VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ADAPTIVNÍ SYSTÉM PRO ŘÍZENÍ OSVĚTLENÍ VE SMART HOME

ADAPTIVE LIGHTING CONTROL SYSTEM IN SMART HOME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ VALÍK**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

156375

Ústav: Ústav inteligentních systémů (UITS)
Student: **Valík Tomáš**
Program: Informační technologie
Název: **Adaptivní systém pro řízení osvětlení ve Smart Home**
Kategorie: Umělá inteligence
Akademický rok: 2023/24

Zadání:

1. Prostudujte problematiku automatizace budov a Smart Home a existující přístupy, aplikující metody strojového učení v řízení.
2. Zaměřte se na subsystém řízení osvětlení. Zvolte vhodné senzory, aktuátory, uživatelská rozhraní a navrhněte inteligentní řízení tak, aby se systém průběžně na základě senzorických dat, interakcí uživatele se systémem a omezujících podmínek učil maximalizovat komfort uživatelů.
3. Navrženou řídicí aplikaci implementujte tak, aby byla použitelná jak v simulovaném, tak v reálném prostředí.
4. 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 vedoucího.

Při obhajobě semestrální části projektu je požadováno:
První 2 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: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 6.11.2023

Abstrakt

Práce se zabývá problematikou řízení osvětlení v chytré domácnosti. Ve většině chytrých domácností je nutné ovládat osvětlení ručně pomocí spínačů nebo mobilních zařízení. V práci je představen adaptivní řídící systém založený na rekurentních neuronových sítích, který se postupně naučí uživatelskou manipulaci s osvětlením a po určitém čase začne řídit osvětlení samostatně.

Abstract

The thesis deals with the issue of lighting control in smart homes. In most smart homes, it is necessary to manually control the lighting using switches or mobile devices. The thesis introduces an adaptive control system based on recurrent neural networks, which gradually learns user manipulation with the lighting and eventually begins to independently control the lighting after a certain period of time.

Klíčová slova

neuronové sítě, rekurentní neuronové sítě, LSTM, chytrá domácnost, domácí automatizace, optimalizace komfortu, adaptace

Keywords

neural networks, recurrent neural networks, LSTM, smart home, home automation, comfort optimization, adaptation

Citace

VALÍK, Tomáš. *Adaptivní systém pro řízení osvětlení ve Smart Home*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Adaptivní systém pro řízení osvětlení ve 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.

.....
Tomáš Valík
6. května 2024

Poděkování

Rád bych poděkoval vedoucímu práce Doc. Ing. Vladimíru Janouškovi Ph.D., za časté konzultace a rady při řešení práce. Dále bych chtěl poděkovat své rodině a přátelům za podporu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Moderní technologie postupují neustále dopředu, lidská civilizace se neustále snaží zjednodušovat si práci a zvyšovat pohodlí.

Pokud se řekne slovní spojení „chytrá domácnost“, vybaví se mi myšlenka na domácnost, ve které majitel nemusí vykonávat žádnou namáhavou činnost. Jeho domov se jednoduše postará o vše samostatně. Jídlo se vaří, oblečení se pere, nečisté nádobí se myje, a to vše bez nutnosti zásahu majitele.

Po příchodu majitele do domu se na základě sensorů a rozpoznaní obličeje okamžitě otevírají vchodové dveře, nastaví se komfortní osvětlění a uživatel se nemusí trápit zbytečnostmi a může v poklidu relaxovat.

Právě na toto téma je založena má bakalářská práce, jejímž cílem je vytvořit chytrý řídící systém pro ovládaní osvětlení v chytré domácnosti bez nutnosti zásahu uživatele.

Realné nasazení systému je samozřejmě poměrně náročným problémem, a proto je v práci navržena simulace domácnosti, na které se inteligentní systém postupně naučí denní rytmus a preference uživatelů na základě dat ze simulovaných senzorů a uživatelské manipulace s osvětlením.

Hlavním cílem této práce je tedy navrhnout systém na zvýšení komfortu obyvatel chytré domácnosti, která je v dnešní době na vzestupu, a poukázat na to, kam nás mohou moderní, stále se vyvíjející technologie posunout v oblasti domácí automatizace.

V kapitole 2 se nachází úvod do řešené problematiky, obdobné přístupy pro predikci uživatelského chování a použité nástroje pro vytvoření adaptivního systému. Kapitola 3 se zaobírá návrhem simulačního modelu, použitými technologiemi a návrhem řídícího systému. V kapitole 4 je popsána implementace simulačního modelu a řídícího systému. Kapitola 5 se věnuje experimentům se systémem, jejich vyhodnocení a následné diskuzi o přechodu k reálnému nasazení.

Kapitola 2

Uvedení do problematiky

Tato kapitola pojednává o problematice v chytré domácnosti a rozebírá obdobné přístupy v automatizaci chytrých budov.

Technologie Smart Home prošla během let významným vývojem. Začátky sahají až do sedmdesátých let, kdy nastala revoluce díky systému X10[3], který přinesl možnost dálkového ovládání. Další přelom nastal na začátku dvacátého století, kde bezdrátové technologie jako Z-wave[18] a Zigbee[5] přinesly nové možnosti, umožňující snažší instalaci a lepší zabezpečení.

Dnešní systémy jsou často otevřené a nadmíru přispůsobitelné a to hlavně díky open-source projektům jako openHAB[9] a Home Assistant[1], které dávají uživatelům vysokou flexibilitu.

2.1 IoT - Internet věcí

Internet věcí (IoT) označuje zařízení propojená přes internet nebo jiné sítě, která spolu vzájemně komunikují. Objekty fyzického světa mohou být připojeny do Internetu prostřednictvím senzorů.

V konzumním trhu je technologie IoT nejvíce spojována se Smart Home, včetně zařízení a spotřebičů (osvětlovací tělesa, termostaty, systémy domácí bezpečnosti, kamery a další), které podporují jednu nebo více běžných ekosystémů a lze je ovládat pomocí zařízení propojených s tímto ekosystémem, jako jsou chytré telefony a hlasoví asistenti[22].

Obrázek 2.1: IoT ve spojení se Smart Home, převzato z [4]

2.2 MQTT protokol

MQTT (message queuing telementry) je lekhý standardní komunikační protokol používaný pro M2M(machine to machine) komunikaci. Zařízení IoT využívají MQTT k efektivnímu přenosu dat přes síť s omezenými zdroji a šířkou pásma. Tento protokol umožňuje snadnou implementaci a efektivní komunikaci mezi zařízeními a cloudem. Historicky byl MQTT vyvinut v roce 1999 pro monitorování ropovodů a v roce 2010 se stal volně dostupným a otevřeným standardem[2]. MQTT zajišťuje tyto důležité vlastnosti:

- lehkost a efektivita
- škálovatelnost
- spolehlivost
- bezpečnost

2.2.1 Klient

Jedná se o jakékoliv zařízení které běží nad MQTT knihovnu. Pokud klient odesílá zprávy, jedná jako vydavatel, pokud zprávy přijímá, zastupuje příjemce. Každé zařízení, které může komunikovat v síti je MQTT klient[2].

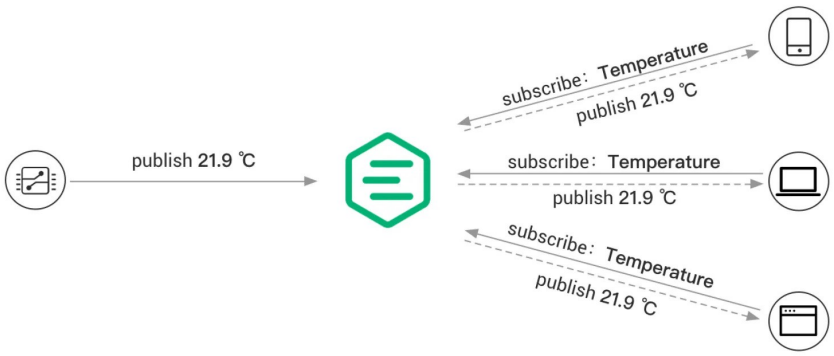
2.2.2 Broker

Je zodpovědný za koordinování zpráv mezi klienty. Dále zajišťuje funkce pro přijímání a filtrování zpráv, identifikaci klientů, kteří odebírají zprávy a zasílaní zpráv[19].

2.2.3 Vzor publish-subscribe

Tento vzor se liší od vzoru klient-server tím, že odděluje klienta, který odesílá zprávy (publisher) od klienta, který přijímá zprávy (subscriber). Klienti nepotřebují navázat přímé spojení a za směrování a distribuci všech zpráv je zodpovědný MQTT Broker.

Následující obrázek zobrazuje MQTT proces publish/subscribe. Teplotní senzor se připojuje k MQTT serveru jako klient a publikuje teplotní data na určité téma, server přijímá zprávu a přeposílá ji klientovi přihlášenému k tématu.



Obrázek 2.2: Publish/subscribe proces, převzato z [19]

2.2.4 Topic(téma)

Témata jsou hierarchické řetězce, které definují kategorii zprávy. Když publisher odesílá zprávy na broker, přiřazují se k určitému tématu. Subscriber se poté může přihlásit k jednomu nebo více tématům. Broker poté směruje zprávy k příslušným klientům na základě jejich odběru témat[20].

Témata se hierarchicky strukturována pomocí „/“ podobně jako URL cesty. Možný zápis může vypadat například takto:

- senzorT/pokoj/teplota
- senzorV/pokoj/vlhkost

V zápisech jsou také podporovány zástupné znaky[19].

2.2.5 Quality of service

Jedná se o mechanismus, který nabízí různé úrovně služeb pro spolehlivou komunikaci. Čím vyšší je úroveň, tím vyšší je spolehlivost doručení zprávy, ale také složitější proces přenosu[23]. Kvalita služby se dělí do tří úrovní:

- QoS 0: Zpráva je doručena nejvýše jednou.
- QoS 1: Zpráva je doručena alespoň jednou.
- QoS 2: Zpráva je doručena právě jednou.

2.3 Simulace

Nedílnou součástí práce je vytvoření simulačního modelu, který vhodně reprezentuje a abstrahuje reálnou domácnost.

Model nám může prokázat validitu adaptivního systému a jak dokáže reagovat na případné změny chování uživatelů.

Jako simulační nástroj byl zvolen PowerDEVS, především pro svoji robustnost a škálovatelnost.

2.3.1 PowerDEVS

PowerDEVS je integrovaný, open-source nástroj pro modelování a simulaci založený na formálním rámci Discrete Event System Specification (DEVS).

Nástroj umožňuje definovat atomické modely DEVS, které lze poté propojit do hierarchických blokových diagramů, aby bylo docíleno možnosti vytvářet složitější systémy. Prostředí automaticky překládá graficky propojené modely do kódu v jazyce C++ , který provádí simulaci.

PowerDEVS také obsahuje kompletní knihovnu pro simulaci spojitých a hybridnách systémů pomocí metod Quantized State System (QSS). Metody provádějí numerické aproximace spojitých časových systémů. Tyto aproximace lze vyjádřit jako modely DEVS[7].

2.4 Predikce chování uživatelů

V případě situací, kdy je žádoucí zvýšit komfort uživatelů, což lze chápat jako vykonání určitých aktivit za uživatele, je nutné mít schopnost do jisté míry odhadnout jejich budoucí záměry a chování. Tato problematika je předmětem zkoumání v řadě vědeckých prací.

V článku[10] se například autoři pokusili vytvořit co nejkomfortnější prostředí pro uživatele na základě analýzy jejich emocí a reakcí, včetně rozpoznávání výrazů obličeje. Tyto poznatky byly následně využity k adaptaci prostředí v chytré domácnosti, jako je například osvětlení, zvuky a další parametry, s cílem automaticky reagovat na potřeby a preference uživatele a vytvořit tak co nejpohodlnější a nejefektivnější prostředí.

V další práci[6] se zabývali zkoumáním energetické spotřeby v chytré domácnosti, což vyžadovalo predikci uživatelského chování k dosažení efektivnější úspory energie. Aby dosáhli těchto cílů, navrhli dva modely uživatelského chování a dva algoritmy pro predikci chování. Prvním modelem uživatelského chování byl Day Type Model, kde se dny klasifikují do různých typů na základě podobnosti v užívání spotřebičů. Dalším modelem byl Semi-Markov Model, který představuje uživatelský model pomocí markovského procesu a umožňuje libovolné rozdělení pravděpodobnostní funkce mezi dvěma uživatelskými interakcemi. Jedná se o model prvního řádu, který nezohledňuje kontextové informace, kde akce uživatele závisí pouze na předchozí akci. Pravděpodobnost přechodu mezi dvěma akcemi se poté odhaduje na základě empirických dat.

V práci[12] navrhli nový algoritmus nazvaný Unsupervised User Behavior Prediction (UUBP), který využívá umělou neuronovou síť a faktor zapomínání k překonání nedostatků předchozích předpovědních algoritmů. Tento algoritmus má vysokou úroveň autonomního a samoorganizujícího se učení a lépe zohledňuje infrekventované a zastaralé operační záznamy uživatelů.

V práci se autorům povedlo experimenty prokázat, že jejich navržený algoritmus je v porovnání s ostatními o mnoho výkonější.

