VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DOHLEDOVÝ A ŘÍDICÍ SYSTÉM PRO HOTEL REALIZOVANÝ PROSTŘEDKY IOT

SUPERVISION AND CONTROL SYSTEM FOR A HOTEL IMPLEMENTED BY IOT MEANS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATEJ HOCKICKO**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Hockicko Matej**
Program: Informační technologie
Název: **Dohledový a řídicí systém pro hotel realizovaný prostředky IoT**
Supervision and Control System for a Hotel Implemented by IoT Means
Kategorie: Softwarové inženýrství

Zadání:

1. Prostudujte problematiku návrhu a realizace řídicích systémů inteligentních budov. Seznamte se s existujícími volně dostupnými technologiemi a vhodnými technickými prostředky.
2. Stanovte požadavky na dohledový a řídicí systém pro hotel. Berte v úvahu sledování přítomnosti osob, řízení osvětlení, kamerový dohled i HVAC.
3. Zvolte vhodné komponenty (senzory, aktuátory i použitelné softwarové komponenty) a navrhněte systém podle zvolených požadavků. Umožněte dodatečné modifikace řídicího systému a rozšiřování systému o další prvky.
4. Navržený systém prototypově realizujte a otestuje jeho funkčnost. V prototypové realizaci lze částečně použít i simulované komponenty.
5. Vyhodnoťte dosažené výsledky.

Literatura:

- Home Assistant. URL: https://www.home-assistant.io
- Openhab. URL: https://www.openhab.org
- Domoticz. URL: https://www.domoticz.com
- Tasmota. URL: https://tasmota.github.io/docs/
- Automatizace. URL: http://www.automatizace.cz/

Pro udělení zápočtu za první semestr je požadováno:

- První 3 body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Táto práca rieši návrh a prototypovú realizáciu dohľadového a riadiacého systému pre hotel s využitím prostriedkov IoT. K riešeniu sa dospelo s využitím mikrokontrolérov ESP, ktoré ovladajú pripojené senzory a aktuátory. Zariadenia ESP sú ďalej združované do skupín a pripájané na cetrálny bod v dosahu. Pomocou tohto bodu sa ďalej prenáša MQTT komunikácia na cloud. V cloude beží služba IoT Core, vďaka ktorej je možné ovládať a uchovávať stav zariadení. Zároveň sa v cloude nachádza vývojový nástroj Node-RED, v ktorom bolo vytvorené grafické rozhranie pre tento systému. Výsledok tento práce umožňuje vzdialené monitorovanie a riadenie hotelu s využitím senzorov a aktuátorov pripojených do tejto siete.

Abstract

This work solves the design and prototype implementation of supervision and control system for a hotel implemented by IoT means. The solution is based on ESP microcontrollers, which control the connected sensors and actuators. ESP devices are futher grouped and connected to central point within range. Using this point, MQTT communication is futher transmitted to the cloud. The cloud runs the IoT Core service, which allows to control and store the status of devices. There is also a Node-RED developmnet tool, in which a graphical interface was created for this system. The result of this work allows remote monitoring and control of the hotel using sensors and actuators connected to this network.

Kľúčové slová

Internet vecí, inteligentná domácnosť, ESP, Onion Omega2, MQTT, AWS, Node-RED, monitorovanie, riadenie, mikrokontroléry, micropython, cloud

Keywords

Internet of Things, smart home, ESP, Onion Omega2, MQTT, AWS, Node-RED, monitoring, control, microcontrollers, micropython, cloud

Citácia

HOCKICKO, Matej. *Dohledový a řídicí systém pro hotel realizovaný prostředky IoT*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Dohledový a řídicí systém pro hotel realizovaný prostředky IoT

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Vladimíra Janouška Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Matej Hockicko
10. mája 2021

Poďakovanie

Rád by som poďakoval môjmu školiteľovi pánovi Doc. Ing. Vladimírovi Janouškovi Ph.D., za rady, nápady a pomoc pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

Spoločne s vývojom informačných technológií vzniká mnoho oblastí, ktorých cieľom je zefektívniť a uľahčiť život jednotlivca a zároveň spoločnosti. Na tento účel sa stále vytvárajú nové prostriedky a riešenia. Jednou z takýchto oblastí je aj IoT [1], ktorá v posledných rokoch zaznamenáva obrovský záujem aj zo strany širokej verejnosti.

S pribúdajúcim počtom riešení je čoraz jednoduchšie a dostupnejšie vytvorenie vlastnej inteligentnej domácnosti (smart home), vďaka ktorej je možné automatizovať mnoho opakujúcich sa úkonov, naplánovať rôzne akcie na základe zmeny určitých sledovaných parametrov, či jednoducho ovládať ktorékoľvek zariadenie v domácnosti s pomocou smartfónu či iných zariadení kedykoľvek a kdekoľvek.

S rastúcou popularitou IoT na sa trhu objavujú mnohé voľne dostupné softvérové riešenia ako napríklad Home Assistant či openHAB vyvíjané pre Raspberry Pi, rovnako ako konkrétne zariadenia s vlastným systémom, ktoré sú vyvíjané technologickými spoločnosťami ako produkty pre inteligentnú domácnosť.

IoT sa dnes dostáva do mnohých oblasti, vrátanie hotelov. IoT inovácie v hoteloch môžu prispievať k lepšej skúseností hostí, zároveň však ponúkajú možnosti ako zvýšiť profit hotela a to pomocou 3 možných stratégií: ponúknutiu hosťom komplexné zážitky, eliminovanie potreby transakčných aktivít a uvoľnenýmčasu pre zmysluplnú ľudskú prácu. [21]

Cieľom tejto práce je preskúmať možnosti IoT oblasti a ich využitie v návrhu a realizácií riadiaceho a dohľadového systému pre hotel. Úlohou tohto systému bude sledovanie prítomnosti osôb v hoteli s pomocou senzoru pohybu a kamerového dohľadu, ovládanie svetiel, meranie teploty a vlhkosti vzduchu a následné ovládanie a sledovanie stavu topenia a klimatizácie na základe nameraných hodnôt.

Prvá časť tejto práce sa zaoberá štúdiou oblasti IoT, využívaných zariadení a protokolov v tejto oblasti, za pomoci ktorých dôjde k návrhu systému. V ďalšej časti sa bude zaoberať požiadavkami na vlastný dohľadový a riadiaci systém, ktorý sa bude nachádzať v hoteli. Úlohou bude preskúmať špecifikácie hotela a stanoviť si požiadavky na systém na základe získaných znalostí z predchádzajúceho bodu. V ďalšej kapitole pôjde o transformáciu požiadaviek do návrhu systému. Následujúca časť bude pojednávať o zostavení systému podľa návrhu s využitím prostriedkov IoT a implementácii hotového riešenia. Predposledná kapitola je venovaná testovaniu systému. Na záver práce budú zhodnotená implementácia, navrhnú sa možné vylepšenia a rozšírenia pre systém a vyhodnotí sa prínos riešenia.

[1] Internet of Things - internet vecí

Kapitola 2

IoT

IoT predstavuje sieť fyzických objektov pripojených do internetu, pričom nejde len o sieť počítačov, ale zariadení všetkých typov, tvarov a veľkostí. K týmto zariadeniam môžu patriť vozidlá, smartfóny, domáce spotrebiče, hračky, fotoaparáty, lekárske prístroje, priemyselné systémy či dokonca ľudia, zvieratá alebo budovy. Ide teda o všetko čo je pripojené, komunikuje a zdieľa informácie s cieľom dosiahnuť trasovanie, kontrolu, online monitorovanie, či administrovanie. [19]

Obr. 2.1: Vizualizácia IoT siete. Obrázok prevzatý z [1]

IoT môžeme rozdeliť na 3 kategórie: monitorovanie a kontrola, big data a obchodné analýzy a zdieľanie informácií a spolupráca. Táto práca sa zaoberá najmä prvou kategóriou. Monitorovacie a riadiace systémy zhromažďujú údaje o výkone zariadení, spotrebe energie a podmienkach prostredia a umožňujú správcom a automatizovaným ovládačom neustále sledovať výkon v reálnom čase kedykoľvek a kdekoľvek. Pokročilé technológie riadenia a monitorovania môžu predpovedať budúce výsledky a optimalizovať operácie, čo vedie k nižším nákladom a vyššej produktivite. Takéto systémy sa často využívajú v inteligentných

domácnostiach, kde je ich hlavnou úlohou ochrana a úspora energie. Domáce spotrebiče a zariadenia s pomocou IoT je možné monitorovať a ovládať aj mimo domova pomocou počítača, tabletu či smartfónu. Tieto zariadenia umožňujú ovládať svetlá, nastavovať klímu, spravovať bezpečnostný systém, dostávať automatické oznámenia o udalostiach alebo odomykať a zamykať dvere. [18]

2.1 Zariadenia

Vďaka popularite IoT oblasti existuje mnoho zariadení v podobe mikrokontrolérov, ktoré za nízku cenu poskytujú dostatočný výpočtový výkon, rôzne druhy konektivity ako napríklad WiFi, Bluetooth, či pripojenie pomocou mobilných sietí.

Tieto mikokontroléry sú vybavené GPIO pinmi, vďaka ktorým vieme získavať informácie so senzorov alebo ovládať pripojené aktuátory. GPIO piny vieme nastaviť ako vstupné alebo výstupné. Ak nastavíme pin ako vstupný, tak na ňom vieme zisťovať jeho logickú úroveň. Výstupným pinom vieme nastavovať logickú úroveň. Popri digitálnych pinoch, ktoré rozlišujú len dva úrovne, 0 a 1, poznáme aj analógové piny, na ktorých vieme rozlišovať niekoľko úrovní vstupného napätia.

Na programovanie mikrokontrolérov vieme využiť programovacie jazyky ako napríklad C/C++, Lua či dokonca Python. K dispozícií je aj množstvo knižníc, ktoré uľahčujú prácu s perifériami pripojenými senzormi a aktuátormi, ale aj so samotným zariadením.

Tieto zariadenia môžu byť pripojené buďto priamo do elektrickej siete, ale takisto ich vieme aj napájať pomocou batérií. Pri napájaní z batérií nám ide o čo najnižšiu spotrebu zariadenia, aby zariadenie dokázalo fungovať napájané z batérie čo najdlhšie. Na to nám slúžia rôzne šetriace režimy, ktorých úlohou je minimalizovať spotrebu. K šetrení energie dochádza pomocou uspania častí mikrokontroléru, ktoré využívajú najviac energie. K súčastiam, ktoré najviac využívajú energiu určite patrí procesor a preto sa na jeho uspanie zameriava značné množstvo týchto režimov. Jadro mikrokontroléru je možné uspať v čase, keď práve nepotrebujeme jeho vypočetný výkon. Následné prebudenie je možné vykonať buďto pomocou časovačov, ktoré sa nachádzajú v mikrokontroléri, alebo pomocou zmeny vstupné signálu na niektorom z pinov.

2.1.1 Onion Omega2

Onion Omega2 ponúka v malom rozmerovom vyhotovení (42.9 × 26.4 × 9.9 mm) dostatok výpočtového výkonu aj na zložitejšie operácie a to vďaka 32 bitovému procesoru s frekvenciou 580MHz. Program je možné uložiť do pamäte s kapacitou 16MB, pričom kapacita RAM je 64MB. Vo variante plus ide o kapacitu 32MB, prípadne 128MB. Táto IoT platforma disponuje 15 GPIO pinmi, podporou pre komunikačné protokoly ako UART, I2C, alebo SPI a samozrejmosťou je WiFi konektivita. Toto zariadené je vybavené predinštalovaným operačným systémom na báze linuxu, ktorý je možné v prípade potreby aktualizovať. [16]

Obr. 2.2: Zariadenie Omega2. Obrázok dostupný na [2]

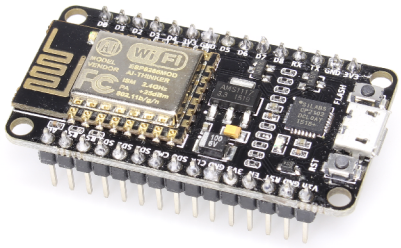
2.1.2 ESP

Čip ESP je možné využiť ako modul, ktorý po pripojení poskytuje WiFi funkcionalitu iným mikrokontrolérom, avšak vďaka jeho špecifikáciám je ho možné využiť aj ako samostatný mikrokontrolér. Keďže tento čip poskytuje GPIO piny, vieme k nemu pripojiť rôzne senzory ako napríklad senzor teploty a vlhkosti alebo napríklad CO2 senzor, z ktorých vieme získavať informácie a ďalej odosielať pomocou WiFi. Platforma ESP sa vyskytuje v rôznych verziách a vyhotoveniach, pričom každá má odlišné špecifikácie ako napríklad počet GPIO pinov, veľkosť pamäte flash, typ antény, či počet a frekvenciu jadier.

Prvá séria s označením ESP8266 ponúka jednojádrový procesor so základnou frekveciou 80Mhz, ktorú je možné v prípade potreby pretaktovať až na úroveň 160MHz. Tento čip podporuje pamäť s kapacitou až 16MB, pričom zvyčajne ide o veľkosť medzi 512KB až 4MB v závislosti od vyhotovenia.

Nástupcom ESP8266 je ESP32, ktoré je samozrejme o niečo drahšie, no na rozdiel od predchodcu ponúka jedno alebo dvojjadrový procesor, s maximálnym taktom až 240MHz, RAM pamäť o veľkosti 520kB, flash pamäť s veľkosťou 4,8 alebo 16MB a zároveň ponúka možnosť konektivity pomocou technológie Bluetooth. Pre šifrovanie sa na čipe nachádza kryptografická hardvérová akcelerácia, ktorú je možné využiť najmä v spojitosti s protokolmi AES, SHA-2 či RSA.

