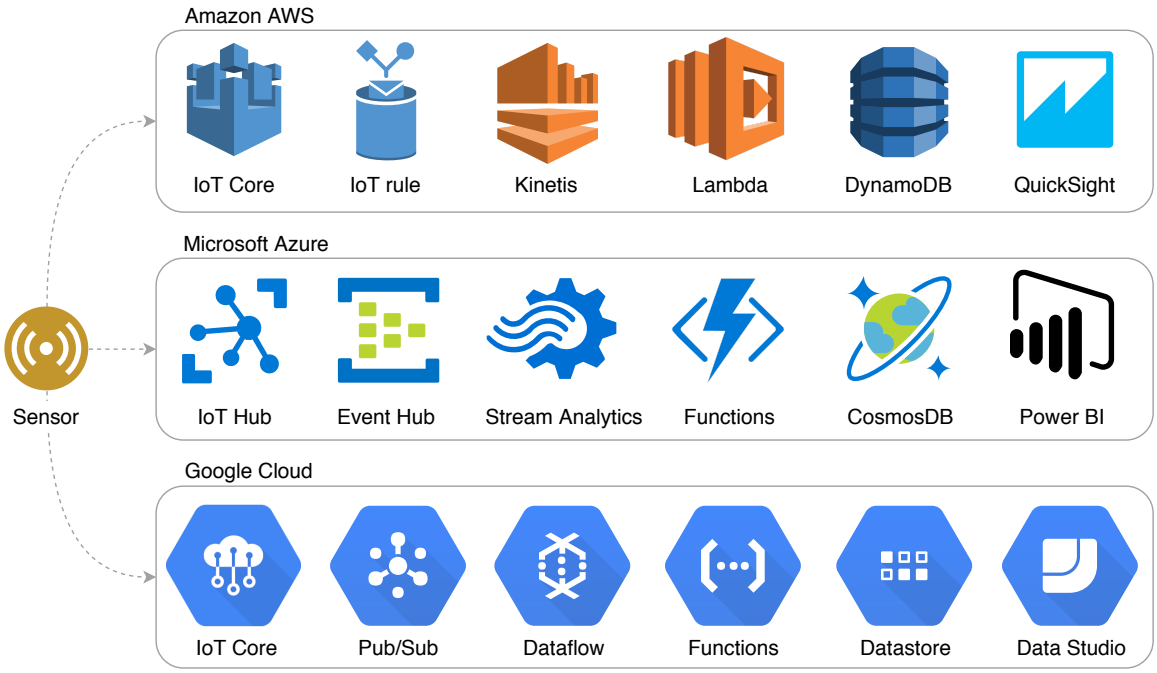
Výpočtové centrá ponúkajú špecializované platformy zamerané pre internet vecí. Tieto platformy, ako služby, umožňujú pripojenie a správu zariadení komunikujúcich so serverom zabezpečeným spôsobom, pri použití rôznych protokolov. Následné sú dáta spracované a uložené do databázy. Ďalej služba ponúka použitie analytických nástrojov nad zaznamenanými dátami. Výsledky zaznamenaných dát a analýz je možné vizualizovať pomocou prehľadných grafov rôznych typov. Všetky spomenuté operácie dokážu prebiehať v reálnom čase. Z pohľadu vývojára je jednoduchšie použiť takto predpripravenú službu, ako vyvíjať celé riešenie od základov [5].

V súčasnej dobe je na trhu množstvo poskytovateľov vzdialených služieb, ktorí ponúkajú platformy pre internet vecí. Platformu s najdlhšiou históriou vlastní a ponúka spoločnosť Amazon – *AWS IoT Core* [1]. Vďaka dlhoročným skúsenostiam sa stala jednou z najobľúbenejších platforiem súčasnosti. Ďalšie známe platformy pre internet vecí ponúkajú aj spoločnosti ako Microsoft – *Microsoft Azure IoT*, IBM – *Watson IoT* na platforme *IBM Bluemix*, Google – *Google Cloud IoT*. Všetky platformy obsahujú aj množstvo dalších služieb, ktoré môžu zákazníci využívať.

Pri vytváraní IoT aplikácie je postup podobný pri všetkých platformách až na malé odlišnosti. V prvom prípade je potreba spustiť IoT službu, ktorá umožňuje spravovať senzory a zároveň je vstupnou bránou dát z pripojených zariadení. Po spustení služby je možné zaregistrovať zariadenia, pre ktoré sú vygenerované jedinečné kľúče (tokeny), ktoré slúžia na autentifikáciu zariadenia. Následne je možné zahájiť komunikáciu pomocou jednoduchých a nenáročných protokolov ako *MQTT*, *AMQP*, ktoré sú populárne pre internet vecí alebo využiť bežné *HTTP* alebo *HTTPS* spojenie [3]. V ďalšom kroku je potrebné vytvorenie služby, ktorá bude komunikovať s databázou. Každá z vyššie spomenutých platforiem ponúka iné databázové systémy, pričom väčšina z nich poskytuje aj vlastný databázový systém. Amazon ponúka *DynamoDB*, Microsoft zase *Cosmos DB*, Google – *Cloud Datastore* a IBM databázu *Cloudant*. Okrem vlastných databáz všetky platformy ponúkajú na výber aj bežne dostupné databázové systémy ako *MongoDB*. Všetky spomenuté databázy sú typu NoSQL, avšak platformy ponúkajú aj klasické relačné SQL databázy.

Dáta sú po prijatí odovzdávané pomocou zreťazenej linky do rôznych služieb. Jednoducho je možné po prijatí správy vykonať funkciu na dekódovanie obsahu, následne vykonať podľa pravidla príslušnú akciu alebo analyzovať prichádzajúci tok dát. Po získaní sledovaných dát sú následne uložené do preferovanej databázy a ďalej zobrazované vo webovej vizualizácii [2]. Obrázok 3.1 znázorňuje služby využívané platformami Amazon AWS a Microsoft Azure pre prijatie dát, vykonanie analýzy nad prijatým prúdom dát, následným uložením do databázy a zobrazením v prehľadných nástenkách a grafoch.

Cena služieb sa odvíja od ich použitia, preto je potrebné špecifikovať, aké služby bude aplikácia využívať, a tomu prispôsobiť výber platformy a celkový návrh aplikácie. Porovnávanie cien služieb medzi platformami je náročné, pretože zákazník často netuší, aké služby a v akom množstve bude v rámci hotovej aplikácie využívať. Poskytovatelia služieb vypočí-



Obr. 3.1: Ukážka zreťazenej linky dát. Zdroj: [1, 5, 2].

tavajú cenu až v priebehu fungovania aplikácie – cena sa teda odvíja od používania aplikácie, čo nie je možné vopred presne vykalkulovať. V prípade, že by sme chceli porovnať cenu za prenesený gigabajt zo zariadení do výpočtového centra, AWS a Azure nám ponúkajú iba ceny za fixný počet správ, pričom AWS zohľadňuje aj dobu pripojenia zariadenia. Preto AWS ponúka miliardu 5 KB správ za 1 € a 8 centov za milión pripojených minút na mesiac[6]. Azure ponúka 2 rady IoT služieb. V prvej, lacnejšej, nacenil denne 400 000 4 KB správ za 8,4 € na mesiac a v drahšej za rovnaký počet správ 21 €. Najzásadnejším obmedzením lacnejšej varianty je zamedzenie možnosti komunikácie smerom zo serveru na zariadenie. Dôležité je podotknúť, že Azure ponúka aj 8000 0.5 KB správ denne zadarmo, čo vychádza v prepočte necelých 125 MB správ mesačne[7]. Google ponúka prvých 250 MB v mesiaci zadarmo a po prekročení tejto hranice si účtuje v prepočte 3,64 € za GB[8]. IBM ponúka prvých 200 MB zadarmo a za každých další MB 0,000965 €, čo vychádza na 0,965 € za GB[9].

Pre funkčnosť aplikácie je okrem spomenutej služby pre prijímanie dát zo zariadení potrebné pripočítať taktiež cenu za databázu, logiku aplikácie a vizualizačnú časť.

3.2.3 Webové služby

V prípade vývoja webovej aplikácie, ktorá pre svoj beh vyžaduje webový server s databázou prístupný z celého internetu, avšak nepožaduje pokročilé nástroje vyššie spomínaných vzdialených služieb, je možné využiť webové hostiteľské služby. Jedná sa o poskytovanie priestoru pre fungovanie webovej stránky na cudzom serveri. Tieto servery ponúkajú možnosť využitia skriptovacieho jazyka PHP a využitia SQL databázy, čím umožňujú prevádz-

[6] https://aws.amazon.com/iot-core/pricing/
[7] https://azure.microsoft.com/en-us/pricing/details/iot-hub/
[8] https://cloud.google.com/iot/pricing
[9] https://www.ibm.com/internet-of-things/spotlight/watson-iot-platform/pricing

kovať nielen webové stránky, ale aj zložitejšie webové systémy, prípadne aplikácie. Väčšina poskytovateľov ponúka spolu s priestorom aj e-mailovú schránku. Obsah webovej stránky je tak dostupný z domény, ktorú je potrebné vopred vlastniť.

Medzi troma poskytovateľmi (A, B, C) s veľkým zastúpením na českom trhu býva samozrejmosťou poskytovanie skriptovacieho jazyka PHP vo verzii 7.0, databáz MySQL ako aj MariaDB a prevádzkovanie celého systému vrátane databáz na rýchlych SSD diskoch s neobmedzeným počtom prenesených dát. Rovnako je samozrejmosťou poskytovanie e-mailovej schránky, plánovač akcií (*CRON*) atď. Poskytovatelia sa väčšinou líšia práve v poskytovaných kapacitách pre jednotlivé služby ako je maximálna veľkosť databázy, e-mailovej schránky, počet a interval plánovaných úloh, poprípade poskytujú ďalšie doplnkové služby. Poskytovatelia ponúkajú rôzne typy balíčkov, ktoré sú naškálované tak, aby drahší balíček ponúkal vyššie kapacity vymenovaných služieb. Poskytovateľ A ponúka za 1,09 €, okrem spomínaných vlastností, databázu s maximálnou kapacitou 1 GB a 3 naplánované úlohy s minimálnym intervalom jednej hodiny. Bez započítania ceny domény je možné prevádzkovať webovú aplikáciu za približne 14 € na rok. Poskytovateľ B ponúka 5 GB databázového priestoru, pričom jedna databáza môže mať najviac 1 GB. Taktiež ponúka 10 GB úložného priestoru a ďalších 10 GB pre e-mailovú schránku. Plánovač udalosti môže byť spúštaný iba v prednastavených intervaloch – 2 hodiny, deň, týždeň, mesiac. Oproti poskytovateľovi A umožňuje automatickú publikáciu zmien pomocou verziovacieho systému GIT, avšak službu spoplatňuje na 4 € mesačne, čo predstavuje necelých 48 €. Tretí poskytovateľ ponúka za 1,65 € 1 GB priestoru, 5 GB priestoru pre e-maily, 10GB pre web a 5 *CRON* úloh, čo predstavuje necelých 20 € ročne.

Okrem ceny poskytovania webových služieb je potrebné pripočítať registráciu domény, ktorá sa líši v závislosti od typu. V dnešnej dobe je možné získať na rok doménu typu `.eu`, za cenu 1,75 € ročne.

