VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

NA SIMULACI ZALOŽENÝ VÝVOJ SYSTÉMU ŘÍZENÍ DISTRIBUCE TEPLA

SIMULATION-BASED DEVELOPMENT OF HEATING CONTROL SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN TOMEČEK**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Tomeček Jan**
Program: Informační technologie
Název: **Na simulaci založený vývoj systému řízení distribuce tepla**
Simulation-Based Development of Heating Control System
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte problematiku řídicích systémů a IoT pro Smart Home. Prostudujte možnosti aplikací umělé inteligence v řídicích systémech, zaměřte se na metody strojového učení.
2. Analyzujte existující systém vytápění s více zdroji. Vytvořte simulační model systému vytápění včetně jeho řízení.
3. Simulační model použijte pro návrh centrálního řízení. Použijte vhodné metody pro optimalizaci využití zdrojů.
4. Řídicí systém realizujte s využitím prvků na bázi SoCs Espressif a Raspberry Pi. Zvolte vhodný způsob realizace s využitím existujících i nově navržených komponent a s potenciální možností další optimalizace.
5. Ověřte funkčnost a vlastnosti systému řízení v reálném provozu. Vyhodnoťte dosažené výsledky a vytvořte plakát shrnující tuto práci.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- První 2 body a část návrhu.

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

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce se zabývá optimalizací ohřevu bojleru z externích zdrojů. V práci jsem vytvořil simulační model systému ohřevu vody. Následně jsem pomocí simulačního modelu navrhl možné optimalizace řízení ohřevu vody. Použitou metodou pro optimalizaci byl algoritmus hlubokého Q-učení. Výsledek této práce ukazuje vyžití simulace pro vývoj a optimalizaci řídících systémů.

Abstract

This thesis is about optimalization of boiler heating from external sources. I have created a simulation model of Heating Control System. Subsequently, using a simulation model, I proposed possible optimizations for water heating control. The used optimization method was deep Q-learning. The result of this work shows the use of simulation for the development and optimalization of control systems.

Klíčová slova

Simulace, Řídící systémy, Chytrá domácnost, Strojové učení, Q-učení, Hluboké Q-učení

Keywords

Simulation, Control system, Smart home, Machine learning, Q-learning, deep Q-learning

Citace

TOMEČEK, Jan. *Na simulaci založený vývoj systému řízení distribuce tepla.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Vladimír Janoušek, Ph.D.

Na simulaci založený vývoj systému řízení distribuce tepla

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.

.....
Jan Tomeček
4. května 2022

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. Vladimíru Janouškovi, Ph.D. za odbornou pomoc.

Obsah

Kapitola 1

Úvod

Pojmy chytrá domácnost a umělá inteligence se v poledních letech dostaly do povědomí většiny lidí. Řešení chytrých domácností nabízí několik společností a existuje i mnoho volně dostupných řešení pro nadšence a kutily.

Chytré domácnosti nám můžou například usnadnit každodenní práci, zautomatizovat denní rutinu, ale i zvýšit efektivitu práce, či snížit spotřebu energií. IoT je jedním z pojmů, které se často pojí k chytrým domácnostem, ale zahrnuje i další oblasti.

S umělou inteligencí, neboli AI, se setkáváme stále častěji. Toto označení můžeme nalézt už i u mobilních telefonů, domácích elektrospotřebičů, grafických karet a u další elektroniky.

Cílem práce je navrhnout řídící systém, který zabezpečuje ohřev bojleru s užitkovou vodou v rodinném domě. Kromě vestavěného elektrického ohřevu bojleru se k ohřevu bojleru využívají externí tepelné zdroje, jako jsou solární panely nebo teplovodní okruh vytápění domu. Tyto zdroje se využívají především pro snížení nákladů za ohřev vody pomocí elektřiny. Řídící sytém by tak měl optimálně využívat dané externí zdroje, než využívalo dosavadní řešení. Pro tuto optimalizaci bude využita umělá inteligence, konkrétně strojové učení.

Pro vývoj řídícího systému je potřebný simulační model celého systému. Optimalizace jsou navrženy pomocí simulačního modelu. Jednou z možných optimalizací je využití strojového učení, konkretně hluboké Q-učení.

Výsledné řešení bude využívat prvky na bázi SoCs Espressif a Raspberry Pi.

V další kapitole této práce je rozebrána problematika Řídících systémů a IoT pro Smarthome a Strojové učení. Podrobněji je pak popsáno posilované učení a umělá neuronová síť. Třetí kapitola se zabývá vytvořením simulačního modelu, nejprve v ní jsou vysvětleny pojmy, které souvisí se simulacemi, dále je analyzován simulovaný systém a následně vytvoření abstraktního a simulačního modelu systému. Čtvrtá kapitola popisuje navržené řešení výsledného řídícího systému. Popisuje použité nástroje a jednotlivé části navrženého systému. V páté kapitole je popis implementace navrženého systému, včetně implementace hlubokého Q-učení. Šestá kapitola zhodnocuje výsledky optimalizovaného řízení ohřevu vody v bojleru a ověřuje je pomocí simulačního modelu. V závěru je vyhodnocen výstup práce.

Kapitola 2

Rozbor problematiky

Tato kapitola rozebírá a popisuje pojmy, které jsou v práci dále použity. Nejprve budou vysvětleny pojmy Řídicí systémy a IoT. Dále bude popsáno strojové učení a jeho druhy. Detailněji bude rozebráno posilované učení a jeho části. Následně Markovské rozhodovací procesy a algorimy Q-učení a hluboké Q-učení. K hlubokému Q-učení budou vysvětleny umělé neuronové sítě včetně pojmu Hluboká neuronová síť. Poté bude popsán umělý neuron a aktivační funkce Sigmouid a ReLU.

2.1 Řídící systémy a IoT pro Smart Home

2.1.1 Řídící systémy

Řídícím systémem rozumíme systém, který propojuje jednotlivé komponenty, které jsou k systému připojeny. Tyto komponenty mohou být například senzory, nebo subsystémy, které vykonávají určitou činnost. Řídící systém se pak stará o správné řízení a ovládání komponent připojených k systému [10].

Tento systém komunikuje s komponenty různými způsoby například pomocí ethernetu, Wi-Fi, protokolu Zigbee, nebo protokolu Z-Wave.

Centrální řídící jednotka, která provozuje tento systém se nazývá hub. Ostatní zařízení v systému se k němu připojují a komunikují s ním. Také poskytuje uživatelské rozhraní, kde zobrazuje informace o systému a poskytuje možnosti interakce s ním.

Úlohy a schopnosti řídícího systému se liší od konkretních aplikací. Průmyslové řídící systémy mohou mít na starost rutiny stojů, také mohou zajišťovat bezpečný provoz v kontaktu s člověkem. Během provozu se může řídící systém setkat s různými chybami, které vyvolá řízený systém. Řídící systém pak musí chybu vyhodnotit a korektně se zachovat. U průmyslových řídících systému může dojít k interakci s člověkem a v situacích, ve kterých se vyskytne porucha systému, špatné vyhodnocení chyby může vést ke zranění. Pokud tyto systémy v běžném provozu dohlíží na správný chod strojů obsluhovaných lidmi, musí splňovat přísné bezpečností podmínky. Řídící systémy chytrých domácností se zpravidla nesetkávají se situacemi, kdy by mohly ohrozit lidský život. Nemusí tak klást velký důraz na bezpečnostní scénáře. Dohlíží na správný chod a komunikaci jednotlivých připojených zařízení. Spouští uživatelem definované rutiny, které může optimalizovat a přizpůsobovat potřebám uživatele automaticky.

2.1.2 IoT - Internet věcí

Definice Internetu věcí není přesně standardizovaná. V principu jde o skupinu zařízení, které jsou schopny vzájemně komunikace přes síť.

Společnost IBA group definovala Internet věcí jako:

„*Internet věcí (Internet of Things, IoT) označení pro síť fyzických přístrojů ovladatelných i na dálku pomocí internetu. Zařízení spolu mohou prostřednictvím internetu komunikovat a vzájemně na sebe reagovat*“ [2].

Společnost Rascasone definovala Internet věcí jako:

„*IoT lze jednoduše vysvětlit jako ekosystém počítačů a chytrých zařízení či strojů, které jsou schopny vzájemně komunikovat nebo spolupracovat bez asistence člověka*“ [8].

IoT tedy zahrnuje velkou skupinu elektroniky. Společnou vlastností je schopnost vzájemné komunikace. Zařízení mohou komunikovat přes různé technologie, nejčastěji se ale, podle názvu, jedná o komunikaci přes internet. Další vlastností těchto zařízení je nízká náročnost na výpočetní výkon. Často se totiž jedná o mikroprocesorová zařízení, která sbírají data ze senzorů. Díky nízkému výpočetnímu výkonu nemají ani vysokou spotřebu energie, což je také pro některé aplikace důležitým požadavkem, protože mohou být napájeny z baterií. IoT je tak vhodný způsob pro realizaci chytrých domácností. Zejména díky snadné rozšířitelnosti počtu zařízení v domácnosti za předpokladu, že umí spolu komunikovat.

2.1.3 Smart Home

Smart Home, neboli chytrá domácnost, je zahrnutí prvků IoT do běžných spotřebičů a elektroniky. Chytrý spotřebič dokáže komunikovat s řídící jednotkou nebo například s aplikací v telefonu. Integrace těchto spotřebičů do domácnosti a jejich vzájemné propojení tvoří chytrou domácnost. Na trhu existuje několik komerčních řešení, ale i volně dostupné řešení pro kutily. Chytrá domácnost má často i hlasového asistenta, který dokáže komunikovat s ostatními zařízeními a poskytuje tak snadné a přívětivé uživatelské rozhraní.

2.2 Strojové učení

Jednou z částí umělé inteligence je Strojové učení. Jedná se o algoritmy a techniky, které počítačovému systému dávají možnost „učit se“. V tomto kontextu učení znamená zefektivnit schopnost přizpůsobení se změnám okolního prostředí pomocí změny vnitřního stavu [11].
Základní metody strojového učení:

- Unsupervised Learning - Učení bez učitele - systém provádí rozhodnutí pouze na základě vstupních dat
- Supervised Learning - Učení s učitelem - systém provádí rozhodnutí na základě vstupních dat a očekávaných výstupních dat
- Reinforcement Learning - Posilované učení - systém mění způsob rozhodování na základě zpětné vazby, za provedenou akci

Možné aplikace strojového učení v řídících systémech jsou například validace dat pomocí klasifikace, uzpůsobení rozhodování systému na základě odhadu chování podle vstupních dat, spouštění rutin systému na základě vypozorovaných návyků uživatelů, nebo optimalizace jednotlivých úloh řídícího sytému.

2.2.1 Učení s učitelem

Jedna z nejpoužívanějších metod strojového učení. Model se učí na základě předem označených dat. Dostane dvojici dat – vstupní data a očekávaný výstup. Následně projde sadu těchto dat a na základě očekávaného výstupu se učí. Model pak dokáže rozdělit data do skupin podle uvedených vlastností. Využívá se pro lineární a logistickou regresi, vícetřídní klasifikaci a jiné [5].Vícetřídní klasifikace spočívá v rozdělení dat do předem stanovených skupin neboli tříd. Regrese spočívá v predikování spojitých hodnot.

Nevýhodou této metody je především zajištění dostatečného počtu dat pro trénování. Také je při učení nutný externí učitel, který musí agentovi ukázat očekávaná chování ve formě zmiňovaných očekávaných výstupů pro vstupní data.

