

Návrh a vývoj domácího asistenta s mikro- počítačem ESP-32 s rádiovým modulom 433 MHz

Šimon Bučka

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Šimon Bučka
Osobní číslo: A22471
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Prezenční
Téma práce: Návrh a vývoj domácího asistenta s mikropočítačem ESP-32 s rádiovým modulem 433 MHz
Téma práce anglicky: Design and Development of a Home Assistant with a Microcomputer ESP-32 with a 433 MHz Radio Module

Zásady pro vypracování

- Seznamte se se základy domácích elektronických asistentů. Seznamte se s mikropočítačem ESP-32, ATtiny85, rádiovým modulem 433 MHz a jejich programováním.
- Navrhněte a otestujte domácího asistenta pro monitorování a sběr údajů ze senzorů pomocí rádiové komunikace na frekvenci 433 MHz. Údaje zasílejte a zpracovávejte na vzdáleném serveru pomocí protokolu MQTT.
- Navrhněte a implementujte uživatelsky přívětivé administrátorské rozhraní pro správu všech senzorů v systému.
- Zhodnoťte možnosti a limity navrženého systému.

Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Slovenština**

Seznam doporučené literatury:

1. CAMERON, Neil. *Electronics Projects with the ESP8266 and ESP32: Building web pages, applications, and WiFi enabled devices*. Berkeley, CA: APress, 2021. ISBN 978-1-4842-6335-8.
2. HODDIE, Peter a PRADER, Lizzie. *IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK*. United States: Apress, 2020. ISBN 9781484250693.
3. AZIZ, Benjamin; ARENAS, Alvaro a CRISPO, Bruno (ed.). *Engineering Secure Internet of Things Systems*. London, United Kingdom: The Institution of Engineering and Technology, 2016. ISBN 978-1-78561-053-0.
4. SINGH, Rajesh; GEHLOT, Anita; GUPTA, Lovi Raj; SINGH, Bhupendra a TYAGI, Priyanka. *Getting Started for Internet of Things with Launch Pad and ESP8266*. Gistrup, Denmark: River Publishers, 2019. ISBN 978-87-7022-068-2.

Vedoucí bakalářské práce: **doc. Ing. Martin Adámek, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo - bakalářskou práci - nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Bakalárska práca popisuje návrh a implementáciu systému domáceho asistenta založeného na mikropočítači ESP32 v kombinácii s rádiovým modulom pracujúcim na frekvencii 433 MHz. Systém umožňuje monitorovanie údajov z teplotných a vlhkostných senzorov poskytutých spoločnosťou SL-LOGIC, ich spracovanie prostredníctvom protokolu MQTT a vizualizáciu cez webové rozhranie. Práca zahŕňa hardvérovú aj softvérovú časť, pričom integruje Node.js server pre spracovanie dát, React aplikáciu pre používateľské rozhranie a demonštruje možnosti ovládania svetla pomocou bezdrôtovej komunikácie. Výsledkom práce je funkčný prototyp domáceho asistenta s modulárnou architektúrou umožňujúcou ďalšie rozširovanie funkcionalít.

Kľúčové slová: internet vecí, ESP32, ATTiny85, 433 MHz, MQTT, domáci asistent, monitorovanie senzorov, webová aplikácia

ABSTRACT

This bachelor thesis describes the design and implementation of a home assistant system based on the ESP32 microcontroller combined with a 433 MHz radio module. The system enables monitoring data from temperature and humidity sensors provided by SL-LOGIC company, processing them via the MQTT protocol, and visualizing them through a web interface. The work includes both hardware and software components, integrating a Node.js server for data processing, a React application for the user interface, and demonstrates the possibilities of controlling lights using wireless communication. The result is a functional home assistant prototype with a modular architecture allowing further expansion of functionalities.

Keywords: internet of things, ESP32, ATTiny85, 433 MHz, MQTT, home assistant, sensor monitoring, web application

Týmto by som sa chcel poďakovať doc. Ing. Martinovi Adámkovi, Ph.D. za odborné vedenie a cenné rady počas spracovania práce. Ďakujem aj Ing. Stanislavovi Zúdorovi, zakladateľovi a majiteľovi spoločnosti SL-LOGIC, za návrh témy, dodanie senzoru na zber dát a pomoc pri realizácii projektu. Ďakujem aj mojej rodine a priateľom za podporu a povzbudenie počas celého štúdia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD	13
I TEORETICKÁ ČASŤ	14
1 INTERNET VECÍ A KOMUNIKAČNÉ TECHNOLOGIE.....	15
1.1 HISTÓRIA A SÚČASNOSŤ IoT	15
1.2 KOMUNIKAČNÉ PROTOKOLY	16
1.2.1 Komunikačný protokol ZigBee	16
1.2.2 Komunikačný protokol LoRa.....	16
1.2.3 Komunikačný protokol MQTT.....	16
2 INTELIGENTNÉ DOMÁCNOSTI	18
2.1 VIRTUÁLNI ASISTENTI	18
2.2 ROZDELENIE DOMÁCICH A VIRTUÁLNYCH ASISTENTOV	18
2.2.1 Home Assistant	18
2.2.2 Google Assistant	19
2.2.3 Amazon Alexa	19
2.2.4 Apple HomeKit.....	19
2.2.5 OpenHAB.....	20
2.3 BEZPEČNOSŤ A RIZIKÁ INTELIGENTNÝCH DOMÁCNOSTÍ	20
2.3.1 Bezpečnostné riziká	20
2.3.2 Odporúčania pre zabezpečenie inteligentnej domácnosti	20
3 MIKROKONTROLÉRY, ICH SÚČASTI A PROGRAMOVANIE	22
3.1 SÚČASTI MIKROKONTROLÉRA	22
3.2 DELENIE MIKROKONTROLÉROV	22
3.2.1 Delenie podľa šírky slova	23
3.2.2 Delenie podľa architektúry	23
3.2.3 Delenie podľa inštrukčnej sady	24
3.3 MIKROKONTROLÉRY PRE IoT	24
3.3.1 Mikrokontrolér ESP8266.....	24
3.3.2 Mikrokontrolér ESP32	25

3.3.3	Mikrokontrolér ATTiny85	25
3.4	PROGRAMOVANIE MIKROKONTROLÉRA	26
3.4.1	Programovacie jazyky pre mikrokontroléry	26
3.5	VÝVOJOVÉ PROSTREDIA	27
3.5.1	Vývojové prostredie Arduino IDE	27
3.5.2	Vývojové prostredie Espressif-IDE	28
3.5.3	Vývojové prostredie Visual Studio Code spolu s PlatformIO	28
3.6	VÝVOJOVÉ PLATFORMY	28
3.6.1	Vývojové dosky Espressif	28
3.6.2	Vývojové dosky Arduino	29
3.6.3	Jednodoskové počítače Raspberry Pi	30
4	MODULY A SENZORY PRE DOMÁCICH ASISTENTOV	31
4.1	RÁDIOVÉ MODULY	31
4.1.1	Typy rádiových modulov	31
4.1.2	Modulácia signálu.....	32
4.1.3	Využitie rádiových modulov	32
4.1.4	433 MHz rádiový modul	33
4.2	SENZORY PRE MONITOROVANIE PROSTREDIA.....	33
4.2.1	Teplotné a vlhkostné senzory	34
4.2.2	Ďalšie typy senzorov pre inteligentné domácnosti.....	34
5	POUŽÍVATEĽSKÉ ROZHRAKIE A POUŽÍVATEĽSKÁ SKÚSEKOST'	35
5.1	PRINCÍPY UI A UX DIZAJNU	35
5.1.1	Responzívny dizajn	35
5.1.2	Prístupnosť a použiteľnosť používateľského rozhrania	35
5.2	MODERNÉ TECHNOLOGIE POUŽÍVATEĽSKÉHO ROZHRAKIA	36
5.2.1	Knižnica React.....	36
5.2.2	Iné populárne technológie používateľského rozhrania	37
5.3	SERVEROVÁ ČASŤ A API	37
5.3.1	Express.js a REST API	37
5.4	AUTENTIFIKÁCIA A BEZPEČKOST'.....	38

5.4.1	JWT autentifikácia	38
II	PRAKTICKÁ ČASŤ	39
6	ANALÝZA A NÁVRH SYSTÉMU	40
6.1	POŽIADAVKY NA SYSTÉM	40
6.1.1	Funkčné požiadavky	40
6.1.2	Požiadavky zamerané na vlastnosti systému	40
6.2	CELKOVÁ ARCHITEKTÚRA SYSTÉMU	41
6.2.1	Hardvérová vrstva.....	41
6.2.2	Komunikačná vrstva	42
6.2.3	Softvérová vrstva.....	42
6.2.4	Tok dát v systéme	42
6.3	NÁVRH HARDVÉROVEJ ČASTI	43
6.3.1	Kľúčové komponenty	43
6.4	NÁVRH SOFTVÉROVEJ ČASTI	43
6.4.1	Architektúra serverovej časti	43
6.4.2	Architektúra používateľského rozhrania.....	44
6.5	NÁVRH DATABÁZOVÉHO MODELU	45
6.5.1	Popis tabuliek a vzťahov	45
7	IMPLEMENTÁCIA HARDVÉROVEJ ČASTI.....	46
7.1	VÝVOJOVÉ PROSTREDIE A POUŽITÉ JAZYKY	46
7.1.1	Arduino IDE	46
7.1.2	Programovací jazyk C++	46
7.2	SCHÉMA ZAPOJENIA A POUŽITÉ KOMPONENTY NA OVLÁDANIE SVETLA..	46
7.3	IMPLEMENTÁCIA OVLÁDANIA SVETLA.....	47
7.3.1	Hardvérová konfigurácia	47
7.3.2	Softvérová implementácia.....	48
7.3.3	Vysielacia jednotka	48
7.3.4	Prijímacia jednotka	50
7.4	INTEGRÁCIA SL-LOGIC TEPLOTNÉHO A VLHKOSTNÉHO SENZORA.....	50
7.4.1	Proces konfigurácie senzora	51

8	IMPLEMENTÁCIA SOFTVÉROVEJ ČASTI	52
8.1	VÝVOJOVÉ PROSTREDIE A TECHNOLOGIE	52
8.2	SERVEROVÁ ČASŤ.....	52
8.2.1	Výber a konfigurácia MQTT brokera.....	52
8.2.2	Nasadenie systému pomocou Docker kontajnerov	52
8.2.3	Implementácia Node.js servera	52
8.2.4	Spracovanie MQTT správ	53
8.2.5	Spracovanie, úprava a ukladanie dát	55
8.3	POUŽÍVATEĽSKÉ ROZHRIANIE.....	56
8.3.1	Štruktúra React aplikácie.....	56
8.3.2	Implementácia používateľského rozhrania	57
8.3.3	Ovládanie svetiel a vizualizácia dát.....	57
8.4	KOMUNIKÁCIA MEDZI KOMPONENTAMI SYSTÉMU	59
8.5	AUTENTIFIKÁCIA A AUTORIZÁCIA.....	59
9	TESTOVANIE SYSTÉMU	60
9.1	TEST SENZORU	60
9.2	TESTOVANIE KOMUNIKÁCIE.....	60
9.3	TESTOVANIE OVLÁDANIA SVETLA.....	61
	ZÁVER.....	62
	ZOZNAM POUŽITEJ LITERATÚRY	63
	ZOZNAM OBRÁZKOV.....	71
	ZOZNAM TABULIEK	72
	ZOZNAM VÝPISOV.....	73
	ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK	74
	ZOZNAM PRÍLOH	77

ÚVOD

Inteligentné domácnosti a systémy domácej automatizácie predstavujú v súčasnosti jedno z najrýchlejšie sa rozvíjajúcich odvetví internetu vecí. Tieto systémy umožňujú monitorovanie a riadenie domácich zariadení, čím poskytujú používateľom vyššiu mieru komfortu, bezpečnosti a energetickej efektivity. S rastúcim záujmom o tieto technológie vzniká potreba vývoja dostupných riešení, ktoré dokážu zbierať, spracovávať a vizualizovať údaje zo senzorov v domácnosti.

Táto bakalárska práca sa zameriava na návrh a vývoj domáceho asistenta založeného na mikropočítači ESP32 v kombinácii s rádiovým modulom pracujúcim na frekvencii 433 MHz. Cieľom je vytvoriť systém, ktorý dokáže monitorovať údaje zo senzorov teploty a vlhkosti, spracovávať ich na serveri prostredníctvom protokolu MQTT a poskytovať používateľovi jednoduché rozhranie pre správu systému a vizualizáciu údajov. Súčasťou riešenia je aj prototypová implementácia ovládania svetla.

Aktuálny stav v oblasti domácich asistentov ponúka mnoho komerčných riešení, ktoré sú často nákladné a limitované z hľadiska rozšíriteľnosti. Existujúce open-source platformy zase často vyžadujú pokročilé technické znalosti pre ich implementáciu. Práca sa preto zameriava na vytvorenie kompaktného a prívetivého používateľského rozhrania.

Projekt bol vypracovaný na základe požiadaviek a v spolupráci so spoločnosťou SL-LOGIC, ktorá poskytla potrebné senzorové moduly. Toto prepojenie s komerčnou sférou dodáva práci praktický rozmer a potenciál pre reálne využitie. Očakávaným prínosom je vytvorenie funkčného prototypu domáceho asistenta s dôrazom na robustnosť a možnosti ďalšieho rozširovania.

I TEORETICKÁ ČASŤ

1 INTERNET VECÍ A KOMUNIKAČNÉ TECHNOLOGIE

Internet vecí (po anglicky internet of things, ďalej IoT), predstavuje dynamickú globálnu sieť ovú infraštruktúru, ktorá umožňuje prepojenie fyzických a virtuálnych objektov prostredníctvom štandardizovaných komunikačných protokolov schopných vymieňať si a využívať informácie. [1, 2]

Táto kapitola sa zameriava na základné aspekty, ktoré sú potrebné na pochopenie fungovania IoT. Začína sa definovaním pojmu IoT, históriou a pokračuje sa komunikačnými protokolmi a technológiami potrebnými na zabezpečenie bezproblémovej komunikácie medzi zariadeniami. Na záver kapitoly je predstavený komunikačný protokol MQTT, ktorý je jedným z najpoužívanějších protokolov v oblasti IoT.

1.1 História a súčasnosť IoT

Začiatky IoT siahajú až do 19. storočia a súvisia s vynálezom telegrafu a jeho schopnosti prenášať informácie kódovaným signálom. Skutočné základy IoT boli však položené v 60. rokoch 20. storočia a sú spojené s vývojom ARPANET-u siete vytvorenej Agentúrou pre pokročilé výskumné projekty (ARPA) Ministerstva obrany USA. Táto sieť bola predchodcom dnešného internetu. V 80. rokoch sa rozšírili lokálne počítačové siete (LAN), čo umožnilo efektívny a široko používaný spôsob komunikácie a zdieľania dokumentov, údajov a iných informácií v rámci skupiny počítačov v reálnom čase. [3]

Významný posun nastal v 90. rokoch s rozvojom internetu, ktorý sa globálne rozšíril. V roku 1999 Ashton vo svojom prejave predstavil pojem IoT ako názov prezentácie pre Procter & Gamble (Americká drogistická firma) na prepojenie novej myšlienky RFID v dodávateľskom reťazci P&G. [4, 3]

Následný pokrok v spracovaní dát, cloud computingu, senzorových technológiách a bezdrôtovej konektivitě umožnil komunikáciu s digitálnymi zariadeniami a fyzickými objektmi v reálnom čase. V dôsledku toho sa inteligentné dopravné siete, prepojené skladovacie nádrže a priemyselné robotické systémy stali normou. [3]

Dnes IoT zahŕňa široké spektrum aplikácií, od inteligentných dopravných sietí až po monitorovanie životného prostredia či optimalizáciu poľnohospodárstva. Neustále sa vyvíja a nachádza uplatnenie v priemysle, zdravotníctve a ďalších oblastiach. [3]

1.2 Komunikačné protokoly

V oblasti IoT existuje množstvo komunikačných protokolov, ako napríklad ZigBee, LoRa, MQTT, CoAP, HTTP a ďalšie. Každý z nich má svoje špecifické vlastnosti a je navrhnutý pre rôzne scenáre použitia. Výber vhodného protokolu závisí od požiadaviek na prenos dát, od energetickej efektívnosti, od dosahu, od bezpečnosti či počtu pripojených zariadení. [5]

V nasledujúcej časti sú predstavené vybrané komunikačné protokoly využívané v IoT, pričom sa zameriavame na ich technické vlastnosti, výhody a typické oblasti nasadenia.