3.2.4 Služby virtuálneho serveru

Virtuálny privátny server (VPS) je server, ktorý beží vo virtualizovanom prostredí prevádzkovanom hostiteľským poskytovateľom. Oproti prevádzkovaniu na systémoch bežnej webovej hostiteľskej služby, virtuálny privátny server dovoľuje prevádzku ľubovolného operačného systému s ľubovolným softvérom. Pre každý virtualizovaný server je vyhradený vopred dohodnutý výkon, čím je prevádzkovanie služieb lepšie odizolované od ostatných zákazníkov ako pri webových hostiteľských službách, na druhú stranu sú služby virtuálneho serveru finančne nákladnejšie na prevádzku a skôr sa tak podobajú IaaS službám vzdialených výpočtových centier. Tie narozdiel od VPS vedia flexibilne škálovať požadovaný výkon za cenu vyšších prevádzkových nákladov [7].

3.3 Lokálne systémy

Alternatívnym riešením ku vzdialenému systému môže byť lokálny systém, kedy si záujemca zadováži a ďalej prevádzkuje server pripojený do vlastnej siete pre vlastné účely. Typickými predstaviteľmi takýchto záujemcov sú veľké spoločnosti a organizácie, ktoré potrebujú vysokú mieru zabezpečenia dát, alebo organizácie s častými problémami s konektivitou. Pri obstarávaní lokálneho systému je nutné myslieť na vysoké počiatočné náklady spojené s obstaraním hardvéru, ktoré je možné do istej miery minimalizovať jeho prenajatím. Pre spoľahlivú a bezproblémovú funkčnosť systému je nevyhnutné zabezpečiť pravidelnú údržbu, ktorá je však spojená s ďalšími nákladmi. V situáciach, kedy nepostačujú aktuálne výpoč-

tové kapacity, nie je možné okamžite zvýšiť výkon, a preto je jediným možným riešením dokúpenie nového hardvéru. Odhliadnuc od spomenutých nevýhod, takéto riešenie prináša plnú kontrolu nad systémom, akú nie je možné docieliť pri vzdialených systémoch, a preto je jednoduchšie zabezpečiť lepší výkon s ohľadom na požiadavky systému. Lokálne systémy ponúkajú lacnejšiu cenu za diskovú kapacitu, lacnejšie softvérové licencie[10].

[10] platí pri licenciách, ktorých cena sa odvíja od množstva použitých jadier

Kapitola 4

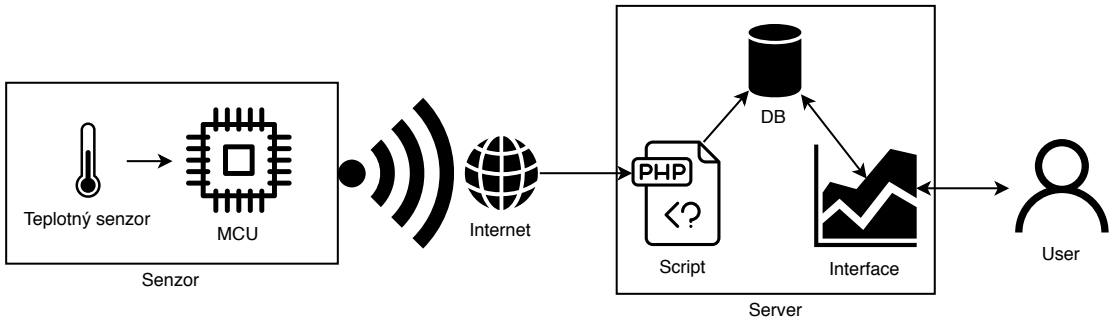
Návrh systému

V rámci návrhu systému je potrebné dodržať požadované špecifikácie. Keďže je systém vyvíjaný ako otvorený projekt zameraný pre počítačových nadšencov, už od začiatku vývoja je kladený dôraz na čo najmenej finančne náročnú prevádzku systému. Z toho dôvodu je potrebné preveriť možnosti prevádzkovania takéhoto systému. Posudzované sú možnosti prevádzky na nasledujúcich systémoch.

- Vzdialené výpočtové centrum
- Špecializovaná služba vo výpočtovom centre
- Vzdialený server poskytujúci priestor pre webové stránky
- Vzdialený virtuálny privátny server
- Lokálny server pod vlastnou správou

Na základe analýzy cien je zvolnený vývoj systému, ktorý je kompatibilný s možnosťami prevádzkovania na serveroch hostiteľských služieb, keďže práve tie poskytujú najlacnejšie, ale zároveň najobmedzenejšie služby. V takomto prípade je vytvorený sýstém zároveň kompatibilný aj s prevádzkou na lokálnych serveroch (Raspberry Pi) ako aj na vzdialených serveroch a centrách.

Návrh celého systému je možné rozdeliť do niekoľkých menších častí: návrh zariadenia snímajúceho teplotu, návrh komunikačného protokolu používaného pri zasielaní nameraných teplôt na server, návrh databázového systému a návrh uživateľského rozhrania. Logické prepojenia častí sú zobrazené na obrázku 4.1.



Obr. 4.1: Schéma navrhovaného systému.

4.1 Návrh senzoru

V prvej časti je potrebné navrhnúť zariadenie, ktoré by v pravidelných časových intervaloch zaznamenávalo okolitú teplotu a následne ju zasielalo na vybraný server. Takéto zariadenie musí byť schopné bezdrôtového pripojenia do internetu a zároveň musí poskytovať vstupné/výstupné piny potrebné na pripojenie teplotného senzora.

Napriek nízkej nákupnej cene spĺňa platforma ESP8266 všetky vymenované vlastnosti. Existujú rôzne varianty tohto čipu, odlišujúce sa počtom pinov a formou plošného spoja, pričom všetky varianty sú dostačujúce na vykonávanie spomenutej činnosti. Pre zjednodušenie programovania čipu je možné využiť vývojárske sady (*NodeMCU*, *Wemos D1*), ktoré okrem samotného čipu obsahujú zabudovaný programátor, ktorý uľahčuje nahrávanie programu na samotný čip. V prípade zamerania sa na inú bezdrôtovú technológiu, ktorá je energeticky úspornejšia, je možné využiť mikrokontrolér spolu so Sixfox modemom. V takomto prípade sú vyššie obstarávacie aj prevádzkové náklady.

Pre monitorovanie teploty je potrebné k zariadeniu pripojiť teplotný senzor. Pri výbere senzora je potrebné zhodnotiť požadované vlastnosti ako presnosť, rozsah a typ komunikácie. V prípade merania teploty vzduchu v budovách, nie je potrebný veľký rozsah, rovnako nie je potrebná presnosť merania väčšia ako na 1 desatinné miesto. Pri týchto požiadavkách je vhodné použiť cenovo výhodný teplotný senzor `DALLAS DS18B20`, ktorého presnosť sa pohybuje okolo $\pm 0,5°$. V prípade rozšírenia o ďalšiu meranú veličinu – vlhkosť – je možné použiť teplotný senzor z rady `DHT`. Lacnejšia varianta `DHT11`, ktorá je mierne drahšia oproti `DS18B20`, meria teplotu s presnosťou iba $\pm 2°$. Drahšia, a zároveň presnejšia verzia `DHT22`, dokáže merať teplotu s presnosťou $\pm 0,5°$, v praxi je možné dosiahnuť presnosť okolo $\pm 0,2°$.

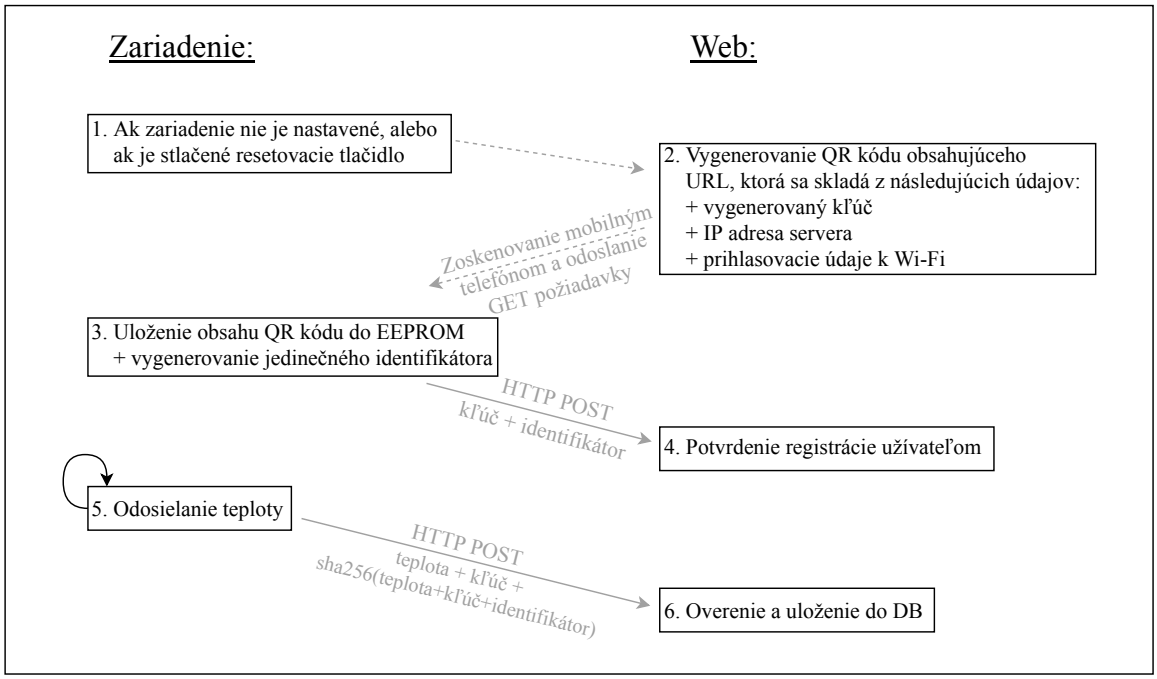
Okrem teplotného senzora je vhodé vybaviť zariadenie tlačidlom, ktoré slúži na uvedenie zariadenia do konfiguračného stavu. Ak zariadenie ešte nebolo nakonfigurované, prechod do tohto stavu je zabezpečený automaticky. V tomto stave sa zo zariadenia stáva Wi-Fi prístupový bod, na ktorom beží webový server, ktorý očakáva konfiguračné dáta zadané HTTP metódou GET na konkrétne URL. Táto metóda je zvolená z dôvodu, aby bolo možné dostať dáta do zariadenia iba pomocou odkazu získaného z QR kódu.

4.2 Návrh komunikačného protokolu

Pre správne interpretovanie odoslaných dát zo senzorov na server je potrebné, aby obe strany boli schopné interpretovať použitý komunikačný protokol. Zariadenia internetu vecí zväčša využívajú jednoduché a nenáročné protokoly, ktoré sú určené hlavne pre zariadenia s malým výkonom. Jedná sa o protokoly *MQTT* (Message Queuing Telemetry Transport) alebo *AMQP* (Advanced Message Queuing Protocol) založené na tzv. *"publish and subscribe"*[1] systéme. Avšak, za účelom prevádzky navrhovaného systému na webovej hostiteľskej službe nie je možné použiť spomínané protokoly z dôvodu ich absencie. Za účelom dosiahnutia kompatibility so všetkými typy serverov je nutné využiť komunikačný protokol *HTTP*. Protokol *HTTPS* nie je na komunikáciu medzi senzorom a serverom využitý z dôvodu veľkej réžie pre monitorujúce zariadenie.

Z dôvodu použitia nezabezpečeného protokolu je potrebné navrhnúť spôsob komunikácie, ktorý by možnému útočníkovi minimalizoval zneužitie systému. Pre tento účel je navrhnutý mechanizmus registrácie, pri ktorom senzor odošle spolu s vopred vygenerovaným kľúčom jedinečný identifikátor. Tento identifikátor, ktorý je zaslaný iba v registračnej

[1] Schéma zasielania správ, kde odosielatelia správ nezasielajú správy priamo špecifickým príjemcom, ale namiesto toho kategorizujú publikované správy do tried, na ktoré sa príjemcovia prihlasujú.