2.2.2 Učení bez učitele

Učení bez učitele používá obecnější přístup, kdy se snaží hledat vzory a souvislosti nad vstupními daty. Stejně jako učení s učitelem obdrží sadu dat, ale tentokrát bez očekávaného výstupu. Model pak dokáže data rozdělit do skupin podle pozorovaných vlastností, bez znalosti významu jednotlivých vlastností. Využívá se například pro K-mean shlukování, analýzu hlavních a nezávislých komponent atd. [5]

2.3 Posilované učení

Posilované (také zpětnovazebné) učení je založeno na agentovi, který interaguje s prostředím a vybírá v daném stavu akci, za kterou dostane určitou odměnu. Agent musí sám zjistit, které akce mu v daném stavu přinesou nejvyšší celkovou odměnu. Musí tedy sám vyzkoušet jednotlivé akce a tím zjistit, které akce v jednotlivých stavech vedou k nejvyšší celkové odměně. Tento proces „pokus – omyl“ je základním principem posilovaného učení [9].

V porovnání s učením s učitelem nemáme dopředu známé správné rozhodnutí. Agent pouze získá odměnu za právě provedenou akci, u učení s učitelem zasahuje do procesu učení externí pozorovatel. Posilované učení tak dokáže nalézt i optimálnější akce pro dané stavy prostředí, které nemusí být zřejmé na první dojem [9].

Posilované učení má tyto základní části:

- **Strategii** – Popisuje způsob učení chování Agenta v daném stavu. Tedy jak se v daném stavu agent rozhodne tak, aby maximalizoval celkovou odměnu. Pokud by agent nenásledoval tuto strategii, tak by vybíral jen nejlépe ohodnocenou akci v daném stavu. Při použití strategie může vybírat akce, které nejsou nejlépe ohodnocené v daném stavu, ale vedou k nejvyšší celkové odměně [9]

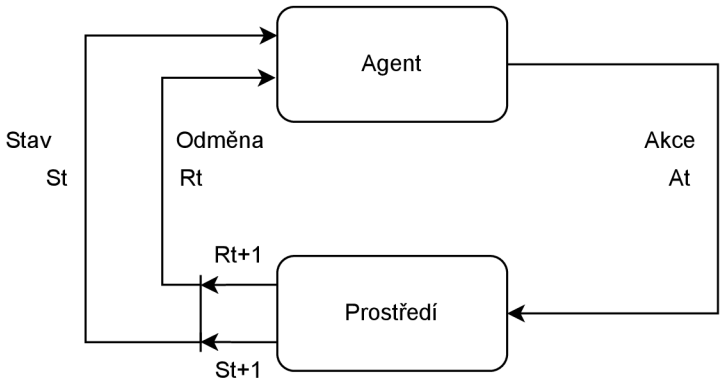
- **Signál odměny** – Cíl posilovaného učení. Agent obdrží odměnu za každou provedenou akci v prostředí. Agentův úkolem je maximalizovat tuto hodnotu v delším běhu. Na základě této odměny agent mění svoji strategii, aby se rozhodl, pokud se dostane do dalšího stavu, pomocí jiné akce [9]

- **Hodnotící funkci** – Ovlivňuje chování agenta ve větším výhledu. Bere v úvahu maximální možnou odměnu, kterou lze získat ze stavu, ve kterém se agent právě nachází [9]
- **Model** – Jedná se o model prostředí ve kterém se agent nachází. Tedy simulaci celého sytému, nebo abstrakci daného systému, pomocí které se bude agent vnímat reálný systém [9]

2.3.1 Markovské rozhodovací procesy

„Markovské procesy jsou náhodné procesy, které splňují Markovovu vlastnost: nasledující stav procesu zavisí jen na aktualním stavu (ne na minulosti)“ [7].

Markovské rozhodovací procesy jsou matematická forma procesu posilovaného učení. Tedy Agent může v daném stavu S_t může vybrat akci A_t, dostupnou v daném stavu. Touto akcí se dostane do nového stavu $S_t + 1$ a obdrží odměnu $R_t + 1$ [9].



Obrázek 2.1: Proces interakce agenta s prostředím. Agent se v čase t nachází ve stavu S_t a obdržel odměnu R_t. Následně se pomocí akce A_t přesunul do stavu $S_t + 1$ a obrdržel odměnu $R_t + 1$ [9]

.

Definice 1 Markovův rozhodovací proces *je čtveřice tvaru* $M = (S, A, P_a(s, s'), R_a(s, s'))$, *kde:*

- S *je konečná množina stavů, do kterých se může agent dostat*
- A *je konečná množina akcí, které může agent vykonat v daném stavu*
- $P_a(s, s')$ *je pravděpodobnost, že akce* a *ve stavu* s *v čase* t *povede v čase* $t+1$ *do stavu* s'
- $R_a(s, s')$ *je odměna, kterou agent obdrží po přechodu stavu na* s' *ze stavu* s *s pravděpodobností přechodu* $P_a(s, s')$

Cíl agenta je tedy zvolit vhodnou strategii tak, aby získal co největší celkovou odměnu. Rozhodovat se ale může pouze na základě aktuálního stavu, ve kterém se nachází, podle pravděpodobnosti funkce přechodu v daném stavu.

2.3.2 Q-učení a hluboké Q-učení

Q-učení spočívá ve výběru nejlépe hodnocené akce v daném stavu. Toto hodnocení se nazývá Q-hodnota. Výsledkem jsou dvojice akce a odměna. Používá Q-tabulku, což je datová struktura, kde se ke každému stavu uchová nejvyšší možná celková odměna za tuto akci – Q-hodnoty. Agent se tedy v každém stavu podle tabulky rozhodne, která akce ho dovede k nejvyšší odměně [1].

K optimalizaci Q-hodnoty pak slouží tato rovnice:

$$Q_{(s,a)} = Q_{(s,a)} + \alpha(r + \gamma maxQ_{(s',a')} - Q_{(s,a)}) \tag{2.1}$$

kde: s = aktuální stav
a = akce
γ = diskontní faktor
α = koeficient učení, $\alpha \in (0, 1\rangle$
r = odměna
$maxQ_{(s',a')}$ = nejvyšší Q-hodnota v následujícím stavu po přechodu akce a ze stavu s

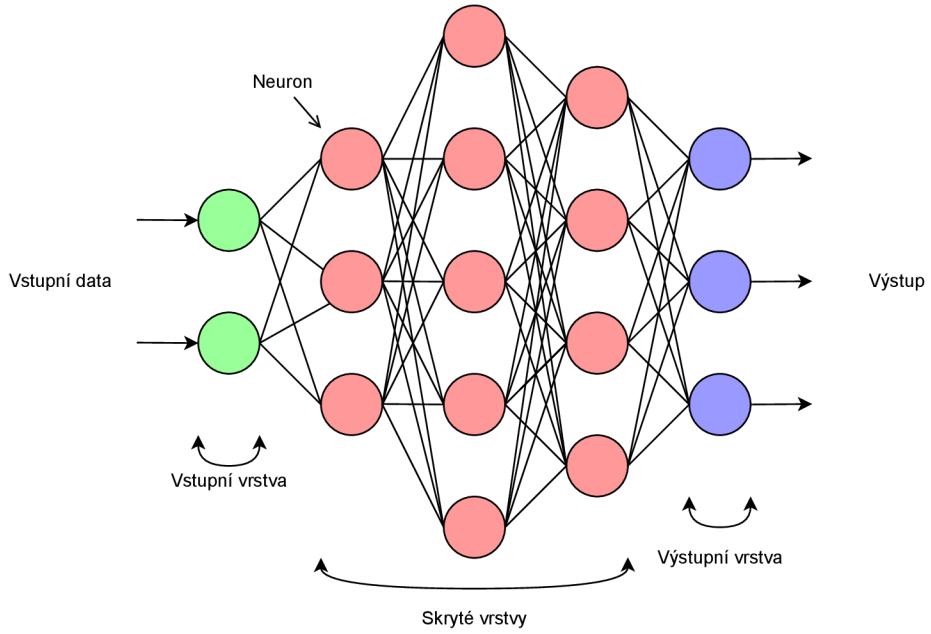
Agent nejprve zkoumá Q-hodnoty jednotlivých akcí, pokud by neprozkoumal dostatek akcí, mohl by zvolit neoptimální strategii. Proto během procesu učení vybírá i náhodné akce s pravděpodobností $\epsilon, \epsilon \in (0, 1\rangle$. Epsilon se během učení snižuje [1].

Hluboké Q-učení využívá, oproti klasickému Q-učení, umělou neuronovou síť. Vstupem této neuronové sítě je stav prostředí a výstupem jsou Q-hodnoty jednotlivých akcí. Hluboké Q-učení je vhodné pro prostředí s velkým počtem stavů nebo u prostředí se spojitými hodnotami stavů. V takovém prostředí není vhodné vytvářet Q-tabulku pro každý stav, ale nahradí ji neuronová síť. Optimalizátor této sítě pak učí neuronovou síť tím, že se snaží zmenšovat výstup ztrátové funkce.

2.4 Umělá neuronová síť

Umělá neuronová síť je hierarchická skupina neuronů a jejich propojení. Každý neuron na výstup posílá zprávy, nebo signály na základě jeho vstupu. Jednotlivé propojení neuronů má nastavenou váhu. Tímto vzniká síť neuronů. Učení neuronové sítě probírá změnou vah jednotlivých spojení neuronů [4].

Modelem hlubokého učení je hluboká neuronová síť. Tato síť je tvořena různými vrstvami neuronů. Každá síť obsahuje vstupní, skryté a výstupní vrstvy. Každá tato vrstva může mít různý počet neuronů, nejméně však jeden. Vstupní vrstva má tolik neuronů, jaký je počet vstupních dat. Výstupní zase takový počet neuronů, aby odpovídal počtu výstupů. Skryté vrstvy se mohou svými počty lišit.

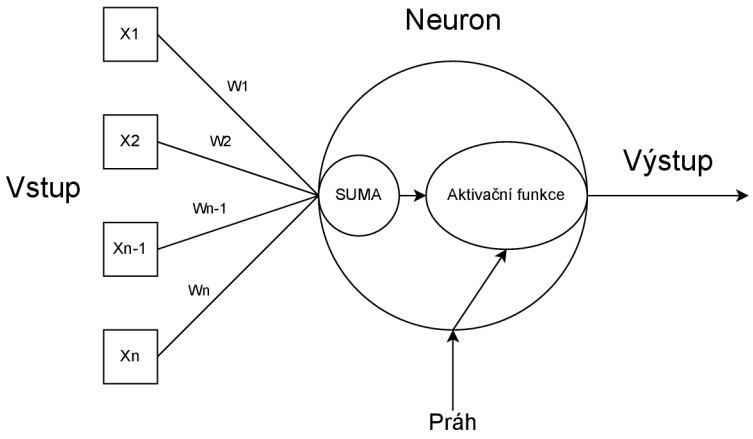


Obrázek 2.2: Ilustrace umělé neuronové sítě. Má celkem 5 vrstev. Jde o plně propojenou síť – každý neuron je propojen se všemi neurony v další vrstvě. Vstupní vrstva má 2 vstupy, následují 3 skryté vrstvy s různým počtem neuronů(3,5,4) a výstupní vrstva má 3 výstupy

Každý neuron má svoji aktivační funkci. Ta vezme sumu všech vstupních signálů a vypočítá výslednou hodnotu, která se při překročení daného prahu propaguje na výstupy neuronu. Tato funkce upravuje hodnoty signálů tak, aby spadaly do požadovaného intervalu [4].

Každé propojení mezi neurony má přidělenou váhu, která mění vliv spojení na neuron, tedy jak moc ovlivní výstup jednoho neuronu vstup toho druhého. Tyto váhy jsou na začátku učení nastaveny náhodně a během učení se upravují.

Při učení neuronové sítě se využívá chybová funkce. Výstup této chybové funkce záleží na tom, jak moc přesné výsledky dostáváme z neuronové sítě. Cílem učení je snižovat hodnotu této chybové funkce.



Obrázek 2.3: Ilustrace neuronu. X jsou vstupní hodnoty. W jsou váhy jednotlivých propojení.