1.2.1 Komunikačný protokol ZigBee

ZigBee je bezdrôtový mesh protokol s nízkou spotrebou energie navrhnutý pre automatizáciu domácností a IoT aplikácie. Poskytuje prenosovú rýchlosť 20-250 kbps s dosahom 10-100 metrov, pričom jeho mesh architektúra umožňuje rozšírenie siete až na 65 000 uzlov. Vyznačuje sa nízkou odozvou a využíva AES-128 šifrovanie pre zabezpečenie komunikácie. [5]

1.2.2 Komunikačný protokol LoRa

LoRa je protokol určený pre IoT aplikácie vyžadujúce veľký dosah a nízku spotrebu energie. Dosahuje vzdialenosti 3 až 15 km v závislosti od prostredia, pričom zariadenia môžu na batérie fungovať niekoľko rokov. Sieť zahŕňa koncové zariadenia, brány, server a cloud, s prenosovou rýchlosťou 0,3-37,5 kbps a využíva AES-128 šifrovanie. Je ideálna pre smart cities, priemyselný IoT, monitorovanie prostredia a logistiku. [5]

1.2.3 Komunikačný protokol MQTT

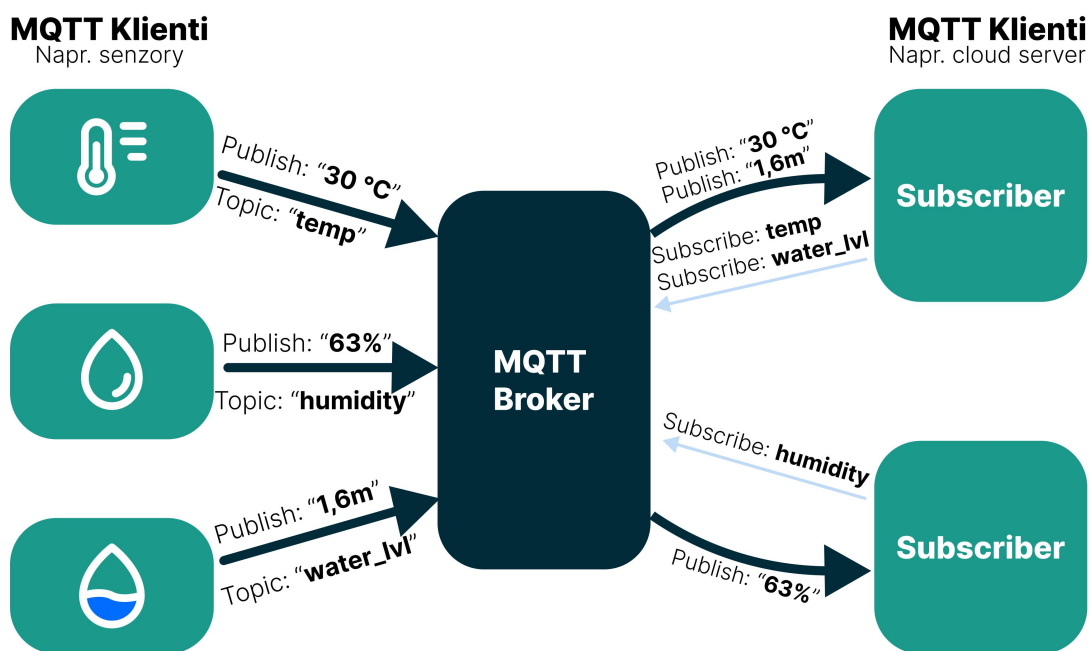
MQTT, alebo Message Queuing Telemetry Transport, je jednoduchý protokol výmeny dát. Bol navrhnutý tak, aby umožňoval zariadeniam s malými výpočtovými výkonmi a úložiskom, komunikáciu, v ktorej sa vyžaduje malá veľkosť kódu, a komunikáciu cez nízkopásmové a nespoľahlivé siete. Medzi týmito zariadeniami umožňuje komunikáciu, ktorá môže byť aj asynchrónna. [1, 6, 7]

Protokol využíva architektúru klient-server. Je založený na výmene správ medzi klientom (client), ktorý môže byť publisher (poskytovateľ správ) alebo subscriber (príjemca správ) a serverom, nazývaným broker (centrálny bod). [6, 8]

Publisher, čo môže byť napríklad teplotný senzor, sa pripojí k brokeru pomocou TCP/IP spojenia a odošle správu (publish), v ktorej nesie informácie a určité témy (topic). Subscriber sa môže prihlásiť k odberu jednej alebo viacerých tém (subscribe) a na základe toho mu budú zasielané informácie. Táto architektúra zaisťuje efektívnu a spoľahlivú komunikáciu medzi zariadeniami. [7]

Zariadenia na MQTT komunikáciu s brokerom používajú určité typy správ. Toto sú tie základné: [6]

- CONNECT - zavedenie spojenia s brokerom
- DISCONNECT - prerušenie spojenia s brokerom
- PUBLISH - publikovať správu témy
- SUBSCRIBE - prihlásenie sa k odberu témy
- UNSUBSCRIBE - odhlásenie od odberu témy



Obrázok 1.1 Ukážka jednoduchšej MQTT komunikácie. (zdroj: vlastný)

2 INTELIGENTNÉ DOMÁCNOSTI

Inteligentná domácnosť je miesto, ktoré poskytuje pohodlné životné podmienky, monitorovanie a automatizáciu domova. IoT pomáha dosiahnuť automatizácie v reálnom čase z akéhokoľvek miesta na svete. Aspekty inteligentnej domácnosti zahŕňajú sieťovú infraštruktúru, inteligentné ovládanie, sieť senzorov, inteligentné funkcie a odozvy. Inteligentné domy založené na IoT môžu pomôcť šetriť energiu a znižovať náklady.

Medzi niekoľko základných spôsobov využitia IoT v inteligentných domácnostiach patria inteligentné spínače, inteligentný merač spotreby energie, monitorovanie kvality ovzdušia, bezpečnosť domácnosti, ovládanie svetla a mnoho ďalšieho. [9]

2.1 Virtuálni asistenti

Virtuálny asistent predstavuje softvérové riešenie, prostredníctvom ktorého môže užívateľ na diaľku interaktívne a jednoducho vykonávať rôzne úlohy či využívať rozličné služby. Ako ovládanie inteligentných zariadení, plánovanie udalostí, prehrávanie hudby, získavanie informácií o počasi či správa e-mailov, na základe užívateľského vstupu. [10]

Virtuálni asistenti sú založení na technológiách, ako spracovanie prirodzeného jazyka, strojové učenie, rozpoznávanie a syntéza reči. Tieto technológie umožňujú virtuálnym asistentom porozumieť a odpovedať na užívateľské požiadavky, prispôbiť sa úlohe a zlepšiť svoje výkony na základe nových informácií získaných v procese. [10]

2.2 Rozdelenie domácní a virtuálnych asistentov

Na trhu existuje množstvo asistentov, každý z nich prináša jedinečné prednosti a prístupy k ovládaniu inteligentnej domácnosti, či už ide o open-source riešenia alebo komerčné virtuálne platformy. Medzi najznámejších virtuálnych asistentov patria Amazon Alexa, Google Assistant, Apple HomeKit, Home Assistant, OpenHAB a ďalší.

2.2.1 Home Assistant

Home Assistant je open-source platforma, ktorá umožňuje užívateľovi vytvoriť jednotný systém na ovládanie množstva inteligentných zariadení. Dokáže komunikovať s mnohými značkami, ktoré by za normálnych okolností navzájom nekomunikovali. Je neustále rozširovaná o nové funkcionality a integrácie vďaka širokej komunite nadšencov. [11]

2.2.2 Google Assistant

Google asistent je virtuálny asistent vyvinutý spoločnosťou Google, ktorý integruje hlasové ovládanie s prístupom k rozsiahlym informáciám a službám jeho ekosystému. Je k dispozícii na Android zariadeniach a je spätý s Google Nest zariadeniami, ktoré boli kedysi známe ako Google Home. Sú vybavené chromecast technológiou a umožňujú ovládanie inteligentných zariadení. Tým sa z Google Assistanta stáva centrálny bod inteligentnej domácnosti. [12]

2.2.3 Amazon Alexa

Amazon Alexa je virtuálny asistent vyvinutý spoločnosťou Amazon. Umožňuje používanie hlasového ovládania, vďaka čomu dokáže na pokyn prehrávať hudbu, poskytovať informácie, doručovať správy, informovať o počasí, ovládať inteligentnú domácnosť a mnoho ďalších funkcií. [13]

Jedným z najznámejších zariadení, ktoré využívajú Alexu, je Amazon Echo. Echo je séria inteligentných reproduktorov, ktoré umožňujú hlasové ovládanie rôznych zariadení inteligentnej domácnosti. Echo zariadenia sú vybavené mikrofónmi s dlhým dosahom, ktoré dokážu zachytiť hlas aj napriek šumu prostredia, čo umožňuje Alexe počuť príkazy aj v rušných podmienkach. [14, 15]

2.2.4 Apple HomeKit

Apple HomeKit je platforma, ktorá umožňuje prepojiť rôzne inteligentné zariadenia do jedného ovládacieho systému. Tento systém nie je samostatný produkt, ale prostredie, v ktorom môžu byť zariadenia ako, svetlá, zámky, termostaty či kamery, ovládané pomocou aplikácie Home a hlasového asistenta Siri. Platforma HomeKit dáva možnosť vytvárať scény a automatizácie, čo umožňuje napríklad nastaviť režim „dobrú noc“, kde sa viaceré zariadenia aktivujú súčasne.

Zariadenia sa pripájajú cez Bluetooth, Wi-Fi, Thread alebo pomocou hubov, ako sú Apple TV alebo HomePod. Jedným z hlavných aspektov HomeKitu je jeho bezpečnosť, keďže každý výrobca zariadení kompatibilných s HomeKitom musí splniť prísne bezpečnostné požiadavky. [16]

2.2.5 OpenHAB

OpenHAB je open-source platforma, ktorá umožňuje užívateľovi vytvoriť si jednotný systém podľa jeho potrieb. Táto platforma je udržiavaná a vyvíjaná veľkou komunitou. Je veľmi flexibilná a prispôsobiteľná, ale vyžaduje určité znalosti a čas na nastavenie a konfiguráciu systému. Taktiež je možné ju používať bez pripojenia na internet, čo zvyšuje bezpečnosť a súkromie užívateľa. [17]

2.3 Bezpečnosť a riziká inteligentných domácností

Napriek mnohým výhodám, ktoré inteligentné domácnosti prinášajú, predstavujú zároveň nové bezpečnostné výzvy a potenciálne riziká pre užívateľov. Tieto riziká môžu zahŕňať neoprávnený prístup k zariadeniam, zneužitie osobných údajov, útoky na sieťovú infraštruktúru a ďalšie. Bezpečnostné incidenty v oblasti inteligentných domácností môžu mať vážne následky, ako sú krádeže, narušenie súkromia a poškodenie majetku. [18, 19]

2.3.1 Bezpečnostné riziká

Medzi najčastejšie bezpečnostné riziká inteligentných domácností patria:

- Prednastavené heslá - mnoho IoT zariadení je dodávaných s prednastavenými heslami, ktoré užívatelia často nemenia. [18]
- Chýbajúce bezpečnostné štandardy - výrobcovia IoT zariadení často neimplementujú jednotné bezpečnostné protokoly, čo vedie k nejednotným a nedostatočným bezpečnostným opatreniam [18]
- Nezabezpečený firmware a softvér - mnohé zariadenia používajú zastaraný alebo nezabezpečený firmware a softvér, čo ich robí zraniteľnými voči známym útokom. [18]
- Fyzické bezpečnostné riziká - napadnuté zariadenia ovládajúceho fyzické systémy (napr. zámky, termostaty, kamery) môžu umožniť neoprávnený prístup do domácnosti, nastavenie nebezpečnej teploty atď. [18]

2.3.2 Odporúčania pre zabezpečenie inteligentnej domácnosti

Na minimalizáciu bezpečnostných rizík inteligentnej domácnosti možno prijať nasledujúce opatrenia:

- Používanie silných a unikátnych hesiel - nastavovanie silných a jedinečných hesiel, ktoré minimalizujú riziko neoprávneného prístupu, pri tvorbe účtov a registrácii zariadení. [19]
- Pravidelné aktualizácie firmware a softvéru - zabezpečenie pravidelných aktualizácií pre IoT zariadenia a aplikácie. Aktualizácie často obsahujú opravy známych bezpečnostných zraniteľností. [19]
- Zabezpečenie Wi-Fi siete - je potrebné používať silné heslo a moderné šifrovanie (WPA2 alebo WPA3), aby sa zabránilo neoprávnenému prístupu. [19]
- Segmentácia siete - je odporúčané vytvorenie samostatnej siete pre IoT zariadenia oddelenú od hlavnej siete s počítačmi a smartfónmi [19]
- Monitorovanie siete - pravidelné sledovanie aktivity v domácej sieti a všímanie si neobvyklého alebo podozrivého správania sa zariadení. Na tento účel je možné využiť nástroje na monitorovanie siete. [19]
- Zabezpečenie fyzického prístupu - dbanie na fyzickú bezpečnosť zariadení, najmä tých, ktoré sú prístupné zvonku. Zváženie použitia dodatočných bezpečnostných prvkov, ako sú zámky alebo bezpečnostné kamery. [19]

3 MIKROKONTROLÉRY, ICH SÚČASTI A PROGRAMOVANIE

Mikrokontroléry sa nachádzajú v riadiacich systémoch automobilových motorov, robotoch, lekárskech prístrojoch, automatoch, v domácych spotrebičoch a mnohých ďalších zariadeniach. Sú to jednoduché miniatúrne počítače určené na ovládanie malých špecifických funkcií väčšieho komponentu bez potreby komplexného užívateľského rozhrania. [20]

Kombináciou viacerých komponentov do jedného systému sú mikrokontroléry vhodné pre úkony vyžadujúce spracovanie signálu v reálnom čase, ako je riadenie motorov a serv a prepojenie s rôznymi typmi senzorov a komunikácie. [21]

3.1 Súčasti mikrokontroléra

Hlavné súčasti, ktoré tvoria mikrokontrolér, sú centrálna procesorová jednotka (CPU), pamäť a periférie.

CPU slúži ako hlavný komponent zodpovedný za vykonávanie pokynov a riadenie operácií. Vykonáva tiež operácie prenosu údajov, ktoré komunikujú s ostatnými komponentmi väčšieho vstavaného systému. [20, 21]

Pamäť ukladá údaje, ktoré procesor prijíma a používa pri pokynoch, na ktoré je naprogramovaný. Sú dva hlavné typy pamäte:[20, 21]

- RAM (Random Access Memory) - dočasné úložisko pre údaje počas vykonávania programu, nestála pamäť vyžadujúca napájanie [20]
- ROM (Read-Only Memory) - energeticky nezávislá pamäť obsahujúca dôležité informácie pre inicializáciu zariadenia [20]

Periférie zahŕňajú vstupno-výstupné rozhrania (I/O), časovače, počítadlá, prevodníky signálu (ADC, DAC) a komunikačné protokoly (UART, SPI, I₂C). Môžu tiež obsahovať LCD obrazovky, ethernetové porty a ďalšie rozhrania. [20, 21]

3.2 Delenie mikrokontrolérov

Mikrokontroléry je možné rozdeliť podľa rôznych kritérií, ako sú šírka slova, architektúra, inštrukčná sada a ďalšie. V nasledujúcej podkapitole sú rozobrané najdôležitejšie aspekty delenia mikrokontrolérov.

3.2.1 Delenie podľa šírky slova

Pod šírkou slova je myslené množstvo bitov, ktoré dokáže mikrokontrolér spracovať naraz. Mikrokontroléry sa najčastejšie delia na 8-bitové, 16-bitové a 32-bitové varianty.

8-bitové mikrokontroléry sú základné typy, ktoré sú lacné a energeticky efektívne, ale s obmedzenými možnosťami spracovania a pamäte. Typicky sa využívajú v jednoduchých zariadeniach ako hračky, domáce spotrebiče a senzory. Známe príklady zahŕňajú Intel 8051, Microchip PIC a Atmel AVR. [20, 22]

16-bitové mikrokontroléry spracovávajú väčší objem údajov, vyznačujú sa vyšším výkonom a rozšírenými možnosťami vstupno-výstupných rozhraní. Sú vhodné pre aplikácie ako riadenie motorov, zber dát a priemyselnú automatizáciu, avšak s vyššou spotrebou energie. Typické príklady sú PIC24, Intel 8096 a Renesas RX. [22, 23]

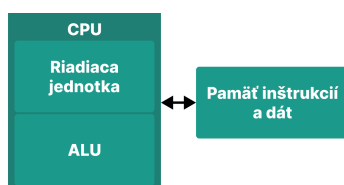
32-bitové mikrokontroléry ponúkajú vysoký výkon, väčšiu kapacitu pamäte a schopnosť riešiť zložité výpočtové úlohy za cenu vyššej spotreby energie a ceny. [22, 24]

3.2.2 Delenie podľa architektúry

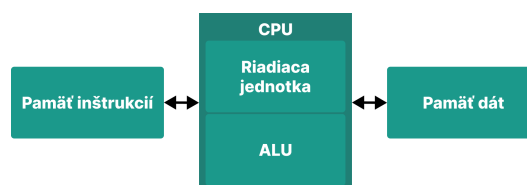
Mikrokontroléry sa podľa spôsobu pripojenia pamätí a procesora delia na dva hlavné typy: Von Neumannova a Harvardská architektúra.

Von Neumannova architektúra používa jednu spoločnú pamäť pre inštrukcie aj dáta, so zdieľanou zberniciou. Výhodou je jednoduchší návrh a nižšie náklady. Nevýhodou je tzv. „Von Neumannov bottleneck“ - procesor nemôže súčasne pristupovať k inštrukciám aj dátam, čo znižuje výkon. [25]

Harvardská architektúra má oddelenú pamäť a zbernice pre inštrukcie a dáta, umožňujúc paralelný prístup a vyššiu rýchlosť. Často sa používa vo vstavaných systémoch, kde je kľúčová efektívnosť. Nevýhodou je zložitejší návrh a vyššie náklady na implementáciu. [25]



Obrázok 3.1 Von Neumannova architektúra.(zdroj: vlastný)



Obrázok 3.2 Harvardská architektúra.(zdroj: vlastný)

3.2.3 Delenie podľa inštrukčnej sady

Inštrukčná sada je súbor inštrukcií, ktoré procesor dokáže vykonávať. Najznámejšie sú RISC a CISC architektúry.

RISC (Reduced Instruction Set Computer) používa menší počet jednoduchých inštrukcií, ktoré sú rýchlo vykonané v jednom takte. Vyznačuje sa pevnou veľkosťou inštrukcií a väčším počtom registrov. Výhodou je rýchlejšie vykonávanie a nižšia spotreba energie, nevýhodou väčšia spotreba pamäte pre zložité operácie. [26]

CISC (Complex Instruction Set Computer) obsahuje komplexné inštrukcie vykonávajúce viacero operácií naraz. Inštrukcie majú rôznu veľkosť a často vyžadujú viac taktov. Výhodou je efektívnejšie využitie pamäte, nevýhodou pomalšie vykonávanie a vyššia spotreba energie. [26]

3.3 Mikrokontroléry pre IoT

Mikrokontroléry pre IoT sú špeciálne navrhnuté na pripojenie a komunikáciu s internetom a inými zariadeniami. Tieto mikrokontroléry sú vybavené rôznymi komunikačnými protokolmi, ako sú napríklad Wi-Fi alebo Bluetooth, čo im umožňuje efektívne prenášať dáta. V tejto podkapitole sa zameriame na niektoré z najpopulárnejších mikrokontrolérov používaných v IoT.

3.3.1 Mikrokontrolér ESP8266

ESP8266 je 32-bitový RISC mikrokontrolér od spoločnosti Espressif Systems s integrovaným 2,4 GHz Wi-Fi modulom. Obsahuje procesor Tensilica Xtensa LX106 s taktovaním 80-160 MHz a je dostupný v rôznych variantoch, ako napríklad ESP-01, ESP-12, ESP-05 a ESP-201, ktoré sa líšia počtom GPIO pinov a veľkosťou pamäte. [7, 27]



Obrázok 3.3 Vývojová doska ESP8266 Lua NodeMcu V3 WIFI CH340. (zdroj: 28)

3.3.2 Mikrokontrolér ESP32

ESP32 je nástupcom ESP8266 a predstavuje výkonný 32-bitový mikrokontrolér, ktorý tvorí základ mnohých IoT projektov vrátane domácich asistentov. Je vybavený jedno- alebo dvoj-jadrovým mikroprocesorom Tensilica Xtensa LX6 s frekvenciou až 240 MHz, čo mu poskytuje podstatne väčší výpočtový výkon v porovnaní s jeho predchodcom. [7]

ESP32, na obrázku 3.4, je dostupný v rôznych variantoch, ktoré sa líšia počtom jadier, veľkosťou pamäte a perifériami. Štandardne disponuje 520 KB SRAM, 448 KB ROM a podporuje pripojenie externej flash pamäte až do veľkosti 16 MB. Z hľadiska konektivity integruje Wi-Fi (IEEE 802.11 b/g/n), Bluetooth (klasický aj BLE) a obsahuje 34 GPIO pinov. Podporuje široké spektrum komunikačných protokolov vrátane SPI, I²C, I²S, ADC, DAC, UART, CAN a Ethernet, čo z neho robí mimoriadne flexibilnú platformu pre široký rozsah IoT aplikácií. [7, 29]

Vďaka týmto vlastnostiam je ESP32 ideálnou platformou pre vývoj domácich asistentov, inteligentných domácností a IoT zariadení vyžadujúcich bezdrôtové pripojenie a spracovanie údajov zo senzorov v reálnom čase.



