Využití fuzzy logiky na mikropočítači

Jozef Varhaník

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jozef Varhaník**
Osobní číslo:	**A20106**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Využití fuzzy logiky na mikropočítači**
Téma práce anglicky:	**Fuzzy Logic Usage in a Microcontroller**

Zásady pro vypracování

1. Vypracujte literární rešerši na dané téma.
2. Vyberte vhodné knihovny pro implementaci fuzzy logiky na mikropočítačích.
3. Ověřte a popište vlastnosti a způsob použití vybraných knihoven.
4. Pro zvolenou knihovnu a zvolený mikropočítač vytvořte ukázkovou aplikaci.
5. Zpracujte dokumentaci popisující implementaci a použití vytvořené aplikace a zhodnoťte její funkčnost.

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

Seznam doporučené literatury:

1. DRIANKOV, Dimiter. An Introduction to Fuzzy Control. Springer, 1996, 332 s. ISBN 3642082343.
2. IBRAHIM, Ahmad. Fuzzy Logic for Embedded Systems Applications. Vyd. 1. Newnes, 2003. ISBN 9780080469904.
3. PEDRYCZ, Witold. Fuzzy control and fuzzy systems. 2nd, extended ed. Somerset: Research Studies Press, 1993. ISBN 978-0-86380-131-0.
4. PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300--110-1.
5. WHITE, Elecia. Making embedded systems. Sebastopol: O´Reilly, c2012. ISBN 9781449302146.

Vedoucí bakalářské práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D.
děkan

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

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že 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;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že 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 samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

Jozef Varhaník

ABSTRAKT

Bakalárska práca sa zaoberá skúmaním fuzzy logiky na mikropočítačoch. V prvej časti sa práca venuje teoretickému popisu fuzzy logiky obecne a následne na mikropočítačoch. Taktiež je uvedených niekoľko príkladov využitia fuzzy logiky a programovacie jazyky, v ktorých sa fuzzy logika implementuje. V praktickej časti sa práca zaoberá výberom vhodných knižníc na implementáciu fuzzy logiky na mikropočítačoch a v iných odvetviach. Pre knižnice, používané v mikropočítačoch je vytvorený ukážkový kód, ktorý overuje vlastnosti knižníc. Ďalej sa praktická časť práce sa zaoberá vytvorením ukážkovej aplikácie fuzzy logiky na mikropočítači. V rámci ukážkovej aplikácie je popísaný výber vhodnej knižnice, výber vývojového prostredia, vhodného mikropočítača a zároveň aj obsahuje dokumentáciu k aplikácií.

Klíčová slova: Fuzzy logika, Mikropočítač, Aplikácie fuzzy logiky

ABSTRACT

The bachelor thesis deals with the investigation of fuzzy logic on microcontrollers. In the first part, the thesis deals with the theoretical description of fuzzy logic in general and then on microcontrollers. Several examples of fuzzy logic applications and programming languages in which fuzzy logic is implemented are also presented. In the practical part, the thesis deals with the selection of suitable libraries for the implementation of fuzzy logic on microcontrollers and in other industries. For the libraries used on microcontrollers, a sample code is developed to verify the properties of the libraries. Further, the practical part of the thesis deals with the creation of a sample application of fuzzy logic on a microcontroller. The sample application describes the selection of a suitable library, the choice of a development environment, a suitable microcontroller and also includes documentation for the application.

Keywords: Fuzzy logic, Microcontroller, Applications of fuzzy logic

Poděkování

Ďakujem vedúcemu práce Ing. Janovi Dolinayovi, Ph. D. za odborné vedenie bakalárskej práce, za rady, pripomienky a čas, ktorý mi venoval.

Taktiež by som sa chcel poďakovať svojej rodine a blízkym, ktorí ma počas celej doby štúdia podporovali.

OBSAH

ÚVOD **9**
I TEORETICKÁ ČÁST **10**
1 ÚVOD DO FUZZY LOGIKY **11**
1.1 DEFINÍCIA FUZZY LOGIKY 11
1.2 FUZZY MNOŽINY 11
1.3 FUZZIFIKÁCIA A DEFUZZIFIKÁCIA 11
1.3.1 Fuzzifikácia 12
1.3.2 Defuzzifikácia 12
1.4 ZÁKLADNÉ OPERÁCIE FUZZY LOGIKY 13
1.4.1 Unárne a binárne operácie 13
1.4.2 Pravidlo súčtu a pravidlo súčinu 13
1.4.3 Pravidlo rozšírenia 14
1.4.4 Pravidlo implikácie 14
1.4.5 Pravidlo agregácie 14
1.4.6 Pravidlo výberu 14
1.5 HLAVNÉ MODELY FUZZY LOGIKY 14
1.5.1 Mamdaniho model 15
1.5.2 Sugenoho model 15
1.6 APLIKÁCIE FUZZY LOGIKY 16
1.7 POROVNANIE FUZZY LOGIKY S BINÁRNOU LOGIKOU 19
1.8 VÝHODY A NEVÝHODY FUZZY LOGIKY 20
1.8.1 Výhody fuzzy logiky 20
1.8.2 Nevýhody fuzzy logiky 20
2 FUZZY LOGIKA V RIADIACICH SYSTÉMOCH A MIKROPOČÍTAČOCH **22**
2.1 ZÁKLADNÉ KONCEPTY FUZZY LOGIKY V RIADIACEJ TECHNIKE 22
2.1.1 Štruktúra fuzzy riadiacich systémov 22
2.1.2 Fuzzy kontrolér a jeho komponenty 23
2.1.3 Kroky pri vytváraní fuzzy kontroléru 24
2.2 VÝHODY FUZZY LOGIKY V RIADIACEJ TECHNIKE 25
2.3 SÚVISLOSŤ APLIKÁCIE FUZZY LOGIKY V MIKROPOČÍTAČOCH S RIADIACOU TECHNIKOU 26
2.4 MIKROPOČÍTAČE A ICH VÝZNAM V MODERNEJ TECHNOLÓGIÍ 27
2.5 APLIKÁCIE FUZZY LOGIKY NA MIKROPOČÍTAČOCH 28
3 PROGRAMOVACIE JAZYKY FUZZY LOGIKY **31**

3.1 PYTHON 31
3.2 MATLAB 31
3.3 JAVA 31
3.4 C++ 31
3.5 R 32
3.6 C# 32
3.7 JAVASCRIPT 32
3.8 RUBY 32
3.9 LISP 32
3.10 OBLASTI POUŽITIA SPOMÍNANÝCH PROGRAMOVACÍCH JAZYKOV 33
3.10.1 Dátová analýza a strojové učenie 33
3.10.2 Inžinierske a vedecké aplikácie 33
3.10.3 Webové a mobilné aplikácie 33
3.10.4 Operačné systémy a grafické aplikácie 33
3.10.5 Podnikový softvér a aplikácie s vysokou úrovňou zložitosti 34
3.10.6 Symbolické a logické výpočty 34
3.10.7 Skriptovanie a rýchla prototypizácia 34
II PRAKTICKÁ ČÁST 35
4 KNIŽNICE NA IMPLEMENTÁCIU FUZZY LOGIKY 36
4.1 EFLL (EMBEDDED FUZZY LOGIC LIBRARY) 36
4.1.1 Popis základných objektov eFLL 36
4.1.2 Funkcie a vlastnosti eFLL 37
4.1.3 Overenie vlastností 38
4.2 FUZZYLITE 38
4.2.1 Funkcie a vlastnosti fuzzylite 38
4.2.2 Overenie vlastností 39
4.3 JFUZZYLITE 41
4.3.1 Funkcie a vlastnosti jFuzzylite 41
4.3.2 Overenie vlastností 41
4.4 SCIKIT-FUZZY 43
4.4.1 Funkcie a vlastnosti Scikit-fuzzy 43
4.5 FUZZY LOGIC TOOLBOX 44
4.5.1 Funkcie a vlastnosti Fuzzy logic toolbox 44
4.6 AFORGE.NET 44
4.6.1 Funkcie a vlastnosti Aforge.net 45
4.7 TABUĽKA POROVNANIA VLASTNOSTÍ UVEDENÝCH KNIŽNÍC 46
5 PRÍPRAVA NA VYTVORENIE UKÁŽKOVEJ APLIKÁCIE 47
5.1 VÝBER KNIŽNICE 47
5.2 VÝBER MIKROPOČÍTAČA 48
5.3 VÝBER VÝVOJOVÉHO PROSTREDIA 49
5.4 ZHRNUTIE VÝBERU 50
6 UKÁŽKOVOVÁ APLIKÁCIA 51

6.1 Štruktúra aplikácie 51
6.2 Súpis komponentov 51
6.3 Schéma zapojenia 52
6.4 Fyzické zapojenie 52
6.5 Zdrojový kód 53
6.5.1 Inicializácia 54
6.5.2 Inicializácia LED diód, relé, rýchlosti komunikácie na sériovom porte a spustenie senzoru DHT22 55
6.5.3 Inicializácia fuzzy množín 55
6.5.4 Nastavenie fuzzy pravidiel 57
6.5.5 Spracovanie teplotných dát a rozhodovanie 59
6.6 Testovanie a hodnotenie funkcionalít 60
6.7 Porovnanie s použitím binárnej logiky 63
6.7.1 Graf binárnej logiky 63
6.7.2 Graf fuzzy logiky 64
6.7.3 Výhody použitia fuzzy logiky v ukážkovej aplikácií 65
6.8 Zhodnotenie a možnosti rozšírenia 65
ZÁVĚR 66
SEZNAM POUŽITÉ LITERATURY 67
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 72
SEZNAM OBRÁZKŮ 73
SEZNAM TABULEK 74
SEZNAM PŘÍLOH 75

ÚVOD

V dnešnej dobe sú mikropočítače neustále využívané v mnohých oblastiach, od domácich spotrebičov, zariadení pre medicínu až po priemyselné aplikácie. S narastajúcim množstvom dát a náročnosťou úloh, ktoré musia mikropočítače vykonávať, je nevyhnutné rozvíjať nové metódy a technológie, ktoré im pomôžu pri spracovaní informácií. Jednou z týchto metód je aj fuzzy logika.

Fuzzy logika je koncept, ktorý sa využíva v rôznych oblastiach, od technických disciplín po sociálne vedy. Má významné uplatnenie v situáciách, kde je potrebné modelovať neurčitosti alebo aproximácie, ktoré sú typické pre ľudské myslenie.

Vzhľadom na dôležitosť mikropočítačov v dnešnej dobe, je využitie fuzzy logiky na mikropočítači stále relevantné a zaujímavé pre výskumníkov a inžinierov.

Využitie fuzzy logiky na mikropočítači má množstvo výhod. Jednou z nich je schopnosť rýchleho a efektívneho riešenia problémov, ktoré by boli inak príliš náročné alebo nemožné riešiť klasickými metódami. Ďalšou výhodou je schopnosť adaptácie a učenia sa z dát, čo umožňuje vytvárať inteligentné systémy s vysokou presnosťou a spoľahlivosťou.

Táto bakalárska práca sa zameriava na využitie fuzzy logiky na mikropočítači a jej aplikácie v rôznych oblastiach. Bude skúmať, ako je možné využiť fuzzy logiku na riešenie konkrétnych problémov a ako ju implementovať na mikropočítači. Zároveň sa bude venovať v praktickej časti návrhu ukážkovej aplikácie fuzzy logiky na mikropočítači. Okrem už spomenutých vecí sa bude práca venovať popisovaniu vhodných nástrojov a knižníc spojených s využitím fuzzy logiky na mikropočítači.

I. TEORETICKÁ ČÁST

1 ÚVOD DO FUZZY LOGIKY

V úvodnej kapitole sa práca zameriava na definíciu fuzzy logiky a vymedzení základných pojmov, ako aj operácií a pravidiel spojených s fuzzy logikou. Kapitola teda obsahuje základné informácie týkajúce sa fuzzy logiky a slúži na to, aby sa čitateľ zoznámil so základmi fuzzy logiky a lepšie tak pochopil nasledujúcim kapitolám teoretickej časti.

1.1 Definícia fuzzy logiky

Fuzzy je slovo pochádzajúce z Angličtiny a má doslovný preklad ako mlhavý, nejasný, neostrý alebo neurčitý. Definícia fuzzy logiky sa teda skrýva v tomto doslovnom preklade [8].

Fuzzy logika, pôvodne navrhnutá Lotfi A. Zadehom v roku 1965 [1], predstavuje rozšírenie klasickej binárnej logiky, ktorá je založená na prístupe "všetko alebo nič". Fuzzy logika sa od klasickej logiky odlišuje tým, že umožňuje zohľadniť neurčitosť a nepresnosť v rámci procesu rozhodovania a popisu javov.

Fuzzy logika sa často využíva v oblastiach, kde je potrebné pracovať s nepresnými a nejednoznačnými údajmi alebo v situáciách, kde sú klasické prístupy nedostatočné alebo nevhodné. Medzi bežné aplikácie fuzzy logiky patria riadenie systémov, predikcia a modelovanie, expertné systémy, rozhodovanie za neurčitých podmienok a spracovanie prirodzeného jazyka [4].

1.2 Fuzzy množiny

Kľúčovým pojmom vo fuzzy logike sú fuzzy množiny, ktoré reprezentujú hodnoty v neurčitom a kontinuálnom spektre [2]. Zatiaľ čo v klasickej logike môže byť prvok množiny buď úplne prítomný (pravda) alebo úplne neprítomný (nepravda), fuzzy množiny umožňujú priradenie medzistupňov príslušnosti medzi 0 a 1. Hodnota príslušnosti k fuzzy množine vyjadruje stupeň, do akej je prvok členom danej množiny [3].

1.3 Fuzzifikácia a defuzzifikácia

Fuzzy logika pracuje s hodnotami, ktoré sú neurčité a nejednoznačné, a preto sa pri jej aplikácii využívajú procesy fuzzifikácie a defuzzifikácie. Fuzzifikácia je proces, ktorý prevádza konkrétne hodnoty (čísla) na stupne príslušnosti k jednotlivým fuzzy množinám (fuzzy hodnoty), zatiaľ čo defuzzifikácia je proces, ktorý prevádza fuzzy hodnoty späť na konkrétne hodnoty (čísla) [2][3]. Táto podkapitola opisuje procesy fuzzifikácie a defuzzifikácie.

1.3.1 Fuzzifikácia

Fuzzifikácia je prvý krok vo fuzzy riadiacom systéme alebo iných aplikáciách fuzzy logiky. Jeho účelom je prevod konkrétnych vstupných hodnôt na stupne príslušnosti k jednotlivým fuzzy množinám, ktoré reprezentujú neurčitosť alebo nejednoznačnosť týchto hodnôt. Fuzzifikácia sa uskutočňuje pomocou takzvaných fuzzifikačných funkcií, ktoré mapujú vstupné hodnoty na stupne príslušnosti ku fuzzy množinám [2][3].

Fuzzifikačné funkcie môžu mať rôzne formy, ako napríklad trojuholníkové, trapezoidálne alebo gaussiánske funkcie. Voľba vhodnej fuzzifikačnej funkcie závisí od konkrétnej aplikácie a od charakteristiky vstupných dát. Po fuzzifikácii vstupných hodnôt sú tieto hodnoty reprezentované fuzzy hodnotami (premennými), s ktorými je možné pracovať v rámci fuzzy logiky [2][3].

1.3.2 Defuzzifikácia

Defuzzifikácia je proces, ktorý sa uskutočňuje po aplikácii fuzzy pravidiel na fuzzy hodnoty, ktoré boli získané fuzzifikáciou vstupných hodnôt. Cieľom defuzzifikácie je získať konkrétnu hodnotu (číslo) z fuzzy hodnôt, ktorá môže byť použitá ako výstup fuzzy riadiaceho systému alebo iných aplikácií fuzzy logiky [2][3].

Defuzzifikácia sa uskutočňuje pomocou rôznych metód, ako napríklad metóda stredu váhy (center of gravity), metóda maximálnej príslušnosti (maximum membership) alebo metóda bisektory (bisector method) [2][3]. Voľba vhodnej defuzzifikačnej metódy závisí od kon krétnej aplikácie a požiadaviek na výstupnú hodnotu.

Metóda stredu váhy (center of gravity) je jednou z najpopulárnejších a najpoužívanejších metód defuzzifikácie. Táto metóda vypočíta výstupnú hodnotu ako vážený priemer hodnôt vo fuzzy hodnote, pričom váhy sú určené stupňami príslušnosti k jednotlivým fuzzy množinám [2][3].

Metóda maximálnej príslušnosti (maximum membership) je založená na výbere hodnoty, ktorá má najvyšší stupeň príslušnosti k fuzzy množine. Táto metóda je jednoduchšia ako metóda stredu váhy, ale môže poskytovať menej presné výstupy [2][3].

Metóda bisektor (bisector method) vypočíta výstupnú hodnotu ako bod, v ktorom je plocha pod funkciou príslušnosti rozdelená na dve rovnaké časti. Táto metóda sa často používa, keď je dôležité získať kompromis medzi presnosťou a zložitosťou výpočtu [2][3].

Po defuzzifikácii sú získané konkrétne výstupné hodnoty, ktoré môžu byť použité ako výsledky fuzzy riadiaceho systému alebo iných aplikácií fuzzy logiky. Procesy fuzzifikácie a defuzzifikácie sú kľúčovými súčasťami fungovania fuzzy logiky a umožňujú efektívne riešenie problémov s neurčitosťou a nejednoznačnosťou v rôznych oblastiach [2][3].

1.4 Základné operácie fuzzy logiky

V rámci fuzzy logiky sú základné operácie rozdelené na unárne a binárne operácie. Základné pravidlá fuzzy logiky zahŕňajú pravidlo súčtu a pravidlo súčinu. Ďalej ešte poznáme pravidlo výberu, rozšírenia, implikácie a agregácie [2].

Základné operácie a pravidlá fuzzy logiky poskytujú teoretický základ pre modelovanie a analýzu neurčitosti a nedeterminizmu v reálnych systémoch. Využitie fuzzy logiky v inžinierskych aplikáciách umožňuje lepšie pochopiť a riadiť tieto systémy, čo vedie k ich efektívnejšiemu a spoľahlivejšiemu fungovaniu.

1.4.1 Unárne a binárne operácie

Unárne operácie sa vzťahujú na jednu fuzzy množinu a zahŕňajú komplement, koncentráciu a rozptýlenie fuzzy množín. Komplement fuzzy množiny A je definovaný ako množina, ktorej príslušnosť k elementom je založená na nepríslušnosti k A [2]. Koncentrácia a rozptýlenie fuzzy množín sú operácie, ktoré modifikujú príslušnosť elementov v množine tak, že zvyšujú alebo znižujú hodnoty príslušnosti [3].

Binárne operácie sa vzťahujú na dve fuzzy množiny a zahŕňajú operácie ako sú prienik, zjednotenie a rôzne implikácie [2]. Prienik (t-norma) a zjednotenie fuzzy množín sú založené na hodnotách príslušnosti elementov a ich kombinácii, pričom prienik predstavuje množinu, v ktorej sú spoločné prvky oboch množín, a zjednotenie predstavuje množinu obsahujúcu všetky prvky oboch množín [3].

1.4.2 Pravidlo súčtu a pravidlo súčinu

Pravidlo súčtu, tiež nazývané pravidlo maximálnej príslušnosti, je založené na princípe, že hodnota príslušnosti elementu k zjednotenej fuzzy množine je maximálna hodnota príslušnosti tohto elementu k jednotlivým množinám [2]. Na druhej strane, pravidlo súčinu, tiež nazývané pravidlo minimálnej príslušnosti, je založené na princípe, že hodnota príslušnosti

elementu k prieniku fuzzy množín je minimálna hodnota príslušnosti tohto elementu k jednotlivým množinám [3].

1.4.3 Pravidlo rozšírenia

Pravidlo rozšírenia sa používa na transformáciu fuzzy množín pomocou matematických operácií, ako sú napríklad súčet, rozdiel, násobenie alebo delenie. Toto pravidlo umožňuje aplikovať tieto operácie na fuzzy množiny a získať nové fuzzy množiny [3].

1.4.4 Pravidlo implikácie

Pravidlo implikácie je kľúčovou súčasťou fuzzy pravidiel a slúži na vytvorenie nových fuzzy množín na základe existujúcich množín a ich vzťahu. Existuje niekoľko typov implikácií, ako napríklad Mamdaniho implikácia, Larsenova implikácia alebo Zadehova implikácia [3].

