VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

Z HLOUPÉHO BOJLERU CHYTRÝ POMOCÍ CHYTRÉ ZÁSUVKY

FROM A DUMB BOILER TO A SMART ONE USING A SMART SOCKET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JACEK MITURA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ZDENĚK MATERNA, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Mitura Jacek**
Program: Informační technologie
Název: **Z hloupého bojleru chytrý pomocí chytré zásuvky**
From a Dumb Boiler to a Smart One Using a Smart Socket
Kategorie: Vestavěné systémy

Zadání:

1. Proveďte rešerši existujících řešení zaměřených na úsporu tepla při elektrickém ohřevu užitkové vody.
2. Vyberte vhodnou chytrou zásuvku umožňující lokální řízení a získávání dat o aktuální spotřebě s dostatečným rozlišením.
3. Navrhněte řízení bojleru na základě historických dat o spotřebě s cílem minimalizovat spotřebu elektrické energie a tepelné ztráty.
4. Implementujte navržené řešení.
5. Proveďte testování a odhadněte docílenou úsporu.
6. Zdrojové kódy a dokumentaci publikujte na GitHubu.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle doporučení vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Materna Zdeněk, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

V dnešních časech se velmi rychle rozrůstá svět chytrých zařízení. Přesto pořídit chytré zařízení pro aplikaci v oblasti ohřevu vody zůstává nadále obtížné. Mezi obvykle důvody komplikující tuto skutečnost patří chybějící možnost napojení na centrální řídící jednotku, nebo nutnost výměny celého bojleru za nový chytrý model. V obou případech je pak dodatečým záporem vyšší cena takového řešení. Proto je cílem této bakalářské práce, vytvořit komplexní a cenově dostupné řešení, zajišťující konverzi stávájícího bojleru na chytrý pomocí chytré zásuvky a centrálního serveru řídícího domácnost. Řešení využívá denní předpověď spotřeby postavenou na základu exponenciálního klouzavého průměru, aby tím dosáhlo úspory elektrické energie.

Abstract

Nowadays, the world of smart devices is growing very fast. Nevertheless, it remains difficult to acquire smart device for water heating applications. The usual reasons complicating this fact include the lack of the possibility of connection to the central control unit, or the need to replace the entire boiler with a new smart model. In both cases, the additional disadvantage is the higher price of such a solution. Therefore, the aim of this bachelor's thesis is to create a comprehensive and affordable solution, ensuring the conversion of an existing boiler to smart using a smart socket and a central server controlling the household. The solution uses a daily consumption forecast based on an exponential moving average to save electricity.

Klíčová slova

chytrá domácnost, chytrá zařízení, chytrá zásuvka, bojler, automatizace, ESP8266, DS18B20, InfluxDB, řídící algoritmus, IoT

Keywords

smart home, smart devices, smart plug, boiler, automation, ESP8266, DS18B20, InfluxDB, control algorithm, IoT

Citace

MITURA, Jacek. *Z hloupého bojleru chytrý pomocí chytré zásuvky*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zdeněk Materna, Ph.D.

Z hloupého bojleru chytrý pomocí chytré zásuvky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zdeňka Materny Ph.D. a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jacek Mitura
5. května 2021

Poděkování

Rád bych poděkoval za odbornou pomoc, konzultace a postřehy udělené mým vedoucím panem Ing. Zdeňkem Maternou Ph.D., čím mi pomohl ubírat se správným směrem při vývoji této práce.

Obsah

Kapitola 1

Úvod

Zhruba 34.5% domácností v ČR využívá k ohřevu teplé užitkové vody elektrickou energii [35]. V této energetické skupině většinu, z pohledu typu ohřívačů, stanoví akumulační ohřívače vody jiným slovy klasické bojlery.