Obr. 4.2: Schéma komunikácie zariadenia so serverom

správe, je nevyhnutný k neskoršiemu odosielaniu nameraných dát, keď je spolu s teplotou a kľúčom zašifrovaný pomocou transformačnej funkcie *SHA256*. Detailnejší návrh komunikácie je zobrazený na obrázku 4.2. Všetky správy zo senzora na server využívajú *HTTP* metódu *POST*.

4.3 Návrh spracovania a ukladania dát

V návrhu systému pre zber dát je potrebné určiť, s akým typom dát bude systém pracovať a do akej databázy budú tieto dáta zaznamenávané. Pre zaistenie výberu optimálnej databázy sú v rámci práce vykonané výkonnostné testy, ktoré simulujú vkládanie dát a následné dotazovanie nad viacerými databázami – dvoma SQL databázami (MySQL a MariaDB) a jednou NoSQL (MongoDB). Pre účely experimentovania je vytvorená pomocou jazyka PHP zjednodušená pseudo-databáza, založená na súborovom systéme, za pomoci ktorej sa zisťuje efektívnosť špecificky zameranej databázy oproti databázovým systémom.

Keďže primárnym zameraním systému je monitorovanie teploty vzduchu menších objektov – domácností, pri výbere databázy sa počíta iba s jednou meranou veličinou, ktorej rozsah sa pohybuje v medziach $-100°C$ až $100°C$. V tomto prípade nie je potrebné monitorovať teplotu s presnosťou väčšou ako na jedno desatinné miesto. V počiatočných fázach návrhu sa vychádzalo z predpokladu, že informácia o teplote bude ukladaná s dĺžkou 8 (prípadne 16) bitov, za použitia mapovacej funkcie, hlavne z dôvodu šetrenia priestoru. Neskôr sa ukázalo, že pri využití tohto spôsobu bola celková veľkosť uložených dát nižšia len o 3% oproti ukladaniu dát ako *DECIMAL*[2]. Spolu s teplotou je rovnako potrebné ukladať identifikátor zariadenia, ktoré dáta odosiela, ako aj časovú jednotku určujúcu čas zaznamenania.

[2]pri použití databázy MariaDB s formátom úložiska MyISAM

4.3.1 Experimentovanie s databázami

Experimentovanie s databázami je dôležitou súčasťou návrhu, keďže na základe výsledku je zvolený databázový systém pre ukladanie dát. Pre tento účel je vytvorený testovací skript, ktorý najprv inicializuje určitý počet senzorov a následne nad nimi v pravidelne definovaných intervaloch generuje simulačné dáta. Nad týmito dátami sú nakoniec vykonávané rôzne dotazy, pričom predmetom sledovaných hodnôt výstupu sú veľkosť využitého priestoru na disku, rýchlosť vkladania dát a rýchlosť odpovede na predpripravené dotazy, ktoré simulujú najčastejšie operácie:

- Dotaz č.1 testuje najčastejší typ dotazu – zobrazenie všetkých údajov za posledné časové obdobie (v tomto prípade posledný mesiac) pre každý senzor.
- Dotaz č.2 testuje agregačné funkcie – mesačný priemer teplôt pre každý senzor.
- Dotaz č.3 testuje nájdenie extrémov – maximálnu nameranú hodnotu senzora.
- Dotaz č.4 čiastočne opakuje 3. dotaz nad hodnotami zo všetkých senzorov. Test zistuje efektívnosť systému v súvislosti s opakujúcimi sa dotazmi.

Všetky testy prebiehajú vo virtualizovanom stroji s operačným systémom *Ubuntu* (vo verzii 14.04 a 16.04), ktorý má pridelené rovnaké zdroje pri každom teste. Testy databáz sú vykonávané s tromi rôznymi nastaveniami senzorov. Prvý test spočíva v simulácii jedného senzora, ktorý po dobu 36 dní odosiela dáta v 10 minútových intervaloch. Namerané hodnoty sú zobrazené v tabuľke 4.1, pričom posledný dotaz je zámerne vynechaný, pretože je totožný s dotazom č.3. Druhý test, ktorého výsledok je zobrazený v tabuľke 4.2, simuluje 10 senzorov odosielajúcich dáta v 10 minútových intervaloch po dobu jedného roka. Tento test produkuje približne pól milióna zápisov do databázy. Posledný test sa odvíja od výsledkov predošlého testu – cieľom je navýšiť počet senzorov pri zachovaní počtu zápisov z predošlého testu. Konkrétne sa jedná o 608 senzorov zaznamenavajúcich dáta v hodinových intervaloch po dobu 36 dní. Hodnoty získané pri poslednom teste sú zobrazené v tabuľke 4.3.

Databáza	MariaDB 10.1.25			MySQL 5.7.20		MongoDB 2.6.12	MongoDB 3.6.3	Súborový systém
Operačný systém	Ubuntu 16.04			Ubuntu 14.04		Ubuntu 14.04	Ubuntu 16.04	Ubuntu 14.04
Formát úložiska	MyISAM	InnoDB	Aria	MyISAM	InnoDB	–	–	–
Veľkosť (KB)	**105**	272	320	**105**	272	567	256	555
Vkladanie hodnôt (s)	**1 170**	1 890	2 020	1 270	2 640	1 340	1 230	5 530
Dotazy (1-4) (ms)	**9**	18	8	13	29	44	–	51 770
Dotaz č.1 (ms)	3	5	**2**	4	7	7	–	25 640
Dotaz č.2 (ms)	**5**	11	**5**	7	13	34	–	25 950
Dotaz č.3 (ms)	**1**	2	**1**	2	9	3	–	180

Tabuľka 4.1: Test č.1 simuluje 1 senzor, ktorý po dobu 36 dní odosiela dáta v 10 minútových intervaloch – spolu 5186 zápisov do databázy.

Databáza	MariaDB 10.1.25			MySQL 5.7.20		MongoDB 2.6.12	MongoDB 3.6.3	Súborový systém
Operačný systém	Ubuntu 16.04			Ubuntu 14.04		Ubuntu 14.04	Ubuntu 16.04	Ubuntu 14.04
Formát úložiska	MyISAM	InnoDB	Aria	MyISAM	InnoDB	–	–	–
Veľkosť (MB)	**10, 4**	20, 5	29, 3	**10, 4**	21, 5	56, 1	31, 1	58, 0
Vkladanie hodnôt (s)	100	200	180	120	215	**80**	110	385
Dotazy (1-4) (ms)	4 340	**1 560**	3 710	3 660	3 200	21 760	–	–
Dotaz č.1 (ms)	310	**60**	220	230	100	1 610	–	–
Dotaz č.2 (ms)	3 520	**580**	2 530	2 460	1 060	18 360	–	–
Dotaz č.3 (ms)	**470**	770	890	900	1 870	1 550	–	–
Dotaz č.4 (ms)	**40**	150	70	70	170	240	–	–

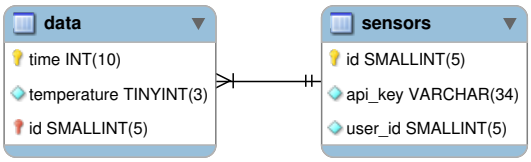
Tabuľka 4.2: Test č.2 simuluje 10 senzorov, ktoré po dobu 1 roka odosielajú dáta v 10 minútových intervaloch – spolu 525620 zápisov do databázy.

Databáza	MariaDB 10.1.25			MySQL 5.7.20		MongoDB 2.6.12
Operačný systém	Ubuntu 16.04			Ubuntu 14.04		Ubuntu 14.04
Formát úložiska	MyISAM	InnoDB	Aria	MyISAM	InnoDB	–
Veľkosť (MB)	**10, 5**	24, 6	29, 3	**10, 5**	23, 6	56, 2
Vkladanie hodnôt (s)	**100**	195	185	120	225	150
Dotazy (1-4) (ms)	71 700	**4 690**	112 240	128 810	6 990	–
Dotaz č.1 (ms)	14 190	**820**	22 930	25 850	1 400	86 400
Dotaz č.2 (ms)	29 990	**2 520**	45 870	52 870	3 430	–
Dotaz č.3 (ms)	27 480	**1 190**	43 360	50 030	2 000	83 330
Dotaz č.4 (ms)	**40**	160	80	60	160	240

Tabuľka 4.3: Test č.3 simuluje 608 senzorov, ktoré po dobu 36 dní odosielajú dáta v hodinových intervaloch – spolu 526528 zápisov do databázy.

Relačné databázy

Pre testovanie relačných databáz je potrebné zostrojiť schému relačnej databázy, ktorá je zobrazená na obrázku 4.3. Pri väčšine výsledkov testovania je potvrdený výkonnostný nárast databázy MariaDB oproti MySQL – z tohto dôvodu sa s využitím databázy MySQL pre vytváraný systém neuvažuje. Výkonnostné testy sú spúštané nad troma rôznymi MariaDB formátmi úložiska – ako je možné vyčítať z tabuliek. Formát InnoDB dokáže najrýchlejšie odpovedať na dotazy, avšak zaberá viac miesta na disku a takiež vkladanie dát je omnoho pomalšie. Naopak, práve v spomínaných úkonoch vyniká formát MyISAM, ktorý dosahuje počas väčšiny testov najlepšie výsledky.



Obr. 4.3: Schéma testovacej databázy.

NoSQL

V testoch je pre porovnanie zastúpená aj jedna NoSQL databáza, avšak, z dôvodu malej podpory na serveroch webových poskytovateľov, nemôže byť tento typ databáz použitý. Pri testovaní databázy MongoDB sú vyskúšané dve rôzne schémy. Prvá schéma pozostáva z dvoch kolekcií, pričom v jednej sú zaznamenávané informácie o senzoroch a v druhej sú ukladané namerané teploty. Druhá schéma je navrhnutá tak, aby využívala iba jednu kolekciu. V tejto kolekcii reprezentuje každý dokument jeden senzor, ktorý v sebe obsahuje pole nameraných teplôt. Výsledkom testov druhej schémy je skutočnosť, že jej výkonnosť rapídne klesá s rastúcim počtom hodnôt v poli.

V prípade MongoDB sú otestované dve verzie tejto databázy. Výsledkom ich testovania je skutočnosť, že aktuálna verzia (3.6.3) dokáže ušetriť oproti verzii 2.6.12 až 45% úložného priestoru. Aktuálna verzia nepodporuje spôsob pristupovania k databáze, aký je využívaný v testovacom skripte pre staršiu verziu, a preto nie sú v tabuľke 4.1 doplnené hodnoty trvania dotazov. V tabuľke 4.2 nie sú zobrazené časy trvania dotazov, z dôvodu veľkej časovej zložitosti.

Z hodnôt v uvedených tabuľkách je možné vyčítať, že najpriaznivejšie časy vkladania údajov v testovaných prípadoch umožnuje práve databáza MongoDB, avšak pri všetkých ostatných sledovaných hodnotách (trvanie dotazov a použitá veľkosť disku) sú jej výsledky nevyhovujúce. NoSQL databázy, na rozdiel od relačných databáz, nemajú automatickú optimalizáciu dotazov – toto je jeden z možných dôvodov, prečo vykonávanie dotazov pri MongoDB trvá dlhší časový úsek.

Súborový systém
Vzhľadom k tomu, že všetky vyššie predstavené databázy sú univerzálne, je vytvorená špecializovaná databáza na úrovni súborového systému. Cieľom databázy je dosiahnuť čo najväčšiu úsporu dát, rýchle zapisovanie aj čítanie. Systém riadenia súborových dát je implementovaný v jazyku PHP s ohľadom na kompatibilitu s cieľovým systémom.