1.4.5 Pravidlo agregácie

Pravidlo agregácie sa používa na kombináciu viacerých fuzzy množín do jednej. Pomocou pravidla agregácie je možné zlúčiť výsledky viacerých pravidiel alebo operácií do jednej fuzzy množiny, ktorá je potom ďalej spracovávaná v fuzzy riadiacom systéme [3].

1.4.6 Pravidlo výberu

Poznáme dva druhy a to pravidlo α-rezov a pravidlo β-rezov.

Pravidlo α-rezov:

Toto pravidlo slúži na extrahovanie konkrétnej časti fuzzy množiny na základe úrovne príslušnosti α. Pomocou α-rezov sa z fuzzy množiny vytvára nová množina, ktorá obsahuje iba prvky s hodnotou príslušnosti rovnou alebo väčšou ako α [2].

Pravidlo β-rezov:

Podobne ako pravidlo α-rezov, aj toto pravidlo slúži na extrahovanie časti fuzzy množiny. Rozdiel je v tom, že β-rezy extrahujú prvky s hodnotou príslušnosti rovnou alebo menšou ako β [2].

1.5 Hlavné modely fuzzy logiky

V tejto podkapitole sa práca zameriava na hlavné modely a typy fuzzy logiky, ktoré sa používajú v praxi. Hlavnými modelmi fuzzy logiky sú Mamdaniho model a Sugenoho model.

Tieto modely predstavujú základ pre fuzzy riadiace systémy, ktoré sa využívajú v mnohých inžinierskych aplikáciách.

1.5.1 Mamdaniho model

Mamdaniho model, tiež známy ako Max-Min model, bol navrhnutý v roku 1975 Ebrahimom H. Mamdanim a Srinivasom Assilianom [5]. Tento model je založený na fuzzy pravidlách, ktoré sú formulované v jazyku prirodzenej logiky. Mamdaniho model pozostáva z troch častí: fuzzifikácie, inferenčného mechanizmu a defuzzifikácie [3].

- Fuzzifikácia: Fuzzifikácia je proces premeny reálnych vstupných hodnôt na fuzzy množiny, ktoré reprezentujú mieru príslušnosti týchto hodnôt k jednotlivým lingvistickým hodnotám.
- Inferenčný mechanizmus: Inferenčný mechanizmus zahŕňa aplikáciu fuzzy pravidiel na fuzzifikované vstupy. Výsledkom tohto procesu sú nové fuzzy množiny, ktoré predstavujú výstup systému v podobe fuzzy hodnôt.
- Defuzzifikácia: Defuzzifikácia je proces premeny fuzzy výstupov na reálne hodnoty. V rámci Mamdaniho modelu sa bežne používa metóda centroidu alebo váženého priemeru na výpočet defuzzifikovaných hodnôt [3].

1.5.2 Sugenoho model

Sugenoho model, tiež známy ako ANFIS (Adaptive Neuro-Fuzzy Inference System) alebo TSK (Takagi-Sugeno-Kang) model, bol navrhnutý v roku 1985 Michio Sugeno a jeho kolegami [6]. Sugenoho model sa podobne ako Mamdaniho model zameriava na fuzzy riadiace systémy, avšak s niekoľkými zásadnými rozdielmi [3].

- Fuzzifikácia: Proces fuzzifikácie v Sugenohovom modeli je rovnaký ako v Mamdaniho modeli, teda premena reálnych vstupných hodnôt na fuzzy množiny.
- Inferenčný mechanizmus: Inferenčný mechanizmus v Sugenohovom modeli je založený na aplikácii fuzzy pravidiel, ktoré sú definované ako lineárne funkcie výstupných premenných. Výsledkom inferenčného mechanizmu sú fuzzy množiny s lineárnymi funkciami, ktoré predstavujú výstup systému.

- Defuzzifikácia: V Sugenoho modeli sa používa metóda váženého priemeru na premenu fuzzy výstupov na reálne hodnoty. Táto metóda zohľadňuje hodnoty príslušnosti jednotlivých výstupných množín a kombinuje ich s lineárnymi funkciami, ktoré boli získané inferenčným mechanizmom. Defuzzifikované hodnoty sú získané ako vážený priemer týchto lineárnych funkcií [3].

Vo všeobecnosti je hlavným rozdielom medzi Mamdaniho a Sugenoho modelom spôsob, akým sú definované a aplikované fuzzy pravidlá. Mamdaniho model pracuje s lingvistickými pravidlami, ktoré sú formulované v jazyku prirodzenej logiky, zatiaľ čo Sugenoho model používa lineárne funkcie na definovanie pravidiel. Výber medzi týmito dvoma modelmi závisí od konkrétnej aplikácie a požiadaviek na fuzzy riadiaci systém.

Okrem Mamdaniho a Sugenoho modelov existuje mnoho ďalších variantov a rozšírení fuzzy logiky, ako napríklad intervalová fuzzy logika, typ-2 fuzzy logika, vážená fuzzy logika a neurónové siete založené na fuzzy logike [7]. Tieto rozšírenia a varianty zohľadňujú špecifické požiadavky rôznych aplikácií a zlepšujú schopnosti fuzzy logiky v modelovaní a riadení komplexných a neurčitých systémov.

1.6 Aplikácie fuzzy logiky

Fuzzy logika sa využíva v mnohých oblastiach vedy a techniky, kde sa stretávame s neurčitosťou, nedeterminizmom a komplexnosťou. V tejto podkapitole sú predstavené niektoré konkrétne aplikácie fuzzy logiky v rôznych doménach, ako sú riadenie, predikcia a spracovanie obrazu.

- Riadenie: Fuzzy logika sa často používa v automatickom riadení, kde zohráva dôležitú úlohu pri modelovaní a riadení neurčitých a nelineárnych systémov. Fuzzy radiče sú implementované v rôznych priemyselných a spotrebných zariadeniach, ako sú klimatizačné systémy, práčky, fotoaparáty, automobily a roboty [4]. Tieto radiče využívajú fuzzy pravidlá na adaptívne riadenie systémov, ktoré zohľadňujú neurčitosť a zmeny v prostredí a parametroch systému.
- Predikcia: Fuzzy logika sa tiež využíva pri prognózovaní a predpovedaní časových radov alebo vzorov v dátach. Fuzzy predikčné modely sú založené na fuzzy pravidlách, ktoré popisujú vzťahy medzi premennými a cieľovou hodnotou. Tieto modely sa používajú v rôznych oblastiach, ako sú ekonomika, meteorológia, energetika a

doprava, na predpovedanie dopytu, cien, znečistenia, počasia a iných premenných [4].

- Spracovanie obrazu: Fuzzy logika sa tiež uplatňuje v oblasti spracovania obrazu, kde sa používa na detekciu hrán, segmentáciu obrazu, šumovú redukciu a iné úlohy. Fuzzy pravidlá a operácie sú použité na zohľadnenie neurčitosti a variabilnosti v intenzitách, textúrach a farbách obrazov. Fuzzy metódy sa často kombinujú s inými technikami, ako sú neurónové siete alebo genetické algoritmy, na vytvorenie robustných a adaptívnych systémov spracovania obrazu [4].

Okrem uvedených aplikácií sa fuzzy logika používa aj v mnohých ďalších oblastiach, ako sú napríklad medicína, geológia, jazykoveda, doprava a environmentálne vedy. Fuzzy logika prináša výhody pri modelovaní a riadení neurčitých a komplexných systémov a poskytuje flexibilné, intuitívne a adaptívne riešenia pre rôzne problémy a úlohy.

- Medicína a zdravotníctvo: Fuzzy logika sa využíva v medicíne a zdravotníctve na diagnostiku, prognózu, monitorovanie a riadenie liečby pacientov. Fuzzy metódy zohľadňujú neurčitosť a variabilitu v medicínskych dátach a modeloch, čo umožňuje lepšie pochopenie a riešenie zdravotných problémov a zlepšenie kvality života pacientov [4].
- Telekomunikácie: Fuzzy logika sa využíva v telekomunikáciách na riadenie a optimalizáciu telekomunikačných systémov, ako sú nap ríklad siete, prenosové linky, zdroje a služby. Fuzzy metódy zohľadňujú neurčitosť a komplexnosť telekomunikačných dát a modelov, čo vedie k lepšiemu riadeniu a využitiu telekomunikačných zdrojov, zlepšeniu kvality služieb a znižovaniu nákladov [4].
- Meteorológia: Fuzzy logika sa tiež používa v meteorológii na predpovedanie počasia, analýzu klimatických zmien a riadenie meteorologických systémov. Fuzzy metódy zohľadňujú neurčitosť a komplexnosť meteorologických údajov a modelov, čo vedie k lepším predpovediam a efektívnejšiemu riadeniu meteorologických zdrojov a opatrení [4].
- Životné prostredie: Fuzzy logika sa uplatňuje v environmentálnom manažmente na modelovanie a analýzu environmentálnych problémov, ako sú znečistenie, zmena klímy, biodiverzita a zdroje. Fuzzy metódy zohľadňujú neurčitosť a komplexnosť

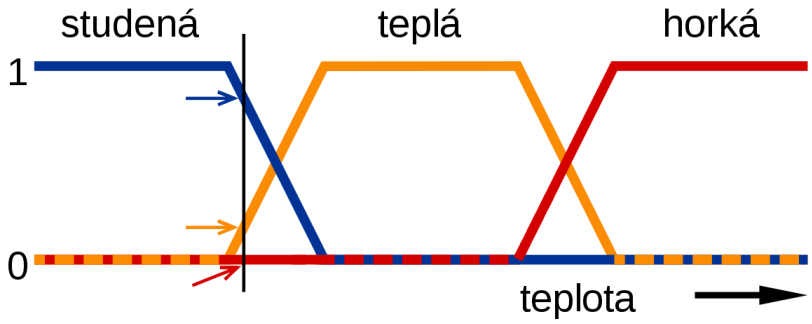
environmentálnych údajov a modelov, čo vedie k lepšiemu pochopeniu environmentálnych problémov a vývoji efektívnych opatrení na ochranu životného prostredia a udržateľný rozvoj [4].

Tieto príklady aplikácií fuzzy logiky ukazujú jej širokú uplatniteľnosť a potenciál v rôznych oblastiach. Fuzzy logika prináša inovatívne riešenia pre modelovanie a riadenie neurčitých a komplexných systémov a poskytuje flexibilné, intuitívne a adaptívne nástroje pre riešenie rôznych problémov a úloh v reálnom svete. V budúcnosti je možné očakávať ďalší rozvoj a inovácie v oblasti fuzzy logiky a jej aplikáciách, čo prispeje k lepšiemu pochopeniu a riadeniu neurčitosti a komplexnosti vo svete.

Nasledujúci obrázok [58] zobrazuje použitie fuzzy množín na definovanie studenej, teplej a horúcej teploty. Jedná sa konkrétne o zjednodušený graf funkcií príslušností k daným teplotám. Ak by bolo napríklad potrebné zistiť, aká teplota (studená, teplá, alebo horúca) je teplota v priesečníku kriviek teplej a horúcej teploty, tak výsledok by nebol iba 0, alebo 1, keďže na príslušnej hodnote teploty (na osi x) sa nachádzajú krivky všetkých troch teplôt. Studená teplota má hodnotu 0 a teplá s horúcou teplotou majú zhodne hodnotu približne 0,5. To znamená že príslušnosť teploty je 0 studená, 0,5 teplá a 0,5 horúca. Nájdené hodnoty príslušnosti by sa mohli zapísať v tvare $x = [a,b,c]$[52], kde x je reálna hodnota teploty na osi x, a je hodnota príslušnosti studenej teploty, b je hodnota príslušnosti teplej teploty a c je hodnota príslušnosti horúcej teploty. Po úprave by bol výsledok nasledovný:

- x_p = [0,0.5,0.5] => fuzzy hodnota.

Proces, pomocou ktorého je takáto fuzzy hodnota získaná je už spomenutá fuzzifikácia[52].



Obrázok 1. Využitie fuzzy množín pri definovaní teploty

1.7 Porovnanie fuzzy logiky s binárnou logikou

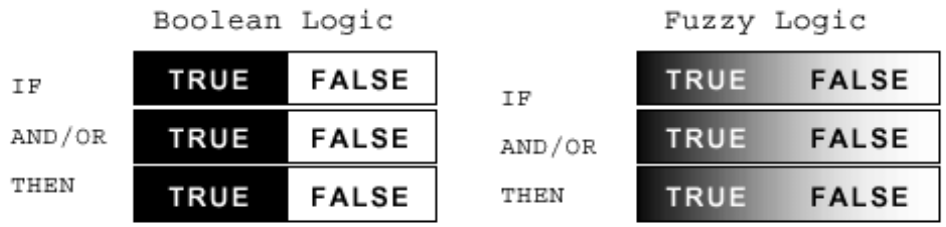
Fuzzy logika a binárna logika sú dva rozdielne prístupy k spracovaniu informácií. Binárna logika, ktorá je základom klasického Booleovského myslenia, pracuje s absolútnymi hodnotami pravdy alebo nepravdy (1 alebo 0 => True/False) a je základom pre digitálne počítače [2]. Na druhej strane, fuzzy logika, zavedená Zadehom v roku 1965 [1], je rozšírením binárnej logiky, ktorá umožňuje pracovať s neistotou a nejednoznačnosťou prostredníctvom členstva v množine s hodnotami v rozmedzí 0 až 1.

Fuzzy logika bola navrhnutá s cieľom modelovať neistotu a komplexitu, ktorá je často prítomná v reálnych situáciách a prirodzenom jazyku [3]. V praxi sa fuzzy logika používa na riešenie problémov, ktoré sú založené na neurčitosti a na ktoré nie je možné použiť tradičné matematické metódy [4]. Na druhej strane, binárna logika je ideálna pre aplikácie, ktoré vyžadujú absolútne presné výsledky.

Jedným z príkladov úspešného použitia fuzzy logiky je Mamdaniho a Assilianov experiment z roku 1975, ktorý ukázal, ako fuzzy logika dokáže modelovať lingvistické premenné a pravidlá [5]. Tento experiment položil základy pre vytvorenie fuzzy riadiacich systémov. Takagi a Sugeno v roku 1985 ukázali, ako fuzzy identifikácia systémov môže byť použitá na modelovanie a riadenie komplexných systémov [6]. Fuzzy logika sa odvtedy stala populárnou metódou v mnohých inžinierskych a vedeckých disciplínach.

Fuzzy logika sa vyznačuje svojou flexibilitou pri modelovaní neurčitosti a nejednoznačnosti, čo umožňuje lepšie napodobňovať ľudské rozhodovanie [7]. Na druhej strane, binárna logika je jednoduchšia a efektívnejšia pri spracovávaní veľkého množstva dát, ale nie je schopná zvládnuť problémy s neurčitosťou a nejednoznačnosťou.

Nasledujúci obrázok porovnania je prevzatý zo zdroja [59].



Obrázok 2. Porovnanie fuzzy logiky a binárnej logiky

1.8 Výhody a nevýhody fuzzy logiky

V porovnaní s klasickými metódami a algoritmami ponúka fuzzy logika viacero výhod, najmä pri riešení problémov s neurčitosťou a nejednoznačnosťou. Avšak, s týmto prístupom sú spojené aj určité nevýhody, ako je zložitosť a potenciálna strata presnosti. Voľba medzi fuzzy logikou a klasickými metódami závisí od konkrétnej aplikácie a požiadaviek na riešenie problému. V situáciách, kde je potrebné zvládnuť neurčitosť a nejednoznačnosť, môže fuzzy logika poskytnúť lepšie výsledky a väčšiu flexibilitu. Na druhej strane, pre aplikácie, ktoré vyžadujú vysokú presnosť a rýchle spracovanie dát, môžu byť klasické metódy a algoritmy vhodnejšou voľbou.

Je dôležité si uvedomiť, že fuzzy logika nie je náhradou za klasické metódy, ale skôr doplnkom, ktorý umožňuje riešiť špecifické problémy, ktoré sú pre klasické metódy ťažko zvládnuteľné. Preto je dôležité zvážiť výhody a nevýhody oboch prístupov pri riešení konkrétnych problémov a zvoliť ten, ktorý najlepšie vyhovuje danému problému a požiadavkám na riešenie.

1.8.1 Výhody fuzzy logiky

1. Schopnosť modelovať neurčitosť a nejednoznačnosť: Fuzzy logika umožňuje lepšie zvládnuť problémy s neurčitosťou a nejednoznačnosťou, ktoré sú často prítomné v reálnych situáciách a prirodzenom jazyku [1][3].
2. Flexibilita: Fuzzy logika je flexibilnejšia pri modelovaní a riadení komplexných systémov, čo umožňuje lepšie napodobňovanie ľudského rozhodovania [3][7].
3. Široká aplikovateľnosť: Fuzzy logika sa úspešne používa v mnohých inžinierskych a vedeckých disciplínach, vrátane riadiacich systémov, predikcie, klasifikácie a optimalizácie [4].
4. Robustnosť: Fuzzy systémy sú často robustné voči chybám alebo nepresnostiam vstupných dát [3].

1.8.2 Nevýhody fuzzy logiky

1. Zložitosť: Fuzzy logika môže byť zložitejšia v porovnaní s klasickými metódami a algoritmami, čo môže viesť k vyššej náročnosti na výpočtový výkon a pamäť [2][7].
2. Strata presnosti: Pri aproximácii môže dôjsť k strate presnosti v porovnaní s klasickými metódami [3].

3. Dostupnosť expertízy: Fuzzy logika vyžaduje špeciálnu expertízu pre návrh a implementáciu fuzzy systémov, čo môže byť ťažšie získať ako expertízu v klasických metódach [7].

2 FUZZY LOGIKA V RIADIACICH SYSTÉMOCH A MIKROPOČÍTAČOCH

Nasledujúca kapitola sa venuje využitiu fuzzy logiky v riadiacich systémoch a mikropočítačoch. Sú tu charakterizované základné koncepty fuzzy logiky v riadiacej technike, výhody použitia fuzzy logiky v riadiacej technike a súvislosť medzi využitím fuzzy logiky v riadiacich systémoch a mikropočítačoch. Ďalej sa v kapitole nachádza základná charakteristika mikropočítačov a ich význam v modernej technológií. Na záver je vymenovaných a aj popísaných päť príkladov využitia fuzzy logiky na mikropočítačoch z reálneho života.

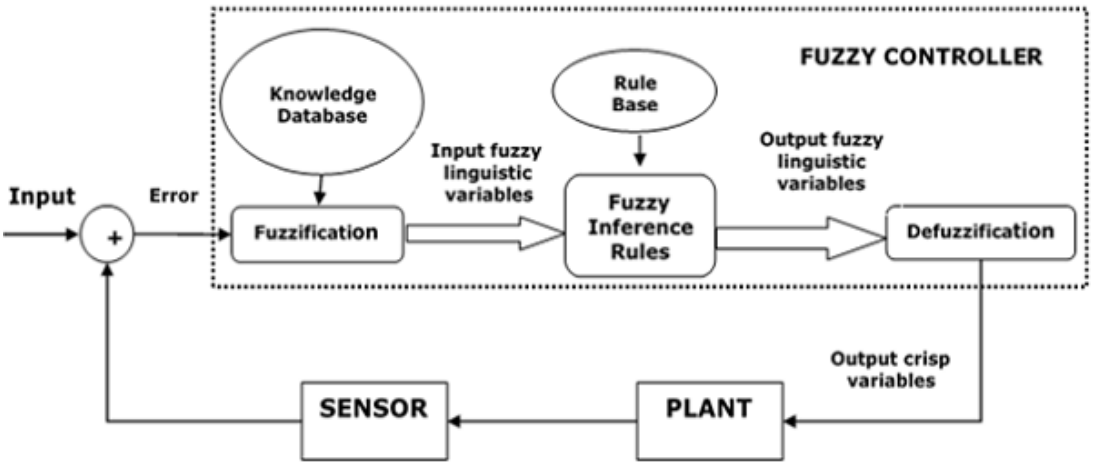
2.1 Základné koncepty fuzzy logiky v riadiacej technike

V úvodnej podkapitole sú definované základné koncepty fuzzy logiky v riadiacej technike. Nachádza sa tu podrobný popis štruktúry fuzzy riadiacich systémov, fuzzy kontroléra a kroky pri jeho vytváraní.