2.4.1 Aktivační funkce

Cílem aktivačních funkcí je omezit výstupní hodnoty neuronů do určitého intervalu. Pokud bychom je nevyužívali, mohli by výstupy neuronů být v rozsahu od mínus nekonečna do plus nekonečna. Tak bychom ale nemohli snadno určit hranici aktivace neuronu. Potřebujeme tedy, alespoň pro skryté vrstvy, nelineární aktivační funkce [4].

Nejpoužívanějšími funkcemi jsou Sigmoid a ReLu.

Lineární aktivační funkce

Lineární aktivační funkce je definovaná rovnicí:

$$y = x \tag{2.2}$$

kde: $x =$ vstup neuronu

Výstupem této funkce jsou hodnoty v intervalu $(-inf, inf)$. Funkce není, díky svému průběhu, vhodná jako aktivační funkce skrytých vrstev neuronových sítí.

Sigmoid

Sigmoid aktivační funkce je definovaná rovnicí:

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

kde: $x =$ vstup neuronu

Výstupem této funkce jsou hodnoty v intervalu $(0, 1)$. Díky průběhu funkce docílíme lepšího procesu učení, neboť odpovídá principu, že menší váha spoje - menší výstup a vyšší váha spoje - větší výstup.

ReLU

ReLU aktivační funkce je popsaná vztahem:

$$y = max(0, x) \tag{2.4}$$

kde: $x =$ vstup neuronu

Platí tedy, že pro záporné vstupy bude na výstupu 0 a kladné vstupy budou na výstupu nezměněny. Její výhodou je, že urychluje výpočet aktivační funkce v procesu učení. Urychluje proces učení tím, že pro záporné vstupy neuron neovlivňuje další neurony.

Kapitola 3

Simulační model

V této kapitole jsou nejprve stručně popsány simulace a pojmy s nimi spojené. Dále je v ní analyzován aktuální systém ohřevu vody v bojleru s více tepelnými zdroji. Podle zjištěných vlastností systému je zde popsán vytvořený abstraktní model. U abstraktního modelu tohoto systému budou popsány použité matematické vztahy. Simulační model byl pro ověření validity implementován v jazyce C++.

Pro ověření validity modelu byla také použita knihovna Simlib [6].

3.1 Simulace

Základní pojmy spojené se simulacemi:

- „***Simulace*** *je získavání nových znalostí o systému experimentováním s jeho modelem*“ [7].
- **Abstraktní model** je zjednodušený popis zkoumaného systému na základě jeho vlastností [7].
- **Simulační model** je implementace abstraktního modelu v programovacím jazyce [7].

Proces vytvoření simulačního modelu tedy zahrnuje:

- Analýzu zkoumaného systému a soupis jeho vlastností
- Vytvoření abstraktního modelu na základě získaných vlastností
- Vytvoření simulačního modelu = programu, který implementuje abstraktní model

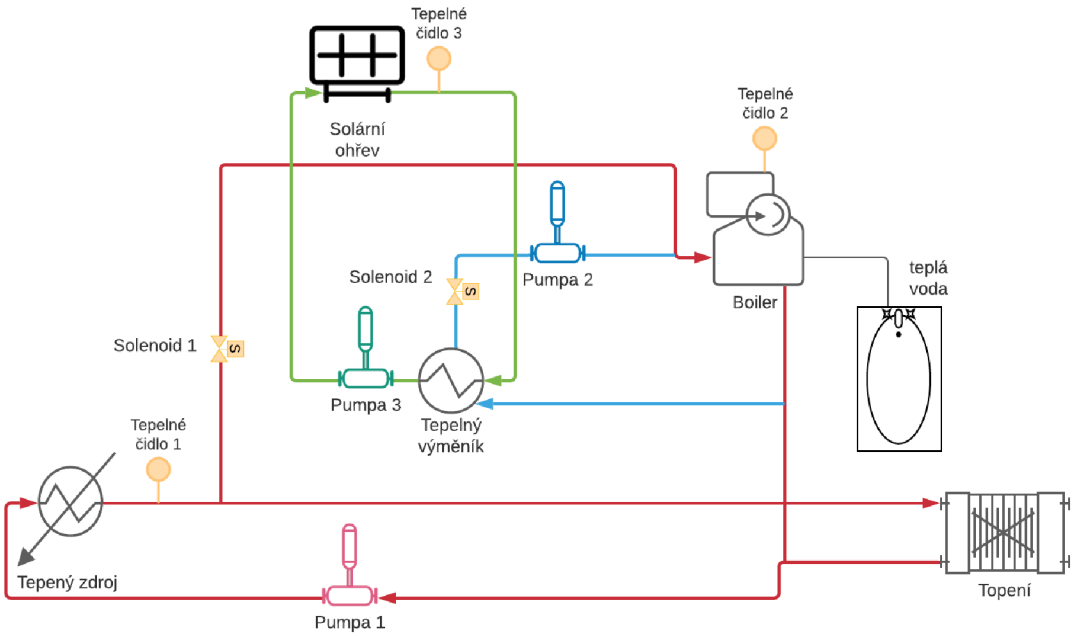
3.2 Analýza aktuálního systému

Aktuální systém ohřevu bojleru se skládá z těchto částí:

- Bojler
- Teplovodní okruh vytápění rodinného domu
- Teplovodní okruh solárních panelů
- Řídící jednotka

O spouštění ohřevu bojleru se stará řídící jednotka. Tu tvoří Arduino nano. Pomocí 3 teplotních čidel (viz schéma aktuálního systému) snímá teploty teplovodních okruhů a bojleru. Na základě pevně stanoveného prahu $5°C$ spouští ohřev bojleru. Ohřev z teplovodního okruhu vytápění rodinného domu probíhá pomocí sepnutí solenoidu 1. Ohřev z teplovodního okruhu solárních panelů probíhá pomocí sepnutí solenoidu 2, pumpy 2 a pumpy 3.

Kvůli pevně stanovenému prahu $5°C$ nemusí využívat ani jeden z externích zdrojů optimálně.



Obrázek 3.1: Schéma aktuálního systému. Systém obsahuje bojler a dva externí zdroje. První zdroj je teplovodní okruh vytápění domu. Ten je ohříván kamny na dřevo. Druhým zdrojem je teplovodní okruh solárních panelů.

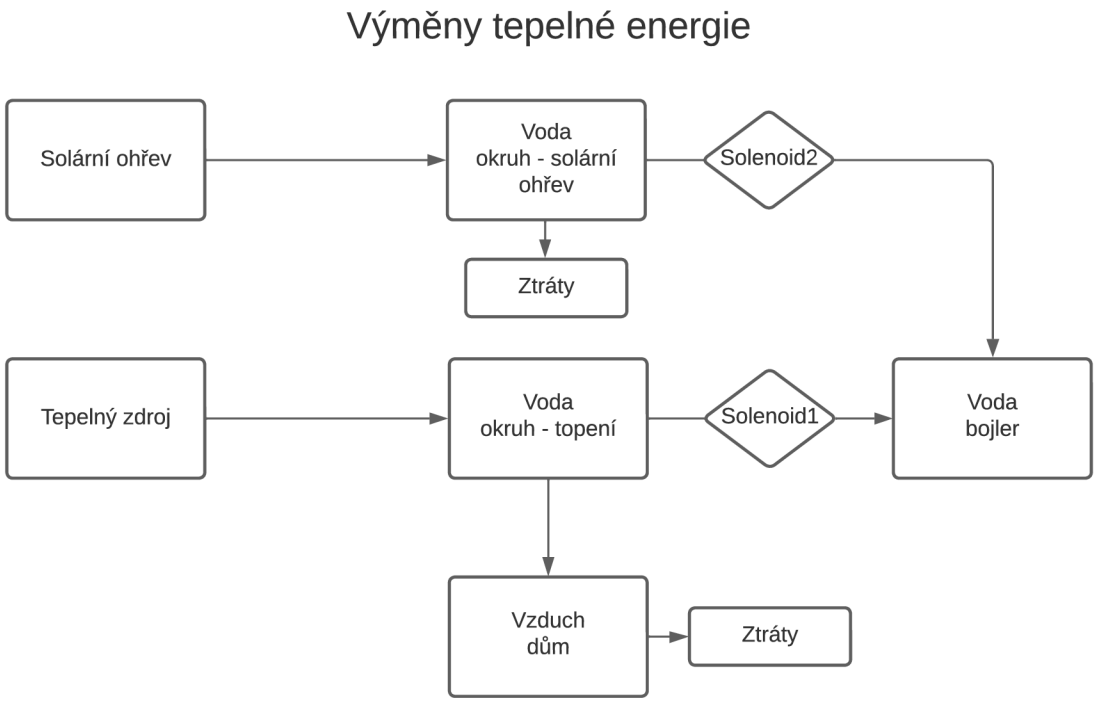
3.3 Použité nástroje

3.3.1 Knihovna Simlib

Simlib je knihovna v programovacím jazyce C++. Slouží pro tvorbu simulačních modelů. Podporuje tvorbu diskrétních i spojitých modelů. Obsahuje třídu `Event` reprezentující událost. Poskytuje řízení simulace pomocí „next-event“ algoritmu za pomoci kalendáře.

3.4 Abstraktní model

Abstraktní model byl vytvořen na základě schématu zapojení systému viz Obrázek 3.1. Pro simulační experimenty je důležitá hlavně tepelná výměna mezi tepelnými zdroji a bojlerem, proto byly ostatní části systému zjednodušeny. Tepelné zdroje také zjednodušeny a jejich chování a parametry průběhu výkonu byly experimentálně odvozeny.



Obrázek 3.2: Graf abstraktního modelu

Obrázek 3.3: Návrh modelu

Abstraktní model byl navržen s ohledem na zmiňované požadavky. Na základě uvedeného abstraktního modelu, byl pomocí knihovny Simlib [6] vytvořen diskrétní simulační model pro ověření validity modelu.

3.5 Simulační model

3.5.1 Ohřev vody v okruhu topení

Tepelný zdroj produkuje tepelnou energii. Tato energie je využita pro ohřátí vody v okruhu topení. Pro výpočet teploty vody je použit vztah:

$$t_2 = \frac{Q}{m * c} + t_1 \tag{3.1}$$

kde: t_2 = nová teplota vody
Q = tepelná energie z tepelného zdroje
m = hmotnost vody v okruhu
c = měrná tepelná kapacita vody
t_1 = teplota vody v okruhu

Odvozeno ze vztahu pro výpočet tepelné energie [12].