Databáza je navrhnutá tak, aby boli dáta ukladané do binárnych súborov s pevným počtom bitov na záznam. Spolu s definovanou štruktúrou súboru je možné prečítať konkrétny záznam pomocou funkcie `fseek`. V rámci vývoja sú otestované 2 typy štruktúr dokumentov, pričom v oboch sa počíta s ukladaním hodnôt v presne stanovených časových intervaloch. Prvá štruktúra vytvára súbory po dňoch, pričom do jedného súboru je v pravidelných intervaloch uložený zvolený počet senzorov. V prípade presiahnutia zvoleného množstva senzorov je vytvorený ďalší súbor. Avšak, v prípade nevhodne zvolenej konštanty, je v súboroch málo reálnych dát, resp. sú vytvárané viackrát. Druhá varianta je navrhnutá tak, aby riešila spomenuté problémy. Z toho dôvodu, každému senzoru prislúcha jeden súbor, ktorý môže obsahovať dáta za aktuálny kalendárny rok. Týmto sa redukuje množstvo súborov, avšak na úkor zväčšenia veľkosti každého z nich. Všetky súbory zaberajú rovnaké miesto, keďže sa pri vytváraní nového súboru vyplní dopredu známy počet bitov binárnymi nulami.

Po zistení nepriaznivo dlhých dôb vkladania hodnôt a dotazovania sa (viď. tabuľka 4.1), je vývoj tejto databázy pozastavený.

Na základe opakovaných testov jednotlivých databáz sa dospelo k záveru, že pri vytváraní systému bude optimálnym riešením využiť databázu MariaDB s formátom úložiska MyISAM. Predmetná databáza dokáže uložiť dáta najefektívnejšom spôsobom s využitím najmenšej diskovej kapacity, čo je veľmi dôležité pri obmedzených kapacitách databáz ponúkaných webovými poskytovateľmi.

4.3.2 Spracovanie a agregácia

Keďže sa nepočíta s predspracovaním dát na strane zariadenia odosielajúceho teplotu, logika spracovania dát sa nachádza na serverovej strane. Ten pri prijatí správy rozhodne, či sa prijaté dáta zapíšu do databázy, alebo nie, pričom toto rozhodnutie je vykonané na základe času zápisu poslednej hodnoty. Na základe pozorovaní je stanovený ideálny časový rozostup meraných teplôt na 10 minút, čo predstavuje približne 140 hodnôt za deň. Pri použití MariaDB s enginom MyISAM a 10 senzormi by systém dokázal vyprodukovať 100 MB dát za dobu približne 9 rokov. Pri možnom budúcom rozšírení meraných veličín je potrebné počítať aj so skorším prekročením danej hranice, preto sa už pri návrhu počíta s algoritmom, ktorý bude istým spôsobom redukovať staršie dáta. Keďže nie je dôležité archivovať staršie hodnoty v 10 minútových intervaloch, navrhnutý algoritmus pravidelne spriemeruje zaznamenané dáta zo všetkých senzorov v rámci každej hodiny, čím zredukuje objem dát až 6-násobne. Aj pri obmedzených možnostiach hostiteľských služieb je možné

vytvorený algoritmus spúšťať v pravidelných časových intervaloch pomocou softvérového démona *CRON*.

4.4 Návrh webovej aplikácie

V prvom kroku návrhu je potrebné zhodnotiť dostupné technológie a rozhodnúť, ktorá bude využitá pri tvorbe aplikácie. Väčšina webových hostiteľských služieb limituje vývoj serverovej časti aplikácie na použitie jazyka PHP. Z tohto dôvodu sa javí ako najvhodnejšie riešenie použiť PHP framework založený na architektúre MVC (Model-View-Controller). Pri výbere voľne dostupného frameworku sa uvažovalo nad celosvetovo rozšíreným PHP frameworkom *Laravel*, alebo nad populárnym českým frameworkom *Nette*. Vďaka veľkej komunite, ktorá vytvára množstvo návodov a rozšírení, je zvolený *Laravel*.

Užívateľské rozhranie pozostáva z vygenerovaného HTML kódu založeného na Bootstrap[3] šablóne *SB Admin*[4] s využitím javascriptovej knižnice *jQuery*. V rámci výberu technológií sa zohľadňuje, aby všetok použitý softvér bol licencovaný pod voľne šíriteľnou MIT alebo GPL licenciou, keďže sa jedná o otvorený projekt.

Dôležitou súčasťou užívateľského rozhrania je zobrazenie nameraných hodnôt v prijateľnej a zrozumiteľnej forme. Ideálnym riešením je použitie voľne dostupných knižníc pre vykresľovanie grafov. Pri výbere boli otestované 3 knižnice – *Chartist.JS*, *Chart.js* a *Highcharts*. Pri testovaní sa zistilo, že najmenšia kňižnica – *Chartist.JS* neobsahuje prvok popisu hodnoty pre jednotlivé body grafu. Knižnica *Chart.js*, však na druhej strane nepodporuje zápis súradníc grafu do jedného pola. Spomínané nevýhody rieši až knižnica *Highcharts*. Pre komerčné využitie požadujú vývojári za licenciu nemalé finančné čiastky, avšak pre nekomerčné používanie je knižnica využiteľná zadarmo v rámci *Creative Commons* (CC) licencie.

[3] jednoduchá a voľne stiahnuteľná sada nástrojov pre tvorbu webu a webových aplikácií
[4] https://github.com/BlackrockDigital/startbootstrap-sb-admin

Kapitola 5

Implementácia

Celý systém pre monitorovanie teploty objektov je možné rozdeliť do dvoch častí, pričom každá využíva iné programovacie techniky a jazyky. Senzorová časť je implementovaná pomocou modifikovaného jazyka C++ za použitia bezplatného vývojového prostredia Arduino IDE. Tento jazyk je zvolený z dôvodu podpory so spomínaným vývojovým prostredím, ktoré je obľúbené v komunite domácich počítačových nadšencov. Podrobnejšie informácie je možné nájsť v sekcii 5.1.

Druhá časť pozostáva z webového systému komunikujúceho s databázou. Táto časť využíva niekoľko webových technológií a jazykov ako PHP s využitím frameworku *Laravel*, HTML, CSS s knižnicou *Bootstrap*, JavaScript s knižnicou *jQuery* a využitím technológie AJAX. Implementácii sa podrobnejšie venuje sekcia 5.4.

V sekciách 4.2 a 4.3 je zdokumentovaný návrh vzájomnej komunkácie spomínaných dvoch častí a spôsob následného spracovania a uskladnenia dát – v rámci tejto kapitoly (sekcia 5.2 a 5.3) je na ich základe popísaná realizácia a implementácia.

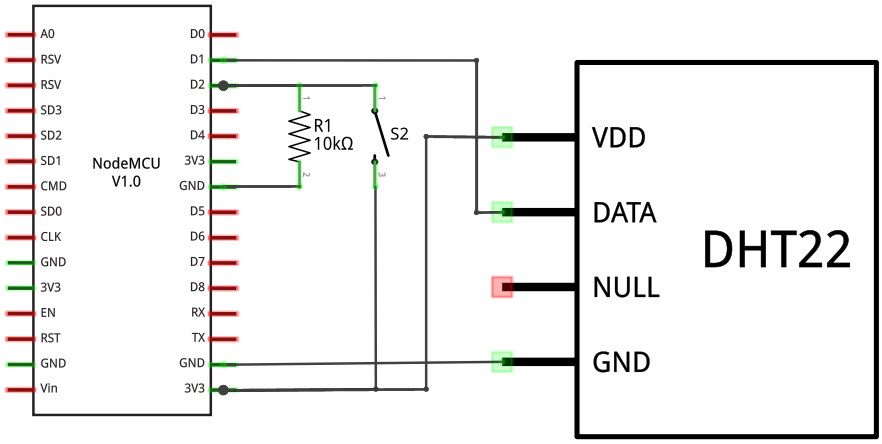
5.1 Implementácia senzoru

Senzorová časť je implementovaná v mierne modifikovanom jazyku C++. Okrem vstavaných knižníc, ponúkaných prostredím Arduino IDE pre ESP8266, program využíva aj tri voľne dostupné knižnice. Knižnice `DHT-sensor-library` a `Adafruit_Sensor` vytvorené pod spoločnosťou Adafruit zjednodušujú komunikáciu s teplotnými senzormi `DHT22` a `DHT11`. Pre vytváranie *hash* hodnoty bola použitá kryptografická knižnica podporujúca algoritmus `SHA256`.

Okrem spomínaného teplotného senzora je k senzoru pripojené aj tlačidlo spolu s pull-down rezistorom (viď. obrázok 5.1). Po stlačení tohto tlačidla je vyvolané prerušenie, v dôsledku čoho sa zariadenia ESP prepne do konfiguračného stavu[1]. V tomto stave sa zo zariadenia stáva prístupový bod, na ktorom beží webový server. Po zaslaní potrebných údajov pomocou odkazu v QR kóde, je zariadenie prepnuté do stavu zberu a odosielania dát, kde je v prvej *HTTP* správe odoslaný jedinečný identifikátor. Po jej úspešnom prijatí serverom sú ďalej zasielané iba správy s nameranou teplotou, ktorých telo obsahuje následujúce údaje.

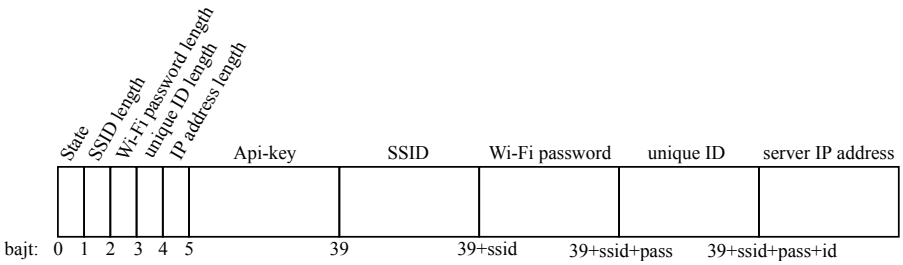
```
teplota + kľúč + SHA256(teplota + kľúč + identifikátor)
```

[1] pri pepínaní stavov je pozmenený nultý bajt pamäte EEPROM a následne vykonaný reštart zariadenia



Obr. 5.1: Ukážka zapojenia teplotného senzora a tlačidla k NodeMCU.

Pre účel prezistentného uchovania prihlasovacích údajov do Wi-Fi siete, identifikačného kľúča[2] a bezpečnostného identifikátora, je použitá vstavaná pamäť EEPROM, ktorej schéma využitia je znázornená na obrázku 5.2. Prvý bajt určuje stav zariadenia, ktorý je reprezentovaný číslom. Číslo `40` značí stav, kedy sa zariadenie pokúša odoslať prvú prihlasovaciu správu. Po jej následnom úspešnom odoslaní prechádza do stavu reprezentovaného číslom `42`. Všetky ostatné čísla značia stav, kedy nie je zariadenie nakonfigurované. Počas konfigurácie dochádza k vyplneniu všetkých potrebných údajov a ich následnému uloženiu do pamäte EEPROM podľa znázorneného obrázku 5.2.



Obr. 5.2: Schéma využitia pamäte EEPROM.

5.2 Implementácia komunikácie

Implementácia komunikačného protokolu je v súlade s návrhom popísaným v sekcii 4.2, a teda ide o *HTTP* komunikáciu s využitím metódy *POST*. Senzory zasielajú na server dva typy správ – registračnú správu a správu s nameranou teplotou. Každý typ je odosielaný na inú adresu servera. Registračné správy sú zasielané na adresu `/sensor/identificate` a namerané hodnoty tak na adresu `/temperature/post`.