Avšak takový bojler ohřívá vodu na předem danou teplotu, kdykoliv je zaznamenán její pokles pod nastavenou mez. Což z hlediska tepelných ztrát není zcela optimální řešení. Zde se nabízí propojení chytré domácnosti a klasického bojleru, který by byl řízen centrálním systémem starajícím se o dům. Systémem, který umí rozpoznat, kdy je člověk doma přítomen a tím pádem mimo jiné např. regulovat vnitřní teplotu a ohřev vody. Tudíž by zajišťoval teplou vodu jen a pouze v časech, kdy je ji zapotřebí, čímž by se eliminovala značná část energetické spotřeby způsobené konstantním ohřevem vody na konkrétní teplotu. Je samozřejmostí, že takováto chytrá zařízení, zde konkrétně bojlery, již v kombinaci s chytrou domácností existují. Problémem však je, že tyto chytré bojlery jsou často dost drahé. Nebo v případě, kdy ze stávajícího chceme udělat chytrý pomocí, na trhu dostupného chytrého termostatu. Jelikož chytrý termostat dané značky je většinou určený pouze pro bojler téže značky. Nevzpomínaje fakt, že tyto řešení často postrádají možnost efektivního zapojení do centrálního řízení chytré domácnosti.

Cílem této práce je vytvořit kompletní řešení pro transformaci klasického bojleru na chytrý, které bude levné z hlediska pořízení a zároveň bude přinášet úsporu energie využité na ohřev vody. Rovněž jej bude možné použít bez ohledu na výrobce bojleru. Mezi hlavní rysy tohoto projektu patří použití chytré zásuvky pro spínání a bezdrátového modulu pro snímaní teplot. Neopomíjeje, že řídící systém bude umístěn na centrálním počítači, aby skýtal možnost integrace do chytré domácnosti, jako celku. Vše s ohledem na jednoduchost, a co největší dosažitelnou úsporu při přípravě teplé vody pro domácnost.

Práce je pomocí kapitol rozčleněna do několika hlavních části. Představení důležitých pojmů a teorie v kapitole 2. Popis momentálně dostupných řešení kapitola 3. Návrh řešení problému 4 a technologie v něm použité 5. Popis problematiky implementace 6. A na závěr neméně důležitá kapitola 7, o průběhu testování v provozu a dosažených úsporách.

Kapitola 2

Důležité pojmy a teorie

V této kapitole jsou vysvětleny důležité termíny a představena teorie, což je nezbytné pro porozumění dalších části této práce. Velmi podstatné je pochopení konceptu *IoT*(popisuje kapitola 2.1), který se hojně využívá v oblasti *smart home* technologii. Další kapitoly vysvětlují *MQTT protokol* 2.2 pro komunikaci zařízení, termodynamické fyzikální zákony 2.3 objasňující výpočty a statistické metody 2.5 potřebné k odhadu vývoje spotřeby.

2.1 IoT - Internet of Things

Zkratka *IoT* z anglického *Internet of Things*, jinak *Internet věcí* v informatice popisuje síť fyzických objektů tzv. věcí vybavených senzory, softwarem a jinými technologiemi pro účel propojení a výměny dat s jinými zařízení nebo systémy prostřednictvím Internetu [3].

Zařízení v domácnosti jsou propojená například pomocí routeru, switche nebo jiné technologie pro zajištění vzájemné komunikace. Data pak vyhodnocuje přímo konkrétní zařízení nebo systém jemu dostupný např. na centrálním serveru domácnosti. *Internet of Things* může být v jisté části také spjatý s *AI* [1], která ku příkladu může tvořit vyhodnocovací a rozhodovací vrstvu celého systému.

IoT se využívá v mnoha prostředích od výrobních procesů přes automobily až k všední domácnosti. Kde v ideální situaci by mezi sebou komunikovala většina elektronických spotřebičů v domě tvoříc dohromady jeden komplexní organismus. Ku příkladu by dům byl schopen předpokládat, se vrátíte domů a zvýšit tak teplotu uvnitř objektu nebo byste obdrželi zprávu, kterou by předala chytrá lednička řídicímu systému, a ten by napsal, že došlo mléko. To všechno díky komunikaci mezi stroji a možnosti předání této informace i mimo lokální síť.