2.1.3 Node MCU



Obr. 2.3: Vývojová doska Node MCU. Obrázok dostupný na [3]

Node MCU je jednou z vývojových dosiek, ktorá nesie ESP čip, ku ktorému pridáva USB port na jednoduché programovanie, čím sa uľahčuje vývoj a práca so samotným čipom. Na doske sa zároveň nachádza signalizačná dióda a dvojica tlačidiel flash a reset. V prvej verzii doska obsahovala ESP8266 čip vo verzii ESP-12, neskôr však vznikla aj verzia s využitím nástupcu ESP32, ktorý prináša k WiFi konektivite takisto prítomnosť Bluetooth modulu, zvyšuje pracovnú frekvenciu a v niektorých čipoch pridáva aj jadro naviac.

2.1.4 ESP32-CAM

ESP32-CAM je vývojová doska, ktorá na sebe nesie ESP32 čip, ku ktorému pridáva možnosť priamo pripojiť modul kamery. Oproti klasickému ESP32 mikrokontroléru sa tu naviac nachádza pamäť typu PSRAM, vďaka ktorej je možné ukladať a spracovávať snímky z kamery. Na výber je veľká škála kamier, ako napríklad klasické kamery, širokouhlé kamery, či kamery s podporou nočného videnia. Na doske sa však nenachádza USB port, preto je nevyhnutný FTDI programátor, ktorý sa postará o správny prevod informácií z USB portu počítača na sériové rozhranie mikrokontroleru a späť.

Obr. 2.4: Vývojová doska ESP32-Cam. Obrázok dostupný na [4]

2.1.5 WeMos D1 Mini

Obr. 2.5: Vývojová doska D1 mini. Obrázok dostupný na [5]

WeMos D1 Mini je podobne ako Node MCU vývojová doska založená na čipe ESP8266. Podobne ako Node MCU poskytuje D1 Mini seriálové pripojenie pomocou USB vstupu, prítomnosť reset tlačidla a flash pamäť s kapacitou 4MB. Rozdielom sú však menšie rozmery dosky, vďaka čomu nájdeme na WeMos D1 Mini o pár pinov menej. Výhodou je dostupnosť rôznych doplnkov, ktoré je možné skladať na seba do výšky. Tieto doplnky poskytujú rôzne senzory, tlačidlá, obrazovky, relé, či doplnky na pripojenie batérie. Čo sa týka senzorov, tie zvyčajne komunikujú pomocou I2C zbernice, pričom majú predurčenú adresu a piny, ktoré využívajú na túto komunikáciu.

2.2 Senzory

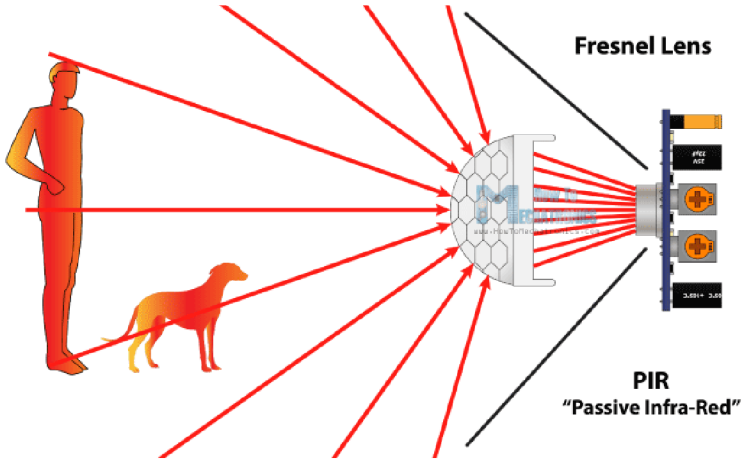
Senzory slúžia na meranie veličín z reálneho sveta. Merať môžme napríklad hodnoty teploty a vlhkosti vzduchu, tlaku, zmeny pohybu, či intenzity svetla. Tieto hodnoty sú prenášané do pripojených zariadení. Vďaka meraniu týchto veličín vieme zisťovať stav sveta okolo nás a adekvátne reagovať za pomoci mikrokontroléra a aktuátorov.

2.2.1 Pohybový senzor

Pohybové senzory sú zariadenia, ktoré reagujú na objekty, zvyčajne ľudí alebo iné živočíchy, ktoré sa pohybujú v ich dosahu. Primárne sa využívajú na zabezpečenie priestorov, či získavanie informácií o výskyte osôb v monitorovaných priestoroch. V spojitosti s aktuátormi sa tieto senzory môžu využívajú aj na automatické rozsvecovanie či zhasínanie svetiel. Poznáme niekoľko typov pohybových senzorov ako napríklad infračervené, ultrazvukové, či mikrovlné, ktoré sa líšia spôsobom akým zaznamenávajú pohyb.

Ľudské bytosti vyžarujú termálnu energiu s vlnovou dĺžkou približne 9-10 50 µm. Túto skutočnosť využívajú PIR senzory[1], ktoré sú navrhnuté na detekovanie infračervenej vlnovej dĺžky. Tieto senzory vieme kalibrovať tak, aby nezaznamenávali domáce zvieratá pomocou zvýšenia prahovej hodnoty. Princíp fungovania je možné vidieť na obrázku 2.6. [20]

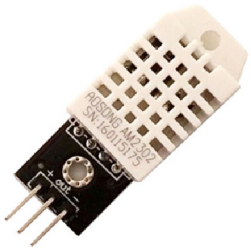
[1]Passive infrared sensor – Pasívny infračervený senzor



Obr. 2.6: PIR senzor pasívne prijíma infračervené vlny z objektov v jeho dosahu a túto zmenu signalizuje na výstupe. Prevzaté z [11]

2.2.2 DHT22

DHT22 je digitálny senzor na snímanie teploty a vlhkosti vzduchu. Tento senzor meria teplotu v rozsahu od -40 až do 80 °C s presnosťou ±0.5 °C a relatívnu vlhkosť v rozsahu 0 - 100 %. Napája sa pomocou vstupného napätia v rozsahu 3.3 až 6 V. Princíp fungovania spočíva v odoslaní start signálu z mikrokontroléra, na čo senzor prejde zo stavu s nízkou spotrebou energie do bežiaceho stavu. Po zmeraní teploty a vlhkosti senzor vyšle odpoveď s dĺžkou 40 bitov, ktorá obsahuje namerané hodnoty a následne prejde opäť do šetriaceho režimu. Pre správne fungovanie by získavanie hodnôt zo senzoru nemalo prebiehať častejšie ako každé 2 sekundy. [14]

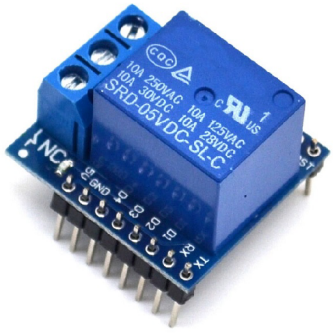


Obr. 2.7: DHT22 senzor teploty a vlhkosti. Obrázok prevzatý z [6]

2.3 Aktuátory

Aktuátor premieňa energiu na pohyb, čo znamená že aktuátory prinášajú pohyb do mechanických systémov. Ako zdroj energie využívajú hydraulickú silu alebo elektrický prúd. Aktuátory dokážu vytvoriť lineárny, rotačný alebo oscilačný pohyb. Poznáme 3 typy aktuátorov: elektrické - ac a dc motory, krokové motory a solenoidy, hydraulické - využívajú hydraulickú tekutinu na ovládanie pohybu a pneumatické - na ovládanie pohybu využívajú stlačený vzduch. Spomedzi týchto aktuátorov sú najviac využívané elektrické aktuátory. Hydraulické a pneumatické systémy umožňujú zvýšenie sily a krútiaceho momentu s využitým menšieho motora. [23]

2.3.1 Relé



Obr. 2.8: Relé modul pre vývojovú dosku WeMos D1 mini. Obrázok dostupný na [7]

Relé je jednoduchá súčiastka, ktorej úlohou je spínanie obvodov. Existuje niekoľko typov relé, pričom najvyužívanejšie je elektromagnetické relé, ktoré spína a rozopína elektrické obvody. Princíp fungovania spočíva v cievke, okolo ktorej po pripojení vzniká magnetické pole, ktoré pôsobí príťažlivo na kontakt, ktorý následne zopne obvod. Takéto relé vieme ovládať pomocou mikrokontroléru, a tým spínať obvod s oveľa vyššou úrovňou napätia, v ktorom sa môže nachádzať napríklad osvetlenie, či iné spotrebiče. Súčasťou relé sú 3 svorky označené `NO` (Normally Open), `COM` (Common pin) a `NC` (Normally Closed). Na `COM` privedieme požadované vstupné napätie. Na jednu z svoriek `NO` alebo `NC` pripojíme zariadenie ktoré chceme spínať. Svorka `NO` je v normálnom stave otvorená, to znamená, že obvod je otvorený. Svorka `NC` naopak uzatvára obvod. Po vyslaní signálu do relé sa stav týchto svoriek vymení. [12]

2.4 Kamera

Na video záznam je možné využiť kameru OV2640. Táto kamera ponúka maximálne rozlíšenie 1600x1200. Kamera zároveň ponúka možnosti nastavenia saturácie, úrovne gamma, vylaďovania hrán, či redukciu šumu. Najdôležitejšou súčasťou tejto kamery je hardvérový JPEG enkodér, ktorý odľahčuje mikrokontrolér tým, že spracovaní snímok komprimuje a tak zníži jeho veľkosť. Tým zároveň prispieva k rýchlejšiemu načítaniu a spracovaniu snímkov na strane mikrokontroléra. [10]

2.5 Programovanie

Na programovanie zariadení dnes existuje mnoho podporných programov, ktoré podstatne uľahčujú a skracujú čas potrebný pre vývoj softvéru. Mikrokontroléry ako napríklad ESP vieme programovať v rôznych jazykoch ako napríklad C/C++, Lua či MicroPython. Rozdiely medzi jazykmi sú najmä v nárokoch na hardware, v rýchlosti implementácie programu, či v rýchlosti vykonávania kódu.

2.5.1 C/C++

S využitím jazyka C a C++ vieme dosiahnuť rýchlejšie vykonávanie programu, v porovnávaní s inými jazykmi. Na programovanie v C/C++ vieme využiť Arduino IDE[2], ktoré poskytuje rôzne podporné knižnice, ktoré je možné stiahnuť priamo pomocou IDE a jednoducho využiť v prípade potreby. Tieto knižnice poskytujú určitý stupeň abstrakcie a uľahčujú prácu s rôznymi protokolmi a zariadeniami.

[2]IDE - vývojové prostredie (Integrated Development Environment)

2.5.2 MicroPython

Ak chceme naopak využiť jazyk Micropython, tak sa nezaobídeme bez nainštalovaného interpretera. Interpretovaný jazyk, akým je python, ale vyžadujú vyššie hardvérové nároky a ich vykonávanie je energeticky náročnejšie a časovo pomalšie. Výhodou tohto jazyka je však rýchlejšie prototypovanie systému, vďaka vyššej abstrakcií jazyka, či využitie rovnakého kódu na rôznych mikrokontroléroch a to bez nutnosti prekladania kódu.

Prefix micro napovedá, že ide o reimplementáciu Python jazyka, ktorá je však zameraná na mikrokontroléry a vstavané systémy. MicroPython beží priamo na zariadení, bez využitia operačného systému. Všetky operácie a služby, ktoré by pre Python poskytoval operačný systém si tak musí MicroPython zabezpečiť sám. Vďaka tomu má MicroPython kompletnú a priamu kontrolu nad hardvérom, čím sa v podstate stáva OS. MicroPython tak ponúka plnú verziu Pythonu 3 spolu s modulmi na prácu s hardvérom ako GPIO piny, rôznymi perifériami a komponentami na čipe[25].

2.6 Komunikácia

Základom internetu vecí je komunikácia. Komunikácia môže prebiehať medzi jednotlivými zariadeniami alebo so vzdialeným cloudom, kde sa často ukladajú a ďalej spracovávajú dáta zo zariadení. S využitím cloudu je takisto možné vzdialene ovládať jednotlivé zariadenia odkiaľkoľvek kde je pripojenie do siete internet. Na komunikáciu medzi zariadeniami sa používajú najmä M2M[3] protokoly.

2.6.1 M2M

Vo väčšine prípadov poskytuje bezdrôtové pripojenie najefektívnejšie pripojenie medzi zariadeniami. V tomto spojení M2M protokoly kombinujú telekomunikačné a informačné technológie. M2M využíva bezdrôtové dátové pripojenie ako spojenie medzi systémami, vzdialenými zariadeniami alebo miestami a jednotlivcami. Toto spojenie sa zvyčajne používa na zhromažďovanie informácií, nastavovanie parametrov, prijímanie alebo zasielanie indikácií o nezvyčajných situáciach alebo na obsluhu online transakcií na nespočetnom množstve zariadení ako napríklad na predajných automatoch. Vo výsledku má nasadenie M2M protokolov za následok zefektívnenie obchodných procesov a zvýšenie produktivity zamestnancov vďaka automatizácií rozhodovania pomocou monitorovania, merania, podávania správ a reakciám na tieto správy. [13]

2.6.2 MQTT

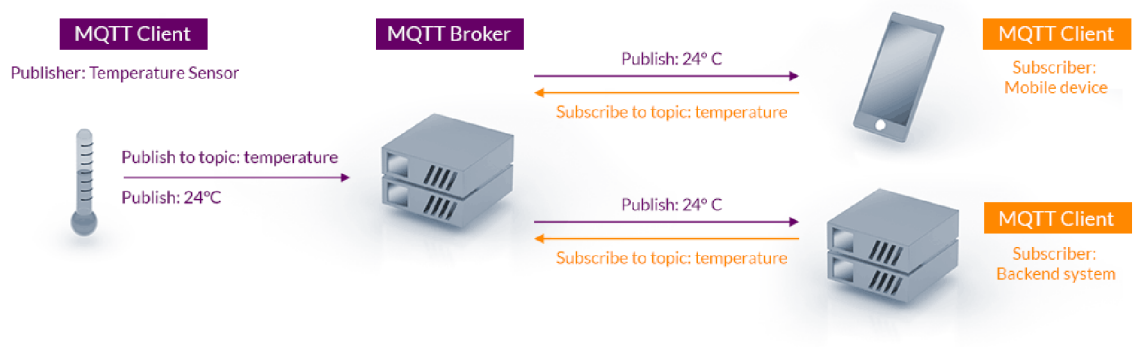
Táto sekcia bola prevzatá z MQTT dokumentácie [15].