[2] tzv. *Api-key*, slúži na identifikáciu zariadenia v systéme

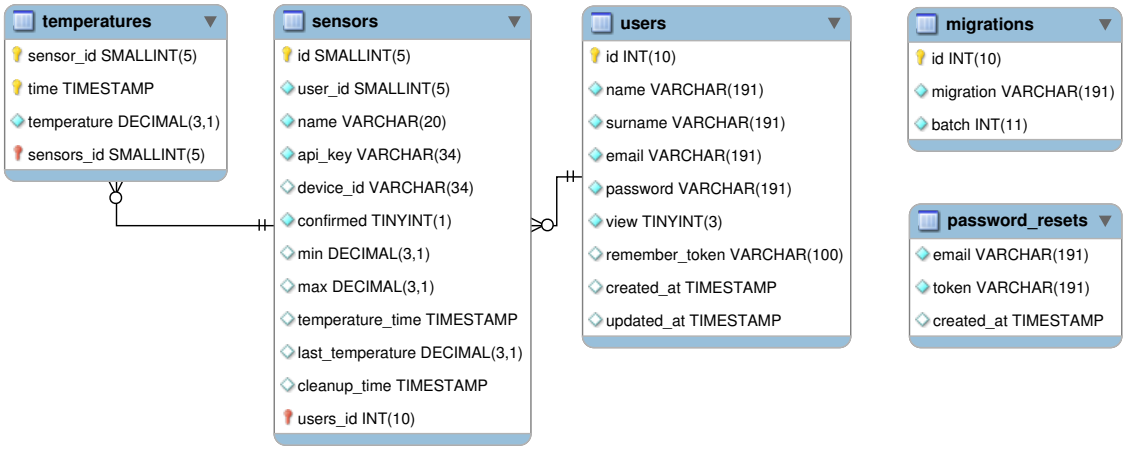
5.3 Implementácia ukladania a spracovania dát zo senzoru

Ako je navrhnuté v kapitole 4.3, dáta sú ukladané do databázy MariaDB s formátom úložiska MyISAM, ktorej výsledná schéma je znázornená na obrázku 5.3.

V rámci návrhu sa vychádzalo z predpokladu, že dáta budú do databázy ukladané s využitím skriptu implementovaného v jazyku PHP bez použitia frameworku. Nakoniec bola táto funkcionalita zahrnutá a implementovaná v použitom frameworku, z dôvodu dosiahnutia čo najväčšej konzistencie systému.

Pre účel poskytovania dát s čo najaktuálnejšou teplotou užívateľovi systému, je interval odosielania dát zo senzoru nastavený na 1 minútu. Napriek tomu si server neukladá všetky prichádzajúce hodnoty do tabuľky určenej pre teplotné hodnoty, ale iba do políčka aktuálnej teploty, nachádzajúceho sa v tabuľke pre senzory. Ak je rozdiel času zaznamenanej teploty v tabuľke teplôt a prichádzajúcej správy väčší ako 10 minút, údaj sa zapíše, inak sa iba aktualizuje údaj s aktuálnou teplotou. Týmto spôsobom sú garantované minimálne 10 minútové rozdiely hodnôt v databáze.

Agregácia starších hodnôt je spúšťaná pomocou načítania adresy `/cleanup`, ktorá sa v rámci prevádzky systému spúšťa automaticky v pravidelných jednodňových intervaloch pomocou softvérového démona *CRON*. Algoritmus je zostavený tak, aby agregoval hodnoty všetkých senzorov v databáze. Agregácia spočíva v spriemerovaní hodnôt v rámci každej hodiny a následnom uložení priemerných hodnôt. V prípade, že je nad daným senzorom vykonávaný agregačný algoritmus v časovom intervale kratšom ako 24 hodín, algoritmus ignoruje predmetný senzor, pričom nevykoná žiadnu akciu.



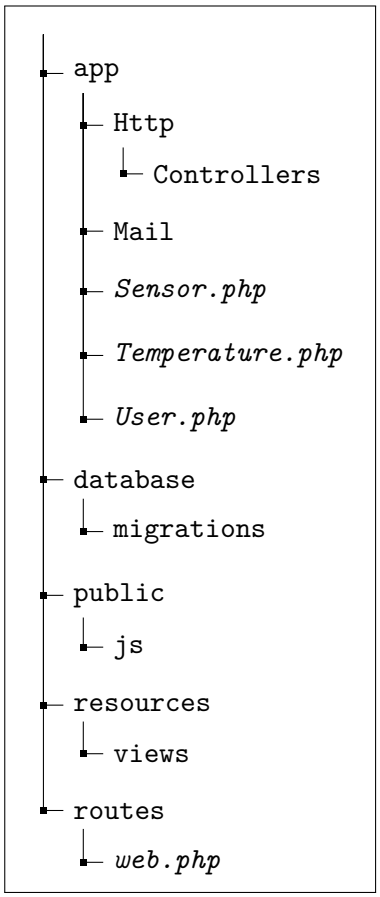
Obr. 5.3: Výsledná schéma databázy.

5.4 Webová aplikácia

Najrozsiahlejšou časťou celej implementácie je webová aplikácia, ktorá je vytvorená pre interakciu užívateľa so systémom. Keďže je táto časť vytváraná pomocou frameworku *Laravel*, ktorý je postavený na návrhovom vzore MVC (model, pohľad, kontrolér), bolo nevyhnutné v rámci implementácie dodržiavať princípy tohto vzoru. Aplikácia je aj z toho dôvodu, rozdelená do troch hlavných častí:

- **Model** – reprezentuje dáta z databázy.
- **Pohľad** – zobrazuje dáta používateľovi.
- **Kontrolér** – na základe požiadaviek sprostredkuje dáta z modelu, ktoré sú následne odovzdané pohľadu.

Vďaka použitiu spomínaného frameworku je možné do vytváraného webového systému jednoducho zaviesť autentifikáciu užívateľov spoločne s ich registrovaním a obnovou hesla. Výhodou použitého frameworku je zapúzdrenie SQL dotazov pomocou *Elequent* modelu, ktorý automaticky vytvára dotazy v SQL jazyku. Framework taktiež dokáže generovať SQL tabuľky, ktorých schémy sú definované pomocou jazyka PHP.
Na obrázku 5.4 sú v zjednodušenej adresárovej štruktúre aplikácie znázornené tie adresáre, v ktorých boli upravované zdrojové súbory.



Obr. 5.4: Zjednodušená adresárová štruktúra webovej aplikácie.

`routes/web.php`

Súbor obsahuje všetky definície *HTTP* žiadostí, ktoré dokáže systém obslúžiť spolu s obslužnou metódou v špecifikovanom kontroléri. Rovnako sú v ňom nadefinované tie žiadosti, ktoré potrebujú pre správnu odpoveď autentifikáciu užívateľa. V prípade vytváraného systému existujú iba tri typy žiadostí – zaslanie registračnej správy zariadenia, správy s informáciou o nameranej teplote a spúšťač pre agregáciu hodnôt v tabuľke.

`app/Http/Controllers/`

V tomto adresári sa nachádzajú kontroléry, ktoré sú na základe požiadaviek používateľa volané zo súboru `app/routes/web.php`. Kontrolér s názvom `SensorController` obsluhuje všetky akcie vykonávané nad senzormi, ako vytvorenie nového senzora, registrácia pomocou QR kódu, zobrazenie dostupných senzorov, editácia, odosielanie hodnôt senzora a mnoho ďalších. Prijímanie teplôt zo senzora zabezpečuje kontrolér `TemperatureController`, ktorý zároveň inicializuje posielanie e-mailov, v prípade prekročenia stanovenej teploty. Kontrolér `UserController` vykonáva zmeny užívateľského účtu. Kontrolér s názvom `HomeController` zabezpečuje zobrazenie domovskej stránky, ktorá je tvorená prehľadom údajov zo všetkých senzorov. Kontrolér `CleanupController` je zameraný na agregáciu starších dát. V adresári `app/Http/Controller/Auth` sa nachádzajú kontroléry pre registráciu a prihlásenie užívateľa, prípadne obnovenie hesla po jeho zabudnutí. Všetky kontroléry pracujú s dátami, ktoré sú reprezentované následujúcimi modelmi.

`app/Sensor.php`

Model senzoru implementujúci časté metódy vykonávané nad senzorom a jeho dátami. Na prepojenie dvoch tabuliek slúži metóda `hasMany()` *Elequent* modelu, ktorá je využitá v metóde znázornenej vo výpise 5.1.

```
public function temperatures() {
  return $this->hasMany(Temperature::class)->select(['time','temperature']);
}
```

Výpis 5.1: Metóda na prepojenie dvoch tabuliek.

`app/Temperature.php app/User.php`

Model teploty a užívateľa nie sú z pohľadu implementácie zaujímavé, avšak je potrebné, aby ich systém obsahoval, pretože sú nevyhnutné k získaniu dát z rovnako pomenovaných tabuliek.

`resources/views/`

V tomto adresári sa nachádzajú pohľady aplikácie, ktoré sú zobrazované na základe žiadostí užívateľa. Tieto pohľady sú uložené v tzv. šablónach vo formáte `blade.php`, v ktorých je možné používať preddefinované konštrukcie pre uľahčenie ich vytvárania. Následne sú šablóny interpretované nástrojom *Blade*, ktorý z nich vytvorí klasické PHP súbory. Keďže užívateľské rozhranie vychádza zo šablóny *SB Admin*, statické zdrojové súbory implementované v jazyku HTML sú prepísané do dynamických stránok vo formáte `blade.php`. Opakujúce sa časti stránky boli vyčlenené do samostatných súborov, ktoré sú následne importované jednotlivými šablónami. Adresár ďalej obsahuje:

- `sb-admin/` adresár, v ktorom sa nachádzajú opakujúce sa časti ako navigačná lišta, päta stránky, modálne okno na odhlásenie sa a skripty používané naprieč celou stránkou.
- `layouts/` adresár osahujúci súbor `master.blade.php`, v ktorom sa nachádza HTML hlavička stránok používaná naprieč celým webovým systémom. V adresári sa nachádza aj schéma stránky pri zabudnutí a obnovení hesla.
- `sensors/` adresár, ktorý obsahuje všetky prezentéry pracujúce so senzormi. Nachádzajú sa v ňom šablóny pre vytvorenie, editáciu, výpis všetkých senzorov, detaily o senzore a generovanie QR kódu na nastavenie senzora.
- `auth/` adresár implementujúci šablóny pre prihlasovanie, registráciu a obnovenie hesla.
- `user/` adresár obsahujúci šablónu pre menenie informácií o užívateľovi.
- `email/` adresár, ktorý obsahuje šablónu e-mailovej správy, ktorá je automaticky generovaná pri prekročení nastavených hodnôt. Šablóna e-mailu je definovaná pomocou Markdown syntaxe, z ktorej následne framework vytvorí HTML správu.
- `home.blade.php` šablóna pre vytvorenie "domovskej" stránky, na ktorej sa nachádza prehľad všetkých senzorov.

`database/migrations/`

V tomto adresári sa nachádzajú súbory, v ktorých sú definované schémy tabuliek. Na základe týchto súborov je tak možné vytvoriť tabuľky v databáze.

`public/js/highcharts.js`

Súbor obsahujúci JavaScriptové funkcie pre spracovanie prijatých dát vo formáte JSON a ich následné dynamické vykreslenie v grafovej knižnici HighCharts.

