



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

DEPARTMENT OF INTELLIGENT SYSTEMS

**MULTIAGENTNÍ SYSTÉM UČÍCÍ SE MAXIMALIZOVAT
KOMFORT UŽIVATELŮ V RÁMCI SMART HOME**

MULTIAGENT SYSTEM LEARNING TO MAXIMIZE USER COMFORT WITHIN SMART HOME

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

RADEK ČÁBELA

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.

BRNO 2023

Zadání bakalářské práce



141486

Ústav: Ústav inteligentních systémů (UITS)
Student: **Čábel Radek**
Program: Informační technologie
Specializace: Informační technologie
Název: **Multiagentní systém učící se maximalizovat komfort uživatelů v rámci Smart Home**
Kategorie: Umělá inteligence
Akademický rok: 2022/23

Zadání:

1. Prostudujte problematiku multiagentních systémů a obklopující inteligence.
2. Prostudujte problematiku automatizace budov a Smart Home a existující přístupy, aplikující multiagentní systémy v řízení.
3. Ve spolupráci s vedoucím zvolte vhodný subsystém Smart Home a specifikujte požadavky na jeho inteligentní řízení. Navrhněte řešení s využitím multiagentního paradigmatu a vhodného middleware tak, aby se průběžně na základě senzorických dat, interakcí uživatele se systémem a omezujících podmínek učil maximalizovat komfort uživatelů.
4. Navrženou řídicí aplikaci implementujte s využitím vybraného multiagentního middleware.
5. Ověřte funkčnost realizované řídicí aplikace v simulovaném prostředí s realistickými modely senzorů a aktuátorů a s vhodně abstrahovaným modelem budovy a chování uživatelů. Vyhodnoťte dosažené výsledky a diskutujte možnosti přechodu k reálnému nasazení.

Literatura:

Dle pokynů školitele.

Při obhajobě semestrální části projektu je požadováno:

První 3 body zadání.

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

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**

Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.

Datum zadání: 1.11.2022

Termín pro odevzdání: 31.7.2023

Datum schválení: 3.11.2022

Abstrakt

Tato práce přichází se způsobem, jak pracovat se zpětnou vazbou, inteligentními prvky domácnosti a tzv. agenty, aby ve výsledku došlo k minimalizaci přímých zásahů obyvatel domu do ovládání chodu domácnosti, a tudíž ke zvýšení jejich komfortu. Výsledná simulace demonstrující funkcionalitu návrhu, je zaměřena na problematiku regulace vytápění domácnosti.

Abstract

This thesis comes with a solution, how to work with feedback, Smart Home devices and "agents" in a way that minimizes direct Smart Home parameters changes coming from house inhabitants and therefore increases their comfort. Resulting simulation demonstrating the functionality of the system design is focused on problematics regarding changing temperature inside of a house.

Klíčová slova

chytrá domácnost, multiagentní systém, posilované učení, optimalizace komfortu, Home Assistant, Python, Spade

Keywords

smart home, multi-agent system, reinforcement learning, comfort optimization, Home Assistant, Python, Spade

Citace

ČÁBELA, Radek. *Multiagentní systém učící se maximalizovat komfort uživatelů v rámci Smart Home*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Multiagentní systém učící se maximalizovat komfort uživatelů v rámci Smart Home

Prohlášení

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

.....
Radek Čábelá
31. července 2023

Poděkování

Rád bych poděkoval vedoucímu práce Doc. Ing. Vladimíru Janouškovi Ph.D., za rady a nastínění problematiky kolem Smart Home. „Brainstorming“ na několikahodinových konzultacích byl zajímavou zkušeností. Dále bych rád poděkoval své rodině za podporu a trpělivost

Obsah

1	Úvod	4
2	Důležité pojmy	5
3	Problematika a požadavky	6
3.1	Automatizace budov a obyvatelé	6
3.1.1	Predikce chování uživatelů	6
3.2	Multiagentní systém a alternativy	7
3.3	Učení	8
3.3.1	Posilované učení	8
3.3.2	Problematika stavového prostoru	9
3.4	Další problematika	9
3.4.1	Generování dat senzorů	9
4	Návrh	10
4.1	Komunikační tok	11
4.1.1	Úroveň zařízení a vnitřní logiky	11
4.1.2	Úroveň Smart Home	12
4.2	Logické celky systému	12
4.2.1	Senzory	12
4.2.2	Artefakty	13
4.2.3	Logika agentů	14
4.2.4	Reflexní agenti	14
4.2.5	Agent učení	15
5	Implementace	17
5.1	Použité technologie a knihovny	17
5.1.1	Spade	17
5.1.2	MQTT	17
5.1.3	Asyncio Paho	18
5.1.4	Další využití knihovny a nástroje	19
5.2	Struktura souborů	19
5.3	Inicializace	19
5.4	Interakce uživatele	19
5.4.1	Časový plán	19
5.4.2	Změna vlastnosti prostředí	20
5.5	Komunikace agentů	20
5.5.1	Asynchronní zpracování zpráv	20

5.6	Učení a volba další akce	21
5.6.1	Modelované objekty	23
5.7	Home Assistant	24
5.7.1	Alternativa	25
6	Závěr	27
6.1	Další výzkum	27
6.1.1	Konkrétní Obyvatelé	27
6.1.2	Porovnání existujících přístupů	28
6.1.3	Splnění požadavků	28
	Literatura	29
A	Manuál	32
A.1	Základní instalace	32
A.2	Pokročilá instalace	32

Seznam obrázků

4.1	Komunikace v návrhu simulačního modelu	10
4.2	Příklad komunikace v sekvencích; uvnitř je 19 °C a přítomna minimálně jedna osoba, uživatel zasílá požadavek o zvýšení teploty, venku je 15 °C . .	11
4.3	Abstraktní plán místností	12
4.4	Komunikace vrstev a rozšířitelnost	12
4.5	Komunikace artefaktu	14
4.6	Motivační vizualizace učení a přeučení v navrhovaném systému.	15
5.1	Výpis při publikování informací na MQTT téma při nastaveném <i>Retain flagu</i>	18
5.2	Implementace stavů agenta pro správu teploty; napodobitelný pro sběr dat z jiných senzorů. Šedě označená část není implementována v simulaci, ale je doporučena pro rozšíření reálného nasazení.	20
5.3	Implementace stavů agenta pro správu uživatelských požadavků; šedě označená část není implementována v simulaci, ale je doporučena pro rozšíření reálného nasazení.	21
5.4	Vzorec vyhodnocení Q-hodnot v daných stavech po provedení daných akcí. .	22
5.5	Vizualizace výsledku učení. Interakcí modelovaného uživatele se systémem požadavků. Dle výsledné analýzy byl v tomto scénáři uživatel s teplotou spokojený 90.62% času. Klíč: červená barva: interval kdy modelovaný uživatel chce zvýšit teplotu z prvotních 20 °C v době reálného požadavku; Fialová: interval, kdy chtěl uživatel snížit teplotu z 20 °C – ukázalo se, že dle reakcí chtěl v daném intervalu teplotu kolem 21 °C; Žlutá: doba kdy simulovaný uživatel není v domě.	23
5.6	Ukázka uživatelského rozhraní vytvořeného v Home Assistant. Ikony teplo- měru a šipky publikují požadavek uživatele přes protokol MQTT	24
5.7	Ukázka názvů automatizací vytvořených v Home Assistant.	25
5.8	Ukázka Home Assistant „Templatingu“. Publikována je zde informace o ča- sovém plánu.	26
A.1	Reset HASS konfigurace.	33

Kapitola 1

Úvod

Když se řekne „komfort“ v kontextu bydlení v domácnosti, každému se vybaví něco trochu jiného. Někdo preferuje, když je vše zorganizované a uklizené, někdo za komfort považuje, když má v domě možnost k relaxaci a odpočinku. Už při svém vzniku v cca 13. století bylo slovo komfort (z fr. conforter) používáno v kontextu pocítění úlevy či posílení nebo podpory. Tato práce se slovem komfort pracuje také ve významu úlevy. Pokud se člověk vrátí domů po dlouhém dni např. v práci, doma je příjemná teplota, vzduch i osvětlení, osoba pocítí komfort, nebo alespoň úlevu, díky úbytku povinností.

Takového zlepšení kvality života se práce snaží docílit navržením transparentního a zároveň dostupného řídicího systému. Systém je navržen jako základní blok, na kterém lze stavět flexibilní odpověď na existující komerční řídicí systémy zmíněné v kapitole 6. Využití v něm jsou tzv. „agenti“. Jde o koncept z odvětví umělé inteligence, kde agent je prvek vnímající okolní prostředí, a který může toto prostředí měnit. Navržený systém disponuje více agenty změřenými na specifické části systému. Kupříkladu agent pro sledování stavu teploty v domě může teplotu ovládat na bázi komunikovaných pokynů od agenta obstarávajícího učení.

Konceptem „agenta pro učení“ se v této práci rozumí agent, který má možnost komunikovat s ostatními jako každý jiný, který obsahuje navíc ve své logice i algoritmus posilovaného učení. Ten umožňuje naučit se jak model měnících se sledovaných vlastností v domě (např. fyzikálních veličin, jako teplota), tak sledovat a učit se cíle daného uživatele.

Navazující část textu je organizována následovně: nejprve, v kapitole 2 je čtenář obeznámen s používanými pojmy a jejich obecnou definicí, případně definicí v kontextu práce. Obsahem kapitoly 3 je seznámení se studovanou teorií a danou problematikou, kterou se v práci povedlo vyřešit, a dalšími tématy, které práci brzdily, nebo byly navrženy na výzkum pro budoucí rozšíření. Obsahem kapitoly 4 je obeznámení se s návrhem systému a kapitola 5 nabízí pohled na zrealizovanou implementaci. Závěrem - kapitola 6, je nabídnut souhrn možností, které byly nalezeny v průběhu řešení práce a zhodnoceny jako zajímavé pro rozšíření systému. Součástí kapitoly je také ohlédnutí se za výsledkem práce.

Kapitola 2

Důležité pojmy

- Smart Home – Smart Home zařízení – technologie v podobě senzorů, měřičů, rozhraní, spotřebičů, a dalších zařízení propojených v síti, umožňující automatizaci a vzdálené ovládání domácího prostředí [12],
- IoT – ekosystém počítačů a chytrých zařízení či strojů, které jsou schopny vzájemně komunikovat nebo spolupracovat bez asistence člověka; z rascasone.com,
- senzor – z aj. *sense* – vnímat, zařízení sledující vjemy okolního světa,
- aktuátor – akční člen – část zařízení působící akci ovlivňující nebo viditelnou pro okolní prostředí
- multiagentní systém – systém, který v prostředí obsahuje více agentů, kteří mezi sebou mohou spolupracovat,
- znalostní báze – znalosti (v kontextu práce) agenta, v podobě stavů uchovaných v paměti, podle kterých se dále rozhoduje,
- simulační model – program pro experimentování s napodobeným modelem reálného systému,
- serializace dat – přizpůsobení dat objektu ke snazšímu přenosu; z hazelcast.com/glossary
- implementace – uvedení myšlenek a návrhu do spustitelné podoby kódu,
- middleware – software působící jako mezičlen (přemostění) zprostředkující danou funkcionalitu.

Kapitola 3

Problematika a požadavky

V této kapitole je uvedena problematika jednotlivých tématických celků, do kterých práce zasahuje. Dají se přibližně vyjmenovat, jako „chytrá domácnost a analýza chování obyvatel“, „posilované učení“, „multiagentní systémy“ a rozšíření návrhu práce zamýšlí využití odlišného přístupu v řešení práce mimo jiné například z „problematiky her“ nastíněné v kapitole 6.