2.1.1 Štruktúra fuzzy riadiacich systémov

Štruktúra fuzzy riadiaceho systému zvyčajne pozostáva z troch základných komponentov: fuzzifikátor, fuzzy kontrolér a defuzzifikátor[3]. Fuzzifikátor slúži na prevod vstupných hodnôt z reálnych čísel na hodnoty fuzzy množín. Fuzzy kontrolér využíva pravidlá založené na fuzzy množinách na riadenie procesu. Defuzzifikátor potom prevádza výstupy z fuzzy kontroléra na reálne hodnoty, ktoré môžu byť použité v riadenom systéme.

Obrázok štruktúry fuzzy riadiaceho systému bol prevzatý zo zdroja [14].



Obrázok 3. Štruktúra fuzzy riadiaceho systému

2.1.2 Fuzzy kontrolér a jeho komponenty

Fuzzy kontrolér je kľúčovou súčasťou fuzzy riadiaceho systému a zabezpečuje riadiaci proces pomocou pravidiel založených na fuzzy množinách[3]. Fuzzy kontrolér zahŕňa dva hlavné typy: Mamdaniho a Sugeno-Takagiho[5][6].

Mamdaniho fuzzy kontrolér bol navrhnutý E. Mamdanim a S. Assilianom v roku 1975[5]. Tento typ kontroléra pracuje s fuzzy pravidlami v tvare "IF-THEN", kde sú podmienky aj dôsledky vyjadrené pomocou fuzzy množín. Po aplikácii fuzzy pravidiel sa získa viacero fuzzy množín, ktoré sú následne agregované do jednej výstupnej fuzzy množiny. Výsledná hodnota je potom získaná pomocou defuzzifikácie[8].

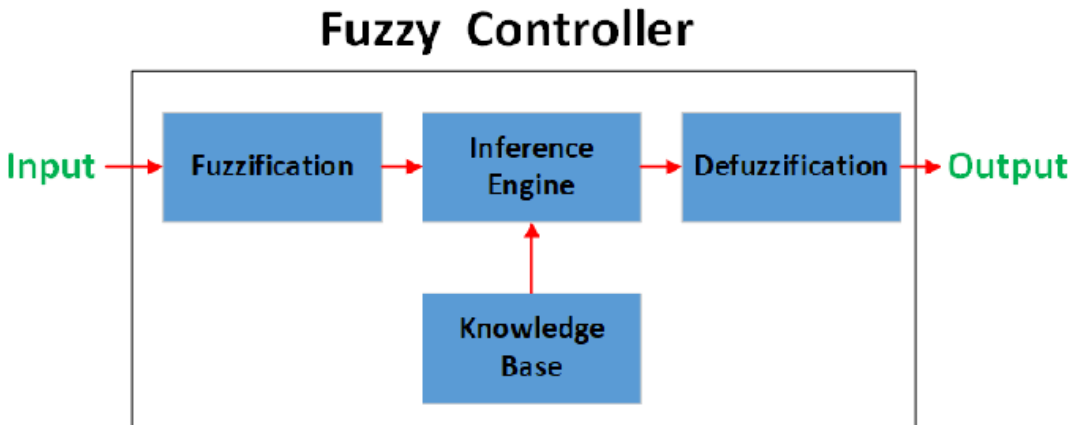
Sugeno-Takagiho fuzzy kontrolér, navrhnutý T. Takagi a M. Sugeno v roku 1985, je iný typ fuzzy kontroléra, kde dôsledky pravidiel sú vyjadrené ako funkcie[6]. Tento typ kontroléra je vhodný pre aplikácie s lineárnymi alebo nelineárnymi modelmi[9]. Výhodou Sugeno-Takagiho kontroléra je jeho matematická jednoduchosť a efektívna implementácia v riadiacich systémoch[10].

Fuzzy kontrolér sa skladá z nasledujúcich komponentov:

- Fuzzifikátor (Fuzzifier) – Fuzzifikátor slúži na prevod vstupných hodnôt z reálnych čísel na hodnoty fuzzy množín. Tento proces je potrebný, pretože fuzzy kontrolér pracuje s fuzzy množinami a ich členskými funkciami[3].
- Báza znalostí (Knowledge Base) – Zaznamenáva všetky fuzzy vstupno-výstupné vzťahy. Okrem toho má funkciu príslušnosti, ktorá určuje vstupy do bázy fuzzy pravidiel a výstupy do riadeného zariadenia.
- Báza pravidiel (Rule Base) – Báza pravidiel obsahuje všetky fuzzy pravidlá použité v fuzzy kontroléri, ktoré sú založené na expertnom vedomí alebo na získaných dátach. Pravidlá sú zvyčajne v tvare "IF-THEN" a popisujú vzťah medzi vstupmi a výstupmi kontroléra[3][7].
- Inferenčný mechanizmus (Inference Engine) – Inferenčný mechanizmus vykonáva dva hlavné kroky: aplikáciu pravidiel a agregáciu výsledkov. Aplikácia pravidiel zahŕňa kombinovanie vstupných fuzzy hodnôt s pravidlami v báze pravidiel. Agregácia výsledkov znamená zlúčenie viacerých výstupných fuzzy množín do jednej výstupnej fuzzy množiny[11].

- Defuzzifikátor (Defuzzifier) – Defuzzifikátor slúži na prevod výstupnej fuzzy množiny na reálnu hodnotu, ktorá môže byť použitá v riadenom systéme. Existuje niekoľko metód defuzzifikácie, ako napríklad centroidná metóda, metóda maximálneho členstva, metóda váženého priemeru a iné[3].

Obrázok štruktúry fuzzy kontroléra bol prevzaný zo zdroja [60].



Obrázok 4. Fuzzy kontrolér

Implementácia fuzzy kontroléra môže byť založená na softvérovej alebo hardvérovej úrovni. Softvérová implementácia zahŕňa výpočty vykonávané na počítači alebo mikrokontroléri[12]. Hardvérová implementácia sa môže skladať z špecializovaných integrovaných obvodov alebo z takzvaného programovateľného hradlového poľa (FPGA)[13]. Výber medzi softvérovou a hardvérovou implementáciou závisí na požiadavkách konkrétnej aplikácie, ako sú výpočtový výkon, energetická efektívnosť alebo náklady.

2.1.3 Kroky pri vytváraní fuzzy kontroléru

1. Identifikácia premenných – Tu sa musia identifikovať vstupné, výstupné a stavové premenné uvažovaného zariadenia[14].
2. Konfigurácia fuzzy podmnožiny – Každej z mnohých fuzzy podmnožín, ktoré tvoria informačný priestor, sa priradí jazyková značka. Treba dbať na to, aby sa do týchto fuzzy podmnožín vždy zahrnuli všetky prvky priestoru[14].
3. Získanie funkcie príslušnosti – Teraz sa získa funkcia príslušnosti pre každú fuzzy podmnožinu, ktorú sa konfigurovala v predchádzajúcom kroku[14].

4. Konfigurácia bázy pravidiel – V tomto kroku sa formuluje báza fuzzy pravidiel priradením vzťahu medzi fuzzy vstupom a výstupom[14].
5. Fuzzifikácia – Následne sa začína proces fuzzifikácie[14].
6. Kombinovanie fuzzy výstupov – Použitím fuzzy približného uvažovania sa nájde fuzzy výstup a spojí ich[14].
7. Defuzzifikácia – Nakoniec sa spustí proces defuzzifikácie na vytvorenie „crisp“ výstupu[14].

2.2 Výhody fuzzy logiky v riadiacej technike

Fuzzy logika prináša niekoľko výhod pre riadiace techniky, ktoré zlepšujú ich výkonnosť a efektivitu. Medzi hlavné výhody patrí flexibilita a adaptabilita fuzzy systémov, zjednodušenie zložitých modelov a robustnosť a odolnosť voči neistotám. Okrem týchto výhod existujú aj ďalšie výhody, ktoré ľudom uľahčujú každodenný život.

1. Flexibilita a adaptabilita fuzzy systémov - Fuzzy logika umožňuje riadiacim systémom efektívne spracovávať neurčité a nepresné informácie, čo im umožňuje prispôsobiť sa rôznym situáciám a zmenám v riadenom prostredí[3]. Vďaka tejto adaptabilite sú fuzzy systémy schopné pracovať v mnohých rôznych aplikáciách, vrátane tých, ktoré sú zložité alebo málo známe.
2. Zjednodušenie zložitých modelov - V niektorých prípadoch môže byť ťažké alebo dokonca nemožné získať presný matematický model pre riadený systém. Fuzzy logika umožňuje vytvorenie zjednodušených modelov, ktoré sa opierajú o expertné vedomie alebo získané dáta[7]. Tieto zjednodušené modely môžu byť ľahšie implementované a upravené ako tradičné matematické modely, čo znižuje nároky na výpočtové zdroje a zlepšuje efektivitu riadiaceho systému.
3. Robustnosť a odolnosť voči neistotám - Fuzzy riadiace systémy sú robustné voči neistotám a rušivým vplyvom, čo je dôležité v mnohých praktických aplikáciách[3]. Ich schopnosť zvládať nepresnosti a neurčitosť umožňuje fuzzy systémom pracovať s chybami a nedostatkami v meraných dátach alebo v modeloch. Táto vlastnosť zvyšuje celkovú odolnosť riadiaceho systému voči nežiaducim vplyvom a neistotám, čo vedie k lepším výsledkom riadenia.
4. Rýchlosť a efektivita rozhodovania - Fuzzy riadiace systémy môžu efektívne a rýchlo generovať rozhodnutia na základe fuzzy pravidiel, ktoré sú založené na expertnom vedomí alebo získaných dátach[3]. Táto vlastnosť umožňuje fuzzy kontrolérom

rýchlo reagovať na zmeny v riadenom prostredí a zlepšuje celkovú efektivitu riadenia.

5. Zlepšenie výkonu v nepresne definovaných oblastiach - Fuzzy logika je mimoriadne vhodná pre aplikácie, kde je potrebné riešiť nepresné alebo neurčité problémy[7]. Fuzzy kontroléry môžu byť úspešne použité tam, kde tradičné riadiace metódy zlyhávajú alebo sú neefektívne.
6. Zníženie nárokov na výpočtové zdroje - Fuzzy kontroléry sú schopné generovať uspokojivé výsledky aj pri použití zjednodušených modelov a menšom množstve výpočtových zdrojov[11]. To znižuje nároky na hardvér a zlepšuje energetickú efektivitu riadiacich systémov, čo je dôležité v rámci moderných vývojových trendov.
7. Ľahšia implementácia a údržba - Fuzzy kontroléry sú relatívne jednoduché na implementáciu a údržbu v porovnaní s niektorými tradičnými riadiacimi metódami, ako sú napríklad optimalizačné metódy alebo neurónové siete[9]. To znižuje náklady spojené s vývojom a údržbou riadiacich systémov a zvyšuje ich dostupnosť pre širokú škálu aplikácií.

2.3 Súvislosť aplikácie fuzzy logiky v mikropočítačoch s riadiacou technikou

Fuzzy logika a jej aplikácia v mikropočítačoch a riadiacich systémoch sú dve úzko prepojené oblasti výskumu a vývoja. Táto súvislosť vychádza z toho, že fuzzy logika poskytuje efektívny spôsob reprezentácie a spracovania neurčitých, nekompletných alebo nepresných informácií, ktoré sú často prítomné v reálnych aplikáciách a systémoch.

V prípade mikropočítačov je fuzzy logika často používaná ako prostriedok na zlepšenie výkonu a adaptability zariadení. Implementácia fuzzy logiky na mikropočítači umožňuje flexibilnejšie riadenie a adaptáciu na zmeny vstupných parametrov alebo vonkajších podmienok. Týmto spôsobom sa dosahuje lepšia efektívnosť, presnosť a spoľahlivosť pri spracovaní dát a rozhodovacom procese. Mikropočítače sú typicky vybavené obmedzenými zdrojmi, preto je dôležité, aby fuzzy logika bola implementovaná efektívne a šetrne k týmto zdrojom.

Riadiace systémy sú základným prvkom v rôznych priemyselných, domácich a dopravných aplikáciách. Fuzzy logika v riadiacich systémoch zohráva kľúčovú úlohu pri zlepšovaní ich výkonu, presnosti a odolnosti voči neistote alebo poruchám. V riadiacich systémoch je fuzzy logika zodpovedná za modelovanie a spracovanie neurčitých premenných, ako sú napríklad rôzne rušivé vplyvy, nepresné merania alebo zmeny prostredia. Pomocou fuzzy logiky je

možné vytvoriť riadiace pravidlá, ktoré sú schopné adaptovať sa na tieto zmeny a zabezpečiť stabilný a efektívny chod systému.

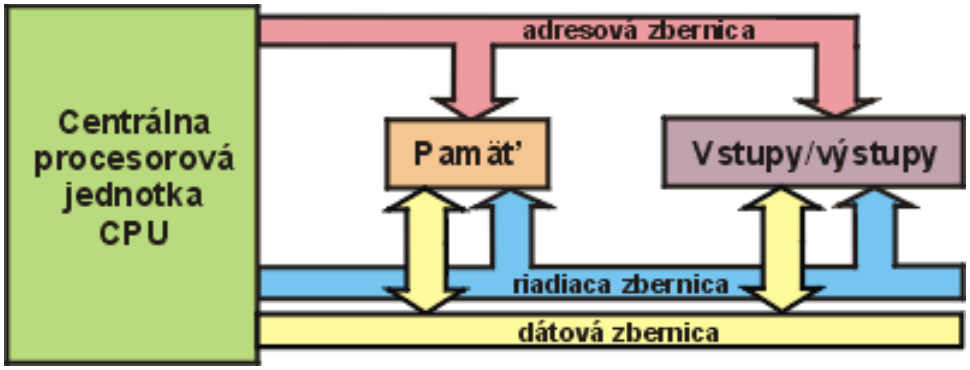
Súvislosť použitia fuzzy logiky na mikropočítači a fuzzy logiky v riadiacich systémoch spočíva v tom, že obidve oblasti sa zaoberajú riešením problémov spojených s neurčitosťou, premenlivými podmienkami a potrebou adaptácie. Taktiež platí vo veľa prípadoch, že mikropočítače s implementovanou fuzzy logikou sú jedným z komponentov fuzzy riadiaceho systému. Aplikácia fuzzy logiky na mikropočítačoch a v riadiacich systémoch je dôkazom širokej uplatniteľnosti tejto metódy a jej významu pre vývoj inteligentných zariadení a systémov. Výsledkom je zvýšená efektívnosť, presnosť a spoľahlivosť týchto systémov, ktoré sú schopné riadiť sa a adaptovať na rôzne neistoty a zmeny vo svojom okolí.

Vývoj fuzzy logiky na mikropočítačoch a v riadiacich systémoch sa neustále zlepšuje a zvyšuje svoju efektívnosť a presnosť. To vedie k širšiemu rozšíreniu a integrácii týchto metód do rôznych odvetví, ako sú energetika, doprava, priemysel či zdravotníctvo. Okrem toho, vďaka pokrokom v oblasti hardvéru a softvéru, sa fuzzy logika na mikropočítačoch stáva čoraz dostupnejšou a vhodnejšou pre mnoho rôznych aplikácií.

2.4 Mikropočítače a ich význam v modernej technológii

Mikropočítače predstavujú základnú zložku mnohých moderných technológií a zariadení[12]. Ich hlavnými komponentami sú mikroprocesor, pamäť, zbernice a ďalšie periférie, ktoré spolu tvoria kompletný počítačový systém[12]. V tejto podkapitole sa práca zameriava na význam mikropočítačov v modernej technológii, najmä v oblasti fuzzy logiky a jej aplikáciách.

Nasledujúci obrázok bol prevzatý zo zdroja [61].



Obrázok 5. Základné časti mikropočítača

V praxi sa fuzzy logika často aplikuje v rámci mikropočítačov, ktoré sú integrované do zariadení a systémov[10]. Medzi takéto zariadenia patrí napríklad automatické riadenie klimatizácie, riadenie motorov, navigačné systémy alebo systémy pre spracovanie signálu[4]. Výhodou použitia mikropočítačov s fuzzy logikou je schopnosť prispôsobiť sa meniacim sa podmienkam a zlepšiť výkon zariadení[10].

Mikropočítače s fuzzy logikou využívajú už spomínané fuzzy kontroléry, ktoré sú založené na fuzzy množinách a pravidlách. Tieto kontroléry poskytujú adaptívne riadenie prostredníctvom aproximácie a interpolácie medzi pravidlami.

Význam mikropočítačov v modernej technológii je značný, pretože umožňujú efektívne implementovať fuzzy logiku do rôznych zariadení a systémov[9][10]. Tieto mikropočítače majú obvykle malé rozmery, nízku spotrebu energie a sú cenovo dostupné, čo prispieva k ich širokému využitiu v rôznych aplikáciách[13].

Mikropočítače sú dôležitou súčasťou modernej technológie, pretože ich integrácia do zariadení a systémov umožňuje zlepšenie výkonu a prispôsobivosť riešeniam založeným na fuzzy logike. Vďaka tomu je možné vytvárať inteligentné systémy, ktoré sa dokážu prispôsobiť rôznym situáciám a zlepšiť kvalitu služieb alebo výrobkov.

Ako už bolo spomenuté v predošlej kapitole, okrem fuzzy logiky sú mikropočítače tiež využívané v mnohých iných oblastiach, ako napríklad v medicíne, telekomunikáciách, doprave, priemysle a spotrebnej elektronike[13].

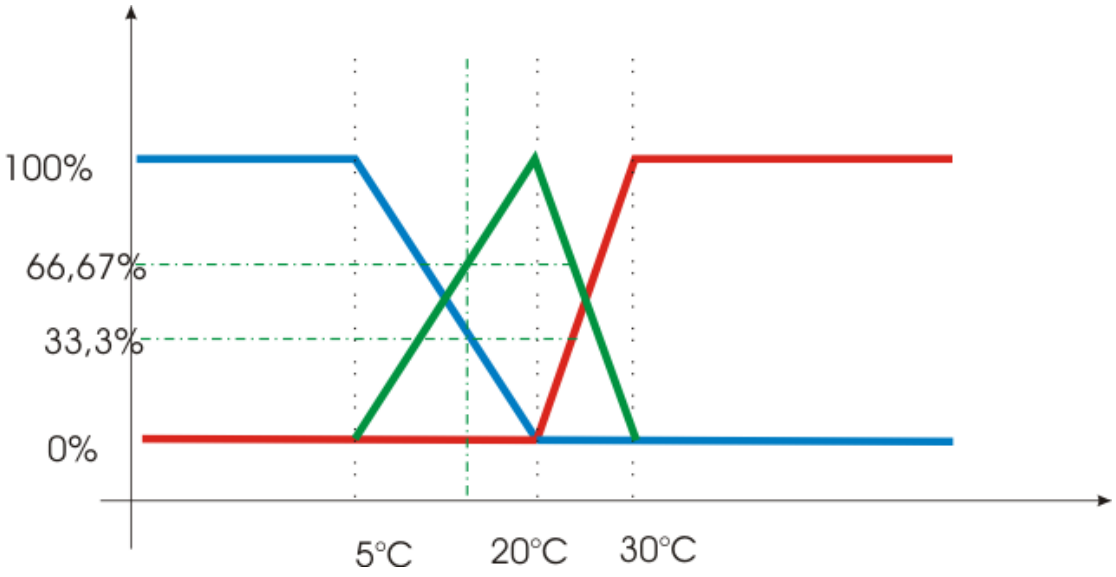
Vývoj mikropočítačov pokračuje rýchlym tempom, a tak je možné sa v budúcnosti tešiť na ďalšie zlepšenia výkonu, efektivity a prispôsobivosti týchto systémov. Tieto inovácie budú mať pozitívny vplyv na celú škálu technologických aplikácií, či už ide o riešenia založené na fuzzy logike alebo o iné oblasti využitia mikropočítačov.

Záverom, mikropočítače sú neoddeliteľnou súčasťou modernej technológie, pričom ich význam spočíva v schopnosti riešiť komplexné úlohy v rôznych oblastiach, ako aj v možnosti integrácie do širokého spektra zariadení a systémov. Fuzzy logika a jej aplikácie predstavujú jednu z mnohých oblastí, v ktorých mikropočítače prinášajú významné výhody a zlepšenia.

