
**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
25. července 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, který bude využívat optimálně externí zdroje pro ohřev bojleru, je potřebný simulační model tohoto systému. Optimalizace jsou navrhnuty pomocí tohoto 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 jednotlivé části navrženého systému. V páté kapitole je popis implementace navrženého systému a použitých nástrojů, 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 [13].

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*“ [3].

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*“ [11].

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 [14].
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 [8].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. [8]

## 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 [12].

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 optimální akce pro dané stavy prostředí, které nemusí být zřejmé na první dojem [12].

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 [12]

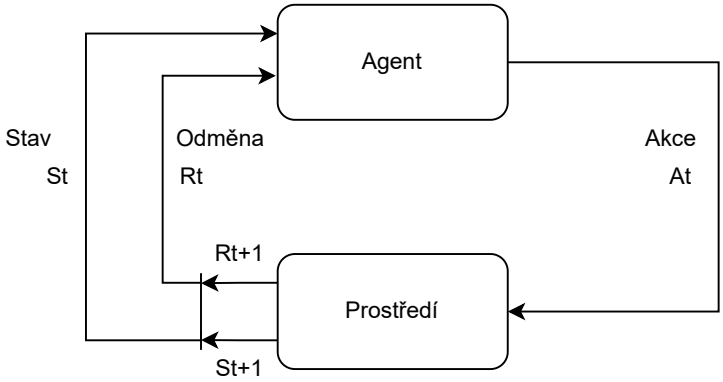
- **Signál odměny** – Cíl posilovaného učení. Agent obdrží odměnu za každou provedenou akci v prostředí. Agentův úkolem je maximalizovat tuto hodnotu v delším běhu. Na základě této odměny agent mění svoji strategii, aby se rozhodl, pokud se dostane do dalšího stavu, pomocí jiné akce [12]

- **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 [12]
- **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 [12]

### 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)“* [10].

Markovské rozhodovací procesy jsou matematická forma procesu posilovaného učení. Tedy Agent může v daném stavu $S_t$ vybrat akci $A_t$, dostupnou v daném stavu. Touto akcí se dostane do nového stavu $S_t + 1$ a obdrží odměnu $R_t + 1$ [12].




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 $R_t + 1$ [12].

**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 [2].

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
$\gamma$ = diskontní faktor
$\alpha$ = 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

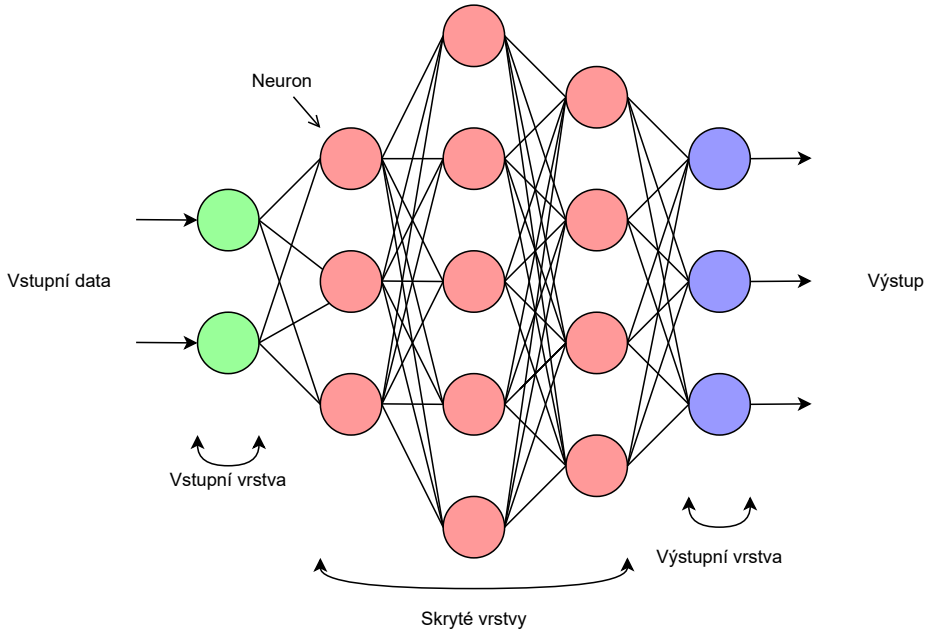
Agent nejprve zkoumá Q-hodnoty jednotlivých akcí, pokud by neprozkoumal dostatek akcí, mohl by zvolit neoptimální strategii. Proto během procesu učení vybírá i náhodné akce s pravděpodobností $\epsilon, \epsilon \in (0,1\rangle$. Epsilon se během učení snižuje [2].

**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 [5].

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.

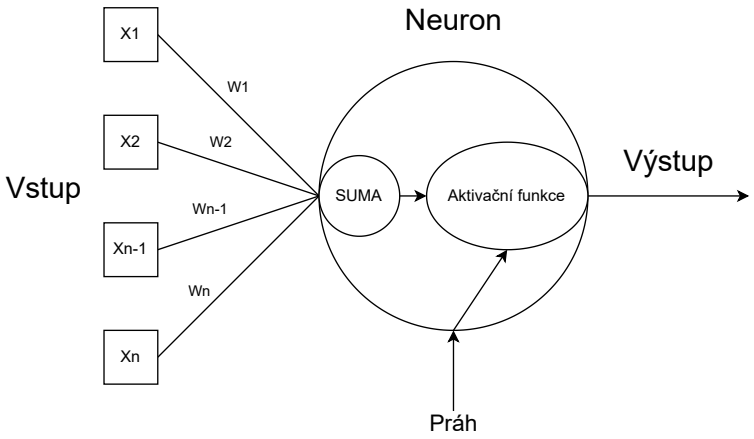



Obrázek 2.2: Ilustrace umělé neuronové sítě. Má celkem 5 vrstev. Jde o plně propojenou síť – každý neuron je propojen se všemi neurony v další vrstvě. Vstupní vrstva má 2 vstupy, následují 3 skryté vrstvy s různým počtem neuronů(3,5,4) a výstupní vrstva má 3 výstupy.

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 [5].

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 [5].

Nejpoužívanějšími funkcemi jsou Sigmoid a ReLu.

#### Lineární aktivační funkce

Lineární aktivační funkce je definována rovnicí:

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

kde: $x$ = vstup neuronu

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

#### Sigmoid

Sigmoid aktivační funkce je definována 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 popsána 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 je popsán návrh simulačního modelu, který je využíván pro optimalizaci využití externích zdrojů pro ohřev bojleru. Jelikož je cílem optimalizace maximální využití externích zdrojů pro ohřev bojleru, je simulace ohřevu bojleru z externích zdrojů nejdůležitější částí simulačního modelu.

Nejprve jsou 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 [9].

## 3.1 Simulace

Základní pojmy spojené se simulacemi:

- „***Simulace*** *je získávání nových znalostí o systému experimentováním s jeho modelem*“ [10].
- **Abstraktní model** je zjednodušený popis zkoumaného systému na základě jeho vlastností [10].
- **Simulační model** je implementace abstraktního modelu v programovacím jazyce [10].

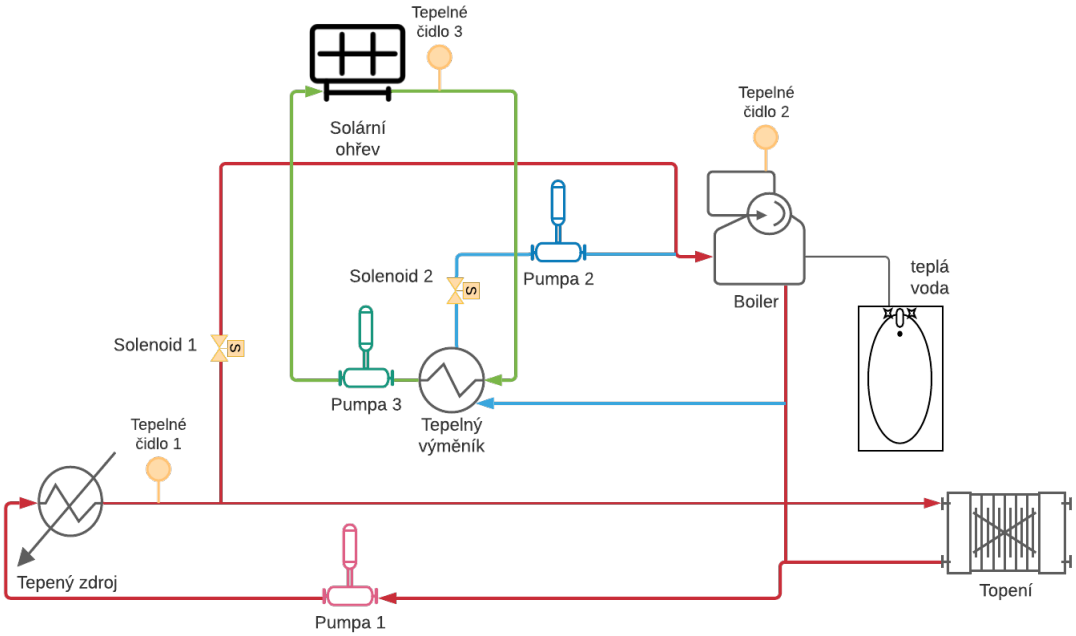
Proces vytvoření simulačního modelu tedy zahrnuje:

- 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




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

Bojler ohřívá užitkovou vodu a obsahuje přibližně 200 kg vody. Je dohříván přes den a užitková voda z něj je využívána primárně večer. Bojler není dokonale izolován a tak se v něm voda sama ochlazuje.

Teplovodní okruh vytápění rodinného domu je dohříván z kamen na dřevo. Má objem obsahuje přibližně 200 kg vody. Do kamen na dřevo se musí přikládat jednou za 60–90 min. Topení ohřívá v domě 1458 kg vzduchu. Dům také není dokonale odizolován od venkovního vzduchu a tak se ochlazuje.

Teplovodní okruh solárních panelů je ohříván maximálně 6 hodin přes den, protože jsou solární panely částečně zastíněny rodinným domem. Okruh obsahuje přibližně 70 kg vody.

Ohřev bojleru z externích zdrojů je ovládán pomocí sepnutí:

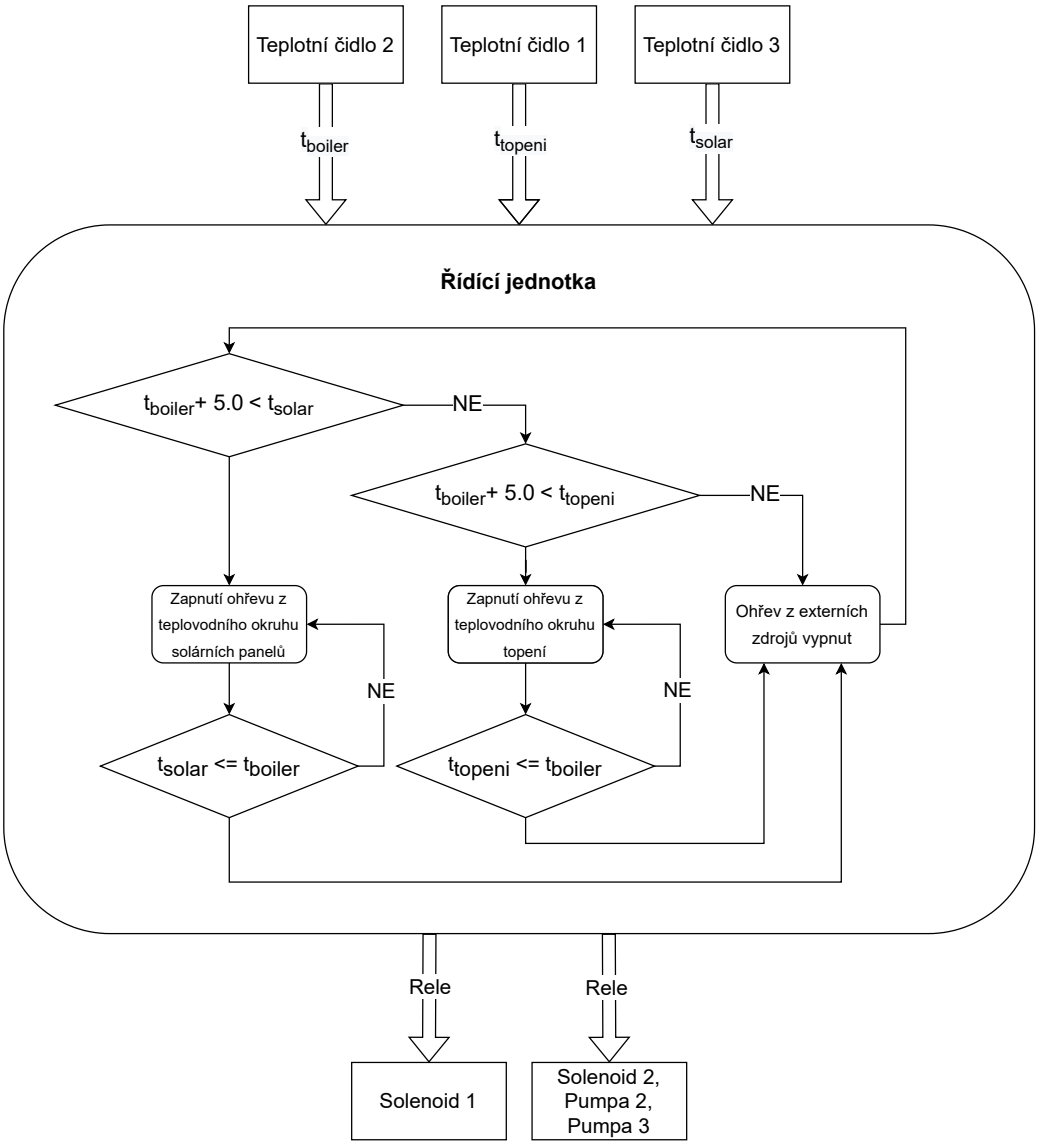
- Solenoidu 1 - ohřev z teplovodního okruhu topení
- Solenoidu 2, pumpy 2 a pumpy 3 - ohřev z teplovodního okruhu solárních panelů

Pumpa 1 je trvale zapnutá, protože zajišťuje oběh vody v okruhu topení. Voda v tomto oběhu musí neustále cirkulovat pro zajištění distribuce tepla.

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.

Spouštění ohřevu bojleru z externích zdrojů probíhá na základě pevně stanoveného teplotního rozdílu $5°C$. Tedy pokud je jeden z externích zdrojů teplejší o $5°C$ spustí se ohřev bojleru až dokud se teploty nevyrovnají.

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




Obrázek 3.2: Schéma řídící jednotky s diagramem její řídící logiky. Řídící jednotka využívá pevně stanovený rozdíl teplot - $5°C$, pomocí kterého zapíná ohřev bojleru z externích zdrojů.

## 3.3 Abstraktní model

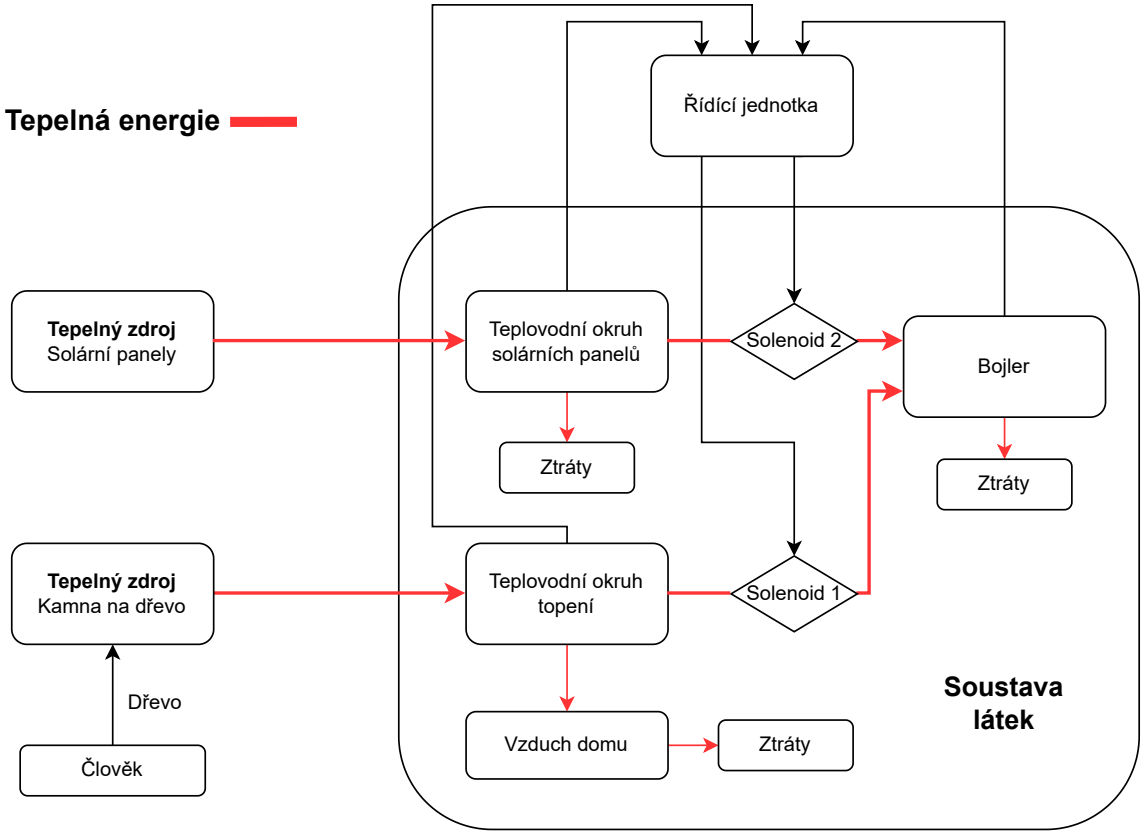
Abstraktní model byl vytvořen na základě schématu zapojení systému viz Obrázek 3.1 s ohledem na požadavky pro optimalizaci.

Pro optimalizaci využití externích zdrojů pro ohřev bojleru je důležitá hlavně tepelná výměna mezi tepelnými zdroji a bojlerem, proto byly ostatní části systému zjednodušeny.