Protokol MQTT je klient-server protokol, ktorý využíva komunikáciu štýlu publish/subscribe. Tento protokol je otvorený a jednoduchý na implementovanie. Protokol je ideálny na použitie na komunikáciu typu M2M v IoT, kde je často nevyhnutná malá veľkosť kódu a nároky na nízky dátový prenos. MQTT beží nad TCP protokolom a pri prenose poskytuje možnosť šifrovania pomocou TLS. Princíp fungovania je možné vidieť na obrázku 2.9.

[3] machine-to-machine - zariadenie zariadeniu

MQTT poskytuje niekoľko rôznych možností pre kvalitu doručenia:

- At most once (najviac raz) - správy sú doručené s využitím best effort[4] techniky. Môže dochádzať k stratám.
- At least once (aspoň raz) - správa bude doručená, ale môžu sa vyskytnúť duplikáty.
- Exactly once (presne raz) - dáta budú doručené práve raz.



Obr. 2.9: Architektúra MQTT. Klient môže vystupovať ako publisher (publikuje správy na určitú tému - topic) alebo ako subscriber (prijíma správy na určitú tému). MQTT broker prijíma správy od publikujúcich a preposiela ich na zariadenia, ktoré sa prihlásili k odberu danej témy. Prevzaté z [8]

Bezpečnosť MQTT

V základnej forme sa MQTT využíva bez akéhokoľvek zabezpečenia, šifrovania, či autentifikácie, avšak otázke bezpečnosti sa nevyhol ani tento protokol. Na transportnej vrstve máme možnosť využitia TLS/SSL protokolu na šifrované posielanie správ, čím zabezpečíme, že nebudú počas prenosu čitateľné a zároveň môžme využiť certifikáty na overenie totožnosti. Na aplikačnej vrstve vieme využiť autentifikáciu pomocou užívateľského mena a hesla. Tým znemožníme pripájanie akýchkoľvek zariadení k brokeru. [24]

2.6.3 I2C

Protokol I2C[5] je synchrónny sériový komunikačný protokol, ktorý je určený na komunikáciu viacerých periférií s jedným hlavným zariadením. Tento protokol využíva dva vodiče označované ako `SDA` a `SCL`. Na zbernici `SCL` sa prenáša hodinový signál, zatiaľ čo na zbernici SDA samotné dáta. Pomocou stavov na oboch zberniciach sa detekuje začiatok vysielania, príjem správy a koniec vysielania. [22]

2.6.4 SPI

SPI[6] sa využíva najčastejšie na komunikáciu so senzormi, analógovými a digitálnymi prevodníkmi, posuvnými registrami, SRAM či inými perifériami. V komunikácii sa vyskytujú

[4] best effort - Najlepšia snaha o doručenie, nezaručuje doručenie
[5] Inter-Integrated Circuit
[6] Serial peripheral interface - sériové periférne rozhranie

dva typy zariadení. Prvým typom je zariadenie s názvom master, ktoré sa vyskytuje vždy práve raz. Toto zariadenie generuje hodinový signál. Druhým typom sú zariadenia slave (otrok), ktorých môže byť pripojených aj niekoľko. Hodinový signál sa medzi zariadeniami posiela pomocou linky označenej `SCLK`. Na linke označenej `CS` sa prenáša signál, pomocou ktorého sa volí jedno zariadenie typu slave, s ktorým bude prebiehať komunikácia. Na komunikáciu slúžia vodiče `MOSI`[7] a `MISO`[8], ktorých názov určuje smer pohybu dát. [17]

2.7 Cloud

V spojitosti s IoT systémami sú často využívané cloudy. Tie slúžia na uchovávanie a spracovávanie informácií ako aj na riadenie zariadení, keďže ku cloudu máme prístup odkiaľkoľvek vďaka internetu. V cloude je možné uložiť množstvo informácií, z ktorých vieme následne vytvárať grafy či rôzne tabuľky, pomocou ktorých vieme sledovať trendy vo veličinách. Ďalej je možné v cloude spracovávať rozsiahle množstva dát, ktoré by sme ináč priamo na mikokontroléri nevedeli vyhodnotiť, z dôvodu nedostatočného výpočtového výkonu.

2.7.1 Google Cloud IoT

Google Cloud IoT poskytuje priestor na uchovávanie, analýzu a spracovávanie údajov z IoT zariadení. S google cloudom je možné komunikovať pomocou MQTT alebo HTTP protokolu. Informácie uložené v cloude je možné spracovávať s využitím umelej inteligencie. Umelú inteligenciu je možné využiť aj pri spracovávaní práve streamovaného videa.

2.7.2 Amazon Web Services

Podobne ako Google Cloud IoT, AWS poskytuje cloudové služby ktoré prepájajú IoT zariadenia s AWS cloudovými riešeniami. Tým pomáha vytvárať riešenia s pomocou najnovších technológií. Základom je služba IoT Core, ktorá slúži na vytváranie a spracovanie zariadení, ktoré vedia komunikovať s cloudom s využitím MQTT protokolu.

2.8 Programové vybavenie

Následujúca sekcia popisuje možnosti využitia voľne dostupných programových nástrojov, pri práci nie len s IoT zariadeniami, na zvýšenie efektívnosti vývoja, uľahčenie práce so zariadeniami a zvýšenie celkového komfortu pri práci.

2.8.1 Node-RED

Node-RED je programovací nástroj, ktorý je založený na toku informácii zostavený nad jazyk JavaScript. Pomocou tohto nástroja vieme zapisovať rôzne chovanie pomocou uzlov (nodes). Každý úzol má svoj význam - obsahuje určité dáta, spracováva ich, prijíma alebo odosiela dáta ďalej, alebo ich zobrazuje do grafickej podoby. Tieto uzly sú spojené do siete, ktorá zobrazuje tok dát. Tento nástroj je založený na grafickom rozhraní, pomocou ktorého je možné jednoducho pridávať uzly a definovať ich chovanie. Cieľom tohto nástroja je zjednodušiť prepájanie rožných aplikácii, hardverových zariadenií a API[9] medzi sebou

[7] Master out, slave in - zo zariadenia master na zariadenie slave
[8] Master in, slave out - zo zariadenia slave na zariadenie master
[9] Application Programming Interface - rohzranie pre programovanie aplikácií

navzájom. Výsledok je následne možné spustiť pomocou jedného kliknutia. Aplikácie vytvorené pomocou Node-RED je po ich vytvorení možné ďalej jednoducho zdielať a to pomocou exportovania definície toku do JSON súboru. [9]

2.8.2 Ampy

Ampy[10] je nástroj využívaný na sériovú komunikáciu so zariadeniami, na ktorých beží Micropython. S pomocou tohto nástroja je možné odosielať súbory na súborový systém zariadení, sťahovať súbory zo zariadení, získať výpis súborov na zariadení alebo prečítať obsah konkrétneho súboru, či priamo vykonávať skripty na týchto zariadeniach. Tento nástroj je napísaný v jazyku Python a používa sa pomocou prikázoveho riadku.

Základné príkazy nástroja Ampy:

- `ls` - Vypíše zoznam súborov nachádzajúcich sa na zariadení
- `get [názov súboru]` - Vypíše obsah konkrétneho súboru na zariadení
- `put [názov súboru]` - Odošle súbor na zariadenie. Je možné takisto definovať názov, pod ktorým sa súbor nakopíruje.

2.8.3 Esptool

Esptool[11] je program zostanevý v jazyku python, ktorého cieľom je správa bootloadera esp zariadení. S jeho pomocou vieme nainštalovať micropython, či vymazať pamäť zariadnia, alebo takisto overiť MAC adresu zariadenia, získať flash ID, alebo zostaviť vlastný obraz pre konkrétne zariadenie.

[10] Ampy - https://github.com/scientifichackers/ampy
[11] Esptool - https://github.com/espressif/esptool

Kapitola 3

Požiadavky na systém

Táto kapitola sa venuje požiadavkám na systém, ktoré je potrebné si stanoviť pred samotným návrhom. Jednotlivé požiadavky budú rozdelené do kategórií podľa oblasti, ktorej sa budú venovať. Podľa týchto požiadaviek sa následne vytvorí návrh riadiaceho a dohľadového systému v nasledujúcej kapitole.

3.1 Architektúra

Systém by sa mal členiť na logické oblasti, pričom v každej oblasti by sa mal nachádzať centrálny bod, ktorý bude poskytovať prístup k WiFi sieti pre koncové zariadenia pripojené k nemu. Tento bod bude sám pripojený do internetu pomocou WiFi siete alebo pomocou ethernetu. Zároveň bude toto zariadenie slúžiť ako prepojenie ku koncovej službe nachádzajúcej sa v cloude s využitím protokolu MQTT.

3.2 Sieť

Koncové zariadenia by sa mali nachádzať vo vlastnej bezdrôtovej sieti, aby sa tak oddelila dátová prevádzka. Zároveň by sa v tejto sieti nemali nachádzať iné zariadenia, ktoré sa nepodieľajú na riadiacom a dohľadovom systéme. Táto sieť by mala mať skryté SSID a byť dostatočne zabezpečená heslom s úrovňou zabezpečenia aspoň WPA2.

3.3 Prenos informácii

Prenos údajov by sa mal odohrávať na bezdrôtovej sieti medzi koncovými zariadeniami, s využitím protokolu MQTT určeného pre prenos informácii na internete vecí. Pomocou protokolu MQTT by sa mali prenášať všetky dôležité správy, ktoré by mali byť zakódované vo formáte JSON, pre jednoduché spracovanie.

3.4 MQTT

Každé koncové zariadenie by malo mať vlastnú tému, do ktorej bude zasielať informácie a poprípade prijímať informácie v prípade potreby. Každé centrálne zariadenie by malo mať takisto vlastnú tému, do ktorej budú spadať podtémy ostatných zariadení pripojených k nemu. Pomocou týchto tém by malo byť možné jednoducho rozpoznať jednotlivé konkrétne zariadenia a účel ich využitia.

3.5 Dostupnosť a riadenie

Zo zariadení by malo byť možné odčítať informácie kedykoľvek pomocou siete internet v rozumnom intervale a zároveň by malo byť možné tieto zariadenia vzdialene ovládať. Oba úkony by mali byť dostupné iba pre autentifikovaných užívateľov.

Riadenie a monitorovanie systému by malo byť dostupné aj vzdialene. Mal by sa tu nachádzať prehľad aktuálneho stavu, poprípade prehľadná história nameraných údajov, zároveň by mala byť dostupná automatické aj manuálne ovládanie prvkov HVAC.

3.6 Bezpečnosť

Všetky zariadenia by mali byť zabezpečené proti fyzickému prístupu aj prístupu z lokálnej siete. Komunikácia medzi zariadeniami pomocou WiFi musí byť takisto zabezpečená, aby nevzniklo riziko odposluchu komunikácie a jej zneužitia.

3.7 Zariadenia a senzory

Senzory by mali byť dostatočne presné, aby nedochádzalo k veľkým odchýlkam pri meraní hodnôt, aby sa tieto hodnoty čo najviac blížili skutočnosti. Zároveň by tieto hodnosti mali byť zaznamenávané v dostatočnej miere, aby bolo možné v rozumnom čase reagovať na rôzne zmeny stavu okolia.

Zariadenia, ktoré budú zhotovovať videozáznam musia mať dostatočný výkon na spracovanie videa. Dôležitým prvkom bude prenos videa po sieti, ktorý by nemal zaťažovať lokálnu sieť, zároveň by prenášané dáta mali byť v dostatočnej kvalite. Zariadenie zaznamenávajúce video by zároveň malo disponovať možnosťou ukladania tohto záznamu na lokálne vymeniteľné úložisko.

3.8 Požiadavky na softvér

Softvér by mal v dostatočnej miere poskytovať možnosť konfigurácie koncových zariadení a senzorov, bez nutnosti upravovania kódu. Túto konfiguráciu by malo byť možné jednoducho zmeniť pomocou konfiguračného súboru. V tomto súbore by sa mali nachádzať základné údaje, pomocou ktorých sa bude možné pripojiť k WiFi sieti, či MQTT brokeru, ďalej údaje o tom, či sú k danému zariadeniu pripojené senzory alebo aktuátory, knižnice potrebné na chod týchto pripojených zariadení a konfigurácia týchto zariadení, ako napríklad ako často majú tieto zariadenia merať a odosielať správy, na aké témy budú tieto údaje odosielané alebo na aké témy bude zariadenie reagovať.