3.5.2 Tepelná výměna

Pro simulaci tepelné výměny mezi teplovodním okruhem topení, vodou v bojleru, vytápěním domu byly použity následující vztahy:

Výpočet výsledné teploty dvou látek při výměně tepla

$$t = \frac{m_1 * c_1 * t_1 + m_2 * c_2 * t_2}{m_1 * c_1 + m_2 * c_2} \tag{3.2}$$

kde: t = výsledná teplota
m_1, m_2 = hmotnost
c_1, c_2 = měrná tepelná kapacita
t_1, t_2 = počáteční teplota

Výpočet tepla, které se může přenést

$$Q = m * c * (t_1 - t) \tag{3.3}$$

kde: Q = teplo
m = hmotnost
c = měrná tepelná kapacita
t_1 = počáteční teplota
t = výsledná teplota

Výpočet výsledných teplot

$$t_{f1} = \frac{-k * Q}{m_1 * c_1} + t_1 \tag{3.4}$$

$$t_{f2} = \frac{k * Q}{m_2 * c_2} + t_2 \tag{3.5}$$

kde: t_{f1}, t_{f2} = výsledné teploty
k = koeficient efektivity přenosu, $k \in \langle 0, 1 \rangle$
Q = teplo
m_1, m_2 = hmotnost
c_1, c_2 = měrná tepelná kapacita
t_1, t_2 = počáteční teplota

3.5.3 Tepelné zdroje

Jednotlivé zdroje (kamna na dřevo a solární panely) byly aproximovány těmito vztahy:

- Kamna na dřevo:

$$P = \frac{65}{t_{voda}} * \frac{25}{t_{vzduch}} * \frac{72000000 * \pi}{180} * sin(\frac{\pi * (T_l - T + 1)}{90}) \tag{3.6}$$

kde: t_{voda} = teplota teplovodního okruhu topení
t_{vzduch} = teplota vzduchu v domě
T_l = modelový čas posledního přiložení
T = modelový čas
P = výkon v J/m

- Solární panely

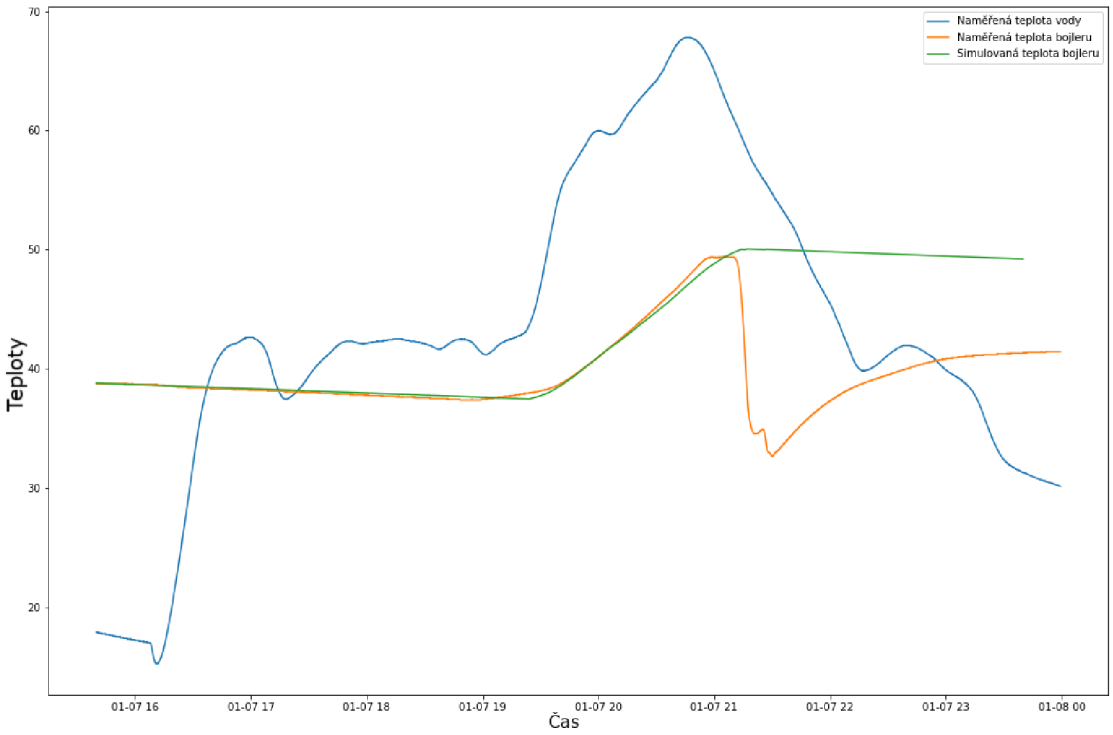
$$P = \frac{65}{t_{voda}} * \frac{2400000 * \pi}{180} * sin(\frac{\pi * (T)}{360}) \tag{3.7}$$

kde: t_{voda} = teplota teplovodního okruhu solárních panelů
T = modelový čas, $T \in \langle 0, 160 \rangle$
P = výkon v J/m

3.5.4 Validita modelu

Validita byla ověřena simulačními experimenty, ve kterých byla použita dosavadní logika spuštění tepelné výměny (ohřevu vody v bojleru) a výstup byl porovnaný s naměřenými hodnotami.

Z výsledků lze prohlásit model za validní.



Obrázek 3.4: Graf výsledků simulace a skutečných naměřených hodnot.

Kapitola 4

Návrh řešení

V této kapitole jsou popsány knihovny, použité při návrhu a implementaci, návrh celého řídícího systému a optimalizace využití zdrojů.

V návrhu řídícího systému budou popsány jednotlivé části a jejich komunikace. Popsány budou i jednotlivé hardwarové prvky, použité při realizaci.

U návrhu optimalizace bude popsána optimalizace pomocí strojového učení, konkrétně pomocí algoritmu hlubokého Q-učení a jednotlivé části modelu strojového učení.

4.1 Použité nástroje

Model prostředí pro agenta strojového učení byl implementován v Pythonu3 pomocí knihovny OpenAI Gym.

Řídící systém využívá jazyky HTML, PHP, Python3 a Arduino. Centrální jednotka navíc bude používat webový server Apache a protokol mDNS pro překlad doménových jmen.

Optimalizace řízení ohřevu bojleru bude probíhat pomocí posilovaného učení konkrétně algoritmem hlubokého Q-učení. Posilované učení bude implementováno v Pythonu3 pomocí knihoven TensorFlow, Keras, Keras RL.

Použité byly také Python knihovny Numpy, Pandas a Matplotlib.

4.1.1 Knihovna OpenAI Gym

OpenAI Gym je knihovna pro programovací jazyk Python. Obsahuje nástroje pro vývoj a porovnávání algoritmů strojového učení. Především slouží pro tvorbu prostředí, se kterým bude agent interagovat a učit se. Nabízí několik hotových prostředí, ale lze si s touto knihovnou vytvořit i vlastní prostředí.

Hlavní třídou je třída `Env`, reprezentující samostatné prostředí. Dále obsahuje modul `spaces`, který obsahuje třídy `Discrete` a `Box`, které slouží jako reprezentace stavového, či akčního prostoru prostředí pro Agenta.

4.1.2 Knihovna TensorFlow

Tensorflow je populární, volně dostupný soubor knihoven určených pro strojové učení. Poskytuje přehledné rozhraní s různými úrovni abstrakce pro vývoj a učení modelů strojového učení. Díky tomu je vhodný pro návrh a řízení celého procesu strojového učení. Součástí je i vysokoúrovňové rozhraní Keras, které poskytuje jednoduché aplikační rozhraní pro knihovny TensorFlow.

4.1.3 Knihovna Keras

Keras je rozhraní nad knihovnou TensorFlow v programovacím jazyku Python. Je určen pro řešení úloh strojového učení se soustředěním na hluboké učení. Zahrnuje vysokou úroveň abstrakce a jednoduchou obsluhu pro tvorbu umělých neuronových sítí. Podporuje výpočty na procesoru, mikroprocesorech a grafických kartách.

Pro tvorbu modelu umělých neuronových sítí obsahuje třídu `Sequential`, reprezentující jednoduchou jednodimenziální síť s jednou vstupní a jednou výstupní vrstvou. Pro skryté vrstvy obsahuje třídy `Dense` a `Flatten`. Třída `Dense` reprezentuje plně propojenou vrstvu neuronové sítě. Třída `Flatten` slouží pro srovnání vstupních signálů vrstvy do jedné dimenze.

4.1.4 Knihovna Keras-RL2

Keras-RL2 je knihovna v programovacím jazyku Python. Poskytuje agenty pro posilované učení. Je založena na knihovně Keras a je kompatibilní s knihovnou OpenAI Gym [3].

Využívanou třídou z této knihovny bude `DQNAgent`. Což je třída reprezentující agenta pro hluboké Q-učení.

4.1.5 Optimalizátor Adam

Optimalizátor Adam je jeden z optimalizátorů, které poskytuje knihovna Keras. Tento optimalizátor je založený na algoritmu Adam. Jeho název vznikl z anglického sousloví „adaptive moment estimation“.Jedná se o stochastický optimalizátor založený na stochastické metodě poklesu gradientu. Metoda je založená na přepočítávání parametru rychlosti učení.

4.1.6 Numpy

Numpy je velmi populární knihovna jejíž základem jsou homogenní pole. Jedná se o implementaci statického pole o pevné velikosti, určené při inicializaci. Je vysoce optimalizovaná a vhodná pro matematické výpočetní operace. Její další předností je nízká paměťová náročnost. Je vhodná zejména při zpracování většího objemu dat.

4.1.7 Pandas

Pandas je knihovna určená zejména pro tvorbu datových rámců. Pracuje s knihovnou Numpy. Díky tomu je také vysoce efektivní pro matematické výpočty a má nízkou paměťovou náročnost. Její dominantou je snadná úprava datových rámců, vytváření pohledů, reorganizace rámců a shlukování dat na základě jejich hodnot.

4.1.8 Matplotlib

Matplotlib je knihovna pro vizualizaci dat. Jedná se o knihovnu pro tvorbu grafů. Poskytuje velké množství typů grafů a jednoduché rozhraní. Umožňuje úpravu vzhledu grafů. Je kompatibilní s knihovnou Numpy.

4.2 Popis hardwarových částí

4.2.1 Arduino nano

Arduino nano je jednočipová vývojová deska od společnosti Arduino[1]. Je založena na mikroprocesoru ATmega328. Její hlavní výhodou je nízká cena, malé rozměry a snadný vývoj aplikací pomocí vývojového prostředí Arduino. ATmega328 je taktován na 16 MHz, takže se nehodí pro výpočetně náročné aplikace.

Díky své nízké ceně byla použita pro spínání ohřevu bojleru. Její cena byla hlavním důvodem při volbě, protože spínání ohřevu musí fungovat i v případě, že ostatní části systému selžou. Je tedy nenákladné mít opatřených několik kusů této desky a v případě poruchy ji hned vyměnit.

4.2.2 ESP-8266

ESP-8266 je mikroprocesor od společnosti Espressif[2]. Jeho hlavní výhoda je Wi-Fi konektivita. Díky tomu je velmi populární a je používán na řadě vývojových desek. Maximální taktová frekvence je 160 MHz, je tedy vhodnější i pro náročnější aplikace.

Díky podpoře Wi-Fi byl použit pro komunikaci s řídící jednotkou.

4.2.3 Raspberry pi 400

Raspberry pi 400 je jednodeskový počítač od společnosti Raspberry Pi[3]. Tento počítač je založen na modelu Raspberry pi 4, také od společnosti Raspberry Pi. Je zabudován do klávesnice. Jedná se o plnohodnotný počítač s čtyřjádrovým 64-bitovým procesorem. Kromě konektorů jako USB a microHDMI obsahuje i 40pinový GPIO header.

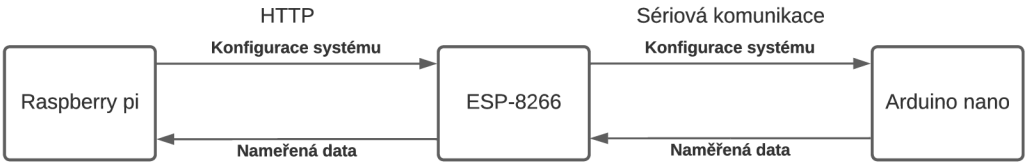
Pro svůj výkon byl použit jako hlavní centrální jednotka.

4.3 Návrh řídícího systému

Řídící systém se bude skládat ze 2 částí:

- mikrokontrolery ESP-8266 a Arduino nano
- centrální jednotka Raspberry pi

Tyto části spolu budou komunikovat pomocí Wi-Fi.



Obrázek 4.1: Schéma komunikace jednotlivých částí řídícího systému.

[1] https://www.arduino.cc/
[2] https://www.espressif.com/
[3] https://www.raspberrypi.com/

4.3.1 Centrální jednotka

Centrální jednotka bude implementována na Raspberry pi 400. Bude poskytovat uživatelské rozhraní přes pomocí HTML stránky, kde uživatelé budou moci kontrolovat naměřené teploty a stav ohřevu vody. Dále bude poskytovat HTTP rozhraní pro mikrokontrolér ESP-8266, přes které bude sbírat naměřená data a zaznamenávat je. Nazpět bude mikrokontroleru posílat hodnoty prahů, při kterém se má spouštět ohřev bojleru z externích zdrojů.