Tepelné zdroje byly také zjednodušeny a jejich chování a parametry průběhu výkonu byly experimentálně odvozeny.

Abstraktní model je navržen jako diskrétní a jeden časový krok odpovídá 1 min reálného času.

Na základě uvedeného abstraktního modelu, byl pomocí knihovny Simlib [9] vytvořen diskrétní simulační model pro ověření validity modelu.




Obrázek 3.3: Schéma abstraktního modelu. Skládá se z 3 hlavních částí. Tepelné zdroje, řídící jednotka, soustava látek, mezi kterými probíhá tepelná výměna.

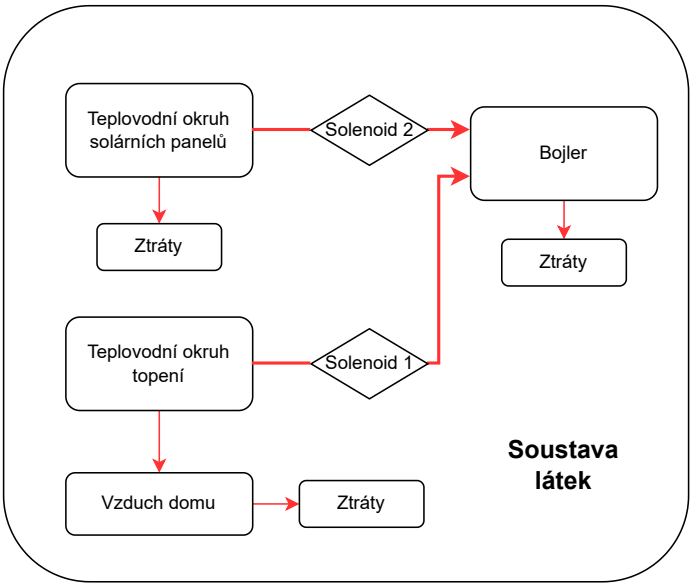
### 3.3.1 Soustava látek

Soustava látek, mezi kterými probíhá tepelná výměna, je hlavní část abstraktního modelu. Reprezentuje jednotlivé teplovodní okruhy, bojler a vzduch domu jako masy látek. U látek pozorujeme jejich teploty. Tyto teploty se mění na základě energie, kterou si látky předávají.

| Část modelu | Látka | Hmotnost | Měrná tepelná kapacita |
|---|---|---|---|
| Bojler | Voda | 200 kg | 4180 |
| Teplovodní okruh solárních panelů | Voda | 70 kg | 4180 |
| Teplovodní okruh topení | Voda | 200 kg | 4180 |
| Vzduch domu | Vzduch | 1458 kg | 1003 |

Tabulka 3.1: Tabulka vlastností částí abstraktního modelu. Měrné tepelné kapacity byly převzaty ze zdroje Molekulová fyzika a termika [6]

#### Celá soustava látek




Obrázek 3.4: Schéma části abstraktního modelu, popisující soustavu látek.

Celá soustava látek si tedy předává energii. Jelikož přenosy energií nejsou dokonalé, přenáší se jen část energie z jedné látky do druhé.

**Přehled přenosů energie a účinností přenosu:**

| Zdroj | Příjemce | Účinnost |
|---|---|---|
| teplovodní okruh solárních panelů | bojler | 2,0 % |
| teplovodní okruh topení | bojler | 1,5 % |
| teplovodní okruh topení | vzduch domu | 1,0 % |

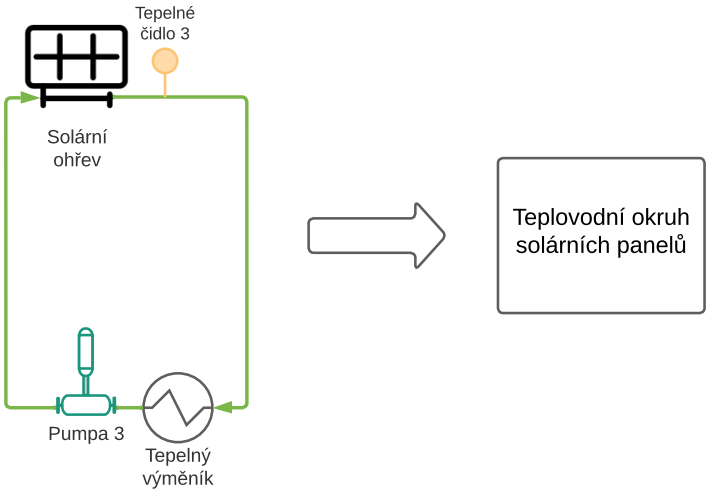
Tabulka 3.2: Přehled přenosů energie a účinností přenosu mezi jednotlivými částmi abstraktního modelu. Účinnosti přenosu energie mezi bojlerem a externími zdroji byly zjištěny pomocí simulačních experimentů. Simulační experimenty jsou popsány na straně 26. Účinnost přenosu mezi okruhem topení a vzduchem domu byla odhadnuta na základě průběhu simulovaných teplot a zkušeností teplot v domě v reálném provozu. Konkrétní hodnoty teplot vzduchu v domě nejsou pro účely optimalizace ohřevu bojleru důležité, ale vzduch domu slouží pouze jako médium, které ochlazuje teplovodní okruh topení.

**Bojler**
Vlastnosti:

- Bojler je reprezentován jako jedna masa vody o hmotnosti 200 kg.
- Energii získává buď z teplovodního okruhu topení, nebo teplovodního okruhu solárních panelů.
- Časem v něm voda sama ztrácí energii.

**Teplovodní okruh solárních panelů**
Vlastnosti:

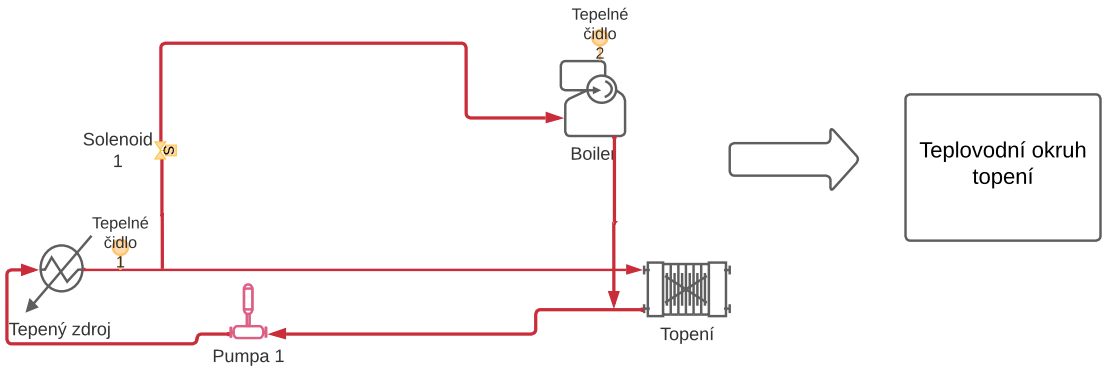
- Teplovodní okruh solárních panelů je reprezentován jako jedna masa vody o hmotnosti 70 kg.
- Energii získává od solárních panelů.
- Časem ztrácí energii
- Pokud je zapnutý ohřev bojleru z toho externího zdroje, vyměňuje si energii s bojlerem




Obrázek 3.5: Schéma ukazující vztah mezi reálným teplovodním okruhem solárních panelů a jeho reprezentací v abstraktním modelu

**Teplovodní okruh vytápění domu**
Vlastnosti:

- Teplovodní okruh vytápění domu je reprezentován jako jedna masa vody o hmotnosti 200 kg.
- Energii získává od kamen na dřevo.
- Vyměňuje si energii se vzduchem domu.
- Pokud je zapnutý ohřev bojleru z toho externího zdroje, vyměňuje si energii s bojlerem.




Obrázek 3.6: schéma ukazující vztah mezi reálným teplovodním okruhem vytápění domu a jeho reprezentací v abstraktním modelu

**Vzduch domu**
Vlastnosti:

- Vzduch domu je reprezentován jako jedna masa vzduchu o hmotnosti 1458 kg.
- Energii získává z teplovodního okruhu topení.
- Ztrácí energii, pokud je teplejší, jak venkovní vzduch.

### 3.3.2 Použité fyzikální veličiny a vztahy

**Teplo**

„Teplo je fyzikální veličina, která souvisí s vnitřní energií tělesa, charakterizuje děj, který probíhá mezi tělesy při tepelné výměně.“[6]

Rovnice pro výpočet tepla[6]:

$$Q = m * c * (t_2 - t_1) \tag{3.1}$$

kde: $Q$ = přijaté nebo odevzdané teplo látkou
$m$ = hmotnost látky
$c$ = měrná tepelná kapacita látky
$t_1$ = počáteční teplota látky
$t_2$ = konečná teplota látky

**Tepelná výměna**
Při dokonalé tepelné výměně platí:

$$Q_1 = Q_2 \tag{3.2}$$

Neboli

„Množství tepla, které přijalo chladnější těleso od tělesa teplejšího je stejné jako množství tepla, které teplejší těleso odevzdalo chladnějšímu.“[7]

Po dosazení vztahů z rovnice 3.1 do rovnice 3.2 dostaneme **kalorimetrickou rovnici**[7], která má tvar:

$$m_1 * c_1 * (t - t_1) = m_2 * c_2 * (t_2 - t) \tag{3.3}$$

kde: $t$ = výsledná teplota obou látek
$m_1, m_2$ = hmotnosti látek
$c_1, c_2$ = měrné tepelné kapacity látek
$t_1$ = počáteční teplota studenější látky
$t_2$ = počáteční teplota teplejší látky

## 3.4 Simulační model

### 3.4.1 Ohřev vody v teplovodních okruzích

Tepelný zdroj produkuje tepelnou energii. Tato energie je využita pro ohřátí vody v teplovodním okruhu. Pro výpočet teploty vody je použita rovnice 3.4, odvozena z rovnice 3.1 pro výpočet tepelné energie.

$$t_2 = \frac{Q}{m * c} + t_1 \tag{3.4}$$

kde: $t_2$ = nová teplota vody
$Q$ = tepelná energie z tepelného zdroje
$m$ = hmotnost vody v okruhu
$c$ = měrná tepelná kapacita vody ($c_{voda} = 4180$)
$t_1$ = teplota vody v okruhu

### 3.4.2 Tepelná výměna

Tepelná výměna mezi jednotlivými látkami v modelu není dokonalá. Výpočet tedy probíhá v těchto krocích:

1. Výpočet výsledné teploty dvou látek při dokonalé výměně tepla
2. Výpočet tepla, které se může přenést při dokonalé výměně
3. Výpočet výsledných teplot, při výměně části energie

Pro tento postup byly použity následující vztahy:

Výpočet výsledné teploty dvou látek při dokonalé výměně tepla - odvozeno z rovnice 3.3.

$$t = \frac{m_1 * c_1 * t_1 + m_2 * c_2 * t_2}{m_1 * c_1 + m_2 * c_2} \tag{3.5}$$

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 - odvozeno z rovnice 3.1

$$Q = m * c * (t_1 - t) \tag{3.6}$$

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

Výpočet výsledných teplot - odvozeno z rovnice 3.1

$$t_{f1} = \frac{-k * Q}{m_1 * c_1} + t_1 \tag{3.7}$$

$$t_{f2} = \frac{k * Q}{m_2 * c_2} + t_2 \tag{3.8}$$

kde: $t_{f1}, t_{f2}$ = výsledné teploty
$k$ = koeficient efektivity přenosu, $k \in \langle 0, 1 \rangle$, odpovídá účinnostem v tabulce 3.2
$Q$ = teplo
$m_1, m_2$ = hmotnost
$c_1, c_2$ = měrná tepelná kapacita
$t_1, t_2$ = počáteční teplota

### 3.4.3 Tepelné zdroje

Průběhy výkonů jednotlivých tepelných zdrojů (kamna na dřevo a solární panely) byly aproximovány pomocí funkce sinus, protože její průběh přibližně odpovídá vypozorovanému chování u tepelných zdrojů. Zdroje jsou tak idealizované. Simulované tepelné zdroje se v procesu optimalizace budou využívat pouze na prvotní naučení agenta posilovaného učení. Pro pozdější naučení se budou využívat naměřené teploty teplovodních okruhů, ze kterých se bojler ohřívá, a tepelné zdroje se simulovat nebudou.

#### Kamna na dřevo

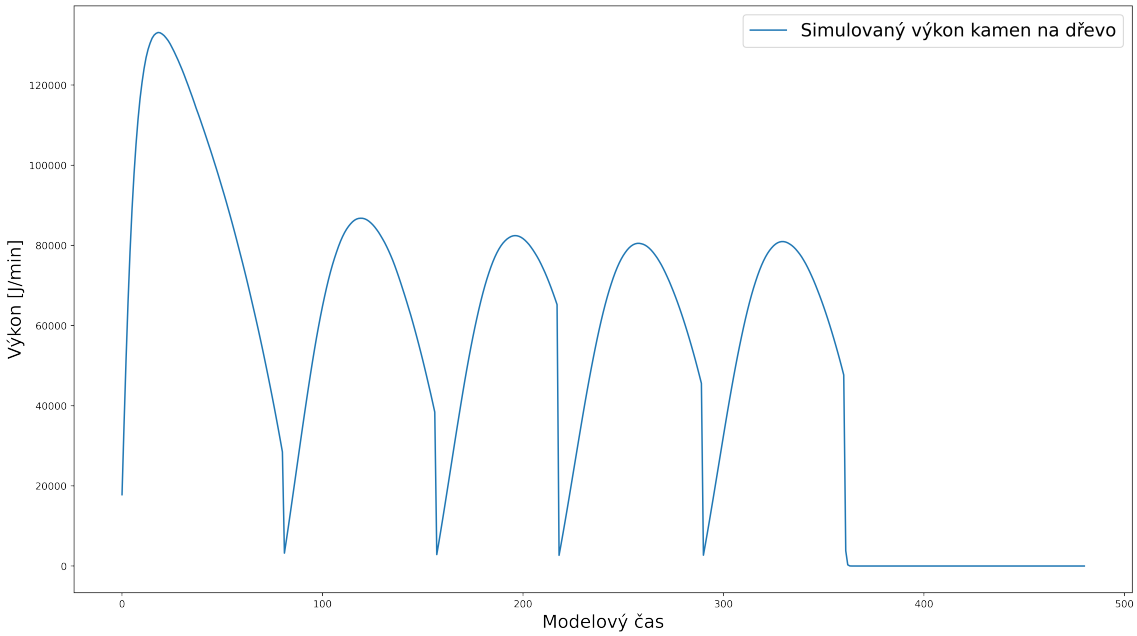
Kamna na dřevo mají maximální výkon 20 kWh. Do kamen se přikládá jednou za 60–90 minut. Kamna mají klapku, která upravuje jejich výkon (regulací vzduchu) podle teploty vody v teplovodním okruhu. Výkon také může ovlivnit člověk podle teploty v domě.

$$P = \frac{65}{t_{voda}} * \frac{25}{t_{vzduch}} * \frac{72000000 * \pi}{180} * sin(\frac{\pi * (T_l - T + 1)}{90}) \tag{3.9}$$

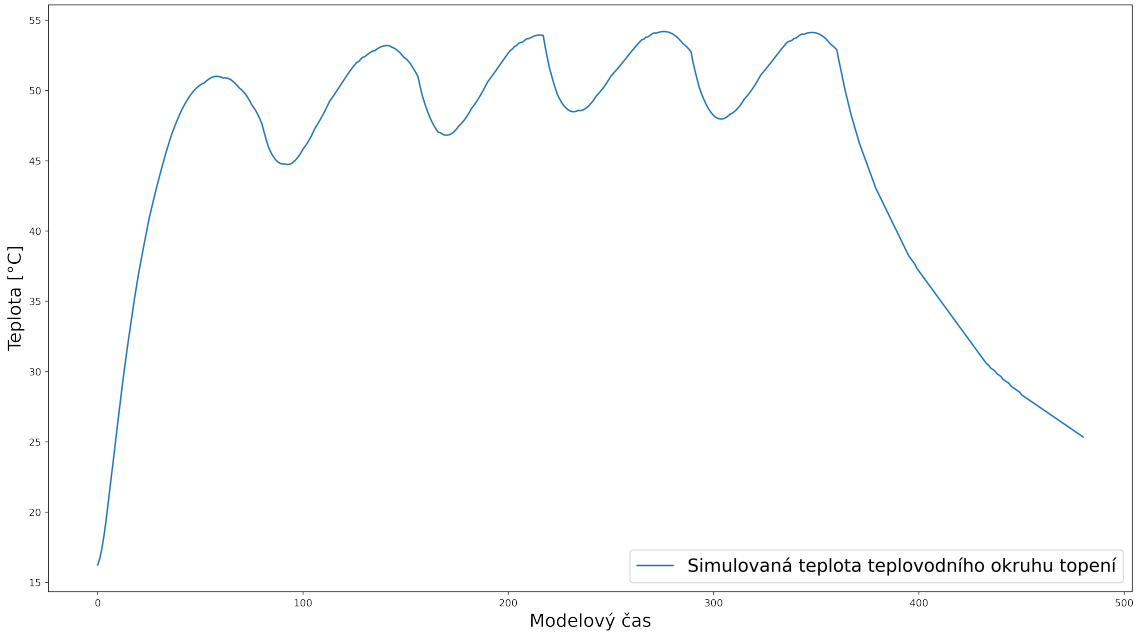
kde: $t_{voda}$ = teplota teplovodního okruhu topení
$t_{vzduch}$ = teplota vzduchu v domě
$T_l$ = modelový čas posledního přiložení
$T$ = modelový čas
$P$ = výkon v J/min

#### Použité konstanty pro změnu průběhu

- $\frac{\pi}{90}$ - změna periody na 180
- $\frac{72000000*\pi}{180}$ - celkový výkon, který tepelný zdroj dodá je 72000000 J/min (= 20kWh)
- $\frac{25}{t_{vzduch}}$ - úprava výkonu člověkem podle teploty v domě
- $\frac{65}{t_{voda}}$ - úprava výkonu mechanickou klapkou na kamnech




Obrázek 3.7: Graf průběhu výkonu simulovaných kamen na dřevo. K přiložení došlo v časech 0, 81, 157, 218, 290. V kamnech se celkově topilo po dobu 6 hodin.