Kapitola 4

Návrh dohľadového a riadacého systému pre hotel

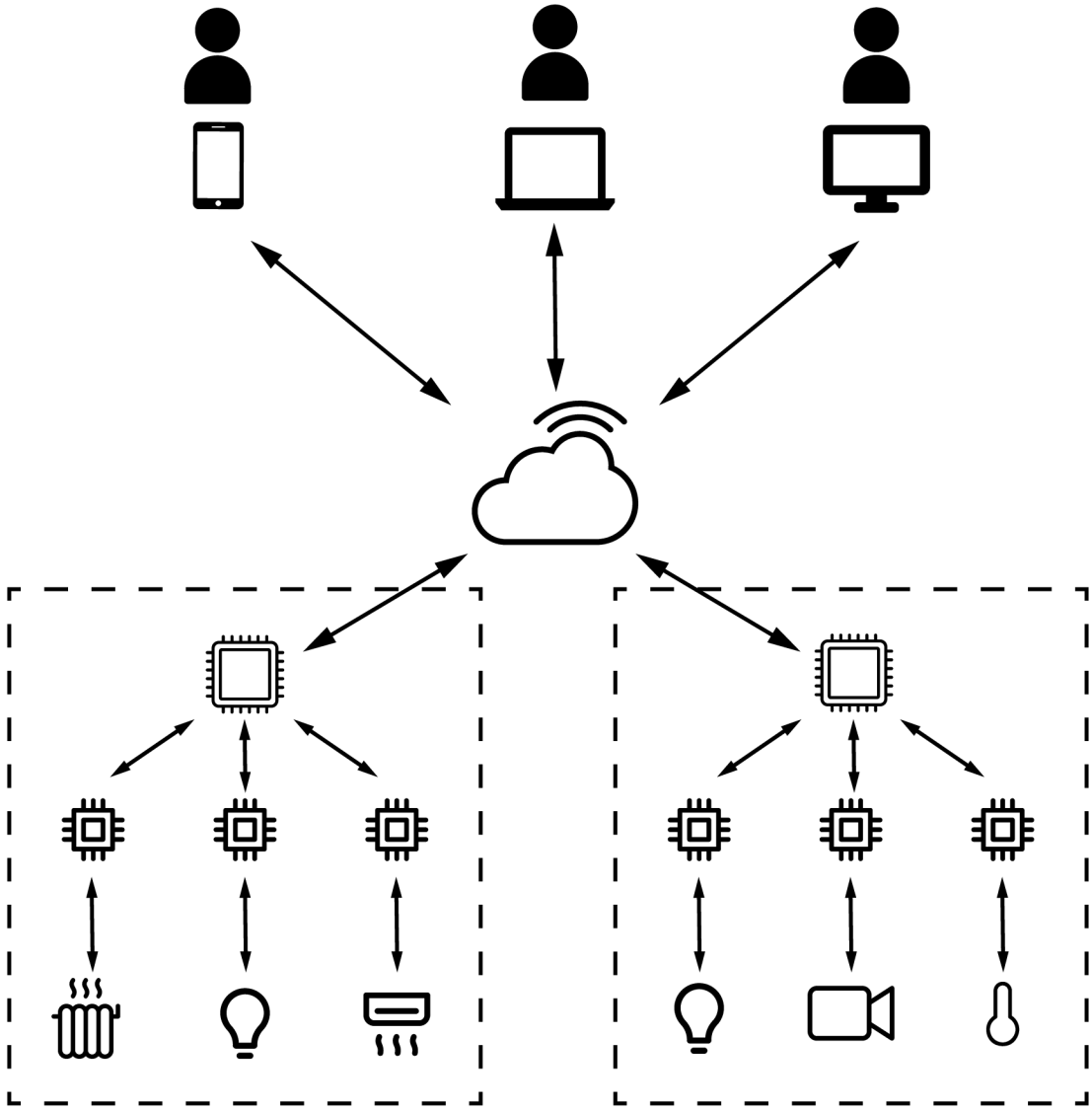
Táto kapitola popisuje spôsob fungovania celého systému. Rozoberie sa základná architektúra, jednotlivé použité zariadenia, spôsob komunikácie a predávania údajov medzi zariadeniami a takisto aj spôsob získavania dát a riadenia systému. Pri návrhu systému sa vychádza z požiadaviek, ktoré boli stanovené v predchádzajúcej kapitole.

4.1 Architektúra systému

Hlavným riadiacim bodom bude cloud, na ktorom pobeží MQTT broker a služba Node-RED. Z tohto cloudu bude možné ovládať všetky zariadenia a vyčítať namerané údaje zo senzorov a zároveň sledovať živé prenosy z kamier. Na cloud budú napojené Onion Omega2 zariadenia, ktoré budú slúžiť ako centrálne body pre svoje okolie. Každá omega bude pokrývať určitú oblasť hotela, v ktorej bude vytvárať WiFi sieť pre systém. Tieto body budú ďalej komunikovať s koncovými zariadeniami v podobe ESP mikročipov. ESP budú získavať hodnoty z pripojených senzorov, či ovládať aktuátory.

4.1.1 Architektúra siete

Pre každú fyzickú oblasť bude jedno centrálne zariadenie a teda Onion Omega 2. Táto oblasť bude fyzicky vyčlenená dosahom WiFi siete zo zariadenia. Môže ísť teda napríklad o jednu väčšiu miestnosť alebo viacero menších miestností, chodbu či iáč významovo vyhradenú oblasť. V tejto oblasti sa môže nachádzať viacero koncových zariadení, ktoré sa budú pripájať do bezdrôtovej siete vytváranej centrálnym bodom. Každá takáto oblasť bude mať svoju vlastnú tému v MQTT protokole, na jednoduché rozpoznanie jednotlivých oblastí. Schému siete je možné vidieť na obrázku 4.1



Obr. 4.1: Architektúra systému. Užívatelia sa pomocou zariadení pripájajú na cloud, odkiaľ prebieha monitorovanie a ovládanie systému. Ku cloudu sú pripojené zariadenia Onion Omega2, na ktoré sú ďalej napojené mikrokontroléry so senzormi a aktuátormi. Prerušovanou čiarou sú znázornené oblasti.

4.2 Komunikácia medzi zariadeniami

Zariadenia budú medzi sebou komunikovať pomocou MQTT protokolu. Centrálny bod v podobe Onion Omega2 bude zhromažďovať všetky údaje zo zariadení vo svojom okolí a tie bude pomocou MQTT mostu ďalej prenášať na cloud, kde sa bude nachádzať hlavný MQTT broker, poprípade na zariadenia k nemu pripojené, ktoré budú prihlásené k danej téme. Pre každý centrálny bod sa vytvorí jeden topic, do ktorého budú posielať údaje, a z ktorého budú prijímať údaje všetky pripojené koncové body.

4.3 Cloud

Pomocou cloudu bude možné sledovať namerané hodnoty. Odoslané údaje zo senzorov sa budú na cloude ukladať a vďaka tomu si bude možné pozrieť ich históriu v čase rovnako ako zobraziť si posledne získanú hodnotu. Cloud bude takisto slúžiť na ovládanie všetkých zariadení na diaľku. V neposlednom rade sa na cloud budú streamovať video záznamy z kamier, vďaka čomu bude možné sledovať živý prenos odkiaľkoľvek.

4.4 Koncové zariadenia

Ako koncové zariadenia sa využijú ESP mikrokontroléry. Tie budú v pravidelných intervaloch určených v konfigurácii zariadenia odosielať správy a kontrolovať nové správy a spracovávať ich obsah, ak sa prihlásili na odber témy. Zariadenia budú takisto kontrolovať svoj stav a v prípade zistenia chyby dôjde k automatickému reštartu zariadenia a opätovnému pripojeniu, aby sa zabezpečila maximálna možná miera dostupnosti.

4.5 Senzory

Senzory budú zaznamenávať hodnoty okolitých veličín. Tie sa budú ďalej odosielať na cloud pomocu koncových zariadení, kde sa budú ďalej spracovávať ukladať a zobrazovať. Zo serveru bude možné tieto údaje odosielať späť na zariadenia, ktoré si registrovali ich prijímanie pomocou MQTT protokolu. Môže ísť napríklad o mikrokontroléry, ktoré sa podľa aktuálnej teploty budú rozhodovať, ako zmeniť aktuálne nastavenie kúrenia či klimatizácie. Aktuálne hodnoty ako aj hodnoty v čase bude možné zobraziť v grafickom rozhraní, ktoré bude vytvorené pomocou služby Node-RED.

4.5.1 Pohybový senzor

Pohybový senzor bude slúžiť ako zabezpečovací prvok, ktorý bude snímať prítomnosť osôb v monitorovanom objekte. Monitorovanie bude prebiehať neustále. V prípade zistenia pohybu sa nastaví zmena vnútorného stavu zariadenia a v pravidelných intervaloch sa bude odosielať aktuálna hodnota stavu.

4.5.2 Senzor teploty a vlhkosti

Pomocou senzoru DHT22 sa bude zaznamenávať hodnota teploty a vlhkosti v prostredí. Pomocou aktuálnej hodnoty budu možné ďalej automaticky riadiť aktuátory, ktoré budú ovládať pripojené kúrenie či klimatizáciu.

4.6 Aktuátory

Aktuátory sa budú využívať na zmenu stavu iných systémov. Využívať sa budú najmä v spojitosti s prvkami HVAC, kedy budeme pomocou aktuátorov ovládať kúrenie a klimatizáciu. Aktuátory budú meniť svoj stav buď automaticky podľa prijímaných správ zo systému, alebo manuálne podľa pokynov užívateľa.

Vrámci aktuátorov sa využije aj relé, pomocou ktorého sa bude spínať obvod, v ktorom sa môže nachádzať osvetlenie alebo ľubovoľné zariadenie, ktoré chceme ovládať týmto spô-

sobom. Spolu so senzorom pohybu tak môže ísť o riešenie, ktoré ktoré bude automaticky zapínať svetlo pri prítomnosti osoby.

Na uchovávanie a zmenu stavu aktuátorov sa využije služba AWS IoT, ktorá obsahuje možnosť vytvárať veci (things). Pre tieto veci vieme vytvárať tiene (shadows), ktoré reprezentujú stav zariadenia, ktoré práve nemusí byť ani pripojené. Pre toto zariadenie si uchovávame aktuálny a požadovaný stav. Tieto tiene je následne možné meniť s pomocou MQTT či REST API správ.

Tieto tiene vieme ďalej využiť v službe Node-RED, ktorú napojíme na AWS a pomocou uzlov vieme naprogramovať chovanie a teda odosielanie určitých MQTT správ, na ktoré budú reagovať koncové zariadenia.

4.7 Kamerový dohľad

Na monitorovanie sa využijú ESP32-CAM vývojové dosky, ktoré obsahujú priamo port na pripojenie kamery. Na komunikáciu s kamerov sa využíva SPI rozhranie. Zariadenie bude posielať zhotovené snímky na cloud, na ktorom bude možné sledovať video naživo. Zároveň bude ESP32 vybavené modulom pre SD kartu, na ktorú bude možné ukladať nasnímané fokty a tak v prípade potreby získať kamerový záznam.

4.8 Konfigurácia ESP

Zariadenia ESP budú konfigurované pomocou JSON súboru, ktorý bude obsahovať podstatné údaje k fungovaniu celého systému. Konfigurácia bude rozdelená do niekoľkých celkov. V sekcii WiFi bude obsahovať SSID a heslo k pripojeniu sa do siete. V sekcii MQTT sa bude nachádzať konfigurácia MQTT brokera, meno a heslo zariadenia, hodnota, ktorá bude určovať interval, v ktorom sa budú kontrolovať prichádzajúce správy a témy, na ktoré sa zariadenie prihlási k odberu. Ďalej sa v konfigurácii bude nachádzať zoznam pripojených senzorov a aktuátorov. V poslednej sekcii sa bude nachádzať výčet týchto pripojených zariadení a ich konkretna konfigurácia. Ako napríklad interval získavania hodnôt a ich odosielania či témy, na ktorú budú správy pre konkrétny senzor alebo aktuátor odosielané.

4.9 Programová štruktúra koncových zariadení

Micropython na koncových zariadeniach by mal obsahovať dva hlavné súbory: `boot.py` a `main.py`. Oba súbory sa spúšťajú po pripojení k zdroju alebo po reštarte zariadenia. Prvý súbor obsahuje kód, ktorý ktorý dokončuje spúšťanie zariadenia. Druhý súbor sa spúšťa po vykonaní prvého. Ten obsahuje hlavný kód, ktorý vykonáva požadovanú funkciu.

Súbor `boot.py` nebude upravovaný. Súbor `main.py` bude zodpovedný za fungovanie koncového zariadenia. Prvým krokom je importovanie potrebných systémových knižníc. Ďalším krokom bude načítanie konfigurácie daného zariadenia nachádzajúceho sa v súbore `config.json`. O načítanie, zmenu a uloženie konfigurácie sa bude starať modul `config.py`. Následne sa importujú vlastné programové moduly a ako posledné sa naimportujú moduly konkrétne pre dané koncové zariadenie. Tieto moduly budú určené v konfigurácii zariadenia.

Základné moduly, ktoré sa budú importovať pri štarte zariadenia sú `wifi.py` a `mqtt.py`. Prvý modul zabezpečí pripojenie k wifi sieti bežiacej na zariadení Onion Omega2 pomocou údajov zadaných v konfiguračnom súbore. Druhý modul bude slúžiť na spojenie a komuni-

káciu s brokerom a takisto bude poskytovať funkcie na odosiela a prijímanie správ a na ich následné spracovanie na zariadení.

Po pripojení zariadenia do wifi siete a pripojení sa k MQTT brokerovi bude nasledovať hlavný cyklus programu, kde sa budú volať jednotlivé funkcie modulov podľa konfigurácie daného zariadenia. Mikrokontrolér bude teda zaznamenávať udalosti a odosielať ich pomocou protokolu MQTT alebo prijímať správy a reagovať na ne, alebo vykonávať oba tieto činnosti zároveň. Tieto činnosti bude vykonávať v pravidelných intervaloch.

Kapitola 5

Implementácia

Následujúca kapitola pojednáva o implementácií návrhu dohľadového a riadiaceho systému pre hotel. Táto kapitola bude členená na menšie celky, ktoré sa budú venovať podrobnejšie jednotlivým častiam implementácie. V každej časti bude popísaný postup riešenia či už hardvérovej alebo softvérovej problematiky.

5.1 Príprava zariadení

Pred prototypovaním projektu je dôležité pripraviť si zariadenia tak, aby s nimi bolo možné ďalej pracovať zamýšľaným spôsobom. Ide najmä o konfiguráciu Onion Omega 2 zariadenia, počas ktorého získame aj najnovšiu aktualizáciu operačného systému. Pri zariadeniach s ESP mikročipmi je nevyhnutná inštalácia micropythonu pred samotným programovaním.