2.2 MQTT

Message Queuing Telemetry Transport zkratkou *MQTT* patří k nejvýznamnějším komunikačním protokolům v rámci IoT systému. Jedná se o aplikační protokol, jehož hlavní předností je malý datový objem hlavičky, a také možnost komunikovat v sítích o omezené datové propustnosti. Zařazujeme jej do skupiny protokolů *Publisher-Subscriber*, jehož principem je výměna zpráv mezi účastníky dvou druhů, *subscriber* (odebíratel) jenž se přihlašuje k odběru zpráv daného tématu(*topic*), *publisher* (vydavatel) který posílá zprávy s určitým tématem (*topic*).[4]

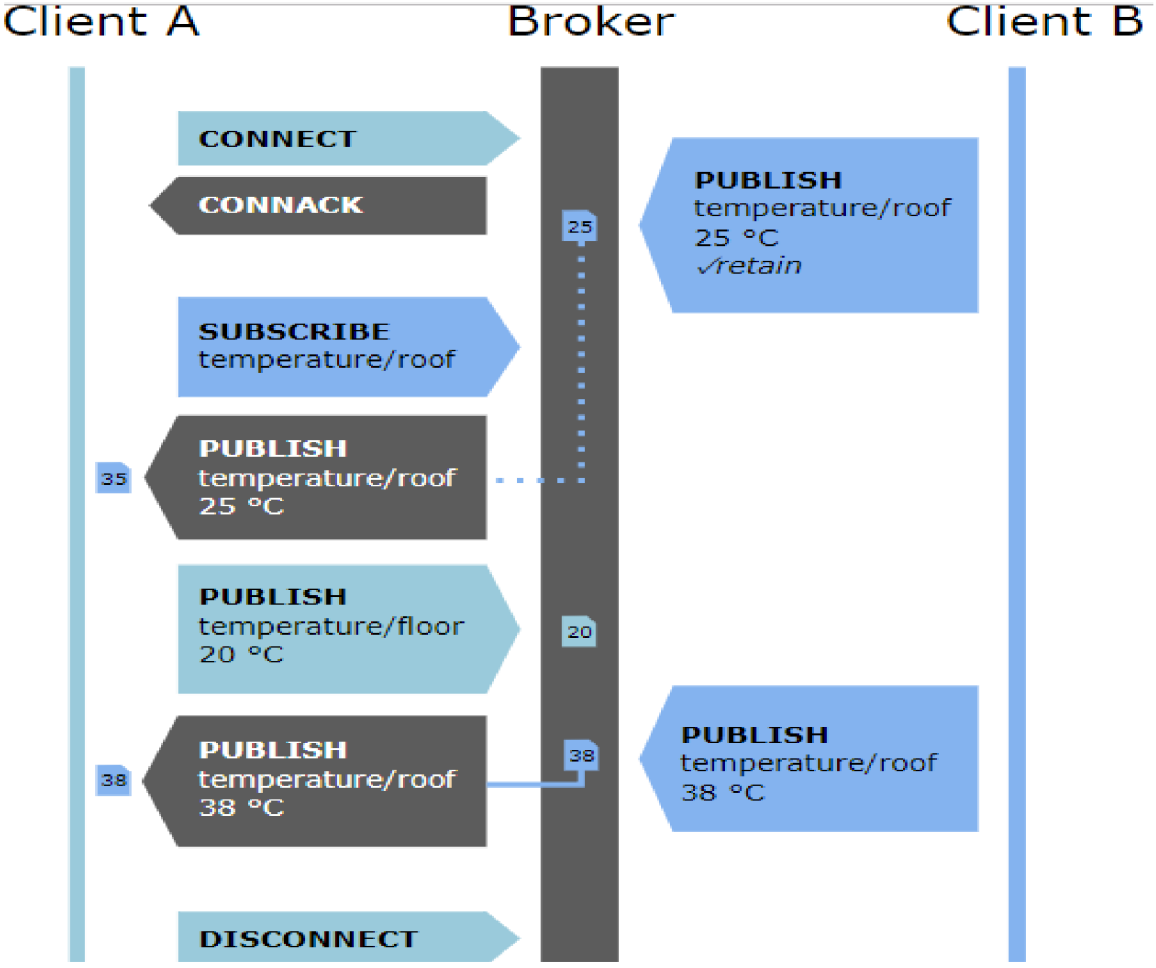
[1] **AI** - artificial intelligence, umělá inteligence