3.1 Automatizace budov a obyvatelé

Prioritou pro tuto práci, bylo přijít s široce uplatnitelným návrhem, možném uvést do reality pro co největší množství typů budov, přijatelným co největším množstvím lidí. Jelikož každá domácnost se liší budova od budovy a lidé v ní žijící osoba od osoby, zakomponovat chování lidí do řízení budovy je stále přetrvávající a aktuální problém. Nasazené komerční systémy poloviny minulého desetiletí, se často řídily předdefinovaným chováním a nastavením, pracujícími s mnoha neurčitostmi. Na oblast předpovídání chování uživatelů v budově se kriticky soustředil globální výzkum posledních let. Jako možný způsob, jak dojít k optimálním řešením, se ukázalo využití multiagentních systémů. V případě, že jsou navrženy splňující kritéria pro inteligenci, adaptaci, flexibility pro komfort v budově a správu spotřeby energie, zmírní limitace budov při otázce spotřeby energie stejně tak, jako zvýší spokojenost obyvatel budovy. [27]

3.1.1 Predikce chování uživatelů

Důležitým úkolem systému pracujícím s komfortem a podporou pod-subjektů, v tomto případě osob žijících v objektu, je porozumět jim. V této práci, jako v každém navrhovaném systému, bylo třeba definovat kategorie a proměnné, použitelné pro výsledný uzavřený simulační model. Přímá predikce závislá na úrovních složitostí, byla jedna z možných schopností, které by mohl výsledný systém demonstrovat. I přes nabízenou jednoduchost výpočtů, problematika ze studovaného článku [24], ale ukázala, že data často vychází z různých úrovní složitostí a množství vstupů konkrétních IoT prvků. Zmíněná nejvíce prováděná akce uživatele v čase (např. rozsvícení světla v 18:00), nemusí znamenat, že při vlivu dalších okolností (např. změna času setmění, příchod nových obyvatel do místnosti) nutně systém musí tuto akci automaticky vykonat. Tato práce v důsledku zmíněné výpočty neimplementovala, nicméně byla mnohými fakty ovlivněna i např. tvrzením, že mnoho současných algoritmů pro předpověď vzorů chování uživatelů vyžaduje velké množství záznamů a před samotným provedením se musí provést sběr dat značných rozměrů.

3.2 Multiagentní systém a alternativy

Pro docílení tvorby inteligentního systému, který ve výsledku zlepší kvalitu a komfort života v domácnosti a případně další parametry, jako např. energetickou šetrnost, je možné zvolit více přístupů. Za zmínku stojí přístup vývojem s využitím inteligentních výpočetních metod, jako např. kombinace neuronových sítí, fuzzy logiky a evolučních algoritmů. Dle závěru článku [27], podpořeného posudkem (pouze k optimalizaci genetickými algoritmy) [15], jsou tyto metody při využití v automatizaci řízení budov sice spojované s revolučním způsobem úspory energie a lepšími dopady na komfort, nicméně zůstávají spíše nevyužívané z důvodu požadavků kvalitní znalostní báze, trénovacího vzorku dat a neschopnosti poradit si s novými, nebo neznámými vlivy v systému.

Zmíněné přístupy byly prozkoumány a některé z nich vycházející modifikace představeny jako možnost rozšíření práce 6. Hlavním cílem, vycházejícím ze samotného zadání práce, bylo do navrhovaného systému zakomponovat koncept více agentů. Toho docílit, představuje mnohá úskalí. Do výsledného návrhu v kapitole 4, byl původně v této práci zamýšlen návrh agentů specifických pro každého obyvatele domu. Takoví agenti, kteří postupem času poznají zájmy sledovaného uživatele, by měli možnost spolu komunikovat a smlouvat ideální podmínky, které by v nejlepším případě navíc přizpůsobili vlivům prostředí. Do cesty takovému řešení, se ale postavila implementační náročnost kooperace agentů sahající do problematiky metod hraní her, volba způsobu učení multiagentního systému, který může vyústit značnou komplexitou [18], samotný algoritmus učení a jeho případná úprava představující další výzvy [28], (možné řešení je zmíněno v další sekci 3.3) a potřeba využití tzv. BDI agentů. Využití tohoto paradigmatu není v problematice Smart Home nejlepší volbou, jelikož ne vždy jde znát rozhodovací procesy obyvatel domu a přesně je modelovat [26]. V takových případech, je doporučen algoritmický přístup.

Práce se potřebovala vypořádat s otázkami spojenými s dosud jmenovanou problematikou. Osvědčeným způsobem, jak toho dosáhnout, bylo definovat požadavky výsledného systému:

- generalizace systému na různé typy budov s n místnostmi,
- přizpůsobení se (nekonkrétním) uživatelům,
- schopnost práce s vnějšími vlivy (např. venkovní teplota),
- možnost definice dodatečných omezení (např. spotřeba energie),
- využití vícero algoritmických agentů pro práci s konkrétními parametry, řídící se konkrétními pravidly a jasnou definicí těchto parametrů a pravidel,
- rozšiřitelnost využití různých senzorů a IoT prvků,
- škálovatelnost návrhu a implementace systému, pro možnost např. rozšíření o další inteligentní metody,
- systém produkující zlepšení v oblasti komfortu uživatelů s možností vizualizovat pozitivní změny v čase.

Závěr (sekce 6.1.3) se ohlíží právě za splněním těchto a dále definovaných požadavků.

3.3 Učení

Hlavní myšlenka docílení komfortu v domácnosti má za cíl automaticky nastavovat parametry IoT prvků, jako např. spotřebičů, před okamžikem, kdy by tak učinil uživatel. Dva vzorové scénáře pracují s tímto cílem s rozdílem v přítomnosti, či nepřítomnosti daného uživatele. Pro takovou funkcionalitu bylo zhodnoceno jako nezbytné implementovat algoritmus učení.

Již zmíněná problematika implikuje, že uvést do chodu učení jako abstrakci specificky nad každým uživatelem, by vedlo k nutnosti implementovat multiagentní učení. Konkrétním algoritmem na míru pro danou problematiku, se jeví [10], který má dokázanou konvergenci k optimálnímu řešení pro jednotlivé agenty. Od tohoto přístupu se ale ustoupilo, jelikož by svou složitostí i rozsahově zastínil zbylou část práce. Dále je zmíněn mezi možnostmi dalšího rozšíření a výzkumu v kapitole 6.

Při hledání dalších možností implementace multiagentního prostředí s učním, byla objevena existence mnoha platforem pro různé multiagentní úkony (např. [2]), ačkoli většina se zdála vhodná hlavně pro řešení navigačních problémů, nebo řešení jednoho cíle. Shrnutí prací a detailní problematiku nabízí [22], navíc s návrhem prostředí virtuálního domova a humanoidních agentů. Vyhovujícím přístupem se ukázalo učení centralizovat pro jednoho „agenta učení“, který výsledný plán odešle dalším agentům v hierarchii. S přítomností obyvatel se v základním konceptu začalo počítat jako s dvěma stavy: *nikdo není přítomen a alespoň jedna osoba přítomna*. Zkoumání chování konkrétních skupin obyvatel je další komplexní odvětví, vyžadující mnoho vstupních dat a změn v učení, důsledkem je to jedno z dalších témat, kterému výsledek práce nechal volný průchod do rozšíření systému 6.

Požadavkem pro navržené učení se stala možnost pracovat zde stále s novými požadavky uživatele a zpracovat je tak, aby se v dlouhodobém měřítku vytvořil plán nových časových intervalů a stavů spotřebičů v nich. Dalším požadavkem, který bylo nutno neopomenout, byla možnost přizpůsobit se prostředí a vlastnostem dané budovy nebo místnosti (např. tepelná izolace, možnost větrání, množství světla. . .) a během učení akumulovat do znalostní báze informace o tomto prostředí. Oba tyto požadavky se dostaly do finálního návrhu. V sekci 5 je popsána inicializace, kdy systém zasílá testovací požadavky a buduje teplotní model místnosti.

3.3.1 Posilované učení

Vhodně se začalo jevit použití posilovaného učení [8]. Hlavní vlastností při využití metod posilovaného učení, je ohodnocení akcí na základě penalizací či odměn. První zvažovaný přístup v odměňování či penalizaci, byl návrh funkce odměňování na základě historických požadavků uživatele, odměny by se vypočítali na základě doby od posledního požadavku. Výstup této funkce se dal snadno převést na tabulkové ohodnocování stavů. Po testování se nicméně ukázalo, že výstup učení není využitelný při měnícím se prostředí a pro implementaci v reálném čase.

Přístup se následně modifikoval na penalizaci pouze takových stavů, kdy systém obdržel libovolný požadavek obyvatele. Takové požadavky se následně akumulují ve znalostní bázi a při mnoha iteracích nového učení a procházení časových intervalů (viz kapitola 5) je učení schopno pracovat s pravděpodobnými reakcemi uživatele a zpracovat je. Podoba provedené akce již vychází z konkrétních stavů systému, ten se pro konkrétní návrh učení omezil pouze nastavení teploty. Výslednou akcí pro daný stav je zde zvýšení, snížení nebo ponechání teploty. Obecně výsledek studie dospěl k nutnosti provádět co nejjednodušší akci a vlivy

ostatních prvků domácnosti vložit do stavového prostoru – tudíž modularizovat vrstvu učení na podsystémy pro každý typ právě zkoumaného aktuátoru.

3.3.2 Problematika stavového prostoru

Stavový prostor finálního učení má tendenci se měnit podle zrovna používaných prvků IoT. Pokud například používáme zařízení dvou-stavového charakteru (např. Světlo – svít, nesvítí) do stavového prostoru je přidán jeden další stav. V případě, že ale máme složitější zařízení např. závislá na prostředí kolem, pro funkčnost posilovaného učení se stavový prostor rozšíří minimálně o jeden další stav (např. Termostat – topit/netopit; zvýšení/pokles teploty). Pokud se řeší čistě kategorie IoT prvků bez vlivu na prostředí, dají se využít kategorie pro jeden nebo více vstupů jmenované v [24] a na vrstvu agenta pro požadavky uživatelů podle článku implementovat i jednoduchou predikci nastavení pro daný čas.

3.4 Další problematika

Práce postupem času odhalila, že některá odvětví, která zdánlivě zapadají do problematiky Smart Home, nepatří do kontextu základní myšlenky této práce, kterou je optimalizace komfortu.