2.5 Aplikácie fuzzy logiky na mikropočítačoch

V tejto podkapitole je práca zameraná na viacero príkladov aplikácií fuzzy logiky na mikropočítačoch, ktoré sú podrobnejšie popísané, s dôrazom na výhody oproti použitiu klasickej logiky.

Na obrázku [62] podobne ako na obrázku číslo 1 je možné vidieť graf funkcií príslušnosti troch druhov teplôt. Je to základný prvok pre riadenie klimatizácie na základe teploty



Obrázok 6. Príklad fuzzy riadenia klimatizácie na základe teploty

1. Riadenie klimatizácie: Fuzzy logika sa často využíva v systémoch automatického riadenia klimatizácie. Mikropočítače s fuzzy kontrolérmi dokážu lepšie prispôsobiť nastavenia klimatizácie aktuálnym podmienkam, čím zlepšujú komfort a účinnosť energetického využitia[10]. Konkrétne sa fuzzy logika uplatňuje napríklad pri kontrole a upravovaní rýchlosti ventilátora klimatizácie a prispôsobení ohrevu, prípadne chladeniu vzduchu. Oproti binárnej logike má výhody v tom, že k správnej funkcionalite stačia implementovať fuzzy pravidlá v lingvistickom tvare, spolu s fuzzy množinami, zatiaľ čo pri binárnej logike by bolo potrebné dané pravidlá implementovať zložitými matematickými metódami a ani po danej implementácií by nebola dosiahnutá taká efektivita, ako pri použití fuzzy logiky[52]. Jedným z výrobcov, ktorý využívajú fuzzy logiku v klimatizáciách je firma Mitsubishi Electric[54].
2. Kontrola motorov: Fuzzy logika na mikropočítačoch môže byť použitá na optimalizáciu riadenia motorov, napríklad v automobiloch alebo priemyselných zariadeniach[4]. Výhodou oproti klasickej logike je schopnosť prispôsobiť sa zmenám zaťaženia, teploty alebo iným podmienkam, čo vedie k zlepšeniu výkonu motorov a zníženiu spotreby paliva[10]. Používa sa konkrétne v systémoch klimatizácie, ABS (protiblokovací brzdový systém) alebo napríkald v ASR (protipreklzová regulácia). Vo veľkom ju využívajú viaceré firmy, jednou z nich je napríklad Toyota[55][56].

3. Navigačné systémy: Fuzzy logika na mikropočítačoch sa tiež využíva v navigačných systémoch, kde pomáha pri výbere optimálnej trasy na základe rôznych faktorov, ako sú vzdialenosť, čas, dopravné podmienky alebo preferencie používateľa[4]. Oproti klasickej logike, ktorá môže pracovať iba s pevnými hodnotami a pravidlami, fuzzy logika umožňuje zohľadniť neurčitosť a nejednoznačnosť, čo vedie k lepším a prispôsobivejším navigačným riešeniam[3][10]. TomTom je jedným z výrobcov navigačných systémov, ktoré využívajú fuzzy logiku[57].
4. Systémy pre spracovanie signálu: Fuzzy logika na mikropočítačoch sa používa aj pri spracovaní signálov, napríklad v systémoch pre rozpoznávanie reči alebo obrazu[4]. Fuzzy logika dokáže lepšie filtrovať šum a rozpoznávať signály v prostredí s vysokou mierou neurčitosti, čím zlepšuje kvalitu spracovania signálov oproti klasickej logike[3][10].
5. Domáca automatizácia: Fuzzy logika sa čoraz viac využíva v oblasti domácej automatizácie, kde mikropočítače riadia rôzne domáce systémy a zariadenia, ako sú osvetlenie, vykurovanie, zavlažovanie alebo zabezpečenie[4]. Vďaka schopnosti pracovať s neurčitosťou a nejednoznačnosťou, fuzzy logika na mikropočítačoch umožňuje inteligentnejšie a prispôsobivejšie riadenie týchto systémov oproti klasickej logike[10]. Napríklad, fuzzy kontrolér môže automaticky prispôsobiť intenzitu osvetlenia v miestnosti na základe vonkajšieho svetla a preferencií používateľa, čo znižuje spotrebu energie a zvyšuje komfort[4][10].

Na záver je možné konštatovať, že aplikácie fuzzy logiky na mikropočítačoch prinášajú významné výhody oproti použitiu klasickej logiky. Vďaka svojim schopnostiam, fuzzy logika umožňuje zlepšenie výkonu, prispôsobivosť a účinnosť rôznych zariadení a systémov. Tieto aplikácie sú dôkazom širokého spektra možností, ktoré fuzzy logika ponúka v modernej technológii.

3 PROGRAMOVACIE JAZYKY FUZZY LOGIKY

Nasledujúca kapitola obsahuje prehľad programovacích jazykov, ktoré sa používajú, respektíve boli používané na implementáciu fuzzy logiky. Ďalej kapitola pojednáva o efektívnych oblastiach použitia daných programovacích jazykov.

3.1 Python

Python je široko používaný vysokoúrovňový programovací jazyk, ktorý je populárny pre svoju jednoduchosť a čitateľnosť [15]. V Pythone existuje viacero knižníc na podporu fuzzy logiky. Jednou z najobľúbenejších knižníc je scikit-fuzzy [16], ktorá je založená na populárnom scikit-learn. Inou knižnicou je pyfuzzy [17], ktorá ponúka jednoduchý a prístupný spôsob implementácie fuzzy logiky.

3.2 MATLAB

MATLAB je výkonný výpočtový jazyk a interaktívne prostredie zamerané na vedecké a inžinierske výpočty [18]. MATLAB podporuje fuzzy logiku prostredníctvom Fuzzy Logic Toolboxu [19], ktorý poskytuje nástroje pre vytváranie, editovanie a vizualizáciu fuzzy inference systémov.

3.3 Java

Java je všeobecne používaný objektovo orientovaný programovací jazyk, ktorý je známy svojou platformovou nezávislosťou a robustnosťou [20]. Jednou z knižníc pre fuzzy logiku v Jave je jFuzzyLogic [21], ktorá umožňuje vytváranie, testovanie a ladenie fuzzy kontrolných systémov. Ďalšou knižnicou je Juzzy [22], ktorá ponúka nástroje pre prácu s fuzzy množinami a fuzzy pravidlami.

3.4 C++

C++ je výkonný programovací jazyk, ktorý poskytuje vysokú úroveň abstrakcie, zatiaľ čo umožňuje prístup k nižším úrovniam počítačových systémov [23]. FuzzyLite je jedna z knižníc, ktorá podporuje fuzzy logiku v C++ [24]. FuzzyLite poskytuje jednoduchý a efektívny spôsob implementácie fuzzy logiky v C++ aplikáciách.

3.5 R

R je programovací jazyk a softvérové prostredie pre štatistické výpočty a grafiku [25]. FRBS (Fuzzy Rule-Based Systems) je balíček, ktorý umožňuje implementáciu fuzzy logiky v R [26]. Balíček obsahuje funkcie pre fuzzy inference, vytváranie fuzzy pravidel a prácu s fuzzy množinami.

3.6 C#

C# je moderný, typovo bezpečný a objektovo orientovaný programovací jazyk navrhnutý pre vývoj softvéru pre platformu Microsoft .NET [27]. V C# existuje niekoľko knižníc a frameworkov na podporu fuzzy logiky. AForge.NET je jedným z týchto frameworkov [28], ktorá obsahuje modul pre fuzzy logiku umožňujúci vytváranie a spracovanie fuzzy pravidiel. Ďalšou knižnicou je DotFuzzy [29], ktorá ponúka jednoduchý spôsob implementácie fuzzy logiky v C# aplikáciách.

3.7 JavaScript

JavaScript je populárny skriptovací jazyk používaný najmä pre webové aplikácie [30]. Javascript-fuzzylogic je jedným z nástrojov, ktorý umožňuje implementáciu fuzzy logiky v JavaScripte. Javascript-fuzzylogic ponúka jednoduchý a prístupný spôsob práce s fuzzy množinami a pravidlami [39].

3.8 Ruby

Ruby je dynamický, objektovo orientovaný programovací jazyk s dôrazom na jednoduchosť a produktivitu [31]. FuzzyLogic je knižnica pre Ruby, ktorá umožňuje implementáciu fuzzy logiky [32]. FuzzyLogic poskytuje jednoduché rozhranie pre prácu s fuzzy množinami, pravidlami a inferenciami.

3.9 Lisp

Lisp je jeden z najstarších programovacích jazykov, ktorý sa používa najmä v oblasti umelej inteligencie [33]. FuzzyLisp je knižnica, ktorá umožňuje implementáciu fuzzy logiky v Lisp [34]. FuzzyLisp poskytuje jednoduché rozhranie pre prácu s fuzzy množinami, pravidlami a inferenciami v kontexte funkcionálneho programovania.

3.10 Oblasti použitia spomínaných programovacích jazykov

V tejto podkapitole sa práca bude zaoberať oblasťami použitia spomínaných programovacích jazykov a ich vhodnosťou pre rôzne typy aplikácií a problémov. Pre každú oblasť je uvedených jeden alebo viac vhodných programovacích jazykov a ich priradenie je zdôvodnené.

3.10.1 Dátová analýza a strojové učenie

V oblasti dátovej analýzy a strojového učenia sa často používajú jazyky Python [15] a R [25], keďže ponúkajú širokú škálu knižníc a nástrojov pre analýzu dát, vizualizáciu a implementáciu rôznych strojových učiacich modelov. V kontexte fuzzy logiky je Python obzvlášť vhodný vďaka knižniciam ako scikit-fuzzy [16] a pyfuzzy [17], ktoré poskytujú nástroje pre jednoduchú implementáciu fuzzy logiky v týchto oblastiach. R je tiež vhodný vďaka knižnici FRBS [26], ktorá umožňuje pracovať s fuzzy pravidlami a modelmi.

3.10.2 Inžinierske a vedecké aplikácie

Pre inžinierske a vedecké aplikácie, kde je dôležitá presnosť a rýchlosť výpočtov, je vhodný jazyk MATLAB [18]. MATLAB je bežne používaný pre numerické výpočty, analýzu signálov a systémov a vizualizáciu dát. Fuzzy Logic Toolbox [19] umožňuje vývojárom pracovať s fuzzy logikou v týchto oblastiach.

3.10.3 Webové a mobilné aplikácie

Java [20] a JavaScript [30] sú široko používané jazyky pre vývoj webových a mobilných aplikácií. Java je univerzálny a robustný jazyk s prenositeľnosťou a škálovateľnosťou, čo ho robí vhodným pre širokú škálu aplikácií. Knižnice ako jFuzzyLogic [21] a Juzzy [22] umožňujú vývojárom implementovať fuzzy logiku v Java aplikáciách. JavaScript je základom webových aplikácií, a nástroj Javascript-fuzzylogic poskytuje možnosti pre implementáciu fuzzy logiky na webových stránkach.

3.10.4 Operačné systémy a grafické aplikácie

C++ [23] je výkonný programovací jazyk bežne používaný pre vývoj operačných systémov, grafických a sieťových aplikácií. C++ je vhodný pre oblasti, kde je potrebná vysoká úroveň kontroly nad hardvérom a rýchlosť výpočtov. Knižnica FuzzyLite [24] poskytuje nástroje pre prácu s fuzzy logikou v C++ aplikáciách.

3.10.5 Podnikový softvér a aplikácie s vysokou úrovňou zložitosti

Java [20] a C# [27] sú vhodné jazyky pre vývoj podnikového softvéru a aplikácií s vysokou úrovňou zložitosti a veľkým množstvom dát. Oba jazyky poskytujú silnú podporu pre objektovo orientované programovanie, čo umožňuje vývojárom navrhovať a implementovať komplexné systémy. V prípade Java, knižnice ako jFuzzyLogic [21] a Juzzy [22] umožňujú vývojárom implementovať fuzzy logiku v aplikáciách. Pre C#, knižnice a frameworky ako AForge.NET [28] a DotFuzzy [29] poskytujú nástroje pre prácu s fuzzy logikou.

3.10.6 Symbolické a logické výpočty

Lisp [33] je vhodný jazyk pre oblasti, kde sú potrebné symbolické a logické výpočty, ako sú umelej inteligencie, expertné systémy a analýza formálnych jazykov. Tento jazyk má silnú podporu pre prácu s logikou a symbolickými výpočtami. Lisp má knižnicu FuzzyLisp [34] na prácu s fuzzy logikou v aplikáciách založených na Lisp.

3.10.7 Skriptovanie a rýchla prototypizácia

Python [15] a Ruby [31] sú vhodné jazyky pre skriptovanie a rýchlu prototypizáciu, keďže poskytujú jednoduché a flexibilné prostriedky pre vývoj kódu. Obidva jazyky majú silnú podporu pre dynamické typovanie a metaprogramovanie. V kontexte fuzzy logiky, Python má dostupné knižnice ako scikit-fuzzy [16] a pyfuzzy [17], zatiaľ čo Ruby poskytuje knižnicu FuzzyLogic [32] na prácu s fuzzy logikou.

II. PRAKTICKÁ ČÁST

4 KNIŽNICE NA IMPLEMENTÁCIU FUZZY LOGIKY

Úvodná kapitola praktickej časti sa bude venovať overovaniu a popisovaniu vlastností, charakteristík a výhod knižníc, ktoré sa používajú na implementáciu fuzzy logiky na mikropočítačoch, ako aj na implementáciu fuzzy logiky v iných odvetviach. Táto kapitola bude využitá aj pri ďalších častiach praktickej časti, kde s pomocou charakteristiky nasledujúcich knižníc bude vybraná vhodná knižnica pre ukážkovú aplikáciu.

4.1 eFLL (Embedded fuzzy logic library)

Knižnica eFLL (Embedded Fuzzy Logic Library) je výkonná a flexibilná knižnica, ktorá bola navrhnutá s cieľom zjednodušiť implementáciu fuzzy logiky do embedded systémov[35]. Táto knižnica umožňuje vývojárom používať fuzzy logické systémy s relatívne malým množstvom úsilia a zručností v oblasti fuzzy logiky. eFLL je postavená na jazyku C++/C, čo umožňuje jej široké využitie v rôznych platformách a aplikáciách [35].

Výhodou použitia eFLL je aj to, že nemá žiadne explicitné limitácie z hľadiska počtu fuzzy pravidiel, alebo vstupov a výstupov. Tieto vlastnosti sú obmedzené jedine konštrukciou a výkonom použitého mikropočítača [35]. Ďalšou výhodou je jej ľahké aplikovanie vo vývojovom prostredi (napríklad Arduino IDE). Knižnica využíva na svoju funkcionalitu iba štandartnú knižnicu jazyka C („stdlib.h“). Stačí si len stiahnuť, prípadne naklonovať knižnicu z repozitára a v IDE si ju následne zahrnúť do projektu [35].

4.1.1 Popis základných objektov eFLL

- Fuzzy object: Tento objekt obsahuje všetky fuzzy systémy a môžu sa prostredníctvom neho ovládať ich vstupy, výstupy, fuzzy množiny a jazykové pravidlá [35].
- FuzzyInput: Tento objekt zoskupuje všetky vstupné položky množín, ktoré patria do rovnakej domény [35].
- FuzzyOutput: Tento objekt je podobný objektu FuzzyInput, používa sa na zoskupenie všetkých výstupných Fuzzy množín, ktoré patria do rovnakej domény [35].
- FuzzySet: Jedna z kľúčových súčastí fuzzy knižnice, každá množina umožňuje modelovanie uvažovaného systému. Vo svojom konštruktore FuzzySet(float a, float b, float c, float d) knižnica teraz podporuje trojuholníkové funkcie príslušnosti, lichobežníkové a singletonové, ktoré sú vytvorené na základe bodov A, B, C a D [35].

- FuzzyRule: Základné pravidlo fuzzy objektu, ktoré obsahuje jeden alebo viac z tohto objektu, je namountované pomocou tohto objektu. FuzzyRule fr = new FuzzyRule (ID, antecedent, consequent) sa použije na definíciu jeho inštancie [35].
- FuzzyRuleAntecedent: Tento objekt sa používa na kombináciu objektu FuzzyRule, ktorý má na starosti zostavenie predchádzajúceho podmieneného výrazu FuzzyRule [35].
- FuzzyRuleConsequent: Tento objekt sa používa na zobrazenie objektu FuzzyRule, ktorý má na starosti zostavenie výstupného výrazu FuzzyRule [35].

4.1.2 Funkcie a vlastnosti eFLL

Knižnica eFLL poskytuje rôzne funkcie a vlastnosti, ktoré umožňujú vývojárom pracovať s fuzzy logikou. Nasledujúce sú hlavné vlastnosti knižnice eFLL [35]:

1. Definícia fuzzy množín: eFLL umožňuje definovať fuzzy množiny s rôznymi tvarmi funkcií príslušnosti, ako sú trojuholníková, trapezoidálna, gaussianská a sigmoidálna.
2. Definícia fuzzy pravidiel: Knižnica poskytuje jednoduchý spôsob definovania fuzzy pravidiel pomocou jazyka IF-THEN. Vývojári môžu vytvárať súbor pravidiel, ktorý popisuje vzťahy medzi vstupmi a výstupmi fuzzy systému.
3. Fuzzy inferencia: eFLL implementuje metódy fuzzy inferencie, ako sú Mamdani a Takagi-Sugeno. Tieto metódy umožňujú výpočet výstupu fuzzy systému na základe vstupov a definovaných pravidiel.
4. Defuzzifikácia: Knižnica obsahuje rôzne metódy defuzzifikácie, ako sú stred hodnôt (centroid), vážený stred (bisector) a maximálna príslušnosť (maximum). Tieto metódy umožňujú previesť fuzzy výstup na konkrétnu hodnotu, ktorá je zrozumiteľná pre ľudí a systémy.
5. Kompatibilita s rôznymi platformami: ako už bolo spomenuté, eFLL je navrhnutá tak, aby bola kompatibilná s rôznymi platformami a operačnými systémami. Je možné ju integrovať do bežných embedded systémov, ako aj do výkonnejších počítačov.

4.1.3 Overenie vlastností

Keďže knižnica eFLL bola vybraná na vytvorenie ukážkovej aplikácie v praktickej časti, je overenie vlastností tejto knižnice súčasťou vytvorenia zdrojového kódu k ukážkovej aplikácií.

4.2 Fuzzylite

Fuzzylite je vysoko výkonná a ľahko použiteľná knižnica pre implementáciu fuzzy logických systémov[36]. Podobne ako eFLL, knižnica bola navrhnutá tak, aby bola jednoduchá na použitie a flexibilná pre rôzne aplikácie. Fuzzylite je napísaná v jazyku C++ a ponúka rozhrania pre viaceré programovacie jazyky, ako sú Java, Python a .NET. To umožňuje jej široké využitie v rôznych platformách a aplikáciách.

4.2.1 Funkcie a vlastnosti fuzzylite

Knižnica fuzzylite poskytuje rôzne funkcie a vlastnosti, ktoré umožňujú vývojárom pracovať s fuzzy logikou. Nasledujúce sú hlavné vlastnosti knižnice fuzzylite[36]:

1. Kontroléri: Mamdani, Takagi-Sugeno, Larsen, Tsukamoto, Inverzný Tsukamoto a hybridné [36].
2. Lingvistické termíny: Základné (trojuholník, trapezoid, obdĺžnik, diskrétne), rozšírené (kosínus, gaussovské, gaussovský produkt, pi-tvar, sigmoidný rozdiel, sigmoidný produkt, špic), hrany (binárne, konkávne, rampa, sigmoid, s-tvar, z-tvar) a funkcie (konštanta, lineárna, funkcia) [36].
3. Aktivačné metódy: Všeobecné, proporcionálne, prahové, prvé, posledné, najnižšie, najvyššie [36].
4. Konjunkcia a implikácia (T-normy): Minimum, algebraický produkt, ohraničený rozdiel, drastický produkt, Einsteinov produkt, Hamacherov produkt, nilpotentné minimum, funkcia [36].
5. Disjunkcia a agregácia (S-normy): Maximum, algebraický súčet, ohraničený súčet, drastický súčet, Einsteinov súčet, Hamacherov súčet, nilpotentné maximum, normalizovaný súčet, neohraničený súčet, funkcia [36].
6. Defuzzifikátory: Integrálne (centroid, bisector, najmenšie z maxím, najväčšie z maxím, priemer maxím) a vážené (vážený priemer, vážený súčet) [36].
7. Zmenšovanie hodnoty (hedges): Akýkoľvek, nie, extrémne, zriedka, čiastočne, veľmi, funkcia [36].