MQTT broker

MQTT broker je software, nasazený lokálně na PC nebo v cloudu, starající se o distribuci zpráv, které zasílají různá zařízení. Plní funkci prostředníka mezi účastníky komunikace typu *publisher/subscriber*. Zařízení které chce využívat *MQTT* protokolu pro komunikaci, musí definovat adresu *brokeru*, jež bude zprostředkovávat zprávy. Přijde-li na *MQTT broker* zpráva s daným tématem *topic*, ten nahlídne, jaké zařízení jsou přihlášené k danému tématu jako *subscriber* a zprávu jim přepošle. Touto architekturou zaniká nutnost, aby zařízení uchovávalo *IP adresy* jiných. Zařízení muže odesílat i přijímat zprávy pro daný *topic*.[4]

Zprávy a jejich typy

Protokol používá sadu různých zpráv pro zajištění běhu. Na obrázku 2.1 vidíme příkladovou komunikaci pro přihlášení k odběru tématu. Jednotlivé typy zpráv můžeme vyčíst z tabulky 2.1 na odkazu [5].



Obrázek 2.1: MQTT posloupnost komunikace. Převzato z [29]

QoS - Quality of service

Každé připojení k *brokeru* může specifikovat zajištění kvality služeb pro zasílaní zpráv.

- **0 (nejvýše jednou)** - zpráva se zašle pouze jednou a nedochází k zasílaní potvrzení o doručeni
- **1 (alespoň jednou)** - zpráva se zasílá příjemci do doby než je úspěšně potvrzeno její doručení
- **2 (přesně jednou)** - zajišťuje se že příjemce obdrží pouze jednou kopii zprávy(pomocí dvouúrovňového handshake)

Bezpečnost

Použitím *MQTT brokeru* lze v *Internet of Things* zajistit jistou míru bezpečnosti a to aby se neautentizovaný účastník nemohl přihlásit k odběru zpráv a taktéž je zasílat. Čím by případně mohl získat kontrolu nad děním v systému. *MQTT broker* proto poskytuje různé možnosti zabezpečení:

- Autentizace uživatelským jménem a heslem
- Port 8883 pro TLS připojení

2.3 Termodynamika pro teplotní výpočty

Pro potřebu budoucích výpočtů spojených s určením množství spotřebované teplé vody na základě rozdílu teplot a také výpočtů energie potřebné pro změnu teploty o X stupňů Celsia, je potřeba uvést základní termodynamické zákony. Právě jejich principy se využívají i v rámci kalorimetrických rovnic. Vypracováno na základě zdrojů [27], [15].

První termodynamický zákon

Je verzí *zákonu o zachování energie (energie může být transformována, nemůže však zaniknout nebo být vytvořena.)* adaptovanou pro termodynamické procesy v níže uvedeném znění.

Věta 1 V procesu bez přenosu hmoty, změna vnitřní energie ΔU je u termodynamického systému rovna energii přijatou formou tepla Q, mínus termodynamická práce W, vykonána tímto systémem na okolním prostředí.

$$\Delta U = Q - W \tag{2.1}$$

V případě kdy dochází i k přenosu hmoty nutno dodat bližší specifikaci.

Věta 2 S patřičným ohledem na základní referenční stavy systému, když dva takové systémy s potenciálně různým chemickým složením, které jsou izolované od okolí a mezi sebou oddělené pomocí nepropustné stěny spojíme do jednoho nového systému, termodynamickou operací odstranění stěny, pak platí vztah:

$$U_0 = U_1 + U_2 \tag{2.2}$$

kde U_0 značí vnitřní energii kombinovaného systému a U_1, U_2 vnitřní energie jednotlivých oddělených systému.