5.1.1 Konfigurácia Onion Omega 2

Prvým krokom k využívaniu zariadenia omega je prvotná konfigurácia. Na konfiguráciu zariadenia je potrebné pripojiť sa k WiFI sieti, ktorú si zariadenie vytvorí po zapnutí. Prihlásime sa prednastaveným heslom `12345678`. Ďalej postupujeme na stránku zariadenia. Tu sa prihlásime menom `root` a heslom `onioneer`. Po úspešnom prihlásení je nevyhnutné pripojiť sa k bezdrôtovej sieti, aby sa dokončila počiatočná inštalácia. V ďalšom kroku je možnosť prihlásenia zariadenia do cloudu, tento krok sa však dá preskočiť. Posledným krokom je stiahnutie najnovšej aktualizácie a inštalácia konzoly. Po aktualizácii sa omega reštartuje a je pripravená na použitie.

5.1.2 MicroPython inštalácia

Inštalácia MicroPythonu na ESP prebieha v niekoľkých krokoch. Najprv je potrebné vyhľadať a stiahnuť aktuálnu verziu pre konkrétne zariadenie a je takisto potrebné zohľadniť dostupnú kapacitu flash pamäte. Pre jednoduchú inštaláciu MicroPythonu na mikrokontrolér je možné využiť nástroj esptool. Tento nástroj je možné nainštalovať priamo z príkazovej riadky pomocou príkazu `pip install esptool`. Nevyhnutná je prítomná inštalácia Pythonu. S využitím tohto nástroja nahráme stiahnutú verziu na vývojovú dosku. Postup inštalácie je popísaný v dokumentácii[1]. Po úspešnom nainštalovaní by malo byť možné pripojiť sa na zariadenie pomocou sériového rozhrania. Po pripojení sa dostaneme do interaktívneho prostredia, kde vieme priamo zadávať a testovať jednotlivé príkazy napísane

[1] https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html

pomocou jazyka Python. Následné programovanie pozostáva z nahratia či zmeny súboru main.py v súborovom systéme ESP.

Pre zariadenie ESP32-CAM je potrebné stiahnuť modifikovanú verziu Micropythonu, ktorá v sebe zahŕňa ovládač pre kamerový snímač, ktorý je možné k doske pripojiť. Takúto verziu je možné stiahnuť z githubu[2]. Následná inštalácia prebieha rovnako ako pri klasickej verzii Micropythonu pomocou nástroja `esptool`.

5.1.3 Konfigurácia MQTT brokera

Na využitie služby MQTT brokera na zariadení Onion Omega je využitý open source program Mosquitto[3]. Na jeho plnohodnotné využitie je potreba jeho konfigurácia. Konfiguráciu je možné zapísať do ľubovoľného súboru, ktorý sa následne predá do programu pomocou prepínača `-c`. V konfigurácii je potrebné nastaviť port služby (primárne 1883), spôsob overovania pripájania klientov, udržiavanie spojenia, prepojenie s brokerom v cloude a pod.

Na začiatok sa do konfigurácie nastavila hodnota `true` pre `allow_annonymous`, čo umožnilo pripojenie akéhokoľvek klienta bez autorizácie. Počas vývoja sa takisto využila možnosť `connection_messages` na zaznamenávanie správ o komunikácií medzi klientmi a brokerom.

Po úspešnom pripojení a zvládnutej komunikácii klientov z brokerom bolo nevyhnutné vyriešiť autorizáciu klientov klientov buď pomocou mien a hesiel alebo pomocou SSL certifikátov. Keďže použitá verzia Micropythonu nepodporovala vytváranie SSL komunikácie s pomocou certifikátu certifikačnej autority, nebolo možné koncové zariadenia pripojiť sa MQTT brokera pomocou tejto možnosti.

Na autorizáciu koncových zariadení boli vytvorené kombinácie mien a hesiel pomocou príkazovej riadky a príkazu `mosquitto_passwd` s prepínačom `-c` za ktorým nasleduje názov súboru, do ktorého sa bude ukladať kombinácia mena a hesla. Nakoniec sa zadá konkrétne meno. Po spustení príkazu a následnej výzve zadáme heslo. Takto si vygenerujeme autentifikačné údaje pre všetky zariadenia. Súbor s menami a heslami je potrebné zadať do konfigurácie brokera cestu k tomuto súboru ako hodnotu `password_file`. Zároveň zmeníme pôvodne nastavenie `allow_annonymous` na hodnotu `false`.

5.2 Konfigurácia ESP

Konfigurácia ESP sa nachádza v súbore `config.json`. Obsahuje všetky nevyhnutné údaje na správne fungovanie koncového zariadenia. Načítanie konfigurácie prebieha v module `config.py`, kde sa vytvorí premenná, ktorá sa následne importuje v prípade potreby, a z ktorej je následne možné získať potrebné údaje.

5.3 Pripojenie koncových zariadení

Následujúca sekcia popisuje pripojenie koncových zariadení v podobe ESP mikrokontrolérov k WiFi sieti a využívanie MQTT protokolu na zasielanie a prijímanie správ.

5.3.1 Pripojenie k WiFi

Pripojenie zariadení ESP k internetu prebieha automaticky po ich zapnutí v skripte main.py, ktorý využíva službu modulu `wifi.py`. Ten sa pripája k wifi na základe hodnôt v konfigu-

[2] ESP32-CAM Micropython - https://github.com/lemariva/micropython-camera-driver
[3] https://mosquitto.org/

račnom súbore. Koncové zariadenia sa pripoja k WiFi sieti, ktorú pre nich vytvára Onion Omega2. Po úspešnom pripojení k sieti nasleduje pripojenie k MQTT brokeru a nastavenie času pomocou NTP protokolu, ktorý nastaví čas modulu RTC na aktuálny čas.

5.3.2 Pripojenie k brokeru

Po úspešnom pripojení do internetovej siete prebieha pripájanie k brokeru, ktorý beží takisto na zariadení Onion Omega2. Pripojenie prebieha pomocou doménového mena zariadenia Onion omega2. Spolu s týmto menom sa ďalšie potrebné údaje ako meno a heslo zariadenia či port, na ktorý sa protokol pripojí nachádzajú v konfigurácií. Pripojenie k MQTT brokeru prebieha pomocou modulu `mqtt.py`. Tento modul využíva služieb štandardnej knižnice `umqtt` nachádzajúcej sa priamo v inštalácii micropythonu, ktorá poskytuje rozhranie pre prácu s MQTT protokolom.

5.4 Pripojenie ku Cloudu

Po úspešnom nakonfigurovaní brokera a pripojení koncových zariadení je potrebné ďalej pripojiť MQTT brokera ku cloudovej službe, pomocou ktorej bude sieť zariadení riadená a monitorovaná. V pôvodnom návrhu sa predpokladalo pripojenie k službe google IoT cloud, avšak táto služba nepodporuje Broker bridging[4], preto sa využila služba AWS IoT Core, ktorá umožňuje takéto prepojenie brokerov.

5.4.1 Pripojenie k službe IoT Core

Na pripojenie MQTT brokera k službe IoT Core najskôr potrebujeme vytvoriť politiku. Tú vytvoríme v službe IoT Core v sekcii `Secure` a podsekcii `Policies`. Následne potrebujeme povoliť práva na využívanie IoT služby zadaním akcie `iot:*` a nastavením políčka `Effect` na `Allow`. Ďalej prejdeme do sekcie `Certificates`, kde vytvoríme nový `One-click certificate`. Vytvorený certifikát, ktorý následne aktivujeme a stiahneme spolu so súkromným kľúčom. Rovnako tak stiahneme certifikát certifikačnej autority. Na záver pripojíme certifikát k vytvorenej politike.

V nasledujúcom kroku vytvoríme konfiguračný súbor pre MQTT brokera, v ktorom nastavíme adresu koncového bodu a témy, ktoré sa budú prenášať. Zároveň nastavíme verziu protokolu MQTT, ID klienta a ďalšie potrebné nastavenia. Nakoniec zadáme cesty k certifikátom a privátnemu kľúču, ktorý sme si vytvorili a stiahli v predchádzajúcom kroku. Po reštartovaní `mosquitto` služby otestujeme správne prepojenie MQTT pomocou sekcie `test` v AWS konzole v službe IoT Core[5].

5.5 Príprava Node-RED

V AWS konzole vyhľadáme službu `CloudFormation` a prejdeme na `Create stack`. Tu zadámame URL adresu `https://automationking.s3.amazonaws.com/NodeRED.yml`. Táto URL adresa obsahuje potrebnú konfiguráciu a príkazy na vytvorenie služby Node-RED. V ďalšom kroku vytvoríme názov služby a vyplníme meno a heslo pre užívateľa na pripojenie k službe. Ostatné políčka ponecháme tak, ako sú. Preskočíme následújuci krok a v posledom kroku klikneme na `Create stack`. Následne prejdeme do služby `EC2`, kde by sme

[4] Broker bridging - vzájomné prepojenie brokerov
[5] AWS MQTT test - https://console.aws.amazon.com/iot/home?region=eu-central-1#/test

mali vidieť nami novo vytvorené zariadenie. Počkáme, kým sa všetko pripraví a zariadenie prejde do bežiacého stavu. V tomto momente by sme mali byť schopný pripojiť sa na službu Node-RED pomocou verejnej IP adresy, ktorú nájdeme v popise zariadenia. Na túto službu prejdeme zadaním IP adresy do URL adresy vyhľadávača a pripojením portu 1880 na koniec. Zobraziť by sa nám mala prihlasovacia stránka, na ktorej zadáme údaje definované pri vytváraní služby. Po úspešnom prihlásení je služba pripravená na používanie.

5.6 Senzory

Následujúca sekcia popisuje postup pri práci zo senzormi, pripojenie k mikrokontroléru, komunikáciu, spôsob získavania hodnôt a odosielania údajov.

5.6.1 Zaznamenávanie teploty a vlhkosti

Teplota a vlhkosť sa zaznamenáva pomocou senzoru DHT22. Ten má tri vodiče. Prvý pin označený `+` pripájame k ESP pinu s označením 5V. Vodič s označením `-` pripájame na zem, ktorá je na zariadení ESP označená znakom G - ground. Posledný vodič s označením `out` slúži na komunikáciu so senzorom. Tento vodič pripojíme na ľubovoľný GPIO pin. Číslo tohto pinu zapíšeme do konfigurácie mikrokontroléru takisto ako názov modulu potrebného na snímanie teploty a vlhkosti.

Modul `dht_sensor.py` zodpovedný za spracovávanie údajov o teplote a vlhkosti využíva knižnicu `dht`, ktorá poskytuje základné funkcie na prácu s týmto senzorom. Dôležité je určiť typ senzora, v našom prípade ide o DHT22 a pin, na ktorom je daný senzor pripojený. Odčítanie hodnôt oboch veličín sa deje pomocou funkcie `measure`.

Samotný modul pozostáva z objektu `DHT`. Ten pri inicializácií prečíta z konfigurácie pin, ku ktorému je pripojený senzor, inicializuje samotný senzor a pripraví premennú na zaznamenávanie stavu zo senzoru. Čítanie údajov zo senzora prebieha vo funkcii `measure`, kde sa načítané hodnoty uložia do premennej `state` spolu s aktuálnym časom, kedy boli tieto údaje namerané. Poslednou funkciou je funkcia `publish`, ktorá je zodpovedná za pravidelné odčítanie hodnôt zo senzora a odosielanie pomocou MQTT protokolu. Odosielané údaje sú serializované do formátu JSON pre jednoduchšie spracovanie. Interval zberu dát je určený v konfiguračnom súbore v sekcii konkrétneho senzoru.

5.6.2 Snímanie prítomnosti osôb

Na snímanie pohybu osôb využijeme PIR senzor. Tento senzor sa podobne ako senzor teploty pripája pomocou troch vodičov. Spôsob pripojenia je rovnaký ako pri predchádzajúcom senzore. Tento konkrétny druh senzora (HC-SR501) navyše poskytuje na doske dva potenciometre, pomocou ktorých vieme nastaviť senzitivitu (to, aká veľká zmena vstupu na senzore je potrebná na zaregistrovanie pohybu) a čas (dĺžka, počas ktorej je daný stav reprezentovaný stálou logickou úrovňou na výstupnom vodiči).

Pohyb osôb a teda ich prítomnosť je zaznamenávaná pomocou čítania logickej úrovne výstupu senzoru. Pri neprítomnosti osôb je na vodiči logická 0, zatiaľ čo pri zmene a teda detekcii pohybu je na vodiči nastavená logická úroveň na hodnotu 1.

Modul zodpovedný za prácu so senzorom pohybu sa nazýva `pir.py`. Pri inicializácii senzoru nastavíme pin, na ktorom budeme čítať výstup senzoru a rovnako tak počiatočný stav. Pre tento pin zároveň nastavíme prerušenie, ktoré sa bude generovať pri nástupnej hrane a teda vždy pri prechode z logickej úrovne 0 na logickú úroveň 1. Prerušeniu nastavíme

funkciu, ktorá sa vykoná po jeho vyvolaní. Konkrétne ide o funkciu `move_detected`. V tejto funkcii nastavíme stav detekovania pohybu na `True`.

Tento modul takisto obsahuje funkciu `publish`, ktorá zodpovedá za odosielanie aktuálneho stavu v pravidelných intervaloch. Môže ísť o častejšie odosielanie v prípade potreby reagovať na udalosť v reálnom čase alebo menej časté odosielanie. K správe sa zároveň pripojí časová značka odoslania.

5.7 Aktuátory

Na ovládanie aktuátorov si pre každé koncové zariadenie vytvoríme samostanú vec (thing) v AWS IoT konzole. Pre každú vec si vytvoríme certifikáty, ktoré neskôr využijeme na pripojenie služby Node-RED k AWS. Pre každú vec si vytvoríme tieň (shadow) a určíme vlastnosti, ktorých stav chceme uchovávať.