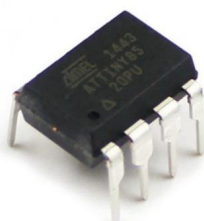
Obrázok 3.4 Vývojová doska ESP32 2. (zdroj: 30)

3.3.3 Mikrokontrolér ATTiny85

Mikrokontrolér ATTiny85 od spoločnosti Microchip Technology predstavuje kompaktný, ale výkonný 8-bitový mikrokontrolér s pokročilou architektúrou RISC. Vyznačuje sa mimoriadne nízkou spotrebou energie, čo ho predurčuje na použitie v zariadeniach napájaných batériami a energeticky úsporných IoT aplikáciách. Disponuje 120 výkonnými inštrukciami, pričom väčšina z nich sa vykonáva v jednom taktovacom cykle, čo zabezpečuje efektívne spracovanie údajov v reálnom čase. [31, 32]

Z hľadiska pamäte ponúka ATTiny85 8 kB programovateľnej flash pamäte s výdržou 10 000 cyklov zápisu/vymazania, 512 bajtov EEPROM s výdržou až 100 000 cyklov zápisu/vymazania a 512 bajtov internej SRAM pamäte. Tento mikrokontrolér obsahuje 6 programovateľných vstupno-výstupných pinov a podporuje režimy nízkej spotreby energie, interný kalibrovaný oscilátor a programovanie cez SPI rozhranie. [32]

Periférne funkcie zahŕňajú 8-bitový časovačpočítadlo, univerzálne sériové rozhranie a 10-bitový ADC prevodník so štyrmi jednosmernými kanálmi. Vďaka týmto vlastnostiam je ATTiny85 vhodný pre široké spektrum aplikácií, kde sú kľúčové faktory nízka spotreba energie, malé rozmery a dostatočný výpočtový výkon. [32]



Obrázok 3.5 ATTiny85 - 8-bitový mikrokontrolér (zdroj: 32)

3.4 Programovanie mikrokontroléra

Programovanie mikrokontrolérov je kľúčovou fázou vývoja vstavaných systémov. Voľba programovacieho jazyka významne ovplyvňuje efektivitu a spoľahlivosť výsledného systému.

3.4.1 Programovacie jazyky pre mikrokontroléry

Programovacie jazyky pre mikrokontroléry sa delia na nízkoúrovňové (strojový kód, jazyk symbolických adries) poskytujúce priamu kontrolu nad hardvérom a vysokoúrovňové (Ada, MicroPython, C, C++), ktoré ponúkajú vyššiu mieru abstrakcie a jednoduchšie programovanie. [33]

V oblasti mikrokontrolérov sa najčastejšie využíva jazyk C++, ktorý kombinuje efektívnu manipuláciu s hardvérovými zdrojmi a pamäťou s možnosťami objektovo orientovaného programovania. Tieto vlastnosti umožňujú modulárny a udržateľný vývoj softvéru aj pre komplexnejšie systémy. [34]

3.5 Vývojové prostredia

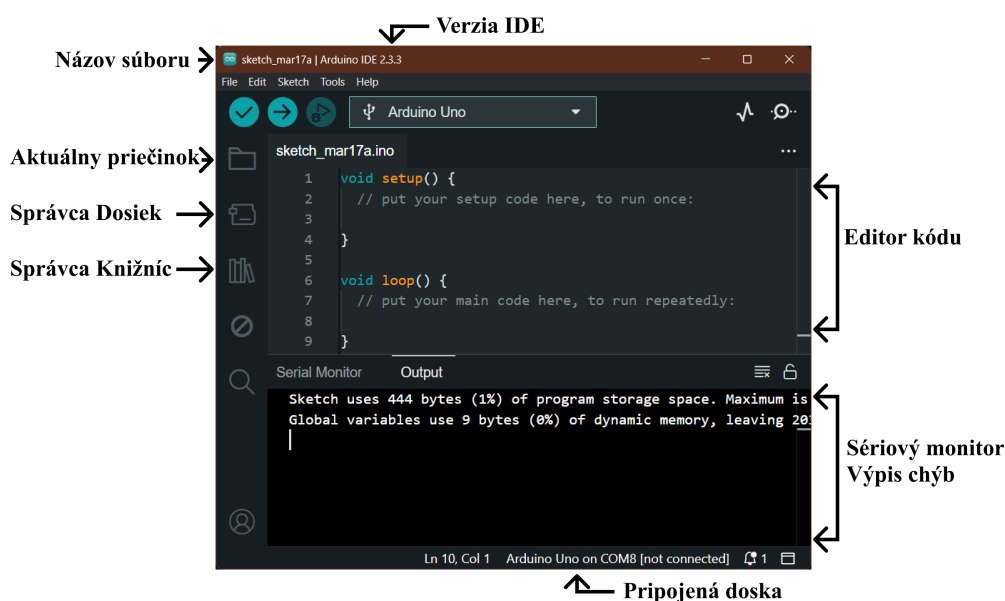
Pod pojmom vývojového prostredia (IDE) sa rozumie softvér, ktorý obsahuje integrované nástroje na písanie, kompilovanie, testovanie a ladenie kódu. Tieto prostredia ponúkajú množstvo funkcií, ako je zvýraznenie syntaxe, automatické dopĺňanie kódu a podpora pre rôzne programovacie jazyky. [35]

3.5.1 Vývojové prostredie Arduino IDE

Arduino IDE od spoločnosti Arduino je navrhnuté tak, aby bolo prístupné pre začiatočníkov. Ponúka množstvo funkcií, ako sú automatické dopĺňovanie kódu, zvýraznenie syntaxe a integrovaný sériový monitor. Umožňuje používateľom písať, kompilovať a nahrávať kód do dosiek Arduino bez obavy o technické detaily. [36]

Je to multiplatformové prostredie, ktoré je dostupné pre Windows, macOS a Linux, ale aj webová aplikácia. Obsahuje množstvo knižníc, ktoré uľahčujú prácu s rôznymi senzormi, modulmi a inými komponentmi. Umožňuje taktiež kompatibilitu s rôznymi doskami a mikrokontrolérmi od rôznych výrobcov za pomoci knižníc. [36, 37]

Poskytuje podporu pre programovanie v C a C++, s možnosťou využitia iných jazykov prostredníctvom špecializovaných knižníc. Umožňuje konfiguráciu pokročilých parametrov mikrokontroléra, no pre komplexnejšie projekty chýbajú nástroje ako verziovanie kódu či jednotkové testovanie. [37]



Obrázok 3.6 Ukážka Arduino IDE. (zdroj: vlastný)

3.5.2 Vývojové prostredie Espressif-IDE

Espressif-IDE je vývojové prostredie založené na Eclipse IDE, určené na tvorbu IoT aplikácií pre mikrokontroléry ESP32. Umožňuje vývoj v jazykoch C a C++ s podporou automatického dopĺňania kódu, zvýraznenia syntaxe a integrovaného debuggeru. Obsahuje konfiguračné nástroje vrátane editora nastavení SDK a podpory pre build systém CMake. [38]

IDE poskytuje aj praktické funkcie pre prácu s hardvérom ako flashovanie cez UART a JTAG, sériový monitor a nástroje na analýzu veľkosti aplikácie a sledovanie pamäte. Tieto vlastnosti umožňujú efektívny vývoj a ladenie IoT aplikácií. [38, 39]

3.5.3 Vývojové prostredie Visual Studio Code spolu s PlatformIO

Visual Studio Code od Microsoftu je populárny textový editor s funkciami ako zvýraznenie syntaxe, automatické dopĺňanie kódu, reštrukturalizáciu kódu a interaktívne ladenie. Umožňuje jednoduchú navigáciu v kóde a prispôsobenie rozhrania. Jeho funkcionality možno rozšíriť pomocou rôznych rozšírení, vrátane PlatformIO. [40]

PlatformIO ako rozšírenie pre VS Code je zamerané na vývoj vstavaných systémov a IoT aplikácií. Podporuje množstvo hardvérových platforiem a mikrokontrolérov, čo umožňuje vývoj pre rôzne zariadenia v jednom prostredí. Ponúka statickú analýzu kódu, integrovaný debugger a nástroje na testovanie. [41, 42]

3.6 Vývojové platformy

Vývojové platformy integrujú mikrokontroléry s napájacími obvodmi, komunikačnými rozhraniami a konektormi pre periférie. Odstraňujú potrebu navrhovať vlastný hardvér, čím umožňujú vývojárom sústrediť sa na programovanie a testovanie funkcionality.[43]

V IoT a domácich asistentoch zohrávajú kľúčovú úlohu vďaka bezdrôtovým modulom, podpore senzorov a dostatočnému výkonu pri nízkej spotrebe energie. Medzi najpopulárnejšie patria dosky ESP32, Arduino a Raspberry Pi.

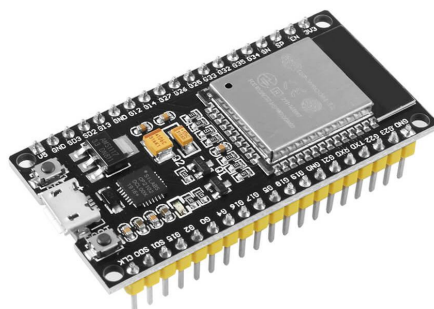
3.6.1 Vývojové dosky Espressif

Vývojové dosky Espressif predstavujú univerzálne platformy určené na rýchly vývoj a testovanie aplikácií s mikrokontrolermi ESP32. Tieto dosky obsahujú všetky potrebné obvody pre napájanie, programovanie a jednoduché pripojenie periférií, pričom väčšina GPIO pinov je

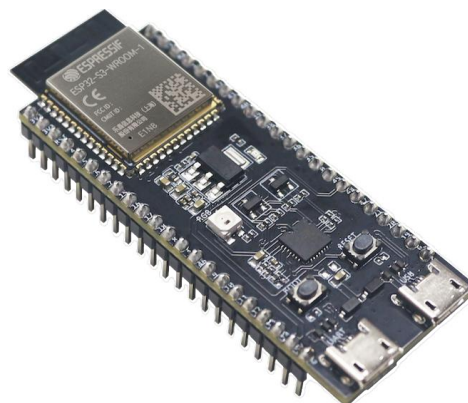
vyvedená na pinové lišty pre ľahké použitie na breadboarde alebo s prepojovacími vodičmi. [29, 44]

Jedným z najrozšírenejších modelov je ESP32 DevKit v1, ktorý je založený na module ESP-WROOM-32. Napájanie je možné cez microUSB konektor alebo priamo cez pin VIN, pričom napájací zdroj je volený automaticky. DevKit v1 je vybavený USB-UART prevodníkom, tlačidlami BOOT a RESET a poskytuje 25 digitálnych I/O pinov, 6 analógových vstupov, 2 DAC výstupy a podporu pre Wi-Fi aj Bluetooth. Vďaka kompaktným rozmerom a univerzálnosti je vhodný pre široké spektrum IoT aplikácií. [45]

Ďalším príkladom je ESP32-S3-DevKitC-1, ktorý poskytuje podporu Wi-Fi a Bluetooth Low Energy. Táto doska poskytuje akceleráciu pre neurónové siete a spracovanie signálov, pričom všetky dostupné GPIO piny sú vyvedené na pinové lišty. Okrem štandardného microUSB portu pre napájanie a programovanie je k dispozícii aj USB OTG rozhranie, RGB LED, LED indikátor napájania, tlačidlá BOOT a RESET. [46]



Obrázok 3.7 Vývojová doska ESP32 DevKit v1. (zdroj: 45)

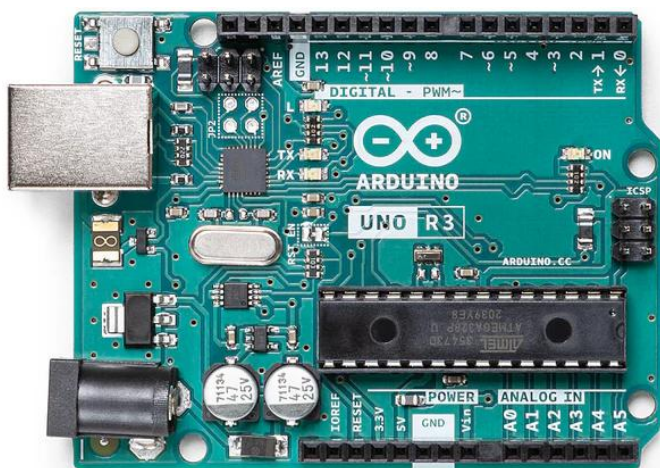


Obrázok 3.8 Vývojová doska ESP32-S3-DevKitC-1.(zdroj: 46)

3.6.2 Vývojové dosky Arduino

Arduino je open-source platforma založená v Taliansku v roku 2005. Vďaka jednoduchému použitiu a širokej komunite podpory je populárnou voľbou pre vývoj IoT, robotiky a ďalších vstavaných projektov. [47]

Najpopulárnejšia doska Arduino Uno je založená na mikrokontroléri ATmega328P od spoločnosti Microchip Technology a poskytuje 14 digitálnych I/O pinov, 6 analógových vstupov a taktovanie 16 MHz. Jej jednoduché USB pripojenie a kompaktný dizajn ju robia ideálnou platformou pre začiatočníkov aj pokročilých vývojárov. [48]



Obrázok 3.9 Arduino Uno. (zdroj: 48)

3.6.3 Jednodoskové počítače Raspberry Pi

Raspberry Pi je jednodoskový počítač vyvinutý britskou nadáciou Raspberry Pi ako cenovo dostupná platforma pre vzdelávanie v informatike, ktorý sa od roku 2012 stal univerzálnym nástrojom pre široké spektrum projektov. [49]

Najrozšírenejší Raspberry Pi 4 model B disponuje 4-jadrovým procesorom ARM Cortex-A72, až 8 GB RAM a integrovaným Wi-Fi a Bluetooth, čo umožňuje jeho použitie ako plnohodnotný počítač alebo riadiaca jednotka pre systémy inteligentnej domácnosti. [50]

Pre vstavaný vývoj slúži model Raspberry Pi Pico s mikrokontrolérom RP2040 (2-jadrový ARM Cortex-M0+) a 26 GPIO pinmi, ktorý predstavuje cenovo dostupnú alternatívu pre menšie IoT projekty. [51]



Obrázok 3.10 Vývojová doska Raspberry Pi 4 model B. (zdroj: 52)



Obrázok 3.11 Vývojová doska Raspberry Pi Pico. (zdroj: 51)

4 MODULY A SENZORY PRE DOMÁCICH ASISTENTOV

Rádiové moduly a senzory sú základnými komponentmi pre vytváranie inteligentných domácych asistentov. Tieto zariadenia umožňujú komunikáciu medzi rôznymi komponentmi systému a poskytujú údaje z okolitého prostredia. V tejto kapitole sa zameriame na rôzne typy rádiových modulov a senzorov, ktoré sú bežne používané v inteligentných domácnostiach.

4.1 Rádiové moduly

Rádiové moduly sú zvyčajne malé elektronické zariadenia využívané na bezdrôtovú komunikáciu medzi dvoma alebo viacerými zariadeniami. Tieto moduly sú spravidla zložené z vysielачa a prijímača, ktoré sú schopné komunikovať na určitej frekvencii. [53, 54]

4.1.1 Typy rádiových modulov

Rádiové moduly zvyčajne obsahujú PCB, vysielací/prijímací obvod, anténu a sériové rozhranie na komunikáciu s hostiteľským procesorom. Rozdeľujú sa na: [53]

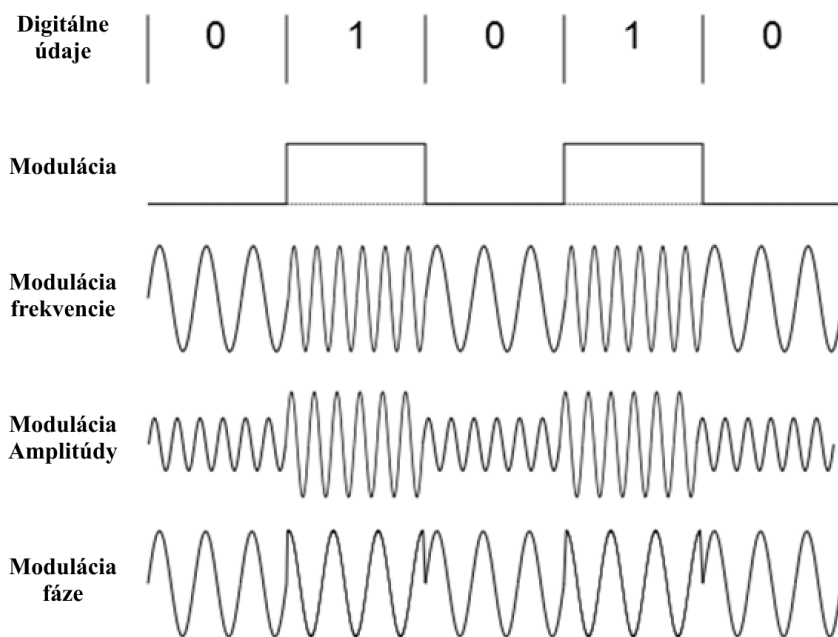
- Vysielací - prenáša a moduluje rádiovú vlnu na prenos údajov. [53]
- Prijímací - prijíma a demoduluje rádiové signály. [53]
- Transceiver - modul kombinuje vysielачa a prijímača. Obvod je zvyčajne navrhnutý pre polovičnú duplexnú prevádzku, aj keď sú k dispozícii plne duplexné moduly. [53]
- Wifi RF modul - modul, ktorý umožňuje bezdrôtovú komunikáciu na frekvencii 2.4 GHz alebo 5 GHz. Tieto moduly sú bežne používané pre zariadenia, ktoré potrebujú pripojenie k internetu alebo komunikáciu s inými zariadeniami v blízkom okolí. [53]
- Bluetooth RF modul - modul, ktorý umožňuje bezdrôtovú komunikáciu v krátkych vzdialenostiach. Bluetooth vysielачa na frekvencii 2.4 GHz a je bežne používaný v zariadeniach, ktoré potrebujú komunikovať s inými zariadeniami v blízkom okolí, ako sú napríklad smartfóny a periférne zariadenia. [53]
- LoRa RF modul - modul, ktorý umožňuje komunikáciu na veľké vzdialenosti s nízkou spotrebou energie. Tieto moduly sú bežne používané pre IoT zariadenia, ktoré vyžadujú komunikáciu na veľké vzdialenosti s nízkou spotrebou energie. [53]

4.1.2 Modulácia signálu

V elektronike modulácia znamená proces premeny informácie na rádiové vlny pridaním informácie k elektronickému alebo optickému nosnému signálu. [55]

Najčastejšie spôsoby modulácie sú: [56, 57]

- ASK - modulácia, kde sú rôzne úrovne amplitúdy priradené rôznym binárnym hodnotám. Amplitúda nosnej vlny je modifikovaná na základe hodnoty digitálneho signálu. [56]
- FSK - modulácia, kde je frekvencia nosnej vlny prepínaná medzi dvoma vopred určenými frekvenciami, pričom jedna reprezentuje binárnu 0 a druhá binárnu 1. [56]
- PSK - modulácia, kde je fáza nosnej vlny upravovaná v špecifických intervaloch na a to základe digitálneho signálu. [56]



Obrázok 4.1 Rozdelenie modulácie signálu. (preložené zo zdroja: 57)

4.1.3 Využitie rádiových modulov

Rádiové moduly zabezpečujú bezdrôtovú komunikáciu medzi elektronickými zariadeniami. Uplatnenie nachádzajú primárne v produktoch so strednou a nízkou sériovou výrobou.