Obrázek 3.8: Graf průběhu teploty teplovodního okruhu topení. K přiložení došlo v časech 0, 81, 157, 218, 290. V kamnech se celkově topilo po dobu 6 hodin.

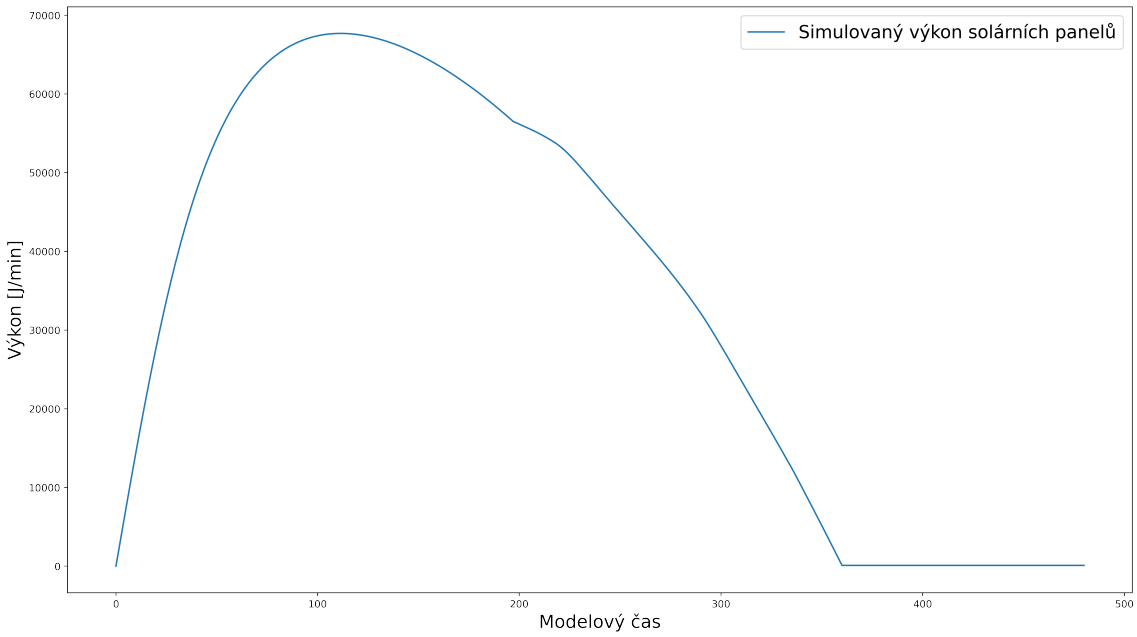
**Solární panely** Solární panely mají maximální výkon 2.66 kWh. Dodávají výkon po dobu 6 hodin.

$$P = \frac{65}{t_{voda}} * \frac{2400000 * \pi}{180} * sin(\frac{\pi * (T)}{360}) \tag{3.10}$$

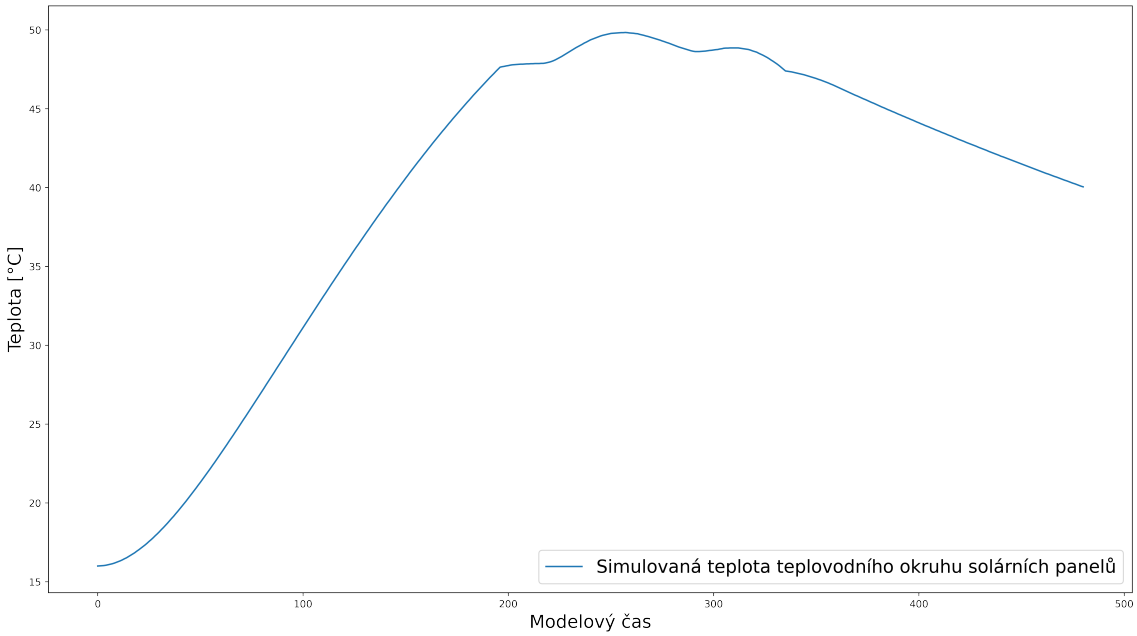
kde: $t_{voda}$ = teplota teplovodního okruhu solárních panelů
$T$ = modelový čas, $T \in \langle 0, 360 \rangle$
$P$ = výkon v J/m

### Použité konstanty pro změnu průběhu

- $\frac{\pi*(T)}{360}$ - změna periody na 720
- $\frac{2400000*\pi}{180}$ - celkový výkon, který tepelný zdroj dodá je 9600000 J/min (= 2.66kWh)
- $\frac{65}{t_{voda}}$ - úprava výkonu




Obrázek 3.9: Graf průběhu výkonu simulovaných solárních panelů.




Obrázek 3.10: Graf průběhu teploty teplovodního okruhu solárních panelů.

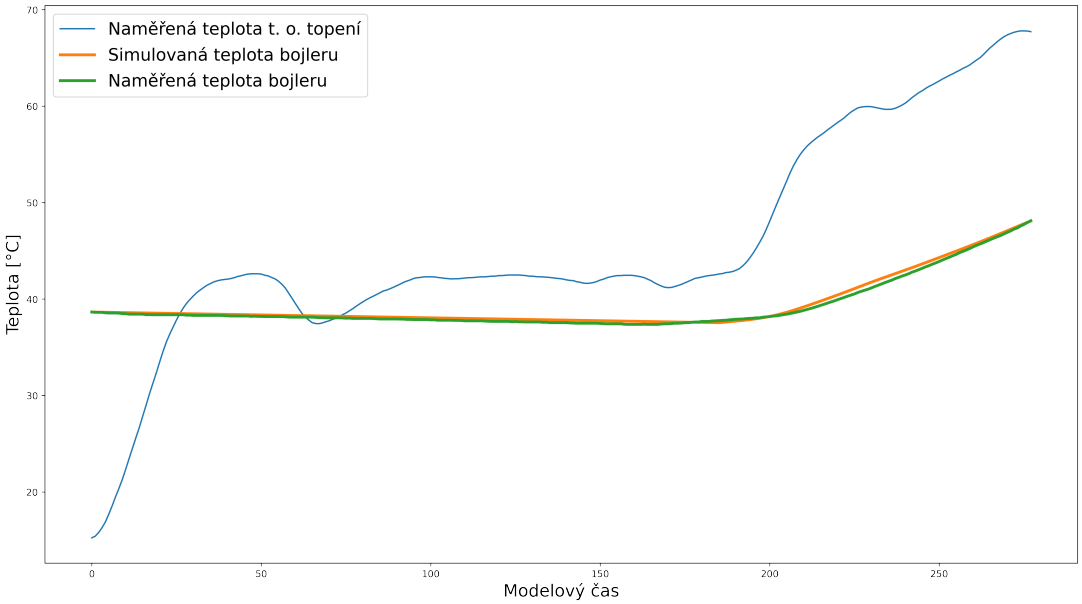
### 3.4.4 Validita modelu

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.

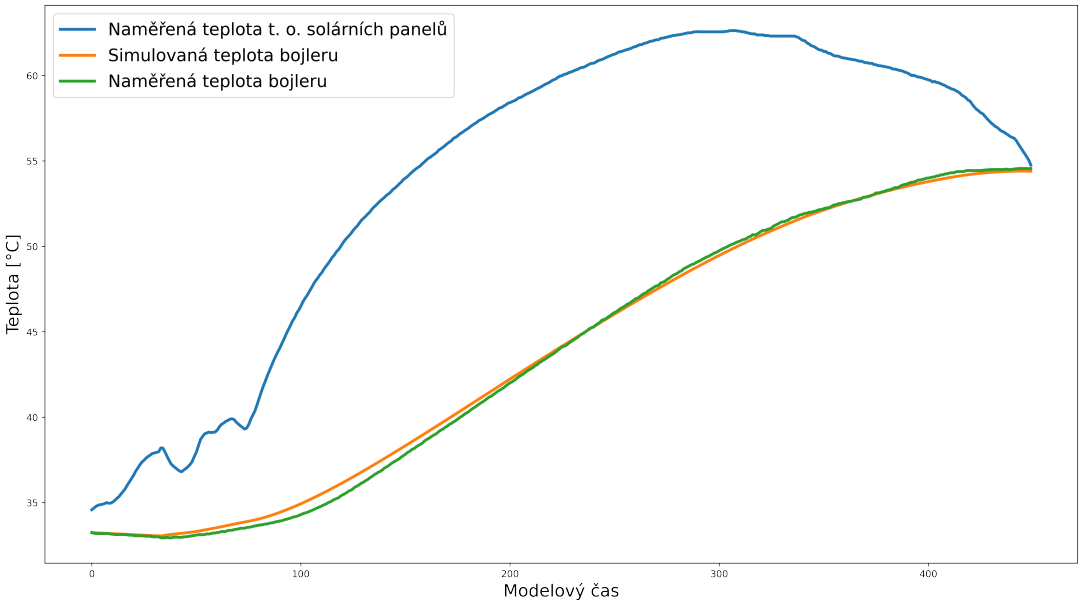
Z výsledků lze prohlásit model za validní.

**Ověření validity tepelné výměny u soustavy látek.**

Pro ověření validity tepelné výměny látek v systému byly použity naměřené teploty teplovodního okruhu topení a bojleru. Naměřená teplota bojleru se porovnala s tou simulovanou.




Obrázek 3.11: Graf výsledků simulace a skutečných naměřených hodnot. Modrá křivka odpovídá naměřeným hodnotám teploty teplovodního okruhu topení. Oranžová odpovídá naměřeným hodnotám teploty bojleru. Zelená je simulovaná teplota bojleru. Z průběhu simulovaných a naměřených teplot bojleru lze prohlásit, že je model validní. Simulovaná teplota bojleru téměř zcela odpovídá té naměřené. Teplota bojleru je navíc měřená senzorem, který má přesnost $\pm 0.5°C$



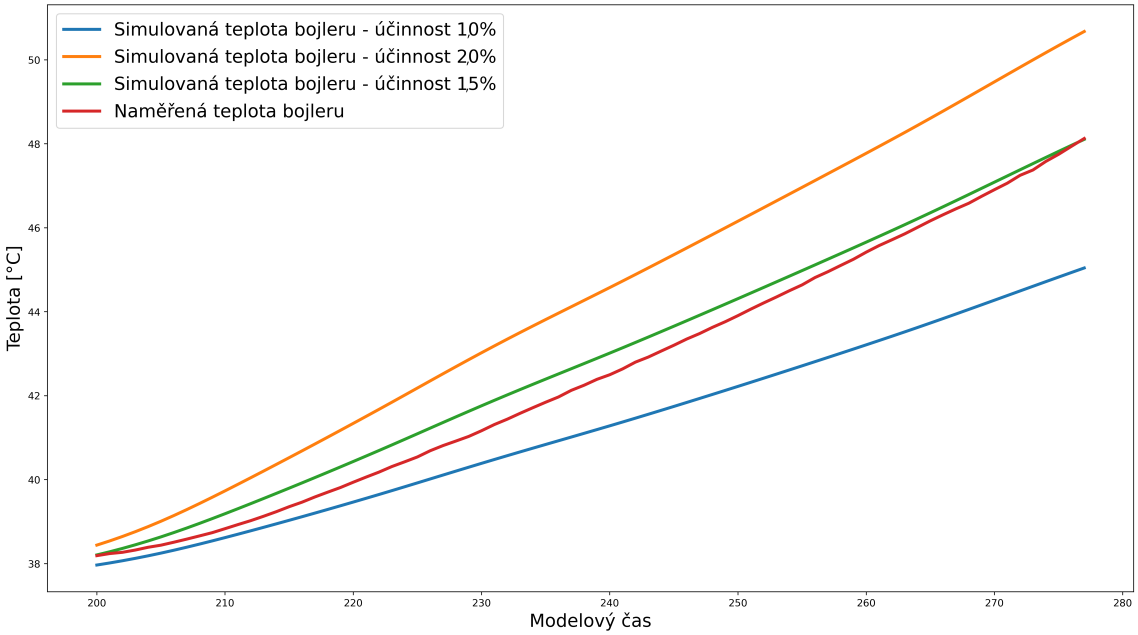

Obrázek 3.12: Graf výsledků simulace a skutečných naměřených hodnot. Modrá křivka odpovídá naměřeným hodnotám teploty teplovodního solárních panelů. Oranžová odpovídá neměřeným hodnotám teploty bojleru. Zelená je simulovaná teplota bojleru. Z průběhu simulovaných a naměřených teplot bojleru lze prohlásit, že je model validní. Simulovaná teplota bojleru téměř zcela odpovídá té naměřené. Teplota bojleru je navíc měřená senzorem, který má přesnost $\pm 0.5°C$

### 3.4.5 Simulační experimenty pro zjištění účinností přenosu

Účinnosti tepelných přenosů v abstraktním modelu byly zjištěny pomocí simulačních experimentů. V experimentech byly použity naměřené hodnoty a simulované hodnoty s nimi byly porovnány.

#### Účinnost přenosu z teplovodního okruhu topení do bojleru

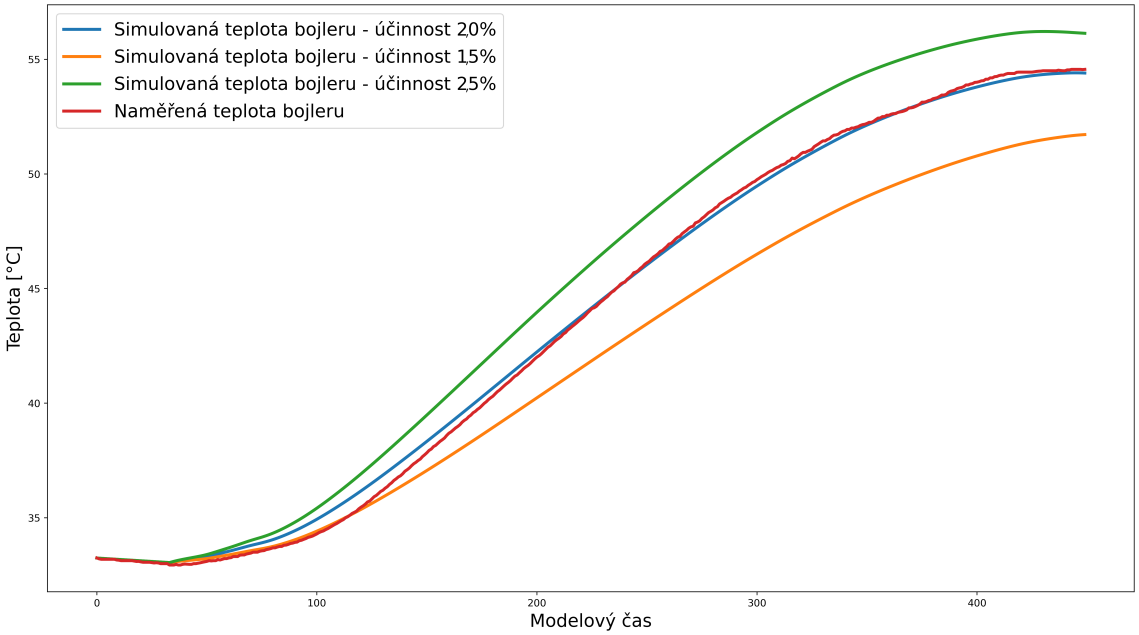
V tomto experimentu byly použity naměřené hodnoty teplovodního okruhu topení. Simuloval se ohřev bojleru. Simulované hodnoty teploty bojleru byly porovnány s těmi naměřenými. Tímto experimentem byla zjištěna účinnost 1,5%.




Obrázek 3.13: Graf průběhu simulovaných teplot bojleru s různými účinnostmi tepelné výměny mezi látkami. Z výsledků lze pozorovat, že stanovená účinnost tepelné výměny mezi teplovodním okruhem topení a bojlerem 1,5% je pro potřeby simulace dostatečně přesná. Teplota bojleru je snímána teplotním senzorem, který má přesnost $\pm 0.5°C$

### Účinnost přenosu z teplovodního okruhu solárních panelů do bojleru

V tomto experimentu byly použity naměřené hodnoty teplovodního okruhu solárních panelů. Simuloval se ohřev bojleru. Simulované hodnoty teploty bojleru byly porovnány s těmi naměřenými. Tímto experimentem byla zjištěna účinnost 2,0%.