2.5 Neuronové sítě

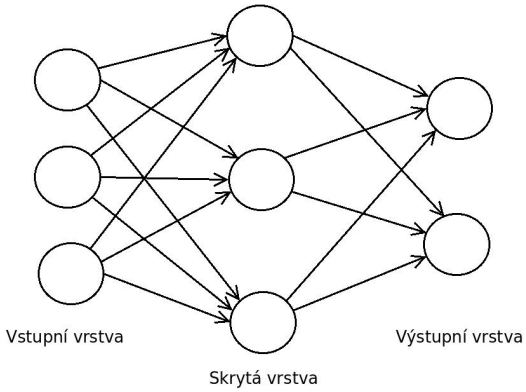
Jde o typ umělé inteligence, který je vzdáleně inspirován stavbou lidského mozku. Neuronové sítě představují jedno z nejzajímavějších programovacích paradigmat, které kdy bylo objeveno. V tradičním přístupu k programování určujeme počítači, jak má vykonávat úkoly tím, že rozdělujeme složité problémy na podprobémy, dobře definované úkoly, které jsou počítači snadno pochopitelné. Na rozdíl od toho u neuronových sítí nepředepisujeme počítači, jak má řešit daný problém. Místo toho se učí z pozorovaných dat a hledá vlastní řešení problému[13].

Základním stavebním prvkem neuronové sítě je matematický model nazývaný umělý neuron 2.5.1. Tyto umělé neurony jsou v neuronových sítích organizovány do vrstev, přičemž rozlišujeme tři hlavní typy vrstev:

- Vstupní vrstva
- Skrytá vrstva
- Výstupní vrstva

Data nejdříve projdou vstupní vrstvou a následují jedna nebo více skrytých vrstev, z nichž se poté dostanou na výstupní vrstvu.

Pokud má neuronová síť alespoň dvě skryté vrstvy, označuje se jako hluboká neuronová síť [14].



Obrázek 2.3: Struktura dopředné neuronové sítě

2.5.1 Umělý neuron

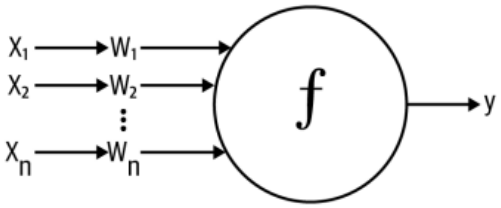
Jak už bylo dříve popsáno umělý neuron, je základní stavební jednotkou neuronové sítě. Stejně jako v biologických neuronech, umělý neuron přijímá několik vstupů, $x_1, x_2, \ldots, x_n$, z nichž každý je násoben určitou vahou, $w_1, w_2, \ldots, w_n$. Tyto vážené vstupy jsou sečteny tak, aby vytvořily logit neuronu.

$$z = \sum_{i=0}^{n} w_i x_i \tag{2.1}$$

. V mnoha případech logit zahrnuje také bias, což je pouze přičtená konstanta. Logit je poté předán funkcí f k vytvoření výstupu $y = f(z)$. Tento výstup může být přenesen na další neurony. Pokud se vstupy přeformulují jako vektor $x = [x_1, x_2, \ldots, x_n]$ a váhy neuronu jako $w = [w_1, w_2, \ldots, w_n]$. Potom je možné znovu vyjádřit výstup neuronu jako:

$$y = f(x \cdot w + b) \tag{2.2}$$

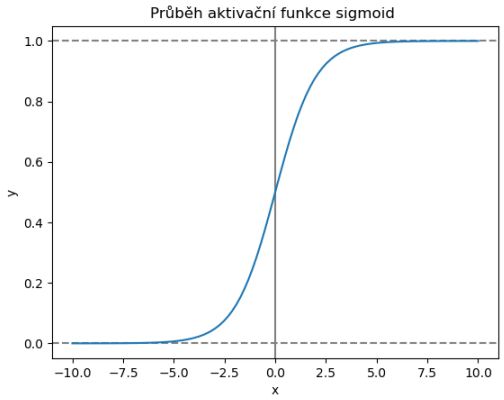
Kde b značí bias. Výstup je možné spočítat provedením skalárního součinu vstupního a váhového vektoru, přičtením biasu k vytvoření logitu a pak aplikací transformační funkce[14].



Obrázek 2.4: Struktura umělého neuronu, převzato z [14]

2.5.2 Aktivační funkce

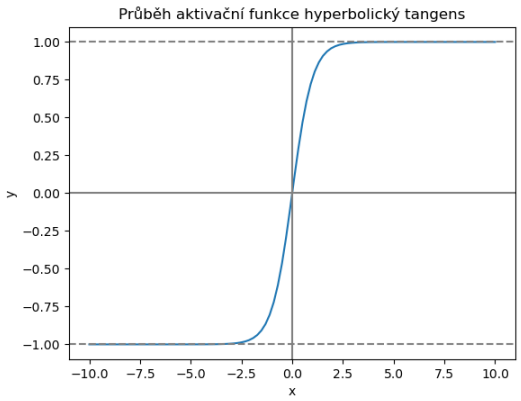
Aktivační funkce jsou speciálně používány v umělých neuronových sítích k transformaci vstupního signálu na výstupní signál, který je následně předán jako vstup do další vrstvy v hierarchii. V umělé neuronové síti spočítáme součet součinů vstupů a jejich odpovídajících vah a nakonec na něj aplikujeme aktivační funkci, abychom získali výstup této konkrétní vrstvy a dodali ho jako vstup do další vrstvy. Neuronové sítě preferují použití nelineárních aktivačních funkcí před lineárními, protože nelineární funkce umožňují síti zachytit složitější vztahy v datech. Lineární aktivační funkce by způsobily, že by se síť chovala podobně jako lineární regresní model, kde by výstup byl pouze lineární transformací vstupu. To by síti neumožnilo adaptovat se na nelineární charakteristiky dat, což je klíčové pro efektivní učení z chybných dat a získávání komplexních poznatků[17].



Obrázek 2.5: Aktivační funkce sigmoid

Aktivační funkce sigmoid 2.5 je často používána jako binarní klasifikátor. Transformuje hodnoty na interval $(0, 1)$. Její matematický zapis je definován takto:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$



Obrázek 2.6: Aktivační funkce hyperbolický tangens

Aktivační funkci hyperbolický tangens(tanh) můžeme popsat rovnící:

$$f(x) = 2\left(\frac{1}{1+e^{-2x}}\right) - 1 \tag{2.4}$$

Jak je z obrázku 2.6 vidět transformuje nám hodnoty na interval $(-1, 1)$. Nadále je tanh upřednostňován před sigmoid funkcí, díky tomu, že jeho gradienty nejsou omezeny na určitý směr.

Mezi další široce používané aktivační funkce patří softmax, což je funkce definována jako kombinace více sigmoid funkcí. Je používána jako vícetřídní klasifikátor a je možné ji popsat následovně[17]:

$$\sigma(z)j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \quad \text{pro } j = 1, \ldots, K \tag{2.5}$$

2.5.3 Trénování neuronové sítě

Trénování neuronové sítě je proces, během kterého dochází k optimalizaci vah mezi jednotlivými neurony[14].

První fáze trénování neuronové sítě, nazývaná dopředný průchod, je proces, během něhož vstupní data procházejí všemi vrstvami a generují výstup. V další fázi se výstupní data porovnají s cílovými(očekávanými) daty, aby mohlo následně dojít k vypočítání chyby. Ztrátová funkce (anglicky loss function) je matematická funkce, která měří rozdíl mezi predikovanými výstupy sítě a skutečnými cílovými hodnotami pro daná vstupní data. Existuje celá řada loss funkcí, ta se ale vždy vybíra na základě problematiky, která se má řešit. Například mean square error je používána pro regresní úlohy a cross entropy loss pro klasifikační úlohy.

Na minimalizaci loss funkce se často používá metoda zvaná gradientní sestup (gradient descent). Tato metoda využívá derivace loss funkce vzhledem k váhám sítě k tomu, aby určila směr a velikost změn vah, které povedou ke snížení chyby. Lokální minimum loss funkce představuje bod, ve kterém je chyba sítě minimální v okolí daných vah. Gradient descent může konvergovat k lokálnímu minimu, které ale nemusí být globálním minimem,tím je myšleno nejlepší možné řešení pro danou úlohu. To může být problémem, zejména pokud má loss funkce mnoho lokálních minim a globální minimum je těžké najít. Kvůli snížení nároků na výpočetní výkon se častěji používá varianta gradient descent a to Stochastic Gradient Descent (SGD).

Hlavním rozdílem mezi SGD a klasickým gradient descent je, že SGD nepoužívá všechna trénovací data k výpočtu gradientu najednou, ale pouze podmnožinu dat. Gradienty pro váhy ve skrytých vrstvách se počítají pomocí algoritmu zvaného zpětná propagace chyby (backpropagation). Tento algoritmus umožňuje vypočítat gradienty loss funkce vzhledem k váhám v celé síti pomocí řetězového pravidla(chain rule) derivací.

Algoritmus backpropagation funguje tak, že nejprve se vypočítá chyba (loss) sítě pro daný vstup a skutečný výstup. Poté se gradienty chyby vzhledem k váhám spočítají postupně vzhledem ke každé vrstvě sítě, začínaje výstupní vrstvou a postupující zpět k vstupní vrstvě. Tento proces se opakuje pro každý minibatch trénovacích dat.

Gradienty jsou spočítány pomocí derivace aktivačních funkcí každého neuronu ve skrytých vrstvách a derivace loss funkce vzhledem k výstupům těchto neuronů. Tyto gradienty jsou poté zpětně propagovány skrze síť a využity k aktualizaci vah pomocí SGD[11].

2.5.4 Rekurentní neuronové sítě

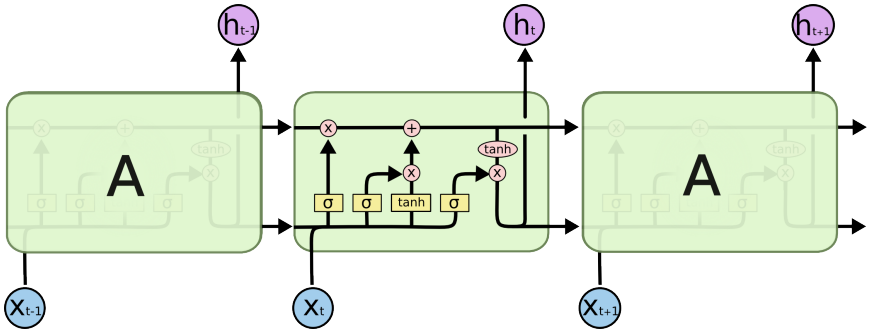
Rekurentní neuronové sítě (RNN) představují určitou variantu neuronových sítí, které slouží k analýze a zpracování sekvenčních dat. Jejich hlavní výhodou je schopnost zachytit vztahy a závislosti mezi jednotlivými prvky v sekvenci a pracovat s daty proměnné délky Této schopnosti je docíleno způsobem, že při výpočtu aktualního stavu vyžadují společně s daty také informace o stavu předchozím. Další vlastností RNN je sdílení vah mezi jednotlivými časovými kroky. Namísto toho, aby se učily nové váhy pro každý časový krok, RNN používají stejné váhy pro všechny časové kroky. Při trénování RNN může docházet k problému s trvanlivostí gradientu, kdy se gradient ztrácí nebo exploduje během zpětného šíření chyb, což může ovlivnit schopnost sítě naučit se dlouhodobé závislosti[8].

2.5.5 Long short term memory

Long short term memory(LSTM) je typ rekurentních neuronových sítí, který byl představen, aby vyřešil problém mizejícího gradientu[14]. Hlavní přínos LSTM sítí, také spočívá v tom, že se dokáží naučit dlouho trvající závislosti v datech. Stejně jako RNN mají řetězcovou strukturu, ale namísto jedné interaktivní vrstvy mají vrstvy čtyři. LSTM má schopnost přidat nebo odebrat informaci do stavu buňky(reprezentace dlouhodobé paměti), která je regulována pomocí struktur nazývaných „brány“. Brány jsou způsob omezit průchod informace. Skládají se z aktivační funkce sigmoid a bodového násobení. LSTM má 3 typy těchto bran – zapomínací bránu, vstupní bránu a výstupní bránu. Slouží k ochraně a kontrole stavu buňky.

V prvním kroku při trénování se rozhodne, která informace bude odstraněna ze stavu buňky. Tento proces probíhá aplikací sigmoid vrstvy na vstupní data spolu s předchozím výstupem LSTM sítě a následným bodovým násobením se stavem buňky. Za tento úkol odpovídá zapomínací brána.

V dalším kroku se rozhodne, jaká informace se má přidat do stavu buňky. Tento proces řídí vstupní brána a probíhá tak, že se nejprve pomocí sigmoid vrstvy rozhodne, které hodnoty budou aktualizovány. Poté vrstva hyperbolického tangensu vytvoří vektor kandidátních hodnot, které mohou být přidány do stavu buňky. Výstupy z těchto dvou vrstev jsou následně vynásobeny a stav buňky je aktualizován. V posledním kroku se vypočítají výstupní hodnoty. Za tento proces odpovídá výstupní brána. Během tohoto procesu se použije sigmoid vrstva na vstupní data, která určí, které části vstupu budou poslány na výstup. Následně se výstup sigmoid funkce vynásobí se stavem buňky, který je převeden na hodnoty v rozmezí $(-1, 1)$ pomocí vrstvy tanh a toto je výstup LSTM sítě[16][15].



Obrázek 2.7: Schéma LSTM sítě [15]

Kapitola 3

Návrh systému

3.1 Návrh simulačního modelu

Pro vybudování adaptivního řídícího systému je klíčové mít přístup k datům. Tyto data jsou nezbytná pro ověření schopnosti navrženého systému učit se a adaptovat.