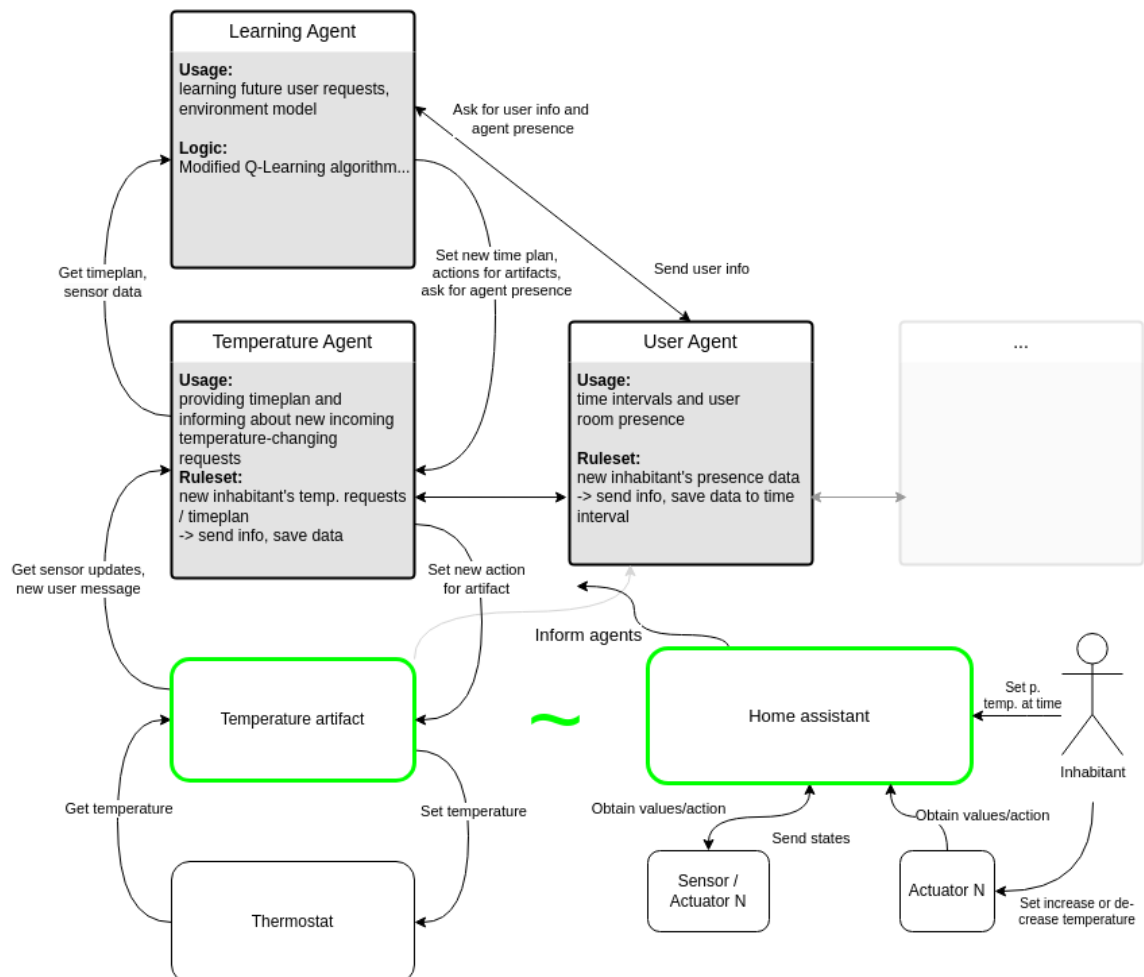
3.4.1 Generování dat senzorů

System Smart Home pracuje v reálných nasazeních s množstvím různorodých senzorů. Cílem této práce není do výsledné případové studie zakomponovat všechny reálné typy senzorů a jejich způsob komunikace. Po studii generování dat [14], práce v simulačním modelu zjednodušila generování dat senzoru, aby zde zůstalo to nejpodstatnější, jako podobný způsob předávání dat a vlastnost, například teplota – desetinné číslo v odpovídajícím intervalu, a aby předávané hodnoty odpovídaly nastavení dalších abstraktních prvků a nebylo zcela náhodně generováno.

Kapitola 4

Návrh

Analýza problematiky spojené s odvětvími, do kterých práce zasahuje, dospěla k závěrečnému návrhu řešení. Je rozdělena na několik částí popsaných a znázorněných v následujících sekcích.



Obrázek 4.1: Komunikace v návrhu simulačního modelu

4.1 Komunikační tok

Nejdůležitější část návrhu je způsob komunikace v systému. Práce si dala za úkol jasně definovat, který prvek bude dostávat informace od koho, jak často, a v jakém formátu tyto informace budou.

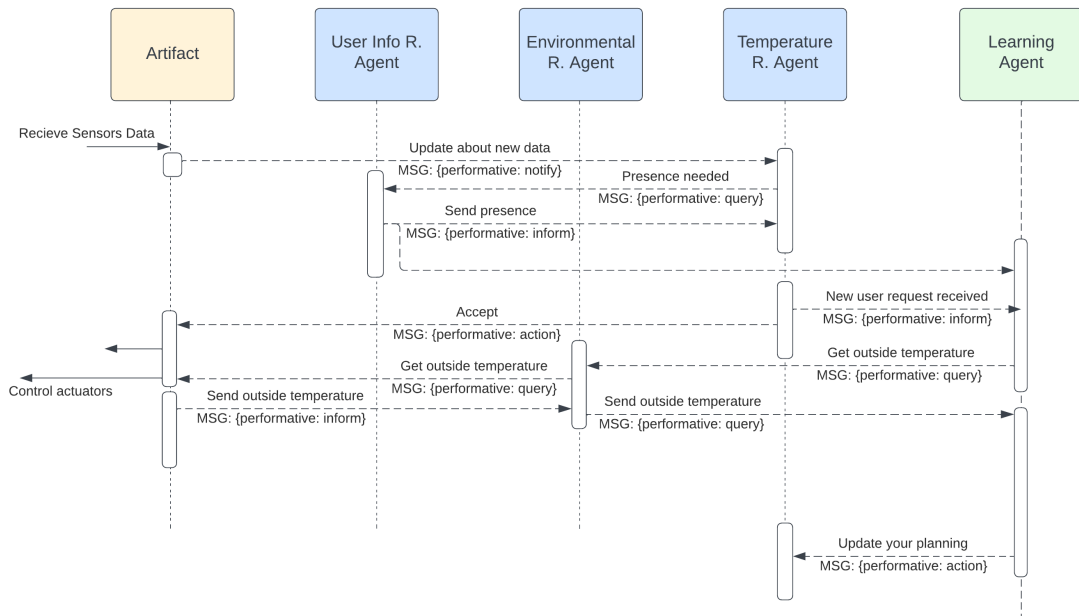
4.1.1 Úroveň zařízení a vnitřní logiky

Když začneme na vrstvě jednotlivých senzorů a aktuátorů (IoT zařízení), dále jen „IoT prvky“ komunikují s vrstvou tzv. artefaktů informace o sledované skupině parametrů. Poskytují jim informaci o novém požadavku, nebo nové, např. naměřené, hodnoty. Artefakt je abstrakcí převzatou ze studie o využití tzv. A-A (Agent-Artifact) meta-modelu [19], dále popsaného v článku z volně dostupného žurnálu Electronics [20] a doplněného o reálnou implementaci v oblasti zavlažování půdy. Každý artefakt slouží jako pasivní prostředník komunikace mezi agenty a IoT prvky, který poskytuje agentům potřebné zdroje.

Agenti, na druhé straně, jsou pro-aktivní prvek systému. Mezi sebou sdílejí informace o přítomnosti a informace potřebné pro soubor vlastních pravidel. Každý agent má „pod sebou“ zpravidla jeden artefakt, kterému zasílá pokyny k provedení akcí. Toto je kompletní funkcionalita agentů, kteří se drží konceptu „reflexních agentů“. Ti dále kromě artefaktů komunikují s agentem učení, kde už jde o agenta zpracovávajícího model okolního světa.

Tento agent získává od reflexních agentů (nutně ne všech) nové informace poskytované v reálném čase. Nekomunikuje přímo s artefakty, ani senzory. Informace zpracuje a před dalším učením požádá ostatní agenty a nové stavy a data, o které se v daný moment zajímá.

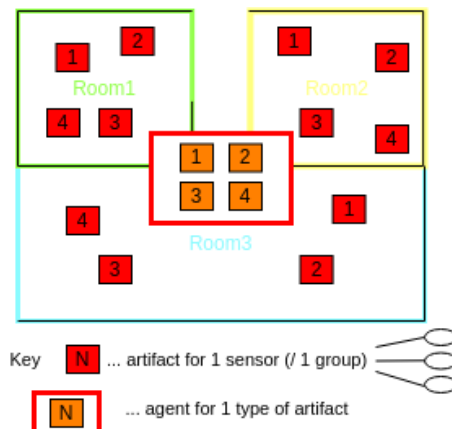
Další informace o konkrétně využitých technologiích v implementaci simulačního modelu a pro reálné využití jsou v sekci 5.



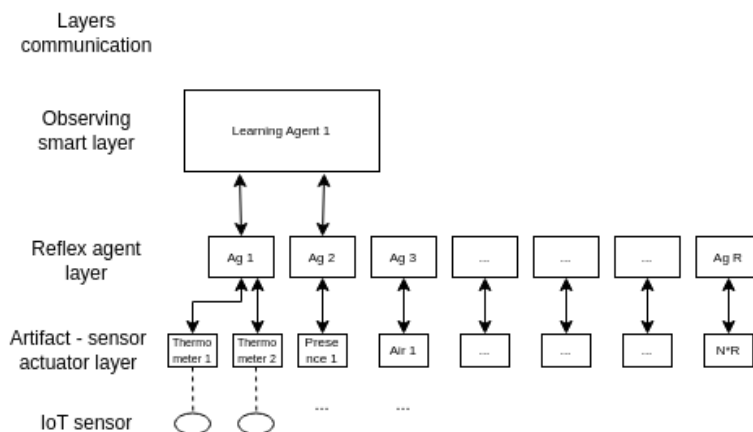
Obrázek 4.2: Příklad komunikace v sekvencích; uvnitř je 19 °C a přítomna minimálně jedna osoba, uživatel zasílá požadavek o zvýšení teploty, venku je 15 °C

4.1.2 Úroveň Smart Home

Na úrovni Smart Home je komunikace navržena v rámci místnosti a škálovatelná dále na celou budovu. V budově existuje alespoň jeden prvek s procesním výkonem na míru složitosti budovy a požadované obklopující intelligence. Tento prvek slouží jako klíčový bod shromažďující artefakty s různých místností. Tomu posílají data po domě rozmístěné senzory. Dále s tímto bodem komunikují, na stejném či jiném zařízení nasazení agenti. Jeden agent v návrhu pracuje s informacemi z jedné nebo více místností. Obdobně jsou implementováni učící se agenti, vyžadující největší výpočetní výkon ze všech jmenovaných elementů systému Smart Home.



Obrázek 4.3: Abstraktní plán místností



Obrázek 4.4: Komunikace vrstev a rozšiřitelnost

4.2 Logické celky systému

4.2.1 Senzory

Vysoce přizpůsobitelná vrstva systému, je vrstva senzorů a zařízení z rodiny IoT. Díky této vrstvě lze zaznamenat různé akce obyvatel domu, stavy místností, ale i informace o venkovním prostředí, relevantní pro logiku systému. Z komunikačních standardů byl zvolen

protokol MQTT (*MQ Telemetry Transport*, dále bude používáno jen MQTT protokol). Tento komunikační protokol umožňuje senzorům zveřejňovat svůj daný stav a ponechat tuto informaci při poslechu daného „téma“ (z aj. *topic*, dále jen téma). Naopak agentům a subsystémům následné logiky umožní odebírat informace z daných témat, které jsou pro daný prvek relevantní. Pro získávání dat ze senzorů bylo navrženo následující schéma témat, nicméně dá se navrhnout odlišná struktura více vyhovující dané problematice, budově, nebo obytného systému.