Obrázek 3.14: Graf průběhu simulovaných teplot bojleru s různými účinnostmi tepelné výměny mezi látkami. Z výsledků lze pozorovat, že stanovená účinnost tepelné výměny mezi teplovodním okruhem solárních panelů a bojlerem 2,0% je pro potřeby simulace dostatečně přesná. Teplota bojleru je snímána teplotním senzorem, který má přesnost $\pm 0.5°C$

# Kapitola 4

# Návrh řešení

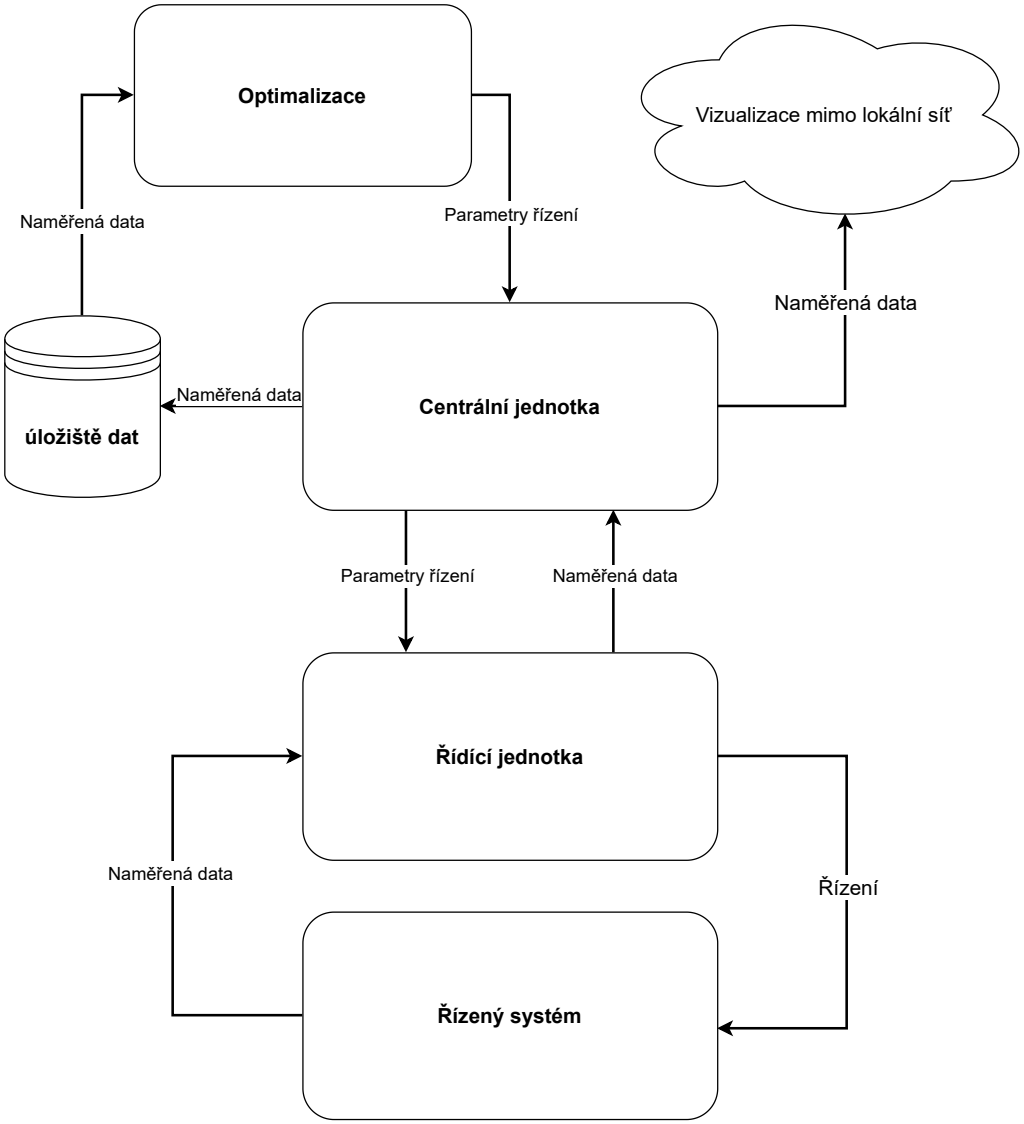
V této kapitole je popsán návrh celého řídícího systému a návrh optimalizace využití externích zdrojů pro ohřev bojleru. V návrhu řídícího systému budou popsány jeho jednotlivé části. U návrhu optimalizace bude popsána optimalizace pomocí strojového učení, konkrétně pomocí algoritmu hlubokého Q-učení a jednotlivé části modelu strojového učení.

## 4.1 Návrh řídícího systému

Řídící systém se bude skládat ze 2 hlavních částí:

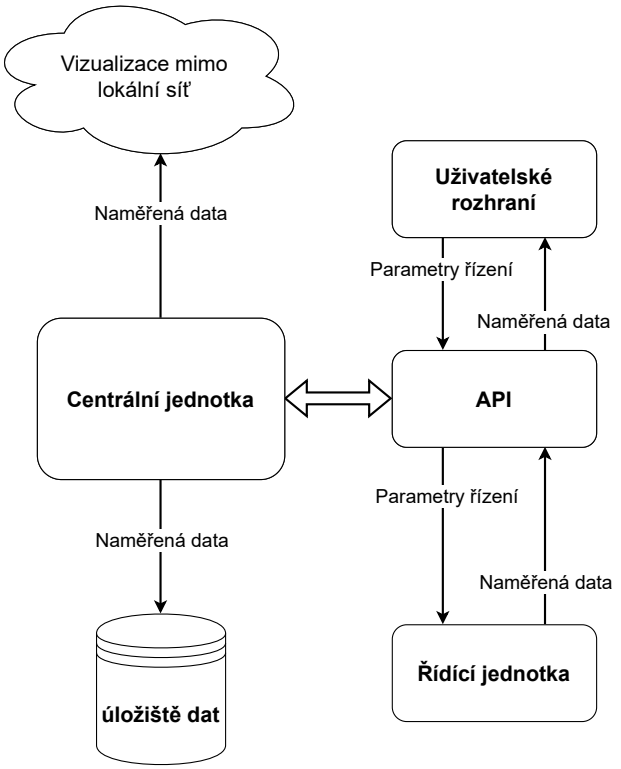
- Řídící jednotka
- Centrální jednotka

Tyto části spolu budou komunikovat pomocí Wi–Fi.




Obrázek 4.1: Schéma návrhu systému. Systém se skládá ze 2 hlavních částí - Centrální jednotka a Řídící jednotka. Řídící jednotka zaznamenává naměřené teploty jednotlivých částí systému a spíná ohřevy bojleru. Centrální jednotka sbírá naměřená data a zasílá parametry řízení.

### 4.1.1 Centrální jednotka



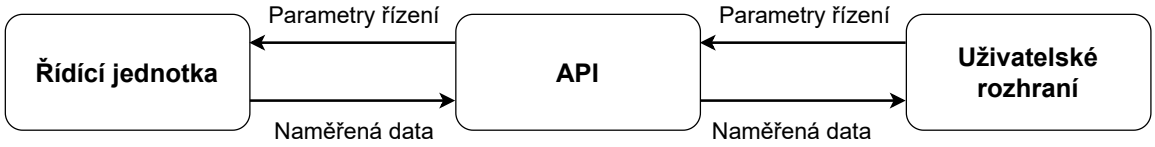

Obrázek 4.2: Schema návrhu centrální jednotky. Centrální jednotka poskytuje API, přes které získává naměřená data od řídící jednotky a také přes něj komunikuje s webovým uživatelským rozhraním. Dále ukládá naměřená data. Poslední úlohou je zasílání naměřených dat na službu na cloudu, která tyto data zprostředkuje mimo lokální síť.

Centrální jednotka bude poskytovat uživatelské rozhraní přes pomocí HTML stránky, kde uživatelé budou moci kontrolovat naměřené teploty a stav ohřevu vody. Dále bude poskytovat HTTP rozhraní řídící jednotku, přes které bude sbírat naměřená data a zaznamenávat je. Nazpět bude řídící jednotce posílat parametry řízení - hodnoty teplotních rozdílů, při kterých se má spouštět ohřev bojleru z jednotlivých externích zdrojů.

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

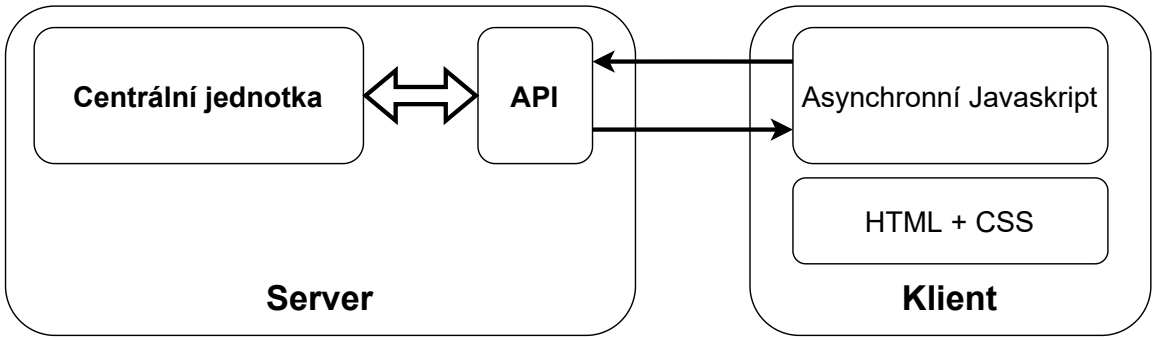
#### Aplikační a uživatelské rozhraní

Centrální řídící jednotka poskytuje jednoduché REST aplikační rozhraní, přes které komunikuje s řídící jednotkou. Přes toto rozhraní komunikuje i s webovým uživatelským rozhraním.




Obrázek 4.3: Schéma popisující komunikaci částí řídícího systému s aplikačním rozhraním centrální jednotky.

Uživatelské rozhraní je navrženo jako webové rozhraní, kdy centrální jednotka zastupuje roli serveru a komunikuje s klientem. Klientskou část tvoří HTML stránka, která komunikuje se serverem pomocí asynchronního Javaskriptu.




Obrázek 4.4: Schéma popisující použitou architekturu - klient–server a uživatelské rozhraní.

### Úložiště dat

Centrální jednotka bude ukládat naměřená data od řídící jednotky. Tyto data budou pak využívána:

- Pro vizualizaci průběhu teplot v průběhu dne
- Pro optimalizaci využití externích zdrojů pro ohřev bojleru

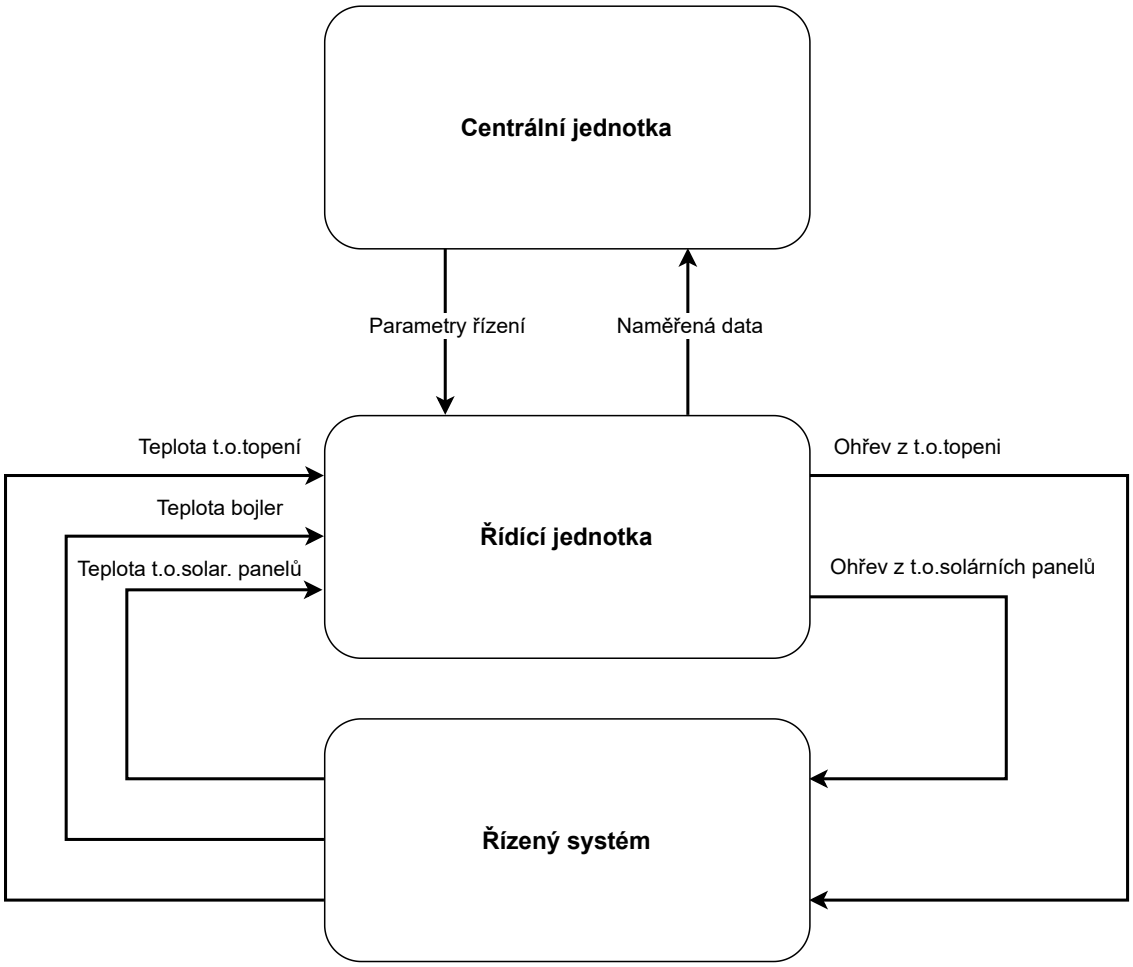
Naměřená data se budou ukládat do souborového systému ve formátu JSON tak, jak je centrální jednotka obdrží. Každý den bude uložený do samostatného souboru. Data v tomto formátu obsahují i nadbytečné znaky formátu JSON, ale snadněji se s nimi pracuje jak při zasílání dat na klienta pro vizualizaci, tak i při optimalizaci.


Obrázek 4.5: Schéma úložiště naměřených dat. Data jsou ukládána ve formátu JSON. Jednotlivé záznamy, které centrální jednotka získá od řídící jednotky, se ukládají do souborů. Každý soubor obsahuje záznamy z jednoho dne a má název ve formátu „Měsíc_Den_Rok“, kde „Měsíc“ je zkratka daného měsíce, „Den“ číslo dne v měsíci a „Rok“ je číslo roku.

```
{"kolektor":"21.44","boiler":"49.25","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:03"}
{"kolektor":"21.38","boiler":"49.25","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:08"}
{"kolektor":"21.38","boiler":"49.19","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:24"}
{"kolektor":"21.31","boiler":"49.19","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:35"}
{"kolektor":"21.38","boiler":"49.25","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:40"}
{"kolektor":"21.38","boiler":"49.19","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:00:46"}
{"kolektor":"21.31","boiler":"49.19","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:01:06"}
{"kolektor":"21.38","boiler":"49.19","kamna":"21.25","releboiler":"0","relesolar":"0","timestamp":"Jul,13,2021 00:01:12"}
```


Obrázek 4.6: Ukázka formátu souboru, kam se ukládají data. Každý řádek je JSON záznam dat. Jednotlivé položky jsou naměřené teploty, stav zapnutí ohřevu, datum a čas záznamu.

### 4.1.2 Řídící jednotka

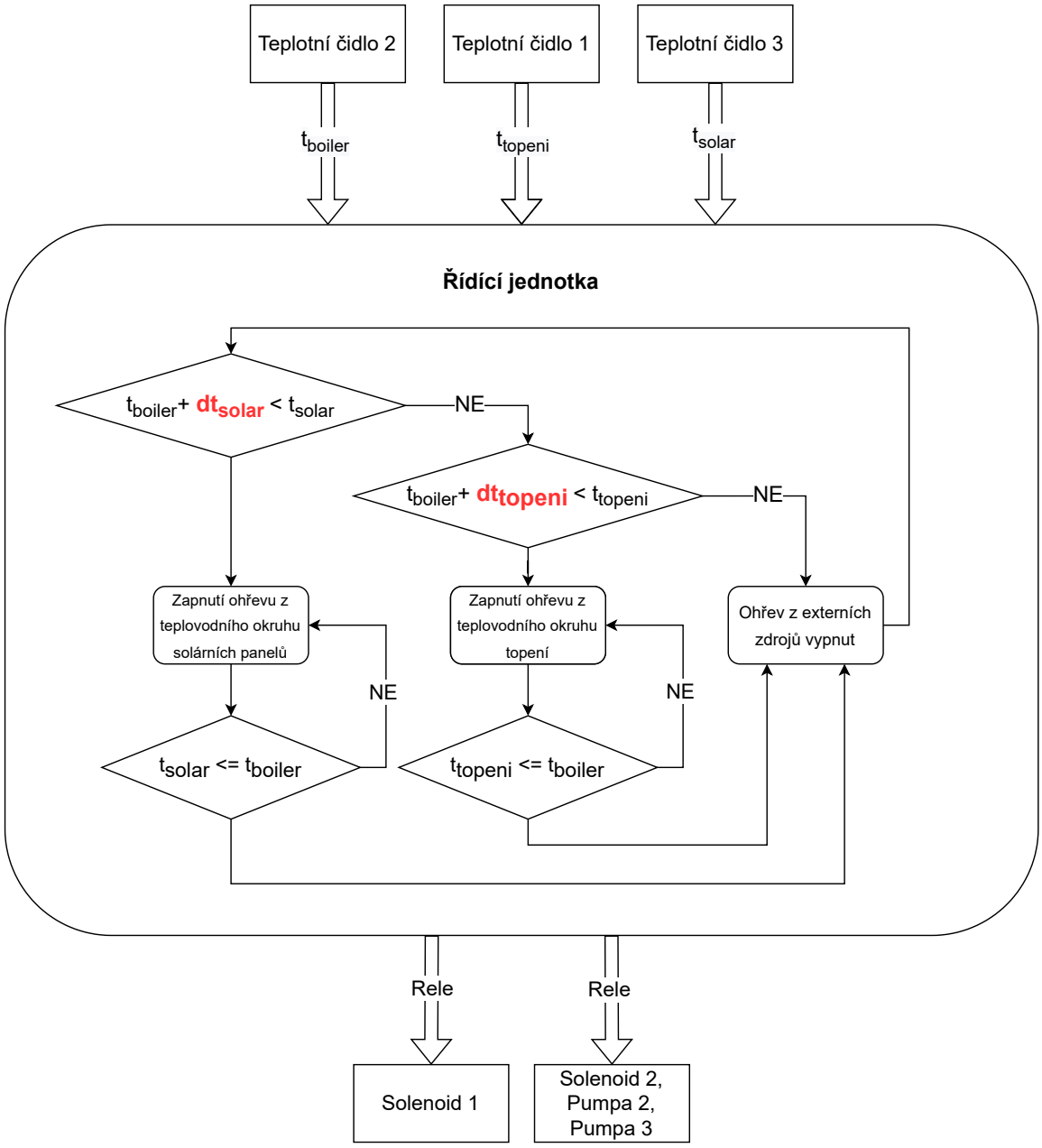



Obrázek 4.7: Schéma návrhu řídící jednotky. Řídící jednotka získává pomocí 3 teplotních senzorů teploty. Pomocí dvou relé zapíná ohřev bojleru z jednotlivých externích zdrojů. Dále zasílá naměřená data centrální jednotce a získává od ní parametry řízení.