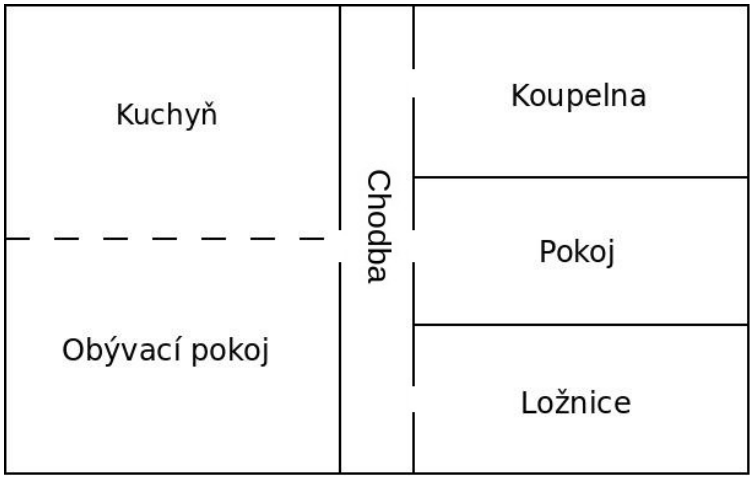
Z tohoto důvodu byl vytvořen simulační model domu, obyvatelé, jenž se v něm pohybují a simulované sensory.

3.1.1 Simulované prostory a obyvatelé

Domov se skládá z 6 místností, přičemž jde o ložnici, kterou sdílejí dvě osoby Z a M. Pokoj ve kterém pobývá osoba D a dále společné prostory – kuchyň, koupelna, chodba a obývací pokoj 3.1.

Osoby se pohybují v domě a provádí běžné činnosti. Například vaření, odpočinek, úklid a tak dále.

Každý z uživatelů má svoji preferovanou intenzitu a barevné spektrum nastaveného osvětlení, kterou po čas jeho činností nastavuje.



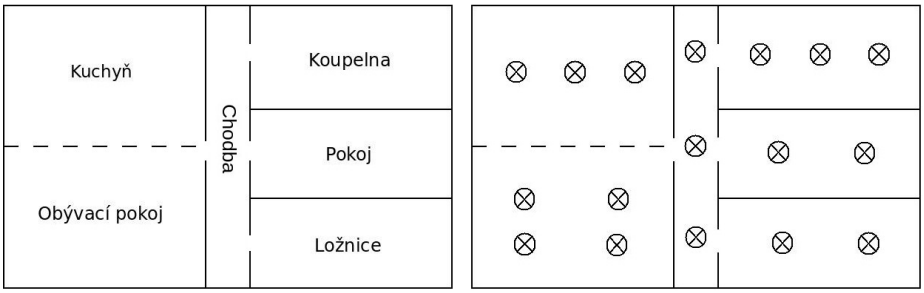
Obrázek 3.1: Půdorys simulovaného domu. Přerušovaná čára značí propojené místnosti. Vynechané spoje značí průchody.

3.1.2 Chytré osvětlení

Jako hlavní zdroj světla jsem zvolil chytré žárovky A19 RGBWW Smart Bulb[1], kterým může uživatel měnit barevné spektrum a intenzitu. V simulovaném domě se vyskytují stropní světla v tomto rozložení.

- Po trojicích se světla nachází v kuchyni, na chodbě a v koupelně.
- Po párech se nachází v pokoji a ložnici.
- A čteřice světel je v obývacím pokoji.

Rozmístění světel po místnostech je na obrázku 3.2 níže.



Obrázek 3.2: Rozmístění světel v domě.

3.1.3 Sensory

Bluetooth sensory

V přostředku každé z místností se nachází Bluetooth sensor, který snímá vzdálenost od mobilního zařízení uživatele. Jako konkrétní varianta byla zvolena vývojová deska ESP32[2] na kterou lze nahrát firmware ESPresence[3]. Sensor nám také podporuje snímání z chytrých hodinek a dalších zařízení.

Data z těchto sensorů nám určí, zda je uživatel přítomen v místnosti, případně jakou aktivitu vykonává. Například při průchodu místností se vzdálenost rychle mění a naopak při odpočinku je spíše konstantní.

Sensor intenzity denního světla

Uživatelé nastavují intenzitu osvětlení v závislosti na venkovní intenzitě světla. Pokud je v místosti dostatek světla, není potřeba kompenzace skrze umělé osvětlení.

Z tohoto důvodu je zde přidán jeden sensor pro snímání přirozeného světla. Konkrétně jsem zvolil variantu Xiaomi GZCGQ01LM[4].

Tento sensor je v modelu umístěn za oknem obývacího pokoje.

[1] https://kaufha.com/blf10/

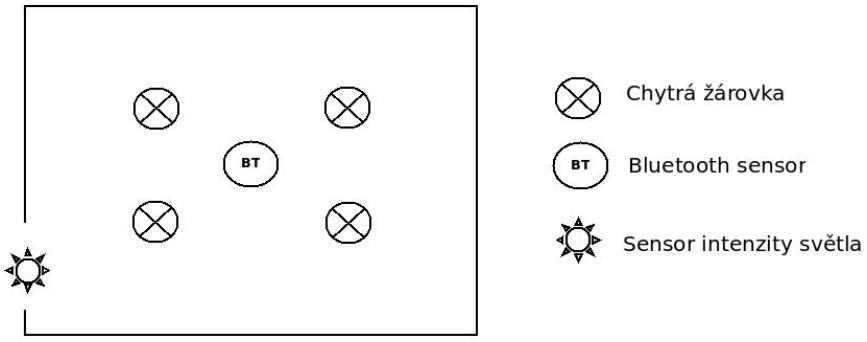
[2] https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/getting_started.html

[3] https://espresense.com/

[4] https://www.zigbee2mqtt.io/devices/GZCGQ01LM.html

Ovládací prvky

Jako ovládací prvky jsem zvolil mobilní zařízení jednotlivých uživatelů. Pokud bude uživatel požadovat rozsvícení konktretního světla, musí v mobilním zařízení vybrat světlo a nastavit požadovanou intenzitu. Jako rozšíření do budoucna je zde možnost přidat hlasové asistenty, pro zvýšení ergonomie.



Obrázek 3.3: Rozmístění sensorů a světel v místnosti.

3.2 Návrh řídícího systému

Řídící systém je navržen tak, aby se postupným sledováním běžných aktivit uživatelů naučil jejich manipulaci s osvětlením.

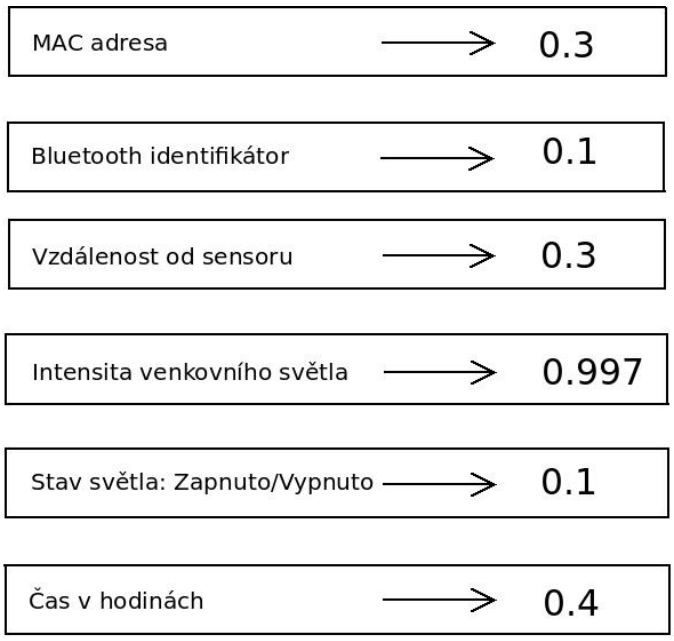
Jádrem řídícího systému je rekurentní neuronová síť LSTM 2.5.5, která se postupně učí na senzorických datech z chytré domácnosti. V prvotním návrhu bylo použít multiagentní systém pro řízení osvětlení. Nicméně po hlubším zkoumání problematiky jsem dospěl k závěru, že není nezbytně nutné využívat multiagentní paradigma, a proto jsem zvolil formu řízení prostřednictvím neuronových sítí. Toto rozhodnutí jsem učinil převážně pro svůj osobní zájem o tuto formu strojového učení.

3.2.1 První varianta návrhu

Na vstup řídícího systému postupně přicházejí MQTT zprávy ze sensorů v domě. Systém tyto data agreguje každou sekundu (v reálném nasazení by bylo vhodnější tento časový interval zkrátit pro rychlejší odezvu - například na polovinu sekundy nebo čtvrtinu, ale pro jednodušší implementaci simulačního modelu byl zvolen tento interval). Následně jsou tato data zakódována do několika rozměrného vektoru čísel a poslána na vstup LSTM sítě.
Data, ze kterých se systém učí vyhodnocovat vypadají následovně:

- Kdo se místosti aktuálně nachází - mac adresa zařízení zachyceného pomocí Bluetooth sensoru
- Identifikátor Bluetooth sensoru - o kterou místnost se jedná
- Vzdálenost od Bluetooth sensoru
- Intenzita venkovního osvětlení
- Aktuální stav světla - vypnuto / zapnuto

- Hodina ve které MQTT zpráva přišla - 0 až 23

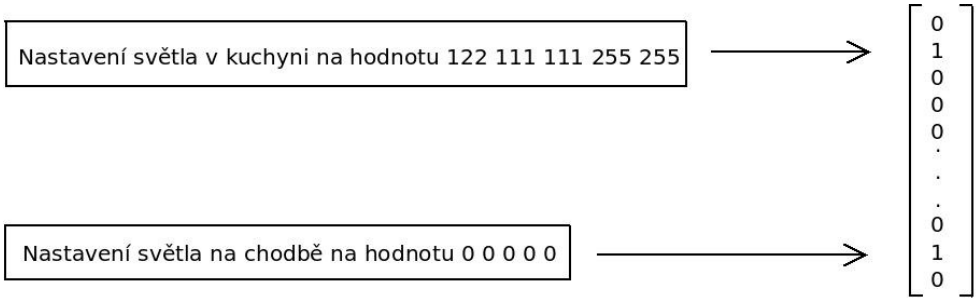


Obrázek 3.4: Zakódování sensorických dat do vstupního vektoru.

Všechny hodnoty jsou převedy na interval $[0, 1)$, pro zajištění lepší stability při trénovaní 3.4. V případě, že se v místnosti nachází více uživatelů, se hodnoty položek vektoru „kdo se v místnosti nachází“ a „vzdálenost od senzoru“ sčítají. Tento způsob reprezentace zajistí, že více lidí v místnosti se tváří jako jedna osoba, která může mít své vlastní preference. Vzdálenost od Bluetooth senzoru slouží zejména k odlišení aktivity prováděné uživatelem – například odpočinek v obývací místnosti, úklid domu a podobně.

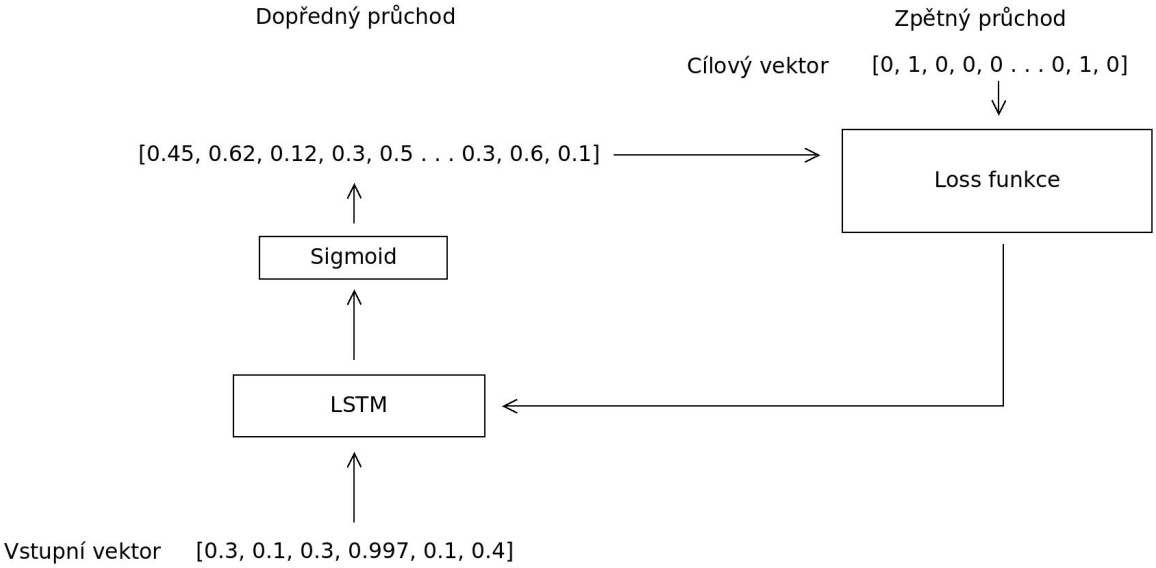
Cílová data pro trénování jsou získávána z MQTT zpráv, které uživatelé odesílají na aktuátory v případě, že s osvětlením manipulovali během daného časového intervalu.
Jako výstupy systému je zde možné využít více variant.
První varianta, se kterou jsem experimentoval byla využít výstup ze sítě jako analogové hodnoty. Přičemž síť by se pokaždé učila konkrétní požadovanou hodnotu, která jí byla předána na vstupu. V tomto případě by síť měla 6 vstupů a 17 výstupů – jeden výstup pro každé světlo. Nicméně po iniciálním experimentování jsem tento přístup opustil, neboť nevykazoval dostatečně přesvědčivé výsledky. Systém měl výrazné obtíže s naučením se specifických hodnot intenzity osvětlení, což zpochybňovalo jeho praktickou použitelnost.