Také bude používat protokol mDNS pro překlad doménového jména v lokální síti. Pro snadnější přístup uživatelů a usnadnění komunikace s mikrokontrolery.

4.3.2 Mikrokontrolery ESP-8266 a Arduino nano

Arduino nano bude měřit teploty bojleru, teplovodního okruhu topení a teplovodního okruhu solárních panelů. Dále bude podle získaných prahů spouštět ohřev bojleru z externích zdrojů a přes sériovou komunikaci vyměňovat data s ESP-8266.

ESP-8266 bude získávat naměřené hodnoty, zobrazovat je na LED display a odesílat je přes HTTP protokol centrální jednotce, v odpovědi získá nové hodnoty prahů a novou konfiguraci pošle nazpět Arduinu nano.

4.4 Optimalizace ohřevu

Cílem optimalizace je nalézt optimálnější práh pro zapínání ohřevu bojleru z externích zdrojů.

Pro optimalizaci využití zdrojů se využije posilované učení. Kontrétně algoritmus hlubokého Q-učení. Jako prostředí se využije simulační model reálného systému.

Výsledkem bude nová hodnota prahu pro řídící jednotku.

4.4.1 Hluboké Q-učení

Pro učení agenta se vytvoří prostředí pomocí knihovny OpenAI Gym. Toto prostředí bude simulačním modelem reálného systému podle abstraktního modelu popsaném v kapitole 3. Dále bude definovat stavový prostor a akce, které může agent provádět. S každou provedenou akcí vyhodnotí a zašle agentovi odměnu.

Hloubková neuronová síť bude vytvořena pomocí knihovny Keras.

Agent bude instancí třídy `DQNAgent` z knihovny Keras-RL2. Prvotní naučení Agenta se provede pomocí zmiňovaného prostředí. Pozdější učení agenta bude probíhat na upraveném prostředí z naměřených dat. Po učení agenta se pomocí naučené hluboké neuronové sítě získají hodnoty nových prahů.

4.5 Hluboké Q-učení

4.5.1 Agent

Jako agent strojového učení bude využit agent z knihovny Keras-RL2. Tento agent bude využívat paměť pro učení a optimalizátor ADAM.

Agent vyžaduje pro interakci s prostředím jeho stav, zda byla dokončena epizoda a odměnu. Pro interakci je důležité rozhraní prostředí, které musí implementovat funkce `step()` a `reset()`. Agent na začátku každé epizody uvede prostředí do výchozího stavu pomocí metody `reset()`. Každý krok interakce volá metodu `step()`.

4.5.2 Funkce odměny

Funkce odměny je důležitou součástí Hlubokého Q-učení. Poskytuje odměnu za každou akci, kterou agent provede.

Za správně provedené akce poskytne agentovi kladnou odměnu. Za nežádoucí akce poskytne agentovi zápornou odměnu. Jelikož se agent snaží maximalizovat celkovou odměnu, měl by uzpůsobovat akce, tak aby získával co nejvyšší kladnou odměnu.

Základní kritéria hodnotící funkce jsou:

- Ohřev vody v bojleru z teplovodního okruhu topení lze zapnout jen pokud je teplota teplovodního okruhu vyšší, než voda v bojleru.
- Ohřev vody v bojleru z teplovodního okruhu solárních panelů lze zapnout jen pokud je teplota teplovodního okruhu vyšší, než voda v bojleru.
- Rozestup mezi zapínáním ohřevu nesmí být menší něž 10 minut, aby se předešlo malým prahům kvůli oscilaci.
- Přednostněji se ohřívá voda z teplovodního okruhu solárních panelů

Funkce odměny je součástí metody `step()` třídy `ModelEnv` implementující prostředí Agenta.

4.5.3 Prostředí

Prostředí pro naučení Agenta bude vytvořeno podle abstraktního modelu popsaného na straně 13. Prostředí bude implementovat třída `ModelEnv`.

Třída `ModelEnv` bude potomkem třídy `Env` z knihovny OpenAI Gym. Kvůli kompatibilitě rozhraní s Agentem bude implementovat tyto funkce:

- Funkce `step()`, tato funkce implementuje časový krok v prostředí. Jedná se tedy o simulační krok. Také implementuje funkci odměny. Návratovou hodnotou této funkce jsou stav systému, odměna, příznak konce epizody a informace o kroku.
- Funkce `reset()`, tato funkce uvede prostředí do výchozího stavu.

Prostředí pro další učení bude také vytvořeno podle abstraktního modelu popsaného na straně 13. Simulovat se bude pouze tepelná výměna. Tepelné zdroje se nebudou simulovat. Teploty teplovodních okruhů se získají z naměřených hodnot.

Stavem prostředí budou teplotní rozdíly. Prvním bude rozdíl teplot okruhu topení a bojleru. Druhým bude rozdíl teplot okruhu solárních panelů a bojleru. Akcemi, které může agent provádět, budou vypnutí všech ohřevů, ohřev bojleru z teplovodního okruhu topení, ohřev bojleru z teplovodního okruhu solárních panelů.

4.5.4 Učení agenta

Na začátku agent dostane hlubokou neuronovou síť s náhodnými váhami přechodů a možné akce, které může vykonávat. Dále inicializuje paměť zkušeností.

Následuje algoritmus hlubokého Q-učení. Ten probíhá v epizodách. Na začátku každé epizody uvede prostředí do výchozího stavu pomocí metody prostředí `reset()`. Uvézt prostředí do výchozího stavu je nutné proto, aby nebylo ovlivněno žádnou akcí.

Následně provádí akce pomocí metody prostředí `step()`. Té předá jako parametr akci a dostane nový stav prostředí, odměnu, příznak konce epizody a informace o kroku. Akce během jedné epizody vybírá buď náhodně, nebo pomocí Q-hodnot z hluboké neuronové sítě. Pokud vybírá akci pomocí hluboké neuronové sítě, vybere tu, která má nejvyšší ohodnocení – Q-hodnotu.

Po provedení akce si uloží do paměti zkušeností stav před provedením akce, nový stav a získanou odměnu. Po provedení určitého počtu kroků, nebo uplynutí epizody agent učí hloubkovou neuronovou síť. Tu učí na základě dávek dat. Tyto dávky získá jako náhodný vzorek dat z paměti zkušeností.

Kapitola 5

Implementace

5.1 Simulační model

Simulační model pro systému byl implementován v Jazyce C++ pomocí knihovny Simlib [6]. Pro vygenerovaní Makefile byl použit nástoj CMake[1]. Části simulačního modelu jsou rozděleny do těchto modulů:

- `main.cpp` - hlavní modul, zpracování argumentů, spouštění simulace
- `environment.cpp` - tepelná výměnu mezi jednotlivými částmi systému
- `controller.cpp` - logika řídící jednotky
- `heatSources.cpp` - tepelné zdroje
- `generator.cpp` - instancuje třídy z ostatních modulů, které jsou využity pro simulaci
- `statistics.cpp` - statistiky průběhů hodnot simulace, tyto statistiky ukládá do souboru `output.csv`

Parametry simulace se dají měnit pomocí argumentů. Pro vizualizaci výstupu byl vytvořen skript v jazyce Python `generateGraphs.py`. Jako argument bere soubor se statistikami, který je výsledkem simulace.

5.2 Hluboké Q-učení

Učení agenta je rozděleno do dvou částí:

- Prvotní naučení na simulačním modelu
- Učení na naměřených datech

5.2.1 Prostředí

Implementované prostředí pro prvotní naučení agenta vychází z abstraktního modelu popsaného na straně 13. Byly ale pozměněny vlastnosti simulovaných tepelných zdrojů tak, že jejich výsledná hodnota výkonu poupravena o náhodné množství. Tím je zajištěn různý průběh těchto zdrojů a agent by se tak měl setkávat s větším počtem stavů.

[1] https://cmake.org/

Jednotlivé tepelné zdroje také nejsou pouštěny zároveň. Šance s jakou se rozhoduje, který se vybere, je 50%.

5.2.2 Agent

Agent je implementován pomocí třídy `DQNAgent` z knihovny `Keras-RL2`. Jako politiku využívá Boltzmannovu politiku - třída `BoltzmannQPolicy` z knihovny `Keras-RL2`. Dále využívá sekvenční paměť o velikosti 50000 – třída `SequentialMemory` z knihovny `Keras-RL2`. Před procesem učení projde 480 kroků pro prvotní naplnění paměti. Jako optimalizátor využívá optimalizátor Adam – třída `Adam` z knihovny `Keras`.

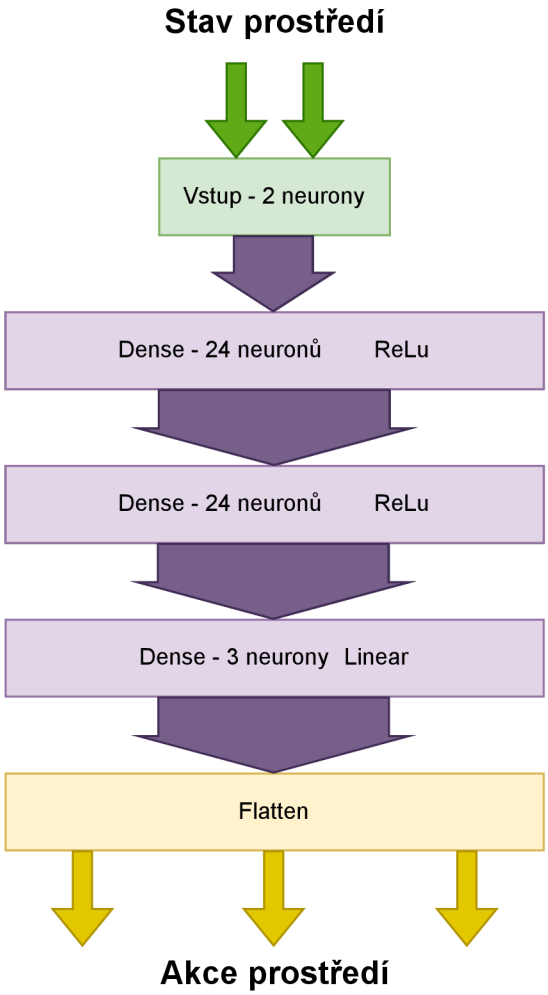
5.2.3 Neuronová síť

Pro implementaci neuronové sítě byla použita knihovna `Keras`. Z ní využívá sekvenční model – třída `Sequential`. Dále byla použita plně propojená vrstva – třída `Dense`, dále byla použita vrstva třídy `Flatten` jako výstupní vrstva, která slouží pro srovnání vstupních signálů vrstvy do jedné dimenze.

Vstupní vrstva obsahuje 2 neurony. Ty odpovídají stavu prostředí. Stav prostředí odpovídá teplotním rozdílům bojleru a teplovodních okruhů. Následují 2 plně propojené vrstvy s aktivační funkcí ReLU (viz strana 10), které obsahují 24 neuronů každá. Následuje skrytá, plně propojená vrstva s lineární aktivační funkcí(viz strana 10). Ta obsahuje 3 neurony. Poslední, výstupní vrstva má 3 neurony. Ty odpovídají 3 akcím, které může agent provádět.

Tyto akce jsou:

- akce 0 - vypnutí všech ohřevů
- akce 1 - ohřev bojleru z teplovodního okruhu topení
- akce 2 - ohřev bojleru z teplovodního okruhu solárních panelů



Obrázek 5.1: Schéma implementované neuronové sítě. Obsahuje 2 vstupní a 3 výstupní neurony. Dále 3 skryté vrstvy. Aktivační funkce jsou napsány u každé vrstvy. Vstupem je stav prostředí, tedy teplotní rozdíly bojleru a teplovodních okruhů. Výstupem jsou Q-hodnoty akcí prostředí, akce prostředí jsou popsány v předchozím odstavci.