Řídící jednotka 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 Wi-Fi komunikovat s Centrální jednotkou.

Spínání ohřevu bojleru bude probíhat pomocí parametru řízení, které získá od Centrální jednotky. Tyto parametry jsou:

- Teplotní rozdíl bojleru a teplovodního okruhu topení
- Teplotní rozdíl bojleru a teplovodního okruhu solárních panelů




Obrázek 4.8: Schéma řídící jednotky, včetně diagramu řídící logiky. Řídící jednotka získává pomocí 3 teplotních senzorů teploty. Pomocí dvou relé zapíná ohřev bojleru z jednotlivých externích zdrojů. Spínání probíhá podle parametrů $dt_{topeni}$ a $dt_{solar}$, které odpovídají parametrům řízení získaných od centrální jednotky - tedy teplotním rozdílům, při kterých se má ohřev zapnout.

## 4.2 Optimalizace ohřevu

Cílem optimalizace je nalézt optimální rozdíl teplot pro zapínání ohřevu bojleru z externích zdrojů.

Metod pro optimalizaci ohřevu bojleru z externích zdrojů lze využít víc. Optimalizace tak není závislá na řídícím systému. Pro tuto optimalizaci lze využít jak simulační model, tak i naměřená data z provozu. Nové parametry pro řízení lze zadat pře uživatelské rozhraní, nebo je zaslat na centrální jednotku přes její aplikační rozhraní.

Optimalizační metody, které by šly pro tuto problematiku použít jsou například:

- Genetické algoritmy, které by optimalizovaly konkrétní teplotní rozdíly použité při řízení ohřevu bojleru.
- Q-učení, při kterém by agent mohl zastupovat roli řídící jednotky.
- Q-učení, při kterém by agent přímo hledal optimální teplotní rozdíly pro řízení ohřevu bojleru

Zvolená metoda optimalizaci využití zdrojů je algoritmus **hlubokého Q-učení**. Agent zastupuje roli řídící jednotky a využívá hlubokou neuronovou síť. Neuronové sítě jsou pro tuto problematiku poměrně komplexní nástroj. Výsledky, ale mohou vést ke komplexnějším metodám řízení. Proto byla tato metoda, jako experimentální, využita.

Pro učení agenta využijeme 2 prostředí. Jako první prostředí se využije simulační model reálného systému, včetně simulovaných tepelných zdrojů. V druhém prostředí se bude simulovat pouze ohřev bojleru a využijí se reálné naměřené teploty teplovodních okruhů, jelikož je simulace tepelných zdrojů v simulačním modelu idealizovaná.

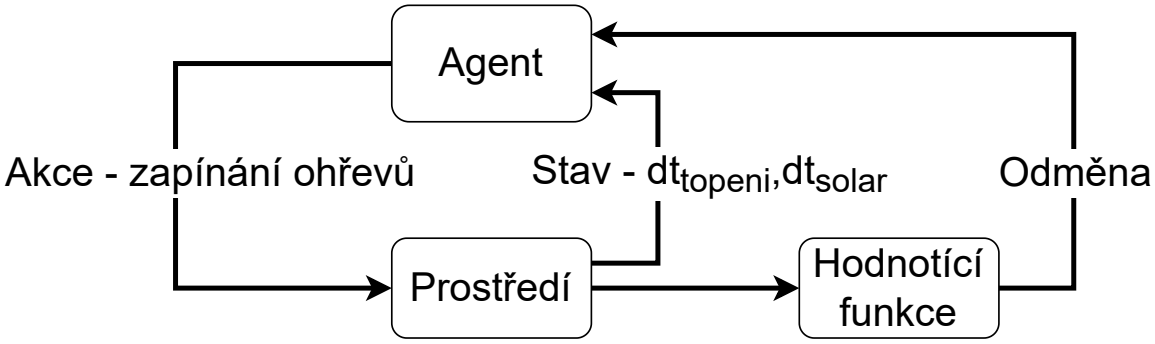
## 4.3 Hluboké Q-učení

Pro nalezení optimálních hodnot teplotního rozdílu pro řízení ohřevu bojleru bude použit algoritmus hlubokého Q-učení. Agent se bude učit na dvou prostředí.

Pro prvotní naučení agenta se využije prostředí, které bude simulovat celý systém ohřevu bojleru, včetně tepelných zdrojů. Na tomto prostředí se agent hlavně naučí provádět akce tak, aby nespouštěl ohřev bojleru, když jsou oba externí zdroje studenější, také aby často nezapínal ohřevy z externích zdrojů.

Pro další učení a samotnou optimalizaci bude použito prostředí, které využívá reálná naměřená data. Konkrétně teploty externích zdrojů - teplovodních okruhů. Tyto teploty přímo odpovídají reálnému chování a simulovat se bude pouze ohřev bojleru.

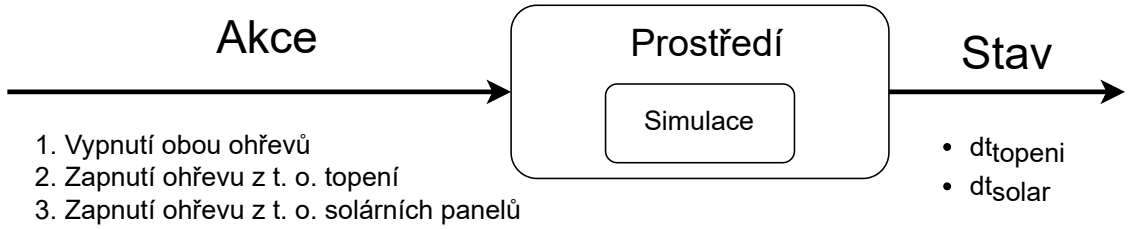
Agent bude u prostředí moci pozorovat jednotlivé teplotní rozdíly bojleru a teplovodních okruhů. A akce, které bude provádět, budou odpovídat zapínání ohřevu bojleru z externích zdrojů. Bude tedy zastupovat roli řídící jednotky.




Obrázek 4.9: Schéma návrhu hlubokého Q-učení. Stav prostředí, které může agent pozorovat budou teplotní rozdíly $dt_{topeni}$ a $dt_{solar}$, které odpovídají rozdílům teplot teplovodních okruhů a bojleru. Akce, které může agent provádět jsou zapnutí ohřevu bojleru z jednotlivých externích zdrojů.

### 4.3.1 Prostředí

Agent bude provádět akce v prostředí, pomocí kterých se bude měnit stav prostředí. 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.



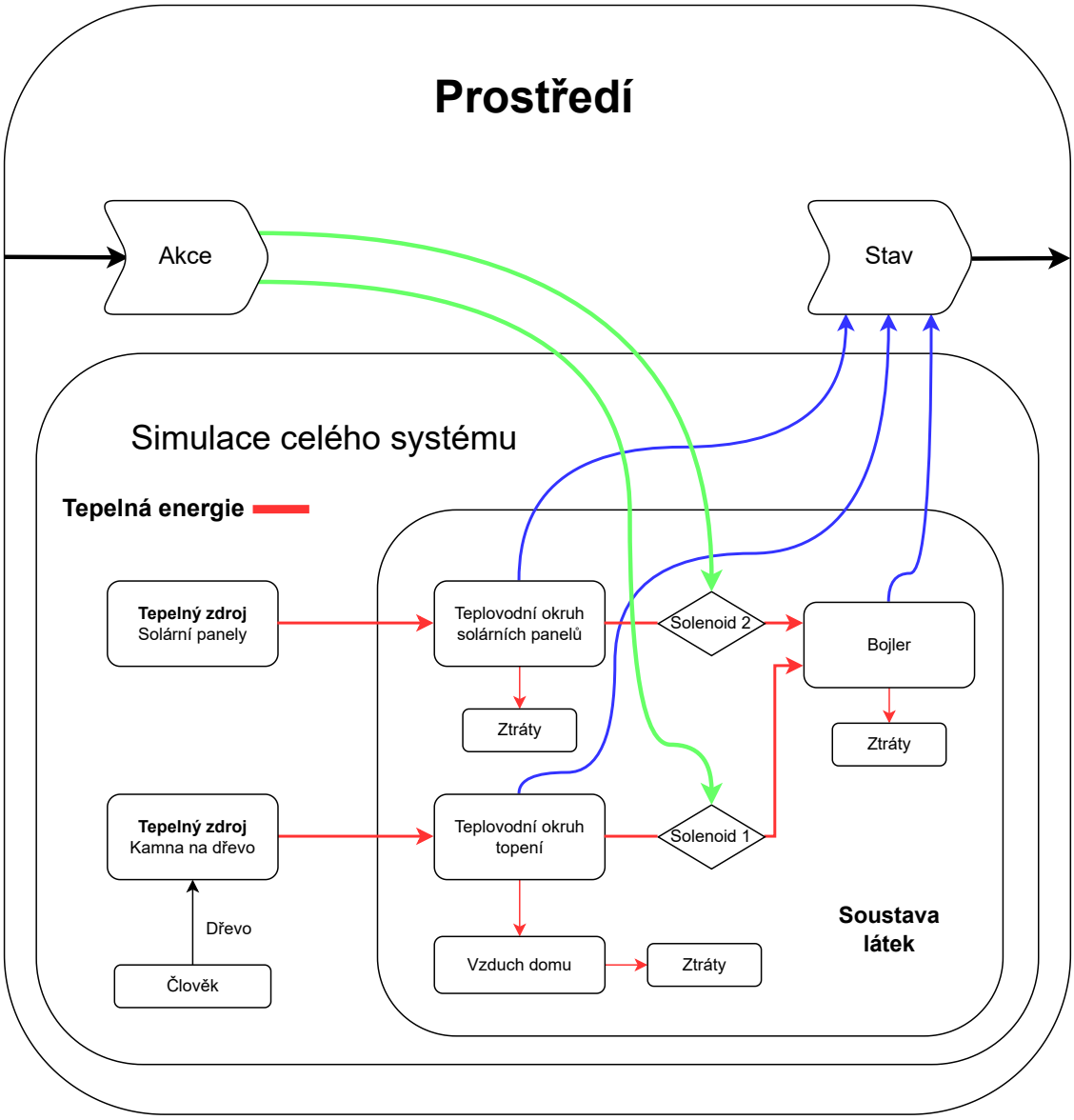

Obrázek 4.10: Schéma popisující prostředí pro učení agenta. V prostředí lze provádět 3 akce a jeho stavem jsou jednotlivé teplotní rozdíly $dt_{topeni}$ a $dt_{solar}$, které odpovídají rozdílům teplot teplovodních okruhů a bojleru.

Pro učení agenta budou využity 2 prostředí:

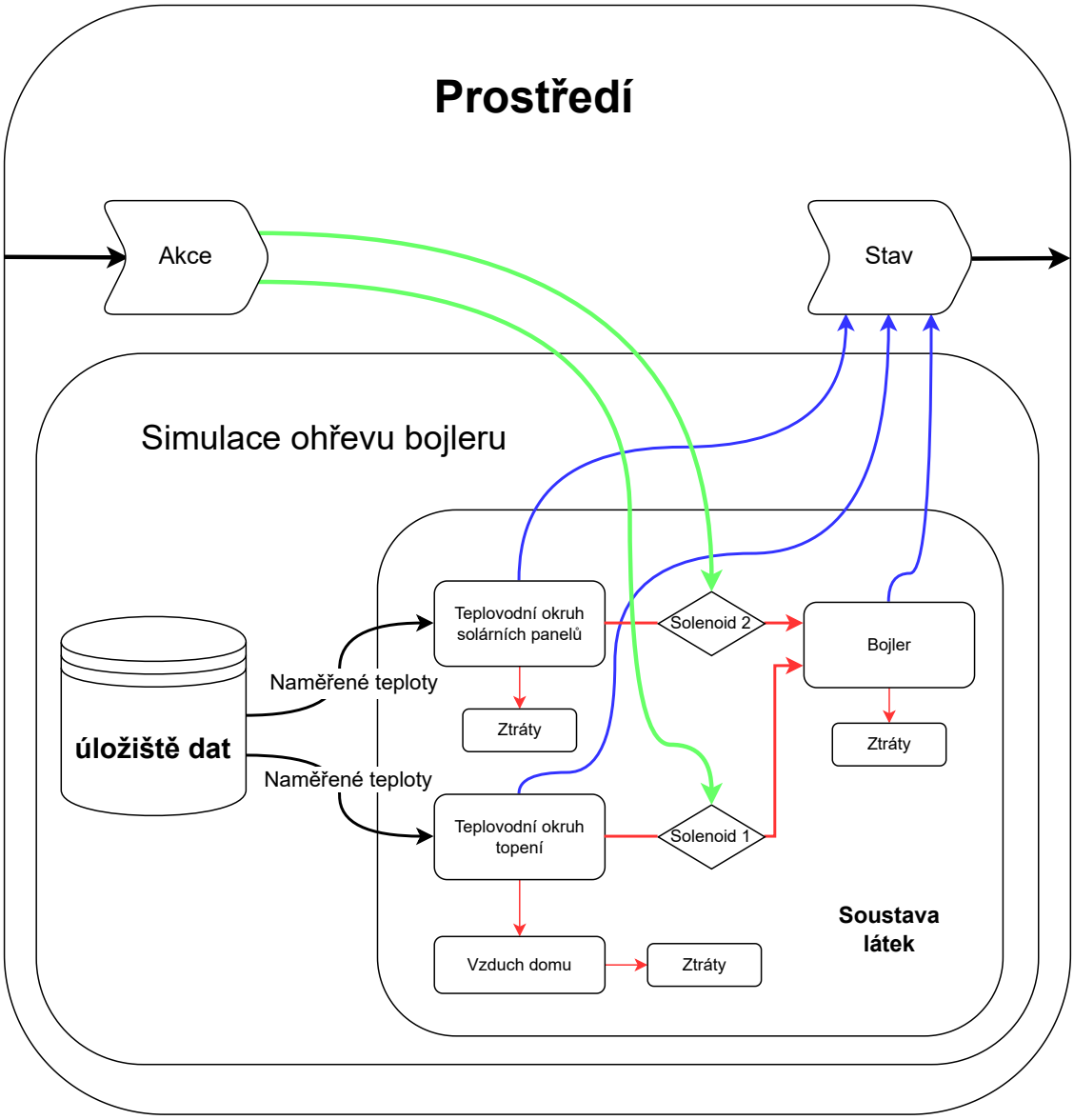
1. Prostředí pro prvotní naučení agenta, které simuluje celý systém ohřevu bojleru, včetně tepelných zdrojů
2. Prostředí pro další učení agenta, které simuluje pouze ohřev bojleru a využívá naměřených dat

Obě prostředí budou využívat abstraktní model, nebo jeho části, který je popsaný na straně 14.

**1. prostředí**




Obrázek 4.11: Schéma popisující prostředí pro prvotní naučení agenta. Prostředí obsahuje simulaci celého systému ohřevu bojleru, včetně simulovaných tepelných zdrojů.




Obrázek 4.12: Schéma popisující prostředí pro další učení agenta. Prostředí simuluje pouze ohřívání bojleru. Pro teploty teplovodních okruhů se využijí naměřené hodnoty.

### 4.3.2 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.

Funkce odměny bude přímo ovlivňovat optimalizaci řízení ohřevu bojleru z externích zdrojů. Agent bude dostávat pozitivní odměnu pokud se bude bojler ohřívat. Aby ale nezapínal ohřev při malých teplotních rozdílech bojleru a externích zdrojů, funkce musí reflektovat

časové rozestupy mezi zapínáním ohřevu. Zapínání ohřevu při malých teplotních rozdílech bojleru a externích zdrojů by vedlo k rychlému opotřebení hardwarových součástek, což není žádoucí. Funkce odměny bude dávat negativní hodnoty pokud bude agent zapínat ohřev, ale externí zdroje budou mít nižší teplotu než bojler.

Tato hodnotící funkce by tak agenta měla navézt k tomu, aby nalezl co nejmenší teplotní rozdíly bojleru a externích zdrojů, aby došlo k lepšímu využití externích zdrojů pro ohřev, ale zároveň takovým, aby nedocházelo k příliš častému zapínání a vypínání ohřevu.

**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.

Konkrétní hodnoty použité v hodnotící funkci jsou popsány na straně 51.

### 4.3.3 Učení agenta

Učení agenta bude probíhat ve dvou fázích:

1. **Učení agenta na plně simulovaném prostředí** - prostředí 1(viz Popis prostředí na straně 35)
2. **Učení agenta na prostředí využívající naměřená data** - prostředí 2(viz Popis prostředí na straně 35)

První fáze slouží k tomu, aby se agent naučil základní logice spouštění ohřevu. Tedy aby zapínal ohřev pouze tehdy, když je jeden z externích zdrojů teplejší, aby nechal zapnutý ohřev dokud je jeden ze zdrojů teplejší než bojler a aby často nezapínal a nevypínal ohřev z externích zdrojů.