5.5 Použité technológie

Okrem spomenutého frameworku sú v systéme využité aj ďalšie technológie, knižnice, prípadne rozšírenia programovacieho rozhrania, pričom pri ich výbere sa zohľadňovala otvorenosť licencie.

Carbon PHP API rozšírenie

Pre jednoduchšiu prácu s časovými jednotkami je do systému zavedené rozšírenie programovacieho rozhrania pre objekt `DateTime`. Rozšírenie umožňuje jednoduchý prevod reťazca, vyjadrujúceho čas, na jednotky času a následné vykonávanie matematických operácií, porovnávanie a mnoho dalších funkcií. Aj z tohto dôvodu je toto rozšírenie využívané hlavne v agregačnom algoritme.

jQuery knižnica

Pre dynamické zmeny stránok webového rozhrania je potrebné využiť jazyk JavaScript. Pre jednoduchšiu manipuláciu s HTML objektami a spravovanie udalostí je možné využiť JavaScriptovú knižnicu jQuery. V implementovanom systéme je okrem iného využitá na expandovanie bočnej navigačnej lišty, prepínanie medzi zvolenými intervalmi zobrazovania hodnôt a mnoho ďalších.

JavaScript Cookie

Z dôvodu jednoduchšieho spracovania dát cookies je využité aplikačné rozhranie, ktoré umožňuje jednoducho vytvárať, zapisovať a čítať súbory cookies. Tato funkcionalita je využívaná pri zapamätaní si stavu (expandovaná/zúžená) bočnej navigačnej lišty a poslednej zvolenej voľby intervalu zobrazenia dát na hlavnej stránke.

QRCode.js

Počas registrácie zariadenia je nutné zadať zariadeniu ESP dáta zo servera. V rámci návrhu sa rozhoduje medzi dvoma alternatívami. V prvej by boli informácie zo servera prepisované ručne na stránky vytvorené ESP zariadením. V druhej by bol zoskenovaný QR kód obsahujúci všetký potrebné údaje v odkaze, ktorý sa jeho následným potvrdením otvorí. Predmetom implementácie je druhá alternatíva, nakoľko predstavuje racionálnejšie riešenie. Keďže je potrebné do zariadenia okrem adresy servera a vygenerovaného kľúča zabezpečiť dodanie aj prihlasovacích údajov na Wi-Fi sieť, je zvolené generovanie QR kódu na klientskej strane bez potreby zasielania týchto údajov na server. Za účelom generovania QR kódu je použitá JavaScriptová knižnica namiesto PHP.

AJAX

Technológia asynchrónneho JavaScriptu je využívaná v rámci dynamickej variability intervalov vykresľovaných grafov na hlavnej stránke systému. Rovnako sa AJAX technológia využíva pri získavaní dát grafov. Nakoľko sa jedná o väčšie množstvo dát, v prípade, ak by sa dáta zasielali spolu so stránkou, načítanie stránky by mohlo trvať aj niekoľko sekúnd. Vďaka využitiu tejto technológie sa stránka načíta okamžite a dáta pre jednotlivé grafy sú prijímané postupne.

HighCharts

Pre vykresľovanie grafov je na základe návrhu použitá knižnica HighCharts. Grafy vykresľované touto knižnicou sú konfigurovateľné pomocou JSON štruktúry, ktorá zároveň obsahuje aj vykresľované dáta. Vo výpise 5.2 je znázornená JSON štruktúra prijatá AJAX odpoveďou zo servera, ktorá je ďalej spracovaná a využitá pre zobrazenie grafu a súhrnných informácií v tabuľke.

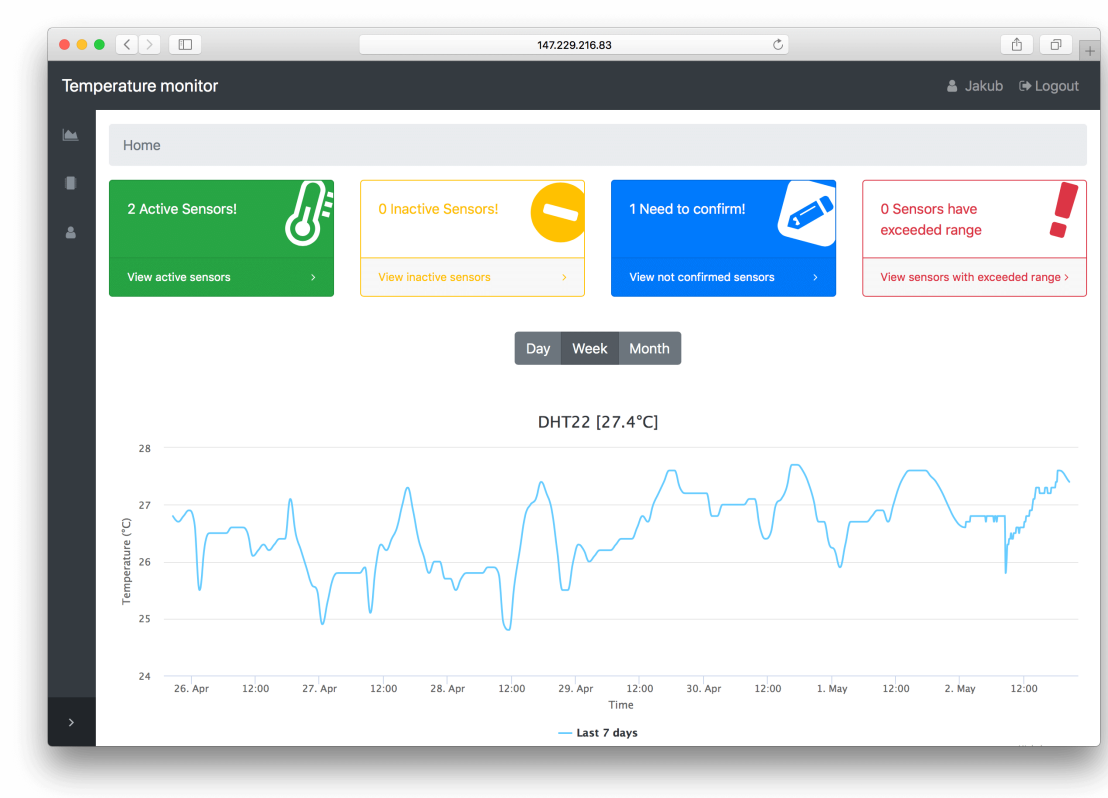
```
{
  "Last 24 hours": [
    { "time": "2018-04-26 13:12:21", "temperature": "26.3" },
    { "time": "2018-04-27 13:08:27", "temperature": "26.3" }
  ],
  "details": {
    "last":
    { "time": "2018-04-27 13:08:27", "temperature": "26.3" },
    "min":
    { "time": "2018-04-27 00:59:23", "temperature": "24.1" },
    "max":
    { "time": "2018-04-26 18:15:33", "temperature": "27.1" },
    "avg": "25.93986"
  }
}
```

Výpis 5.2: Štruktúra AJAX odpovede.

Kapitola 6

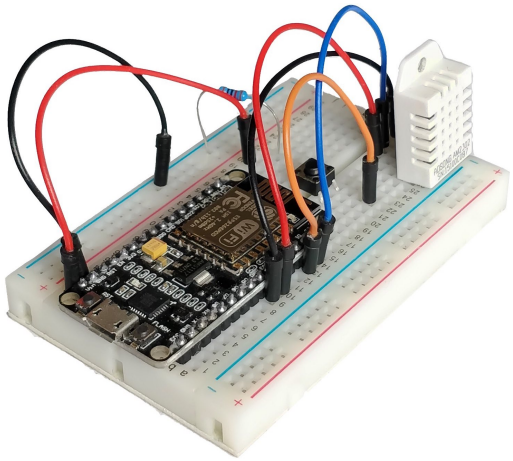
Výsledný systém a testovanie

Výsledný systém sa skladá z dvoch častí – senzorová a webová. Keďže senzorová časť nemá žiadne užívateľské rozhranie, celý systém je ovládaný z webového systému. Ten umožňuje okrem zobrazovania senzorových dát, tieto senzory zároveň aj spravovať. Každý používateľ si tak môže registrovať vlastné senzory podľa postupu popísaného v prílohe A.3. Po registrovaní senzora je možné sledovať jeho dáta v interaktívnych grafoch. Na obrázku 6.1 je znázornená časť úvodnej stránky, ktorá okrem informácií o stavoch vlastnených senzorov, zobrazuje aj ich grafy, spoločne s doplňujúcimi údajmi v tabuľkách.



Obr. 6.1: Časť úvodnej stránky webového systému.

Počas implementácie je systém neustále testovaný na jednom z kompatibilných systémov. Konkrétne sa jedná a lokálny systém bežiaci na platforme Raspberry Pi. Server nepretržite prijíma dáta z dvoch senzorov postavených na platforme NodeMCU 1.0, založených na čipe ESP8266. K zariadeniu boli pripojené senzory `DHT22` a `DHT11`. Na obrázku 6.2 je zobrazený prototyp senzora využívaný pri testovaní systému.

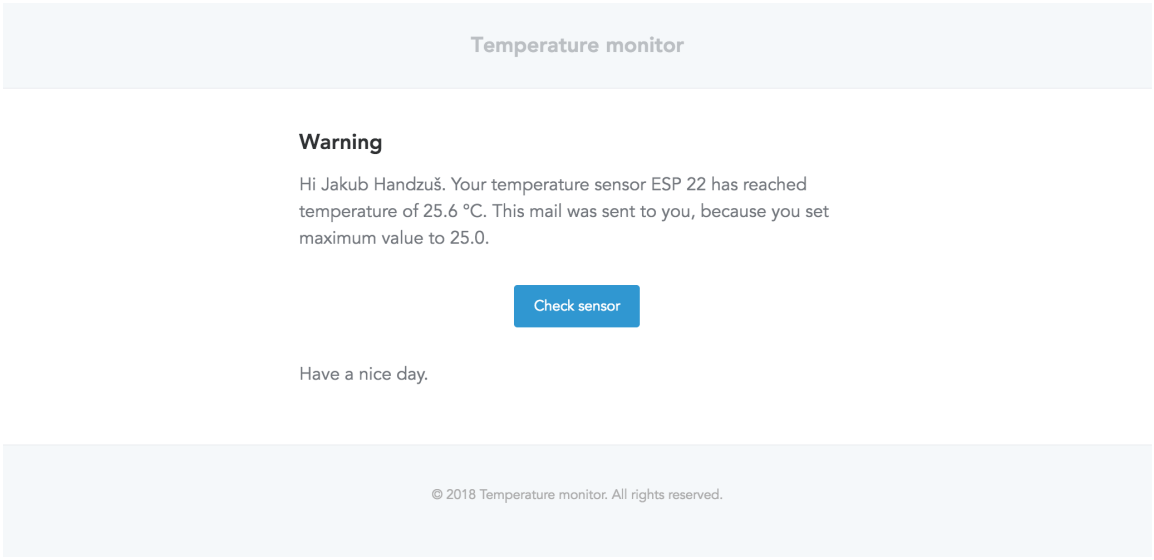


Obr. 6.2: Testovací senzor.

V rámci testovania systému a vizualizácie dát je generované veľké množstvo testovacích hodnôt, aby nebolo potrebné čakať na naplnenie systému reálnymi dátami. Za týmto účelom je vytvorený skript pre vkladanie hodnôt, ktorý okrem teploty a identifikácie senzora obsahuje v zaslaných správach aj časový údaj. Rovnako je do systému pridaný skript, ktorý umožňuje prijímať spolu s teplotou aj údaje o zaznamenanom čase a nevyžaduje overovanie pomocou transformačnej funkcie. Avšak pri reálnom používaní je táto funkcionalita znemožnená. Aj vďaka tomuto spôsobu generovania dát je jednoduché otestovať agregačný algoritmus. Na obrázku 6.3 je možné vidieť graf s históriou teplôt, ktoré sú generované náhodne, pred prvým spustením agregačného algoritmu a po ňom. Z obrázku je evidentné, že vďaka spriemerovaným hodnotám je graf zbavený krátkych zákmitov.