Druhý termodynamický zákon

Druhý termodynamický zákon určuje přirozený směr, jakým procesy probíhají. Je empirický a pravděpodobnostní, a směr vývoje systému charakterizuje pomocí *entropie*. Jeho význam muže být vyjádřen v následující větě.

Věta 3 Teplo nemůže samovolně proudit z chladnějšího tělesa do teplejšího.

Entropie pak narůstá až do svého maxima, které je dosaženo v momentě, kdy systém dojde do termodynamické rovnováhy a jeho nehomogennost prakticky vymizí.

Kalorimetrická rovnice

Kalorimetrickou rovnicí můžeme popsat situaci, kdy do izolované nádoby umístíme $1, 2, ..., n$ těles o hmotnostech m_1 až m_n s měřenou tepelnou kapacitou c_1 až c_n a teplotami T_n. Za předpokladu, že dané tělesa mezi sebou nereagují a nedojde během výměny k změně stavu skupenství. V takovém případě bude tepelná výměna probíhat tak dlouho, dokud nenastane rovnovážný stav, při němž se teploty těles vyrovnají na výslednou teplotu T (větší než minimální teplota T_i a menší než maximální teplota T_j). Z prvního a druhého termodynamického zákona pak vyplývá, že úbytek vnitřní energie těles o vyšší teplotě je roven přírůstku vnitřní energie těles o teplotě nižší, jelikož platí vztah 2.2 a zároveň 2.2, kde $W = 0$. Teplo $Q_i = m_i c_i (T_i - T)$ které odevzdá těleso, se rovná teplu $Q_j = m_j c_j (T_j - T)$ přijatému tělesem chladnějším.

$$\sum_{i=1}^{n} c_i m_i T_i = (\sum_{i=1}^{n} c_i m_i) T \tag{2.3}$$

Obecně tedy platí, že *teplo odevzdané jedním tělesem (teplejším) je stejné, jako teplo přijaté tělesem druhým (chladnějším)* $Q_{odevzdane} = Q_{prijate}$. Tudíž můžeme odvodit vztah pro výslednou ustálenou teplotu $\boldsymbol{T}$ z rovnice 2.3, který bude ve tvaru 2.4.

$$T = \frac{\sum_{i=1}^{n} c_i m_i T_i}{\sum_{i=1}^{n} c_i m_i} \tag{2.4}$$

Práce a teplo

Tyto vzorce uplatníme u výpočtu doby potřebné pro ohřev vody na jmenovanou teplotu v dalších částech této práce.

V ideálním případě, kdy nedochází k tepelným ztrátám v důsledku přenosu tepla do okolí, lze změnu teploty tělesa definovat změnou jeho vnitřního tepla Q, tudíž platí vztah

$$Q = mc\Delta T \tag{2.5}$$

kde Q je teplo, které musí těleso přijmout/odevzdat aby změnilo svojí teplotu o rozdíl ΔT, přičemž je třeba tento rozdíl vynásobit měřenou tepelnou kapacitou c, to jest energii potřebnou pro změnu teploty tělesa o hmotnosti $1kg$ o 1 stupeň Celsia a rovněž jeho hmotností m.

Dále, můžeme vyjádřit čas t, definující délku trvání této změny při změně vnitřního tepla o Q, v důsledku dodání energie ze zdroje s výkonem P, pracujícího s jistou efektivitou η. To pomocí vzorce

$$t = \frac{Q}{P\eta} \tag{2.6}$$

opět za předpokladu, že nedochází ke ztrátám během přenosu a vůči okolnímu prostředí.

2.4 Sdílení tepla

Pro potřeby této konkrétní práce je třeba využít jak sdílení tepla vedením, tak sdílení tepla prouděním, jelikož prostředí nádrže bojleru představuje styk tekutiny s pevným tělesem. Takovouto skutečnost lze popsat prostupem tepla, zde konkrétně válcovou stěnou (objasňuje další podkapitola).