5.2.4 Prvotní naučení

Prvotní naučení je implementováno v Python skriptu `initialLearning.py`.

Tento skript obsahuje třídu `ModelEnv`, která je potomkem třídy `Env` z knihovny `OpenAI Gym`. Ta implementuje prostředí popsané v návrhu na straně 20. Toto prostředí pro učení obsahuje náhodné výchozí hodnoty teplot, také je do simulace tepelných zdrojů přidán šum. Tím by měl model prostředí lépe odpovídat reálnému systému a agent by se měl naučit reagovat na širší rozsah vstupních dat.

Prvotní učení probíhá v 2.400.000 krocích, to odpovídá 10.000 kompletním simulačním cyklům. Tento počet cyklů se dá upravit parametrem skriptu. Po provedení učení se uloží získané váhy do souboru `neural_network.h5f`. Následně se získají nově nalezené prahy. Nově nalezené prahy se uloží ve formátu JSON[2] do souboru `thresholds.json`.

[2]JSON = JavaScript Object Notation je formát zápisu dat, který slouží primárně pro přenos dat.

5.2.5 Učení z naměřených dat

Pozdější učení agenta za provozu je implementováno v Python skriptu `continuousLearning.py`.

Tento skript pracuje se souborem s naměřenými daty. Cestu k tomuto souboru přijímá jako parametr při spuštění skriptu. Oproti prvotnímu naučení je třída `ModelEnv` upravena tak, že nesimuluje tepelné zdroje. Načte data do datagramu z knihovny `Pandas`, tyto data převzorkuje podle minut, za účelem simulace. V každém kroku se pak berou teploty teplovodních okruhů z datagramu a simuluje se pouze tepelná výměna.

Agent načte váhy neuronové sítě ze souboru `neural_network.h5f` a provede 24.000 kroků. Následně se přepíše soubor `neural_network.h5f` nově získanými váhami neuronové sítě. Poté se získají nově nalezené prahy. Nově nalezené prahy se uloží ve formátu JSON[3] do souboru `thresholds.json`.

5.2.6 Nalezení nových prahů

Nalezení nových prahů probírá pomocí naučené neuronové sítě. Nejprve se nalezne práh pro ohřev z teplovodního okruhu topení, poté práh pro ohřev z teplovodního okruhu solárních panelů.

Nalezení probíhá v cyklu. Začíná se s minimálním prahem $0,25$. Poté se pomocí metody `predict()` získají Q-hodnoty z neuronové sítě pro jednotlivé akce a zvyšuje práh o krok $1/128$ ($= 0,0078125$). Dokud nebude Q-hodnota akce pro zapnutí ohřevu vetší, než Q-hodnota pro akci vypnutí ohřevu, nebo dokud práh nebude roven 5.

Minimální práh tedy může být $0,25°C$ a maximální $5°C$. Optimalizované prahy jsou tak v intervalu $\langle 0.25, 5 \rangle$

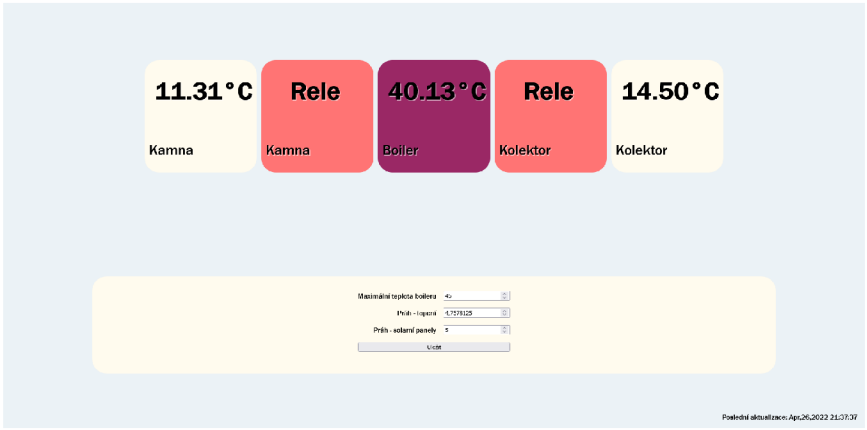
5.3 Centrální jednotka

Centrální jednotka používá Web server Apache pro poskytnutí webového rozhraní. Pro implementaci jsou použity PHP skripty a Python skripty. A je implementovaná na Raspberry pi 400.

5.3.1 Klientská část

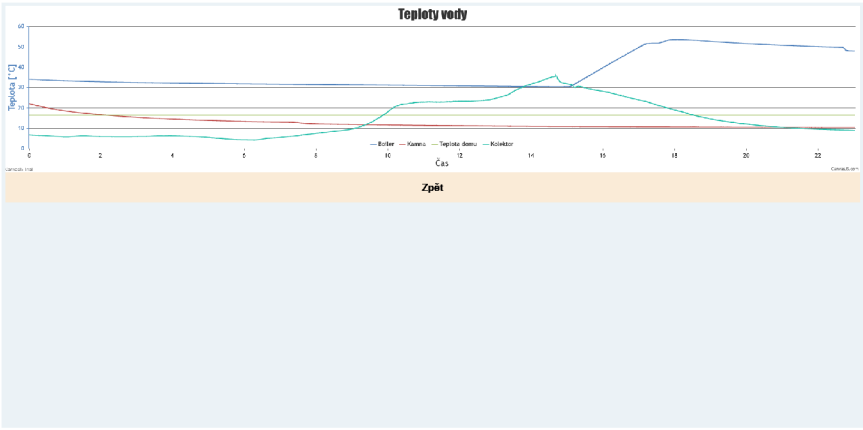
Uživatelské rozhraní bylo implementováno za použití HTML a CSS. Pro komunikaci se serverem využívá asynchroní javaskript. Ten jednou za 10 sekund získává data ze serveru. Umožňuje také nastavit maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení.

[3] JSON = JavaScript Object Notation je formát zápisu dat, který složí primárně pro přenos dat.



(a) Hlavní stránka - počítače

(b) Hlavní stránka - mobilní zařízení



(c) Stránka s průběhem teplot

Obrázek 5.2: Navržené uživatelské prostředí řídícího sytému. Hlavní stránka obsahuje všechny naměřené hodnoty a stav jednotlivých ohřevů z externích zdrojů. Další stránka obsahuje graf průběhu teplot během aktuálního dne.

5.3.2 Serverová část

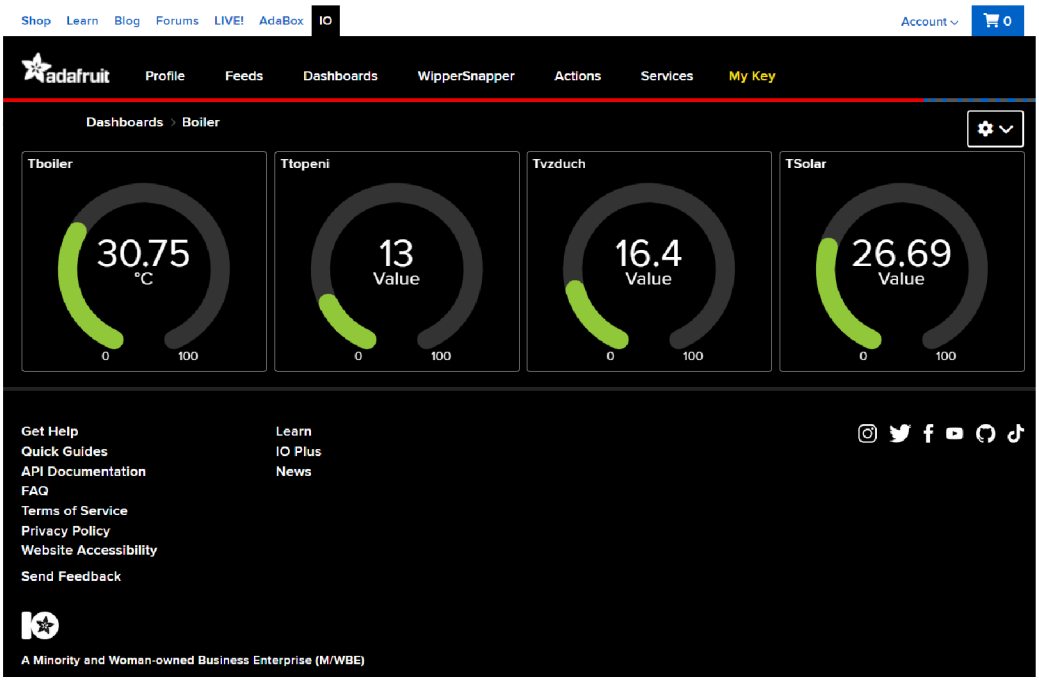
Pro implementaci serverové části aplikace byly použity tyto PHP skripty:

- `writefile.php` – Rozhraní pro ESP-8266, přes GET dotaz získá data od ESP-8266, která uloží do souboru `datastorage.txt` a `stats/datum.txt`, kde `datum` je aktuální datum a tento soubor slouží jako historie naměřených teplot. V odpovědi zašle pomocí JSON maximální teplotu na kterou se může bojler ohřívat z teplovodního okruhu topení, aktuální čas a prahy pro spouštění ohřevu. Také spouští Python skript `adafruitIO.py`.
- `getData.php` – Rozhraní pro klienta, které získá data ze souborů `datastorage.txt` a `config.txt`, spojí je a pomocí JSON odešle klientovi.

- `changeSettings.php` – Rozhraní pro klienta, které získá maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení a prahy pro ohřev bojleru od klienta. Následně tyto hodnoty uloží do souboru `config.txt`.
- `graph.php` – Skript, který odešle klientovi html stránku s grafem teplot v daném dni, využívá knihovnu `Canvasjs`.

Dále byly použity tyto skripty v jazyce Python:

- `getRoomTemp.py` – Skript, který získává teplotu místnosti. Je nutné, aby běžel na pozadí, neboť potřebuje práva superuživatele, protože přistupuje k hardwarovým pinům Raspberry pi 400.
- `adafruitIO.py` – Skript, který pošle naměřená data na server. https://io.adafruit.com/. Tento skript využívá knihovny `Adafruit_IO` v jazyce Python. Adafruit IO platforma pro vizualizaci dat a interakci s nimi pro IoT projekty. Především je využívána pro dostupnost naměřených dat mimo lokální síť.



Obrázek 5.3: Snímek webu Adafruit IO s naměřenými hodnotami.

5.4 Mikrokontrolery ESP-8266 a Arduino nano

Kód pro oba mikrokontrolery byly vytvořeny, zkompilovány a nahrány do mikrokontrolerů v prostředí Arduino. Pro knihovny a podporu ESP-8266 byly použity správci desek z adresy http://arduino.esp8266.com/stable/package_esp8266com_index.json. Všechny použité knihovny byly získány z manažera knihoven ve vývojovém prostředí Arduino[4]

[4] https://www.arduino.cc/

5.4.1 ESP-8266

Pro implementaci kódu pro mikrokontroler ESP-8266 byly použity knihovny:

- `ESP8266WiFi` - knihovna pro podporu Wi-Fi
- `ESP8266HTTPClient` - knihovna pro podporu http klienta
- `WiFiClient` - knihovna pro podporu http klienta přes Wi-Fi
- `ArduinoJson` - knihovna pro práci s JSON formátem
- `Adafruit_GFX` - knihovna pro podporu oled displeje
- `Adafruit_SSD1306` - knihovna pro podporu oled displeje

Po zapnutí, mikrokontroler inicializuje sériovou linku, pro komunikaci s Arduino nano. Poté inicializuje spojení s oled displejem a připojí se k Wi-Fi. Následuje smyčka mikrokontroleru, kde volá funkce `SerialGetData()`, `GetHTTP()` a `DisplayInfo()`.