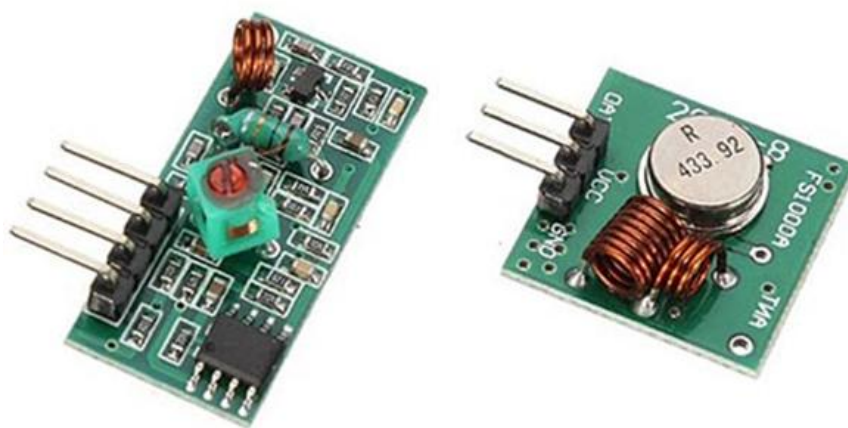
V spotrebiteľskej elektronike sa využívajú najmä v diaľkových ovládačoch, bezdrôtových klávesniciach a myšiach. V inteligentných domácnostiach tvoria základ IoT systémov

na ovládanie osvetlenia, vykurovania a zabezpečenia. Významné uplatnenie majú aj v priemyselnej automatizácii, zdravotníckych zariadeniach a dopravných systémoch. [53]

4.1.4 433 MHz rádiový modul

433 MHz rádiový modul je systém skladajúci sa z dvoch modulov, vysielača a prijímača. Vysielačom je generovaný a vysielaný modulovaný RF signál. Prijímačom je tento signál zachytávaný, zosilňovaný a demodulovaný. Tieto moduly sú jednoduché na použitie a vhodné pre jednosmernú komunikáciu. Vysielačom môžu byť len vysielané informácie a prijímačom len prijímané. [58]

Využíva ASK moduláciu, čo umožňuje jednoduché použitie s mikrokontrolérmi. Prenosová rýchlosť sa pohybuje od 1 do 10 Kbps [58, 9]



Obrázok 4.2 433 MHz prijímač (vľavo) a vysielač (vpravo). (zdroj: 59)

4.2 Senzory pre monitorovanie prostredia

Senzory sú základom inteligentných domácností. Ich hlavnou úlohou je snímať fyzikálne alebo chemické veličiny v okolí a prevádzať ich na elektrický signál. S prepojením s mikrokontrolérom je možné tieto hodnoty prenášať do počítača alebo iného zariadenia. [60]

Sú dva hlavné typy fungovania senzorov: [60]

- Pasívne senzory - detekujú zmeny v prostredí bez externého zdroja energie, napr. teplotné senzory
- Aktívne senzory - vyžadujú externý zdroj energie na detekciu zmien v prostredí, napr. ultrazvukové senzory

V moderných IoT aplikáciách sú senzory často prepojené s cloudovými službami. Dáta ďalej analyzujú a využívajú na automatizáciu alebo optimalizáciu procesov v domácnosti. [60]

4.2.1 Teplotné a vlhkostné senzory

Teplotné a vlhkostné senzory merajú a konvertujú teplotu a vlhkosť vzduchu na elektrický signál. [61]

Príkladom teplotného a vlhkostného senzora je DHT11, ktorý predstavuje cenovo dostupný digitálny senzor. Je tvorený z kapacitného prvku na meranie vlhkosti a termistora s negatívnym teplotným koeficientom na meranie teploty. Pri zmene vlhkosti dochádza k zmene kapacity medzi elektródami, čo integrovaný obvod spracuje a prevedie do digitálnej formy. DHT11 dokáže merať teplotu s presnosťou ± 2 °C a vlhkosť s presnosťou ± 5 %. [62]

Ďalším príkladom je senzor HTU2X, ktorý predstavuje presný, digitálny senzor na meranie relatívnej vlhkosti a teploty vzduchu. Na rozdiel od DHT11 využíva na komunikáciu s mikrokontrolérom dvojvodičové I²C rozhranie, čo zjednodušuje jeho implementáciu do komplexnejších systémov. HTU2X sa vyznačuje vyššou presnosťou, kde pri meraní vlhkosti dosahuje typickú presnosť ± 2 % a pri meraní teploty až ± 0.3 °C. [63]

4.2.2 Ďalšie typy senzorov pre inteligentné domácnosti

Okrem teplotných a vlhkostných senzorov existuje množstvo ďalších senzorov, ktoré sú kľúčové pre zvýšenie komfortu, bezpečnosti a efektivity v inteligentnej domácnosti. Medzi najpraktickejšie patria: [64]

- Pohybové senzory - najčastejšie PIR senzory reagujúce na zmeny tepelného vyžarovania, používané pre automatizáciu osvetlenia a bezpečnostné systémy [64]
- Senzory dymu a CO - detegujú prítomnosť dymu alebo nebezpečného, bezfarebného plynu CO [64]
- Dverové a okenné senzory - magnetické senzory pre detekciu otvorenia/zatvorenia, používané pre zabezpečenie a automatizáciu [64]

5 POUŽÍVATEL'SKÉ ROZHRAŇIE A POUŽÍVATEL'SKÁ SKÚSE- NOSŤ

Táto kapitola sa zaoberá základnými princípmi používateľského rozhrania (UI) a používateľskej skúsenosti (UX). V tejto kapitole sú vysvetlené pojmy responzívny dizajn, prístupnosť a použiteľnosť, ich význam a potreba v modernom webovom dizajne. Ďalej sú predstavené moderné technológie používateľského rozhrania, rozdelenie rozhrania, architektúra serveru a API. Na koniec je zdôraznená podstata autentifikácie a bezpečnosti.

5.1 Princípy UI a UX dizajnu

UI a UX je možné tiež brať ako súhrn pravidiel, ktoré pomáhajú vytvárať používateľsky prívetivý dizajn. UI dizajn sa zaoberá vzhľadom a štruktúrou webových stránok alebo produktu. Jeho cieľom je zaujať a udržať pozornosť používateľa, aby sa mohol ľahko navigovať a bezproblémovo vykonať požadovanú činnosť. UX dizajn sa zameriava na vytvorenie lepšej skúsenosti pre používateľa. Je zameraný na používateľa, jeho správanie a očakávania od produktu alebo služby. [65]

5.1.1 Responzívny dizajn

Responzívny dizajn je kľúčovým prvkom moderného webového dizajnu. Zabezpečuje to, že webová stránka je optimalizovaná pre zobrazenie na rôznych zariadeniach, ako sú počítače, tablety a mobilné telefóny. Responzívny dizajn zabezpečuje tiež to, že sa obsah stránky prispôsobí veľkosti obrazovky zariadenia, na ktorom je zobrazený. [66]

Bežným prístupom k responzívnemu dizajnu je takzvaný „Mobile First“ prístup. Tento prístup sa zameriava na návrh a vývoj pre mobilné zariadenia ako prvý krok. Následne sa dizajn postupne rozširuje na väčšie obrazovky.

5.1.2 Prístupnosť a použiteľnosť používateľského rozhrania

Prístupnosť a použiteľnosť sú kľúčové aspekty používateľského rozhrania. WCAG predstavuje súbor pravidiel ako vytvoriť webové stránky a mobilné aplikácie tak, aby boli prístupné a použiteľné pre čo najširší okruh ľudí vrátane seniorov, zrakovo či sluchovo znevýhodnených osôb a ľudí s obmedzeniami pohybu. [67]

Z legislatívneho hľadiska je v Českej republike prístupnosť definovaná zákonom č. 99/

2019 Sb., o prístupnosti internetových stránok a mobilných aplikácií, ktorý implementuje európsku smernicu 2016/2102. V roku 2023 bol prijatý aj zákon č. 424/2023 Sb., implementujúci European Accessibility Act, ktorý sa vzťahuje na širšiu oblasť výrobkov a služieb. [68]

Pri implementácii používateľského rozhrania je potrebné dodržiavať štyri základné princípy WCAG: [68]

- Vnímateľnosť obsah a komponenty používateľského rozhrania musia byť implementované tak, aby ich používatelia mohli vnímať, napríklad možnosti zväčšenia textu, popisky a alternatívne texty pre obrázky [69, 70]
- Ovládateľnosť používateľ musí byť schopný ovládať všetky funkcie a komponenty používateľského rozhrania bez ohľadu na to, aké zariadenie používa [71]
- Zrozumiteľnosť informácie a ovládanie používateľského rozhrania musia byť zrozumiteľné, intuitívne a nápomocné [72]
- Stabilita používateľského rozhrania musí byť stabilné a spoľahlivo interpretované množstvom zariadení [73]

Praktické príklady zahŕňajú použitie väčšieho písma a výrazných kontrastov pre zobrazenie údajov, intuitívne usporiadanie ovládacích prvkov, alternatívne možnosti ovládania a responzívny dizajn pre rôzne zariadenia. [67]

5.2 Moderné technológie používateľského rozhrania

Používateľské rozhranie, taktiež nazývané ako Frontend, je časť webovej aplikácie, ktorá sa zaoberá vizuálnym zobrazením a interakciou s používateľom. Moderné technológie umožňujú vytvárať interaktívne a dynamické webové stránky, ktoré sú rýchle a efektívne. Medzi najpopulárnejšie technológie používateľského rozhrania patria React, Vue.js a Angular. [74]

5.2.1 Knižnica React

React je flexibilná a deklaratívna open-source JavaScript knižnica na tvorbu používateľských rozhraní. Stala sa jednou z najpopulárnejších knižníc na tvorbu používateľského rozhrania vďaka jednoduchosti, ktorú poskytuje pri vytváraní webových aplikácií. [74, 75]

Knižnica React je založená na komponentoch, ktoré slúžia ako stavebné bloky aplikácie. Vďaka týmto komponentom zjednodušuje tvorbu jednostránkových aplikácií (SPA), čo sú

webové aplikácie, ktoré majú iba jeden HTML súbor na vykreslenie celého kódu, so zameraním na efektívnosť a udržateľnosť. Tieto znovupoužiteľné komponenty v JavaScripte znižujú čas vývoja a komplexnosť. Okrem základného vytvárania používateľského rozhrania ponúka aj ďalšie technológie, ako Flux a React Native. React Native umožňuje vytvárať webové aj mobilné aplikácie pomocou rovnakého zdrojového kódu. [76, 75]

Je možné taktiež využiť TypeScript, ktorý je nadstavbou JavaScriptu. Pridáva statické typovanie, čo znamená, že je možné definovať typy premenných a funkcií. Tým sa zvyšuje prehľadnosť, čitateľnosť a bezpečnosť kódu. [77]

5.2.2 Iné populárne technológie používateľského rozhrania

Okrem Reactu patria medzi ďalšie významné technológie aj Angular a Vue.js. Angular, vyvíjaný spoločnosťou Google, je komplexný framework napísaný v TypeScript, využíva MVC architektúru a obojsmerné dátové viazanie na efektívne budovanie jednostránkových aplikácií. [74, 78]

Vue.js, predstavený v roku 2014, získal popularitu vďaka svojej jednoduchosti a nízkej krivke učenia. Tento progresívny framework umožňuje vytvárať responzívne UI komponenty a SPA aplikácie s dobrou dokumentáciou, čo ho robí vhodným pre začiatočníkov aj pokročilých vývojárov. [76, 79]

5.3 Serverová časť a API

Serverová časť, taktiež nazývaná ako backend, je časť webovej aplikácie, ktorá sa zaoberá spracovaním dát a logikou aplikácie. Zabezpečuje komunikáciu s databázou, spracovanie dát a poskytuje ich používateľskému rozhraniu.

5.3.1 Express.js a REST API

Express.js je minimalistický a flexibilný Node.js webový framework, ktorý poskytuje robustné nástroje pre vývoj webových a mobilných aplikácií. Jeho jednoduchá architektúra umožňuje rýchly vývoj serverových služieb a API. [80]

REST API je typ webového API, ktorý využíva HTTP požiadavky na interakciu s webovými zdrojmi. Je založený na princípoch REST, čo je softvérový architektonický štýl definujúci súbor obmedzení pre tvorbu škálovateľných a distribuovaných systémov. [81]

REST API typicky sprístupňuje zdroje ako URL adresy a podporuje sadu operácií (tzv. CRUD operácie) na interakciu s týmito zdrojmi: [81]

- Create (POST) - vytvorenie nového zdroja
- Read (GET) - získanie existujúceho zdroja
- Update (PUT/PATCH) - aktualizácia existujúceho zdroja
- Delete (DELETE) - odstránenie existujúceho zdroja

Express.js zjednodušuje vytváranie REST API pomocou middleware funkcií, ktoré spracovávajú požiadavky a odpovede. Pre IoT aplikácie a domáce asistenty predstavuje ideálnu voľbu vďaka svojej nenáročnosti na systémové zdroje a možnosti škálovania. [80]

5.4 Autentifikácia a bezpečnosť

V kontexte webových aplikácií a IoT systémov hrá autentifikácia a bezpečnosť dôležitú úlohu. Pri vývoji domáceho asistenta je potrebné zabezpečiť, aby k dátam a ovládaniu zariadení mali prístup len oprávnení používatelia.

5.4.1 JWT autentifikácia

JWT predstavuje otvorený štandard pre bezpečný prenos informácií medzi stranami ako kompaktný a samostatný JSON objekt. JWT sú digitálne podpísané, čo zaručuje integritu prenášaných údajov a možnosť overiť ich pôvod. [82]

Proces autentifikácie pomocou JWT je takýto: používateľ sa prihlási pomocou svojich prihlasovacích údajov, server overí tieto údaje a vygeneruje JWT token, ktorý následne pošle klientovi. Klient uloží token a pri každej ďalšej požiadavke na chránený zdroj priloží tento token v hlavičke. Server overí platnosť tokenu a na základe v ňom obsiahnutých údajov rozhodne o prístupe. [82]

II PRAKTICKÁ ČASŤ

6 ANALÝZA A NÁVRH SYSTÉMU

Domáci asistent predstavený v tejto práci slúži ako systém pre zber, spracovanie a vizualizáciu údajov zo senzorov v domácnosti. Systém poskytuje aj spätnú komunikáciu umožňujúcu spínanie svetla.

Táto kapitola sa zaoberá analýzou a návrhom systému, pričom sa zameriava na požiadavky na systém, celkovú architektúru a návrh hardvérovej, softvérovej a databázovej časti.

6.1 Požiadavky na systém

6.1.1 Funkčné požiadavky

Funkčné požiadavky popisujú, čo by mal systém robiť a aké funkcie by mal poskytovať. Tieto požiadavky sú zamerané na správanie systému a interakciu s používateľom.

Tabulka 6.1 Funkčné požiadavky systému

	Funkčné požiadavky
01	Zber dát zo senzorov
02	Odosielanie dát na server pomocou MQTT
03	Monitorovanie teploty a vlhkosti
04	Základné administrátorské rozhranie
05	Vizualizácia aktuálnych dát
06	Zabezpečenie MQTT komunikácie
07	Vizualizácia zaznamenaných dát
08	Možnosť ovládania svetiel
09	Možnosť registrácie a prihlásenia

6.1.2 Požiadavky zamerané na vlastnosti systému

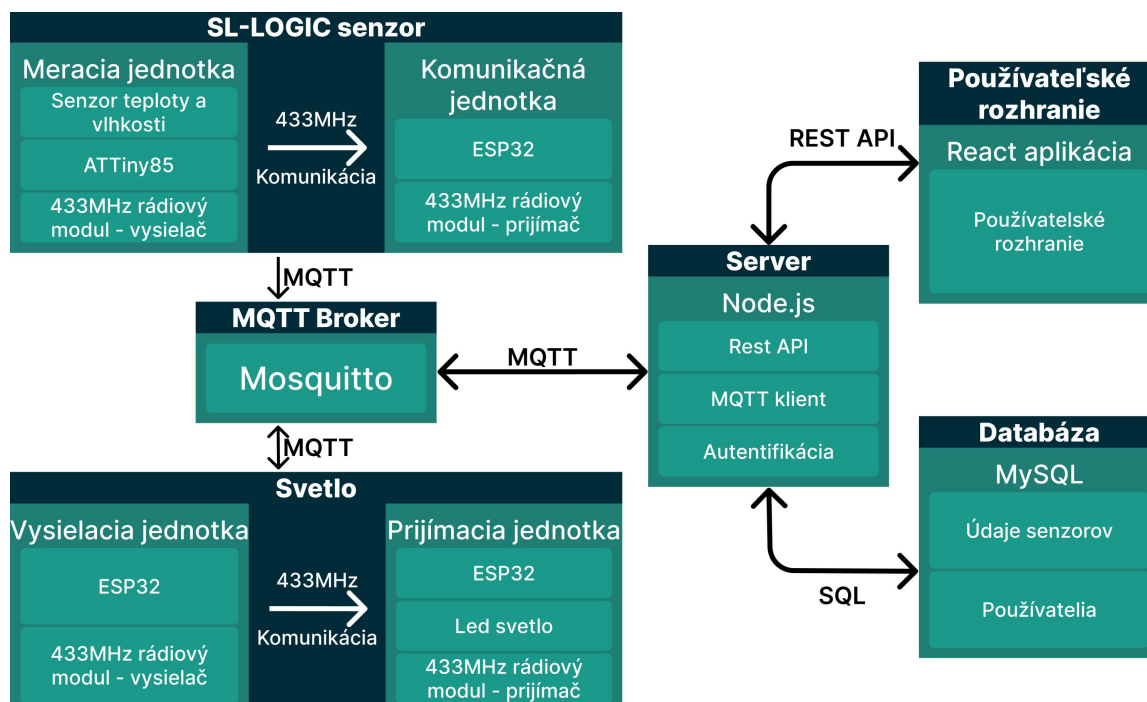
Požiadavky zamerané na vlastnosti systému, taktiež známe ako nefunkčné požiadavky, sú dôležité pre celkovú kvalitu systému a jeho schopnosť splniť očakávania používateľov.