Obr. 6.3: Ukážka grafu pred a po agregácii.

Počas testovania je pre zasielanie e-mailových správ využitá služba *Mailgun*, ktorá ponúka po vytvorení bezplatného účtu možnosť zaslania až 1000 správ mesačne na vopred schválené adresy. V rámci prevádzky systému na serveri webového poskytovateľa služieb sa počíta s využitím pridelenej e-mailovej schránky. Na obrázku 6.4 je zobrazená ukážka vygenerovaného e-mailu, ktorý je zaslaný v prípade prekročenia prednastavenej prahovej hodnoty.



Obr. 6.4: Ukážka vygenerovanej e-mailovej správy pri prekročení stanovenej teploty.

Kapitola 7

Záver

Cieľom tejto práce bolo navrhnúť a implementovať systém, ktorý spracováva dáta z teplotných senzorov pripojených na zvolenú platformu. Systém mal byť navrhnutý s dôrazom na efektívnosť ukladaných dát s ohľadom na typické operácie vykonávané nad dátami.

Teoretická časť je z toho dôvodu zameraná na spôsoby spracovania dát z IoT platforiem a ich ukladanie do rôznych databázových systémov, vrátane prevádzkovania IoT systému. Jej ďalšiou súčasťou sú zdokumentované spôsoby merania teploty, hardvérové platformy umožňujúce zber dát, ako aj možnosti pripojenia do internetovej siete pomocou rôznych komunikačných technológií.

V rámci návrhu je z dôvodu zabezpečenia efektívnosti systému porovnávaných niekoľko databázových systémov a jedna špecializovaná databáza, pričom pre tento účel boli využité vlastné automatizované testy. Spôsob prevádzkovania je zvolený s ohľadom na čo najnižšie náklady na zriadenie ako aj samotnú prevádzku systému.

Implementovaný systém sa vo finálnom prevedení skladá z dvoch častí. Senzorová časť využíva čip ESP8266, ktorý v pravidelných intervaloch meria teplotu pomocou senzora DHT22 a následne odosiela namerané údaje pomocou HTTP protokolu na server. Z dôvodu kompatibility s poskytovateľmi webových služieb je serverová časť implementovaná pomocou aplikačného frameworku Laravel, postaveného na programovacom jazyku PHP, s využitím databáze MariaDB s MyISAM formátom úložiska.

Výsledný systém tak umožňuje meranie teploty menších objektov s dôrazom na následne spracovanie a ukladanie nameraných hodnôt pri zachovaní čo najnižších nákladov, čím je systém vhodný pre domácich počítačových nadšencov.

Systém umožňuje rozšírenie aj o ďalšie senzory, v dôsledku čoho je možné zaznamenávať aj iné veličiny. Nakoľko je vytváraný systém pomerne široko zameraný existujú rôzne možnosti profilácie. Pre zjednodušené nasadenie do vzdialeného výpočtového centra, VPS servera alebo na domáci server typu Raspberry Pi, je možné zabaliť systém do softvérového kontajnera, ktorý je následne možné spustiť jedným príkazom.

Literatúra

[1] Amazon AWS IoT Developer Guide. [Online; navštívené 12.04.2018].
Dostupné z: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

[2] Google Cloud IoT Core Documentation. [Online; navštívené 13.04.2018].
Dostupné z: https://cloud.google.com/iot/docs/

[3] IBM Cloud Docs Internet of Things Platform. [Online; navštívené 12.04.2018].
Dostupné z:
https://console.bluemix.net/docs/services/IoT/getting-started.html

[4] *MariaDB Changelogs*. [Online; navštívené 15.02.2018].
Dostupné z: https://mariadb.com/kb/en/library/changelogs/

[5] Microsoft Azure IoT Hub Documentation. [Online; navštívené 13.04.2018].
Dostupné z: https://docs.microsoft.com/en-us/azure/iot-hub/

[6] *MyISAM Overview*. [Online; navštívené 17.02.2018].
Dostupné z: https://mariadb.com/kb/en/library/myisam-overview/

[7] VPS Hosting guide. [Online; navštívené 26.05.2018].
Dostupné z: https://www.webhostingsecretrevealed.net/vps-hosting-guide/

[8] Anthony T. Velte, R. E., Toby J. Velte: *Cloud Computing*. Computer Press, a.s., 2011, ISBN 978-80-251-3333-0.

[9] Bartholomew, D.: *MariaDB vs. MySQL*. 2012, [Online; navštívené 15.02.2018].
Dostupné z:
http://www.odbms.org/wp-content/uploads/2014/03/mariadb-vs-mysql.pdf

[10] Char, K.: Internet of Things System Design with Integrated Wireless MCUs. Technická správa, Silicon Labs, 2015.
Dostupné z: https://www.silabs.com/documents/public/white-papers/Internet-of-Things-System-Design-with-Integrated-Wireless-MCUs.pdf

[11] Davies, A.: On LPWANs: Why Sigfox and LoRa are rather different, and the importance of the business model. Marec 2015, [Online; navštívené 30.01.2018].
Dostupné z:
http://rethinkresearch.biz/articles/on-lpwans-why-sigfox-and-lora-are-rather-different-and-the-importance-of-the-business-model/

[12] Dolinský, I. P.: Senzory II. KEMT FEI TU Košice, 2015, [Online; navštívené 10.02.2018].
Dostupné z:
https://data.kemt.fei.tuke.sk/SK_rozhrania/_materialy/Senzory%20II.pdf

[13] Espressif System: *ESP8266EX*. 2017, verzia 5.7, [Online; navštívené 28.01.2018].
Dostupné z: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[14] Halvorsen, H.-P.: Introduction to Database Systems, 2016, [Online; navštívené 22.04.2018].
Dostupné z:
http://home.hit.no/~hansha/documents/database/training/Introduction%20to%20Database%20Systems/Introduction%20to%20Database%20Systems.pdf

[15] Lethaby, N.: Wireless connectivity for the Internet of Things. Technická správa, Texas Instruments Incorporated, Október 2017, [Online; navštívené 29.01.2018].
Dostupné z: http://www.ti.com/lit/wp/swry010a/swry010a.pdf

[16] LoRa Alliance: *LoRa Alliance Technology*. [Online; navštívené 18.02.2018].
Dostupné z: https://www.lora-alliance.org/what-is-lora

[17] Microsoft Azure: *What is IaaS*. [Online; navštívené 24.02.2018].
Dostupné z: https://azure.microsoft.com/en-us/overview/what-is-iaas/

[18] Nayak, A.; Poriya, A.; Poojary, D.: Type of NOSQL databases and its comparison with relational databases. *International Journal of Applied Information Systems*, ročník 5, č. 4, 2013: s. 16–19.

[19] Raycom: *Internet věcí - Bezdrátově a s velkým dosahem*. [Online; navštívené 18.02.2018].
Dostupné z: http://www.raycom.cz/data/article/filemanager/LoRa.pdf

[20] Sigfox: Sigfox Technology Overview. [Online; navštívené 30.01.2018].
Dostupné z: https://www.sigfox.com/en/sigfox-iot-technology-overview

[21] Solà Campillo, O.: Security issues in Internet of Things. 2017, [Online; navštívené 30.01.2018].
Dostupné z:
https://upcommons.upc.edu/bitstream/handle/2117/109290/Security%20Issues%20in%20Internet%20of%20Things.pdf?sequence=1&isAllowed=y

[22] Upton, E.; Halfacree, G.: *Raspberry Pi - uživatelská příručka*. 2017, ISBN 9788025141403.

[23] Zendulka, J.; Rudolfová, I.: Databázové systémy IDS – Studijní opora, 2016.

[24] Špringl, V.: Měření teploty - polovodičové odporové senzory teploty. 2004, [Online; navštívené 9.02.2018].
Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/dokumentace/mereni-teploty-polovodicove-odporove-senzory-teploty.html

Príloha A

Používanie systému

Úlohou tejto prílohy je oboznámiť používateľa s vytváraným systémom. Keďže sa jedná o otvorený systém, zdrojové kódy sa nachádzajú voľne dostupné a ktokoľvek má možnosť si systém stiahnuť a uviesť do prevádzky, prípadne vykonať rôzne zmeny. Pre úspešne zostavenie programu a uvedenie do prevádzky je v sekciách A.1, A.2 popísaný postup a potrebné prerekvizity. V sekcii A.3 je popísaná registrácia senzora a v sekcii A.4 sú zdokumentované možnosti zobrazenia dát.

A.1 Senzorová časť

Pre úspešné zostavenie a nahranie programu na ESP8266 čip je možné využiť viaceré programy. Tento návod popisuje postup pri využití vývojárskeho prostredia Arduino IDE. Po prvotnom spustení je potrebné do programu doinštalovať balíček obsahujúci potrebné dáta pre nahrávanie programov na dosky typu ESP8266. Keďže program využíva voľne dostupné knižnice, je potrebná ich doinštalácia pomocou odkazov uvedených priamo v zdrojovom kóde. Ide o knižnice pre vytváranie transformačnej funkcie `SHA256` a pre získavanie dát z `DHT` senzorov.

Pre správne fungovanie zariadenia je potrebné toto zariadenie vybaviť teplotným snímačom a tlačidlom, pre prípadne zmeny konfigurácie, podľa schémy na obrázku 5.1. V prípade iného typu zapojenia, prípadne použitia teplotného snímača `DHT11`, je nutná zmena konštánt na začiatku zdrojového súboru. Následne je nevyhnutné program pomocou vývojárskeho prostredia skompilovať a nahrať do zariadenia.

A.2 Webová časť

Pre úspešné nasadenie webovej časti systému je potrebné, aby server spĺňal následujúce požiadavky:

- PHP vo verzii 7.0 alebo novšie
- webový server Apache
- databáza MariaDB
- Composer pre správu PHP závislostí

Pomocou súbora `db_init.sql` je potrebné vytvoriť štruktúru databázy a následné naklonovanie *git* repozitára

Nasadenie sa líši v závislosti od využitého servera. V každom prípade je dôležité naklonovať *git* repozitár a pomocou súbora `db_init.sql` vytvoriť štruktúru databázy. V súbore `.env` je potrebné nastaviť prístupové údaje k databáze a mailovému SMTP serveru.