Druhou variantu, kterou jsem zvolil jako o dost úspěšnější, bylo využít síť jako klasifikátor stavů, kde stav představuje požadovanou intenzitu uživatele a prvním základním stavem je nevysílat žádnou zprávu. Na výstupu sítě se poté nachází aktivační funkce sigmoid 2.5.2 namísto lineární , která převede analogické hodnoty na míru jistoty stavu pro každé světlo. A z tohoto vektoru poté stačí vybrat výstup s nejvyšší hodnotou pro dané světlo.



Obrázek 3.5: Zakódování dat z ovládacích prvků do cílového vektoru.

V momentě, kdy uživatel odešle MQTT zprávu na osvětlení 3.5, systém se podívá do tabulky známých stavů. Pokud se intenzita nachází v tabulce jako cílová data pro vyhodnocení, použije se index stavu v tabulce (nastavení hodnoty 1 na indexu stavu v cílovém vektoru). Pokud se intenzita v tabulce nenachází, systém ji do tabulky přidá a opět pošle index jako cílová data pro neuronovou síť 3.6.



Obrázek 3.6: Zaslání vstupního a cílového vektoru do LSTM sítě.

V tomto případě ale dochází k limitaci počtu možných zapamatovatelných stavů. Původně zamýšleno bylo zvolit počet možných zapamatovatelných stavů na 100 pro každé světlo. Experimentováním jsem ale dospěl k závěru, že při tomto množství výstupů, konkrétně, 1700 výstupů pro celou LSTM síť, je proces učení příliž pomalý na to, aby byl v praxi použitelný.

Počet stavů pro každé světlo, jsem tedy zvolil na 10, kvůli průběžným příznivým výsledkům. Ovšem i v tomto případě jsem narazil na problém, přičemž výstupy stavů světel se příliž ovlivňovaly. LSTM síť se tak často učila být v stavu, kdy by neměla vytvářet žádný výstup, což komplikovalo proces učení a snižovalo efektivitu modelu.

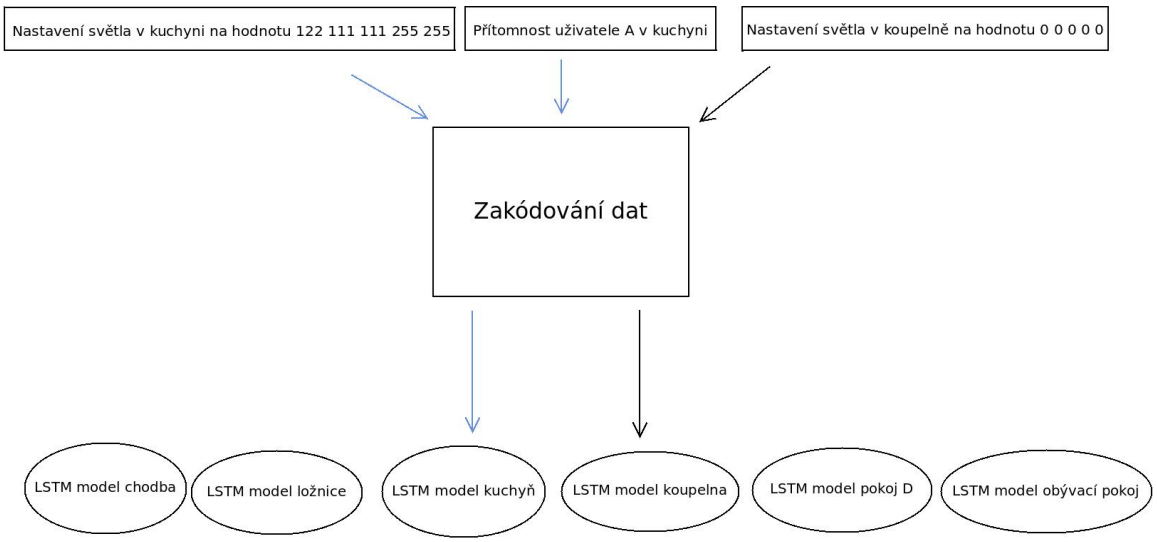
Jinými slovy, pokud uživatel ovládal pouze jedno světlo, síť se učila, že by všechna ostatní světla měla zůstat v danou chvíli beze změny stavu. Tento stav měl za následek

situaci, kdy síť, pokud bylo nutné skutečné rozsvícení málo používáného světla, nesprávně předpokládala, že by světlo mělo zůstat beze změny stavu.

3.2.2 Druhá finální varianta návrhu

Jako další krok při návrhu architektury jsem se rozhodl implementovat LSTM síť pro každou místnost zvlášť, aby se minimalizovalo vzájemné ovlivňování jednotlivých světel. V této finální architektuře je řídící systém složen ze šesti samostatných LSTM sítí. Tento přístup vykázal nejlepší výsledky během experimentů.

V případě že přijdou nějaká data od Bluetooth sensorů, informace budou poslána na model místnosti kde se sensor nachází. Obdobným způsobem je tomu tak u světel 3.7.

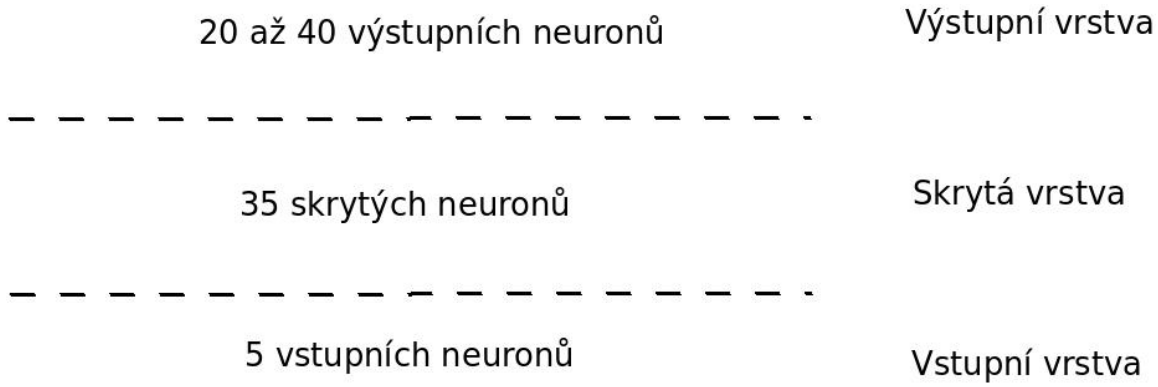


Obrázek 3.7: Architektura řídícího systému.

Všechny LSTM modely mají jednu skrytou vrstvu která čítá 35 skrytých neuronů. K tomuto počtu sem došel experimentováním se systémem.

V případě, že LSTM síť měla příliš malý počet skrytých neuronů, tedy 10 až 20, nedocházelo k naučení všech potřebných závislostí v datech. Pokud síť měla naopak příliš velký počet skrytých neuronů, docházelo ke ztrátě informací a pomalému učení – k tomuto problému docházelo při užití více nežli 60 skrytých neuronů.

Jelikož jsou sítě rozděleny podle místností, není potřeba zasílat na vstup informaci o tom, kde se uživatel nachází. Počet vstupů modelů je tedy vždy stejný – 5 vstupů.



Obrázek 3.8: Architektura LSTM modelů.

Počet výstupů se liší pro každou LSTM síť a závisí na počtu světel v každé místnosti a množství stavů daného světla. Jako optimalizační algoritmus byl vybrán Adam. Jako Loss funkce pro minimalizaci chyby byla zvolena cross-entropy.

LSTM model	**Počet výstupů**
Pokoj uživatele D	20
Chodba	30
Kuchyně	30
Obývací pokoj	40
Koupelna	30
Ložnice	20

Tabulka 3.1: Počet výstupů pro jednotlivé LSTM modely

3.2.3 Formát MQTT zpráv od sensorů

Každý sensor komunikuje pomocí MQTT zpráv a dle nalezených informací, mohou zprávy vypadat následovně.

Bluetooth sensory mají takovýto formát[5]:

```
{
  "id": "md:00e0:8",
  "rssi@1m": -59,
  "rssi": -81,
  "mac": "6eeeee56a23d",
  "raw": 4.25,
  "distance": 4.39,
  "speed": 0,
}
```

Z těchto dat je možnost získat místnost, ve které se Bluetooth senzor nachází, konkrétně z položky ID. Tento údaj poslouží k určení, na vstup kterého LSTM modelu mají být data zaslána. Nadále jde zde k dispozici mac adresa zařízení, které bylo v dosahu sensoru. Ostatní

[5] https://espresense.com/

hodnoty mohou být v dalších fázích vývoje také užitečné, ale v simulačním modelu jsem je pro jednoduší implementaci zanedbal.

Vybrané chytré žárovky posílají data v tomto formátu[6]:

```
{
  "NAME": "Smart bulb kitchen",
  "GPIO": [0,0,0,0,416,419,0,0,417,420,418,0,0,0],
  "FLAG": 0,
  "BASE": 18,
  "CMND": "SO105 1|RGBWWTable 204,204,122,153,153"
}
```

Z MQTT zpráv zasílané na světla je možné získat název světla a nastavenou intenzitu.

Snímače venkvoní intenzity posílají zprávy v následujícím formátu[7]:

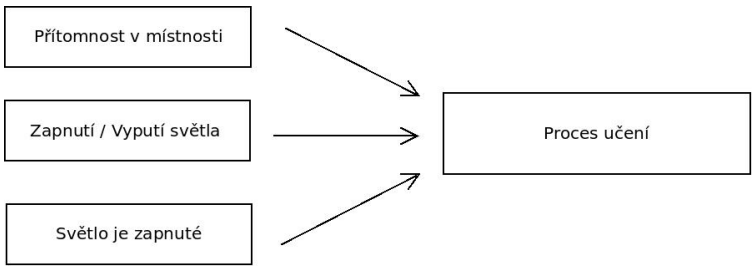
```
{
  "illuminance_lux": "300"
}
```

Z této zprávy je možné získat venkovní intenzitu osvětlení. Toto pomáha modelu odlišit pokud je například den či noc.

Důležitou poznámkou je, že v simulačním modelu je do všech zpráv přidáno navíc časové razítko. Je to pouze z důvodu synchronizace simulačního modelu a řídícího systému. V reálném nasazení by řídící systém tento údaj získával sám ze skutečného času.

3.3 Průběh učení a adaptace

Řídící systém začíná učit se v okamžiku, kdy obdrží sensorická data z Bluetooth sensorů nebo od ovládacích prvků uživatelů v době, kdy manipulují s osvětlením. V prvním návrhu systém zpracovával a učil se ze všech sensorických dat. Pokud uživatelé nebyli doma, systém se učil z času a intenzity venkovního osvětlení a neposkytoval žádné změny stavů osvětlení na výstupu. Tento přístup se však ukázal jako chybný, neboť opakovaně vedl k závěrům, že se model neuronové sítě naučil s nejvyšší jistotou neposkytovat žádné změny stavů, a to i v případě, že bylo požadováno rozsvícení světla. Z tohoto důvodu jsem navrhl princip učení tak, že systém začne učit se pouze v případě, že přijme MQTT zprávu s nastavením světla na určitou intenzitu, nebo pokud dostává data z Bluetooth sensorů 3.9.



Obrázek 3.9: Situace při kterých se systém učí

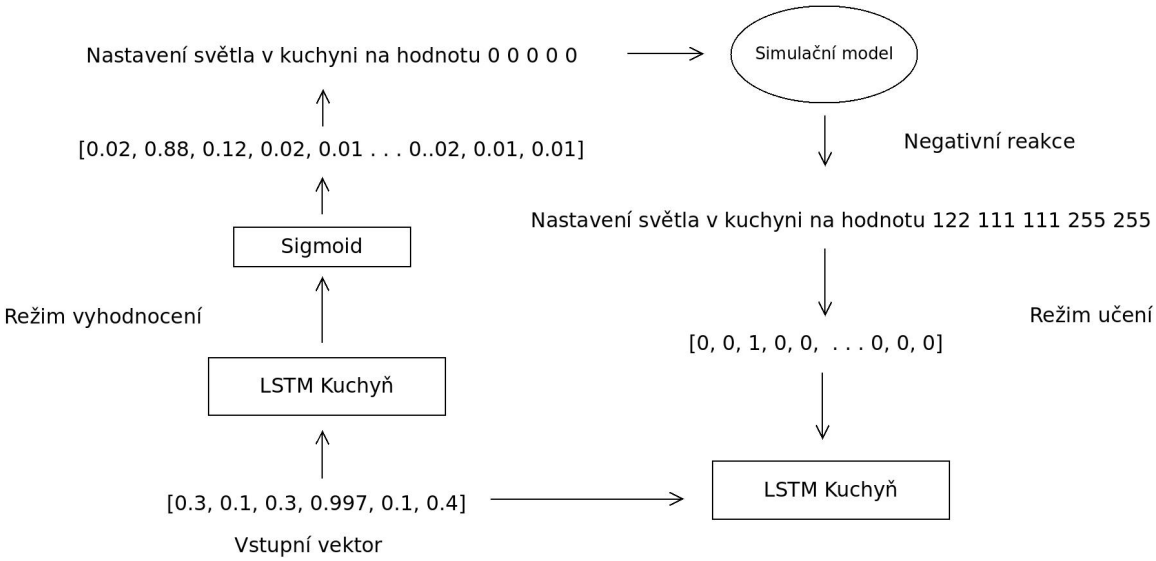
[6] https://kaufha.com/blf10/

[7] https://www.zigbee2mqtt.io/devices/GZCGQ01LM.html

Dále se také učí, dokud je nějaké světlo rozsvícené. Z této implementace jsem získal nejlepší výsledky v procesu učení. Rychlost učení závisí také na frekvenci manipulace s osvětlením. Lze říci, že při nízké intenzitě venkovního světla, nebo v zimním období se systém bude učit rychleji.