Tabulka 6.2 Požiadavky zamerané na vlastnosti systému

	Nefunkčné požiadavky
01	Používateľské rozhranie zabezpečené autentifikáciou
02	Jednoduché, intuitívne a responzívne používateľské rozhranie
03	Bezpečná komunikácia medzi serverom a klientom
04	Schopnosť obsluhovať viacero senzorov

6.2 Celková architektúra systému

Systém je navrhnutý ako viacvrstvová architektúra s hardvérovou a softvérovou časťou, ktoré spolu komunikujú prostredníctvom štandardizovaných protokolov. Obrázok 6.1 znázorňuje celkovú architektúru systému s vyznačenými komponentmi a komunikačnými protokolmi.



Obrázok 6.1 Celková architektúra systému. (zdroj: vlastný)

6.2.1 Hardvérová vrstva

Hardvérová vrstva pozostáva z nasledujúcich komponentov:

- **SL-LOGIC senzor** - senzorová jednotka poskytnutá firmou SL-LOGIC, ktorá obsahuje senzor HTU2X na meranie teploty a vlhkosti a ATTiny85 mikrokontrolér. Tento mikrokontrolér spracováva údaje zo senzora a odosiela ich pomocou rádiového modulu na frekvencii 433 MHz do komunikačnej jednotky.
- **Svetlo** - taktiež sa skladá z dvoch častí: vysielacia jednotka, ktorá pomocou ESP32 spracováva príkazy na zapnutie alebo vypnutie svetla a následne ich cez 433 MHz rádiový modul odošle prijímacej jednotke, ktorý ovláda svetlo.

6.2.2 Komunikačná vrstva

Komunikačná vrstva zabezpečuje prenos dát medzi komponentami:

- Rádiová komunikácia na frekvencii 433 MHz - slúži na prenos údajov medzi vysielacím a prijímacím modulom.
- MQTT protokol - slúži na prenos správ medzi ESP32 a serverom.
- REST API - umožňuje komunikáciu medzi používateľským rozhraním a serverom.
- SQL - používa sa na ukladanie a spracovanie údajov v databáze.

6.2.3 Softvérová vrstva

Softvérová vrstva má tieto časti:

- Server - implementovaný v Node.js s využitím Express frameworku. Server prijíma údaje z MQTT brokera, spracováva ich a ukladá do databázy pomocou SQL procedúr.
- Databáza - slúži na ukladanie údajov zo senzorov, informácií o používateľoch a konfigurácie systému.
- Používateľské rozhranie - webové rozhranie implementované pomocou React frameworku, ktoré umožňuje používateľovi monitorovať hodnoty zo senzorov, prezerat' historické údaje a ovládať svetlá.
- MQTT broker - zabezpečuje komunikáciu medzi hardvérovými komponentmi a serverom.

6.2.4 Tok dát v systéme

Systém spracováva dva hlavné dátové toky:

- Monitorovanie - údaje zo senzorov sú odosielané prostredníctvom 433 MHz rádiového modulu do ESP32, následne cez MQTT broker do servera, kde sú spracované a uložené v databáze. Používateľ prístupuje k týmto údajom cez webové rozhranie, ktoré komunikuje so serverom pomocou REST API.
- Ovládanie - príkazy zadané používateľom vo webovom rozhraní sú odoslané na server, ktorý ich publikuje na príslušnú MQTT tému. ESP32 prijíma tieto príkazy a vykoná požadovanú akciu.

6.3 Návrh hardvérovej časti

Návrh hardvérovej časti systému vychádza z požiadaviek na bezdrôtovú komunikáciu a spoľahlivé meranie environmentálnych parametrov. Táto časť poskytuje prehľad hlavných hardvérových komponentov a ich vzájomných vzťahov, zatiaľ čo detailná implementácia je popísaná v kapitole 7.

6.3.1 Kľúčové komponenty

Pri návrhu hardvérovej časti systému boli vybrané nasledujúce kľúčové komponenty s ohľadom na ich vlastnosti, dostupnosť a vhodnosť:

Ako centrálna výpočtová jednotka bol zvolený mikrokontrolér ESP32 pre jeho integrované WiFi pripojenie, dostatočný výpočtový výkon a širokú podporu vývojových nástrojov.

Pre bezdrôtovú komunikáciu medzi jednotlivými zariadeniami boli vybrané rádiové moduly pracujúce na frekvencii 433 MHz. Tieto moduly boli zvolené pre ich spoľahlivosť, nízku energetickú náročnosť a prevádzku v bezlicenčnom frekvenčnom pásme.

Pre meranie hodnôt z okolia bol implementovaný senzorový modul od firmy SL-LOGIC, ktorý integruje senzor HTU2X pre presné meranie teploty a vlhkosti a mikrokontrolér AT-Tiny85 pre spracovanie a odosielanie údajov. Tento modul bol vybraný pre svoju spoľahlivosť, dobrú energetickú efektivitu a jednoduchú integráciu do navrhovaného systému.

Pre demonštráciu možnosti ovládania výstupných zariadení je v systéme implementované ovládanie LED diódy, ktorá reprezentuje svetelný zdroj v domácnosti.

6.4 Návrh softvérovej časti

Softvérová časť systému je navrhnutá s dôrazom na modularitu, škálovateľnosť a jednoduchú údržbu. Pre zabezpečenie konzistentného nasadenia všetkých komponentov je systém navrhnutý na využitie kontajnerizačnej technológie Docker. Táto kapitola popisuje základnú architektúru a návrh softvérových komponentov, zatiaľ čo detailná implementácia vrátane nasadenia pomocou Docker kontajnerov je popísaná v kapitole 8.

6.4.1 Architektúra serverovej časti

Serverová časť je navrhnutá na platforme Node.js s Express.js frameworkom a je rozdelená do logických vrstiev pre lepšiu udržiateľnosť kódu. Súčasťou návrhu je aj využitie MQTT

brokera pre komunikáciu s IoT zariadeniami.

MQTT klient tvorí základnú vrstvu pre komunikáciu s hardvérovými komponentmi. Zabezpečuje pripojenie k MQTT brokeru, prijímanie správ z tém, na ktoré je prihlásený, a publikovanie príkazov pre ESP32 zariadenia.

API vrstva definuje REST endpointy, ktoré slúžia na komunikáciu s používateľskou aplikáciou. API poskytuje metódy pre:

- Autentifikáciu a autorizáciu používateľov (registrácia, prihlásenie)
- Získanie aktuálnych a historických údajov zo senzorov
- Správu zariadení (pridávanie, úprava, odstránenie)
- Ovládanie pripojených zariadení (zapnutie/vypnutie LED)

Databázová vrstva zabezpečuje perzistenciu dát v relačnej databáze MySQL, poskytujúc efektívne uloženie a prístup k údajom zo senzorov a používateľským dátam.

Autentifikácia je realizovaná pomocou JWT tokenov pre identifikáciu používateľov.

6.4.2 Architektúra používateľského rozhrania

Architektúra používateľského rozhrania je navrhnutá ako SPA využívajúca React framework kombinovaný s TypeScript pre typovú bezpečnosť pre minimalizovanie chýb pri vývoji. Kladie dôraz na komponentový prístup pre znovupoužiteľnosť kódu.

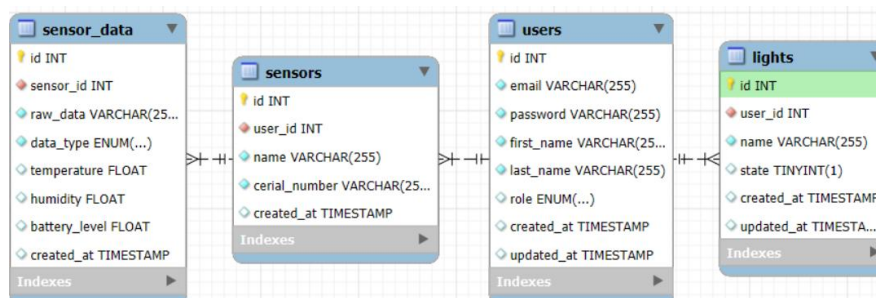
Komunikácia so serverom je uskutočňovaná s pomocou služieb, ktoré poskytujú rozhranie pre volanie REST API. Každá služba je zodpovedná za komunikáciu s konkrétnou časťou API:

- Služba pre zabezpečenie autentifikácie a autorizácie používateľov - prihlásenie, registrácia a správa používateľských účtov
- Služba pre prácu s údajmi zo senzorov - získavanie aktuálnych a historických údajov
- Služba pre ovládanie zariadení - zapínanie a vypínanie svetiel

Navigácia v používateľskom rozhraní je implementovaná s pomocou React Router a rozhranie je navrhnuté s dôrazom na vizuálnu konzistentnosť a responzivitu.

6.5 Návrh databázového modelu

Databáza systému je navrhnutá tak, aby efektívne uchovávala namerané údaje zo senzorov, informácie o registrovaných zariadeniach a používateľské údaje. Obrázok 6.2 znázorňuje ERD diagram databázy, ktorý zobrazuje vzťahy medzi jednotlivými tabuľkami.



Obrázok 6.2 ERD diagram databázy. (zdroj: vlastný)

6.5.1 Popis tabuliek a vzťahov

Databázový model pozostáva z nasledujúcich tabuliek:

Tabuľka 6.3 Databázový model

Názov tabuľky	Anglický názov	Popis
Používatelia	users	Používatelia systému
Senzory	sensors	Senzory pripojené k systému
Údaje zo senzorov	sensor_data	Historické údaje zo senzorov
Svetlá	lights	Svetlá pripojené k systému

Vzťahy medzi tabuľkami sú definované takto:

Tabuľka 6.4 Vzťahy medzi tabuľkami

Tabuľky	Vzťah	Popis
Používatelia - Senzory	1:N	Jeden používateľ môže mať viacero senzorov
Používatelia - Svetlá	1:N	Jeden používateľ môže mať viacero svetiel
Senzory - Údaje zo senzorov	1:N	Jeden senzor môže mať viacero historických údajov

7 IMPLEMENTÁCIA HARDVÉROVEJ ČASTI

Táto kapitola sa zaoberá implementáciou hardvérovej časti systému, pričom popisuje vývojové prostredie, použité jazyky a jednotlivé komponenty systému. Pre prehľadnosť sú v nasledujúcich častiach uvedené len najpodstatnejšie časti kódu. Kompletná implementácia vrátane podrobnej dokumentácie je dostupná v prílohe.

7.1 Vývojové prostredie a použité jazyky

7.1.1 Arduino IDE

Arduino IDE je využité na programovanie s ESP32 modulov pre jeho jednoduché a intuitívne prostredie. Umožňuje rýchle nahrávanie kódu do modulov a podporuje množstvo knižníc, ktoré sú potrebné pre prácu s komponentmi systému.

7.1.2 Programovací jazyk C++

Programovací jazyk C++ je zvolený pre jeho efektívnosť a širokú podporu vo vstavaných systémoch. Jeho základné konštrukcie sú dostatočne flexibilné na implementáciu potrebných funkcií pre ovládanie svetla a komunikáciu medzi modulmi.

7.2 Schéma zapojenia a použité komponenty na ovládanie svetla

Systém na ovládanie svetla je realizovaný pomocou dvoch ESP32 modulov, ktoré komunikujú prostredníctvom rádiových modulov na frekvencii 433 MHz. Celé zapojenie je rozdelené na dve hlavné časti: vysielacu a prijímaciu jednotku.

Vysielacia jednotka je založená na ESP32 LOLIN module, ktorý slúži ako most medzi MQTT serverom a rádiovým vysielateľom. Táto jednotka prijíma príkazy z MQTT brokera cez WiFi a následne ich preposiela pomocou 433 MHz rádiového vysielateľa prijímacej jednotke. Okrem toho obsahuje aj tlačidlo na manuálne ovládanie svetla a informačnú LED diódu pre signalizáciu stavu.

Prijímacia jednotka využíva ESP32 WROOM modul, ktorý prijíma signály z rádiového prijímateľa na frekvencii 433 MHz a ovláda pripojené svetlo reprezentované LED diódou. Nevyžaduje WiFi pripojenie, čo znižuje jej energetickú náročnosť a umožňuje umiestnenie aj v miestach bez WiFi pokrytia.

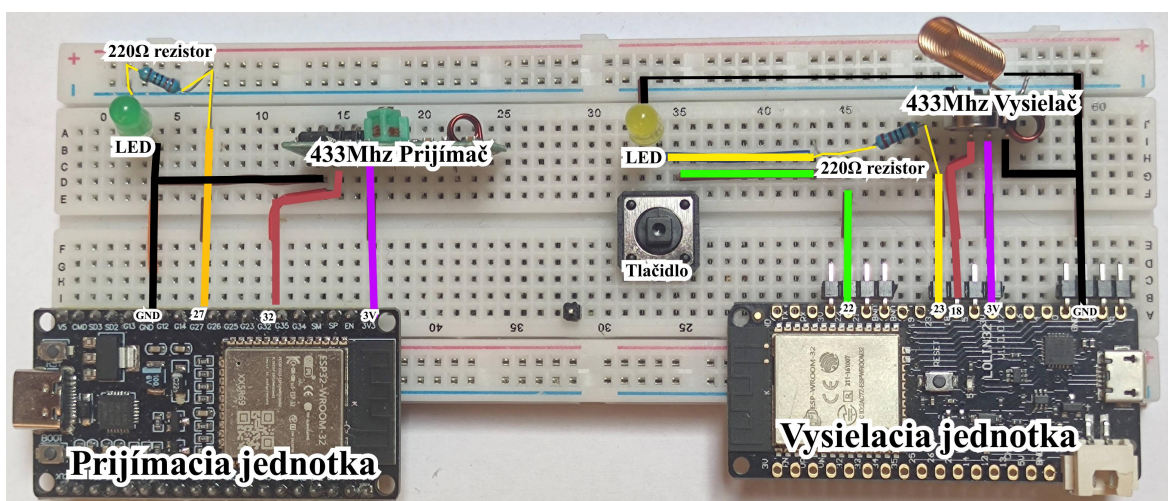
Výsledné priradenie pinov pre obe jednotky je uvedené v tabuľke 7.1.

Toto rozdelenie na dve samostatné jednotky umožňuje flexibilné umiestnenie v domácnosti. Pre demonštráciu zapojenia sú ale obidve jednotky umiestnené na jednom napájacom poli ako je zobrazené na obrázku 7.1.

Všetky komponenty boli vybrané s ohľadom na ich dostupnosť, cenu a jednoduchú integráciu do systému. ESP32 moduly obsahujú všetky potrebné periférie pre realizáciu projektu. Rádiové moduly na frekvencii 433 MHz ponúkajú dobrý dosah signálu aj cez steny a prekážky, čo je vhodné pre domáce použitie. Zoznam použitých komponentov je uvedený v prílohe.

Tabulka 7.1 Použité piny ESP32 modulov

Vysielacia jednotka		Prijímacia jednotka	
Pin	Funkcia	Pin	Funkcia
GND	GND	GND	GND
3.3V	3.3V	3.3V	3.3V
22	Tlačidlo	32	433 MHz prijímač
23	LED dióda	27	LED dióda
18	433 MHz vysielateľ		



Obrázok 7.1 Schéma zapojenia systému na ovládanie svetla. (zdroj: vlastný)

7.3 Implementácia ovládania svetla

7.3.1 Hardvérová konfigurácia

Pri implementácii ovládania svetla bolo potrebné zohľadniť rozdiely medzi ESP32 WROOM a LOLIN modulmi, ktoré majú odlišné usporiadanie pinov a špecifické vlastnosti. Pri výbere

pinov bolo dôležité vyhnúť sa tým, ktoré majú špeciálne funkcie alebo obmedzenia, a zároveň zabezpečiť prehľadné fyzické rozloženie komponentov na napájacom poli.

7.3.2 Softvérová implementácia

Softvérová implementácia ovládania svetla je rozdelená do dvoch hlavných častí: kód pre vysielaciu jednotku a kód pre prijímaciu jednotku. Kód je štruktúrovaný do funkcií, ktoré sú zodpovedné za jednotlivé úlohy.

Vysielacie a prijímacie jednotky využívajú jednoduchú binárnu hodnotu, čo minimalizuje chyby pri prenose a interpretácii dát. Implementácia tiež poskytuje riešenie pre obnovu spojenia v prípade výpadku WiFi alebo MQTT brokeru. V prílohe sa nachádzajú kompletné zdrojové kódy pre obidve jednotky s pridaným logovaním.

7.3.3 Vysielacia jednotka

Na začiatku programu pre vysielaciu jednotku je potrebné definovať použité knižnice a globálne premenné. Pre komunikáciu s MQTT brokerom je použitá knižnica PubSubClient, ktorá je určená pre ESP32 moduly. Pre komunikáciu s rádiovým modulom je použitá knižnica RH_ASK, ktorá je určená pre 433 MHz rádiové moduly. Všetky knižnice sú dostupné v Arduino IDE a je potrebné ich nainštalovať pred kompilovaním kódu.

Ďalej vo výpise 7.1 sú definované premenné pre WiFi pripojenie, MQTT broker a témy pre prichádzajúce a odosielané správy.

```
1 const char* ssid = "Nazov_WiFi";           // Názov WiFi siete
2 const char* password = "Heslo_WiFi";       // Heslo k WiFi sieti
3 // IP adresa počítača s MQTT brokerom, napr. 192.168.1.14
4 const char* mqtt_server = "IP_adresa_brokera";
5 // Názov svetla, ktorý sa bude zobrazovať v MQTT správe
6 #define LIGHT_NAME "KUCHYNA"
7 // Téma pre prichádzajúce správy
8 const char* subscribe_topic = "esp32/led/" LIGHT_NAME "/control";
9 // Téma pre odosielanie správ
10 const char* publish_topic = "esp32/led/" LIGHT_NAME "/state";
11 ...
12 ...
```

Výpis 7.1 Inicializácia premenných pre WiFi a MQTT

Pre získanie IP adresy počítača, na ktorom bude bežať MQTT broker, je možné v príkazovom riadku Windows použiť príkaz „ipconfig“.