- **SerialGetData()** příjme data z mikrokontroleru Arduino nano přes sériovou linku. Data si předávají ve formátu JSON. Následně data načte do příslušných proměnných.
- **GetHTTP()** vezme naměřená data, získaná od mikrokontroleru Arduino nano a pošle je na Centrální jednotku – Raspberry pi. K tomu využije HTTP protokol s metodou GET. V odpovědi dostane data ve formátu json. Následně zavolá funkci `SerialSendData(HTTPClient& http)`, které předá objekt klienta, který obsahuje získaná data.
  - `SerialSendData(HTTPClient& http)` získá z klienta data ve fomátu JSON. Z těch získá čas a maximální hodnotu na kterou se může bojler ohřívat z teplovodního okruhu topení a jednotlivé prahy. Následně zašle data mikrokontroleru Arduino nano.
- **DisplayInfo()** zobrazí data na dipleji. Data které zobrazuje jsou:
  - Teplota bojleru
  - Teplota teplovodního okruhu topení
  - Teplota teplovodního solárních panelů
  - Čas
  - Stav zapnutí dohřívání bojleru vestavěným elektrickým ohříváním
  - Maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení
  - Prahy pro ohřev bojleru z externích zdrojů

5.4.2 Arduino nano

Pro implementaci kódu pro mikrokontroler Arduino nano byly použity knihovny:

- `OneWire` - knihovna pro komunikaci s teplotními čidly
- `DallasTemperature` - knihovna pro získávání naměřených teplot z teplotních čidel
- `ArduinoJson` - knihovna pro práci s JSON formátem

Po zapnutí, mikrokontroler inicializuje sériovou linku, pro komunikaci s ESP-8266. Poté nastaví výstupní piny a naváže spojení s teplotními čidly a získá naměřené teploty. Následuje smyčka mikrokontroleru, kde volá funkce `GetSensorData()`, `ControlLogic()`, `SerialSendData()` a `SerialGetData()`.

- **GetSensorData()** získá naměřená data z teplotních senzorů
- **ControlLogic()** implementuje řídící logiku.
- **SerialSendData()** vezme naměřená data a stav zapnutí ohřevu z jednotlivých zdrojů a pošle je ve formátu JSON mikrokontroleru ESP-8266 přes sériovou linku.
- **SerialGetData()** získá data od mikrokontroleru ESP-8266 ve formátu JSON ze sériové linky. Konkrétně:
  - Čas - pro zapínání ohřevu bojleru z vestavěného elektrického ohřívání.
  - Maximální teplota, na kterou se může bojler ohřívat z teplovodního okruhu topení
  - Práh pro zapnutí ohřevu bojleru z teplovodního okruhu topení
  - Práh pro zapnutí ohřevu bojleru z teplovodního okruhu solárních panelů

Řídící logika mikrokontroleru funguje na principu výše zmiňovaných prahů. Pokud je jeden z externích zdrojů teplejší o daný práh, než teplota bojleru, zapne se ohřívání bojleru z externího zdroje, dokud se jejich teploty nevyrovnají. Z uvedených externích zdrojů je prioritní ohřev z teplovodního okruhu solárních panelů, neboť je nejméně nákladný. Pokud není v časovém rozmezí 15:00 až 19:00 bojler ohřátý na Maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení, spustí se ohřívání vestavěným elektrickým zdrojem. Toto časové rozmezí je stanoveno podle levnějšího tarifu elektřiny tzv. „nočního proudu“.

Kapitola 6

Zhodnocení výsledků

6.1 Optimalizované prahy

Výsledkem prvotního učení byly prahy:

Práh t. o. topení	Práh t. o. solárních panelů
0,25	5

Tabulka 6.1: Výsledné prahy po prvotním učení na simulačním modelu

Tyto prahy odpovídaly krajním hodnotám, takže nejsou vhodné pro reálné nasazení. Příčinou těchto výsledků je nejspíše aproximace průběhu výkonu tepelných zdrojů. Ty dostatečně neodpovídaly reálnému chování tepelných zdrojů. Proto se agent ještě naučil na naměřených datech.

Výsledkem učení z dat z provozu byly tyto hodnoty prahů:

Práh t. o. topení	Práh t. o. solárních panelů
4,7578125	5.0

Tabulka 6.2: Výsledné prahy po učení na naměřených datech z reálného provozu.

Z těchto výsledků můžeme vyvodit, že se agent částečně chová podle očekávání. Tedy práh pro ohřev z teplovodního okruhu se zmenšil, takže dochází k zapnutí ohřevu o něco dříve.

Výsledné prahy byly ověřeny pomocí simulačních experimentů.

6.2 Simulační experimenty

Pro simulační experimenty byl použit stejný simulační model, jako pro ověření validity modelu viz strana 13. V simulačních experimentech byly použity hodnoty prahů získané z hlubokého Q-učení, původní hodnoty prahů a ručně zvolené prahy. Cílem experimentů je ověřit, zda výsledky nově nalezených prahů pomocí hlubokého Q-učení odpovídají očekávání.

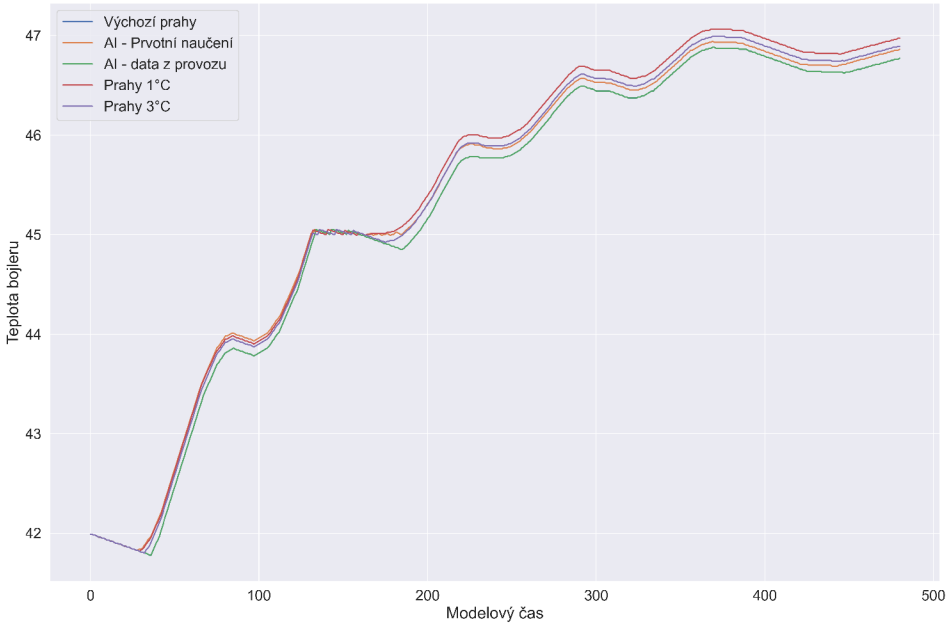
Přehled výsledků simulačních experimentů:

Typ	$\mathbf{P_t}$	$\mathbf{P_{sp}}$	$\mathbf{T_{max}}$	**Čas**	**N**	**M**
Původní prahy	5	5	46,8761	369	3	1
AI – Prvotní naučení	0.25	5	46,9353	369	9	1
AI – data z provozu	4,7578125	5	46,8761	369	3	1
Ručně zvoleno	3	3	46,9901	370	4	1
Ručně zvoleno	1	1	47,0636	369	5	1

Tabulka 6.3: Výsledky simulačních experimentů. P_t odpovídá prahu, při kterém se zapne ohřev z teplovodního okruhu topení. P_{sp} odpovídá prahu, při kterém se zapne ohřev z teplovodního okruhu solárních panelů. T_{max} odpovídá maximální teplotě bojleru, na kterou se během simulace ohřál. Čas je modelový čas dovršení maximální teploty bojleru. N udává počet zapnutí dohřívání z teplovodního okruhu topení a M počet zapnutí dohřívání z teplovodního solárních panelů

Z provedených experimentů lze pozorovat, že výsledky agenta odpovídají očekávání. Po prvotním učení neuronová síť zapínala ohřev bojleru z t. o. topení už při prahu $0.25°C$ a tím zvýšila maximální teplotu bojleru. Můžeme ale pozorovat, že práh $0.25°C$ vedl k častému zapínání a vypínání ohřevu, což není žádoucí. Po dalším naučení na reálných datech neuronová síť zapínala ohřev bojleru z t. o. topení při nově nalezeném prahu $4,7578125°C$. Tento práh v simulaci nedokázal zvýšit maximální teplotu bojleru, ani urychlit dosažení této teploty, ale ani nezvýšil počty zapnutí ohřevu. V reálném nasazení by tento výsledek nejspíše vedl k optimálnějšímu chování. Výsledky experimentů s ručně zvolenými prahy ukazují, že nižší hodnoty prahů vedou k vyšší maximální teplotě bojleru, ale také dochází k častějšímu zapínání a vypínání ohřevu.

Z provedených experimentů můžeme usoudit, že nově nalezené hodnoty prahů (4,7578125 a 5) od naučeného agenta jsou optimálnější, než původní prahy.



Obrázek 6.1: Graf porovnání průběhu teplot bojleru v simulačních experimentech

Kapitola 7

Závěr

V této práci jsem navrhl a popsal realizaci řídícího systému pro ohřev vody v bojleru z externích zdrojů. Pro nalezení optimálnějších prahů pro řízení využití těchto externích zdrojů jsem využil posilované učení, konkretně algoritmus Hlubokého Q-učení.

Nejprve jsem analyzoval problematiku řídících systému a IoT pro chytré domácnosti a popsal strojové učení, posilované učení a algoritmus Q-učení a Hlubokého Q-učení. Dále jsem popsal umělé neuronové sítě, umělý neuron a simulace.

Následně jsem analyzoval systém ohřevu vody v mém rodinném domě. Podle zjištěných vlastností jsem navrhl abstraktní model tohoto systému. Pomocí tohoto modelu jsem vytvořil prostředí pro agenta hlubokého Q-učení. Agenta hlubokého Q-učení jsem nejprve naučil na simulačním modelu a následně jsem vytvořil prostředí pro jeho další učení na naměřených datech.

Navrhl jsem řešení nového řídícího systému, který umožňuje měnit hranice zapínání ohřevu vody v bojleru a je realizován prvky na bázi SoCs Espressif a Raspberry Pi. U řídícího systému jsem také navrhl uživatelské rozhraní, aby naměřená data bylo možné vizualizovat a řídit ohřev bojleru z teplovodního okruhu topení domu.

Nově získané parametry pro řízení ohřevu vody v bojleru, získané z algoritmu hlubokého Q-učení, jsem ověřil pomocí simulačních experimentů. Po ověření chování systému s novými parametry, jsem tento navržený systém realizoval a nasadil do provozu.

Výstupem práce je realizovaný řídící systém ohřevu vody v bojleru a skripty v jazyce Python implementující hloubkové Q-učení. Další výstup je simulační model tohoto systému.

Pro další vývoj a docílení lepší možné optimalizace by bylo využití výkonnějšího hardware místo Arduino nano pro samostatné řízení a implementovat strojové učení přímo na tomto hardware, místo aktuální řídící logiky. Systém by tak mohl ještě více optimalizovat využití externích zdrojů.

Literatura

[1] Brychta, A. *POSILOVANÉ UČENÍ PRO HRANÍ ROBOTICKÉHO FOTBALU.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc.RNDr. Pavel Smrž, Ph.D.

[2] IBAgroup. *Internet věcí* [online]. 2021 [cit. 30.12.2021]. Dostupné z: https://ibacz.eu/trendy/internet-veci/.

[3] McNally, T. *Keras-rl2* [online]. GitHub, 2016. Dostupné z: https://github.com/taylormcnally/keras-rl2.

[4] Moolayil, J. *Learn Keras for Deep Neural Networks.* 1. vyd. Apress, 2019. ISBN 978-1-4842-4240-7.