Druhá fáze slouží k tomu, aby se agent učil na reálných datech z provozu. Tato fáze se může opakovat i periodicky na nově naměřených datech. Cílem této fáze je optimalizovat parametry pro řízení.

Výsledkem učení agenta bude naučená neuronová síť z které pak lze získat optimální parametry pro řízení ohřevju vody.

# Kapitola 5

# Implementace

## 5.1 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.

### 5.1.1 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.

### 5.1.2 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.

### 5.1.3 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.

### 5.1.4 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.

### 5.1.5 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 [4].

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

### 5.1.6 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.[1]

### 5.1.7 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.

### 5.1.8 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.

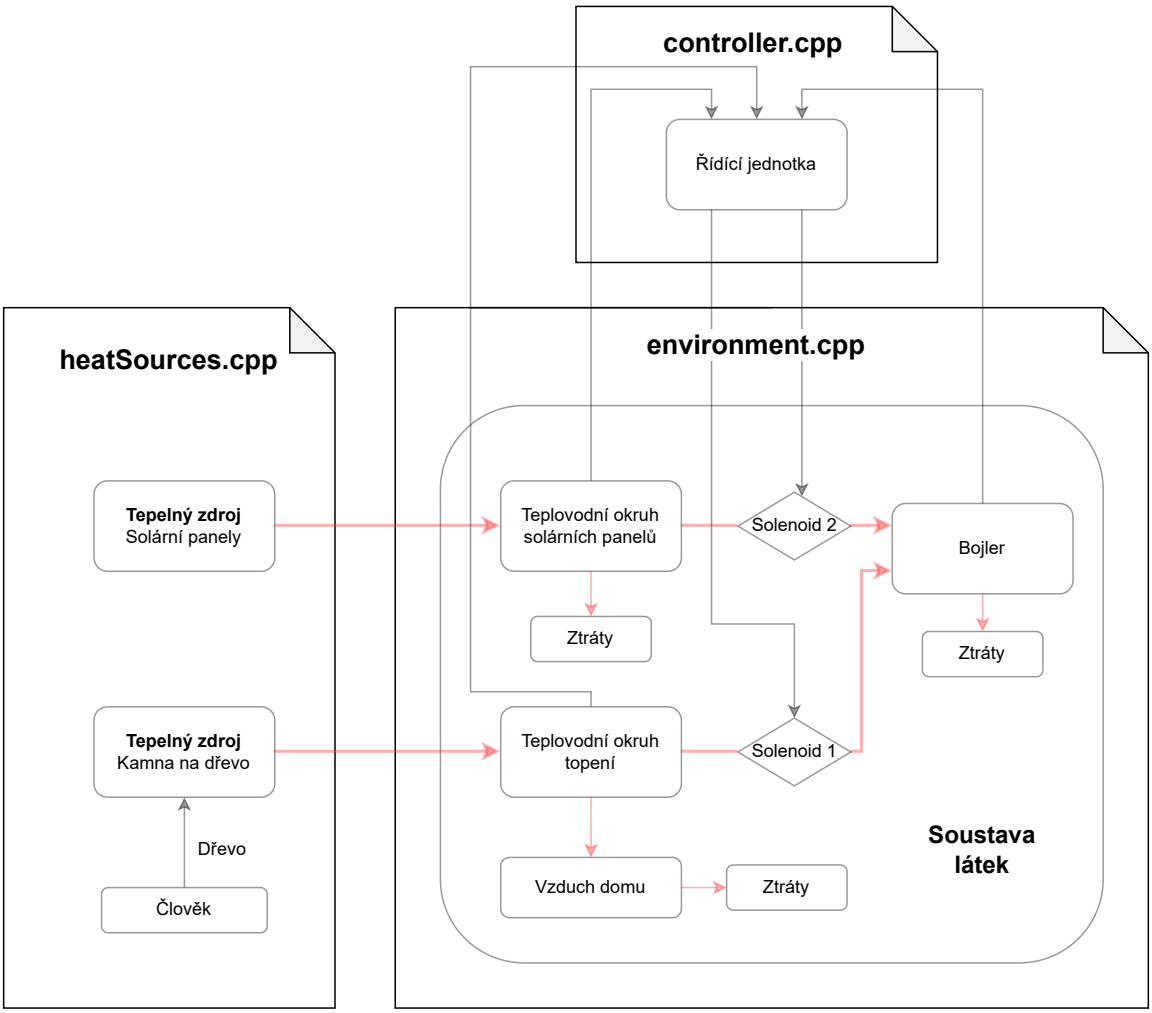
### 5.1.9 Matplotlib

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.

## 5.2 Simulační model

Simulační model pro systému byl implementován v Jazyce C++ pomocí knihovny Simlib [9]. Parametry simulace se dají měnit pomocí argumentů. Pro vygenerovaní Makefile byl použit nástoj CMake[1]. Části simulačního modelu jsou rozděleny do těchto modulů:

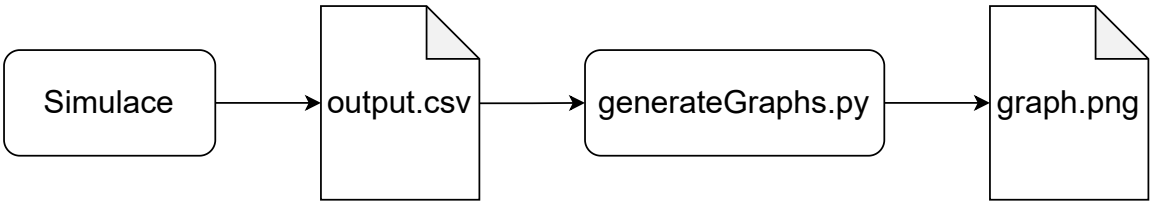
- `main.cpp` - hlavní modul, zpracování argumentů, spouštění simulace
- `environment.cpp` - tepelná výměnu mezi jednotlivými částmi systému
- `controller.cpp` - logika řídící jednotky
- `heatSources.cpp` - tepelné zdroje
- `generator.cpp` - instancuje třídy z ostatních modulů, které jsou využity pro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




Obrázek 5.1: Schéma rozdělení částí simulačního modelu do jednotlivých modulů.

[1] https://cmake.org/

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




Obrázek 5.2: Schéma popisující výstup simulace, včetně skriptu pro vizualizaci simulovaných hodnot.

## 5.3 Řídící systém

### 5.3.1 Popis hardwarových částí

#### Arduino nano

Arduino nano je jednočipová vývojová deska od společnosti Arduino[2]. 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.

#### ESP-8266

ESP-8266 je mikroprocesor od společnosti Espressif[3]. 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.

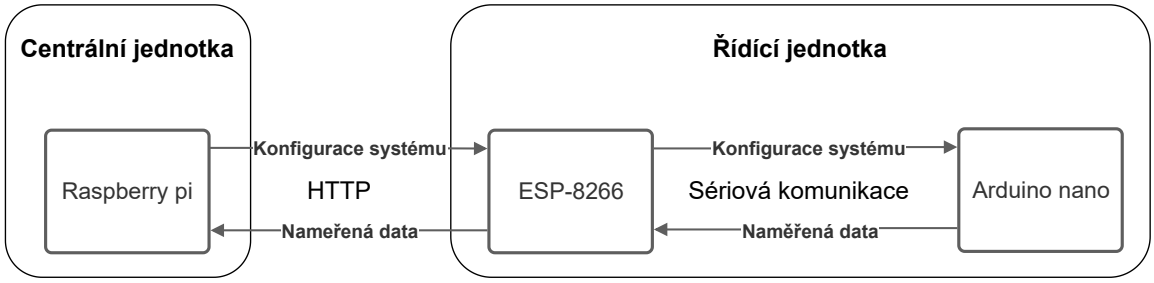
Díky podpoře Wi-Fi byl použit pro komunikaci s řídící jednotkou.

#### Raspberry pi 400

Raspberry pi 400 je jednodeskový počítač od společnosti Raspberry Pi[4]. 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.

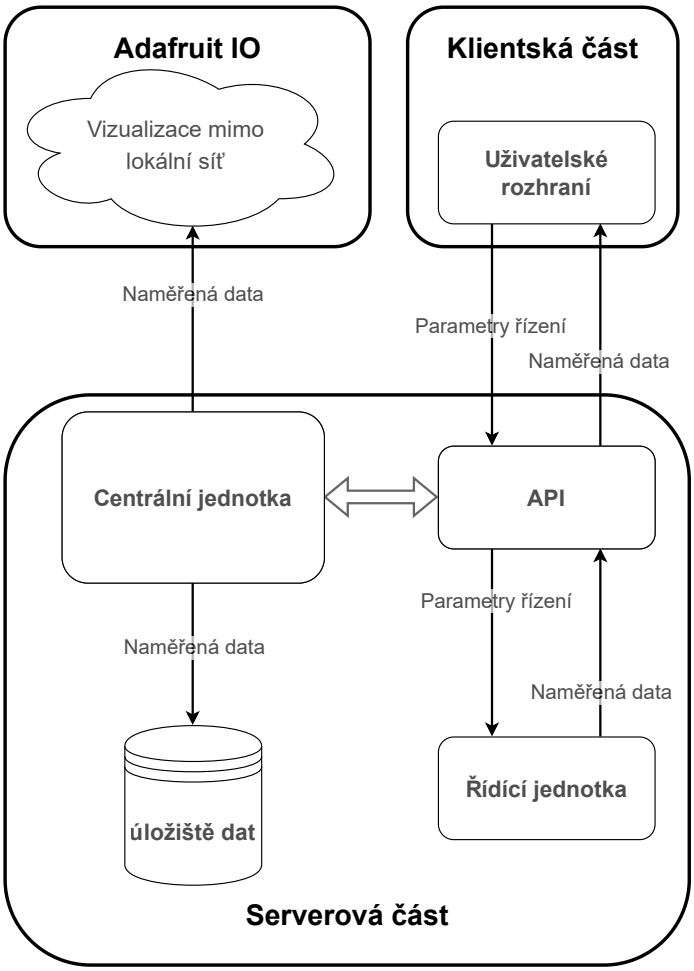
[2] https://www.arduino.cc/
[3] https://www.espressif.com/
[4] https://www.raspberrypi.com/




Obrázek 5.3: Schéma komunikace jednotlivých částí řídícího systému.

## 5.4 Centrální jednotka

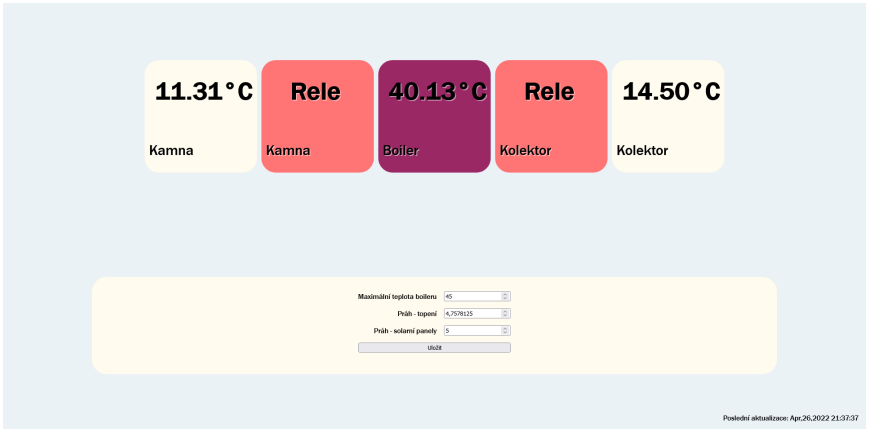
Centrální jednotka používá Web server Apache pro poskytnutí webového rozhraní. Pro implementaci jsou použity PHP skripty a Python skripty. A je implementovaná na Raspberry pi 400.




Obrázek 5.4: Schéma rozdělení implementace centrální jednotky do jednotlivých částí.

### 5.4.1 Klientská část

Uživatelské rozhraní bylo implementováno za použití HTML a CSS. Pro komunikaci se serverem využívá asynchronní 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.




(a) Hlavní stránka - počítače


(b) Hlavní stránka - mobilní zařízení

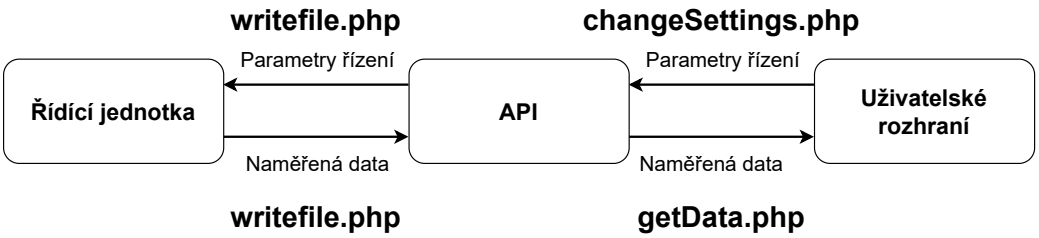

(c) Stránka s průběhem teplot

Obrázek 5.5: Navržené uživatelské prostředí řídícího sytému. Hlavní stránka obsahuje naměřené hodnoty a stav jednotlivých ohřevů z externích zdrojů. Další stránka obsahuje graf průběhu teplot během aktuálního dne.

### 5.4.2 Serverová část

Pro implementaci serverové části centrální jednotky 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`[5].

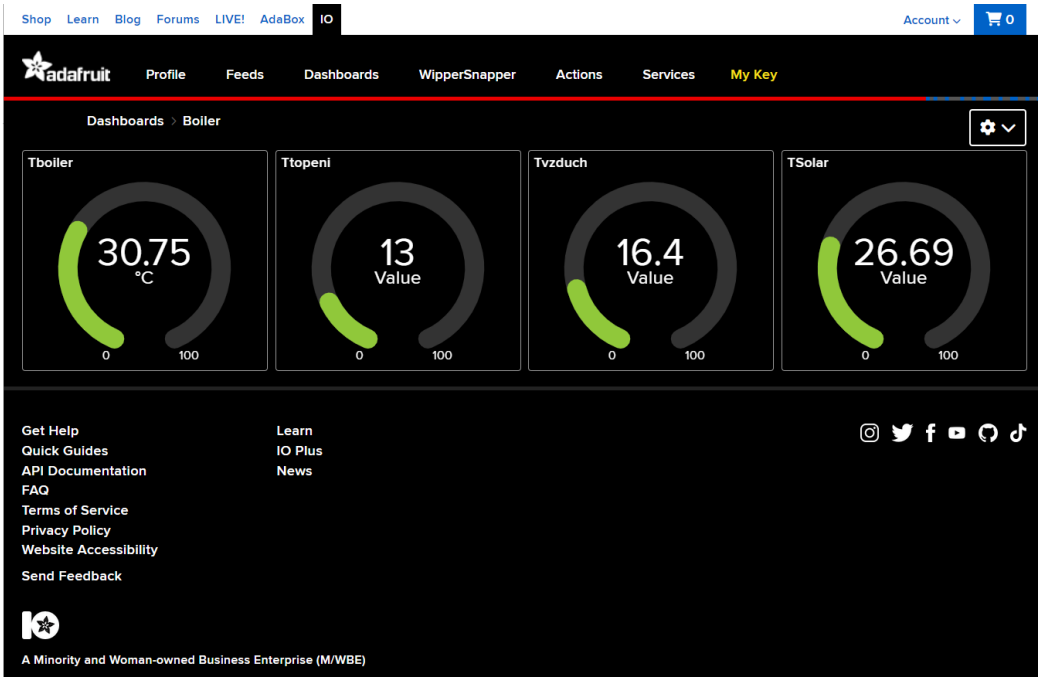



Obrázek 5.6: Schéma popisující implementaci aplikačního rozhraní centrální jednotky v jednotlivých PHP skriptech.

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.

[5] https://canvasjs.com/




Obrázek 5.7: Snímek webu Adafruit IO s naměřenými hodnotami.

## 5.5 Řídící jednotka

Pro implementaci řídící jednotky byly použity 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[6]

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 parametry řízení a novou konfiguraci pošle nazpět Arduinu nano.

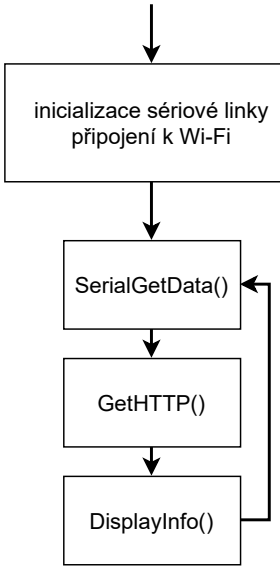
[6] https://www.arduino.cc/

### 5.5.1 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()`.




Obrázek 5.8: Diagram popisující základní smyčku mikrokontroleru ESP-8266.

**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 displeji. Data které zobrazuje jsou:

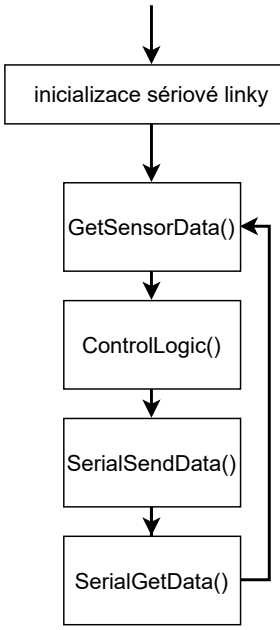
- Teplota bojleru
- Teplota teplovodního okruhu topení
- Teplota teplovodního solárních panelů
- Čas
- Stav zapnutí dohřívání bojleru vestavěným elektrickým ohříváním
- Maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení
- Prahy pro ohřev bojleru z externích zdrojů

### 5.5.2 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()`.




Obrázek 5.9: Diagram popisující základní smyčku mikrokontroleru Arduino nano.

**GetSensorData()** získá naměřená data z teplotních senzorů. Tedy teploty bojleru, teplovodního okruhu topení a teplovodního okruhu solárních panelů.