Vo výpise 7.2 je zobrazená inicializácia rádiového modulu na pine 18. Pin 4 je definovaný

pre prijímač a pin 0xff pre spínanie vysielacza, avšak v tomto zapojení nie sú využité.

```
1 ĤĤ_ASK radio(2000, 4, 18, 0xff);
2 ...
3 ...
```

Výpis 7.2 Inicializácia rádiového modulu

Následne vo výpise 7.3 je zobrazené pripojenie na WiFi sieť.

```
1 ..WiFi.mode(WIFI_STA); // Nastavenie režimu WiFi
2 WiFi.begin(ssid, password); // Pripojenie na WiFi sieť
3 ...
4 ...
```

Výpis 7.3 Pripojenie na WiFi sieť

Po pripojení na WiFi sieť je potrebné nakonfigurovať komunikáciu s MQTT brokerom. Výpis 8.2 zobrazuje toto nastavenie vrátane callback funkcie. Port 1883 je predvolený štandardný port pre MQTT komunikáciu. Pre zabezpečenie stabilného pripojenia je implementovaná funkcia reconnect(), ktorá automaticky obnovuje spojenie v prípade výpadku.

```
1 ...// Nastavenie MQTT servera
2 client.setServer(mqtt_server, 1883);
3 client.setCallback(callback); // Nastavenie callback
   funkcie pre spracovanie prichádzajúcich správ
4 ...
5 void callback(char*topic, byte*payload, unsigned int length){
6   // Vytvorenie správy zo vstupu
7   String message;
8   for (unsigned int i = 0; i < length; i++) {
9     message += (char) payload[i];
10  }
11  ...
```

Výpis 7.4 Pripojenie na MQTT broker

Na odosielanie hodnôt je implementovaná funkcia „sendValue()”, zobrazená vo výpise 7.5. Táto funkcia prevádza logický stav na binárnu hodnotu (1 alebo 0) a odosiela ju prostredníctvom 433 MHz rádiového modulu, pričom čaká na potvrdenie úspešného odoslania.

```
1 void sendValue(bool state){
2   uint8_t value = state ? 1 : 0;
3   radio.send(&value, 1);
4   radio.waitPacketSent();
5 }...
```

Výpis 7.5 Odosielanie hodnôt cez 433 MHz vysieláč

Hlavná smyčka kontroluje pripojenie na WiFi a MQTT broker. Ak je pripojenie prerušené, pokúsi sa obnoviť spojenie. Následne sa kontroluje prichádzajúca správa z MQTT brokeru a na základe prijatej hodnoty sa odosiela signál cez 433 MHz vysielateľ.

7.3.4 Prijímacia jednotka

Prijímacia jednotka má jednoduchšiu implementáciu, keďže jej úlohou je len prijímať signály zo 433 MHz rádiového prijímača a ovládať LED diódu. Na strane prijímača je inicializácia rádiového modulu podobná ako na strane vysielateľa, avšak s inými pinmi. Vypis 7.6 zobrazuje inicializáciu prijímača a hlavnú smyčku, ktorá kontroluje prichádzajúce správy. Ak je prijatá správa, nastaví LED diódu na HIGH alebo LOW v závislosti od prijatej hodnoty.

```
1 RH_ASK radio(2000, 16, 5, 0xff);
2
3 void loop() {
4     // Zásobník pre prijaté dáta
5     uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
6     // Dĺžka zásobníka
7     uint8_t buflen = sizeof(buf);
8
9     // Kontrola prichádzajúcej správy
10    if (radio.recv(buf, &buflen)) {
11        buf[buflen] = '\0'; // Zabezpečenie ukončenia reťazca
12
13        if (buf[0] == 1) {
14            digitalWrite(LED_PIN, HIGH);
15        } else {
16            digitalWrite(LED_PIN, LOW);
17        }
18    }
19 }
20 }
```

Výpis 7.6 Prijímanie hodnôt zo 433 MHz prijímača

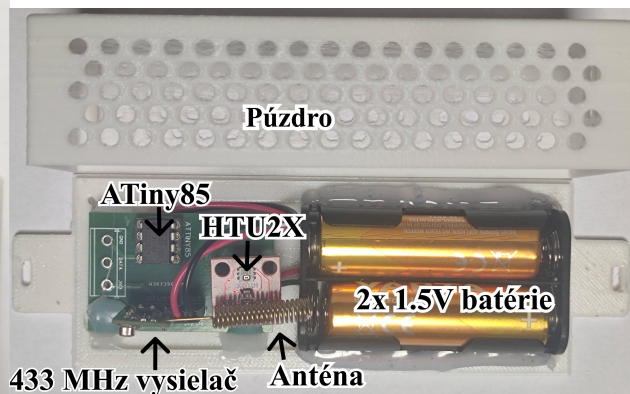
7.4 Integrácia SL-LOGIC teplotného a vlhkostného senzora

V práci bol použitý hotový senzorový modul poskytnutý firmou SL-LOGIC, ktorý bol integrovaný do navrhnutého systému. SL-LOGIC senzor sa skladá z dvoch vzájomne komunikujúcich častí. Meracia jednotka využíva mikrokontrolér ATTiny85 na spracovanie údajov zo senzora HTU2X, zabezpečuje čítanie hodnôt teploty a vlhkosti a následne ich odosiela prostredníctvom 433 MHz rádiového vysielateľa. Táto jednotka je napájaná z dvoch 1,5V batérií v sérii, čo zabezpečuje dlhú životnosť a flexibilné umiestnenie v domácnosti bez závislosti na pevnom napájaní.

Druhú časť tvorí komunikačná jednotka založená na module ESP32, ktorá prijíma vysielané údaje cez 433 MHz rádiový prijímač, spracováva ich a následne preposiela na MQTT broker. Integrácia tohto senzorového riešenia do navrhovaného systému umožňuje v reálnom čase poskytovať aktuálne informácie o teplote a vlhkosti v domácnosti.



Obrázok 7.2 SL-LOGIC senzor - komunikačná jednotka. (zdroj: vlastný)



Obrázok 7.3 SL-LOGIC senzor - meracia jednotka. (zdroj: vlastný)

7.4.1 Proces konfigurácie senzora

Konfigurácia SL-LOGIC senzora prebieha prostredníctvom webového rozhrania, ktoré je dostupné po pripojení k dočasnému prístupovému bodu vytvorenému senzorom pri prvom spustení. Proces konfigurácie pozostáva z nasledujúcich krokov:

1. Po pripojení napájania senzor vytvorí WiFi sieť s názvom „ESP_XXXX“, kde XXXX je jedinečný identifikátor zariadenia
2. Po pripojení k tejto sieti je potrebné otvoriť webovú adresu 192.168.4.1
3. V konfigurácii je potrebné zadať parametre domácej WiFi siete (SSID a heslo)
4. Následne je potrebné zadať adresu MQTT brokera a port (štandardne 1883)
5. Po zadaní všetkých potrebných údajov je potrebné kliknúť na tlačidlo „Save“ (uložiť)

Po dokončení konfigurácie sa senzor reštartuje a pripojí k zadanej WiFi sieti. V prípade úspešného pripojenia začne odosielať namerané údaje na nakonfigurovaný MQTT broker.

8 IMPLEMENTÁCIA SOFTVÉROVEJ ČASTI

Softvérová časť systému je implementovaná ako webová aplikácia, ktorá umožňuje používateľom ovládať svetlo a monitorovať stav senzorov. Aplikácia je rozdelená na dve hlavné časti: serverovú časť a používateľské rozhranie. Serverová časť spracováva dáta zo senzorov, zabezpečuje autentifikáciu používateľov a komunikuje s MQTT brokerom. Používateľské rozhranie poskytuje možnosť interakcie so systémom.

Kompletná implementácia je uvedená v prílohe.

8.1 Vývojové prostredie a technológie

Pre implementáciu bol zvolený Visual Studio Code z dôvodu rozsiahlej podpory webových aplikácií a rozšírení. Hlavné technológie zahŕňajú JavaScript pre serverovú časť (Node.js ekosystém, REST API, MQTT spracovanie), TypeScript pre používateľské rozhranie (statická typová kontrola, spoľahlivosť kódu) a SQL pre MySQL databázu ukladajúcu údaje zo senzorov a používateľské účty.

8.2 Serverová časť

8.2.1 Výber a konfigurácia MQTT brokera

V systéme je použitý MQTT broker Mosquitto, ktorý zabezpečuje komunikáciu medzi IoT zariadeniami a serverovou časťou aplikácie. Mosquitto bol zvolený pre svoju dostupnosť, jednoduchú konfiguráciu a spoľahlivosť pri práci s protokolom MQTT.

8.2.2 Nasadenie systému pomocou Docker kontajnerov

Pre zjednodušenie nasadenia a elimináciu potreby manuálnej inštalácie je celý systém vrátane Mosquitto MQTT brokera, serverovej časti a databázy implementovaný ako sada Docker kontajnerov. Toto riešenie zabezpečuje konzistentnú konfiguráciu a umožňuje rýchle nasadenie celého systému. Pre toto nasadenie je použitý súbor „docker-compose.yml“, ktorý definuje všetky potrebné služby a ich konfigurácie.

8.2.3 Implementácia Node.js servera

Serverová časť je implementovaná pomocou Node.js s použitím frameworku Express.js, ktorý poskytuje robustnú infraštruktúru pre tvorbu webových aplikácií a API. Celá apliká-

cia je štruktúrovaná podľa vzoru MVC (Model-View-Controller), ktorý rozdeľuje kód do logických celkov, čím zvyšuje jeho prehľadnosť a udržateľnosť.

Základná štruktúra servera inicializuje Express aplikáciu, nastavuje aplikačné programové vybavenie pre spracovanie dát a definuje základné endpointy pre komunikáciu s používateľskou aplikáciou. Úryvok tejto inicializácie je zobrazený na nasledujúcom výpise 8.1.

```
1  // Definovanie ako express aplikáciu
2  const app = express();
3  // Nastavenie lokálneho portu na ktorom server beží
4  const port = 3500;
5  // Nastavenie hosta na ktorom server beží,
6  const host = "0.0.0.0"; // server bude počúvať na všetkých
7  // dostupných sieťových rozhraniach
8  // Nastavenie aplikačného programového vybavenia
9  app.use(cors());
10 app.use(express.json());
11
12 // Po úspešnom inicializovaní databázy sa spustí server
13 initializeDatabase().then(async (success) => {
14 ...
```

Výpis 8.1 Inicializácia Node.js servera

Server poskytuje REST API endpointy pre komunikáciu s používateľskou aplikáciou, zabezpečuje autentifikáciu používateľov pomocou JWT a spracováva dáta zo senzorov prostredníctvom MQTT protokolu.

8.2.4 Spracovanie MQTT správ

Kľúčovým komponentom systému je služba „mqttService.js“, ktorá zabezpečuje komunikáciu medzi serverom a IoT zariadeniami prostredníctvom protokolu MQTT. Táto služba sa inicializuje pri spustení servera. Výpis 8.2 demonštruje príkazy na pripojenie k MQTT brokeru, odber správ z tém a spracovanie prichádzajúcich správ.

```
1  // Spracovanie pripojenia na broker
2  client.on("connect", () => {
3    // Prihlásenie na odber tém
4    // esp32 pre LED | gw pre senzory
5    const topics = ["esp32/led/state", "gw/#"];
6    client.subscribe(topics, (err) => {
7
8  // Spracovanie správy
9  client.on("message", async (topic, message) => {
10 ...
```

Výpis 8.2 Spracovanie MQTT správ

V kóde je vidieť, že sa spracovávajú správy z dvoch tém. Prvá téma „esp32/led/#“ slúži na prijatie správ o stave LED diódy, zatiaľ čo druhá téma „gw/#“ slúži na prijatie správ zo senzorov. Správy sú spracovávané v závislosti od témy, pričom sa volajú príslušné funkcie na aktualizáciu stavu LED diódy alebo ukladanie údajov zo senzorov do databázy.

SL-LOGIC senzor odosiela správy vo formáte „ST315:231:58X“, kde:

- S - označuje začiatok správy
- T315 - sériové číslo senzora
- 231 - surová hodnota teploty, ktorú je potrebné previesť na skutočnú teplotu vydelením desiatimi ($231/10 = 23,1$ °C)
- 58 - vlhkosť v percentách (58 %)
- X - označuje koniec správy

Pri spracovaní údajov je potrebné zohľadniť špeciálne prípady:

- Ak je hodnota vlhkosti 254, znamená to, že správa obsahuje informáciu o stave batérie. V tomto prípade druhý údaj (napr. 160) predstavuje napätie batérie v desatinách, ktoré sa prepočíta na volty podľa vzorca: $(\text{hodnota}/10) * 0,2$. Napríklad ST315:160:254X znamená napätie batérie $160/10 * 0,2 = 3,2$ V.
- Ak je hodnota vlhkosti vyššia ako 100, ide o chybu a hodnota je obmedzená na 100 %.

Pre spracovanie dát zo senzorov je implementovaná funkcia vo výpise 8.3, ktorá extrahuje údaje z prijatej správy a ukladá ich do databázy. Funkcia najprv skontroluje, či správa začína a končí správnymi znakmi (S a X). Následne extrahuje sériové číslo senzora a vyhladá ho v databáze. Ak je senzor nájdený, vytvorí záznam s nameranými údajmi.

```

1  const receivedSensorData = async (topic, messageStr) => {
2    if (messageStr.startsWith("S") && messageStr.endsWith("X"))
3      {
4        // Extrakcia sériového čísla senzora (napr. T310)
5        const sensorSerialMatch = messageStr.match(/S(T\d+):/);
6        if (sensorSerialMatch && sensorSerialMatch[1]) {
7          const serialNumber = sensorSerialMatch[1];
8          // Vyhľadanie senzora v databáze podľa sériového čísla
9          const sensor = await getSensorBySerialNumber(
10             serialNumber);
11          if (sensor) {
12            // Vytvorenie záznamu nameraných údajov s ID nájdené
13              ho senzora
14          ...

```

Výpis 8.3 Spracovanie údajov zo senzorov

Na ovládanie svetla je implementovaná opačná komunikácia, ktorá umožňuje odosielať príkazy z aplikácie na ESP32 pomocou funkcie „publishMessage“ vo výpise 8.4. Táto funkcia zabezpečuje, že ak je klient pripojený k brokeru, príkaz sa odošle na zadanú tému.

```
1 //`Topic v tomto prípade je "esp32/led"
2 // a správa je buď zapnúť alebo vypnúť LED diódu (1/0)
3 export const publishMessage = (topic, message) => {
4   if (client && client.connected) {
5     if (typeof message === "number" || typeof message === "
6         boolean") {
7       client.publish(topic, message.toString());
8     }
9     console.log(`Published to ${topic}: ${message}`);
10  }
11 };
12 ...
```

Výpis 8.4 Odosielanie príkazov na ESP32

8.2.5 Spracovanie, úprava a ukladanie dát

Pre ukladanie dát zo senzorov a stavu svetiel je použitá databáza MySQL, ktorá poskytuje spoľahlivé a škálovateľné riešenie. Databázová vrstva je implementovaná pomocou modelov, ktoré poskytujú metódy na vytvorenie tabuliek, vkladanie a získavanie údajov.

Databázová vrstva je implementovaná pomocou knižnice „mysql2-promise“, ktorá poskytuje jednoduché rozhranie pre prácu s MySQL databázami v Node.js. Vo výpise 8.5 je zobrazená konfigurácia databázy s pripojením na MySQL server.

```
1 //`Definovanie pripojenia na databázu
2 let con = mysql.createPool({host, user, password,});
3 async function initializeDatabase() {
4   try {
5     // Vytvorenie databázy, ak neexistuje
6     await con.query('CREATE DATABASE IF NOT EXISTS ${database}');
7     // Výber databázy pre budúce dotazy
8     await con.query('USE ${database}');
9     // Ak databáza existuje, pripojenie sa k nej
10    con = mysql.createPool({host,user,password, database});
11  }
12  ...
13 ...
```

Výpis 8.5 Konfigurácia databázy

Databázová vrstva je rozdelená do viacerých modelov, ktoré sú zodpovedné za prácu s konkrétnymi tabuľkami ako sú vytvorenie, ukladanie a získavanie údajov. Výpis 8.6 zobrazuje vytvorenie tabuľky „sensors“. Rovnakým spôsobom sú vytvorené aj ostatné tabuľky,

ako napríklad „users“, „sensor_data“ a „lights“.

```

1 ... // Vytvorenie tabulky senzorov
2   const query_sensors = {
3     CREATE TABLE IF NOT EXISTS sensors (
4       id INT AUTO_INCREMENT PRIMARY KEY,
5       user_id INT NOT NULL,
6       name VARCHAR(255) NOT NULL,
7       serial_number VARCHAR(255) NOT NULL,
8       created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
9       FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
10        CASCADE
11     );
12   await con.query(query_sensors);
13 ...

```

Výpis 8.6 Ukážka vytvorenia tabuľky

8.3 Používateľské rozhranie

8.3.1 Štruktúra React aplikácie

Používateľské rozhranie je implementované pomocou knižnice React s využitím TypeScriptu, čo poskytuje výhody typovej kontroly počas vývoja. Celá aplikácia je štruktúrovaná podľa komponentového prístupu, ktorý umožňuje jednoduchú údržbu a znovupoužiteľnosť kódu.

Základná štruktúra aplikácie začína komponentom App, ktorý obaluje celú aplikáciu do AuthProvider komponentu a definuje základné smerovanie pomocou BrowserRouter. Výpis 8.7 zobrazuje základnú štruktúru aplikácie.

```

1 function App() {
2   return (
3     <AuthProvider>
4       <BrowserRouter>
5         <AppRouter />
6       </BrowserRouter>
7     </AuthProvider>
8   )
9 }

```

Výpis 8.7 Základná štruktúra aplikácie v App.tsx

Smerovanie v aplikácii je implementované pomocou React Router, pričom v komponente „AppRouter“ sa rozlišuje medzi verejne dostupnými a chránenými endpointmi.

Architektúra aplikácie sa skladá z nasledujúcich kľúčových vrstiev:

- Kontextové API - zabezpečuje zdieľanie stavu naprieč aplikáciou.
- Služby - abstrakcia komunikácie so serverovým API.
- Komponenty - znovupoužiteľné stavebné bloky používateľského rozhrania.

- Stránky - špecifické komponenty pre jednotlivé stránky aplikácie.
- Pomocné funkcie - všeobecné pomocné funkcie.

8.3.2 Implementácia používateľského rozhrania

Používateľské rozhranie aplikácie je navrhnuté s dôrazom na jednoduchosť a intuitívnosť ovládania. Pre štylizáciu je implementovaný Bootstrap framework v kombinácii s vlastnými SCSS štýlmi, čo poskytuje jednotný vizuálny jazyk naprieč celou aplikáciou.

Rozloženie stránky definované v komponente „Layout“ je rozdelené na tri hlavné časti:

- Topbar - horný panel s logom a prístupom k profilu.
- Navbar - bočný panel s hlavnou navigáciou aplikácie.
- Content container - priestor pre zobrazenie obsahu.

Aplikácia je plne responzívna, s prispôbením pre mobilné zariadenia. Pri menšom rozlíšení obrazovky sa bočný panel automaticky skryje a zobrazí sa len po kliknutí na tlačidlo reprezentujúce navigáciu, čo optimalizuje využitie dostupnej plochy obrazovky.