`<objekt> / <místnost> / [lokace] / <vlastnost> / [příkaz]`

Například:

```
dum / obyvaci_pokoj / teplota
dum / chodba / obyvatel_1 / zadost_zvysit_teplotu
```

V systému využívajícím takový způsob komunikace, narážíme na první limitaci a to, použít senzory podporující tento komunikační protokol. Naopak množství senzorů limitací není, díky podpoře zástupného znaku (*wildcard*) v protokolu MQTT, jež umožňují artefaktům naslouchat více senzorům najednou. Limitací ale je potřeba využití centrálního MQTT *brokeru*, (centrálního prvku pro implementaci protokolu), přes který probíhá přihlašování logických celků k poslechu daných témat a publikování dat na jedno dané téma (publikování na více témat zároveň, není v protokolu povoleno). Implementace simulačního modelu 5 pracuje s abstraktními senzory. Ty komunikují v podobě publikování informací na daná témata ve formátu, které využívají některé reálné senzory.

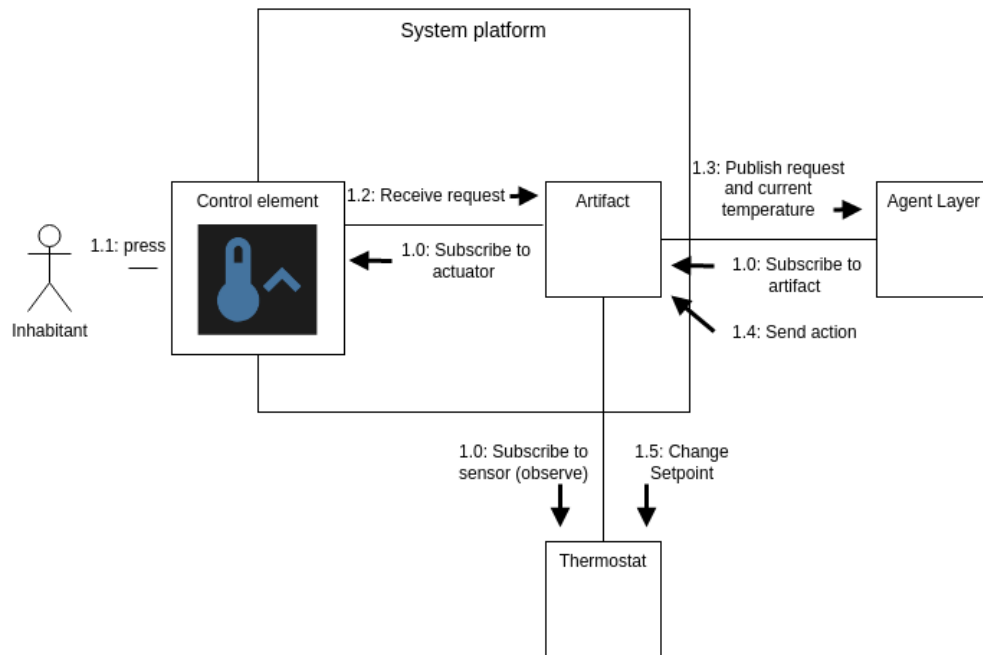
4.2.2 Artefakty

V této práci byl převzat již zmíněný koncept tzv. artefaktů ve 4.1.1, který umožňuje další ovládání aktuátorů a sledování stavů senzorů, případně zprostředkování těchto informací dále. Koncept IoT artefaktů byl převzat, jakožto komunikace s vrstvou artefaktů, která podporuje pozorovatelné parametry, jako například dostupnost jednotlivých prvků, které artefakt zaštiťuje. Obohacen byl o možnost uchování všech artefaktů na centralizovaném místě. Dále o možnost využít vhodnou platformu, která zpřehlední výskyt aktuátorů v systému a umožní je ovládat. V konkrétní implementaci jde o systém „Home Assistant“ dále zmíněný v kapitole 5.

Na obrázku 4.3 je znázorněna možnost využití skupin IoT zařízení sledovaných jedním artefaktem. Při konkrétní implementaci, není vývojář odkázaný na jediný postup, který toto umožňuje dodržet. K dispozici má na výběr stále nespočet komunikačních protokolů daných senzorů a různých platformy. Stačí, aby systém splňoval následující kritéria:

- Informace je vždy předána, i když v daném okamžiku příjemce není dostupný,
- v systému spravujícím artefakty je možnost uchování dat,
- umožnit příjem informací jednoduchým způsobem, pro zařízení s malou výpočetní silou (IoT zařízení).

Po propojení tyto prvky nepředstavují komplikovaný subsystém. Naopak je správně implementovaný systém jednoduchý na ovládání a měl by být snadno přijat uživateli. Komunikační diagram 4.5 reprezentuje příklad funkcionality artefaktu.



Obrázek 4.5: Komunikace artefaktu

4.2.3 Logika agentů

Aby navržený agenti přispívali k zadostiučnění obyvatel, je třeba, aby měli možnost rychle reagovat na požadavky obyvatel. 5 agentů si jde představit jako 5 paralelních procesů připravených obratem reagovat na novou změnu v systému. Logika agentů by měla být napsána v programovacím jazyce podporujícím paralelismus. To už v dnešní době patří ke zcela běžným vlastnostem jazyků. Dále je vhodné, aby byl zvolený programovací jazyk objektově orientovaný, například pro stavbu modelu okolního světa a modelu o uživateli a jeho snadné uložení ve znalostní bázi agenta. Jde sice o nepovinnou součást návrhu, nicméně proces tvorby znalostní báze agentů může zpřehlednit využití agentních programovacích jazyků, případně ucelení (dále jen *framework*) vybraného jazyka, podporující alespoň komunikační paradigmat agentů.

4.2.4 Reflexní agenti

Stav daných skupin artefaktů sledují reflexní agenti. Dle pravidel daného agenta a přijatého požadavku, je vyvozena akce, kterou by měl artefakt provést nad danými IoT aktuátory. Pokyn k akci agent odešle artefaktu. Je třeba, aby podoba (formát) akce byl v systému jasně definován. Artefakt může přijímat pouze hodnotu pro daný aktuátor, nebo definovat strukturu přijímaných dat. Pro definici struktury návrh v této práci doporučuje využití již existujících standardů pro výměnu a serializaci dat, jako například JSON nebo YAML.

Dále by agenti měli obsahovat obecně definované vlastnosti reflexních agentů popsané například zde [7]. Tito agenti by si neměli pamatovat své minulé akce a řídit se jimi. Implementovaná pravidla, by měla být jasně definovaná. V této práci musí být reflexní agenti schopní přijímat celý systém pravidel. V simulaci a implementaci je tato myšlenka dále rozvedena. Reflexní agent, například sledující teplotu v místnosti by měl implementovat zpracování přijatých atributů, pravidel, co, komu, kdy poslat za zprávu a metody, které bu-

dou pracovat s dále definovanou datovou strukturou dále určující, jakou akci, kdy provést. Tento princip otevře dveře dalšímu důležitému typu agenta.

4.2.5 Agent učení

Na nejvyšší úrovni, v kontextu priority a směru komunikace, se nachází učící se agent. Vybaven logikou pro přijímání zpracovaných požadavků obyvatel a jejich časových razítek, má možnost naučit se způsobu, jak požadavkům vyjít vstříc v dalších časových intervalech. Kvůli požadavkům na funkci učícího se agenta je třeba specifikovat diskrétní časové intervaly pro příjem požadavků od obyvatel. Čím menší časové intervaly zvolíme, tím více narostou požadavky na hardware, konkrétně na operační paměť. Směr, kterým vede případné snížení intervalů na minimum, tj. na spojitý čas je popsán v sekci 3.3.

Navržený mechanismus, poskytne jako svůj výstup agentům pro správu daných artefaktů pevně daný návod, jak hodnoty artefaktů v daných časech automaticky upravit. Učící se agent, vždy obdrží nově zpracovaný požadavek uživatele, aby posoudil zda-li je v souladu s naučeným návodem, či naopak. Ve většině případů není. Tehdy s existujícím modelem provede učící agent přeučení a reflexním agentům předá nové instrukce pomocí dané datové struktury. Popsanou funkcionalitu popisuje pseudokód:

Algoritmus 1 Pseudokód přeučení agenta

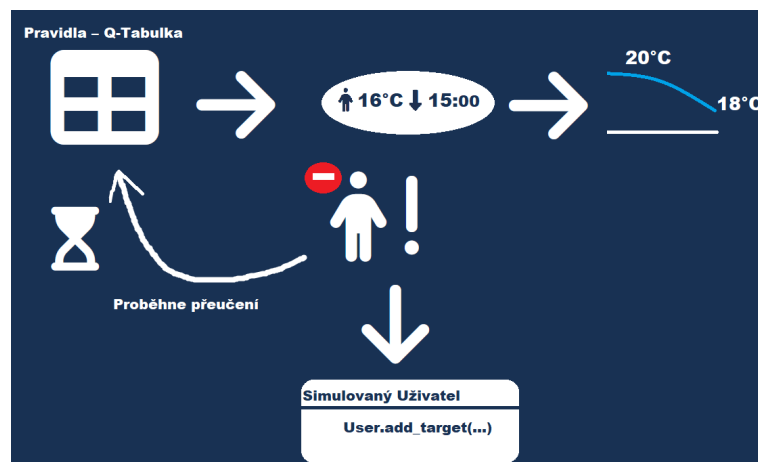
```

if Message received & Message Type is User Request then
    time ← Message Timestamp
    newmodel ← LEARN(Current Model, Message Request type)
    ruleset ← CREATE RULESET(newmodel)
    SEND RULESET(ruleset)
end if

```

Agent je navržen tak, aby jednou za uplynutí několika časových intervalů provedl nahodile akci v systému a přeučil se model vlastností okolního světa. Příkladem je opět změna teploty. Pokud klesá rychleji po vypnutí termostatu, nejspíše je otevřené okno, nebo nízká venkovní teplota a zasílaná pravidla ostatním agentům by se tomu měla uzpůsobit.

Pro shrnutí a připomenutí: V kontextu návrhu modelu učícího agenta mluvíme o modelovaném uživateli a jeho cílech nebo o modelovaných vlastnostech prostředí. V návrhu



Obrázek 4.6: Motivační vizualizace učení a přeučení v navrhovaném systému.

budovy lze prostředí definovat jako společné hodnoty více artefaktů umístěných v jiných místnostech, jinými slovy na jiných serverech.

Kapitola 5

Implementace

Jakožto ověření konceptu byl vytvořen simulační model. Text je dále členěn dle jednotlivých fází simulace. S těmito fázemi odpovídajícím způsobem souvisí i realita. Případy, kdy se použití implementovaného programu liší od podoby při reálném použití, jsou v této kapitole vždy explicitně zmíněny.

5.1 Použité technologie a knihovny

5.1.1 Spade

Pro implementaci agentů byl zvolen Spade. Jde o prostředí, uvedené v knihovně IEEE Explore zde [21], (z jiných definic „middleware“), pro vývoj multiagentního systému postaveném na programovacím jazyce Python. Spade je postaven na protokolu okamžitého zasílání zpráv XMPP, podporující i jeho rozšíření o notifikace v XEP-0184. K funkci agentů je třeba je registrovat na centrální XMPP server v domácnosti, nebo i externí, mimo síť Smart Home. Na oficiálním webu jsou uvedeny možnosti.

V implementaci této práce byl každý agent registrován na server. Dále pomocí předprogramovaných šablon každý agent, „znal“ druhého agenta. Takto jsou ignorovány jakékoli jiné zásahy do bezpečnosti XMPP komunikace agentů. Se vzorem ve standardu FIPA, používají agenti dané *performative* typy zpráv, agenti zasílají buď dotazy očekávající okamžitou odpověď *query*, nebo notifikace *notify*, nebo informují např. o nových stavech artefaktů pro případ, že jim druhý agent naslouchá, pomocí *inform*.