**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

**ControlLogic()** implementuje řídící logiku.

Řídící logika mikrokontroleru funguje na principu zmiňovaných rozdílů teplot bojleru a externích zdrojů. Pokud je jeden z externích zdrojů teplejší než teplota bojleru o daný teplotní rozdíl, 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. (viz schéma řídící jednotky na straně 33)

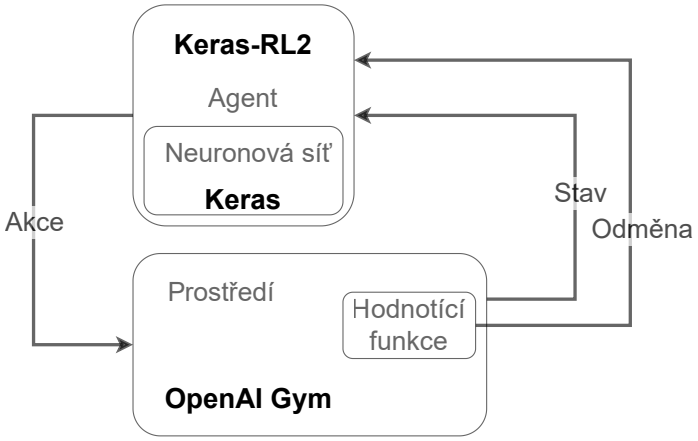
Pokud není v časovém rozmezí 15:00 až 19:00 bojler ohřátý na Maximální teplotu, na kterou se může bojler ohřívat z teplovodního okruhu topení, spustí se ohřívání vestavěným elektrickým zdrojem. Toto časové rozmezí je stanoveno podle levnějšího tarifu elektřiny tzv. „nočního proudu“.

## 5.6 Optimalizace

Pro učení agenta bylo 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.

Jako agent strojového učení bude využit agent z knihovny Keras-RL2. Agent bude využívat neuronovou síť. Hluboková neuronová síť bude vytvořena pomocí knihovny Keras.

Po učení agenta se pomocí naučené hluboké neuronové sítě získají hodnoty nových parametrů řízení - hodnoty teplotních rozdílu bojleru a teplovodních okruhů.




Obrázek 5.10: Schéma popisující které knihovny byly použity pro jednotlivé části algoritmu hlubokého Q-učení.

### 5.6.1 Prostředí a hodnotící funkce

Pro učení agenta budou implementována 2 prostředí. Obě budou implementována pomocí knihovny OpenAI Gym. Součástí těchto prostředí je také hodnotící funkce.

První prostředí je pro naučení agenta základnímu chování a bude využívat celý simulační model.

Druhé prostředí je pro samotnou optimalizaci řízení. To využívá naměřená data a simuluje pouze ohřev bojleru z externích zdrojů.

Návrhy obou prostředí jsou popsány v kapitole 4.3.1 na straně 35.

Prostředí implementuje třída `ModelEnv`, ta je potomkem třídy `Env` z knihovny OpenAI Gym. Kvůli kompatibilitě rozhraní s Agentem bude implementovat tyto 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.
- `reset()`, tato funkce uvede prostředí do výchozího stavu.

#### Prostředí pro prvotní naučení agenta

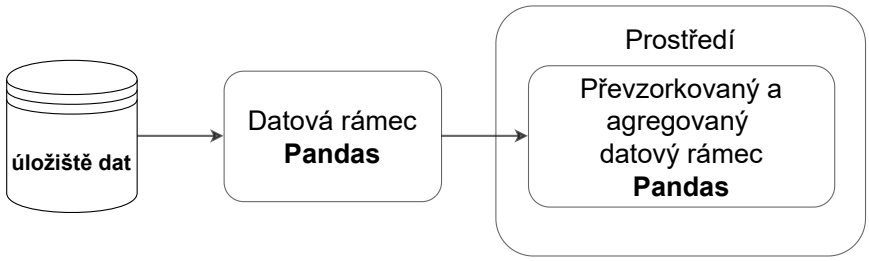
U tohoto prostředí byly oproti simulačnímu modelu pozměněny vlastnosti simulovaných tepelných zdrojů tak, že jejich výsledná hodnota výkonu poupravena o náhodné množství. Tím je zajištěn různý průběh těchto zdrojů a agent by se tak měl setkávat s větším počtem stavů.

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

#### Prostředí pro další učení agenta

Toto prostředí využívá naměřených dat a simuluje pouze ohřev bojleru z externích zdrojů. Konkrétně tedy využívá naměřené teploty teplovodních okruhů topení a solárních panelů. Tyto lze získat z úložiště centrální jednotky.

Data jsou načtena do datového rámce knihovny Pandas. Následně jsou převzorkována a agregována podle minut, jelikož jeden simulační krok odpovídá jedné minutě reálného času. Prostředí pak v každém kroku načítá data z datového rámce a simuluje ohřev bojleru.




Obrázek 5.11: Schéma popisující využití naměřených dat v prostředí agenta. Data se načítají do datového rámce knihovny Pandas. Následně jsou data převzorkovány a agregovány.

**Hodnotící funkce**

Cílem optimalizace je maximální využití externích zdrojů pro ohřev bojleru. Hodnotící funkce je tak tedy definovaná těmito pravidly:

- výchozí odměna – 0.5
- pokud je ohřev zapnutý ale externí zdroj je studenější, než bojler – odměna $-1$
- pokud došlo ke změně řízení před méně, než 10 kroky(odpovídá 10 minutám) – odměna $-1$
- bojler se ohřívá – odměna 1

### 5.6.2 Agent

Jako agent strojového učení bude využit agent z knihovny Keras-RL2, konkrétně bude instancí třídy `DQNAgent`. 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`.

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()`.

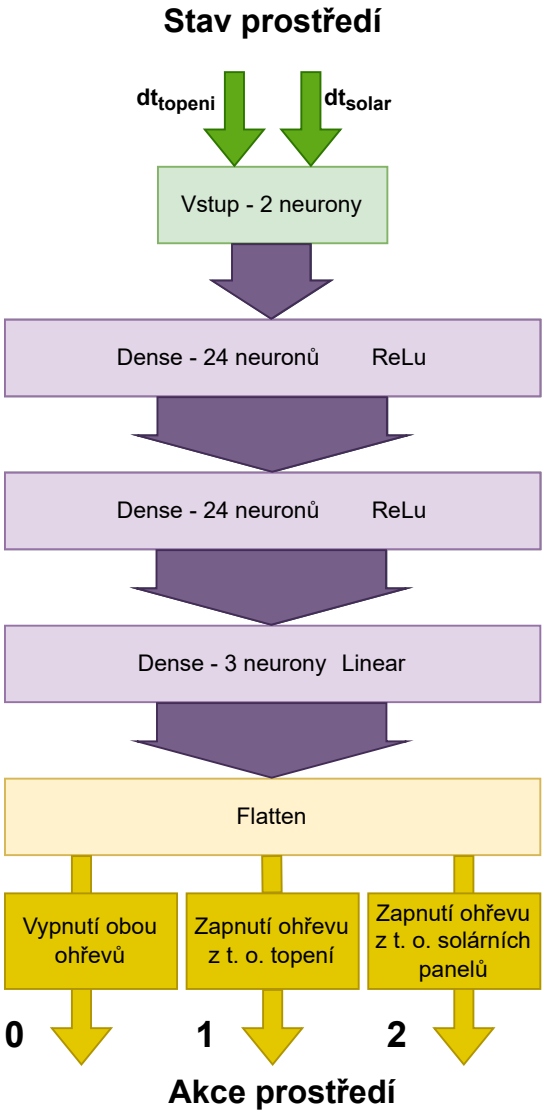
### 5.6.3 Neuronová síť

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 5.12: 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. Velikost implementované sítě byla zvolena tak, aby síť nebyla příliš rozsáhlá, ale aby dokázala danou problematiku zpracovat. Při vývoji tato síť splnila tyto předpoklady.

### 5.6.4 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()`. Uvést 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.

### 5.6.5 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 35. 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 do souboru `thresholds.json`.

### 5.6.6 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.

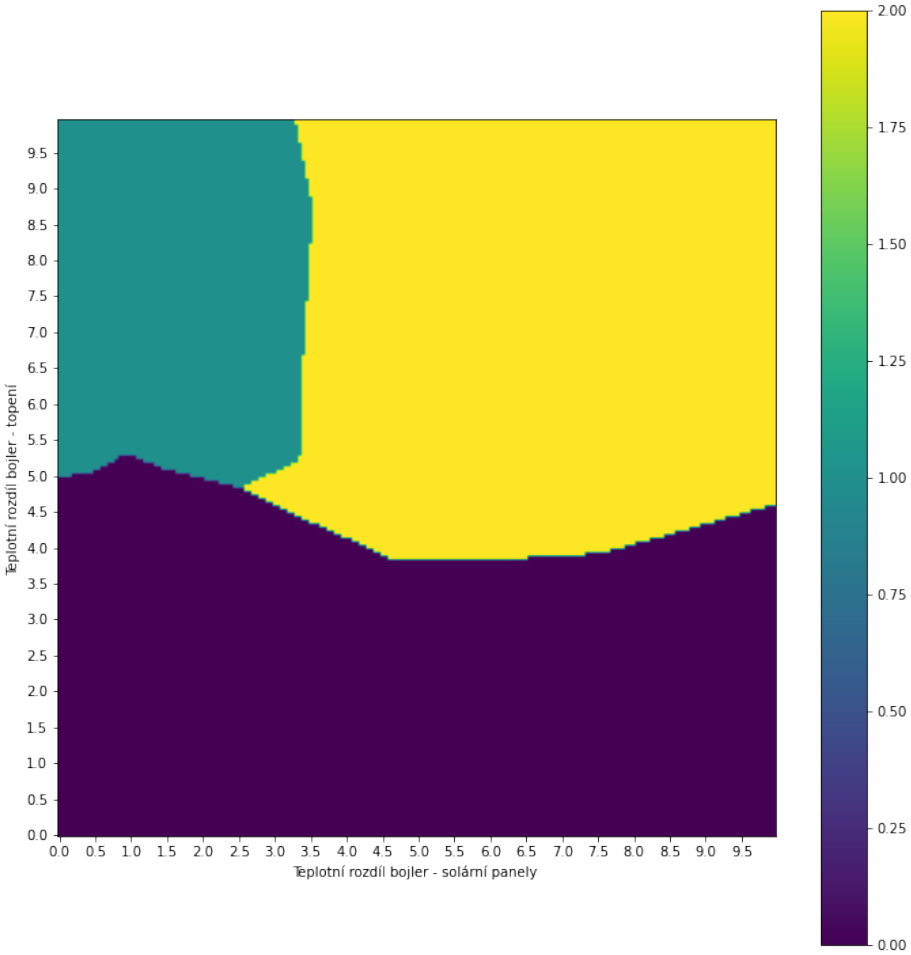
Agent načte váhy neuronové sítě ze souboru `neural_network.h5f` a provede 24.000 kroků. Následně se přepíše soubor `neural_network.h5f` nově získanými váhami neuronové sítě. Poté se získají nově nalezené prahy. Nově nalezené prahy se uloží ve formátu JSON do souboru `thresholds.json`.

# Kapitola 6

# Zhodnocení výsledků

## 6.1 Výsledky optimalizace

Výsledkem naučeného agenta je neuronová síť, která rozdělila stavový prostor.




Obrázek 6.1: Vizualizace rozdělení stavového prostoru. Hodnoty odpovídají akcím agenta, kdy 0 značí vypnutí obou ohřevů, 1 značí zapnutí ohřevu bojleru z teplovodního okruhu topení a 2 značí zapnutí ohřevu z teplovodního okruhu solárních panelů.

Jako řídící jednotka by šla tedy použít tato naučená neuronová síť. Požadavky řídícího systému a výsledek optimalizací ale dovolují zjednodušit použití neuronové sítě a získat z ní pouze 2 parametry řízení, které budou použity společně s upravenou původní logikou řídící jednotky.

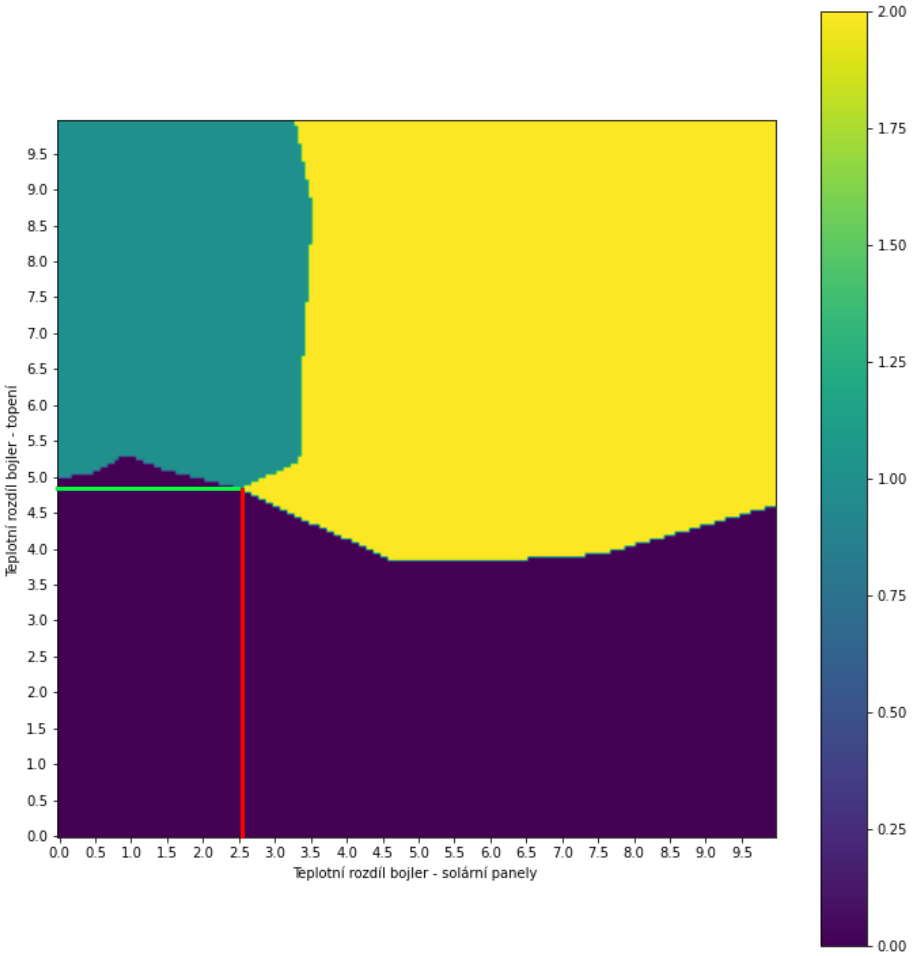
### 6.1.1 Nalezení nových parametrů řízení

Naučený agent rozdělí stavový prostor na primárně 3 části:

- oblast kdy jsou ohřevy z externích zdrojů vypnuté, protože ani jeden z externích zdrojů není dostatečně zahřátý
- oblast, kdy se bojler zahřívá z teplovodního okruhu topení
- oblast, kde se bojler zahřívá z teplovodního okruhu solárních panelů

Jelikož jsou výsledné oblasti pevně rozděleny, lze z nich získat hodnoty, kdy se zapínají jednotlivé ohřevy z externích zdrojů. Situací, kdy jsou oba externí zdroje aktivní (oba se ohřívají), nastává v reálném provozu jen pár. Tedy jen asi 14 dní v roce jsou solární panely dostatečně ohřívány sluncem tak, aby mohli bojler ohřát a zároveň je ještě dům vytápěn. Tedy lze výsledný stavový prostor zjednodušit na 2 teplotní rozdíly, pro každý tepelný zdroj jeden.

Tyto teplotní rozdíly se naleznou prohledáním stavového prostoru a nalezením nejnižší hodnoty, kdy došlo k zapnutí ohřevu z daného externího zdroje. Stavový prostor se prochází s krokem 0.05.




Obrázek 6.2: Graf stavového prostoru, který rozdělil agent. V grafu jsou naznačeny nové teplotní rozdíly, při kterých dojde k zapnutí ohřevu z externích zdrojů. Červenou barvou je naznačený teplotní rozdíl pro zapnutí ohřevu z teplovodního okruhu solárních panelů. Zelenou barvou je naznačený teplotní rozdíl pro zapnutí ohřevu z teplovodního okruhu topení.

### Optimalizované parametry řízení

Výsledkem optimalizace jsou tedy tyto teplotní rozdíly, při kterých dojde k zapnutí ohřevu bojleru z externích zdrojů.

| Teplotní rozdíl pro ohřev z t. o. topení | Teplotní rozdíl pro ohřev z t. o. solárních panelů |
|---|---|
| 4,85 | 2,6 |

Tabulka 6.1: Výsledné teplotní rozdíly pro řízení ohřevu bojleru z externích zdrojů.

Z těchto výsledků můžeme vyvodit, že agent našel optimální parametry pro řízení ohřevu bojleru. Tedy zmenšil oba původní prahy a tak dochází k dřívějšímu zapnutí ohřevu z externích zdrojů.

Výsledné parametry řízení byly ověřeny pomocí simulačních experimentů.