Farebná schéma aplikácie je konzistentne definovaná v centrálnom súbore „colors.scss“, čo zabezpečuje jednotný vzhľad a jednoduchú úpravu vizuálneho štýlu. Pre lepšiu používateľskú skúsenosť sú implementované aj vizuálne prvky ako ikony.

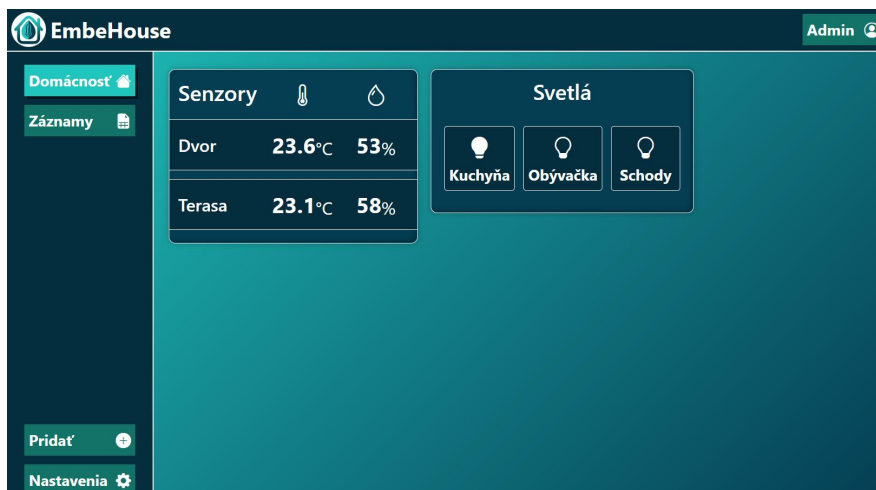
Prvotný návrh aplikácie bol realizovaný pomocou nástroja Figma, čo umožnilo rýchle prototypovanie a testovanie dizajnu pred samotnou implementáciou. V prílohe sú uvedené pôvodné návrhy aplikácie.

8.3.3 Ovládanie svetiel a vizualizácia dát

Pre zobrazenie a interakciu so systémom sú implementované špecializované komponenty, ktoré poskytujú intuitívne používateľské rozhranie. Na hlavnej stránke sú integrované dve hlavné kategórie komponentov:

- Senzory - zobrazujú aktuálne hodnoty teploty a vlhkosti v prehľadnej forme
- Svetlá - umožňujú jednoduché ovládanie svetiel pomocou grafických prvkov

Komponent svetiel vizuálne indikuje stav svetla (zapnuté/vypnuté) pomocou ikony žiarovky a reaguje na interakciu používateľa zmenou svojho stavu. Pri kliknutí na komponent sa odošle požiadavka na zmenu stavu svetla. Obrázok 8.1 znázorňuje hlavnú stránku aplikácie, na ktorej sú zobrazené oba uvedené komponenty.



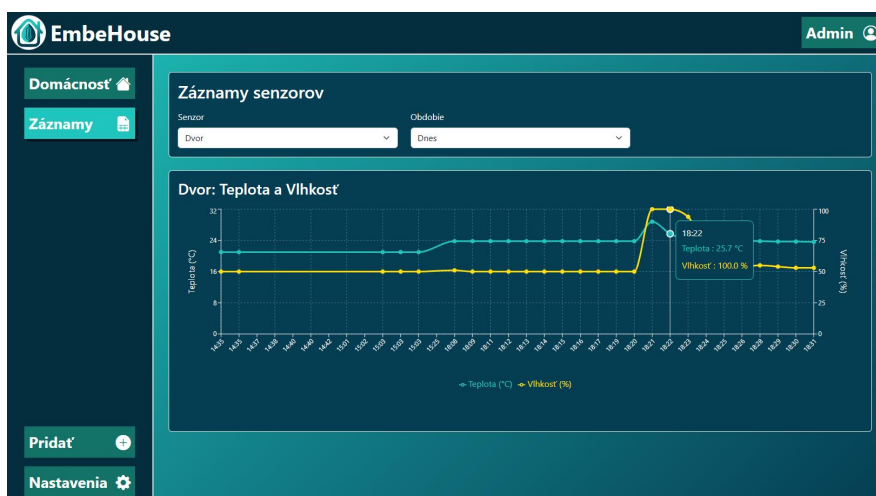
Obrázok 8.1 Aplikácia EmbeHouse. (zdroj: vlastný)

Pre vizualizáciu historických údajov zo senzorov je implementovaná stránka „Záznamy“, ktorá poskytuje možnosť zobrazenia dát v grafickej forme. Používateľ má možnosť vybrať konkrétny senzor a definovať časové obdobie, za ktoré chce údaje zobrazit’.

Vizualizácia je realizovaná pomocou knižnice „Recharts“, ktorá umožňuje vytvorenie interaktívnych a responzívnych grafov. Údaje sú vyobrazené formou grafu, ktorý súčasne znázorňuje vývoj teploty a vlhkosti v čase (obrázok 8.2).

Proces vizualizácie prebieha v nasledujúcich krokoch:

1. Používateľ vyberie požadovaný senzor a špecifikuje časové obdobie
2. Systém spracuje údaje z databázy do formátu kompatibilného s knižnicou Recharts
3. Výsledný graf poskytuje prehľadnú vizualizáciu časového vývoja meraných údajov



Obrázok 8.2 Zobrazenie historických údajov zo senzorov. (zdroj: vlastný)

8.4 Komunikácia medzi komponentami systému

Aplikácia využíva REST API na komunikáciu medzi používateľskou aplikáciou a serverom. API rozhranie je navrhnuté podľa štandardných princípov REST s využitím HTTP metód a JSON formátu pre prenos dát.

Všetky endpointy okrem registrácie a prihlásenia vyžadujú platný JWT v hlavičke „Authorization“, čím je zabezpečený prístup len pre autorizovaných používateľov.

8.5 Autentifikácia a autorizácia

Aplikácia implementuje zabezpečenie založené na JWT tokene, ktorý je generovaný serverovou časťou pri úspešnom prihlásení alebo registrácii používateľa. Tento token obsahuje informácie o používateľovi a je podpísaný tajným kľúčom, čím sa zabezpečuje jeho integrita a autenticita. Výpis 8.8 zobrazuje generovanie tokenu JWT.

```
1 export const generateToken = (user) => {
2   return jwt.sign(
3     {
4       id: user.id,
5       email: user.email,
6       role: user.role,
7     },
8     JWT_SECRET,
9     { expiresIn: JWT_EXPIRES_IN }
10  );
11 };
```

Výpis 8.8 Generovanie JWT tokenu

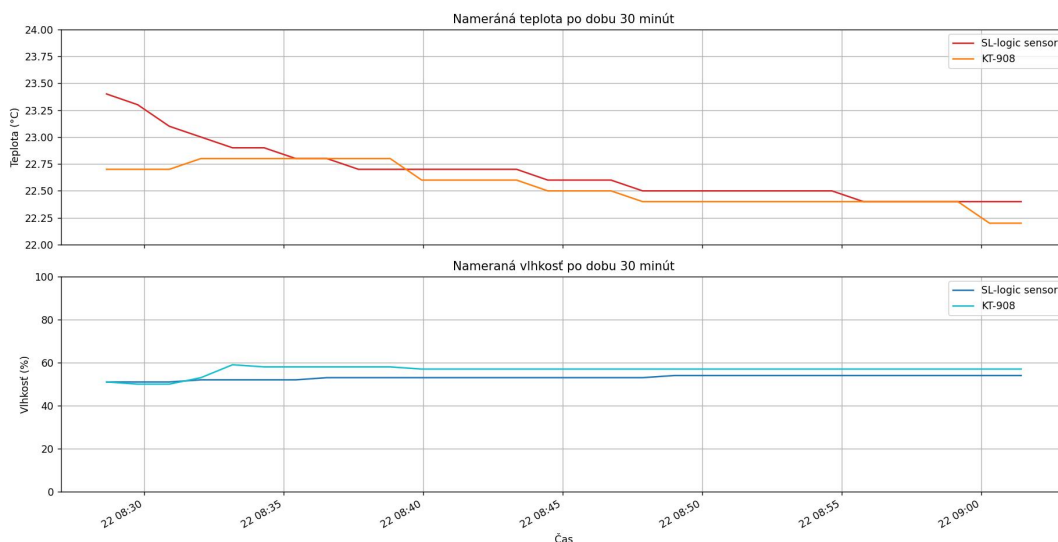
Táto implementácia zabezpečenia kombinuje overené postupy na strane serverovej časti aj používateľského rozhrania, čím poskytuje účinný mechanizmus na ochranu citlivých údajov a funkcionalít systému.

9 TESTOVANIE SYSTÉMU

9.1 Test senzoru

V rámci testovania presnosti a spoľahlivosti senzoru boli vykonané komparatívne merania s komerčne dostupnými referenčnými zariadeniami. Na testovanie presnosti merania teploty a vlhkosti pomocou SL-LOGIC senzorov bola použitá metóda porovnávania s referenčným senzorom KT-908, ktorého teplomer má presnosť $\pm 1^\circ\text{C}$ a vlhkomer $\pm 5\%$. [83]

Testovanie prebiehalo v interiéri pri bežnej izbovej teplote ($22\text{--}24^\circ\text{C}$). Výsledky je vidno na obrázku 9.1. Z grafu je možné vyčítať, že SL-LOGIC senzor vykazuje veľmi dobrú presnosť merania teploty a vlhkosti v porovnaní s referenčným zariadením KT-908. Odchýlky pohybovali v rozmedzí $\pm 0,7^\circ\text{C}$ pre teplotu a $\pm 7\%$ pre vlhkosť.



Obrázok 9.1 Porovnanie meraní teploty a vlhkosti v pri bežnej izbovej teplote medzi SL-LOGIC senzorom a referenčným zariadením KT-908 (zdroj: vlastný)

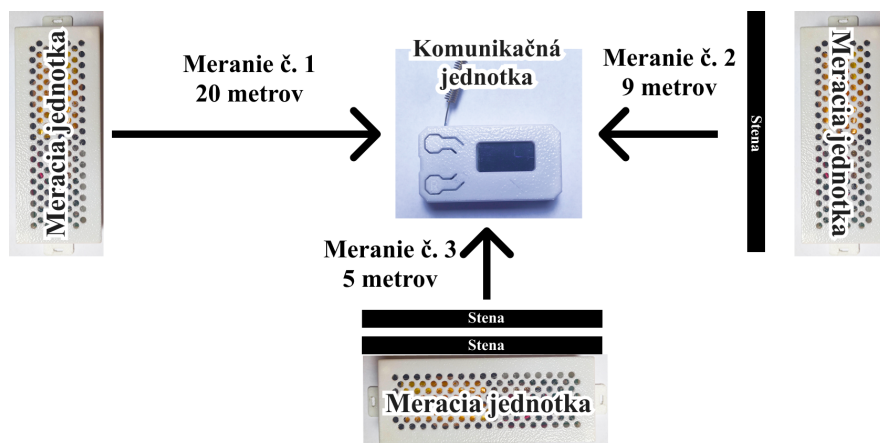
9.2 Testovanie komunikácie

Testovanie komunikácie bolo zamerané na overenie spoľahlivosti prenosu údajov prostredníctvom 433 MHz rádiových modulov a následne cez MQTT protokol na server. Testy boli uskutočnené tak, aby identifikovali prípadné obmedzenia systému.

Pre testovanie dosahu 433 MHz komunikácie boli vykonané merania v rôznych typoch prostredí:

- Meranie č. 1 - priama viditeľnosť: obe časti senzora boli umiestnené vo voľnom priestore bez prekážok, pričom spoľahlivý dosah dosiahol až 20 metrov

- Meranie č. 2 - jedna prekážka: meracia jednotka bola umiestnená za jednou stenou, dosah sa znížil na približne 9 metrov
- Meranie č. 3 - dve prekážky: meracia jednotka bola umiestnená za dvoma stenami, dosah klesol na približne 5 metrov



Obrázok 9.2 Testovanie dosahu 433 MHz rádiového modulu v rôznych prostrediach. (zdroj: vlastný)

Testovanie MQTT komunikácie pozostávalo z monitorovania úspešnosti prenosu správ medzi ESP32 a serverom. Počas dvojhodinového testu s prednastaveným intervalom odosielania správ každých 68 sekúnd bolo odoslaných a úspešne prijatých 100% správ. Tento výsledok potvrdzuje vysokú spoľahlivosť a stabilitu komunikácie v rámci systému.

9.3 Testovanie ovládania svetla

Testovanie funkcionality ovládania svetla bolo zamerané na overenie spoľahlivosti a odozvy systému na príkazy používateľ'a. Testy zahŕňali ovládanie prostredníctvom webového rozhrania, ako aj pomocou fyzického tlačidla na vysielacej jednotke.

Pre posúdenie spoľahlivosti ovládania cez webové rozhranie bol vykonaný test pozostávajúci zo 400 opakovaných príkazov na zapnutie a vypnutie svetla, pričom systém dosiahol úspešnosť vykonania 97,25% príkazov. Priemerný čas od zadania príkazu vo webovom rozhraní po spracovanie a odoslanie zmeny stavu svetla na broker bol 91 ms pri stabilnom internetovom pripojení.

Testovanie fyzického ovládania pomocou tlačidla na vysielacej jednotke preukázalo 98,3% spoľahlivosť v 300 opakovaných testoch s priemernou odozvou 72 ms. Tento výsledok potvrdzuje, že systém je schopný rýchlo a efektívne reagovať na príkazy používateľ'a.

ZÁVER

Táto bakalárska práca sa zamerala na návrh a vývoj domáceho asistenta s využitím moderných IoT technológií, konkrétne mikrokontroléru ESP32 v kombinácii s rádiovým modulom na frekvencii 433 MHz. Cieľom bolo vytvoriť systém na monitorovanie údajov zo senzorov s možnosťou ovládania svetiel v domácnosti, ktorý by poskytoval používateľsky prívetivé rozhranie.

V súlade so zadaním bol implementovaný funkčný prototyp systému pozostávajúci z hardvérovej časti (senzor teploty a vlhkosti, ESP32 moduly, rádiové moduly) a softvérovej časti (Node.js server, MQTT broker, React aplikácia). Systém demonštruje možnosti bezdrôtovej komunikácie medzi IoT zariadeniami a serverom prostredníctvom protokolu MQTT.

Navrhnuté používateľské rozhranie poskytuje vizualizáciu aktuálnych aj historických dát zo senzorov, umožňuje ovládanie svetiel a zahŕňa systém autentifikácie používateľov. Testovanie systému potvrdilo jeho spoľahlivosť pri bežnom používaní, s dosahom rádiovkej komunikácie postačujúcim pre štandardnú domácnosť.

Pri hodnotení limitácií riešenia boli identifikované nasledujúce oblasti pre potenciálne zlepšenie:

- Aktualizácia dát v reálnom čase - implementácia websocket komunikácie namiesto opakovaného dotazovania pre efektívnejšiu komunikáciu
- Energetická efektívnosť - použitie úspornejšieho mikrokontroléra v prijímacej jednotke ovládania svetla
- Bezpečnosť komunikácie - rozšírenie MQTT brokera o TLS/SSL a autentifikáciu pre zvýšenie bezpečnosti prenášaných údajov

Napriek týmto limitáciám predstavuje vytvorený systém dobrý základ pre praktické nasadenie v oblasti domácej automatizácie. Medzi možné budúce rozšírenia patrí integrácia ďalších senzorov a prepojenie s inými IoT platformami.

Záverom možno konštatovať, že navrhnuté riešenie spĺňa stanovené ciele, je praktickou ukážkou aplikácie teoretických znalostí a predstavuje základ pre budúci vývoj komplexnejších systémov domácej automatizácie.

ZOZNAM POUŽITÉJ LITERATURY

- [1] AZIZI, Benjamin; ARENAS, Alvaro a CRISPO, Bruno, 2016. *Engineering Secure Internet of Things Systems*. Institution of Engineering and Technology (The IET). 1. vyd. ISBN 978-1-78561-053-0.
- [2] VERMESAN, Ovidiu; FRIESS, Peter; GUILLEMIN, Patrick; GUSMEROLI, Sergio; SUNDMAEKER, Harald et al., 2011. *Internet of Things Strategic Research Roadmap*. In: *Internet of Things - Global Technological and Societal Trends from Smart Environments and Spaces to Green Ict*. River Publishers, s. 9–54. ISBN 978-1-003-33860-4. Dostupné z: https://www.researchgate.net/publication/267566519_Internet_of_Things_Strategic_Research_Roadmap.
- [3] *Responsive environments*, 2016. Online. Dostupné z: <https://www.britannica.com/technology/responsive-environments>. [cit. 2025-02-04].
- [4] *That ‘Internet of Things’ Thing*, 2009. Online. Dostupné z: <https://www.rfidjournal.com/expert-views/that-internet-of-things-thing/73881/>. [cit. 2025-02-03].
- [5] LI, Guowei, 2024. *8 IoT Protocols and Standards Worth Exploring in 2024*. Online. Dostupné z: <https://www.emqx.com/en/mqtt>. [cit. 2025-05-09].
- [6] *Co je MQTT a k čemu slouží ve IIoT? Popis protokolu MQTT*, 2025. Online. Dostupné z: <https://ipc2u.cz/blogs/news/mqtt-protokol>. [cit. 2025-02-13].
- [7] MALÝ, Martin, 2024. *ESP32 prakticky*. Edici CZ.NIC. ISBN 978-80-88168-79-9.
- [8] HODDIE, Peter a PRADER, Lizzie, 2020. *IoT development for ESP32 and ESP8266 with JavaScript: a practical guide to XS and the moddable SDK*. Apress, United States. Dostupné z: <https://doi.org/9781484250709>.
- [9] SINGH, Rajesh; GEHLOT, Anita; GUPTA, Lovi Raj; SINGH, Bhupendra a TYAGI, Priyanka, 2019. *Getting Started for Internet of Things with Launch Pad and ESP8266*. In: *Interfacing of 433 MHz RF Transmitter and Receiver*. River Publishers, s. 137–142. ISBN 978-87-7022-068-2. Dostupné z: <https://app.knovel.com/hotlink/pdf/id:kt012KYQZ2/getting-started-internet/interfacin-introduction>.