5.7.1 Ovládanie svetiel

Na ovládanie svetiel využijeme relé. Na obsluhu relé slúži modul `relay.py`. Pri inicializácii sa z konfigurácie načíta pin, pomocou ktorého sa bude ovládať dané relé a odošle sa požiadavok na vec (thing), ktorú dané relé reprezentuje aby sa získal požadovaný stav, na ktorý sa následne relé nastaví. Následne sa v tomto module nastaví spätné volanie funkcie, ktorá je zodpovedá za zmenu stavu a reportovanie aktuálneho stavu pri prijatí požiadavku na zmenu aktuálneho stavu relé.

Funkcia `change_state` je zodpovedná za menu stavu relé pri obdržaní správy. Zároveň sa po zmene stavu odošle správa potvrdzujúca zmenu. Táto správa sa odošle len v prípade, že došlo k zmene stavu relé.

5.7.2 Kúrenie a klimatizácia

Pre kúrenie a klimatizáciu boli vytvorené moduly `heating.py` a `ac.py`. Oba moduly poskytujú rovnakú funkcionalitu ako v prípade relé. Po pripojení na MQTT brokera zašlú požiadavku na aktuálny stav ich tieňa a tento stav následne nastavia. Nakoniec nastavia spätné volanie funkcie `change_state`, ktorá bude reagovať na zmeny stavu pri prijatí správy na konkrétnu tému vzťahujúcu sa k danej veci. Keďže koncové zariadenia iba simulujú vykonávanie akcie, nedochádza k zapínaniu a vypínaniu topenia a klimatizácie, namiesto toho sa na zariadení vypína a zapína LED dióda, ktorá odráža aktuálny stav zariadenia. V budúcnosti sa predpokladá pripojenie koncového zariadenia, ktoré bude takto pomocou mikrokontroléra ovládané.

5.8 Kamerový dohľad

Na zaznamenávanie snímok sa využíva zariadenie ESP32-CAM s modulom pre SD kartu a kamerovým modulom OV2640, na komunikáciu protokol MQTT.

Na zaznamenávanie a spracovávanie obrazu bol vytvorený modul `cam.py`. Tento modul na začiatok inicializuje kameru s nastavením, ktoré je možné nastaviť v konfigurácii a zároveň pripojí prenosné úložisko v podobe SD karty, ak je v konfigurácii nastavené ukladanie fotiek pomocou funckie `mount_SD`.

Funkcia `check_free_mem` v pravidelnom intervale určenom v konfigurácii kontroluje voľné miesto na SD karte a údaje ďalej prenáša pomocou MQTT. Kapacita SD karty a

voľné miesto na karte sa vypočíta pomocou funkcie `uos.statvfs`. Táto funkcia sa zavolá s parametrom, ktorý určuje cestu, na ktorú bola pripojená SD karta. Funkcia vráti hodnoty ako veľkosť bloku, celkový počet blokov a počet voľných blokov, z ktorých sa následne dopočítajú údaje o kapacite na SD karte.

Úlohou funkcie `publish` je zaznamenať fotografiu pomocou pripojeného senzoru. Vo funkcii sa sa vytvorí úloha na kontrolu voľného miesta na karte a ďalej sa sa v cykle periodicky zaznamenávajú fotky. Podľa konfigurácie sa ďalej uloží na SD kartu, pričom ako názov sa nastaví aktuálny čas. Následne sa fotka prenesie pomocou MQTT protokolu v kódovaní Base64 a hneď za ňou nasleduje čas zhotovenia fotky v JSON formáte.

5.9 Hlavný cyklus programu

Po zapnutí zariadenia na ktorom beží micropython dôjde k spusteniu dvoch skriptov. Prvým je `boot.py`, v ktorom je možné nastaviť rôzne vlastnosti konkrétneho zariadenia pred samotným spustením. Podstatným pre chod je súbor `main.py`, ktorý sa spúšťa následne a predstavuje hlavný program vykonávaný mikrokontrolérom.

Vykonávanie programu začína načítaním potrebných knižníc. Následne sa na zariadení rozsvieti LED dióda, ktorá signalizuje pripájanie zariadenia. Ďalej sa importuje modul wifi, ktorého úlohou je pripojiť sa k WiFi sieti pomocou konfigurácie v súbore `config.json`.

Po úspešnom pripojení do siete internet nasleduje pripojenie na MQTT brokera pomocou modulu `mqtt.py`. Ten obsahuje triedu `MQTTconn`, ktorá v inicializácii vykoná pripojenie k brokeru pomocou konfigurácie v sekcii `mqtt`. V inicializácii sa volajú ďaljšie dve funkcie `connect` a `subscribe_to_all`. Prvá funkcia sa pokúša pripojiť na brokera, ak sa to nepodarí, dôjde k reštartu zariadenia. Úlohou druhej je prihlásiť sa na odber všetkých tém, ktoré sú zahrnuté v konfigurácii.

Funkcia `mqtt_msg_recaived` sa zavolá vždy pri doručení správy. V tejto funkcii sa po prijatí skontroluje zoznam obsahujúci témy, na ktorých čakajú ostatné moduly správu. Ak sa niektorá téma zhoduje, zavolá sa funkcia zodpovedná za spracovanie tejto témy. Témy z ktorých chce modul dostávať odpovede si zaregistruje pomocou funkcie `set_callback`. Do tejto funkcie sa ako parametre predá téma a funkcia, ktorá sa má spustiť po prijatí správy na konkrétnu tému.

Funkcia `wait_for_response` vykonáva aktívne čakanie na príchod správy. Táto funkcia blokuje vykonávanie programu, dokým nepríde správa na požadovanú tému. Funkcia sa využíva najmä pri aktuátoroch, kedy čakáme na príchod očakávaného stavu zariadenia.

Názov funkcie `publish` napovedá, že pôjde o odosielanie správ. Následujúca funckia `reconnect` vykonáva pravidelné odpájanie a opätovné pripájanie zariadenia k MQTT brokeru. Cieľom tejto procedúry je zabezpečiť pravidelné spojenie, ktoré sa bez tejto funkcie časom prerušilo bez detekcie tohto stavu.

Posledná procedúra slúži na skontrolovanie prichádzajúcich správ v pravidelnom intervale. Prichádzajúce správy sú spracovávané, dokým nedôjdeme na poslednú. Interval je možné nastaviť v konfigurácii.

Vykonávanie programu ďalej pokračuje importovaním modulov `sensors` a `actuators`. Prvý modul je zodpovedný za načítanie všetkých modulov pre senzory a vytvorenie objektov z ich tried. Pre každý objekt následne zaradí funkciu `publish` do harmonogramu pomocou prostriedkov knižnice `uasyncio`. Senzory v tejto funckii vykonávajú zaznamenávanie hodnôt a odosielanie správ. Druhý modul vykonáva podobnú činnosť ako prvý spomenutý, v tomto prípade však iba vytvorí objekty pre všetky aktuátory, ktoré sú zapísané v konfigurácii.

Vykonávanie súboru `main.py` pokračuje nastavením času pomocou funkcie `settime` z knižnice `ntptime`. K nastavenému času sa pripočítajú 2 hodiny, podľa časového pásma.

V tomto momente dôjde k vypnutiu led diódy, čím sa signalizuje ukončenie pripájania zariadenia. Pomocou knižnice `uasyncio` sa vytvorí nekonečný cyklus, v ktorom sa zariedenie pravidelne odpája a pripája na MQTT, zároveň sa kontrolujú prijaté správy, vykonáva sa činnosť senzorov a výsledok sa ďalej odosiela.

5.10 Monitorovanie a ovládanie

Monitorovanie a riadenie systému prebieha pomocou vývojového nástroja Node-RED. S využitím tohto nástroja bolo vytvorené užívateľské rozhranie na monitorovanie a riadenie systému pre hotel. Do vývojového prostredia Node-RED boli ďalej doinštalované ďalšie potrebné doplnky, ktoré rozšírili paletu uzlov, ktoré je možné využiť. Konkrétne šlo o doplnky `node-red-dashboard`[6], vďaka ktorému bolo možné jednoducho vytvoriť grafické užívateľské rozhranie a doplnok `node-red-contrib-ui-artless-gauge`[7], ktorý rožširuje predchádzajúci doplnok o nové uzly.

Vďaka doplnku `node-red-dashboard` vieme vytvárať grafické rozhranie, ku ktorému je možné pristúpiť pomocou URl adresy, ktorá sa skladá z IP adresy, portu 1880 a cesty `ui`. Jednotlivé uzly doplnku je možné združovať do skupín. Každému uzlu vieme nastaviť jeho šírku a výšku. Rovnako tak vieme rozmery nastaviť aj pre jednotlivé skupiny, ktorých zobrazenie vieme upraviť v nastavení doplnku. Skupiny widgetov vieme ďalej priradiť do rôznych záložiek, ktoré môžu reprezentovať rôzne oblasti systému v hoteli.

5.10.1 Pripojenie k MQTT

Na pripojenie k MQTT využívame uzly `mqtt in` a `mqtt out`. Každému uzlu potrebujeme nastaviť server, na ktorý sa má pripojiť. Zadáme teda rovnaké konfiguračné údaje ako pri nastavení MQTT brokera na zariadení Onion Omega2. Ďalej potrebujeme vytvoriť nový certifikát a kľúč pre Node-RED klienta na bezpečné pripojenie. Na vytvorenie použijeme rovnaký postup ako v 5.4.1. Povolíme bezpečné pripojenie pomocou SSL/TLS a nahrajeme potrebné certifikáty a kľúč do nastavení. Konfiguráciu uložíme a ďalej využijeme aj pri ostatných MQTT uzloch.

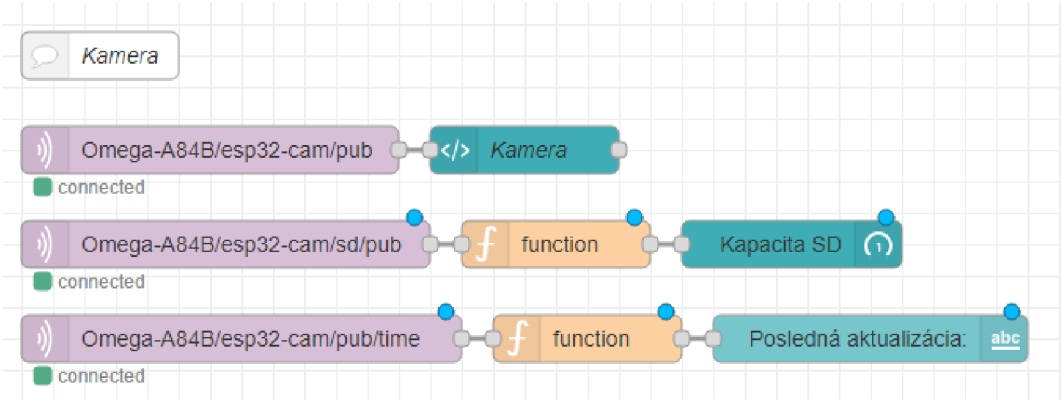
5.10.2 Kamerový dohlaď

Ako prvý prvok sa v rozhraní nachádza widget pre stream z kamery. Pomocu Node-RED boli vytvorené tri uzly. Prvý uzol prijíma údaje o obrázku v kódovaní Base64 ktoré ďalej preposiela do uzlu `template`. Tento uzol obsahuje následujúci HTML kód, ktorý využíva vstupné dáta nachádzajuce sa v objekte `msg.payload` na renderovanie obrazu.

```
<div>
    <img src="{{msg.payload}}" id='img'
    style="width:100%; height: 100%; object-fit: contain;">
</div>
```

[6] Node-RED dashboard - https://flows.nodered.org/node/node-red-dashboard
[7] Node-RED artless gauge - https://flows.nodered.org/node/node-red-contrib-ui-artless-gauge/

Druhý uzol je prihlásený na odoberanie témy, do ktorej koncové zariadenie zasiela údaje o kapacite SD karty. Za týmto uzlom nasleduje funkcia, ktorej úlohou je vypočítať aktuálne zaplnenú kapacitu SD karty v percentách, pomocou údajov o kapacite karty a voľnom mieste na karte. Výsledok sa posiela do uzlu `artless-gauge`. Tento uzol prehľadme zobrazuje využité miesto na SD karte pomocou lineárneho zobrazenia.