## 6.2 Simulační experimenty pro ověření výsledků

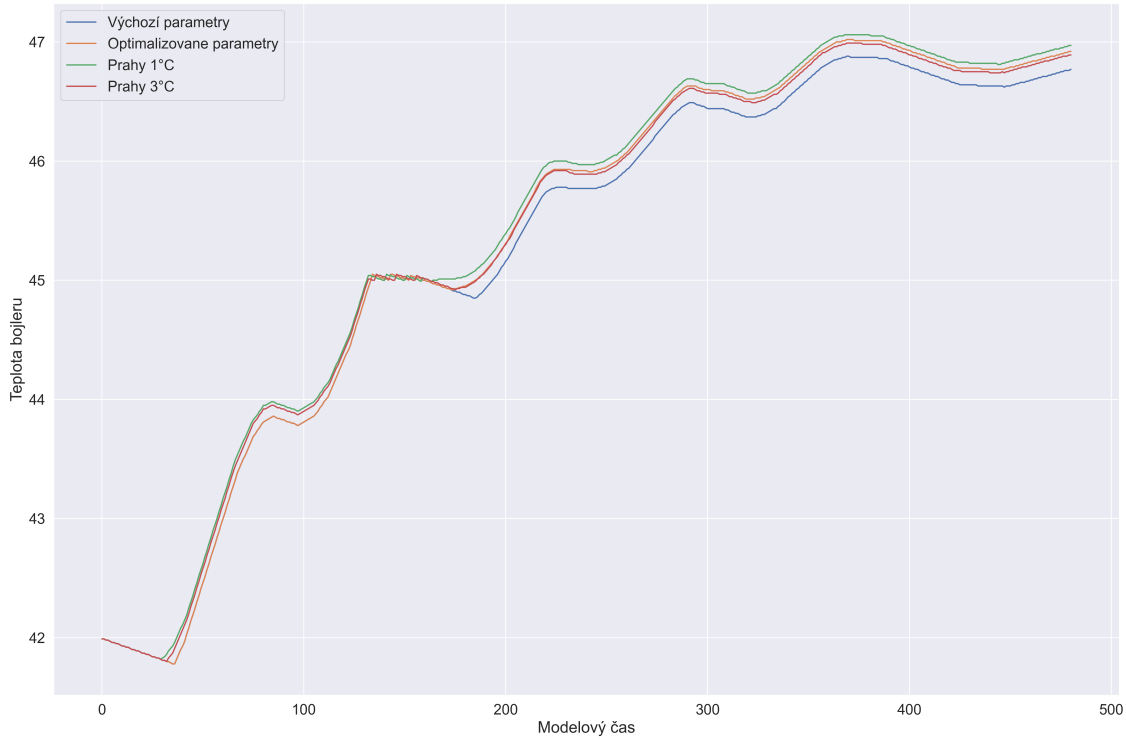
Pro simulační experimenty byl použit stejný simulační model, jako pro ověření validity modelu viz strana 19. V simulačních experimentech byly použity hodnoty prahů získané z optimalizace pomocí hlubokého Q-učení, původní hodnoty prahů a ručně zvolené prahy. Cílem experimentů je ověřit, zda výsledky nově nalezených prahů pomocí hlubokého Q-učení odpovídají očekávání.
Přehled výsledků simulačních experimentů:

| **Typ** | $\mathbf{dt_{topeni}}$ | $\mathbf{dt_{solar}}$ | $\mathbf{T_{max}}$ | **Čas** | **N** | **M** |
|---|---|---|---|---|---|---|
| Původní prahy | 5 | 5 | 46,8761 | 369 | 3 | 1 |
| Optimalizovane prahy | 4,85 | 2,6 | 47,0178 | 369 | 3 | 1 |
| Ručně zvoleno | 3 | 3 | 46,9901 | 370 | 4 | 1 |
| Ručně zvoleno | 1 | 1 | 47,0636 | 369 | 5 | 1 |

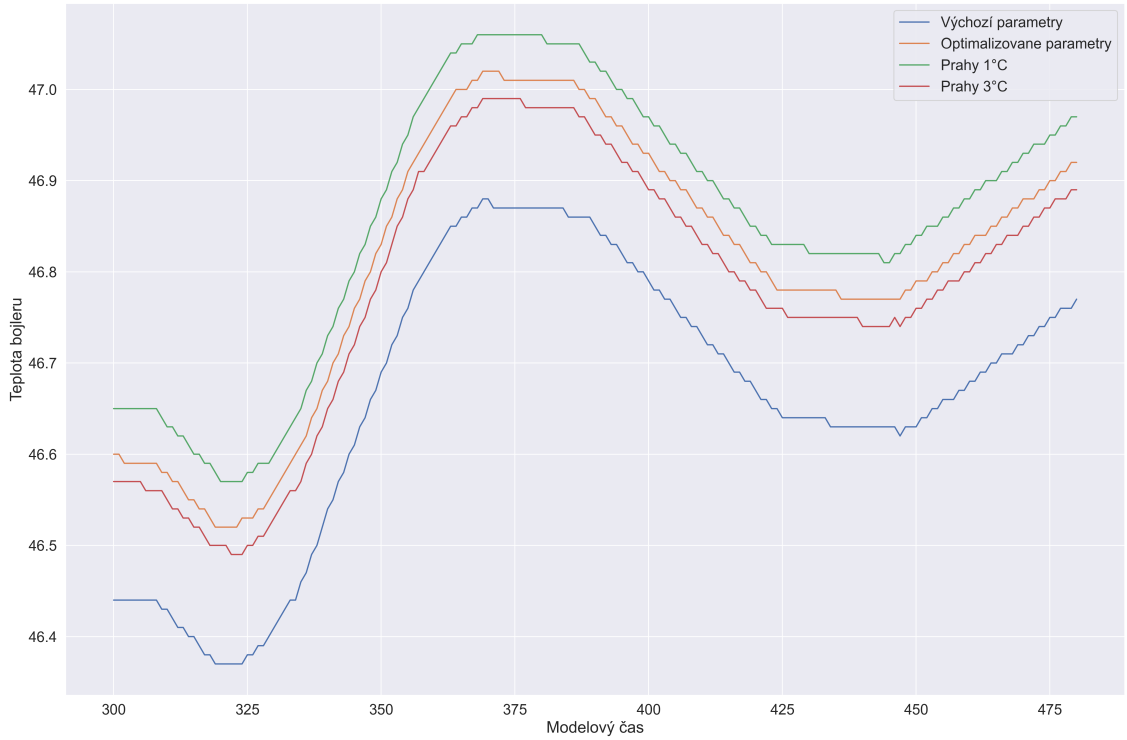
Tabulka 6.2: Výsledky simulačních experimentů. $dt_{topeni}$ odpovídá teplotnímu rozdílu, při kterém se zapne ohřev z teplovodního okruhu topení. $dt_{solar}$ odpovídá teplotnímu rozdílu, při kterém se zapne ohřev z teplovodního okruhu solárních panelů. $T_{max}$ odpovídá maximální teplotě bojleru, na 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 optimalizace odpovídají očekávání. Výsledné teplotní rozdíly vedly k vyšší teplotě bojleru při stejném počtu zapnutí ohřevů. V reálném nasazení by tento výsledek nejspíše vedl k optimálnímu 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 (4,85 a 2,6) od naučeného agenta jsou optimální, než původní prahy.




Obrázek 6.3: Graf porovnání průběhu teplot bojleru v simulačních experimentech. Lze na něm pozorovat průběh ohřevu bojleru s různými parametry ohřevu. Největší rozdíl můžeme pozorovat zhruba po čase 150, kdy se průběhy teplot nejvíce rozejdou. Na grafu lze pozorovat, že optimalizované parametry řízení vedou téměř k nejvyšší teplotě – oranžová křivka. Ručně zvolené parametry, rozdíly teplot při kterých se zapne ohřev, $1°C$ sice vedou k o trochu vyšší teplotě, ale dochází při nich k častému zapínání a vypínání ohřevu, což není žádoucí – zelená křivka. Detaily porovnání experimentů naleznete v tabulce 6.2




Obrázek 6.4: Detailní pohled na část grafu (obrázek č. 6.3), který ukazuje cílové teploty bojleru. Na nich můžeme pozorovat, že optimalizované parametry řízení vedou k větší teplotě, než prahy původní. Vyšší teploty bylo dosaženo teplotními rozdíly, při kterých se spouští ohřev, $1°C$. Rozdíl je ale příliš malý v porovnání s navýšením počtem zapnutí a vypnutí ohřevu – viz tabulka 6.2

# Kapitola 7

# Závěr

V této práci jsem navrhl a popsal realizaci řídícího systému pro ohřev vody v bojleru z externích zdrojů. Pro nalezení optimálních 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 parametry 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. Řídící systém je navržený tak, že je možné k optimalizaci využití externích zdrojů pro ohřev bojleru použít i jiné optimalizační metody.

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] Brownlee, J. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning* [online]. 2017 [cit. 17.7.2022]. Dostupné z: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[2] 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.

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

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

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

[6] Omasta, M. R. *Molekulová fyzika a termika* [online]. 2015 [cit. 24.6.2022]. Dostupné z: https://publi.cz/books/264/06.html.

[7] Omasta, M. R. *Molekulová fyzika a termika* [online]. 2015 [cit. 24.6.2022]. Dostupné z: https://publi.cz/books/264/07.html.

[8] 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/.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

# 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

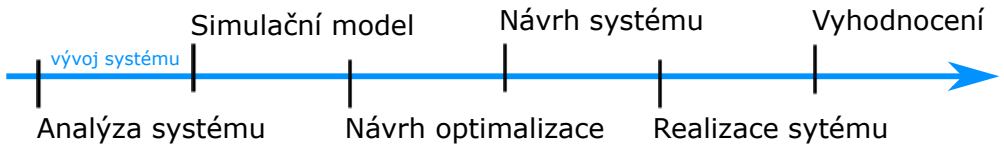
# Příloha B

# Plakát

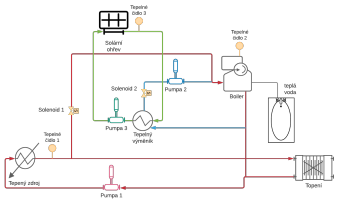
# NÁVRH SYSTÉMU ŘÍZENÍ DISTRIBUCE TEPLA

Jan Tomeček
xtomec09

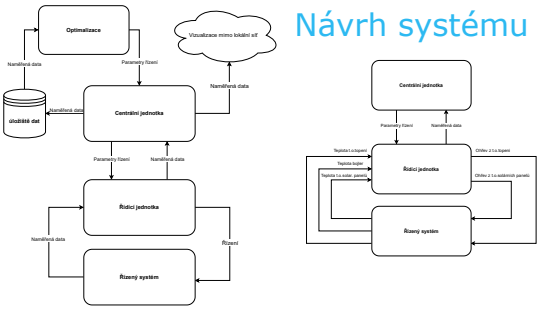
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



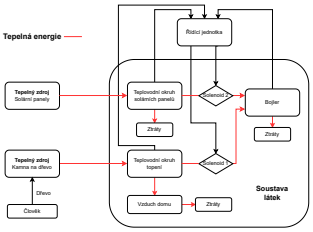

## Anyláza



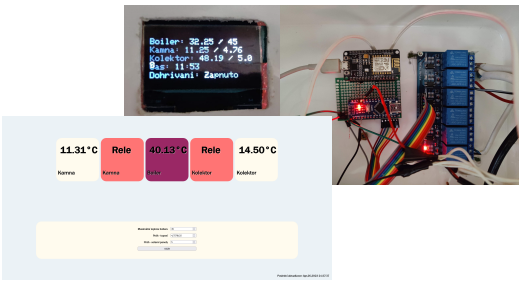
## Návrh systému



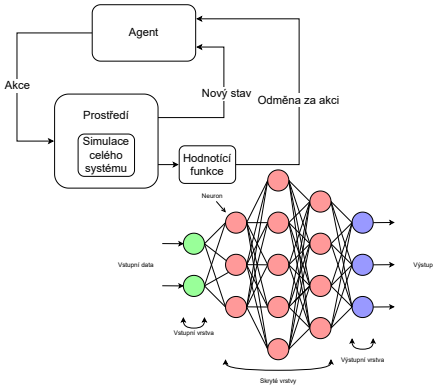

## Simulační model



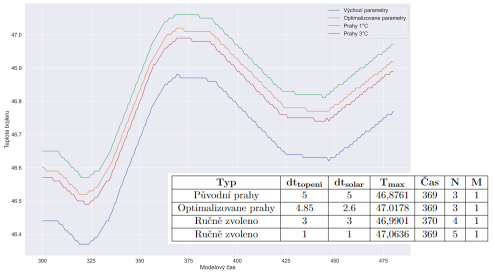

## Realizace systému




## Návrh optimalizace



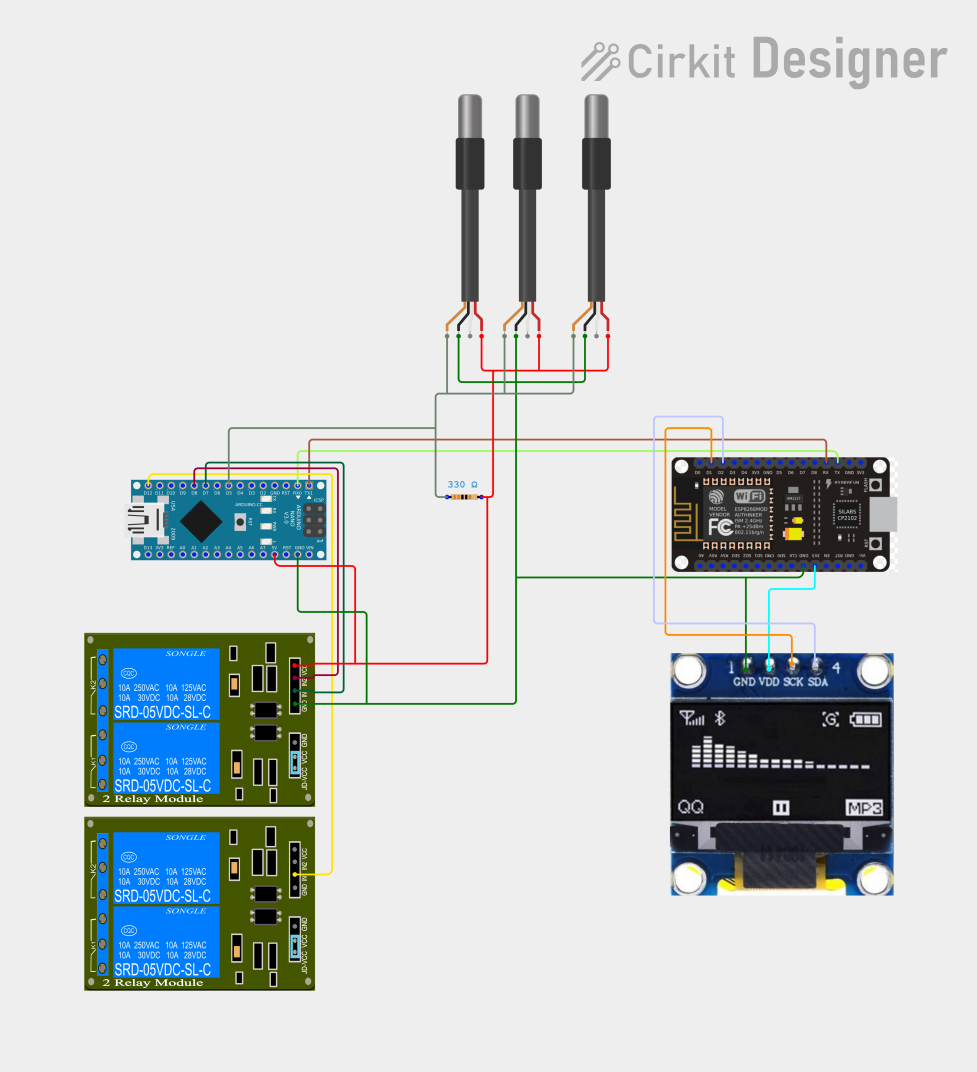

## Vyhodnocení optimalizace




| Typ | $dt_{topeni}$ | $dt_{solar}$ | $T_{max}$ | Čas | N | M |
|---|---|---|---|---|---|---|
| Původní prahy | 5 | 5 | 46,8761 | 369 | 3 | 1 |
| Optimalizovane prahy | 4.85 | 2.6 | 47,0178 | 369 | 3 | 1 |
| Ručně zvoleno | 3 | 3 | 46,9901 | 370 | 4 | 1 |
| Ručně zvoleno | 1 | 1 | 47,0636 | 369 | 5 | 1 |

# Příloha C

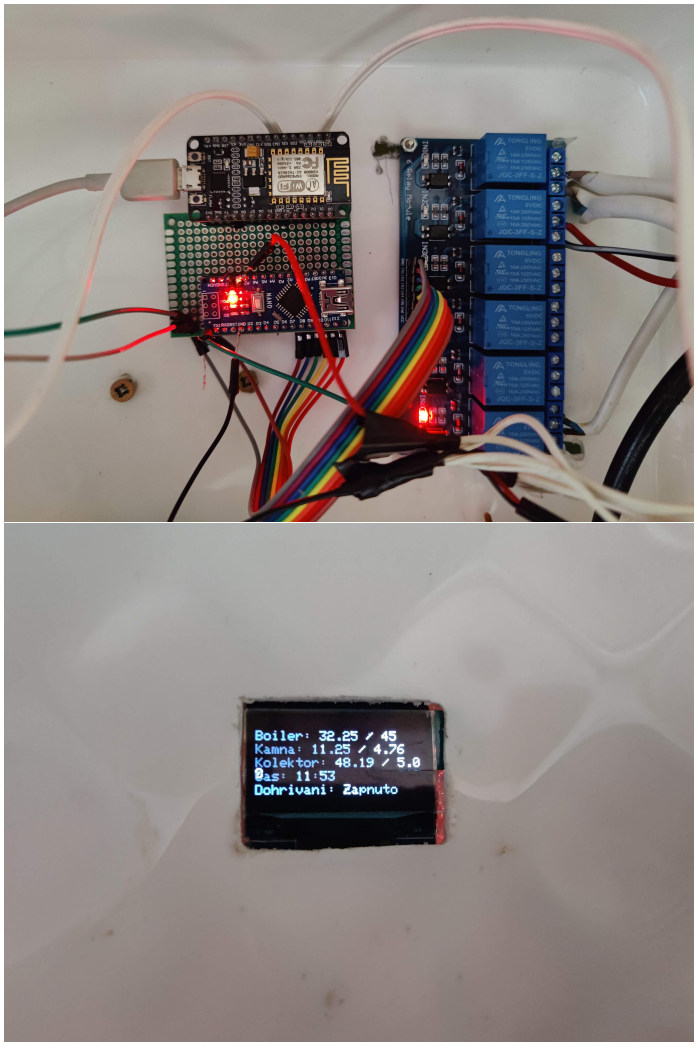
# Schéma zapojení mikrokontrolerů




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.