Po uplynutí určeného času se řídící systém přepíná do režimu evaluace a začíná řídit osvětlení samostatně. V této fázi by systém měl již znát základní informace o manipulaci s osvětlením a pouze se učí ze svých chyb. Tímto způsobem data z ostatních senzorů slouží pouze k vyhodnocení. Jakmile systém nenastaví požadovanou intenzitu, a uživatel na to zareaguje svým nastavením, systém se na tuto změnu adaptuje.

V případě příchozí MQTT zprávy o nastavení světla v režimu evaluace systém vyhodnotí ostatní příchozí zprávy a následuje režim doučování. V režimu doučování se sestaví cílový vektor z přijaté MQTT zprávy. Vstupní data nejsou přidána z aktuálního časového úseku; místo toho se použije předchozí vektor, který byl použit pro vyhodnocení, a společně s cílovým vektorem jsou poslány na vstup LSTM sítě. Ta se přepne zpět do režimu trénování a přeučí se na základě vzniklé chyby 3.10.



Obrázek 3.10: Zaslání vstupního a cílového vektoru do LSTM sítě v rámci režimu adaptace

Kapitola 4

Implementace

4.1 Implementace simulačního modelu

Jádro simulačního modelu se skládá z Petriho sítí[1] pro pohyb osob v domácnosti. Každý uživatel má definované určité aktivity, které během dne vykonává. Petriho sítě jsou propojeny s atomickými bloky, které reprezentují simulované senzory. Zde se nachází blok pro Bluetooth sensor, snímač intenzity venkovního osvětlení a aktuátor. V případě, že z Petriho sítí obdržíme informaci o přítomnosti osoby v místnosti, spustí se simulace činností uživatele – pohyb, změna intenzity světel. Následně, jakmile uživatel opustí místnost, činnost senzorů se ukončí.

Bloky pro manipulaci uživatelů se světly mají také na svém vstupu přivedeny nastavenou intenzitu od řídícího systému a intenzitu venkovního osvětlení poskytnutou od snímače.

Uživatelé se v systému také rozhodují, jakou akci udělat v případě, že systém nastaví určitou intenzitu. Dále uživatelé nastavují intenzity podle intenzity venkovního osvětlení. Například, pokud je brzo ráno, uživatel nastaví jinou intenzitu než přes noc.

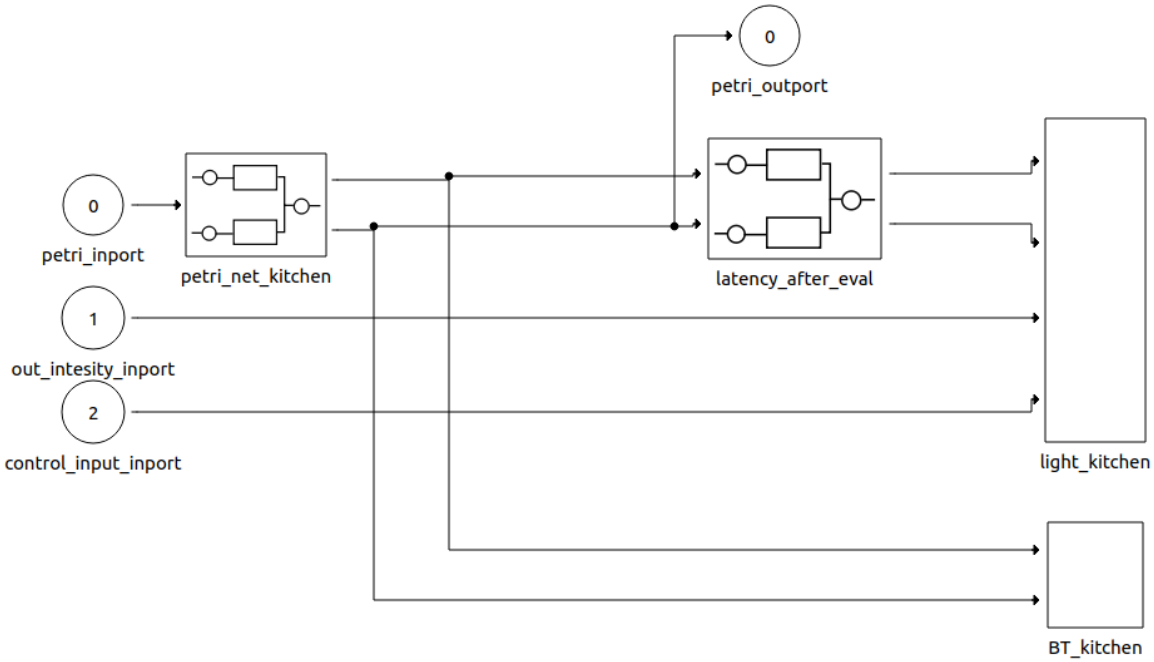
Každý z uživatelů má také nastavenou svoji preferovanou intenzitu, kterou nastavuje při příchodu do místnosti. V případě, že je více uživatelů v místnosti, osoby navzájem nemění intenzitu světel. Uživatelé si také navzájem nevypínají světla. Pokud se stane, že někdo z místnosti odchází, ale v místnosti zůstává ještě nějaká osoba, světlo se nevypíná.

Pro implementaci řídícího systému bylo nutné použít také globální proměnné, a to z důvodu synchronizace mezi procesy a pravděpodobně také kvůli dosažení limitu systému powerDEVS při velkém počtu spojů. Pokud bylo v simulaci příliš mnoho spojů, systém přestával správně fungovat a nepodařilo se mi spustit simulaci. Jelikož můj návrh vyžadoval velké množství těchto spojení, bylo nutné v určitých případech použít globální proměnné.

Ty jsou využity při komunikaci s řídícím systémem. Veškeré bloky sensorů posílají data do globálního pole znaků a blok pro komunikaci s řídícím systémem je ve fázi výstupu odesílá na řídící systém. Další globální proměnné byly využity v implementaci místnosti, kde tak slouží k propojení více atomických bloků místností pro ověření, zda se v nich někdo nachází.

Na obrázku 4.1 je vidět implementace konkrétní aktivity v kuchyni. Ze složeného bloku `petri_net_kitchen`, ve kterém jsou Petriho sítě, vedou dva výstupy, které mají reprezentovat dobu, po kterou je uživatel v místnosti a koná svou aktivitu. První výstup reprezentuje vstup do místnosti, a pokud v rámci tohoto výstupu přijde informace do atomických bloků `light_kitchen` a `BT_kitchen`, spustí se proces simulovaného bluetooth sensoru. Současně,

[1] Petriho sítě jsou nástrojem používaným pro modelování a analýzu distribuovaných systémů



Obrázek 4.1: Model aktivity v systému powerDEVS

v případě kladných podmínek, se zajistí zaslání MQTT zprávy v bloku `light_kitchen`. Jakmile přijde hodnota z bloku `petri_net_kitchen` z druhého výstupu, uživatel odchází, a blok `BT_kitchen` přestane simulovat aktivitu uživatele v místnosti. Pokud došlo k rozsvícení světel, v bloku `light_kitchen` se vyšle MQTT zpráva o vypnutí. Dále jsou zde tři vstupy složeného bloku. Vstup číslo 0 je informace z Petriho sítě o příchodu do místnosti, na který navazuje jediný výstup na obrázku `petri_outport`. Jakmile uživatel dokončí svou činnost, může pokračovat do jiné místnosti. Vstup číslo 1 je intenzita venkovního osvětlení, podle kterého uživatelé mohou měnit preferovanou intenzitu v reakci na různé úrovně přirozeného osvětlení.

Poslední vstup, číslo 2, je vstup řídícího systému, který může v režimu vyhodnocování v místnosti již nastavit určitou intenzitu, na kterou uživatel může reagovat. Poslední blok na obrázku je složený blok `latency_after_eval`, který přidává systému zpoždění jednoho kroku v případě, že se systém přepne do režimu evaluace. Tento blok byl implementován pomocí upraveného bloku Petriho přechodu, který v závislosti na režimu řídícího systému může zpozdit manipulaci s osvětlením. Tato implementace byla provedena za účelem zajistit čas, který bude využit pro nastavení osvětlení za uživatele. Celý popsaný složený blok na obrázku 4.1 existuje pro každou místnost.

4.2 Implementace řídícího systému

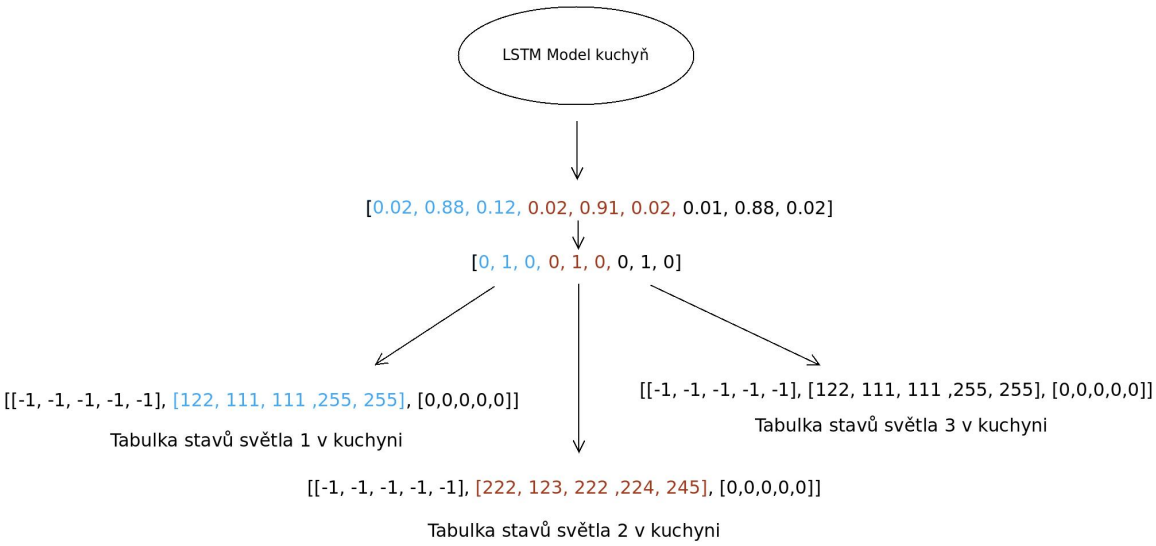
Řídící systém byl implementován v jazyce Python za použití knihovny pytorch[2], která poskytuje nástroje a techniky pro vytváření a trénování neuronových sítí. Jak je popsáno v sekci 4.3, simulační model a řídící systém jsou propojeny pomocí pojmenovaných rour. V momentě, kdy se ze simulace vygenerují data v jednom časovém kroku, jsou odeslána na řídící systém.

Systém nejdříve provede zpracování dat, v rámci něhož jsou data převedena z textové podoby do formátu JSON[3]. Následně se provede transformace dat do zmíněných vektorů a podle toho, zda data obsahují informaci o nastavení světla na určitou hodnotu nebo informaci z Bluetooth sensorů, bude zahájen proces trénování sítě.

Jako cílový vektor je zde použita informace o tom, jaká světla byla nastavena na některou z intenzit. Tyto data se převedou do již zmíněné tabulky stavů pro každé světlo. V případě, že se intenzita nenachází v tabulce, je do ní přidána. Po nastaveném čase se systém přepíná do režimu evaluace, při kterém do simulačního modelu zasílá MQTT zprávy s nastavenými hodnoty intenzit.

Na obrázku 4.2 je popsán průběh zpracování výstupu od LSTM modelu. Model nám vrací vektor míry jistoty daného stavu pro každé světlo. Na obrázku 4.2 je zobrazena zjednodušená varianta, při které systém vybírá pouze ze tří stavů pro každé světlo. V reálné implementaci je deset stavů pro každé světlo. Model má na obrázku celkem 9 výstupů, jelikož v této místnosti máme celkem 3 světla, což znamená 3 stavy pro každé světlo. Z tohoto vektoru je následně vybrán index stavu s nejvyšší hodnotou pro každé 3 po sobě jdoucí položky vektoru. Následně se index použije jako index do tabulky, kde jsou uchovány stavy jednotlivých světel, systém zvolený stav vezme a vytvoří MQTT zprávu s danou intenzitou. Jakmile jsou sestaveny všechny MQTT zprávy, jsou zaslány do simulačního modelu. Vektor s hodnotami nul a jedniček je zde chápán jako vybraný stav světla, označen hodnotou „1“.

Při spuštění programu je také možnost zadat parametr „-e číslo“ při jehož zadaní specifikujeme dobu, po které se má systém přepnout do režimu vyhodnocování.



Obrázek 4.2: Zpracování výstupu od LSTM modelu

[2] https://pytorch.org/
[3] JavaScript Object Notation (JSON) Je způsob zápisu dat určený pro přenos

4.3 Komunikace řídícího systému a simulačního modelu

Simulační model a řídící systém jsou propojeny pomocí pojmenovaných rour(anglicky named pipes)[21]. V inicializační fázi při spuštění si řídící systém a simulační model vymění informace, kdy se přejde do fáze adaptace, tj. kdy začne řídící systém vysílat MQTT zprávy za simulované osoby.