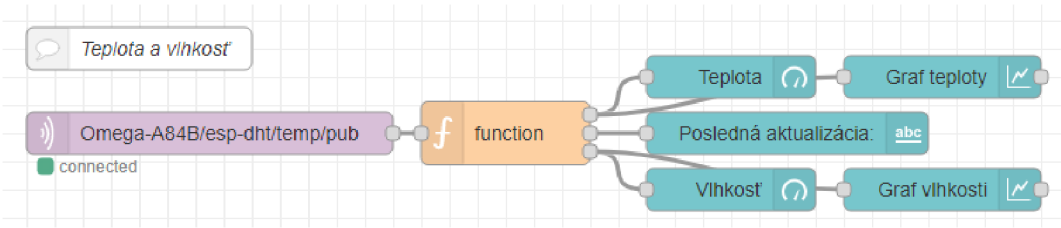


Obr. 5.1: Zobrazenie toku kamery v Node-RED

Posledný MQTT uzol čaká na príchod správ témy `Omega-A84B/esp32-cam/pub/time`. Do tejto témy je zasielaný čas vytvorenia fotky vo formáte JSON. Za týmto uzlom nasleduje funkcia, ktorá vyberie čas z prepošle ho do uzla `text`. Textový uzol zobrazí údaje o poslednej aktualizácii fotky z koncového zariadenia. Tok je možné vidieť na obrázku 5.1

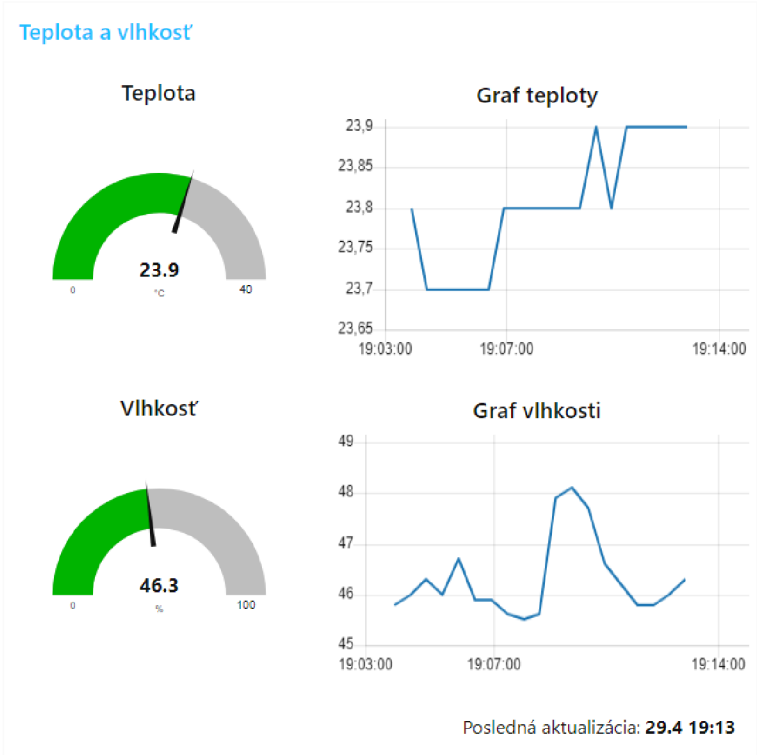
5.10.3 Monitorovanie teploty a vlhkosti

Druhým prvkom v grafickom rozhraní je monitorovanie teploty a vlhkosti vzduchu zo senzoru. Zobrazujú sa tu aktuálne dáta zo senzoru rovnako ako priebeh a zmeny nameraných hodnôt v čase pomocou grafu.



Obr. 5.2: Zobrazenie toku teploty a vlhkosti v Node-RED

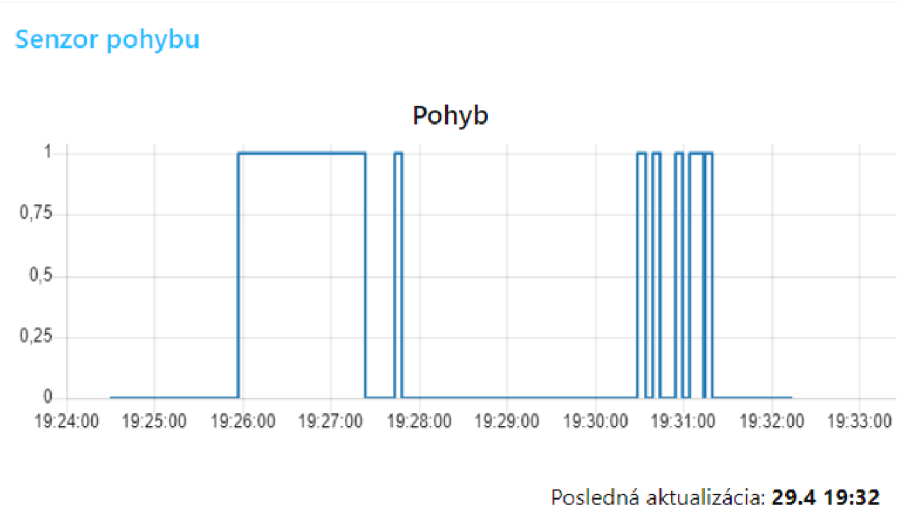
Monitorovanie prebieha pomocou jedného uzlu, ktorý prijíma správy na zadanej téme. Funkcia na ktorú sa tento uzol napája ďalej spracováva JSON správu a rozdeľuje ju na 3 výstupy. Výstup teploty a vlhkosti sa napája na uzly `gauge` a `chart`. Časový výstup sa napája na uzol typu `text`.



Obr. 5.3: Rozhranie na monitorovanie teploty a vlhkosti v Node-RED

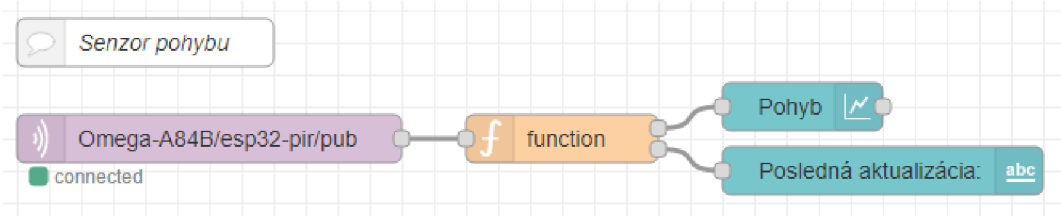
5.10.4 Monitorovanie pohybu

Tretí prvok monitorovacieho a riadiaceho rozhrania je widget zobrazujúci informácie o pohybe v monitorovanej oblasti. Nachádza sa tu graf, ktorý reprezentuje zmeny hodnoty senzoru v čase. Hodnota 0 predstavuje stály stav senzoru a teda, že nedochádza k žiadnej zmene, hodnota 1 udáva zmenu, čo znamená, že došlo k zaznamenaniu pohybu.



Obr. 5.4: Grafické rozhranie zobrazujúce zmeny v zaznamenávaní pohybu

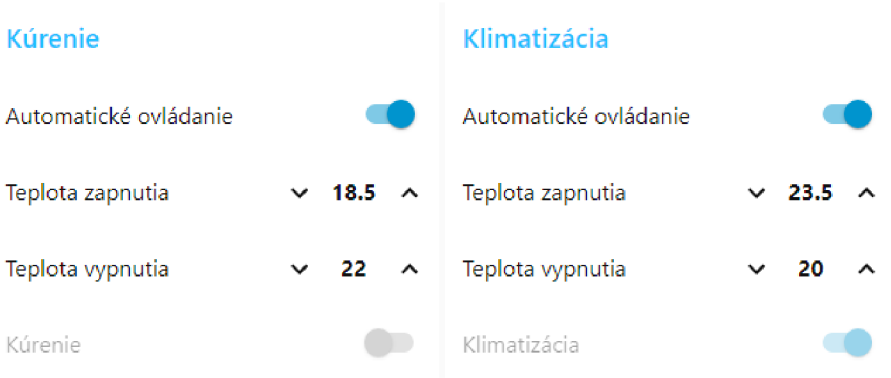
Tok sa skladá zo vstupu reprezentujúceho uzol prijímajúci MQTT správy. Prijáté dáta vo formáte JSON sú posielané do funkcie, ktorá ich rozdelí na dva výstupy pre grafový uzol `chart` a pre uzol `text` zobrazujúci čas poslednej aktualizácie stavu senzora.



Obr. 5.5: Zobrazenie toku senzora pohybu v Node-RED

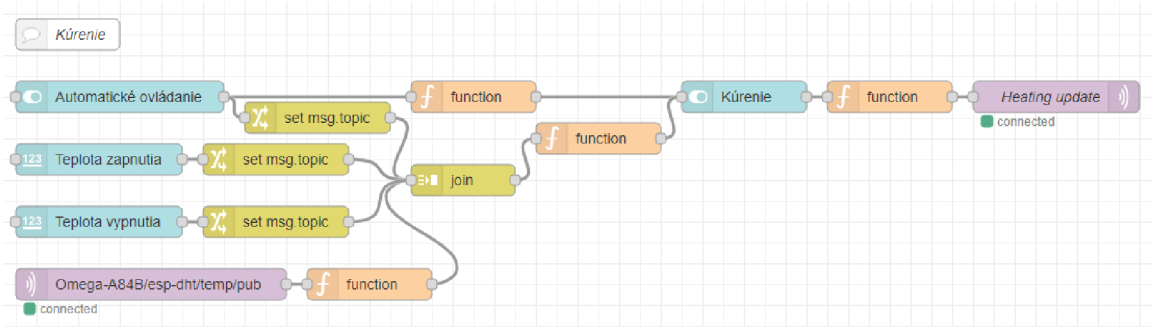
5.10.5 Kúrenie a klimatizácia

Pre kúrenie a klimatizáciu je vytvorené jednoduché ovládanie, ktorým je možné zapnúť automatické ovládanie, pri ktorom dochádza k automatickej zmene stavu pri zmene nameranej hodnoty teploty a to na základe vstupných podmienok. Úroveň teploty pri ktorej má dôjsť k zmene stavu zariadenia je možné priamo nastaviť. Prítomná je takisto možnosť manuálneho vypnutia a zapnutia, ktoré je dostupné po vypnutí automatického ovládania.



Obr. 5.6: Ovládanie kúrenia a klimatizácie pomocou grafického rozhrania

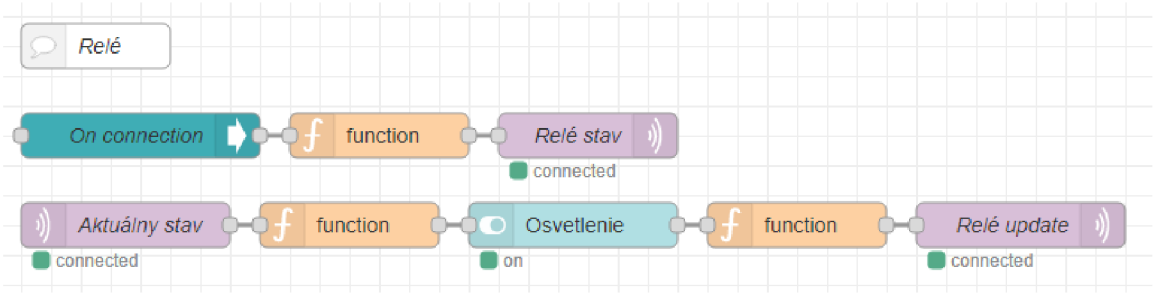
Tok riadenia kúrenia je možné vidieť na obrázku 5.7. Pozostáva zo 4 vstupov. Prvým je uzol `switch`, pomocou ktorého sa vypína alebo zapína automatické ovládanie. Po prepnutí sa výstup odosiela do funkcie, ktorá neguje tento vstup a posiela ho ďalej do uzlu `switch`, ktorý priamo ovláda kúrenie. Preposielaná správa povoľuje alebo zakazuje manuálne ovládanie kúrenia podľa stavu prepínača automatického kúrenia. Zároveň sa stav ďalej posiela do uzla `join`. Do rovnakého uzla sa pripájajú aj údaje z uzlov typu `text`, vďaka ktorým je možné nastaviť prahové hodnoty pre zapnutie a vypnutie. Ako posledný vstup do uzlu `join` vchádza hodnota teploty, ktorá sa získa zo senzoru pomocou MQTT protokolu. Uzol join spája tieto hodnoty do jednej správy, ktorú ďalej pri zmene niektorého zo vstupov posiela do funkcie, ktorej úlohou je automaticky riadiť kúrenie. Pri zmene stavu kúrenia sa tento stav ďalej reportuje odoslaním aktualizačnej správy pomocou MQTT na konkrétnu tieň veci. Na obdobnom princípe pracuje aj ovládanie klimatizácie.



Obr. 5.7: Tok ovládania kúrenia v Node-RED. Obdobný tok je použitý aj na ovládanie klimatizácie.

5.10.6 Ovládanie svetiel

Ovládanie svetiel sa skladá z dvoch tokov. Prvý tok obsahuje uzol `ui control`, ktorý má výstup nastavený na `Connect event only`. Pri pripojení na ovládací systém. Tento uzol vyšle signál, ktorý dôjde až k uzlu `mqtt out`. Ten odošle prázdnu správu na tému `$aws/things/ESP_relay/shadow/name/Relay_shadow/get`, ktorá slúži na získanie stavu. Ten príde ako odpoveď v téme `get/accepted`. Túto správu prijme na vstupe uzol s názvom `Aktuálny stav` a ďalej prepošle do funkcie, ktorá podľa obsahu správy nastaví `Osvetlenie` na požadovaný stav. Pri zmene stavu prepínača sa zároveň odošle správa na tému `update`. Pri prijatí správy sa odošle potvrdenie na tému `update/accepted`, ktorú očakáva koncové zariadenie. Tok ovládania je možné vidieť na obrázku 5.8



Obr. 5.8: Tok ovládania svetiel v Node-RED

Kapitola 6

Testovanie

Táto kapitola popisuje postup pri testovaní jednotlivých zariadení, komunikácie, monitorovania hodnôt zo senzorov a ovládanie pripojených aktuátorov. V tejto kapitole sú spomenuté problémy, ktoré sa vyskytli počas testovania a spôsob ich odstránenia. Zároveň sa v tejto kapitole spomína získavanie výsledkov z testovania a ich vyhodnotenie.

6.1 Koncové zariadenia

Testovanie koncových zariadení prebiehalo pomocou nastavenia v konfigurácii. V konfiguračnom súbore `config.json` sa nachádza možnosť `debug`, ktorej hodnota bola nastavená `true` počas testovania. V jednotlivých moduloch sa následne vyskytovali sekcie kódu, ktoré vypisovali aktuálny stav na zariadeniach na sériové rozhranie. Zariadenia boli počas testovania pripojené pomocou USB portu k počítaču, na ktorom bežal program Putty[1]. Pomocu tohto programu boli zaznamenávané správy na sériovom rozhraní. Rýchlosť sériového rozhrania bola nastavená na 115200 baud.

6.2 Onion Omega2

K zariadeniu Onion Omega2 sa počas vývoja a testovania pripájalo taktiež pomocou programu Putty. Narozdiel od koncových zariadení sa však využil protokol SSH. Testovanie MQTT brokeru prebiehalo pomocou spustenia programu na popredí pomocou prízaku `mosquitto -c /etc/mosquitto/mosquitto.conf -v`. Prepínač `-v` zaistil vypisovanie ladiacich správ priamo do konzoly.