8. Importéry: FuzzyLite jazyk, Fuzzy Inference System, Fuzzy Control Language (fcl) [36].
9. Exportéry: C++, Java, FuzzyLite jazyk, FuzzyLite Dataset, R skript, Fuzzy Inference System, Fuzzy Control Language[36].

4.2.2 Overenie vlastností

Overovanie vlastností knižníc fuzzylite a jfuzzylite prebiehalo pomocou vytvorenia jednoduchého ukážkového kódu na otestovanie základných funkcií knižníc. V oboch prípadoch sa jedná o kód, v ktorom sa pomocou fuzzy logiky na základe vstupnej teploty riadi rýchlosť ventilátora klimatizácie.

Kód je rozdelený do niekoľkých kľúčových častí: definovanie vstupnej premennej, výstupnej premennej, pravidiel a procesu regulácie.

Najskôr je definovaná vstupná premenná "temperature". Táto premenná je inicializovaná a spolu s ňou je nastavený rozsah hodnôt (0 až 50 stupňov), ako aj príslušné fuzzy termíny ("Cold", "Good" a "Hot"), ktoré sú reprezentované trojuholníkovými funkciami príslušnosti. Každý termín pokrýva určitý rozsah hodnôt teploty.

Potom je definovaná výstupná premenná "fanSpeed". Tá je inicializovaná, je jej nastavený rozsah hodnôt (0 až 2) a sú jej pridelené fuzzy termíny ("Low", "Medium" a "High"). K výpočtu hodnoty tejto premennej je použitý defuzzifikátor Centroid a na agregáciu viacerých pravidiel je použitý algoritmus algebraického súčtu.

Ďalej je vytvorený blok pravidiel. Pravidlá fuzzy logiky sú definované ako "ak je teplota 'Cold', potom je rýchlosť ventilátora 'Low'" a podobne pre ostatné stavy teploty.

V poslednej časti kódu sa aplikuje logika fuzzy systému. Pre každú z preddefinovaných hodnôt teploty sa vypočíta rýchlosť ventilátora a vypíše sa na štandardný výstup.

Tento kód ilustruje použitie fuzzy logiky na riadenie zariadenia podľa nameraných hodnôt, v tomto prípade teploty prostredia. Použitie fuzzy logiky môže viesť k prirodzenejšímu a efektívnejšiemu riadeniu v porovnaní s tradičnými metódami.

```
fl::InputVariable* temperature = new fl::InputVariable;
temperature->setEnabled(true);
temperature->setName("Temperature");
temperature->setRange( minimum: 0.0,  maximum: 50.0);
temperature->addTerm( term: new fl::Triangle( name: "Cold",  vertexA: 0.0,  vertexB: 10.0,  vertexC: 20.0));
temperature->addTerm( term: new fl::Triangle( name: "Good",  vertexA: 10.0,  vertexB: 20.0,  vertexC: 30.0));
temperature->addTerm( term: new fl::Triangle( name: "Hot",  vertexA: 20.0,  vertexB: 30.0,  vertexC: 50.0));
engine.addInputVariable( inputVariable: temperature);

fl::OutputVariable* fanSpeed = new fl::OutputVariable;
fanSpeed->setEnabled(true);
fanSpeed->setName("FanSpeed");
fanSpeed->setRange( minimum: 0.0,  maximum: 2.0);
fanSpeed->setDefuzzifier(new fl::Centroid());
fanSpeed->setDefaultValue(fl::nan);
fanSpeed->setAggregation(new fl::AlgebraicSum);
fanSpeed->addTerm( term: new fl::Triangle( name: "Low",  vertexA: 0.0,  vertexB: 0.5,  vertexC: 1.0));
fanSpeed->addTerm( term: new fl::Triangle( name: "Medium",  vertexA: 0.5,  vertexB: 1.0,  vertexC: 1.5));
fanSpeed->addTerm( term: new fl::Triangle( name: "High",  vertexA: 1.0,  vertexB: 1.5,  vertexC: 2.0));
engine.addOutputVariable( outputVariable: fanSpeed);

fl::RuleBlock* ruleBlock = new fl::RuleBlock;
ruleBlock->setConjunction(new fl::AlgebraicProduct);
ruleBlock->setDisjunction(new fl::AlgebraicSum);
ruleBlock->setImplication(new fl::AlgebraicProduct);
ruleBlock->setActivation(new fl::General);
ruleBlock->addRule( rule: fl::Rule::parse( rule: "if Temperature is Cold then FanSpeed is Low",  engine: &engine));
ruleBlock->addRule( rule: fl::Rule::parse( rule: "if Temperature is Good then FanSpeed is Medium",  engine: &engine));
ruleBlock->addRule( rule: fl::Rule::parse( rule: "if Temperature is Hot then FanSpeed is High",  engine: &engine));
engine.addRuleBlock(ruleBlock);
```

Obrázok 7. Definovanie fuzzy pravidiel a fuzzy množín v knižnici fuzzylite

```
double temperatureValues[] = { [0]: 5.0,  [1]: 15.0,  [2]: 25.0,  [3]: 35.0,  [4]: 45.0};

for (size_t i = 0; i < sizeof(temperatureValues)/sizeof(double); ++i) {
    temperature->setValue(temperatureValues[i]);
    engine.process();
    std::cout << "Temperature: " << temperatureValues[i] << ", FanSpeed: " << fanSpeed->getValue() << std::endl;
}
```

Obrázok 8. Pole na vzorové vstupné teploty a cyklus na výpis výstupných teplôt

```
Temperature: 5, FanSpeed: 0.5
Temperature: 15, FanSpeed: 0.75
Temperature: 25, FanSpeed: 1.25
Temperature: 35, FanSpeed: 1.5
Temperature: 45, FanSpeed: 1.5
```

Obrázok 9. Výstup kódu

Knižnica podporuje mnoho funkcií na výpočty pre fuzzifikáciu a defuzzifikáciu, avšak ich použitie je náročné. Ďalšou nevýhodou je nutnosť inštalácie ďalších nástrojov (C++ kompilátor, MinGw32-make) na správnu kompiláciu knižnice. Zahrnutie knižnice do projektu je taktiež náročné.

4.3 jFuzzylite

Jfuzzylite je knižnica pre implementáciu fuzzy logických systémov v jazyku Java[37]. Knižnica je založená na už existujúcej knižnici fuzzylite, ktorá bola pôvodne navrhnutá v jazyku C++[36]. Jfuzzylite bola vytvorená s cieľom poskytnúť vývojárom, ktorí preferujú jazyk Java, rovnaké možnosti a výhody ako pri použití knižnice fuzzylite.

4.3.1 Funkcie a vlastnosti jFuzzylite

Knižnica jFuzzylite má rovnaké vlastnosti a funkcie ako jej pôvodná verzia v jazyku C++ fuzzylite. Líši sa len použitým programovacím jazykom použitým na definovanie objektov a metód na prácu s fuzzy logikou [37].

Nevýhoda jFuzzylite je, že určitá časť mikropočítačov nepodporuje jazyk Java. Na druhej strane je ale pravda že Java môže byť v niektorých oblastiach efektívna.[36][37][38].

4.3.2 Overenie vlastností

Ako už bolo spomenuté, v rámci overovania vlastností bol vytvorený rovnaký ukážkový kód pre knižnice fuzzylite a jfuzzylite. Popis kódu je teda nasledovný.

Hlavný objekt knižnice jfuzzylite, ktorý je použitý v tomto kóde, je *Engine*. *Engine* je základným stavebným kameňom pre implementáciu fuzzy logických systémov v knižnici jfuzzylite. V tomto prípade je vytvorený *Engine* s názvom "TemperatureControl".

InputVariable je trieda, ktorá reprezentuje vstupnú premennú fuzzy systému. V tomto kóde je vytvorená jedna vstupná premenná *temperature*, ktorá reprezentuje teplotu v stupňoch Celzia od 0 do 50. Pre túto premennú sú definované tri fuzzy množiny - "Cold", "Good" a "Hot", ktoré sú reprezentované trojuholníkovými funkciami príslušnosti (Triangle).

OutputVariable je trieda reprezentujúca výstupnú premennú fuzzy systému. V kóde je definovaná výstupná premenná *fanSpeed*, ktorá reprezentuje rýchlosť ventilátora a môže nadobúdať hodnoty v rozsahu 0.0 až 2.0. Pre túto výstupnú premennú sú definované tri fuzzy množiny - "Low", "Medium" a "High", tiež reprezentované trojuholníkovými funkciami príslušnosti.

RuleBlock je trieda, ktorá obsahuje súbor fuzzy pravidiel. V tomto prípade obsahuje tri pravidlá, ktoré definujú, aká rýchlosť ventilátora by mala byť nastavená v závislosti od aktuálnej teploty. Pravidlá sú definované v lingvistickom tvare, napríklad "ak je teplota chladná,

potom je rýchlosť ventilátora nízka". Použitie lingvistických pravidiel je jednou z kľúčových vlastností fuzzy logiky, ktorá ju odlišuje od tradičnej binárnej logiky.

Následne sa pre každú hodnotu teploty v poli *temperatureValues* nastaví aktuálna hodnota premennej *temperature*, vyhodnotia sa fuzzy pravidlá a vypočíta sa výstupná hodnota premennej *fanSpeed*.

```
public class jfuzzylite_ukazka {
    Run | Debug
    public static void main(String[] args) {
        Engine engine = new Engine();
        engine.setName(name:"TemperatureControl");

        InputVariable temperature = new InputVariable();
        temperature.setEnabled(enabled:true);
        temperature.setName(name:"Temperature");
        temperature.setRange(minimum:0.0, maximum:50.0);
        temperature.setLockValueInRange(lockValueInRange:true);
        temperature.addTerm(new Triangle(name:"Cold", vertexA:0.0, vertexB:10.0, vertexC:20.0));
        temperature.addTerm(new Triangle(name:"Good", vertexA:10.0, vertexB:20.0, vertexC:30.0));
        temperature.addTerm(new Triangle(name:"Hot", vertexA:20.0, vertexB:30.0, vertexC:50.0));
        engine.addInputVariable(temperature);

        OutputVariable fanSpeed = new OutputVariable();
        fanSpeed.setEnabled(enabled:true);
        fanSpeed.setName(name:"FanSpeed");
        fanSpeed.setRange(minimum:0.0, maximum:2.0);
        fanSpeed.setDefuzzifier(new Centroid());
        fanSpeed.setDefaultValue(Double.NaN);
        fanSpeed.setLockPreviousValue(lockPreviousValue:false);
        fanSpeed.setAggregation(new Maximum());
        fanSpeed.addTerm(new Triangle(name:"Low", vertexA:0.0, vertexB:0.5, vertexC:1.0));
        fanSpeed.addTerm(new Triangle(name:"Medium", vertexA:0.5, vertexB:1.0, vertexC:1.5));
        fanSpeed.addTerm(new Triangle(name:"High", vertexA:1.0, vertexB:1.5, vertexC:2.0));
        engine.addOutputVariable(fanSpeed);

        RuleBlock ruleBlock = new RuleBlock();
        ruleBlock.setConjunction(new AlgebraicProduct());
        ruleBlock.setDisjunction(new AlgebraicSum());
        ruleBlock.setImplication(new Minimum());
        ruleBlock.setActivation(new General());
        ruleBlock.addRule(Rule.parse(rule:"if Temperature is Cold then FanSpeed is Low", engine));
        ruleBlock.addRule(Rule.parse(rule:"if Temperature is Good then FanSpeed is Medium", engine));
        ruleBlock.addRule(Rule.parse(rule:"if Temperature is Hot then FanSpeed is High", engine));
        engine.addRuleBlock(ruleBlock);

        double[] temperatureValues = {5.0, 15.0, 25.0, 35.0, 45.0};

        for (double temp : temperatureValues) {
            temperature.setValue(temp);
            engine.process();
            System.out.println("Temperature: " + temp + ", FanSpeed: " + fanSpeed.getValue());
        }
    }
}
```

Obrázok 10. Kód na implementáciu fuzzy logiky v knižnice jfuzzylite

```
Temperature: 5.0, FanSpeed: 0.5000000000000002
Temperature: 15.0, FanSpeed: 0.7500000000000002
Temperature: 25.0, FanSpeed: 1.2500000000000002
Temperature: 35.0, FanSpeed: 1.4999999999999998
Temperature: 45.0, FanSpeed: 1.5000000000000002
```

Obrázok 11. Výstup kódu

Triedy a metódy v knižnici fungujú na tom istom princípe ako fuzzylite. Platí to aj pri zložitej kompilácií knižnice na získanie Java archivačného súboru (.jar). Je nutné mať okrem JDK na použitie Javy ešte aj Maven na vytvorenie .jar súboru.

4.4 Scikit-fuzzy

Scikit-fuzzy je flexibilná softvérová knižnica pre fuzzy logické systémy, ktorá je implementovaná v jazyku Python[16]. Táto knižnica je založená na scikit-learn, všeobecne uznávanej knižnici pre strojové učenie v Python[16]. Scikit-fuzzy bola vytvorená s cieľom poskytnúť intuitívne a efektívne nástroje pre prácu s fuzzy logikou.

Knižnica umožňuje definovať fuzzy množiny a pravidlá, ktoré sú potom využité na inferenciu a defuzzifikáciu[16].

Scikit-fuzzy tiež podporuje vizualizáciu fuzzy množín a inferenčných systémov, čo umožňuje lepšie porozumieť a analyzovať vytvorené modely. To je zásadné pri vývoji a ladení fuzzy logických systémov [16].

4.4.1 Funkcie a vlastnosti Scikit-fuzzy

Scikit-fuzzy poskytuje širokú škálu nástrojov a funkcií určených pre prácu s fuzzy logikou. Hlavné vlastnosti a funkcie knižnice scikit-fuzzy sú[16]:

1. Definícia fuzzy množín: Scikit-fuzzy umožňuje definovať a manipulovať s fuzzy množinami, pričom poskytuje rôzne funkcie príslušnosti, ako sú gaussovské, sigmoidálne, trapezoidálne a mnohé ďalšie.
2. Fuzzy pravidlá: Knižnica umožňuje definovať fuzzy pravidlá pomocou jednoduchého jazyka založeného na IF-THEN logike. Toto umožňuje vývojárom intuitívne definovať vzťahy medzi vstupmi a výstupmi v fuzzy systéme.
3. Fuzzy inferencia: Scikit-fuzzy podporuje Mamdaniho a Takagi-Sugeno inferenčné systémy, čo umožňuje flexibilitu v implementácii fuzzy systémov.
4. Defuzzifikácia: Knižnica poskytuje rôzne metódy defuzzifikácie, vrátane centroidu a bisectra, umožňujúce previesť fuzzy výstup na konkrétnu hodnotu.
5. Integrácia so scikit-learn: Ako súčasť scikit, scikit-fuzzy je kompatibilná s inými knižnicami v scikite, čo umožňuje vytvárať komplexné systémy založené na fuzzy logike a strojovom učení.

4.5 Fuzzy logic toolbox

Fuzzy Logic Toolbox je produkt spoločnosti MathWorks, ktorý poskytuje nástroje pre prácu s systémami a algoritmami fuzzy logiky[19]. Táto knižnica je integrovaná do prostredia MATLAB, čo umožňuje vývojárom využívať silu a flexibilitu tohto výpočtového prostredia pri práci s fuzzy logikou.

Knižnica poskytuje možnosti vizualizácie, ktoré sú veľmi užitočné pre analýzu a ladenie fuzzy logických systémov[19].

4.5.1 Funkcie a vlastnosti Fuzzy logic toolbox

Fuzzy Logic Toolbox ponúka širokú škálu nástrojov a funkcií určených pre prácu s fuzzy logikou. Medzi hlavné vlastnosti a funkcie tejto knižnice patrí[19]:

1. Grafické užívateľské rozhranie: Fuzzy Logic Toolbox ponúka interaktívne grafické užívateľské rozhranie pre návrh a testovanie fuzzy inferenčných systémov.
2. Definícia a manipulácia s fuzzy množinami: Knižnica umožňuje definovať a manipulovať s fuzzy množinami pomocou širokého spektra funkcií príslušnosti.
3. Fuzzy pravidlá a inferencia: Fuzzy Logic Toolbox podporuje vytváranie a modifikáciu fuzzy pravidiel, ako aj implementáciu fuzzy inferencie pomocou rôznych metód.
4. Defuzzifikácia: Knižnica ponúka rôzne metódy defuzzifikácie, vrátane centroidu, bisectra, strednej hodnoty maxima a ďalších.
5. Simulácia a validácia fuzzy systémov: Fuzzy Logic Toolbox umožňuje simuláciu a validáciu fuzzy systémov, čo pomáha pri návrhu a ladení týchto systémov.

4.6 Aforge.net

AForge.Net je open-source framework zameraný na vývoj aplikácií v oblasti počítačového videnia, umelej inteligencie a robotiky[28]. Tento framework je implementovaný v jazyku C#, čo umožňuje jeho integráciu do širokého spektra .NET aplikácií. Jeho implementácia do prostredia Visual studio je veľmi jednoduchá (pomocou NuGet správy balíčkov).

V praxi sa AForge.Net využíva vo veľmi rôznorodých aplikáciách. V oblasti fuzzy logiky, môže byť použitý na riešenie problémov, ktoré vyžadujú manipuláciu s nejasnosťami a neurčitosťami. Napríklad, môže byť použitý na návrh a implementáciu fuzzy regulátorov v robotike alebo na modelovanie neurčitostí v systémoch umelej inteligencie.

Napokon, je dôležité poznamenať, že AForge.Net je open-source projekt. To znamená, že jeho zdrojový kód je voľne dostupný a môže byť modifikovaný a rozširovaný podľa potrieb konkrétnej aplikácie. Toto poskytuje vývojárom možnosť prispôsobiť nástroj presne ich potrebám a prípadne prispieť k jeho ďalšiemu vývoju[28].

4.6.1 Funkcie a vlastnosti Aforge.net

AForge.Net obsahuje množstvo modulov a funkcií, ktoré pokrývajú rôzne aspekty počítačového videnia, umelej inteligencie a robotiky. V kontexte fuzzy logiky, AForge.Net poskytuje modul pre návrh a implementáciu fuzzy inferenčných systémov. Tento modul obsahuje nasledujúce vlastnosti[28]:

1. Definícia fuzzy množín: Knižnica poskytuje funkcie pre definovanie a manipuláciu s fuzzy množinami.
2. Fuzzy pravidlá a inferencia: AForge.Net umožňuje definovať a upravovať fuzzy pravidlá a implementovať fuzzy inferenciu.
3. Defuzzifikácia: Knižnica poskytuje metódy pre defuzzifikáciu fuzzy výstupov.

4.7 Tabuľka porovnania vlastností uvedených knižníc

Nasledujúca tabuľka obsahuje prehľadne všetky vlastnosti, ktoré knižnice/frameworky, spomenuté vyššie podporujú, prípadne nepodporujú. Zároveň tabuľka zobrazuje zložitosť pridania jednotlivých knižníc do projektov (v prípade nutnosti inštalovania ďalších nástrojov na kompiláciu knižnice sa jedná o vysokú zložitosť, v prípade jednoduchej inštalácie priamo z vývojového prostredia sa jedná o nízku zložitosť). Tabuľka bola vytvorená na základe informácií získaných v tejto kapitole.

Tabuľka 1. Vlastnosti knižníc na implementáciu fuzzy logiky

Knižnica	Open source	Použitie na mikropočítači	Vizuálne GUI	Zložitosť zahrnutia do projektu
eFLL	Áno	Áno	Nie	Nízka
fuzzylite	Áno	Áno	Áno	Vysoká
jfuzzylite	Áno	Áno	Áno	Vysoká
Scikit-fuzzy	Áno	Nie	Nie	Nízka
Fuzzy logic toolbox	Nie	Nie	Áno	Nízka
Aforge.net	Áno	Nie	Nie	Nízka

5 PRÍPRAVA NA VYTVORENIE UKÁŽKOVEJ APLIKÁCIE

V nasledujúcej kapitole praktickej časti sa práca bude v prvom rade venovať výberu najvhodnejšej knižnice pre implementáciu fuzzy logiky v ukážkovej aplikáciu. V ďalších častiach sa práca venuje výberu vhodného mikropočítača a vývojového prostredia na implementáciu aplikácie. Popis a dokumentácia samotnej ukážkovej aplikácie je dostupný v nasledujúcej kapitole.