Po této inicializační fázi začnou simulované osoby provádět své činnosti v domě. Po uplynutí jednoho časového úseku se v bloku, který zajišťuje komunikaci se řídícím systémem, odešlou všechna nasbíraná data.

Řídící systém postupně zpracovává tato data a odesílá zpět do simulačního modelu MQTT zprávy. V případě, že se systém stále učí a nedává žádné zprávy na výstup, v MQTT zprávě jsou odeslány hodnoty „-1“, což má reprezentovat nenastavení žádné hodnoty. Tato komunikace slouží pouze k zajištění synchronizace mezi oběma procesy.

Jakmile je systém připraven začít řídit osvětlení, v simulačním modelu je implementován upravený blok přechodu Petriho sítí, který začne po určitém časovém úseku vnášet do chování uživatelů zpoždění jednoho simulačního kroku. Toto zpoždění se týká pouze bloku, ve kterém uživatelé manipulují s osvětlením. Tímto způsobem je zajištěn čas pro řídící systém, aby mohl případně vykonat akci, například nastavení určité intenzity nebo vypnutí světla. V případě, že systém nastaví správnou intenzitu, simulovaný uživatel na to nijak nereaguje.

Pokud systém nastaví nesprávnou intenzitu, simulovaná osoba na to zareaguje zpětným přenastavením na intenzitu, kterou původně požadovala. Tuto informaci řídící systém také zpracuje a snaží se ji adaptovat.

Jakmile uplyne nastavený čas, simulační model ukončí komunikaci zprávou o doběhnutí časového limitu a systémy přestanou komunikovat a ukončí se.

Kapitola 5

Experimenty a průběh učení

Tato část práce se zaměřuje na experimentaci s navrženým systémem. Během experimentů bude sledováno, jak rychle probíhá fáze trénování a adaptace modelů při různých situacích v simulačním modelu.

Dále se v experimentech bude měřit tzv. metrika uživatelského komfortu. Tuto metriku jsem definoval jako situaci, kdy se nějaká aktivita provádí za uživatele a tím pádem ji nemusí uživatel provádět sám. Aby bylo možné tuto metriku měřit, definoval jsem ji jako poměr počtu v procentech všech nastavených intenzit ať už od systému nebo od uživatele a správně nastavených intenzit.

Všechny grafy komfortu jsou mírně posunuty kvůli nutnosti nasbírat dostatečný počet dat pro vizualizaci.Pro urychlení doby běhu simulace uživatelé manipulují s osvětlením průměrně jednou za deset sekund.

$Komfort$ =počet správně nastavených intenzit / počet všech nastavených intenzit

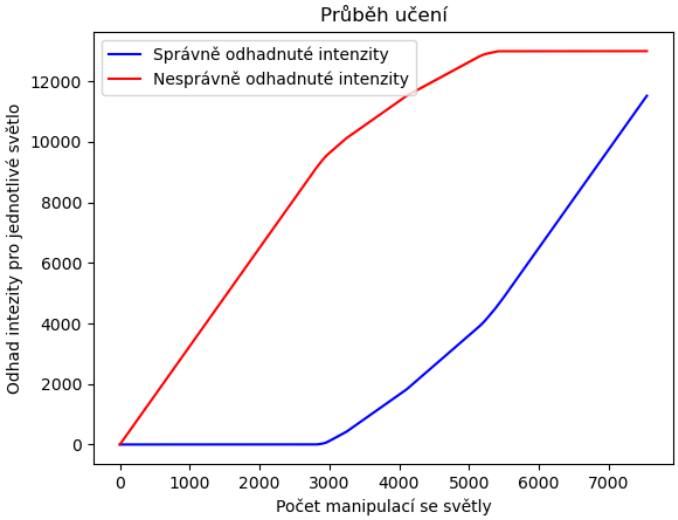
5.1 Experiment 1

V tomto experimentu je zkoumán průběh učení se systému při běžných činnostech v domě osoby M. Osoba M vykonává aktivity celkem ve 4 místnostech: kuchyni, obývacím pokoji, chodbě a koupelně.

Osoba má pouze jednu preferovanou intenzitu, kterou během svých aktivit nastavuje a při konci své činnosti světla vypíná. Na grafu 5.1 níže lze vidět průběh učení se systému jakou hodnotu dát na výstup. Graf je vytvořen dvěma křivkami, které znázorňují počet správně a špatně odhadnutých stavů pro každé světlo. Do grafu není zahrnut stav, ve kterém systém nemá na výstupu nastavovat žádnou hodnotu.

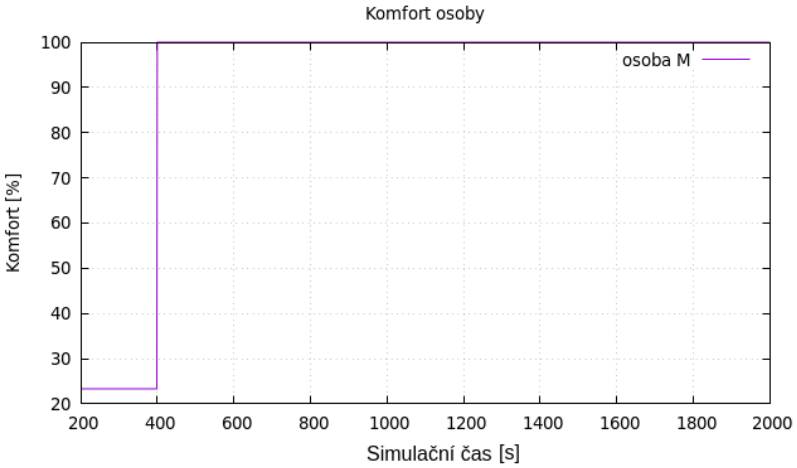
Jak lze z grafu vyčíst, systém začal správně odhadovat intenzity po zhruba 3000 manipulacích. Manipulací se světlem je myšleno nastavení určité intenzity nebo vypnutí světla.

Přepočítání na dny se hůře odhaduje, jelikož zde hraje podstatnou roli, jak často jsou lidé v domácnosti, jaká je aktuální intenzita přirozeného osvětlení, kolik lidí je v domácnosti a další faktory. V případě, že bych vztáhl tento proces na osobní domácnost, domnívám se, že by trval kolem dvou měsíců v zimním období.



Obrázek 5.1: Průběh učení se systému v případě jedné osoby v domácnosti a aktivitách ve 4 místnostech

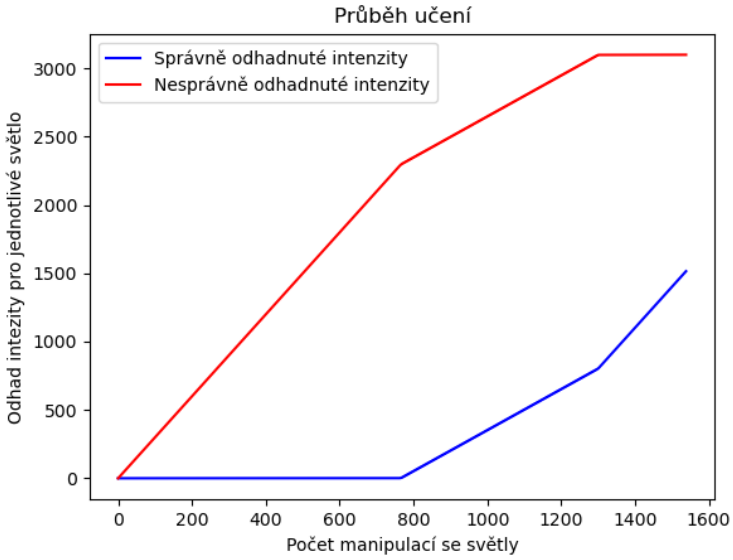
Na druhém grafu 5.2 níže lze vidět průběh komfortu uživatele po aplikaci takto natrénovaného modelu. Z dat je patrné, že komfort osoby je vysoký a systém dokáže velmi dobře manipulovat s osvětlením za uživatele. Osoba se cítí komfortně, až na menší počáteční nestabilitu.



Obrázek 5.2: Komfort osoby M

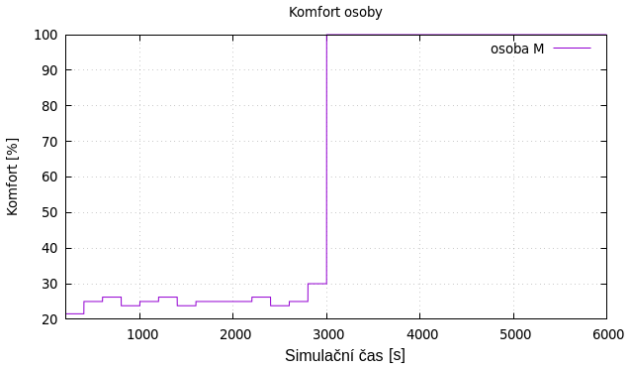
5.2 Experiment 2 - Adaptace novým změnám

U tohoto experimentu je zaměření kladeno na délku trvání adaptace systému na nové změny. V simulačním modelu je opět zahrnuta osoba M, která provádí činnosti v jedné místnosti, konkrétně v kuchyni, pro zjednodušení. Simulovaná osoba po přepnutí do režimu vyhodnocení změní svou preferovanou intenzitu a sleduje se rychlost adaptace systému na tuto změnu. Na prvním grafu 5.3 v této sekci je zobrazen proces učení se systémem manipulací s osvětlením.



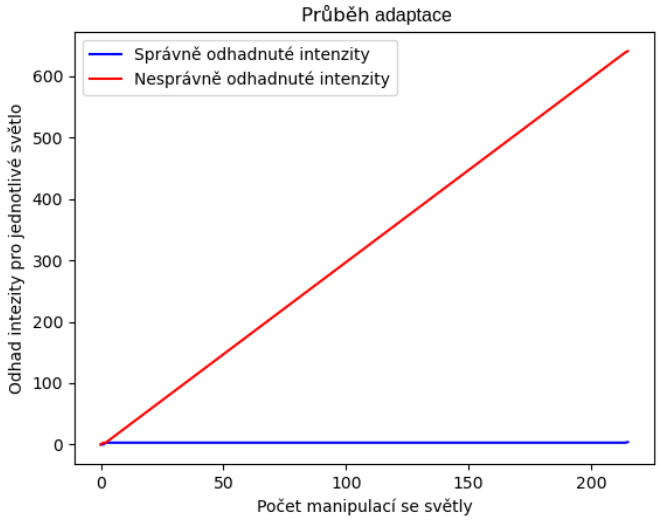
Obrázek 5.3: Průběh učení systému v situaci, kdy jedna osoba provádí aktivity v jedné místnosti s jedním preferovaným nastavením osvětlení.

V grafu 5.3 lze vidět, že systém se naučil odhadovat intenzitu po 800 manipulacích. Nyní v grafu 5.4 níže lze vidět komfort uživatele pokud změnil preferovanou intenzitu po natrénování modelu.



Obrázek 5.4: Komfort osoby M po změně preferované intenzity

V tomto 5.4 případě lze vidět, že míra komfortu uživatele má již určitou hranici, jelikož řídící systém je naučen vypínat světla v místnostech.

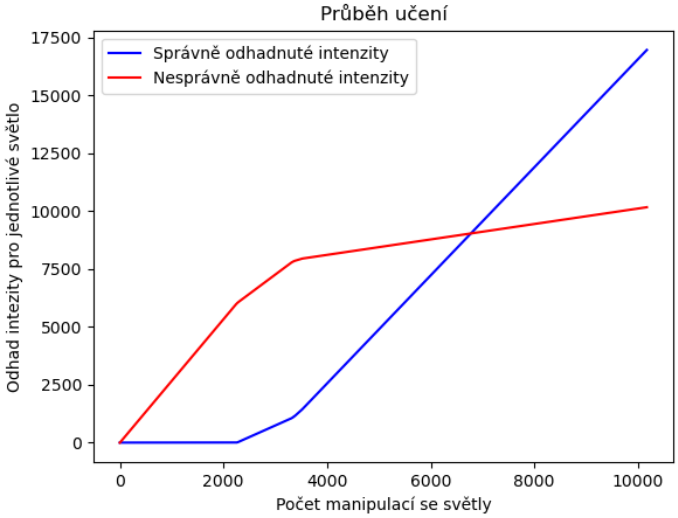


Obrázek 5.5: Průběh adaptace systému při změně intenzity

Na grafu 5.5 lze vidět, že se systém adaptoval na novou preferovanou intenzitu po 200 nových nastaveních intenzity. Pokud toto srovnáme s fází, kdy se systém čistě učil, je tento počet manipulací snížen zhruba na čtvrtinu původního času. Je zde patrné, že systém již má naučený určitý druh chování uživatele, ale stále je potřeba systém naučit nový stav osvětlení.Jsem přesvědčen že chytrou manipulací s daty, je možné tento čas adaptace zkrátit.