[5] Oracle. *What is Machine Learning?* [online]. 2022 [cit. 15.4.2022]. Dostupné z: https://www.oracle.com/cz/data-science/machine-learning/what-is-machine-learning/.

[6] Peringer, P. *SIMLIB/C++* [online]. 2021 [cit. 30.12.2021]. Dostupné z: https://www.fit.vutbr.cz/~peringer/SIMLIB/.

[7] Petr Peringer, M. H. *Modelovaní a simulace* [online]. 2021 [cit. 16.4.2022]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIMS-IT%2Flectures%2FIMS-2021-09-20.pdf&cid=14664.

[8] Rascasone. *INTERNET VĚCÍ (IOT): DEFINICE, PŘÍKLADY VYUŽITÍ, PRODUKTY* [online]. 2020 [cit. 30.12.2021]. Dostupné z: https://www.rascasone.com/cs/blog/iot-internet-veci-definice-produkty-historie.

[9] Richard S. Sutton, A. G. B. *Reinforcement Learning: An Introduction* [online]. 1. vyd. The MIT Press, 2018 [cit. 19.4.2022]. ISBN 978-1-4842-4240-7. Dostupné z: http://www.incompleteideas.net/book/RLbook2020.pdf.

[10] SMEJKAL, J. *Návrh a realizace IoT pro monitorování a řízení chytré domácnosti.* [online]. Brno, 2021. [cit. 30.12.2021]. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis-file/23957/23957.pdf.

[11] Wikipedia. *Strojové učení* [online]. 2021 [cit. 30.12.2021]. Dostupné z: https://cs.wikipedia.org/wiki/Strojov%C3%A9_u%C4%8Den%C3%AD.

[12] Wikipedia. *Teplo* [online]. 2021 [cit. 11.12.2021]. Dostupné z: https://cs.wikipedia.org/wiki/Teplo.

Příloha A

Obsah přiloženého paměťového média

- xtomec09.zip
  - xtomec09.pdf – PDF soubor této práce
  - Doc – zdrojové soubory práce práce
  - SimModel – zdrojové soubory simulačního modelu, včetně knihovny simlib
    - README.txt – manuál
  - Posilovane_uceni – zdrojové soubory posilovaného učení
    - README.txt – manuál
  - Ridici_system – zdrojové soubory řídicího systému
    - raspberry_pi – zdrojové soubory centrální jednotky
    - arduino_nano – zdrojove soubory pro Arduino nano
    - esp – zdrojove soubory pro ESP8266
    - README.txt – manuál

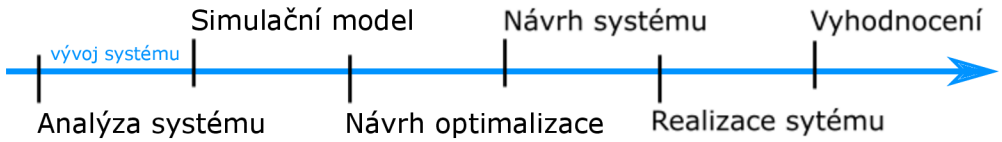
Příloha B

Plakát

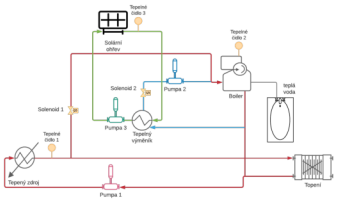
NÁVRH SYSTÉMU ŘÍZENÍ DISTRIBUCE TEPLA

Jan Tomeček
xtomec09

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

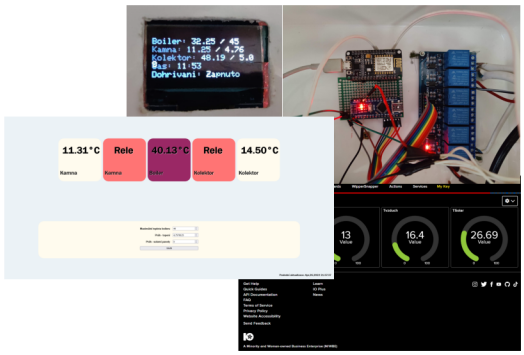


Anyláza

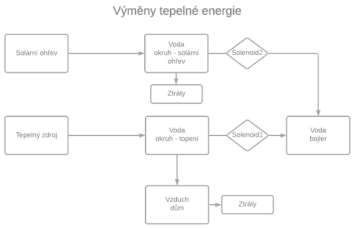


Návrh systému

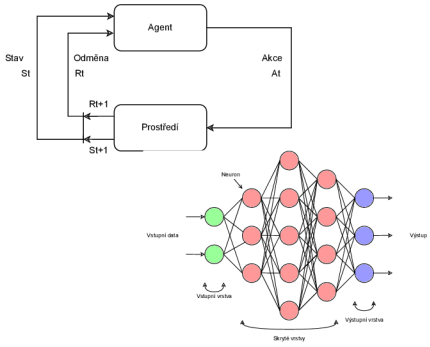
Realizace systému



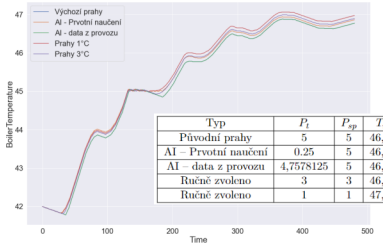
Simulační model



Návrh optimalizace



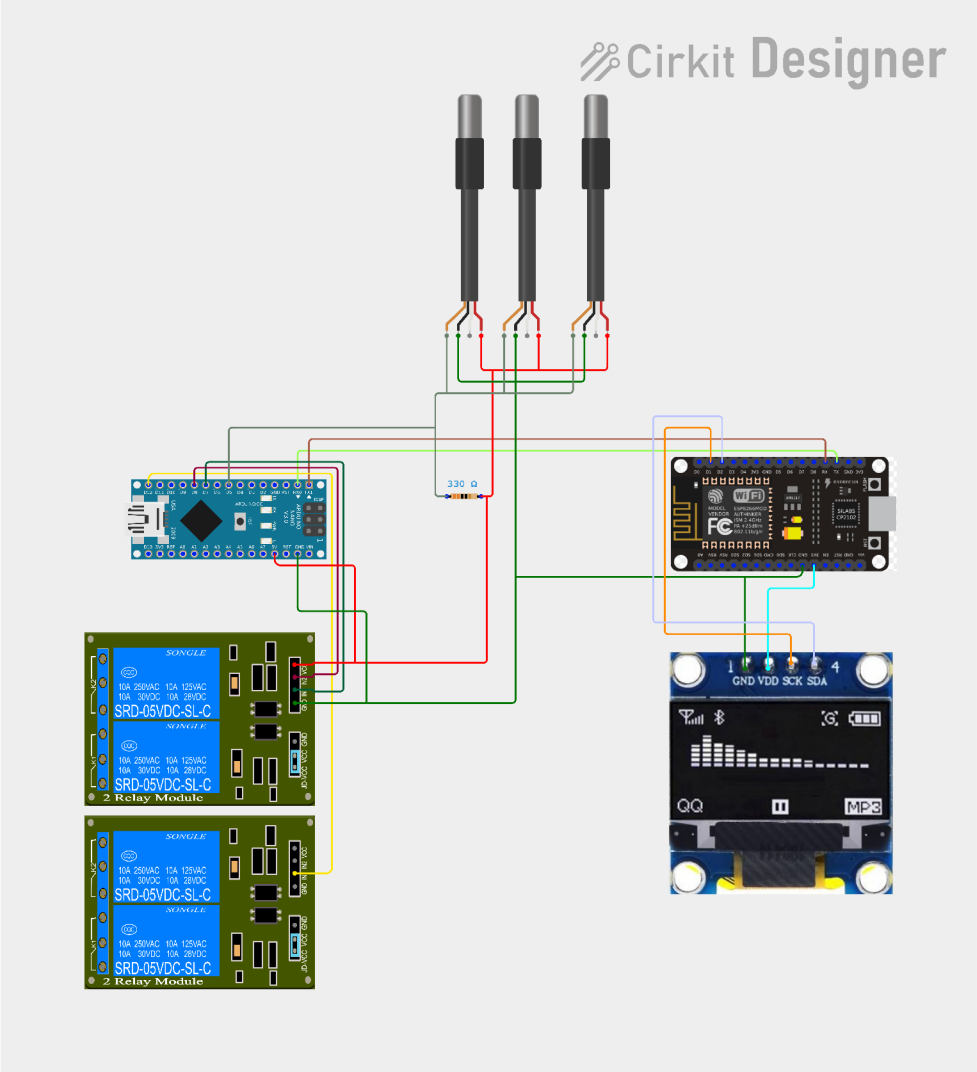
Vyhodnocení optimalizace



Typ	P_i	P_{vp}	T_{max}	Čas	N	M
Původní prahy	5	5	46,8761	369	3	1
AI – Prvotní naučení	0.25	5	46,9353	369	9	1
AI – data z provozu	4,7578125	5	46,8761	369	3	1
Ručně zvoleno	3	3	46,9901	370	4	1
Ručně zvoleno	1	1	47,0636	369	5	1

Příloha C

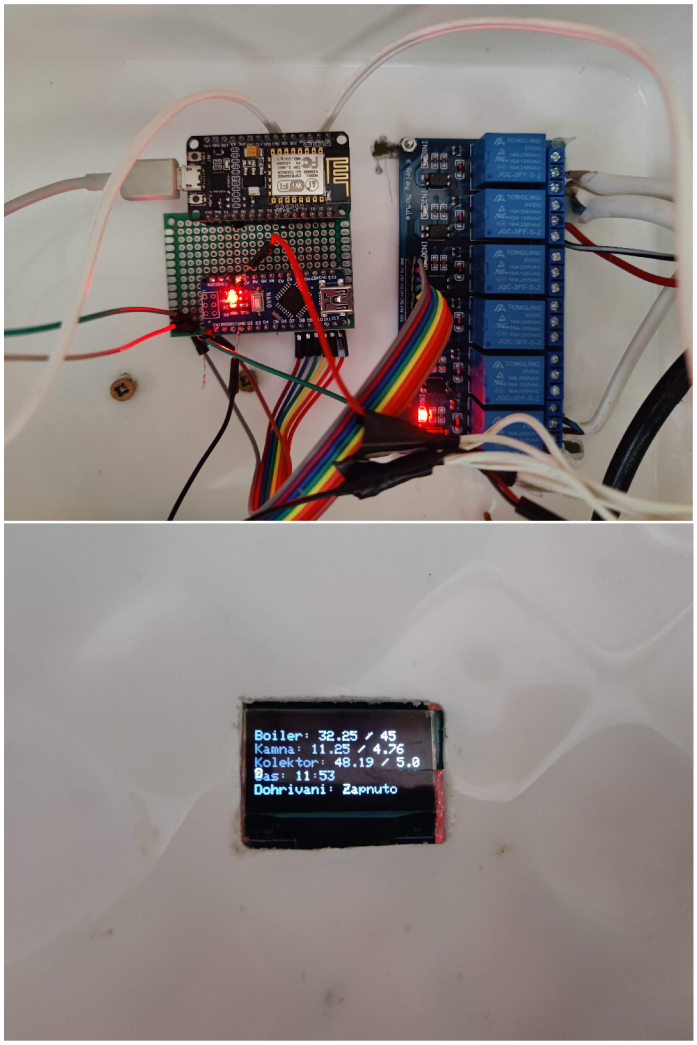
Schéma zapojení mikrokontrolerů



Obrázek C.1: Schéma zapojení komponentů. Schéma bylo vytvořeno pomocí nástroje Cirkit Designer https://www.cirkitstudio.com/

Příloha D

Fotografie realizovaného systému



Obrázek D.1: Fotografie části realizovaného systému. Jde o část s mikrokontrolery Arduino nano a ESP-8266.