Stejně jako u dále zmíněného konceptu MQTT i Spade podpruje paradigma *Publish-Subscribe*, tj. „publikuji a odebírám“, nebo také „odesílám a naslouchám“. Tato práce si vystačila s klasickým odesláním a přijímáním zpráv definovaným v XMPP protokolu.

5.1.2 MQTT

MQ Telemetry Transport protokol, jehož využití v práci je detailně popsáno v sekci 4.2.1, je v práci široce využíváný a zmíněn na více místech v tomto textu. Specifikace je v tomto dokumentu [3]. Byl instalován broker verze 3.1.1, nezbytný pro chod systému, dále popsáný v manuálu instalace.

V protokolu se často využívá tzv. „flagů“ (značek). Nejpodstatnějšími jsou *QoS* (kvalita služby) a *Retain* (uchování) flagy. Pokud chceme, aby byl agent ujistěn, že pokyn o změně teploty došel do cíle, použijeme *QoS* hodnoty 1. Pokud chceme, aby poslední zpráva publikovaná na dané téma na něm chvíli „zůstala“, použijeme *Retain flag* hodnoty 1, v opačném

případě ponecháme 0. Home Assistant 5.7 nabízí [integraci](#) [11] pro protokol MQTT, po jehož použití se objeví nové entity nebo automatizace, které umožní využití protokolu. Užitečné příklady byly uvedeny na následujícím [webu](#) [23], ze kterého pochází i vizualizace možného výpisu při registraci MQTT zařízení na obrázku 5.1.

```

>>> QOS has no affect on retained messages
Creating client 1 with clean session set to True
Published message= OFF ,topic=bulb1 qos= 0 retain= True
Subscribed,topic=bulb1 QOS= 0
message received OFF No impact
Ending
>>> ===== RESTART =====
>>> Trying to delete retained message
Creating client 1 with clean session set to True
Published message= OFF ,topic=bulb1 qos= 0 retain= False
Subscribed,topic=bulb1 QOS= 0
message received OFF No impact
Ending
>>> ===== RESTART =====
>>>
Creating client 1 with clean session set to True
Published message= OFF ,topic=bulb1 qos= 0 retain= False
Subscribed,topic=bulb1 QOS= 0
message received OFF No impact
Ending
>>> ===== RESTART =====
>>>
Creating client 1 with clean session set to True
Published message= OFF ,topic=bulb1 qos= 0 retain= True
Subscribed,topic=bulb1 QOS= 0 Works
Ending

```

Obrázek 5.1: Výpis při publikování informací na MQTT téma při nastaveném *Retain flagu*

5.1.3 Asyncio Paho

MQTT tématům bylo třeba naslouchat i v kódu jednotlivých agentů v jazyce Python a využívajících *middleware* Spade. k tomu posloužila knihovna `paho-mqtt` [16], respektive její asynchronní varianta `asyncio-paho` [9].

Knihovna práci ulehčí možností použití metod zpětného volání, které naslouchají než přijde MQTT zpráva na dané téma, nebo v případě vlastního publikování lze díky knihovně definovat další logiku přesně v místě, kdy MQTT *broker* detekoval zprávu daného agenta. Často je v implementaci práce využita metoda s následujícím kódem:

Algoritmus 2 Tvorba „Paho“ klienta pro poslech MQTT zpráv. Agent si po příjmu zprávy uloží do znalostní báze přijatá data uložená do `client._userdata`.

```

async with AsyncioPahoClient(uuid) as client:
    client._userdata = None
    client.username_pw_set(username, password)
    _ = await mqtt.listen_to_topic(client, topic)
    await asyncio.sleep(SECONDS)
    self.set("temperature", client._userdata)

```

V příkladu si lze povšimnout, že MQTT integrace systému HomeAssistant vyžaduje autentizaci zařízení posílajících MQTT zprávy.

5.1.4 Další využití knihovny a nástroje

Knihovna *numpy* – matematické funkce, vektorizace, práce s poli... , internetové nástroje [draw.io](#) a [Lucidchart](#) pro tvorbu diagramů a [Eclipse Mosquitto](#) broker pro MQTT.

5.2 Struktura souborů

Kód programu a každý soubor je dokumentován, aby byla jasná funkcionality souborů. U důležitých metod určených ke znovupoužitelnosti je snaha o dokumentování dle konvencí PEP257 a PEP8 [4] [5].

5.3 Inicializace

Počáteční stav inicializuje, tj. uvede do chodu následující.

- Agenti jsou přihlášení na XMPP server. Reálné nasazení by se zde mělo obohatit ošifrováním hesel agentů.
- Do znalostní báze agentů jsou nastavena MQTT témata sledovaných artefaktů.
- Je nastaven čas a způsob posunu v čase během simulace.
- Je spuštěna sekvence sběru dat o prvotním nastavení teplotního modelu prostředí. Nebo jsou tyto údaje nastaveny manuálně.
- Nastaví se prvotní hodnoty parametrů znalostní báze agentů. Vytvoří se jejich Spade šablony a aktivují se Spade.Behaviour chování pro poslech daných témat, pro příjem zpráv od ostatních agentů, a kontrolu parametrů ve znalostní bázi.
- V případě, že v lokální síti ještě není spuštěný Home Assistant, spustíme jej, případně spustíme alespoň MQTT broker publikujeme data na MQTT témata dle manuálu v příloze.
- Po inicializaci zobrazení výpisu od učícího se agenta s informací, že se naučil strategii pro daný den.

5.4 Interakce uživatele

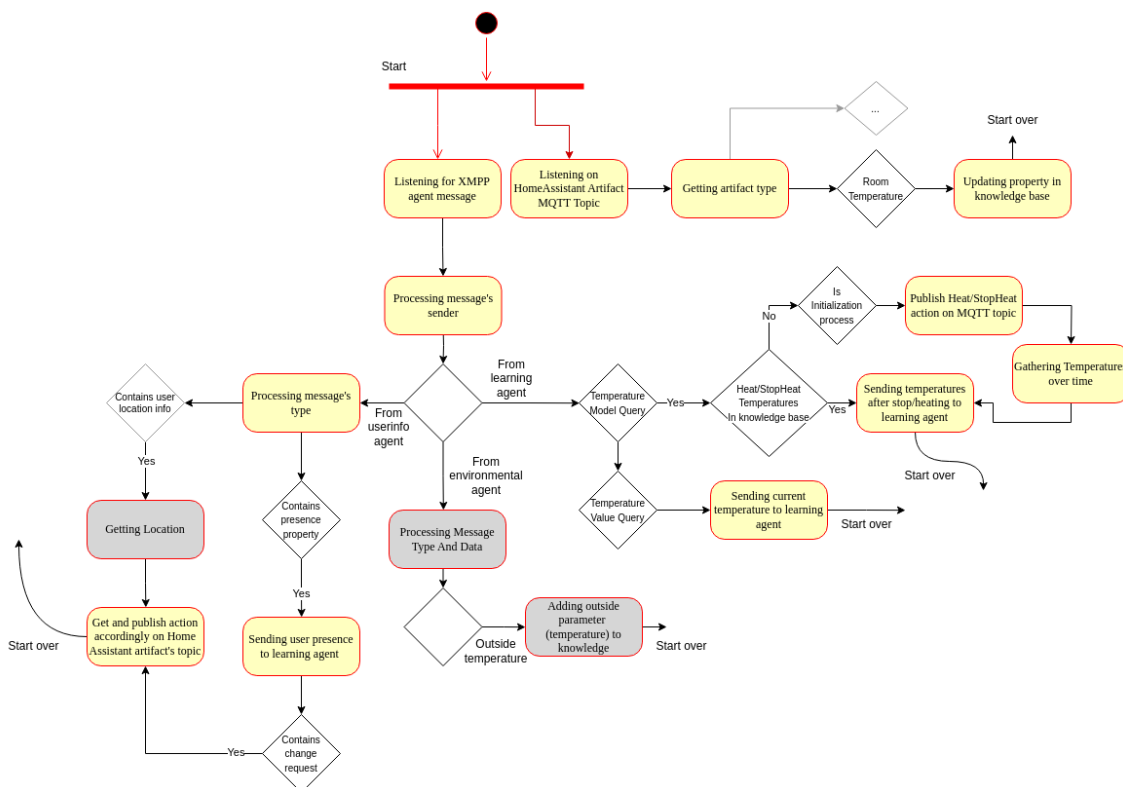
5.4.1 Časový plán

V okamžiku, kdy je nastaven časový plán, tj. informace, kdy začít a přestat topit v závislosti na dané teplotě, informace je odděleně publikovaná na témata. Informaci o plánu naslouchá agent pro uživatelské požadavky (dále *UserInfo* agent). Ten informaci zpracuje s časem, ve kterém ji přijal. V okamžiku, kdy agentovi přijde zpráva od učícího se agenta s obsahem „timeplan“, je toto pokyn pro odeslání časového plánu. Do té doby agent v simulaci s plánem nepracuje, pouze reaguje na požadavky uživatelů.

5.4.2 Změna vlastnosti prostředí

Požadavky změny konkrétní vlastnosti prostředí (tj. v simulaci teplota), zpracuje agent pro správu teploty v místnosti – **TemperatureAgent**. Agent má chování pro poskytnutí základní teploty prostředí a nebo pro odběr žádostí o zvýšení, či snížení teploty. V případě, že přijde žádost o změnu teploty, agent provede akci snížení nebo zvýšení a odešle učicímu se agentu notifikaci dle komunikačního standardu FIPA [1].

5.5 Komunikace agentů

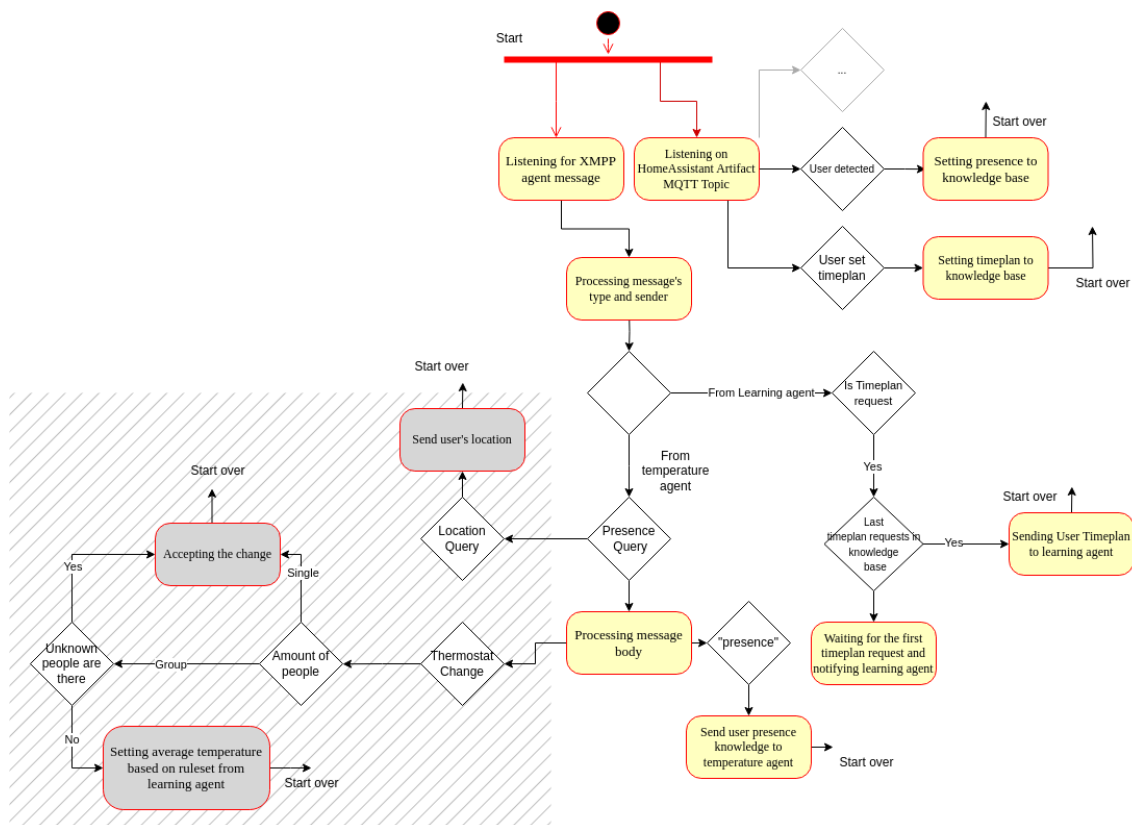


Obrázek 5.2: Implementace stavů agenta pro správu teploty; napodobitelný pro sběr dat z jiných senzorů. Šedě označená část není implementována v simulaci, ale je doporučena pro rozšíření reálného nasazení.