6.3 MQTT komunikácia

Testovanie komunikácie v rámci protokolu MQTT prebiehalo prostredníctvom služby AWS IoT Core. V sekcii `test`[2] je možné prihlásiť sa na odber viacerých tém a zároveň posielať vlastné správy na ľubovoľnú tému. Za pomoci tohto nástroja sa kontrolovalo očakávané chovanie a správny formát správ. Zároveň sa pomocou odosielania správ testovali rôzne udalosti a správne chovanie koncových zariadení.

[1] Putty - https://www.putty.org/

[2] https://eu-central-1.console.aws.amazon.com/iot/home?region=eu-central-1#/test

6.4 Senzory

Očakávané chovanie senzorov bolo otestované samostatne podľa špecifikácii jednotlivých senzorov. Pri testovaní senzora pohybu bolo nevyhnutné vyladiť úroveň potenciometru tak, aby senzor dokázal reagovať na pohyby, ale aby zároveň nedochádzalo k náhodným falošným detekciám vďaka okolitým vplyvom. Na testovanie sa využila LED dióda a 5V zdroj. PIR senzor sa napojil na zdroj napätia. Na výstupný pin bola napojená LED dióda, prostredníctvom ktorej dochádzalo k signalizácii pohybu. V rámci testov bol PIR senzor namierený na časť izby, v ktorej nedochádzalo k pohybu. Testovanie prebiehalo celú noc, pričom sa neočakávala žiadna zmena, čím došlo k overeniu, že nedochádza k falošným detekciám.

6.5 Kamera

Počas testovania kamery boli skúšané rôzne rozlíšenia a rôzne nastavenia kamery ako saturácia, vyváženie bielej farby, či kvalita výsledného obrazu. Najlepšie výsledky boli dosiahnuté pomocou predvolených nastavení, preto sa tieto nastavenia nachádzajú aj v konfigurácii. Za účelom zvýšenia počtu snímkov za sekundu, však bolo nutné znížiť rozlíšenie kamery. Rozlíšenie je možné kedykoľvek podľa potreby zmeniť v konfigurácii zariadenia.

6.6 Aktuátory

Testovanie aktuátorov prebiehalo nahratím potrebných modulov a nastavením `debug` na hodnotu `true` v konfiguračnom súbore `config.json`. Pomocou testovacej sekcie v službe `IoT Core` sa odosielali správy na zmenu stavu na zariadení a zároveň sa pomocou sériovej komunikácie sledoval príjem a reakcia na správy. Po vytvorení grafického rozhrania sa aktuátory testovali priamo z tohto prostredia. Simulovaním zmien v hodnotách teploty sa zároveň testovalo automatické ovládanie kúrenia a klimatizácie.

6.7 Testovanie systému

Chovanie celého systému sa testovalo zapojením všetkých zariadení. V pravidelných intervaloch sa na senzoroch sledovalo, či dochádza k aktualizácii údajov a sledovala sa ich súvislosť s okolitým stavom. Na aktuátoroch sa sledovala vyžadovaná zmena stavu, pri reakcii so systémom, testovalo sa získanie aktuálne vyžadovaného stavu po reštarte zariadenia. Na všetkých zariadeniach sa pozorovalo pravidelné odosielanie informácii a zároveň sa testovalo opätovné automatické pripojenie pri chybovom stave.

Kapitola 7

Záver

Hlavným cieľom tejto práce bolo navrhnutie a vytvorenie monitorovacieho a riadiaceho systému pre hotel s využitím prostriedkov IoT. Pri tvorbe tohto systému sa zameriavalo najmä na sledovanie prítomností osôb, riadenia osvetlenia, kamerový dohľad a HVAC.

Systém bol navrhnutý s využitím protokolu MQTT na prenos informácií a správ vrámci tejto siete. Na správu, komunikáciu a ovládanie senzorov a aktuátorov využitých v systéme sa použili mikrokontroléry ESP. Tieto koncové body sa ďalej pripájajú na zariadenia Onion Omega2, ktorých úlohou je preposielať správy na cloud.

Cloud slúži na monitorovanie a ovládanie systému s pomocou služby Node-RED a služby AWS IoT Core. V službe Node-RED bolo vytvorené grafické užívateľské rozhranie na správu systému. Toto rozhranie poskytuje možnosť sledovania živého prenosu z kamery, vyčítanie aktuálnych hodnôt pre teplotu a vlhkosť rovnako ako vyčítanie zmien týchto veličín v čase z grafu, monitorovanie prítomnosti osôb a v neposlednom rade aj takisto ovládanie osvetlenia, či kúrenia alebo klimatizácie. Ovládať kúrenie a klimatizáciu je možné rovnako tak manuálne ako aj automaticky vďaka nastaveniu úrovní pre automatické zapnutie a vypnutie. Toto všetko je možné vykonávať na diaľku odkiaľkoľvek a kdekoľvek s prístom na internet.

Tento systém ďalej umožňuje jednoduché rozšírenie o ďalšie prvky v prípade potreby. Do budúcnosti je možné pridávať moduly pre prácu s novými senzormi a aktuátormi, vytvárať nové oblasti na monitorovanie a riadenie či vylepšovať aktuálne existujúce riešenie.

Túto prácu by som ďalej chcel využiť pri nasadzovaní systému v konkrétnom hoteli a rozširovať možnosti tohto systému o nové funkcie, ktorých cieľom bude uľahčiť riadenie a monitorovanie a zároveň zefektívniť chod hotela. Plánujem pridanie ovládania LED svetiel, ktorým bude možné nastaviť konkrétnu farbu, alebo vytvárať zaujímavé efekty, rozšíriť systém o možnosť lokálneho ovládania, bez nutnosti pripájania sa na vzdialenú službu, optimalizovať spotrebu energie zariadení za účelom napájania z batérie a mnohé ďalšie vylepšenia. Z dlhodobého hľadiska uvažujem nad vytvorením vlastného servera a služby, ktorú bude možné poskytovať aj pre ostatné hotely.

Literatúra

[1] Dostupné z: https://www.e-zigurat.com/innovation-school/wp-content/uploads/sites/5/2019/03/20190326-iot-in-five-years2-1.jpg.

[2] Dostupné z: https://onion.io/wp-content/uploads/2018/09/Omega2-Official-Shot-Omega-Only.png.

[3] Dostupné z: https://upload.wikimedia.org/wikipedia/commons/e/e5/Nodemcu_amica_bot_02.png.

[4] Dostupné z: https://www.tinytronics.nl/shop/image/cache/data/product-2132/ESP32CAM_1-1000x1000.jpg.

[5] Dostupné z: https://cdn.myshoptet.com/usr/www.laskarduino.cz/user/shop/big/620-3_foto-4.jpg?5ec787c8.

[6] Dostupné z: https://components101.com/asset/sites/default/files/components/DHT22-Sensor.jpg.

[7] Dostupné z: https://grobotronics.com/images/detailed/110/5pcs-1-channel-relay-shield-v2-version-2-for-wemos-d1-mini-esp8266-wifi-module_grobo.jpg.

[8] Dostupné z: https://mqtt.org/assets/img/mqtt-publish-subscribe.png.

[9] *About Node-RED*. [cit. 2021-4-10]. Dostupné z: https://nodered.org/about/.

[10] *OV2640 – Specs, Datasheets, Cameras, Features, Alternatives*. [cit. 2021-1-23]. Dostupné z: https://www.arducam.com/ov2640/.

[11] *PIR sensor working principle*. [cit. 2021-1-21]. Dostupné z: https://robu.in/wp-content/uploads/2020/05/PIR-Motion-Sensor-How-It-Works.png.

[12] *Relé v praxi, co to je a k čemu slouží?* [online]. [cit. 2021-3-25]. Dostupné z: https://ocemsemluvi.cz/rele-v-praxi-co-to-je-a-k-cemu-slouzi/.

[13] Alam, M., Nielsen, R. H. a Prasad, N. R. The evolution of M2M into IoT. 2013, s. 112–115. DOI: 10.1109/BlackSeaCom.2013.6623392.

[14] Aosong Electronics Co., L. *Digital-output relative humidity temperature sensor/module DHT22*. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[15] Banks, A., Briggs, E., Borgendale, K. a Gupta, R. *MQTT Version 5.0* [online], 7. marca 2019 [cit. 2021-16-01]. Dostupné z: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[16] Corporation, O. *Onion Omega2 overview*. 2020 [cit. 2021-1-17]. Dostupné z: https://onion.io/omega2/.

[17] Dhaker, P. Introduction to SPI Interface. *Analog Dialogue*. 2018, zv. 52, č. 9. Dostupné z: https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf.

[18] In Lee, K. L. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*. 2015, zv. 58, č. 4, s. 431–440.

[19] Keyur K Patel, S. M. P. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *International Journal of Engineering Science and Computing*. 2016, zv. 6, č. 5. Dostupné z: http://ostadr.ir/trans/iot/i4.pdf.

[20] Kodali, R. K., Jain, V., Bose, S. a Boppana, L. IoT based smart security and home automation system. In: *2016 International Conference on Computing, Communication and Automation (ICCCA)*. 2016, s. 1286–1289. DOI: 10.1109/CCAA.2016.7813916.

[21] Narraidoo, G. *The Role of Internet of Things in Hotels' Profitability*. 2020. Dizertačná práca. Walden University. Dostupné z: https://scholarworks.waldenu.edu/dissertations/8957/.

[22] SFUPTOWNMAKER. *I2C* [online]. [cit. 2021-3-25]. Dostupné z: https://learn.sparkfun.com/tutorials/i2c/all.

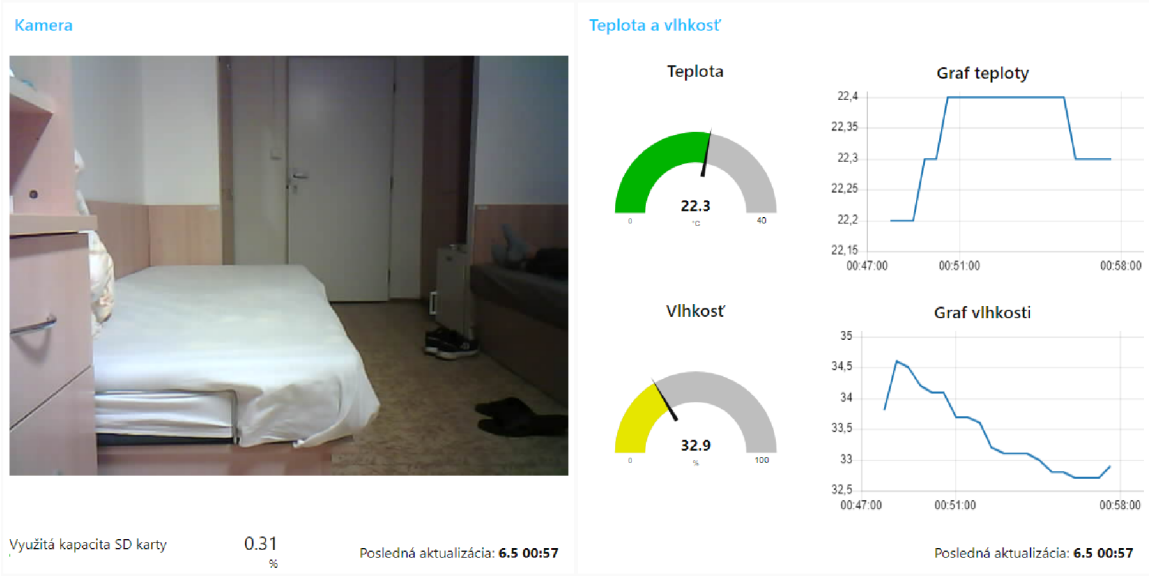
[23] Somayya Madakam, S. T. Internet of Things (IoT): A Literature Review. *Journal of Computer and Communicationss*. 2015, zv. 3, č. 5, s. 164–173. DOI: 10.4236/jcc.2015.35021.

[24] Team, T. H. *MQTT Security Fundamentals* [online]. 2015 [cit. 2021-3-25]. Dostupné z: https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/.

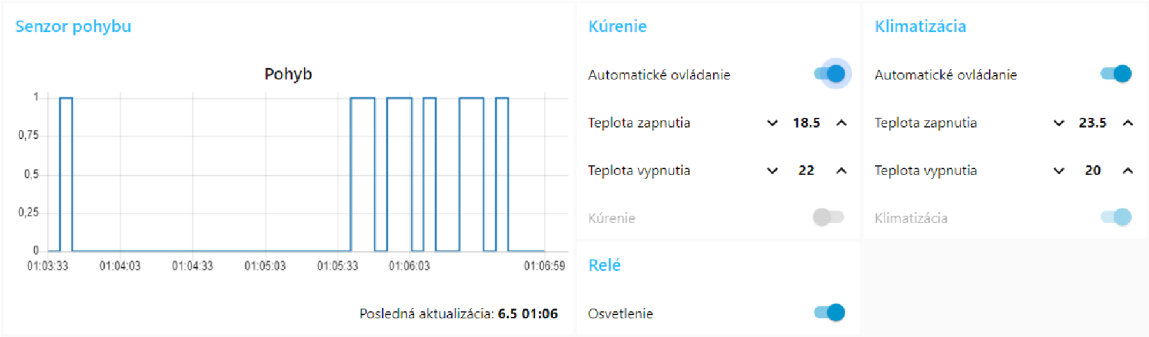
[25] Tollervey, N. H. *Programming with MicroPython: Embedded Programming with Microcontrollers & Python*. 1. vyd. O'Reilly Media, 2017. ISBN 9781491972731.

Príloha A

Grafické rozhranie na monitorovanie a riadenie systému



Obr. A.1: Grafické rozhranie monitorovacieho a riadiaceho systému. Časť 1/2



Obr. A.2: Grafické rozhranie monitorovacieho a riadiaceho systému. Časť 2/2