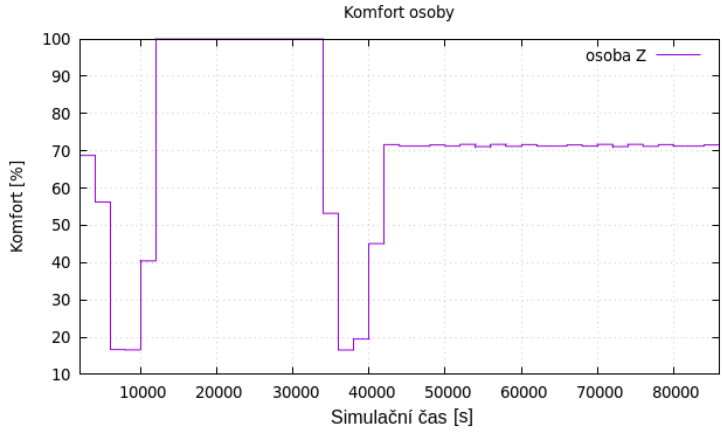
5.3 Experiment 3 - Různé intenzity počas dne

Tento experiment se opět zaměřuje na jedinou osobu v domácnosti, osobu Z. Osoba Z vykonává aktivity ve třech různých místnostech – v kuchyni, obývacím pokoji a ložnici. Uživatel tentokrát však nastavuje různé intenzity během dne v závislosti na venkovní intenzitě. Pokud je úroveň venkovní intenzity vysoká, uživatel s osvětlením vůbec nemanipuluje.



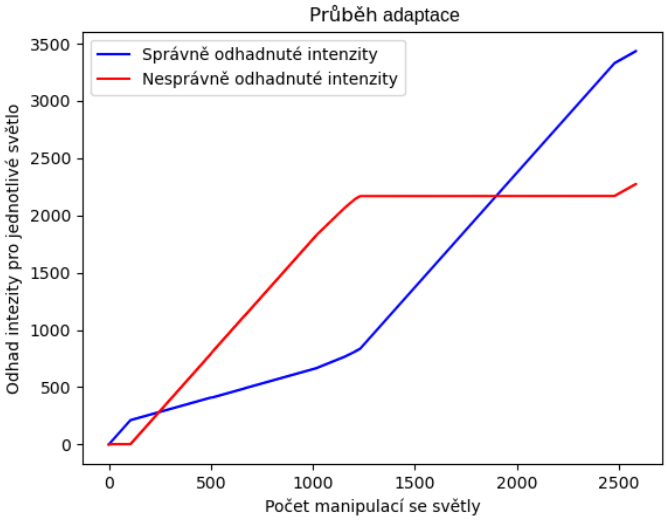
Obrázek 5.6: Průběh učení systému při více intenzitách během dne.

Z grafu 5.6 zde lze vidět, že pro systém není příliš obtížné zapamatovat si různé intenzity během dne. Jediné úseky, kdy si není jistý s danou hodnotou, jsou při přechodu z jedné intenzity na jinou. Tento interval trvá kolem 5000 simulačních kroků, poté je si systém jistý jakou hodnotu má zvolit. V případě že by tento interval byl přepočten na reálný čas vycházelo by to přibližně na 83 minut času. Toto vychází z předpokladu, že simulační čas je přepočten jako jedna sekunda času reálného – $5000/60 \approx 83$.



Obrázek 5.7: Komfort uživatele Z při různých intenzitách během dne

Z grafu komfortu 5.7 je patrné, že systém kolísá při přechodu z jedné intenzity na druhou, ale poté se ustálí. Nejvyšší komfort uživatele je zaznamenán v případě, kdy je dostatek přirozeného osvětlení, a tudíž manipulace se světlem není zcela nezbytná.

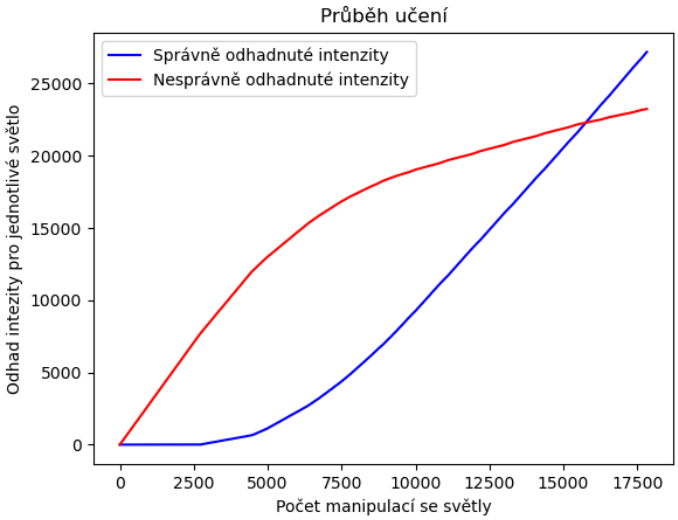


Obrázek 5.8: Průběh adaptace systému v případě nastavení více inenzit počas dne

V graf 5.8 lze vidět průběh adaptace systému v případě špatně nastavené intenzity. Jak již bylo zmíněno, systém se učí především v případě přechodu z jedné intenzity na druhou.

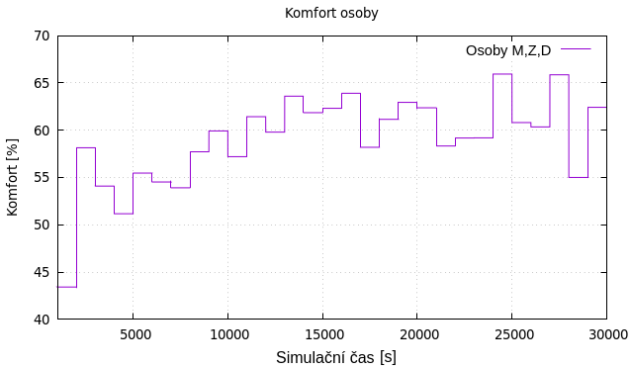
5.4 Experiment 4 - Více lidi v domácnosti

V tomto experimentu se v domě pohybují všichni obyvatelé domu. Konkrétně se jedná o 3 osoby - M, Z, D. Simulovaní obyvatelé přitom vykonávají aktivity ve všech místnostech, což vede k většímu počtu osob v jedné místnosti. Každá z osob má svou preferovanou intenzitu, kterou nastavuje při vstupu do místnosti. Pokud je nějaká intenzita již nastavená, což znamená, že už je někdo v místnosti, při příchodu dalšího obyvatele se intenzita již nemění a nově příchozí se s nastavenou intenzitou smíří. V situaci, kdy osoba opouští místnost, ale někdo zde stále vykonává aktivitu, nedochází k vypnutí osvětlení.



Obrázek 5.9: Průběh učení systému při větším počtu osob.

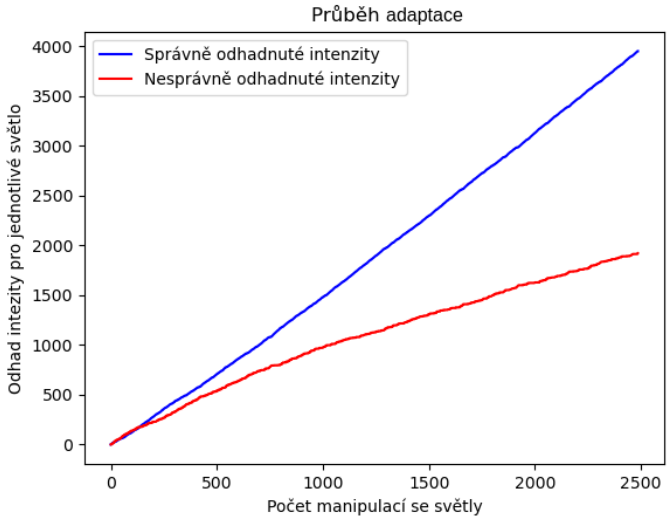
Na průběhu učení 5.9 lze v tomto případě vidět hladší průběh, pravděpodobně z důvodu jiné intenzity pro každého uživatele. Také je zde vidět, že systém začne odhadovat správné intenzity opět zhruba po 3000 manipulacích.



Obrázek 5.10: Komfort osob při větším počtu

Z grafu komfortu 5.10 lze usoudit, že se systému i v tomto případě daří nastavovat intenzity poměrně dobře. Problémy má nejspíše jen v případech, kdy do místnosti přijde naráz více lidí, poté není jistý, jakou intenzitu má nastavit. Další situace, která může snižovat

komfort, je ta, kdy jedna osoba odejde z místnosti a vypne světlo během sběru dat, a poté do místnosti přijde další osoba a znovu nastaví osvětlení, čímž systém ztrácí jasno v tom, jakou akci provést.

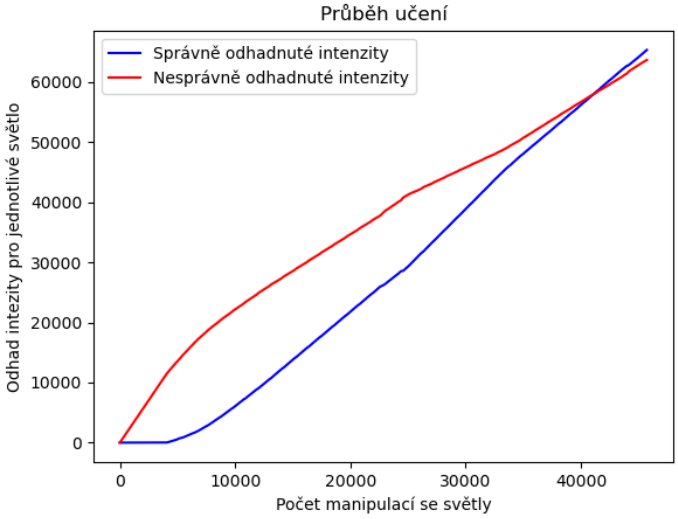


Obrázek 5.11: Průběh adaptace systému při větším počtu osob

Na grafu adaptace 5.11 lze vidět, že se systém stále učí ze svých chyb a snaží se minimalizovat nepohodlí uživatelů.

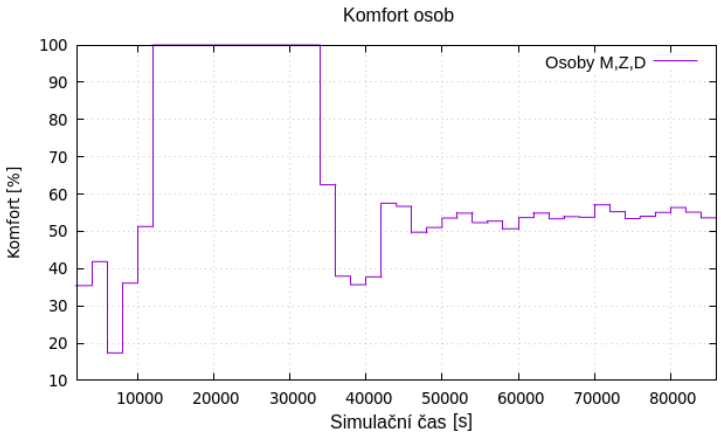
5.5 Experiment 5 - Více lidí v domácnosti při různých intenzitách během dne

V rámci tohoto experimentu se zde opět pohybují všichni účastníci domácnosti ve všech místnostech, tedy osoby M, Z a D. Nyní ale každá osoba reaguje na venkovní osvětlení a podle jeho úrovně se rozhoduje, jakou preferovanou intenzitu pro dané osvětlení zvolí. Stejně jako v předchozím experimentu 5.3 si osoby vzájemně nemění osvětlení ani ho nevypínají v případě, že odcházejí z místnosti a někdo se v ní stále nachází.



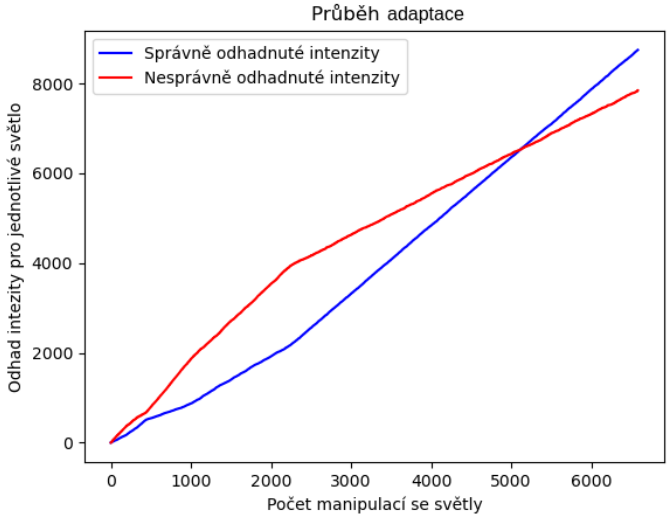
Obrázek 5.12: Průběh učení při větším počtu osob s různými intenzitami během dne.

Z průběhu učení 5.12 lze vidět, že se systém začíná učit hodnoty zhruba při stejném počtu manipulací jako při předešlých experimentech, avšak není už zdaleka tak přesný jako při méně naročném chování uživatelů. Doba odhadu trvání v realném čase by se zde dala odhadovat kolem 6 až 9 měsíců učení, než by systém dosáhl hranice komfortu jež je uvedena na grafu 5.13. Tento úsudek je odvozen na základě počtu manipulací na grafu 5.12.



Obrázek 5.13: Komfort osob při větším počtu s různými intenzitami během dne.

Z grafu komfortu lze vidět, že uživatelé jsou jistě nejspokojenější v případě, že je dostatek přirozeného osvětlení a není potřeba žádné umělé. Tento interval trvá zhruba od simulačního času 9000 do 40000. Hranice komfortu se dále drží kolem 55 procent, což je nejspíše způsobeno okolnostmi podobnými jako v případě experimentů 5.3 a 5.4.



Obrázek 5.14: Průběh adaptace systému při větším počtu osob s různými intenzitami během dne.

Z průběhu adaptace 5.14 vidíme, že systém v této fázi má již potíže nastavit pro všechny uživatele intenzitu, kterou v závislosti na okolních podmínkách požadují, ale stále se adaptuje a snaží se své odhady zlepšovat.