5.1 Výber knižnice

V tejto podkapitole sa práca venuje výberu čo najvhodnejšej knižnice na implementáciu ukážkovej aplikácie. Výber vhodnej knižnice na implementáciu fuzzy logiky do embeddded systémov je jednou z kľúčových súčastí pre tvorbu praktickej časti práce.

Výber vhodnej knižnice bol z troch knižníc, ktoré boli spomenuté a aj popísané v záverečnej kapitole teoretickej časti. Rozhodovanie teda prebiehalo spomedzi týchto knižníc:

- eFLL (Embedded fuzzy logic library)
- Fuzzylite
- jFuzzylite

Všetky z vymenovaných knižníc spĺňali podmienku toho, že sa dajú použiť pre implementáciu projektov v mikropočítačoch. Bolo teda nutné zvážiť všetky výhody a nevýhody, ktorými dané knižnice disponujú.

Knižnica jFuzzylite by nebola vhodnou voľbou, keďže určité množstvo mikroprocesorov na trhu nepodporuje jazyk Java. Hoci je Java v istých aplikáciách efektívna, jazyk C prípadne C++ je efektívnejší na programovanie embedded systémov. Jazyk C je celkovo aplikovaný v 80% embedded zariadeniach [38].

Rozhodovanie teda prebiehalo ďalej medzi knižnicami fuzzylite a eFLL, kde bolo nakoniec rozhodnuté najmä z dôvodu jednoduchej inštalácie a práce z knižnicou použiť eFLL.

Využitím knižnice eFLL v tejto bakalárskej práci je očakávaná efektívna implementácia fuzzy ukážkovej aplikácie, ktorá bude schopná adaptovať sa na rôzne podmienky a zabezpečiť optimálne riadenie procesu.

Na záver je nutné podotknúť, že výber vhodnej knižnice pre prácu s fuzzy logikou na mikropočítači je veľmi dôležitý a značne, či už v prípade ukážkovej aplikácie, ale aj vo všeobecnosti znižuje zložitosť a čas potrebný na naprogramovanie správnej funkcionality projektu.

5.2 Výber mikropočítača

Výber mikropočítača je kľúčovou súčasťou návrhu a vývoja hardvérových projektov. Mikropočítač predstavuje jadro zariadenia, ktoré riadi a spracováva všetky údaje a informácie. Pri výbere mikropočítača je dôležité zvážiť niekoľko faktorov, vrátane výkonnosti, spotreby energie, dostupnosti programovacích nástrojov a podpory komunity.

V rámci praktickej časti tejto bakalárskej bolo vybrané použitie mikropočítača ESP32 (DevKit C) od firmy Espressif[40]. Tento mikropočítač bol vybraný na základe jeho výkonnosti, širokej podpory, flexibilnosti a nízkej ceny.

ESP32 je 32-bitový mikroprocesor, ktorý disponuje dvoma jadrami Xtensa LX6 s maximálnou frekvenciou až 240 MHz[41]. Medzi jeho hlavné výhody patrí podpora bezdrôtových komunikačných štandardov Wi-Fi a Bluetooth, vysoká výkonnosť, malá spotreba energie a veľký počet periférií[40][41][42]. ESP32 tiež podporuje širokú škálu programovacích jazykov, vrátane C/C++ a MicroPython, čo umožňuje flexibilné programovanie a rýchly vývoj aplikácií[40][41][42]. Ďalšími výhodami sú nízka cena, mnoho vývojových nástrojov a veľká komunita spolu s rozsiahlou dokumentáciou[41][42][43].

Na porovnanie, iné populárne mikropočítače, ako sú Arduino UNO (ATmega328P)[44][45] alebo Raspberry Pi Pico (RP2040)[46], majú tiež svoje silné stránky, ale v určitých oblastiach sa nemožno vyrovnať ESP32. Napríklad Arduino UNO je jednoduché na použitie a má silnú podporu komunity, avšak jeho výkonnosť a funkčnosť je obmedzená, a nie je schopný podporovať Wi-Fi alebo Bluetooth bez externých modulov[45]. Na druhej strane, Raspberry Pi Pico má výkonný dvojjadrový mikroprocesor a podporuje MicroPython, avšak opäť chýba mu natívna podpora pre Wi-Fi a Bluetooth[46].

Na záver, výber ESP32 ako mikroprocesora pre túto prácu bol motivovaný jeho vysokou výkonnosťou, širokou funkcionalitou, podporou bezdrôtových komunikačných štandardov, podporou viacerých programovacích jazykov, aktívnou komunitou a prijateľnou cenou. Tieto vlastnosti ho robia ideálnym kandidátom pre vývoj a realizáciu ukážkového projektu v praktickej časti.

5.3 Výber vývojového prostredia

Vo svete mikrokontrolérov a vývojových dosiek je k dispozícii množstvo vývojových prostredí (IDE), ktoré poskytujú sadu nástrojov na tvorbu, testovanie a ladenie softvéru. Pri výbere vhodného prostredia bakalársku prácu bolo zvážených niekoľko kľúčových aspektov, ako sú dostupnosť, použiteľnosť, kompatibilita a podpora jazyka.

Hlavnými kandidátmi boli Arduino IDE[47], PlatformIO[48], Atmel Studio[49] a MicroPython[50]. Atmel Studio je robustné a profesionálne IDE, ktoré poskytuje podrobné nástroje na ladenie a programovanie mikrokontrolérov od spoločnosti Atmel, vrátane tých, ktoré sú na doskách Arduino[49]. PlatformIO je výkonné open-source prostredie, ktoré je viac modulárne a flexibilnejšie ako Arduino IDE[48]. MicroPython je interpretovaný jazyk, ktorý umožňuje programovať mikrokontroléry v jazyku Python, čo by mohlo byť atraktívne pre tých, ktorí dávajú prednosť tomuto jazyku[50].

Napriek silným stránkam ostatných prostredí bude v praktickej časti práce použité Arduino IDE[47] z dôvodu jeho jednoduchosti, dostupnosti a širokého prijatia v komunite[51]. Arduino IDE je open-source a multiplatformové vývojové prostredie, ktoré je kompatibilné s množstvom mikrokontrolérov a podporuje vývojové dosky Arduino. Svojou jednoduchosťou a intuitívnym rozhraním je vhodné pre začiatočníkov, ale zároveň poskytuje dostatočnú flexibilitu pre pokročilých užívateľov. Taktiež Arduino IDE podporuje použitie už spomínaného mikroprocesoru ESP32 a knižnice eFLL[35][47]. Používa upravený jazyk C++ (inak nazývaný Arduino), ktorý je široko používaný v priemysle a preto poskytuje dobrú základňu pre ďalšie štúdium a prácu v oblasti embedded systémov. Napriek tomu, že niektoré pokročilejšie funkcie a nástroje, ktoré sú k dispozícii v Atmel Studiu alebo PlatformIO, v Arduino IDE chýbajú, pre účely praktickej časti práce je Arduino IDE dostatočné a efektívne[47].

Arduino IDE má tiež niektoré nevýhody, akými sú napríklad obmedzené možnosti ladenia, menšiu flexibilitu v porovnaní s niektorými pokročilejšími prostrediami alebo relatívne pomalšiu rýchlosť kompilácie[47]. Avšak v kontexte práce, tieto nevýhody neboli dostatočne významné na to, aby zmenili finálne rozhodnutie.

V konečnom dôsledku, výber vývojového prostredia závisí na konkrétnych potrebách a cieľoch projektu. V tomto prípade bolo rozhodnuté, že Arduino IDE je najvhodnejšie pre praktickú časť práce vzhľadom na jeho jednoduchosť, podporu širokého spektra hardvéru a silnú komunitnú podporu. Je ale pravda, že v budúcnosti by bolo vhodné zvážiť prechod na

pokročilejšie prostredie, ak by sa zmenili potreby práce alebo v prípade vývoja pokročilejšieho systému.

5.4 Zhrnutie výberu

Pre ukážkovú aplikáciu bol teda vybraný mikropočítač ESP32 (Devkit C), vývojové prostredie Arduino IDE a knižnica pre implementáciu fuzzy logiky eFLL.

6 UKÁŽKOVOVÁ APLIKÁCIA

Táto kapitola obsahuje popis a detailnú dokumentáciu k vytvoreniu, a implementácií ukážkovej aplikácie – fuzzy teplomera. Na záver kapitola obsahuje porovnanie s použitím binárnej logiky a možnosti rozšírenia aplikácie v budúcnosti.

6.1 Štruktúra aplikácie

Aplikácia je založená na mikropočítači ESP32 (DevKitC), ktorý je srdcom celého systému. Ako už bolo spomenuté, mikropočítač ESP32 bol vybraný pre jeho vysoký výkon, nízku spotrebu energie a schopnosť komunikovať s množstvom rôznych typov periférií. Celý projekt je implementovaný v prostredí Arduino IDE, ktoré umožňuje jednoduché programovanie a ladenie mikroprocesorov ESP32.

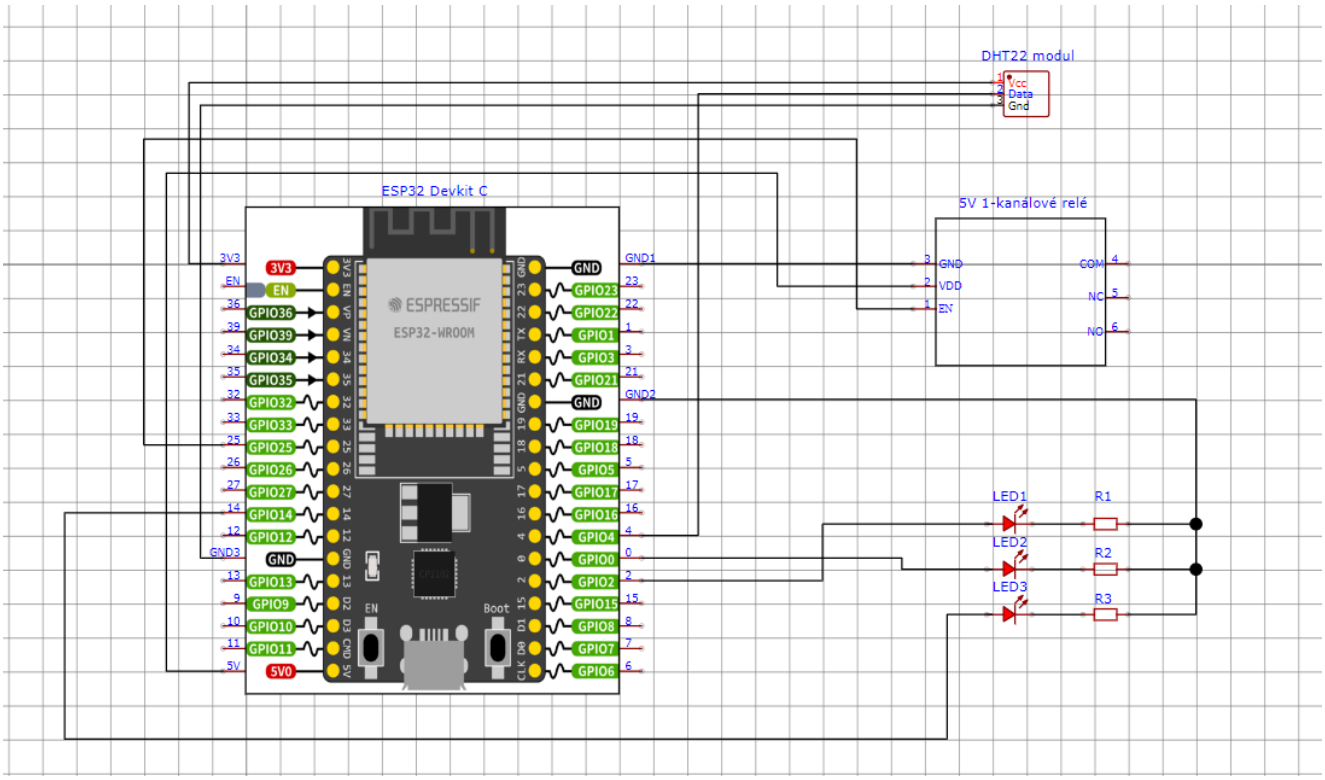
6.2 Súpis komponentov

Pred začiatkom implementácie projektu je potrebné zabezpečiť všetky potrebné komponenty. Nasledujúci zoznam obsahuje všetky súčiastky použité v tomto projekte:

- ESP32 DevKitC mikropočítač: Toto je hlavný riadiaci prvok projektu, ktorý umožňuje komunikáciu s ostatnými súčiastkami a riadenie ich funkcií.
- DHT22 teplotný senzor: Tento senzor je použitý na meranie teploty prostredia.
- 5V relé modul: Tento modul je použitý na otvorenie a uzatvorenie obvodu v závislosti od výsledkov merania teploty.
- Breadboard: Breadboard je použitý na fyzické zapojenie všetkých súčiastok projektu.
- Vodiče: Sú použité na spojenie jednotlivých komponentov na breadboard.
- LED diódy: Tri LED diódy sú použité na vizuálnu indikáciu teplotných stavov - horúco, studené a ideálne.
- 75 Ω rezistory: Tieto rezistory sú použité na obmedzenie prúdu pretekajúceho LED diódami. Hodnota rezistorov bola doporučená výrobcom na použitie s LED diódami pri napätí 3.3 V[53].

6.3 Schéma zapojenia

Prvým krokom pri realizácii projektu je vytvorenie vhodnej schémy zapojenia. Schéma zapojenia je esenciálny nástroj pre pochopenie a vizualizáciu, ako sú jednotlivé komponenty projektu spojené.



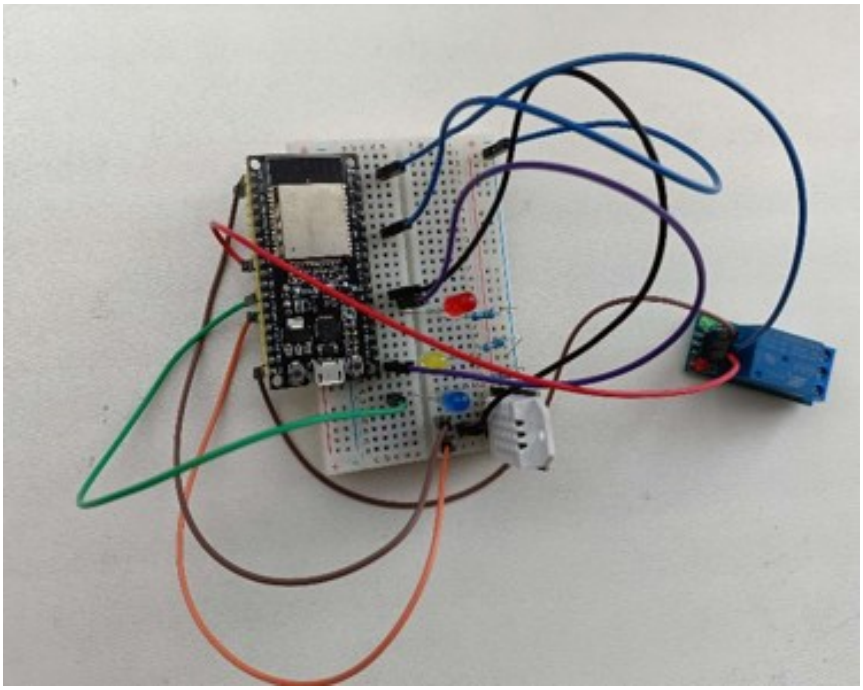
Obrázok 12. Schéma zapojenia

6.4 Fyzické zapojenie

Nasledujúci postup popisuje zapojenie komponentov na breadboard a ich pripojenie k ESP32:

1. Pripojenie ESP32 na breadboard: ESP32 DevKitC je vložený na krajnú časť poľa tak, aby bolo možné pristupovať k pinom na pravej strane ESP32 priamo z breadboardu a k ostatným pinom je zabezpečený prístup spojením cesty na breadboarde s daným pinom alebo priamym zapojením k súčiastkam.

2. Pripojenie DHT22 na breadboard: DHT22 je umiestnený na breadboard tak, aby nevznikali kolízie s ostatnými komponentmi. Je umiestnený s pravej dolnej časti poľa.
3. Pripojenie DHT22 na ESP32: Dátový pin DHT22 je pripojený na GPIO pin 4 na ESP32. Napájanie pre DHT22 je zabezpečené pripojením jeho VCC pinu na 3.3V výstup na ESP32 a GND pinu na GND na ESP32.
4. Pripojenie LED diód: Anódy troch LED diód sú pripojené na GPIO piny 2, 0 a 14 na ESP32. Katódy LED diód sú pripojené na 75 ohmové rezistory, ktoré sú potom pripojené na GND na ESP32.
5. Pripojenie relé modulu: VCC pin relé modulu je pripojený na 5V výstup na ESP32 a GND pin je pripojený na GND na ESP32. Riadiaci pin relé je pripojený na GPIO pin 25 na ESP32.



Obrázok 13. Fyzické zapojenie teplomera

6.5 Zdrojový kód

Zdrojový kód pre projekt bol vytvorený v prostredí Arduino IDE s použitím knižníc eFLL a DHT senzor library. Kód definuje fuzzy pravidlá pre rozhodovanie medzi studenou, ideálnou a horúcou teplotou. Zdrojový kód je súčasťou prílohy bakalárskej práce.

Tento kód je napísaný v jazyku Arduino, ktorého základ je v jazyku C/C++. Základným princípom je získať teplotné dáta zo senzora DHT22, spracovať ich pomocou fuzzy logiky a na základe toho ovládať relé a LED diódy.

Kód začína deklarovaním a inicializáciou potrebných knižníc, konštánt a premenných.

Funkcia setup() sa zavolá pri štarte programu a inicializuje sériovú komunikáciu, DHT senzor a nastavuje LED diódy ako výstupy. Taktiež sa tu vytvárajú fuzzy premenné a pravidlá.

Funkcia loop() sa vykonáva nepretržite počas behu programu. Číta teplotu z DHT22 senzora a v prípade platnej hodnoty ju poskytuje ako vstup do fuzzy systému. Na základe výstupu z fuzzy systému sa potom rozhoduje, či sa má zapnúť relé a ktorá LED dióda sa má rozsvietiť.

V podstate máme tri hlavné časti kódu:

1. Inicializácia: Táto časť kódu definuje všetky potrebné premenné a inicializuje systém. To zahŕňa inicializáciu DHT22 senzora, nastavenie GPIO pinov na výstup pre LED diódy a vytvorenie fuzzy logického systému pre rozhodovanie.
2. Nastavenie fuzzy pravidiel: V tejto časti kódu sa definujú fuzzy množiny a pravidlá. V našom prípade máme tri fuzzy množiny pre teplotu (nízka, ideálna, vysoká) a dve fuzzy množiny pre akciu (robiť nič, zapnúť). Pravidlá potom definujú, aká akcia sa má vykonať pre danú teplotu.
3. Spracovanie teplotných dát a rozhodovanie: Táto časť kódu zahŕňa čítanie dát z DHT22 senzora a ich spracovanie prostredníctvom fuzzy logického systému. Na základe výstupu z tohto systému sa potom rozhoduje, ktorá LED dióda sa má rozsvietiť a či sa má zapnúť relé.

Jednotlivé časti kódu sú podrobnejšie popísané v nasledujúcej časti.

6.5.1 Inicializácia

Začína sa definovaním a inicializáciou knižníc, konštánt a objektov potrebných pre prácu s DHT22 senzorom a fuzzy logikou.

delayMS = 5000; je použitý vo funkcií loop() na oneskorenie medzi meraniami. V tomto prípade sa jedná o odchýlku 5 sekúnd po každom meraní.

DHT_Unified dht(DHTPIN, DHTTYPE); vytvorí inštanciu senzora DHT s pinom a typom senzora definovanými konštantami *DHTPIN* a *DHTTYPE*.

Fuzzy fuzzy = new Fuzzy();* inicializuje nový fuzzy logický systém.