Pro popis komunikace v čase platí navržený diagram 4.1. V případě nepřítomnosti **UserInfo** agenta, agent pro správu teploty komunikuje s učícím se agentem rovnou po přijetí požadavku. Logika agentů je znázorněna v následujících stavových diagramech 5.2, 5.3. Společným prvkem agentů je cyklus příjmu zpráv od ostatních agentů. Šedě zbarvené stavy jsou v simulaci vynechány, jelikož by zkomplikovaly přehlednost vizualizace finálního výstupu. Pro reálné nasazení implementace lze zvážit rozšíření implementace o tyto stavy.

5.5.1 Asynchronní zpracování zpráv

Reflexní agenti znázornění na diagramech periodicky naslouchají daným MQTT tématům. Cyklické naslouchání bylo zvoleno pouze u zpráv s ostatními agenty, nicméně lze využít



Obrázek 5.3: Implementace stavů agenta pro správu uživatelských požadavků; šedě označená část není implementována v simulaci, ale je doporučena pro rozšíření reálného nasazení.

XMPP a Spade funkcionality poštovních schránek a kontrolovat jejich stav jednou za daný časový interval.

Poslech MQTT témat pomocí knihovny *Asyncio Paho* si žádá využít tzv. *callback* funkce (pro zpětné volání), jinými slovy odkazy na metody, které se mají zavolat až obdržíme zprávu na daném tématu. Jelikož mimo jiné, tyto odkazy představují sdílený datový prostředek, byl při poslechu témat využit princip semaforu [6]. Díky němu se agenti při využívání MQTT poslechu střídají. Pokud jsou agenti implementováni na jednom subsystému, je nutné použít tento, nebo podobný synchronizovaný přístup práce se sdílenými prostředky.

5.6 Učení a volba další akce

Implementaci učení zastupuje metoda `training` v souboru `learning_sim.py`. Fáze v implementaci učícího se agenta jsou:

- Před-startovní fáze učení modelu z vývoje artefaktů (teploty) v minulých intervalech,
- první iterace učení,
- odeslání návodu,
- přeučení a experimenty.

Učení je silně závislé na množství dat, které obdrží. Aby se dalo využít efektivně v reálných scénářích, je třeba získat data to vývoji relevantních fyzikálních veličin v domácnosti. V případě teploty v simulačním modelu této práce, je spuštěna sekvence počátečního sběru dat dle zaslanych požadavků na zvýšení, či snížení. Po obdržení dat z artefaktů je při úspěchu vytvořen základní model, později využitý pro tvorbu modelu vývoje teploty v místnosti a výsledku zvýšení a snížení teploty v čase.

Příklad funkce simulace: V první iteraci obyvatel domu nastaví požadavky na části dne, kdy topit a netopit (v závislosti na standardní teplotě, např. 20 °C). Tato informace se odešle agentu učení. V reálném nasazení se obohatí informacemi o přítomnosti a nepřítomnosti obyvatele. Učení probíhá dle vzorce pro metodu Q-Learning [25] .

$$Q(s, a) = Q(s, a) + \alpha \cdot \left(r(s') + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right)$$

Obrázek 5.4: Vzorec vyhodnocení Q-hodnot v daných stavech po provedení daných akcí.

Součástí stavového prostoru s je:

- přítomnost – dva stavy,
- derivace teploty – tři stavy,
- teplota – 20 stavů, číslo od 15 do 25 s granularitou po 0.5,
- časový interval – 15 minutové intervaly jednoho dne,

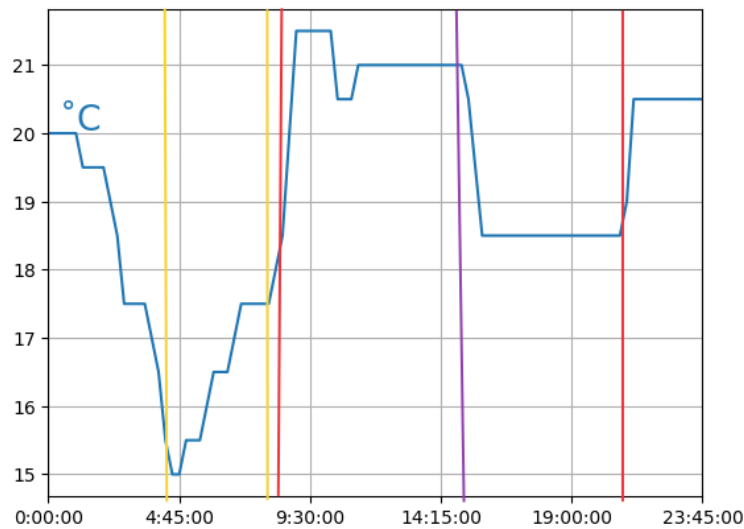
akcí učícího agenta je následně snížení teploty, či ponechání teploty, což může vyústit vypnutím termostatu, a zvýšením teploty v důsledku vyvolávajícím zapnutí termostatu, nebo ponecháním zapnutého termostatu. Akce se volí pomocí *epsilon-greedy* algoritmu, což znamená, že s faktorem určité náhody definované konstantou *epsilon* se nevybere vždy nejlépe ohodnocená akce, ale akce náhodná.

Z průchodů stavového prostoru se ve finále vyvodí ideální *set-pointy* termostatu, tedy body, kdy by se v domě měla zvyšovat, či snižovat teplota. Po analýze se vyvodí výsledný plán jednoho dne, v reálném systému by se následně měl hlídat plán pro všední dny, víkendy a případně roční období, vše dále v závislosti na přítomných osobách, o které by se musel obohatit stavový prostor 6.

Přidanou funkcionalitou algoritmu je napodobování případných požadavků obyvatele v daném čase. Za podmínky, že obyvatel domu v právě testovaném čase a při dané teplotě někdy zaslal požadavek na změnu teploty, je agent při učení pokaždé, kdy se do tohoto stavu znovu dostane, negativně penalizován. V případě, že se v danou chvíli nevyskytne požadavek modelovaného uživatele, je penalizace počítána dle doby, před kolika intervaly se naposledy vyskytl požadavek obyvatele. Do reálného systému by se tento způsob odměn měl modifikovat jednoduchým parametrem pro zdůraznění stáří daného požadavku.

Plán, který je následně výstupem učení se vytvoří ideálním průchodem stavového prostoru v čase v závislosti na Q hodnotách. Parametry algoritmu jsou:

- ϵ , zmíněn výše, ustálen na hodnotě 0.8
- *idling*, podobně jako ϵ při výběru akce přidává váhu volbě nicnedělání před úpravou hodnoty termostatu. (0.6)



Obrázek 5.5: Vizualizace výsledku učení. Interakcí modelovaného uživatele se systémem požadavků. Dle výsledné analýzy byl v tomto scénáři uživatel s teplotou spokojený 90.62% času. Klíč: červená barva: interval kdy modelovaný uživatel chce zvýšit teplotu z prvotních 20 °C v době reálného požadavku; Fialová: interval, kdy chtěl uživatel snížit teplotu z 20 °C – ukázalo se, že dle reakcí chtěl v daném intervalu teplotu kolem 21 °C; Žlutá: doba kdy simulovaný uživatel není v domě.

- faktor slevy ohodnocení stavu v závislosti na minulém stavu γ (0.2)
- koeficient učení, nastavený na pevnou hodnotu 0.99 (ve vzorci výše α)

Výsledkem učení je zde jak návod, při jakém stavu (přítomnost, derivace teploty, teplota, čas), provést jakou akci (zvýšit, snížit, ponechat). Pokud uživatel zasáhne novým požadavkem, reagují na něj v okamžiku pouze reflexní agenti a hlavní učení se s novou informací přeúčí a v dalším intervalu slouží jako vstup a návod, jak teplotu v místnosti ovládat.

5.6.1 Modelované objekty

Modelované objekty učícího se agenta, aneb uživatel a okolní prostředí mají v implementaci metody kromě dokumentace vhodné i k nastínění v tomto textu. Objekty nalezneme v souborech `sim_temperature.py` a `sim_user.py`

Mezi důležité parametry objektu simulovaného uživatele patří `target_temp_time` tedy seznam časových intervalů a teplot, které učící agent považuje za cíl daného uživatele. Výskyt těchto požadavků – cílů, je návaznou funkcí rozložen po celém dni pro aktivní zasílání zpětné vazby při učení. Objekt simulovaného uživatele `sim_temperature.py` dále obsahuje hodnoty pro zaznamenání odchodu a příchodu, kdy simulovaný uživatel nezasílá požadavky.

Model teploty okolního světa v souboru `sim_temperature.py`, rozšiřitelný na ostatní parametry, obsahuje algoritmus pro výpočet další teploty, dle současné teploty a teplot získaných 5 a 5 pokusných kroků popsaných v sekci inicializace 5.3.