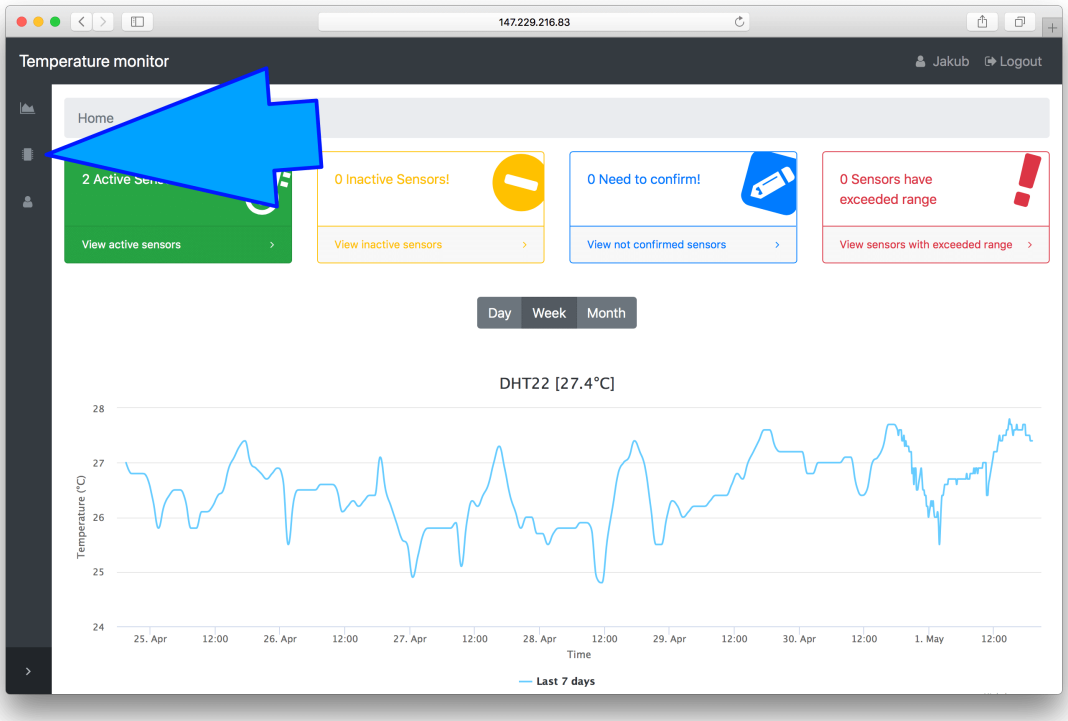
A.3 Registrácia zariadenia

Po úspešnej realizácii vyššie uvedených krokoch je potrebné vykonať následujúce akcie:

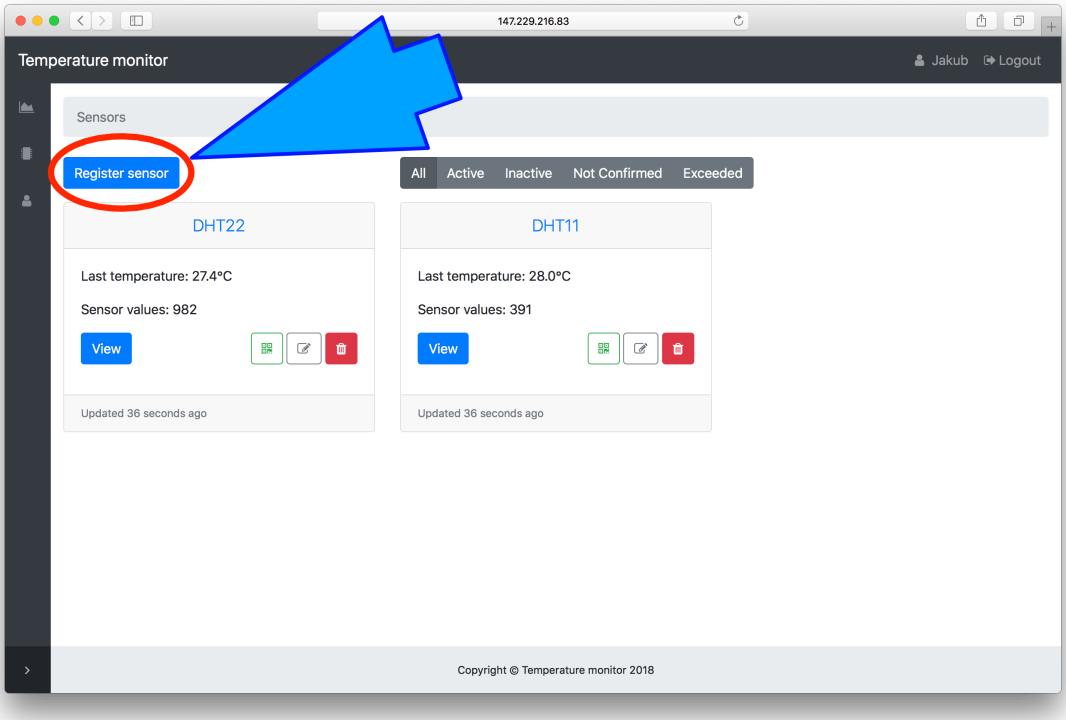
1. Otvoriť v počítači stránku webového systému a následne sa prihlásiť, prípadne sa registrovať.
2. Kliknúť na záložku so senzormi a následne registrovať nový senzor. (viď. obrázok A.1 a A.2)
3. Zadať meno senzora, a v prípade záujmu notifikácie pri prekročení prahovej hodnoty, je možné vyplniť hodnoty.
4. Vyplniť prístupové údaje od Wi-Fi, ku ktorej bude zariadenie ESP pripojené a vygenerovať QR kód. (viď. obrázok A.3)
5. Zapnúť ESP zariadenie do registračného módu stlačením predpripraveného tlačidla.
6. Pripojiť sa mobilným telefónom na vytvorenú Wi-Fi sieť s názvom “ESP”, ktorá je bez zabezpečenia.
7. Oskenovať QR kód mobilným telefónom a otvoriť oskenované URL.
8. Následne je potrebné na webovej stránke potvrdiť novo zaregistrovaný senzor. (viď. obrázok A.4)

A.4 Zobrazenie dát

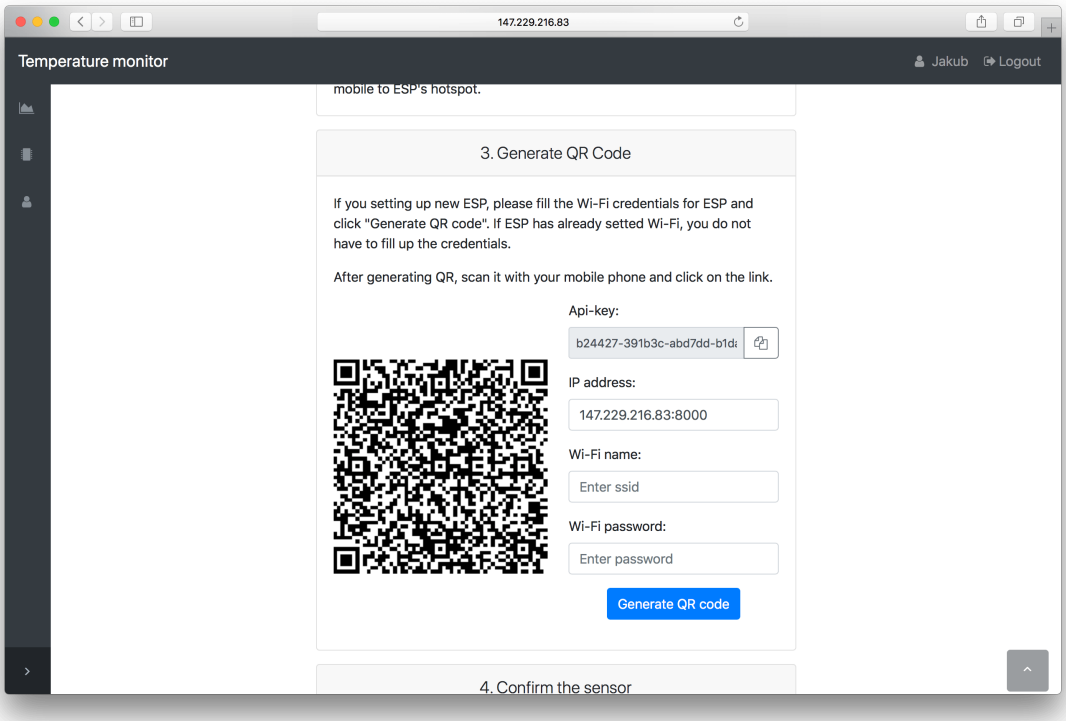
Namerané dáta zo všetkých senzorov je možné zobraziť na úvodnej stránke, ktorá umožňuje používateľovi zvolenie časového intervalu, pre ktorý sú následne vykreslené grafy (viď. obrázok A.5). Rovnako existuje možnosť zobraziť namerané dáta pre konkrétny senzor, kde sú vykreslené dáta za posledný deň, týždeň a mesiac (viď. obrázok A.5, ktorý zobrazuje reálne namerané dáta).



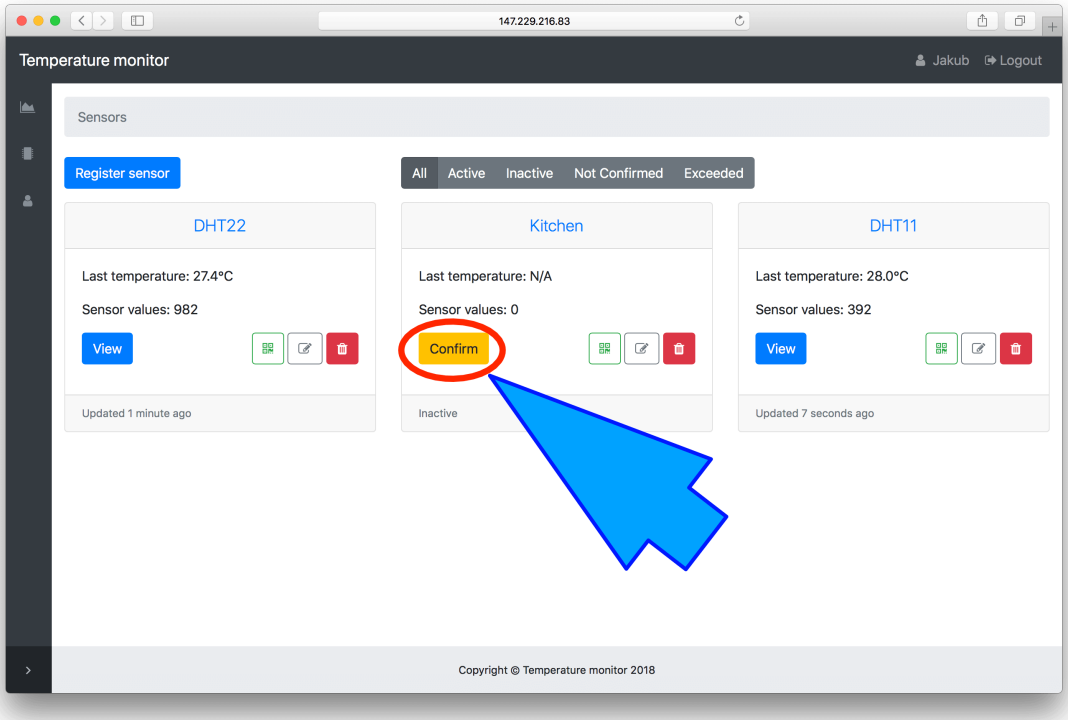
Obr. A.1: Domovská obrazovka zobrazujúca dáta zo všetkých senzorov.



Obr. A.2: Zoznam všetkých senzorov.



Obr. A.3: Generovanie QR kódu.



Obr. A.4: Potvrdenie vytvoreného senzora.

Obr. A.5: Ukážka reálne nameraných hodnôt.

Príloha B

Obsah pamäťového média

Na priloženom pamäťovom médiu sa v adresári `src` nachádzajú zdrojové súbory implementovaného systému:

- podadresár `laravel` – obsahuje implementovanú serverovú časť v jazyku PHP
- podadresár `eps` – obsahuje zdrojový kód programu pre zariadenie ESP8266
- podadresár `database` – obsahuje skript pre inicializáciu štruktúry databázy

Adresár `doc` obsahuje text tejto práce vo formáte PDF a v podadresári `tex` sa nachádzajú zdrojové súbory pre jej vygenerovanie, zaznamenané vo formáte LaTeX.