6.5.2 Inicializácia LED diód, relé, rýchlosti komunikácie na sériovom porte a spustenie senzoru DHT22

Serial.begin(9600); slúži na nastavenie rýchlosti sériového portu. Bez tohto základného príkazu by nebolo možné po skompilovaní tento zdrojový kód správne nahrať do mikroprocesora.

dht.begin(); spustí snímač teploty DHT22.

Príkazy *pinMode(x,y)* slúžia na nastavenie požadovaných pinov na vstup alebo výstup. Prvý parameter obsahuje číslo pinu, ktoré je v tomto prípade makro, definované v inicializácií. Druhý parameter obsahuje hodnotu v podobe makra, ktorá určuje, či je pin nastavený na vstup alebo výstup.

```
Serial.begin(9600);
dht.begin();

pinMode(RELAY_PIN, OUTPUT);
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
```

Obrázok 14. Nastavenie pinov LED diód, relé, rýchlosť komunikácie na sériovom porte a spustenie senzoru DHT22

6.5.3 Inicializácia fuzzy množín

Jednou z úvodných častí zdrojového kódu je inicializácia vstupných a výstupných fuzzy množín.

Inicializácia vstupných fuzzy množín

V tejto časti kódu je vytvorený nový vstupný fuzzy objekt *temp* reprezentujúci teplotu. Tento objekt má tri fuzzy množiny: *low*, *ideal* a *high*. Každá množina je definovaná štyrmi bodmi,

ktoré reprezentujú hranice a vrcholy trojuholníkových funkcií príslušnosti. Tieto množiny sú potom pridané do objektu *temp* pomocou metódy *addFuzzySet()*. Nakoniec je *temp* pridaný do hlavného fuzzy systému pomocou metódy *addFuzzyInput()*. Hodnoty vo vstupných množinách znamenajú nasledovné:

- Low: (0, 0, 15, 20) - teplota je "nízka", ak je menšia alebo rovná 15 stupňom. Ak je medzi 15 a 20 stupňami, je stále považovaná za nízku, ale s nižším stupňom príslušnosti. Nad 20 stupňov nie je považovaná za nízku.
- Ideal: (18, 22, 24, 26) - teplota je "ideálna", ak je medzi 22 až 24 stupňami. Ak je medzi 18 a 22 alebo medzi 24 a 26, je stále považovaná za ideálnu, ale s nižším stupňom príslušnosti. Mimo týchto hodnôt nie je považovaná za ideálnu.
- High: (24, 30, 35, 35) - teplota je "vysoká", ak je rovná alebo vyššia ako 30 stupňov. Ak je medzi 24 a 30 stupňami, je stále považovaná za vysokú, ale s nižším stupňom príslušnosti. Pod 24 stupňov nie je považovaná za vysokú.

```
FuzzyInput* temp = new FuzzyInput(1);

FuzzySet* low = new FuzzySet(0, 0, 15, 20);
temp->addFuzzySet(low);
FuzzySet* ideal = new FuzzySet(18, 22, 24, 26);
temp->addFuzzySet(ideal);
FuzzySet* high = new FuzzySet(24, 30, 35, 35);
temp->addFuzzySet(high);

fuzzy->addFuzzyInput(temp);
```

Obrázok 15. Inicializácia vstupných fuzzy množín

Inicializácia výstupných fuzzy množín

Podobne ako pri vstupných množinách, vytvárame nový výstupný fuzzy objekt *action*, ktorý má dve fuzzy množiny: *doNothing* a *turnOn*. Tieto množiny sú definované rovnako ako vstupné množiny, a pridané do objektu *action* pomocou metódy *addFuzzySet()*. Nakoniec je action pridaný do hlavného fuzzy systému pomocou metódy *addFuzzyOutput()*. Hodnoty vo výtupných množinách znamenajú nasledovné:

- doNothing: (0, 0, 45, 50) - akcia "nerob nič" je vybraná, ak je výstup menší alebo rovný 45. Ak je medzi 45 a 50, je stále považovaná za možnú, ale s nižším stupňom príslušnosti. Nad 50 už nie je možnosť "nerob nič" považovaná za vhodnú.
- turnOn: (50, 55, 100, 100) - akcia "zapni" je vybraná, ak je výstup rovný alebo vyšší ako 100. Ak je medzi 55 a 100, je stále považovaná za možnú, ale s nižším stupňom príslušnosti. Pod 55 už nie je možnosť "zapni" považovaná za vhodnú.

```
FuzzyOutput* action = new FuzzyOutput(1);

FuzzySet* doNothing = new FuzzySet(0, 0, 45, 50);
action->addFuzzySet(doNothing);
FuzzySet* turnOn = new FuzzySet(50, 55, 100, 100);
action->addFuzzySet(turnOn);

fuzzy->addFuzzyOutput(action);
```

Obrázok 16. Inicializácia výstupných fuzzy množín

6.5.4 Nastavenie fuzzy pravidiel

Fuzzy pravidlá sú definované v *setup()* funkcii. Pre každý vstup (v tomto prípade teplota) a výstup (akcia) sa vytvárajú fuzzy množiny, ktoré reprezentujú rôzne hodnoty vstupu a výstupu. Napríklad pre teplotu máme množiny 'low', 'ideal' a 'high'.

Potom sa vytvárajú fuzzy pravidlá, ktoré definujú, aká akcia sa má vykonať pri danej teplote. Napríklad, pravidlo "IF temperature ISlow THEN action IS ‚turnOn‘" hovorí, že ak je teplota nízka, systém by mal zapnúť relé.

Všetky tieto pravidlá sú pridané do fuzzy systému pomocou funkcie *addFuzzyRule()*. Každé pravidlo je identifikované unikátnym číslom, ktoré je prvým argumentom v konštruktore FuzzyRule.

Fuzzy pravidlá sú definované pomocou antecedentu (predpokladu) a konzekventu (výsledku). Antecedent je vstupná premenná a jej množina (v tomto prípade teplota a jej množina low, ideal alebo high), ktoré sú spojené dohromady pomocou metódy *joinSingle()*. Konzekvent je výstupná premenná a jej množina (v tomto prípade akcia a jej množina *doNothing* alebo *turnOn*), ktoré sú pridané pomocou metódy *addOutput()*.

Výsledné pravidlá teda definujú, ako systém reaguje na rôzne hodnoty teploty, a umožňujú fuzzy logike rozhodovať, kedy aktivovať relé a ktorú LED diódu rozsvietiť.

Prvé pravidlo je definované nasledovne:

```
// Vytvorenie FuzzyRule "IF temperature IS high THEN action IS doNothing"
FuzzyRuleAntecedent* ifTemperatureHigh = new FuzzyRuleAntecedent();
ifTemperatureHigh->joinSingle(high);
FuzzyRuleConsequent* thenDoNothing = new FuzzyRuleConsequent();
thenDoNothing->addOutput(doNothing);
FuzzyRule* fRule1 = new FuzzyRule(1, ifTemperatureHigh, thenDoNothing);
fuzzy->addFuzzyRule(fRule1);
```

Obrázok 17. Nastavenie fuzzy pravidla pre horúcu teplotu

Toto pravidlo definuje, že ak je teplota vysoká (definované v množine *high*), systém by mal nevykonať žiadnu akciu (definované v množine *doNothing*). V praxi to znamená, že ak je teplota príliš vysoká, relé zostane vypnuté a rozsvieti sa prvá LED dióda indikujúca horúcu teplotu.

Druhé pravidlo je podobné:

```
// Vytvorenie FuzzyRule "IF temperature IS ideal THEN action IS doNothing"
FuzzyRuleAntecedent* ifTemperatureIdeal = new FuzzyRuleAntecedent();
ifTemperatureIdeal->joinSingle(ideal);
FuzzyRuleConsequent* thenDoNothing2 = new FuzzyRuleConsequent();
thenDoNothing2->addOutput(doNothing);
FuzzyRule* fRule2 = new FuzzyRule(2, ifTemperatureIdeal, thenDoNothing2);
fuzzy->addFuzzyRule(fRule2);
```

Obrázok 18. Nastavenie fuzzy pravidla pre ideálnu teplotu

Ak je teplota ideálna (definované v množine *ideal*), systém by mal tiež nevykonať žiadnu akciu (definované v množine *doNothing*). V praxi to znamená, že ak je teplota ideálna, relé zostane vypnuté a rozsvieti sa druhá LED dióda indikujúca ideálnu teplotu.

Tretie pravidlo je mierne odlišné:

```
// Vytvorenie FuzzyRule "IF temperature IS low THEN action IS turnOn"
FuzzyRuleAntecedent* ifTemperatureLow = new FuzzyRuleAntecedent();
ifTemperatureLow->joinSingle(low);
FuzzyRuleConsequent* thenTurnOn = new FuzzyRuleConsequent();
thenTurnOn->addOutput(turnOn);
FuzzyRule* fRule3 = new FuzzyRule(3, ifTemperatureLow, thenTurnOn);
fuzzy->addFuzzyRule(fRule3);
```

Obrázok 19. Nastavenie fuzzy pravidla pre studenú teplotu

Ak je teplota nízka (definované v množine *low*), systém by mal vykonať akciu (definované v množine *turnOn*). V praxi to znamená, že ak je teplota nízka, relé sa aktivuje a rozsvieti sa tretia LED dióda indikujúca nízku teplotu.

6.5.5 Spracovanie teplotných dát a rozhodovanie

V *loop()* funkcii sa najprv číta teplota z DHT22 senzora. Ak teplota nebola nameraná, tak sa na výstupe vypíše chybová hláška. Ak bola nameraná, kód pokračuje vo vetve „else“. Táto teplota sa potom poskytuje ako vstup do fuzzy logického systému prostredníctvom *fuzzy->setInput(1, event.temperature);*. Systém potom vykoná fuzzy inferenciu pomocou *fuzzy->fuzzify();*.

Výstup z fuzzy systému sa potom defuzzifikuje (prevádza sa na konkrétnu hodnotu) pomocou: *float output = fuzzy->defuzzify(1);*.

```
// Oneskorenie medzi meraniami definované vyššie
delay(delayMS);

// Zisťovanie teploty
sensors_event_t event;
dht.temperature().getEvent(&event);

if (isnan(event.temperature)) {
  Serial.println("Error reading temperature!");
}
else {
  Serial.print("Temperature: ");
  Serial.print(event.temperature);
  Serial.println(" *C");

  // Spustenie fuzzy inferencie
  fuzzy->setInput(1, event.temperature);
  fuzzy->fuzzify();

  float output = fuzzy->defuzzify(1);
  Serial.println("Výsledok po defuzzifikácii: ")
  Serial.print(output);
```

Obrázok 20. Fuzzifikácia a defuzzifikácia teplotných dát

Na základe výstupu z fuzzy systému sa potom rozhoduje, ktorá LED dióda sa má rozsvietiť a či sa má zapnúť relé. Ak je výstup väčší ako 75.0, systém určuje, že je chladno, zapne relé a rozsvieti príslušnú LED diódu. Ak je výstup medzi 35.0 a 75.0, systém určuje, že je ideálna teplota, relé zostáva vypnuté a rozsvieti sa druhá LED dióda. Ak je výstup menší alebo rovný 35.0, systém určuje, že je horúco, relé zostáva vypnuté a rozsvieti sa prvá LED dióda.

```
// Rozhodnutie založené na fuzzy výstupe
if (output > 75.0) {
  Serial.println("\nCOLD\n");
  digitalWrite(RELAY_PIN, HIGH);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
}
else if (output > 35.0) {
  Serial.print("\nIdeal\n");
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
}
else {
  Serial.print("\nHOT\n");
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
}
```

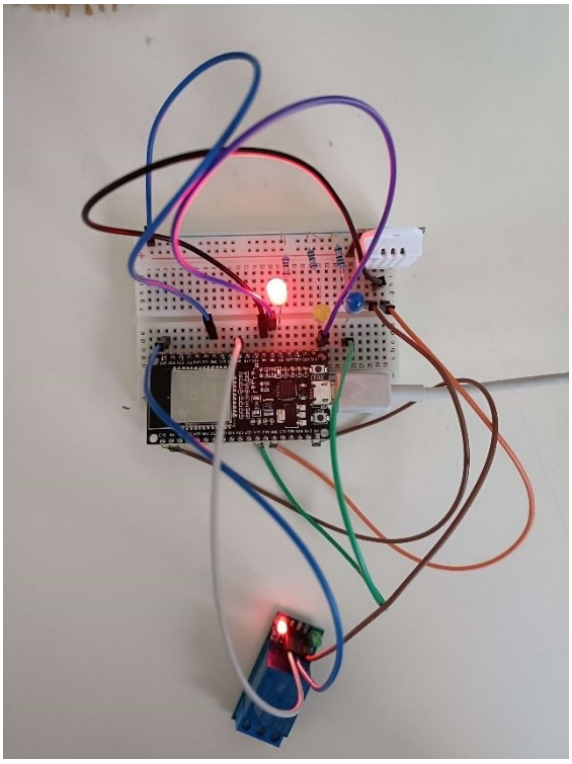
Obrázok 21. Rozhodovanie založené na fuzzy výstupe

Toto je hlavná logika, ktorá ovláda relé a LED diódy na základe teploty získanej z DHT22 senzora a spracovanej pomocou fuzzy logiky.

Celkový kód teda skombinuje hardvérové komponenty (DHT22 senzor, relé, LED diódy) a softvérové techniky (fuzzy logika) na vytvorenie systému, ktorý dokáže reagovať na zmeny teploty v prostredí. Výsledkom je dynamický systém, ktorý sa dokáže prispôsobiť rôznym podmienkam a reagovať na ne.

6.6 Testovanie a hodnotenie funkcionalít

Po úspešnom zapojení a naprogramovaní ESP32, bol projekt testovaní na funkčnosť. Senzor DHT22 bol porovnávaný s klasickým izbovým teplomerom a zistilo sa, že správne meria teplotu. Výstup po defuzzifikácií sa zobrazuje na sériový monitor v Arduino IDE spolu so skutočnou teplotou a druhom teploty (HOT, IDEAL, alebo COLD). Výstup na sériovom monitore bol porovnávaný so správaním sa zapojenia a bolo zistené, že zapojenie správne reaguje na zmeny teploty. Nasledujúce obrázky obsahujú výstupy na sériovom monitore spolu s výstupom na fyzickom zapojení.

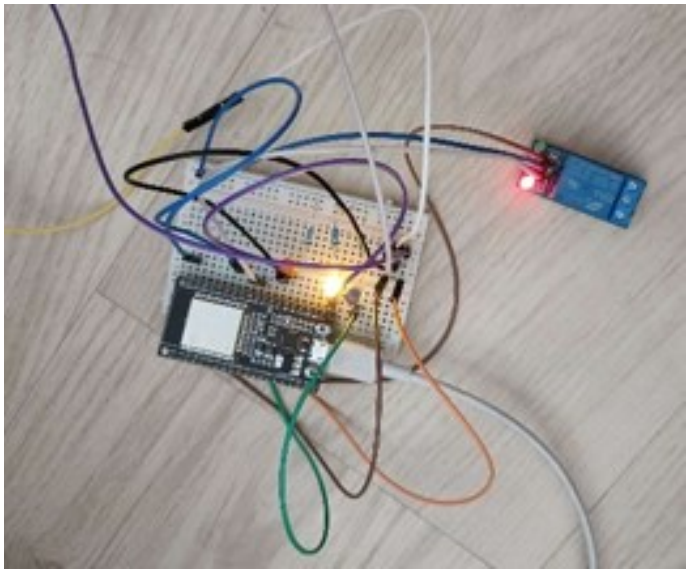


Obrázok 22. Výstup fyzického zapojenia pri horúcej teplote

```
Temperature: 27.10 *C
Výsledok po defuzzifikácií:
24.36
HOT
```

Obrázok 23. Výstup na sériovom monitore pri horúcej teplote

Ako je možné vidieť pri horúcej teplote sa rozsvieti červená LED dióda a relé ostáva rozopnuté (nesvieti zelená LED dióda na relé module). Teplota je podľa fuzzy logiky považovaná za horúcu približne od 24 °C.

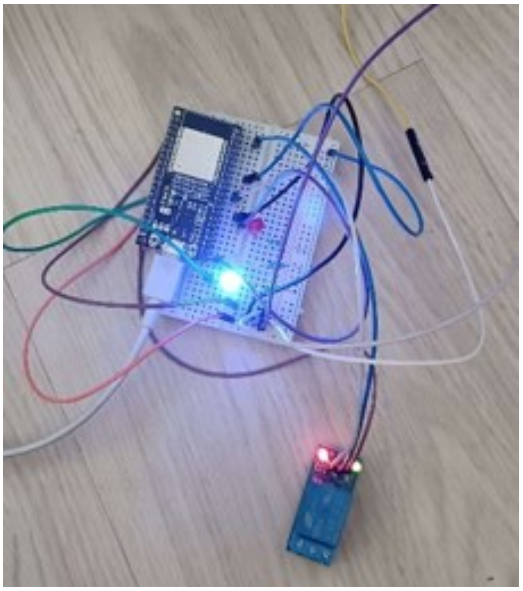


Obrázok 24. Výstup fyzického zapojenia pri ideálnej teplote

```
Temperature: 18.60 *C
Výsledok po defuzzifikácií:
57.64
Ideal
```

Obrázok 25. Výstup na sériovom monitore pri ideálnej teplote

Pri ideálnej teplote sa rozsvieti žltá LED dióda a relé ostáva rozopnuté. Teplota je považovaná za ideálnu približne od 17,5 do 24 °C. V prípade merania ideálnej a aj studenej teploty je senzor DHT22 umiestnený na chladnejšom mieste.



Obrázok 26. Výstup fyzického zapojenia pri studenej teplote

```
Temperature: 15.80 *C
Výsledok po defuzzifikácií:
76.03
COLD
```

Obrázok 27. Výstup na sériovom monitore pri studenej teplote

V prípade studenej teploty na fyzickou zapojení svieti modrá LED dióda a relé modul je zopnutý (indikuje zelená LED dióda na relé module). Teplota je považovaná za studenú približne do hodnoty 17,5 °C.

6.7 Porovnanie s použitím binárnej logiky

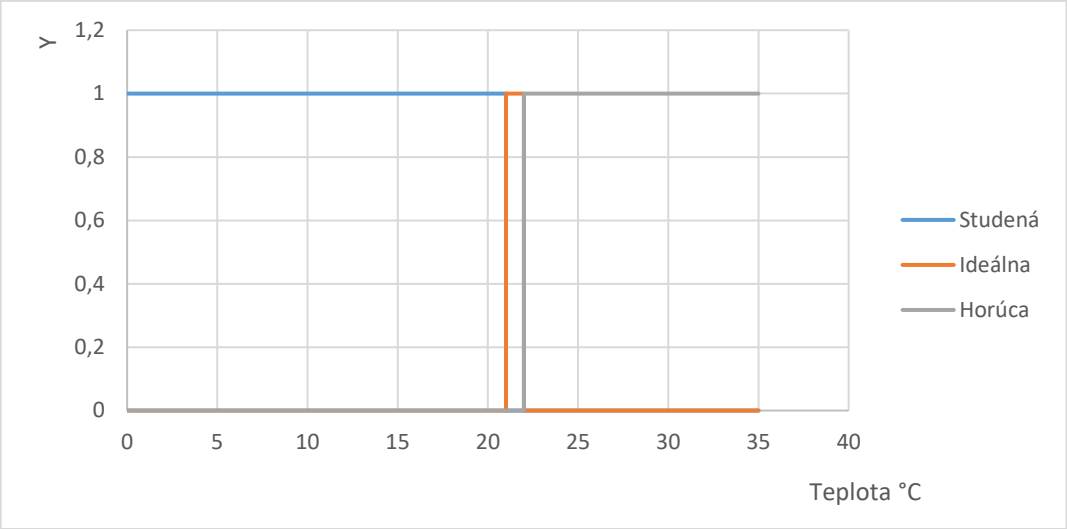
Aplikáciu teplomeru by bolo taktiež možné vytvoriť aj pomocou klasickej binárnej logiky. Avšak by mala táto aplikácia viaceré obmedzenia najmä v prípade rozšírenia aplikácie o pripojenie k ohrievaču, alebo klimatizácií s viacerými stupňami ohrevu/výkonu. Táto podkapitola sa venuje porovnaniu fuzzy teplomeru a klasického binárneho teplomeru. Taktiež sú v podkapitole zostrojené grafy definície teploty podľa fuzzy logiky a podľa binárnej logiky na lepšie pochopenie limitácií binárnej logiky oproti fuzzy logike.