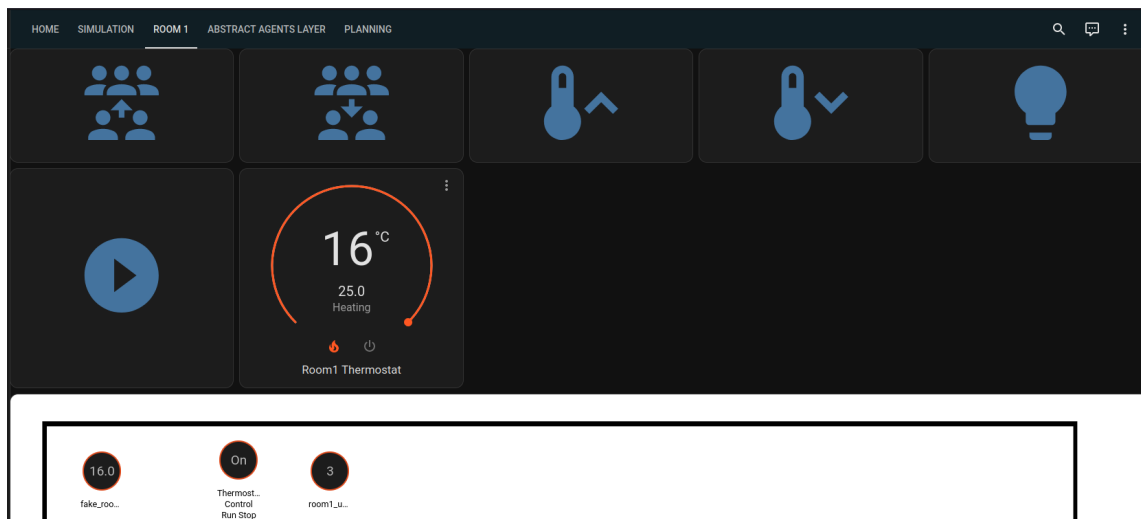
Algoritmus 3 Simulovaná změna teploty dle počtu kroků od posledního spuštění termostatu. Funkce `nearest_pointfive` je v práci široce používaná funkce vracující diskrétní hodnotu po půlstupních.

```
next_temperature = None
if is_decrease:
    next_temperature = nearest_pointfive(
        current_temperature*
        (1/self._EXP_STOP_CONSTANTS[step]))
else:
    next_temperature = nearest_pointfive(
        current_temperature*
        (self._EXP_HEAT_CONSTANTS[step]))
if BOTTOM_LIMIT <= next_temperature <= UPPER_LIMIT:
    return next_temperature
if next_temperature > UPPER_LIMIT:
    return UPPER_LIMIT
if next_temperature < BOTTOM_LIMIT:
    return BOTTOM_LIMIT
```

5.7 Home Assistant

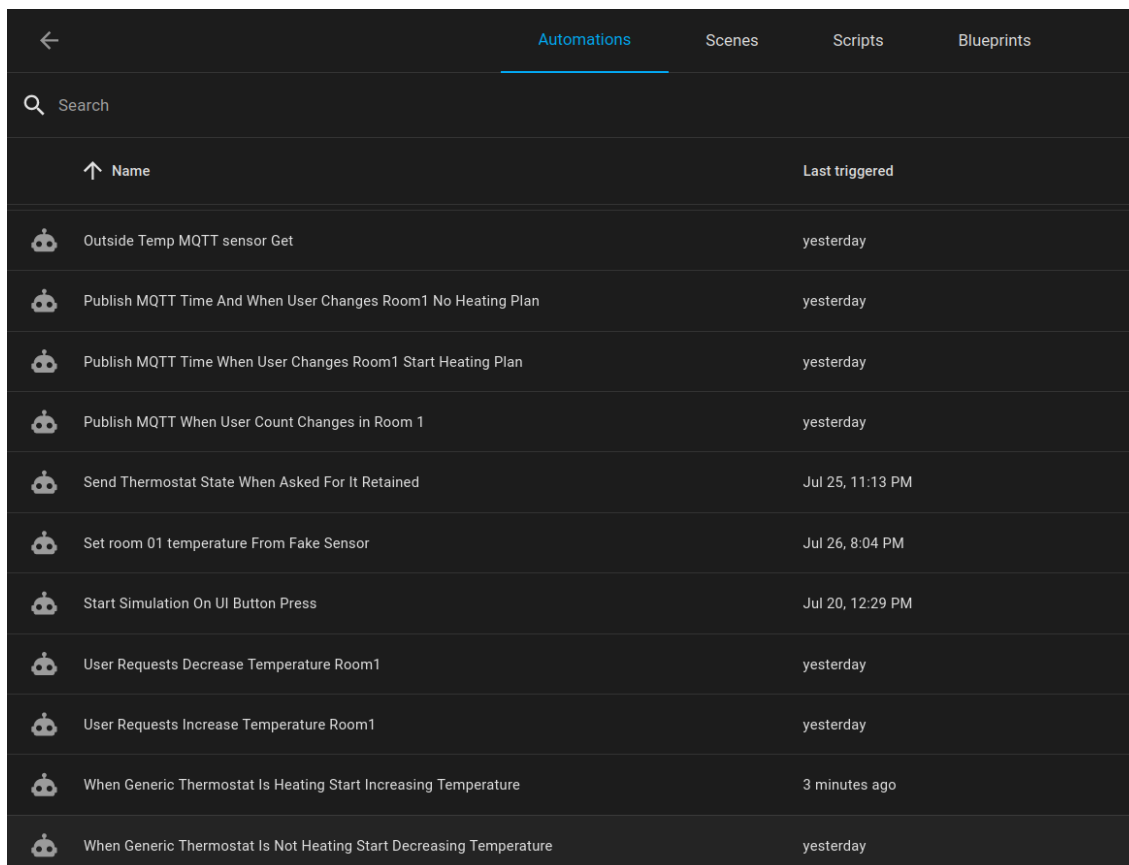
K této práci je poskytnut obraz, který lze importovat do nainstalovaného systému Home Assistant (varianta Docker). Instalace je dále popsána v manuálu v příloze.

Home Assistant je rozhraní pro automatizaci Smart Home, využitě v této práci. Vytvořili se zde automatizace publikující informace simulovaných senzorů na daných MQTT tématech. Dále zde bylo vytvořeno jednoduché rozhraní simulující požadavky uživatele.



Obrázek 5.6: Ukázka uživatelského rozhraní vytvořeného v Home Assistant. Ikony teplo-
měru a šipky publikují požadavek uživatele přes protokol MQTT

Během implementace logiky s využitím Home Assistant, byly nejčastěji využíván kon-
cept entit, které v tomto systému představují pomocné prvky, jako číslo, posuvník, nebo



	Automations	Scenes	Scripts	Blueprints
←	Search			
↑	Name	Last triggered		
👤	Outside Temp MQTT sensor Get	yesterday		
👤	Publish MQTT Time And When User Changes Room1 No Heating Plan	yesterday		
👤	Publish MQTT Time When User Changes Room1 Start Heating Plan	yesterday		
👤	Publish MQTT When User Count Changes in Room 1	yesterday		
👤	Send Thermostat State When Asked For It Retained	Jul 25, 11:13 PM		
👤	Set room 01 temperature From Fake Sensor	Jul 26, 8:04 PM		
👤	Start Simulation On UI Button Press	Jul 20, 12:29 PM		
👤	User Requests Decrease Temperature Room1	yesterday		
👤	User Requests Increase Temperature Room1	yesterday		
👤	When Generic Thermostat Is Heating Start Increasing Temperature	3 minutes ago		
👤	When Generic Thermostat Is Not Heating Start Decreasing Temperature	yesterday		

Obrázek 5.7: Ukázka názvů automatizací vytvořených v Home Assistant.

tlačítko, nicméně jejich běžné užití bylo pozměněno a využito k napodobení zpráv z artefaktů.

Často přitom byl využit na formátu **YAML** založený [Templating](#). Umožnil větší kontrolu nad výstupními daty a například využití kódování do formátu **JSON** pro simulování dat.

5.7.1 Alternativa

Jako jedna z existujících architektur je porovnáváno *Base Cube One*. Možná podoba reálného rozšířeného řešení z této práce se systému v některých faktech podobná. Zmíněný systém disponuje jednoduchou možností tvorby architektury dle lokace senzorů. Detaily nemohly být ověřeny, jelikož je použití systému za platební bránou. Home Assistant je zdarma, ale mnoho funkcionalit ke znázornění senzorů a prvků v prostoru nepodporuje a proto tato práce navrhuje využít i jiné rozhraní pro tvorbu systému Smart Home.

```
1 service: mqtt.publish
2 data:
3   qos: 1
4   retain: true
5   topic: room1/plan/heating
6   payload: >-
7     {{ strptime(states('input_datetime.input_heating01'), '%Y-%m-%d %H:%M:%S')
8     }}
9
```

Obrázek 5.8: Ukázka Home Assistant „Templatingu“. Publikována je zde informace o časovém plánu.

Kapitola 6

Závěr

6.1 Další výzkum

Tato sekce nabízí řešení vyvozená s problémy při návrhu a ze studií, které vedli k finální podobě práce.

6.1.1 Konkrétní Obyvatelé

Pokud bychom se chtěli zaměřit na komfort konkrétních obyvatel, je možné celý systém rozšířit. První možností je zaznamenávat přítomnost jednotlivců, nebo skupin. Agent spravující události nad uživateli může aplikovat logiku pro zpracování více příchozích požadavků najednou (např. zprůměrovat je) a učící se agent pracuje se znalostní bází dané skupiny o určitém počtu lidí. Ve finále se dále rozšíří jeho stavový prostor, ale ne výrazně.

Druhou zkoumanou možností je práce s přítomností konkrétních obyvatel budovy. Bezpochyby, je mnoho možností jak problém řešit. Značně se rozšíří funkcionalita fyzické vrstvy systému, která nyní vyžaduje rozhraní pro přihlášení uživatelů, nebo detekci zařízení, která obyvatelé vlastní. V případě, že by byl některý uživatel nedetekovatelný, by agent pro správu uživatel pracoval i druhým způsobem, kde by pouze reagoval na požadavky.

Dané ID se zpracuje a učícímu se agentovi se poskytne společně se záznamem o čase a provedené zpětné vazbě. Jelikož takových požadavků může v krátkých úsecích přijít více, nabízí se více přístupů jak tento problém řešit. Lze například přidat k těmto požadavkům váhu, a k akcím pravděpodobnost provedení, což dále vede na úpravu učení, s využitím *Policy Gradients* [13]. Pro tuto metodu by bylo třeba logiku učícího se agenta rozšířit o znalostní bázi pracující s tím, jak často je uživatel v domě a jak často například zasílá požadavky, jinými slovy, jak často je nespokojen.

Odlišná cesta je smlouvání mezi danými požadavky. Jako možnost se jeví vrstvu současného agenta pro správu uživatelů roztržít na více agentů zastupujících jednoho uživatele. Pokud bychom prostředí těchto agentů definovali jako kooperativní a kompetitivní zároveň, tedy, že se každý agent snaží co nejvíce uspokojit požadavky uživatele, jako možnost učení se jeví varianta, kdy tito agenti zasílají své stavy v iteracích centrálnímu učícímu se agentovi. Systém uživatelských agentů by se obohatil o učení a obdržení odměn, přestali by být reflexivní, ba naopak mnohem rozvinutější. . . Jak je zmíněno v [28], takové nastavení není příliš prozkoumané odvětví a řeší ho práce již uvedená v sekci problematika 3. Proto je v této práci ponechána jako způsob jiného řešení schůdnější varianta.

6.1.2 Porovnání existujících přístupů

Existujících řešení na téma aplikací zpřístupňujících jednoduchou tvorbu Smart Home s přidanou vrstvou inteligence není mnoho. Komerční řešení většinou nabídnou půdu pro propojení prvků IoT, nicméně po průzkumu reálných řešení nebyl nalezen žádný systém, který si člověk může například koupit a ihned pracovat s multiagentním prostředím rychle upravujícím jeho komfort. Dostupné jsou prvky jako chytrý termostat, který má za cíl vyhovět energetické spotřebě a nastavenému plánu, nikoli celé systémy.

Na druhou stranu existují návrhy věnující se tématu učení v chytré domácnosti. Mezi jeden takový patří [17]. Návrh agentů je zde proveden jiným způsobem. S uživatelem komunikují agenti rozhraní, kteří zároveň komunikují se systémem. Akce naučená druhým typem „inteligentních agentů“ je pak uživateli nabídnuta přičemž uživatel může vyjádřit spokojenost nebo nespokojenost s nabídnutou akcí. To je kontrastem této práce, jelikož ta chtěla komfort pojmout jako minimální zásahy uživatele do chodu. Oba přístupy se zdají být obstojné. Zmíněná práce rozšířila působnost na širší škálu IoT zařízení a dále využívá *Deep Reinforcement Learning*, přičemž předvedený algoritmus pouze počítá odměnu daným intervalům a příliš detailů není představeno, stejně jako reálné využití.