5.6 Přechod k reálnému nasazení

Při nasazení systému do reálného provozu by se musela vyřešit nejpodstatnější změna oproti simulaci, a to čas, po který se systém učí. Při experimentech bylo ověřeno, že se navržený systém dokáže učit a adaptovat manipulaci s osvětlením. Avšak k tomu je zapotřebí poměrně častá manipulace se světly. V experimentu 5.2 bylo zjištěno, že pro jednu místnost je potřeba kolem 1300 manipulací, aby systém dokázal dobře odhadovat výsledky. V každé domácnosti se pravděpodobně liší počet zapnutí a vypnutí světla, ale i v případě, že k takovýmto aktivitám dochází velmi často, by naučení modelu zabralo nejméně několik měsíců.

Na tyto problémy existuje řada řešení, která by mohla tento proces urychlit. Jedním z možností je nasbírat po nějakou dobu data z domácnosti, na těchto datech provést vhodnou formu augmentace a následně předtrénovat model před jeho nasazením.

Pro správnou volbu intenzit pro uživatele existuje také možnost předtrénovat systém na případy, kdy je požadováno nastavit určitou intenzitu. Avšak trénování by se mělo zaměřit na všechny stavy lišící se od nezaslání žádné zprávy a vypnutí světla. Tímto způsobem by systém měl přehled o tom, kdy nastavit určitou intenzitu, aniž by znal konkrétní hodnotu a dokázal by se poté preferované intenzitě rychle adaptovat.

Jako další varianta, jak zrychlit proces učení, je možné modelu zaslat některá data na vstup vícekrát. Například v situaci, kdy se manipuluje s osvětlením, může být daný vektor poslán na vstup sítě například 10krát. Tímto způsobem by bylo teoreticky možné násobně zrychlit proces učení.

Další varianta, kterou je nutné brát v potaz, jsou Bluetooth senzory, které pro zjednodušení simulačního modelu nejsou implementovány tak, aby signál jednoho zařízení zasahoval do více místnosti. Jako řešení se nabízí možnost měřit signály v jednotlivých místnostech domu a v případě nedostatečné síly signálu by systém tato data zanedbával

Navržený systém je také nutné začlenit do dohledového systému. Jako kompatibilní variantu pro navržené sensory a aktuátory bych zvolil systém Home Assistant[1]. Jako zařízení, na kterém by systém běžel, by bylo možné zvolit Raspberry Pi[1].

Řídící systém se, tak jak je nyní navržen, přepíná do režimu evaluace pomocí přepínače při spuštění. V případě reálného nasazení se lze vydat více směry. Jendím z nich je mít možnost uživatele nechat sledovat grafy správně a špatně odhadnutých intenzit a volbu, kdy přepnout systém do režimu vyhodnocení, nechat na uživateli. Další možností je nechat systém sledovat statistiky procesu učení a v momentě, kdy dojde k dostatečné úrovni správných odhadů, systém přejde do režimu vyhodnocení automaticky.

[1] https://www.raspberrypi.org/

Kapitola 6

Závěr a zhodnocení

V této práci se podařilo navrhnout a implementovat systém založený na rekurentních neuronových sítích LSTM. Navržený systém dokáže predikovat chování uživatelů v kontextu manipulací s osvětlením. Výsledky zveřejněné v experimentech dokazují, že se systém dokáže učit a adaptovat změnám. Ve výsledcích je nadále vidět, že se komfort uživatelů pohybuje v přijatelných rozmezích v závislosti na jejich chování. Z mého pohledu mohlo v práci dojít k lepším výsledkům za použití různých heuristik jako forma zaslání modelu některých zpráv vícekrát. Avšak velmi podstatnou částí práce bylo vynaloženo na implementaci simulačního modelu za použití nástroje PowerDEVS. Tento nástroj bych ohodnotil jako velmi robustní pro budování simulačních modelů, avšak ne zcela dokončený a velmi stroze zdokumentovaný.

V porovnání s existujícími přístupy k řízení osvětlení lze konstatovat, že navržený adaptivní systém se úspěšně přizpůsobuje uživatelským preferencím a zvyšuje tak komfort uživatelů během pobytu v domácnosti. Na rozdíl od tradičních metod řízení osvětlení, jako je například použití pouhých senzorů pohybu, je navržený systém schopen uchovávat a využívat různé uživatelské preference. Díky tomu lze tento systém charakterizovat jako personalizovaný bez nutnosti složité konfigurace, která je často potřebná u tradičních řídících systémů. Tímto způsobem se systém stává efektivním nástrojem pro zlepšení uživatelského zážitku a poskytuje uživatelům větší pohodlí a kontrolu nad osvětlením v jejich prostředí.

Literatura

[1] *Home Assistant* [online]. [cit. 2024-04-25]. Dostupné z: https://www.home-assistant.io/.

[2] *What is MQTT?* [online]. [cit. 2024-02-21]. Dostupné z: https://aws.amazon.com/what-is/mqtt/.

[3] *What is X10 Home Automation* [online]. [cit. 2024-02-21]. Dostupné z: https://www.x10.com/pages/allaboutx10.

[4] *Internet of Things* [online]. 2020 [cit. 2024-03-02]. Dostupné z: https://its.ucsc.edu/news/internet-of-things.html.

[5] *What is Zigbee?* [online]. 2023 [cit. 2024-02-21]. Dostupné z: https://homey.app/en-us/wiki/what-is-zigbee/.

[6] Bao, K., Allerding, F. a Schmeck, H. User behavior prediction for energy management in smart homes. In: *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. 2011, sv. 2, s. 1335–1339. DOI: 10.1109/FSKD.2011.6019758.

[7] Bergero, F. a Kofman, E. PowerDEVS: A tool for hybrid system modeling and real-time simulation. *Simulation*. Leden 2011, sv. 87, s. 113–132, [cit. 2024-03-02]. DOI: 10.1177/0037549710368029.

[8] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Granger, J. *OpenHAB Review* [online]. [cit. 2024-02-21]. Dostupné z: https://butterflymx.com/blog/openhab-review/.

[10] Jaihar, J., Lingayat, N., Vijaybhai, P. S., Venkatesh, G. a Upla, K. P. Smart Home Automation Using Machine Learning Algorithms. In: *2020 International Conference for Emerging Technology (INCET)*. 2020, s. 1–4. DOI: 10.1109/INCET49848.2020.9154007.

[11] Krohn, J., Beyleveld, G. a Bassens, A. *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*. 1st. Addison-Wesley Professional, 2019. ISBN 0135116694.

[12] Liang, T., Zeng, B., Liu, J., Ye, L. a Zou, C. An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network For Smart Home. *IEEE Access*. 2018, sv. 6, s. 49237–49247. DOI: 10.1109/ACCESS.2018.2868984.

[13] Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

[14] Nithin Buduma, J. P. *Fundamentals of Deep Learning*. Second. O'Reilly Media, 2022. ISBN 9781492082187.

[15] Olah, C. *Understanding LSTM Networks* [online]. 2015 [cit. 2024-04-23]. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[16] Sebastian Raschka, V. M. *Python Machine Learning*. Second. Packt, 2017. ISBN 9781787125933.

[17] Sharma, S., Sharma, S. a Athaiya, A. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*. Květen 2020, sv. 04, s. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[18] Shea, S. *Z-Wave* [online]. 2018 [cit. 2024-02-21]. Dostupné z: https://www.techtarget.com/iotagenda/definition/Z-Wave.

[19] Team, E. *What Is the MQTT Protocol and How Does it Work?* [online]. 2023 [cit. 2024-02-21]. Dostupné z: https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt.

[20] Team, E. *MQTT Broker: How It Works, Popular Options, and Quickstart* [online]. 2024 [cit. 2024-02-21]. Dostupné z: https://www.emqx.com/en/blog/the-ultimate-guide-to-mqtt-broker-comparison.

[21] Wikipedia. *Named pipe* [online]. 2022 [cit. 2024-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Named_pipe.

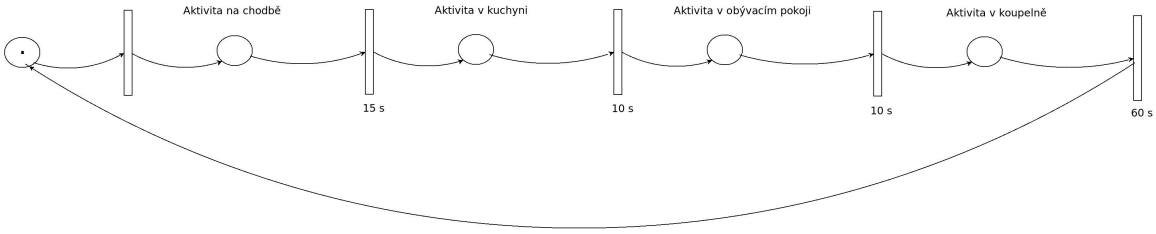
[22] Wikipedia. *Internet of things* [online]. 2024 [cit. 2024-02-21]. Dostupné z: https://en.wikipedia.org/wiki/Internet_of_things#.

[23] Zhou, Z. *MQTT QoS 0, 1, 2 Explained: A Quickstart Guide* [online]. 2023 [cit. 2024-02-21]. Dostupné z: https://www.emqx.com/en/blog/the-ultimate-guide-to-mqtt-broker-comparison.

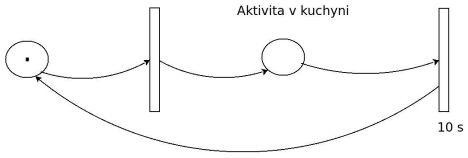
Příloha A

Petriho sítě pro jednotlivé experimenty

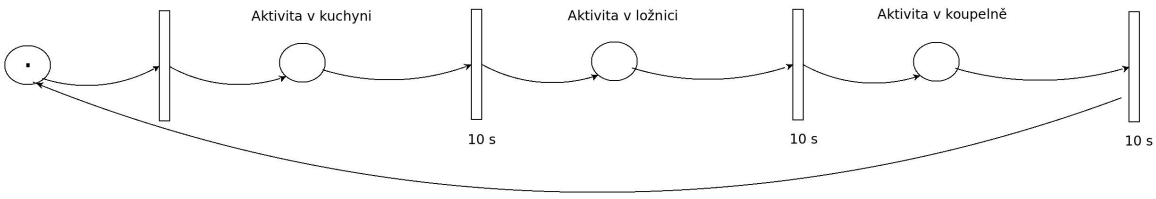
V této příloze se nacházejí Petriho sítě modelující aktivity vykonávané uživateli.



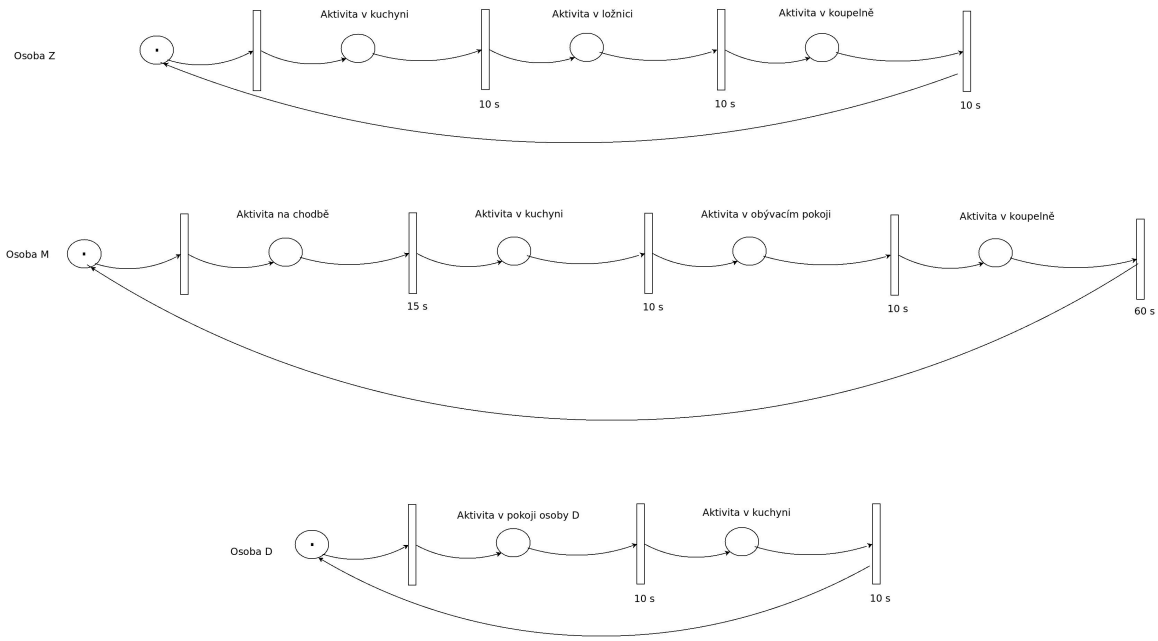
Obrázek A.1: Petriho síť pro experiment 5.1



Obrázek A.2: Petriho síť pro experiment 5.2



Obrázek A.3: Petriho síť pro experiment 5.3



Obrázek A.4: Petriho síťe pro experimenty 5.4 a 5.5

Příloha B

Obsah přiloženého paměťového média

```
/
├── latex/
├── src/
│   ├── controlSystem/
│   └── simulationModel/
│       ├── customAtomicBlocks/
│       └── powerDEVSModels/
├── README.md
└── xvalik04.pdf
```

- složka src obsahuje zdrojové kódy programů
- složka latex obsahuje zdrojové kódy a potřebné soubory pro vyhotovení textu práce
- soubor README.md obsahuje spouštěcí manuál programů