6.7.1 Graf binárnej logiky

Ako už bolo niekoľkokrát spomenuté binárna logika pracuje iba s absolútnymi hodnotami 0 a 1. To znamená, že teplota môže byť definovaná iba ako:

- 100% studená, 0% ideálna a 0% horúca
- 0% studená, 100% ideálna a 0% horúca
- 0% studená, 0% ideálna a 100% horúca

V grafe sa teda funkcie príslušnosti teplôt nikdy nebudú prekrývať, lebo funkcie príslušnosti neklesajú lineárne. Ak by sa považovala ideálna teplota medzi 21 a 22 °C, graf by vypadal nasledovne:

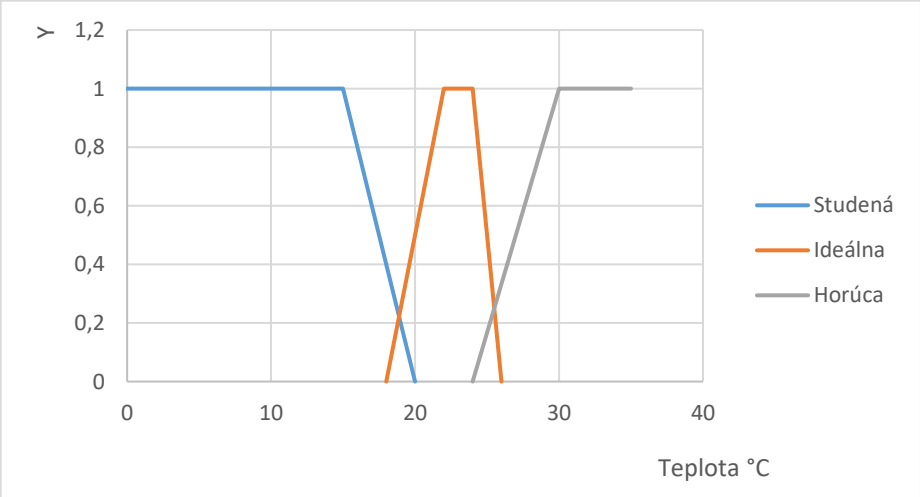


Obrázok 28. Graf teplôt teplomera s použitím binárnej logiky

Je možné vidieť, že studená teplota je pri teplotách 0 – 21 °C, ideálna 21 – 22 °C a horúca 22 – 35 °C a teploty sa neprekrývajú (ak sa nerátajú hodnoty 0).

6.7.2 Graf fuzzy logiky

Nasledujúci graf bol zostrojený podľa vstupných množín, ktoré boli použité v kóde fuzzy teplomeru. Graf je veľmi podobný ako graf obrázku číslo 1.



Obrázok 29. Graf teplôt fuzzy teplomera

Je možné vidieť, že funkcie príslušnosti sa prekrývajú, konkrétne ideálna teplota s horúcou a ideálna teplota so studenou.

6.7.3 Výhody použitia fuzzy logiky v ukážkovej aplikácií

Najväčšou výhodou použitia fuzzy logiky oproti binárnej logiky je pri prekrývaní funkcií príslušnosti teplôt. Fuzzy logika dokáže na základe lingvisticky definovaných pravidiel rozhodnúť, či je potrebné zopnúť relé, alebo nie. Túto funkcionalitu je možné implementovať pomocou binárnej logiky, napríklad pomocou vetvenia programu (podmienky *IF*). Na druhej strane by pri rozšírení aplikácie, napríklad o ohrievač s rôznymi stupňami výkonu ohrevu došlo k zložitej implementácií úrovní výkonu do kódu, zatiaľ čo pri použití fuzzy logiky by stačilo pridať nové fuzzy množiny pre výstup, pomocou ktorých by sa výkon ohrievača zvyšoval, alebo znižoval podľa potreby. Týmto spôsobom by taktiež nedochádzalo k zbytočnému opotrebovávaniu relé z dôvodu častého spínania a rozopínania.

6.8 Zhodnotenie a možnosti rozšírenia

Na základe vykonaných testov aplikácia ukázala správnu funkcionalitu pri ideálnej, studenej a horúcej teplote.

V budúcnosti je možné projekt rozšíriť o ďalšie funkcie, ako napríklad možnosť pripojenia ohrievača, alebo klimatizácie, ktorá v prípade horúcej teploty bude chladiť a v prípade studenej teploty ohrievať. Ďalej by bolo možné rozšíriť aplikáciu o integráciu s webovým rozhraním pre vzdialený monitoring a ovládanie, alebo pridanie ďalších senzorov pre sledovanie vlhkosti, svetelnosti a iných environmentálnych faktorov.

ZÁVĚR

Bakalárska práca bola venovaná využitiu fuzzy logiky na mikropočítači. Cieľom práce bolo zhodnotiť aplikáciu a efektívnosť fuzzy logiky na mikropočítačoch a overiť vlastnosti knižníc na implementáciu fuzzy logiky na mikropočítačoch.

V teoretickej časti je v prvom rade definovaná fuzzy logika obecne, spolu so základnými charakteristikami a pojmami. V ďalšej časti sa práca venuje využitiu fuzzy logiky na mikropočítačoch. Sú tu uvedené aj niektoré konkrétne príklady, napríklad klimatizácie, domáca asistencia, alebo automobily. Použitie fuzzy logiky v týchto odvetviach zvyšuje bezpečnosť a komfort ľudí. Na záver teoretickej časti sú uvedené programovacie jazyky, v ktorých je možné fuzzy logiku implementovať.

Praktická časť sa úvodnej časti venovala popisom niekoľkých knižníc na implementáciu fuzzy logiky, medzi ktorými boli aj knižnice pre implementáciu fuzzy logiky na mikropočítačoch. Konkrétne sa jedná o knižnice fuzzylite, jfuzzylite a eFLL. Pre spomenuté knižnice boli v rámci overovania vlastnosti vytvorené ukážkové kódy, na základe sa hodnotili ich vlastnosti a použiteľnosť. Ako posledné bola ku knižniciam vypracovaná tabuľka vlastností, pomocou ktorej boli zhrnuté najdôležitejšie vlastnosti knižníc.

Ako poslednému sa práca venovala vytvoreniu ukážkovej aplikácií fuzzy logiky na mikropočítači – fuzzy teplomeru. V prvom rade bolo nutné vybrať vhodnú knižnicu na základe už získaných poznatkov z predchádzajúcej kapitoly. Ďalej sa muselo vybrať vhodné vývojové prostredie, konkrétne Arduino IDE a mikropočítač – ESP32 (Devkit C). Následne bolo potrebné vybrať ďalšie komponenty (LED diódy, rezistory, senzor, relé modul, breadboard) na úspešné zapojenie projektu. Potom bola vytvorená schéma zapojenia, podľa ktorej bolo v ďalšom kroku vytvorené fyzické zapojenie. V ďalšom kroku bol vytvorený zdrojový kód s použitím knižnice eFLL, v ktorom sa na základe vstupných teplôt rozhodovalo medzi studenou, ideálnou a horúcou teplotou za pomoci fuzzy logiky. Aplikácia bola testovaná, kde sa hodnotilo správne správanie sa pri rôznych teplotách. Bolo zistené, že aplikácia reaguje správne na rôzne druhy teplôt. V poslednom rade bola aplikácia porovnávaná s použitím binárnej logiky s dôrazom na výhody fuzzy logiky oproti binárnej logike.

Celkovo, táto práca poskytuje prehľad k pochopeniu fuzzy logiky, jej praktických aplikácií, a jej výhod v porovnaní s binárnou logikou. Taktiež práca ponúka hodnotenie knižníc pre implementáciu fuzzy logiky na mikropočítačoch.

SEZNAM POUŽITÉ LITERATURY

[1] ZADEH, L.A. Fuzzy sets. Information and Control. 1965, 8(3), 338-353.

[2] KLIR, George J. Fuzzy Sets and Fuzzy Logic Theory and Applications. New Jersey: Prentice Hall, 1995. ISBN 0-13-101171-5.

[3] ROSS, Timothy J. Fuzzy logic with engineering applications. 3rd ed. Chichester: Wiley, c2010. ISBN 978-0-470-74376-8.

[4] Fuzzy Logic - Applications [online]. [cit. 2023-05-03]. Dostupné z: tutorialspoint.com/fuzzy_logic/fuzzy_logic_applications.htm

[5] MAMDANI, Ebrahim H. a Srinivas ASSILIAN. An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies, 1975, 7(1), 1-13.

[6] TAKAGI, T. a M. SUGENO. Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics, 1985, 15(1), 116-132.

[7] MENDEL, Jerry M. Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Upper Saddle River, New Jersey 07458: Prentice Hall PTR, 2001. ISBN 0-13-040969-3.

[8] JURA, Pavel. Základy fuzzy logiky pro řízení a modelování. Brno: Vysoké učení technické v Brně, Nakladatelství VUTIUM, 2003. ISBN 8021422610.

[9] DRIANKOV, Dimiter. An Introduction to Fuzzy Control. Springer, 1996, 332 s. ISBN 3642082343.

[10] IBRAHIM, Ahmad. Fuzzy Logic for Embedded Systems Applications. Vyd. 1. Newnes, 2003. ISBN 9780080469904.

[11] PEDRYCZ, Witold. Fuzzy control and fuzzy systems. 2nd, extended ed. Somerset: Research Studies Press, 1993. ISBN 978-0-86380-131-0.

[12] PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN - technická literatura, 2004, 159 s. ISBN 80-7300-110-1.

[13] WHITE, Elecia. Making embedded systems. Sebastopol: O´Reilly, c2012. ISBN 9781449302146.

[14] Fuzzy Logic - Control System [online]. [cit. 2023-05-04]. Dostupné z: https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_control_system.htm

[15] About Python [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.python.org/about/

[16] Scikit-Fuzzy [online]. 2012 [cit. 2023-05-07]. Dostupné z: https://pythonhosted.org/scikit-fuzzy/overview.html

[17] PyFuzzy [online]. 2009 [cit. 2023-05-07]. Dostupné z: https://pypi.org/project/pyfuzzy/

[18] MathWorks: Matlab [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.mathworks.com/products/matlab.html

[19] MathWorks: Fuzzy Logic Toolbox [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.mathworks.com/products/fuzzy-logic.html

[20] Java [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.java.com/en/download/help/whatis_java.html

[21] JFuzzyLogic [online]. 2013 [cit. 2023-05-07]. Dostupné z: https://jfuzzylogic.sourceforge.net/html/about.html

[22] Juzzy - A Java based toolkit for Type-2 Fuzzy Logic [online]. 2013 [cit. 2023-05-07]. Dostupné z: https://ieeexplore.ieee.org/document/6613298

[23] About C++ [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://isocpp.org/about

[24] FuzzyLite [online]. 2010 [cit. 2023-05-07]. Dostupné z: https://fuzzylite.com/

[25] The R Project for Statistical Computing [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.r-project.org/

[26] MILLO, Giovanni a Gianfranco PIRAS. Spatial Panel Data Models in R [online]. 2012, 17.4., 47(1), 1-30 [cit. 2023-05-07]. Dostupné z: doi:10.18637/jss.v047.i01

[27] C# [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://dotnet.microsoft.com/en-us/languages/csharp

[28] Aforge.Net Framework [online]. 2012 [cit. 2023-05-07]. Dostupné z: http://www.aforgenet.com/framework/

[29] DotFuzzy [online]. 2008 [cit. 2023-05-07]. Dostupné z: https://github.com/MicheleBertoli/DotFuzzy

[30] JavaScript [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.javascript.com/

[31] Ruby [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.ruby-lang.org/en/

[32] Fuzzy-Logic [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://www.ruby-doc.info/gems/fuzzy-logic/0.0.2

[33] Lisp (programming language) [online]. 2023 [cit. 2023-05-07]. Dostupné z: https://en.wikipedia.org/wiki/Lisp_(programming_language)

[34] ARGÜELLES MENDEZ, Luis. A Practical Introduction to Fuzzy Logic using LISP [online]. Cham: Springer International Publishing, 2016 [cit. 2023-05-07]. Studies in Fuzziness and Soft Computing. ISBN 978-3-319-23185-3. Dostupné z: doi:10.1007/978-3-319-23186-0

[35] EFLL [online]. 2023 [cit. 2023-05-09]. Dostupné z: https://github.com/alve-soaj/eFLL

[36] Fuzzylite docs. [online]. 2017 [cit. 2023-05-09]. Dostupné z: https://fuzzy-lite.github.io/fuzzylite/?q=documentation/

[37] Jfuzzylite [online]. 2017 [cit. 2023-05-09]. Dostupné z: https://github.com/fuzzy-lite/jfuzzylite/

[38] Top 10 Best Embedded Programming Languages to Learn in 2023 [online]. 2023 [cit. 2023-05-09]. Dostupné z: analyticsinsight.net/top-10-programming-languages-to-develop-embedded-systems-in-2023/

[39] Javascript-fuzzylogic [online]. 2023 [cit. 2023-05-10]. Dostupné z: https://www.npmjs.com/package/javascript-fuzzylogic

[40] ESP32-DevKitC [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.espressif.com/en/products/devkits/esp32-devkitc

[41] Vývojová deska ESP32-DevKitC 38pin [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://dratek.cz/arduino/51547-esp32-devkitc-development-board-38pin.html

[42] ESP32-WROOM-32 Datasheet [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[43] Espressif: English Forum [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.esp32.com/viewforum.php?f=23

[44] Klon Arduino UNO R3 ATmega328P CH340 mini USB [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://dratek.cz/arduino/1353-klon-arduino-uno-r3-atmega328p-ch340-mini-usb.html

[45] Arduino UNO R3, Pin Diagram, Specification and Applications [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.elprocus.com/what-is-arduino-uno-r3-pin-diagram-specification-and-applications/

[46] Raspberry Pi Pico [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-pico/

[47] Arduino Integrated Development Environment (IDE) v1 [online]. [cit. 2023-05-11]. Dostupné z: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics

[48] Professional collaborative platform for embedded development [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://docs.platformio.org/en/latest/

[49] Atmel® Studio 7 [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://microchipdeveloper.com/atstudio:studio7intro

[50] Micropython [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://micropython.org/

[51] Arduino Forum [online]. 2023 [cit. 2023-05-11]. Dostupné z: https://forum.arduino.cc/

[52] What Is Fuzzy Logic?: Fuzzy Logic, Part 1 [online]. 2023 [cit. 2023-05-18]. Dostupné z: https://www.mathworks.com/videos/fuzzy-logic-part-1-what-is-fuzzy-logic-1629280959444.html

[53] LED dioda [online]. 2023 [cit. 2023-05-19]. Dostupné z: https://dratek.cz/arduino/1031-led-dioda-cervena-5mm.html?gclid=CjwKCAj-wvJyjBhApEiwAWz2nLe02f_rxp0-YHN_ftaMmr1j4bY6Lajk_4VUyEao25qb9BU4OjlePmRoCPWAQAvD_BwE

[54] Mitsubishi Electric: What makes our Heat Pump technology superior? [online]. 2023 [cit. 2023-05-21]. Dostupné z: https://www.mitsubishi-electric.co.nz/heat-pump/advantage.aspx

[55] Fuzzy logic control for lateral vehicle guidance. IEEE Control Systems [online]. 1994, 14(4), 55-63 [cit. 2023-05-21]. ISSN 1066-033X. Dostupné z: doi:10.1109/37.295971

[56] KONCZ, Annamária, Pokorád POKORÁD a Zsolt Csaba JOHANYÁK. FUZZY LOGIC IN AUTOMOTIVE ENGINEERING. Gradus. 2018, 5(2), 194-200. ISSN 2064-8014.

[57] TomTom: Fuzzy search [online]. 2023 [cit. 2023-05-21]. Dostupné z: https://developer.tomtom.com/search-api/documentation/search-service/fuzzy-search

[58] Fuzzy logika [online]. 2023 [cit. 2023-05-24]. Dostupné z: https://cs.wikipedia.org/wiki/Fuzzy_logika

[59] Rain Water Harvesting and Artificial Recharge - Scientific Figure on ResearchGate [online]. 2023 [cit. 2023-05-24]. Dostupné z: https://www.researchgate.net/figure/Boolean-Logic-Vs-Fuzzy-Logic_fig4_293061911

[60] Control of Electric Vehicles Charging Without Communication Infrastructure - Scientific Figure on ResearchGate. [online]. 2023 [cit. 2023-05-24]. Dostupné z: https://www.researchgate.net/figure/Main-components-of-the-fuzzy-controller_fig3_330081551

[61] Mikroprocesory, mikropočítače a ich základné bloky [online]. [cit. 2023-05-24]. Dostupné z: https://data.kemt.fei.tuke.sk/Mikroprocesorova-Technika/_web/wwwfiles/kapitola%2002.htm

[62] Fuzzy logika [online]. 2023 [cit. 2023-05-24]. Dostupné z: https://sk.wikipedia.org/wiki/Fuzzy_logika

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

FPGA Programovateľné hradlové pole.

ABS Protiblokovací brzdový systém.

ASR Protipreklzová regulácia.

FRBS Fuzzy Rule-Based Systems (systém založený na fuzzy pravidlách).

JDK Java development kit (vývojová sada nástrojov Java).

GUI Graphical user interface (používateľské rozhranie).

V Volt (jednotka elektrického napätia).

Ω Ohm (jednotka elektrického odporu).

GPIO General purpose input output (pin, ktorý je možné nastaviť na vstupúvýstup).

GND Ground (záporná polarita elektrického obvodu).

VCC Kladná polarita elektrického obvodu (napájacie napätie).

SEZNAM OBRÁZKŮ

Obrázok 1. Využitie fuzzy množín pri definovaní teploty 18
Obrázok 2. Porovnanie fuzzy logiky a binárnej logiky 19
Obrázok 3. Štruktúra fuzzy riadiaceho systému 22
Obrázok 4. Fuzzy kontrolér 24
Obrázok 5. Základné časti mikropočítača....27
Obrázok 6. Príklad fuzzy riadenia klimatizácie na základe teploty....29
Obrázok 7. Definovanie fuzzy pravidiel a fuzzy množín v knižnici fuzzylite 40
Obrázok 8. Pole na vzorové vstupné teploty a cyklus na výpis výstupných teplôt 40
Obrázok 9. Výstup kódu 40
Obrázok 10. Kód na implementáciu fuzzy logiky v knižnice jfuzzylite 42
Obrázok 11. Výstup kódu 42
Obrázok 12. Schéma zapojenia....52
Obrázok 13. Fyzické zapojenie teplomera....53
Obrázok 14. Nastavenie pinov LED diód, relé, rýchlosť komunikácie na sériovom porte a spustenie senzoru DHT22 55
Obrázok 15. Inicializácia vstupných fuzzy množín....56
Obrázok 16. Inicializácia výstupných fuzzy množín....57
Obrázok 17. Nastavenie fuzzy pravidla pre horúcu teplotu....58
Obrázok 18. Nastavenie fuzzy pravidla pre ideálnu teplotu....58
Obrázok 19. Nastavenie fuzzy pravidla pre studenú teplotu 58
Obrázok 20. Fuzzifikácia a defuzzifikácia teplotných dát....59
Obrázok 21. Rozhodovanie založené na fuzzy výstupe 60
Obrázok 22. Výstup fyzického zapojenia pri horúcej teplote....61
Obrázok 23. Výstup na sériovom monitore pri horúcej teplote....61
Obrázok 24. Výstup fyzického zapojenia pri ideálnej teplote 62
Obrázok 25. Výstup na sériovom monitore pri ideálnej teplote 62
Obrázok 26. Výstup fyzického zapojenia pri studenej teplote 62
Obrázok 27. Výstup na sériovom monitore pri studenej teplote 63
Obrázok 28. Graf teplôt teplomera s použitím binárnej logiky 64
Obrázok 29. Graf teplôt fuzzy teplomera 64

SEZNAM TABULEK

Tabuľka 1. Vlastnosti knižníc na implementáciu fuzzy logiky 46

SEZNAM PŘÍLOH

Príloha P1: Ukážkové kódy – dostupné na CD.

Príloha P2: Knižnice pre implementáciu fuzzy logiky – dostupné na CD.