- [10] *What is a Virtual Assistant (AI Assistant)?*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/what-is-a-virtual-assistant-ai-assistant/>. [cit. 2025-03-01].
- [11] *What is Home Assistant, how does it work, and what do you need to get started?*, 2023. Online. Dostupné z: <https://www.pocket-lint.com/what-is-home-assistant-how-does-it-work/>. [cit. 2025-03-01].
- [12] *What is Google Assistant and what can it do?*, 2023. Online. Dostupné z: <https://www.pocket-lint.com/what-is-google-assistant/>. [cit. 2025-03-01].
- [13] *What Is Alexa?*, 2025. Online. Dostupné z: <https://developer.amazon.com/en-GB/alexa>. [cit. 2025-03-01].
- [14] *What is Alexa and what can Amazon Echo do?*, 2023. Online. Dostupné z: <https://www.pocket-lint.com/what-is-alexa-how-does-it-work-and-what-can-amazons-alexa-do/>. [cit. 2025-03-01].
- [15] *Amazon Echo*, 2023. Online. Dostupné z: <https://www.techtarget.com/searchaws/definition/Amazon-Echo>. [cit. 2025-03-01].
- [16] *HomeKit: Everything You Need to Know*, 2022. Online. Dostupné z: <https://www.macrumors.com/guide/homekit/>. [cit. 2025-03-01].
- [17] *Who are we*, 2025. Online. Dostupné z: <https://www.openhab.org/about/who-we-are.html>. [cit. 2025-03-16].
- [18] *Introduction to Smart Home Cybersecurity Risks*, 2024. Online. Dostupné z: <https://medium.com/@RocketMeUpCybersecurity/introduction-to-smart-home-cybersecurity-risks-a88a0feb6f1f>. [cit. 2025-05-04].
- [19] *Smart home security: how to protect your smart home*, 2025. Online. Dostupné z: <https://www.kaspersky.com/resource-center/preemptive-safety/smart-home-security>. [cit. 2025-05-04].
- [20] *What is a microcontroller (MCU)?*, 2024. Online. Dostupné z: <https://www.techtarget.com/iotagenda/definition/microcontroller>. [cit. 2025-02-21].

- [21] *What is a microcontroller?*, 2024. Online. Dostupné z: <https://www.ibm.com/think/topics/microcontroller>. [cit. 2025-02-21].
- [22] SAINI, Manish Kumar, 2023. *Difference Between 8-Bit and 16-Bit Microcontroller*. Online. Dostupné z: <https://www.tutorialspoint.com/difference-between-8-bit-and-16-bit-microcontroller>. [cit. 2025-04-28].
- [23] *Microcontroller and its Types*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/microcontroller-and-its-types/>. [cit. 2025-04-28].
- [24] SHEPARD, Jeff, 2022. *MCU Bits and Bytes: sizes, types, and uses*. Online. Dostupné z: <https://www.microcontrollertips.com/mcu-bits-and-bytes-sizes-types-and-uses-fa/>. [cit. 2025-05-01].
- [25] *Difference between Von Neumann and Harvard Architecture*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/>. [cit. 2025-05-01].
- [26] *Computer Organization: RISC and CISC*, 2024. Online. Dostupné z: <https://www.geeksforgeeks.org/computer-organization-risc-and-cisc/>. [cit. 2025-05-01].
- [27] *ESP8266 module comparison: ESP-01, ESP-05, ESP-12, ESP-201, Test Board and NodeMCU*, 2015. Online. Dostupné z: <https://blog.squix.org/2015/03/esp8266-module-comparison-esp-01-esp-05.html>. [cit. 2025-03-16].
- [28] *ESP8266 Lua NodeMcu V3 WIFI CH340*, 2025. Online. Dostupné z: <https://dratek.cz/arduino/122953-lua-nodemcu-esp8266-v3-wifi-modul.html>. [cit. 2025-03-17].
- [29] *Guide to IoT with ESP32*, 2025. Online. Dostupné z: <https://www.nabto.com/guide-to-iot-esp-32/>. [cit. 2025-05-02].
- [30] *Arduino Nano*, 2025. Online. Dostupné z: <https://store.arduino.cc/en-cz/products/arduino-nano>. [cit. 2025-03-17].
- [31] BARROZO, Jharwin, 2023. *Introduction to the ATtiny85 Microcontroller*. Online. Dostupné z: <https://www.flux.ai/p/blog/introduction-to-the-attiny85-microcontroller>. [cit. 2025-04-28].

- [32] *ATTINY85-20U IC MCU 8BIT 8KB FLASH 8DIP ATMEL*, 2025. Online. Dostupné z: <https://dratek.cz/arduino/1512-attiny85-20pu-ic-mcu-8bit-8kb-flash-8dip-atmel.html>. [cit. 2025-05-020].
- [33] *What is a Low Level Language?*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/what-is-a-low-level-language/>. [cit. 2025-03-06].
- [34] *Introduction to C++ Programming Language*, 2024. Online. Dostupné z: <https://www.geeksforgeeks.org/introduction-to-c-programming-language/>. [cit. 2025-03-08].
- [35] *What is IDE?*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/what-is-ide/>. [cit. 2025-04-25].
- [36] *Arduino IDE*, 2025. Online. Dostupné z: <https://www.tpointtech.com/arduino-ide>. [cit. 2025-03-17].
- [37] *What are the pros and cons of using Arduino IDE vs. other development tools for embedded software?*, 2025. Online. Dostupné z: <https://www.linkedin.com/advice/0/what-pros-cons-using-arduino-ide-vs-other-development>. [cit. 2025-04-23].
- [38] *Espressif IDE*, 2025. Online. Dostupné z: <https://docs.espressif.com/projects/espressif-ide/en/latest/index.html>. [cit. 2025-04-24].
- [39] *Espressif IoT Development Framework*, 2025. Online. Dostupné z: <https://github.com/espressif/esp-idf>. [cit. 2025-04-24].
- [40] *Why did we build Visual Studio Code?*, 2025. Online. Dostupné z: <https://code.visualstudio.com/docs/editor/whyvscode>. [cit. 2025-04-25].
- [41] *PlatformIO*, 2025. Online. Dostupné z: <https://platformio.org>. [cit. 2025-04-25].
- [42] *What is PlatformIO?*, 2025. Online. Dostupné z: <https://docs.platformio.org/en/latest/what-is-platformio.html>. [cit. 2025-04-25].
- [43] *An Overview of Development Board*, 2021. Online. Dostupné z: <https://www.utmel.com/blog/categories/pcb/an-overview-of-development-board>. [cit. 2025-04-25].

- [44] CAMERON, Neil, 2021. *Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and Wifi Enabled Devices*. Apress L. P., Berkeley, CA. Dostupné z: <https://doi.org/9781484263365>.
- [45] *ESP32 Development Board - DEVKIT V1*, 2025. Online. Dostupné z: <https://grobotronics.com/esp32-development-board-devkit-v1.html?sl=en>. [cit. 2025-05-03].
- [46] *ESP32-S3-DevKitC-1 v1.1*, 2025. Online. Dostupné z: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitc-1/user_guide.html. [cit. 2025-05-03].
- [47] *What is Arduino?*, 2018. Online. Dostupné z: <https://www.arduino.cc/en/Guide/Introduction>. [cit. 2025-03-16].
- [48] *Arduino Uno Rev3*, 2025. Online. Dostupné z: <https://store-usa.arduino.cc/products/arduino-uno-rev3>. [cit. 2025-03-17].
- [49] *The Epic Story of the Raspberry Pi*, 2025. Online. Dostupné z: <https://raspberrytips.com/raspberry-pi-history/>. [cit. 2025-03-17].
- [50] *Raspberry Pi 4 Model B Specifications*, 2025. Online. Dostupné z: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>. [cit. 2025-03-17].
- [51] *Raspberry Pi Pico*, 2021. Online. Dostupné z: <https://datasheets.raspberrypi.com/pico/pico-product-brief.pdf>. [cit. 2025-03-17].
- [52] *Top 40 Raspberry Pi 4 Projects That You Must Try in 2025*, 2020. Online. Dostupné z: <https://www.seeedstudio.com/blog/2019/09/29/top-20-best-raspberry-pi-4-projects-that-you-must-try-now>. [cit. 2025-03-17].
- [53] *What Are RF Modules and How Do They Work?*, 2024. Online. Dostupné z: <https://www.rficy.com/NewsDetail.aspx?ID=197>. [cit. 2025-02-07].
- [54] *Working of an RF Module*, 2025. Online. Dostupné z: <https://www.millenniumsemi.com/blog/working-of-an-rf-module/>. [cit. 2025-02-07].

- [55] *What is modulation?*, 2025. Online. Dostupné z: <https://www.techtarget.com/searchnetworking/definition/modulation>. [cit. 2025-02-07].
- [56] SOLUTIONS, Cadence PCB, 2025. *RF Modulation Types*. Online. Dostupné z: <https://resources.pcb.cadence.com/blog/2023-rf-modulation-types>. [cit. 2025-02-08].
- [57] ANALYSIS, Cadence System, 2025. *Types of Digital Modulation*. Online. Dostupné z: <https://resources.system-analysis.cadence.com/blog/msa2021-types-of-digital-modulation>. [cit. 2025-02-08].
- [58] *433 MHz RF Receiver Module*, 2018. Online. Dostupné z: <https://components101.com/modules/433-mhz-rf-receiver-module>. [cit. 2025-02-09].
- [59] *Complete Guide for RF 433MHz Transmitter/Receiver Module With Arduino*, 2019. Online. Dostupné z: <https://randomnerdtutorials.com/rf-433mhz-transmitter-receiver-module-with-arduino/>. [cit. 2025-02-09].
- [60] GERLÉE, Kasia, 2023. *16 Types of Sensors Used in IoT*. Online. Dostupné z: <https://freeway.com/16-types-of-sensors-used-in-iot/>. [cit. 2025-05-06].
- [61] *What is an IoT Temperature and Humidity Sensor, and how does it work?*, 2023. Online. Dostupné z: <https://ubibotus.com/blog/what-is-an-iot-temperature-and-humidity-sensor-and-how-does-it-work/>. [cit. 2025-05-06].
- [62] *A Brief on DHT11 Sensor*, 2025. Online. Dostupné z: <https://www.elprocus.com/a-brief-on-dht11-sensor/>. [cit. 2025-05-06].
- [63] *HTU21D Humidity Sensor Hookup Guide*, 2025. Online. Dostupné z: <https://learn.sparkfun.com/tutorials/htu21d-humidity-sensor-hookup-guide#htu21d-overview>. [cit. 2025-05-06].
- [64] WRITER, Staff, 2016. *Smart home sensors*. Online. Dostupné z: <https://www.ibm.com/think/topics/smart-home-sensors>. [cit. 2025-05-06].
- [65] *Principles of UI/UX Design*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/principles-of-ui-ux-design/>. [cit. 2025-03-18].

- [66] *Responsive Web Design*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/responsive-web-design/>. [cit. 2025-03-18].
- [67] VRANKOVÁ, Michaela, 2025. *Co je to WCAG*. Online. Dostupné z: <https://pristupne-stranky.cz/co-je-to-wcag/>. [cit. 2025-05-07].
- [68] VRANKOVÁ, Michaela, 2025. *Požadavky na přístupnost*. Online. Dostupné z: <https://pristupne-stranky.cz/zakon-a-standardy/>. [cit. 2025-05-07].
- [69] *Vnímatelnost*, 2025. Online. Dostupné z: <https://pristupne-stranky.cz/1-vnimatelnost/>. [cit. 2025-05-07].
- [70] *Kritérium 1.1.1 Netextový obsah – úroveň A*, 2025. Online. Dostupné z: <https://pristupne-stranky.cz/1-vnimatelnost/pravidlo-1-1-textove-alternativy/1-1-1-netextovy-obsah/>. [cit. 2025-05-07].
- [71] *Ovládatelnost*, 2025. Online. Dostupné z: <https://pristupne-stranky.cz/2-ovladatelnost/>. [cit. 2025-05-07].
- [72] *Srozumitelnost*, 2025. Online. Dostupné z: <https://pristupne-stranky.cz/3-srozumitelnost/>. [cit. 2025-05-07].
- [73] *Stabilita*, 2025. Online. Dostupné z: <https://pristupne-stranky.cz/4-stabilita/>. [cit. 2025-05-07].
- [74] *List of Front-End Technologies*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/list-of-front-end-technologies/>. [cit. 2025-03-23].
- [75] *React Introduction*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/reactjs-introduction/>. [cit. 2025-03-23].
- [76] *Angular vs React vs Vue: Core Differences*, 2023. Online. Dostupné z: <https://www.browserstack.com/guide/angular-vs-react-vs-vue>. [cit. 2025-03-23].
- [77] *TypeScript Introduction*, 2025. Online. Dostupné z: https://www.w3schools.com/typescript/typescript_intro.php. [cit. 2025-03-23].

- [78] *What Is Angular? A Guide to the Angular Framework*, 2025. Online. Dostupné z: <https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular>. [cit. 2025-03-23].
- [79] *Difference between ReactJS and Vue.js*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/difference-between-reactjs-and-vue-js/>. [cit. 2025-03-23].
- [80] *Express.js*, 2025. Online. Dostupné z: <https://www.geeksforgeeks.org/express-js/>. [cit. 2025-05-07].
- [81] KHAN, Shahzaib, 2023. *Building a RESTful API with Express.js: A Beginner's Guide*. Online. Dostupné z: <https://medium.com/@skhans/building-a-restful-api-with-express-js-a-beginners-guide-dcb1a1e3520d>. [cit. 2025-05-07].
- [82] *What Is a JWT and How It Works*, 2024. Online. Dostupné z: <https://www.descope.com/learn/post/jwt>. [cit. 2025-05-07].
- [83] *Digitální teploměr IN/OUT, s vlhkoměrem, KT-908*, 2025. Online. Dostupné z: https://www.zarovkarna.cz/product/elektronika-pro-domacnost/teplomery-a-vlhkomery/digitalni-teplomer-in_out_-s-vlhkomerem_/384. [cit. 2025-05-022].

ZOZNAM OBRÁZKOV

Obrázok 1.1	Ukážka jednoduchej MQTT komunikácie	17
Obrázok 3.1	Von Neumannova architektúra	23
Obrázok 3.2	Harvardská architektúra	23
Obrázok 3.3	Vývojová doska ESP8266	24
Obrázok 3.4	Vývojová doska ESP32	25
Obrázok 3.5	ATTiny85 - 8-bitový mikrokontrolér	26
Obrázok 3.6	Ukážka Arduino IDE	27
Obrázok 3.7	Vývojová doska ESP32 DevKit v1	29
Obrázok 3.8	Vývojová doska ESP32-S3-DevKitC-1	29
Obrázok 3.9	Vývojová doska Arduino Uno	30
Obrázok 3.10	Vývojová doska Raspberry Pi 4 model B	30
Obrázok 3.11	Vývojová doska Raspberry Pi Pico	30
Obrázok 4.1	Rozdelenie modulácie signálu	32
Obrázok 4.2	433 MHz prijímač a vysielateľ.....	33
Obrázok 6.1	Architektúra systému	41
Obrázok 6.2	ERD diagram	45
Obrázok 7.1	Schéma zapojenia systému na ovládanie svetla	47
Obrázok 7.2	SL-LOGIC senzor - komunikačná jednotka	51
Obrázok 7.3	SL-LOGIC senzor - meracia jednotka	51
Obrázok 8.1	Aplikácia EmbeHouse	58
Obrázok 8.2	Zobrazenie historických údajov	58
Obrázok 9.1	Porovnanie meraní teploty a vlhkosti	60
Obrázok 9.2	Testovanie dosahu 433 MHz rádiového modulu	61

ZOZNAM TABULIEK

Tabulka 6.1	Funkčné požiadavky systému	40
Tabulka 6.2	Požiadavky zamerané na vlastnosti systému.....	40
Tabulka 6.3	Databázový model	45
Tabulka 6.4	Vzťahy medzi tabulkami.....	45
Tabulka 7.1	Použité piny ESP32 modulov	47

ZOZNAM VÝPISOV

Výpis 7.1	Inicializácia premenných pre WiFi a MQTT	48
Výpis 7.2	Inicializácia rádiového modulu	49
Výpis 7.3	Pripojenie na WiFi sieť	49
Výpis 7.4	Pripojenie na MQTT broker	49
Výpis 7.5	Odosielanie hodnôt cez 433 MHz vysielateľ	49
Výpis 7.6	Prijímanie hodnôt zo 433 MHz prijímača	50
Výpis 8.1	Inicializácia Node.js servera	53
Výpis 8.2	Spracovanie MQTT správ	53
Výpis 8.3	Spracovanie údajov zo senzorov	54
Výpis 8.4	Odosielanie príkazov na ESP32	55
Výpis 8.5	Konfigurácia databázy	55
Výpis 8.6	Ukážka vytvorenia tabuľky	56
Výpis 8.7	Základná štruktúra aplikácie v App.tsx	56
Výpis 8.8	Generovanie JWT tokenu	59

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

ADC	Analog to digital converter
ALU	Arithmetic logic unit
API	Application programming interface
ARM	Advanced RISC Machine
ASK	Amplitude-shift keying
BLE	Bluetooth low energy
CAN	Controller area network
CISC	Complex instruction set computer
CO	oxid uhoľnatý
CPU	Central processing unit
CRUD	Create, read, update, delete
CoAP	Constrained Application Protocol
DAC	Digital to analog converter
DC	Direct current
FSK	Frequency-shift keying
GHz	Gigahertz
GPIO	General-purpose input/output
HTML	Hypertext Markup Language
HTTP	Hypertext transfer protocol
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
I ₂ C	Inter-Integrated Circuit
I ² S	Inter-Integrated Circuit Sound
IDE	Integrated Development Environment
IoT	Internet of things
JTAG	Joint Test Action Group
JWT	JSON Web Token
KB	Kilobyte
Kbps	Kilobits per second
LCD	Liquid crystal display
LED	Light-emitting diode

LoRa	Long range
MB	Megabyte
MHz	Megahertz
MQTT	Message Queuing Telemetry Transport
MVC	Model-view-controller
ms	Milisekunda
OTG	On-the-go
PCB	Printed circuit board
PIO	Programmable input/output
PIR	Passive infrared sensor
PSK	Phase-shift keying
RAM	Random-access memory
REST	Representational state transfer
RF	Radio-frequency
RFID	Radio-frequency identification
RGB	Red-green-blue
RISC	Reduced instruction set computer
ROM	Read-only memory
SDK	Software development kit
SPA	Single-page application
SPI	Serial peripheral interface
SQL	Structured Query Language
SRAM	Static random-access memory
TLS/SSL	Transport Layer Security / Secure Sockets Layer
UART	Universal asynchronous receiver / transmitter
UI	User interface
USB	Universal serial bus
UX	User experience
VS Code	Visual Studio Code
WCAG	Web Content Accessibility Guidelines
Wi-Fi	Wireless Fidelity

ZOZNAM PRÍLOH

PRÍLOHA P I: Zoznam súborov v zip archíve

PRÍLOHA P I: ZOZNAM SÚBOROV V ZIP ARCHÍVE

Zoznam súborov, ktoré sú súčasťou zip archívu s bakalárskou prácou, vrátane všetkých potrebných zdrojových kódov, dokumentácie a ďalších materiálov:

- Bakalársku prácu vo formáte PDF/A
- Zoznam použitých komponentov
- Zoznam použitých knižníc
- Návod na použitie
- Docker Compose konfiguračný súbor pre jednoduché nasadenie systému
- Kompletne zdrojové kódy serverovej časti
- Kompletne zdrojové kódy používateľskej časti
- Zdrojové kódy pre ESP32 moduly
- Obrázky a videozáznamy demonštrujúce funkčnosť systému
- Dodatočné materiály