6.1.3 Splnění požadavků

Požadavky definované v kapitole problematiky 3 jsou nyní znovu posouzeny.

Generalizace systému na různé typy budov a místností se v simulačním modelu nestihla ověřit, nicméně návrh s tímto konceptem počítá a nic nebrání konceptu agentů sledovat dané místnosti.

Výstup simulace učení také ukázal, že probíhá přizpůsobení se uživatelům.

Pro práci s prostředím, umožňuje návrh tvorbu agentu správy prostředí. Mohou sledovat vlivy vnějšího světa, jako např. venkovní teplotu. V simulačním modelu tyto agenti nejsou testování, a model teploty nepracuje s parametrem venkovní teploty. Nicméně umožňuje stavět model teploty na reálných reakcích prostředí. Stejně tak spotřeba energie není v návrhu zmíněna. Nastíněno je, jak s touto problematikou pracovat. Snadno se o tuto informaci může obohatit učící se agent a stavět model teploty i v závislosti na spotřebě a venkovní teplotě.

Kritérium pro tvorbu architektury závisující na jednotlivých agentech splňuje. A díky MQTT je plně rozšiřitelný novými IoT zařízeními. Stejně tak se dá rozšířit i např. nahrazením, či modifikací algoritmu učení.

Zatímco požadavky testovaného uživatele systém průměrně na 90% uspokojil, na reálných uživateli koncept nebyl testován, i když je určitě cílem autora toto realizovat.

I přesto, že se práce dostala do mnohých úskalí, výsledek nabídl návrh a ukázkou, jakou cestou domácí automatizace zaměřené na komfort obyvatel, je možné se vydat. Rozhodně není tak pokročilý jako existující automatizační systémy. Koncept artefaktů a agentů v něm ale vytváří dynamičnost, jež při rozšíření systému umožní sledovat různé atributy domácnosti a učit se ovládat různé prvky, tak jak by si to přál uživatel, který zrovna není doma, nebo by rád před spaním nemusel nastavovat termostat, nastavovat klimatizaci a otvírat nebo zavírat okna.

Literatura

- [1] *Agents Communication Protocols*. Dostupné z: <http://ktiml.mff.cuni.cz/~pilat/en/multiagent-systems/communication-protocols/>.
- [2] *Emergent tool use from multi-agent interaction*. Dostupné z: <https://openai.com/research/emergent-tool-use>.
- [3] *MQTT 3.1.1 Specification*. Dostupné z: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>.
- [4] *PEP 257 – Docstring Conventions*. Dostupné z: <https://peps.python.org/pep-0257/>.
- [5] *PEP 8 – Style Guide for Python Code*. Dostupné z: <https://peps.python.org/pep-0008/>.
- [6] *Synchronization Primitives*. Dostupné z: <https://docs.python.org/3/library/asyncio-sync.html#asyncio.Semaphore>.
- [7] *Types of AI Agents*. Dostupné z: <https://www.javatpoint.com/types-of-ai-agents>.
- [8] *What is reinforcement learning*. Dostupné z: <https://www.synopsys.com/ai/what-is-reinforcement-learning.html>.
- [9] AMUNDSEN, T. *Asynchronous I/O (asyncio) Paho MQTT client*. Dostupné z: <https://pypi.org/project/asyncio-paho/>.
- [10] ARSLAN, G. a YÜKSEL, S. *Decentralized Q-Learning for Stochastic Teams and Games*. 2016.
- [11] ASSISTANT, H. *HomeAssistant MQTT Integrations*. Dostupné z: <https://www.home-assistant.io/integrations/mqtt/>.
- [12] COOK, D. J. Computer science. How smart is your home? *Science*. United States: [b.n.]. březen 2012, sv. 335, č. 6076, s. 1579–1581.
- [13] HEESWIJK, P. Wouter van. *Policy gradients in reinforcement learning explained*. Towards Data Science, Apr 2023. Dostupné z: <https://towardsdatascience.com/policy-gradients-in-reinforcement-learning-explained-ecec7df94245>.
- [14] KULKARNI, A. *How to explore timescaledb using simulated IOT sensor data*. Timescale Blog, Jul 2023. Dostupné z: <https://www.timescale.com/blog/how-to-explore-timescaledb-using-simulated-iot-sensor-data/>.

- [15] LI, T., SHAO, G., ZUO, W. a HUANG, S. Genetic Algorithm for Building Optimization: State-of-the-Art Survey. In: únor 2017, s. 205–210. DOI: 10.1145/3055635.3056591.
- [16] LIGHT, R. *Paho-Mqtt*. Dostupné z: <https://pypi.org/project/paho-mqtt/>.
- [17] MIRRA, J. a ANALIDE, C. Cover Reinforcement Learning Based Approach for Smart Homes. In: SILVA, F., ed. *Intelligent Environments 2018*. Intelligent Environments 2018. Sv. 23, s. 38–47. Ambient Intelligence and Smart Environments. DOI: 10.3233/978-1-61499-874-7-38.
- [18] NOWÉ, A., VRANČX, P. a DE HAUWERE, Y.-M. Game Theory and Multi-agent Reinforcement Learning. In: WIERING, M. a OTTERLO, M. van, ed. *Reinforcement Learning: State-of-the-Art*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 441–470. DOI: 10.1007/978-3-642-27645-3_14. ISBN 978-3-642-27645-3. Dostupné z: https://doi.org/10.1007/978-3-642-27645-3_14.
- [19] OMICINI, A., RICCI, A. a VIROLI, M. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*. Dec 2008, sv. 17, č. 3, s. 432–456. DOI: 10.1007/s10458-008-9053-x. ISSN 1573-7454. Dostupné z: <https://doi.org/10.1007/s10458-008-9053-x>.
- [20] PALANCA, J., RINCON, J., JULIAN, V., CARRASCOSA, C. a TERRASA, A. Developing IoT Artifacts in a MAS Platform. *Electronics*. 2022, sv. 11, č. 4. DOI: 10.3390/electronics11040655. ISSN 2079-9292. Dostupné z: <https://www.mdpi.com/2079-9292/11/4/655>.
- [21] PALANCA, J., TERRASA, A., JULIAN, V. a CARRASCOSA, C. SPADE 3: Supporting the New Generation of Multi-Agent Systems. *IEEE Access*. 2020, sv. 8, s. 182537–182549. DOI: 10.1109/ACCESS.2020.3027357.
- [22] PUIG, X., SHU, T., LI, S., WANG, Z., TENENBAUM, J. B. et al. Watch-And-Help: A Challenge for Social Perception and Human-AI Collaboration. *CoRR*. 2020, abs/2010.09890. Dostupné z: <https://arxiv.org/abs/2010.09890>.
- [23] SAYS:, S. *MQTT for beginners tutorials and course*. Jul 2023. Dostupné z: <http://www.steves-internet-guide.com/mqtt-basics-course/>.
- [24] VAVILOV, D., MELEZHIK, A. a PLATONOV, I. Smart home user's behavior prediction. In: Září 2013, s. 1–4. DOI: 10.1109/ICCE-Berlin.2013.6697984. ISBN 978-1-4799-1411-1.
- [25] WATKINS, C. J. C. H. a DAYAN, P. Q-learning. *Machine Learning*. May 1992, sv. 8, č. 3, s. 279–292. DOI: 10.1007/BF00992698. ISSN 1573-0565. Dostupné z: <https://doi.org/10.1007/BF00992698>.
- [26] WOLFE, S., SIERHUIS, M. a JARVIS, P. To BDI, or not to BDI: Design choices in an agent-based traffic flow management simulation. In: Leden 2008, s. 63–70. DOI: 10.1145/1400549.1400558.
- [27] ZEILER, W., TIMILEHIN, L. a ADUDA, K. Towards Multi-agent Systems in Building Automation and Control for Improved Occupant Comfort and Energy Efficiency -

State of the Art, Challenges. In: Listopad 2013, s. 718–722. DOI: 10.1109/ISDEA.2013.572.

- [28] ZHANG, K., YANG, Z. a BASAR, T. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. *CoRR*. 2019, abs/1911.10635. Dostupné z: <http://arxiv.org/abs/1911.10635>.

Příloha A

Manuál

A.1 Základní instalace

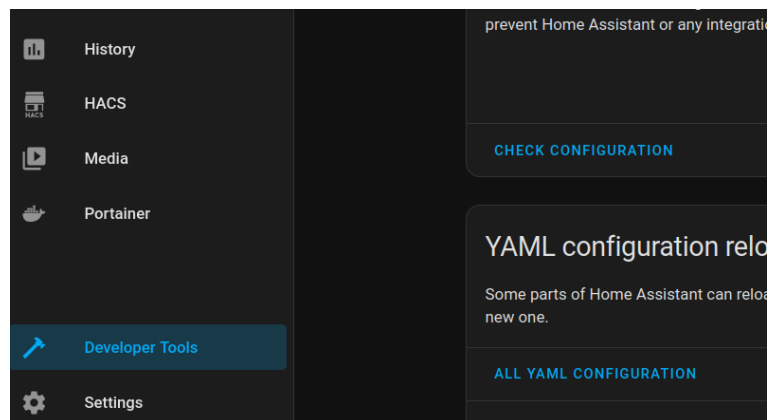
Tato instalace nepočítá s použitím systému Home Assistant.

Pokyny k ovládání simulačního modelu: Nainstalujte a spusťte MQTT Broker [dle manuálu](#), nebo dle následujících pokynů zadávejte MQTT příkazy:

1. publikovat na no_heat a heat topic alespoň jeden datum časového plánu:
 - (a) `mosquitto_pub -h localhost -t room1/plan/heating -r -m "2023-07-01 17:00:00"`
 - (b) `mosquitto_pub -h localhost -t room1/plan/noheating -r -m "2023-07-01 19:00:00"`
2. instalujte potřebné Python knihovny příkazem
`python3 -m pip install -r requirements.txt`
3. spusťte `./src/sim.py` bez parametrů
4. publikujte požadavky zvýšení a snížení teploty:
 - (a) `mosquitto_pub -h localhost -t house/room1/user/temp_increase -r -m "new"`
 - (b) `mosquitto_pub -h localhost -t house/room1/user/temp_decrease -r -m "new"`

A.2 Pokročilá instalace

1. Instalujte HomeAssistant dle návodu v [tomto](#) odkazu.
2. v Home Assistant rozhraní Portainer zvolte možnost importovat Image
3. zvolte soubor image na datovém médiu této práce
4. zkopírujte konfigurační soubory z datového média a adresáře hass do adresáře `/opt/homeassistant/config`
5. zkopírujte konfigurační soubory z datového média a adresáře mosquitto do `opt/mosquitto/config`.



Obrázek A.1: Reset HASS konfigurace.

6. restartujte konfigurace dle obrázku [A.1](#)
7. V případě že vše funguje, nastavte časový plán v záložce Overview – Planning
8. instalujte potřebné Python knihovny příkazem

```
python3 -m pip install -r requirements.txt
```

 - (a) Spustíte
 - (b) spustíte `./src/sim.py` bez parametrů pro následné napodobení reakcí na požadavky v záložce Overview – Room1
 - (c) alternativně s parametrem `real`, pro reálný čas a prvotní inicializaci, k funkcionality je nutné dodržet předchozí kroky a ujistit se že dané Home Assistant entity mají existující atributy. Pokud něco nefunguje, „proklikejte“ prvky UI v kartě Overview – ...