Šíření tepla rozdělujeme do více typů, dle fyzikálních principů pomocí nichž se teplo v prostoru šíří. To jest na sdílení tepla vedením (kondukce), sdílení tepla prouděním tekutiny (proudění) a sdílením tepla zářením.

Sdílením tepla vedením chápeme přenos tepelné energie v důsledku mikropohybu molekul, jejichž kinetická energie je úměrná termodynamické teplotě T. Molekuly s vyšší teplotou předávají během srážek přebytek kinetické energie molekulám s teplotou nižší. Teplota se tímto způsobem šíří zejména v tuhých tělesech.

Sdílení tepla prouděním tekutin značí transport tepelné energie mikropohybem molekul a jejich shluků, kde v důsledku viskozity si molekuly s vyšší teplotou při víření vyměňují místo s molekulami s teplotou nižší, a tím přenášejí tepelnou energii z oblasti vyšší teploty do oblasti teploty nižší v rámci tekutin.

Sdílení tepla zářením je v podstatě elektromagnetické vlnění, jehož energie je úměrná čtvrté mocnině termodynamické teploty T molekul hmotnosti. Zdrojem tohoto záření je hmotnost, ale energie se v prostoru šíří jako energie elektromagnetických vln. Proto je tento způsob jediný přípustný během transportu energie vakuem.

Teplo se šíří mezi tělesy s vyšší teplotou na ty s teplotou nižší všemi třemi způsoby, zpravidla však jeden z nich zásadně převyšuje ostatní. Výjimky nastávají jen v případech extrémně vysokých teplot a tlaků. Předpokládáme tudíž, že v tuhých tělesech se teplo šíří pouze vedením a v případě tekutin pouze prouděním. V případě záření transport tepelné energie roste s rostoucí termodynamickou teplotou T [32].

Prostup tepla jednoduchou válcovou stěnou

Jednoduchou válcovou stěnou rozumíme stěnu složenou pouze z jednoho materiálu, což znamená, že je homogenní a není třeba uvažovat rozdílné tepelné vlastnosti materiálu. Z pohledu vedení tepla máme několik faktorů mimo materiál stěny jež jej ovlivňují ve značné míře. Prvním z nich je prostředí na vnější straně stěny, to jest teplota, pohyb, a druh okolního media. Druhým je zmíněný materiál stěny a jeho tloušťka. Dalším pak medium z vnitřní strany stěny, o kterém je potřeba znát stejné vlastnosti jako pro vnější.

Tyto informace jsou využity v rovnici 2.7, jež vyjadřuje teplo sdělené 1m válcové stěny při uvážení všech metod sdílení tepla tj. jeho prostup.

$$q = k(T_1 - T_2) = \frac{\pi(T_1 - T_2)}{\frac{1}{\alpha_1 d_1} + \frac{1}{2\lambda} ln(\frac{d_1}{d_2}) + \frac{1}{\alpha_2 d_2}} \tag{2.7}$$

Tu lze pak převést do tvaru 2.8, který vyjadřuje teplotu vnější stěny T_{s2} při vynechaní činitele vnějšího prostředí.

$$T_{s2} = T_1 - \frac{q}{\pi}(\frac{1}{\alpha_1 d_1} + \frac{1}{2\lambda} ln(\frac{d_1}{d_2})) \tag{2.8}$$

Kde tato teplota závisí na teplotě vnitřního media T_1, množství sděleného tepla q a sumě tepelného odporu R, která sestává s tepelného odporu válcové stěny a odporu na styku tekutiny se stěnou. Analogicky sečtenou jako u série rezistorů. Kde d_1 je vnitřní průměr stěny, d_2 průměr stěny vnější, λ součinitel tepelné vodivosti a α_1 součinitel přestupu tepla.

Přičemž součinitel tepelné vodivosti λ závisí především na druhu materiálu a teplotě. Za teplotu se bere aritmetický průměr mezi povrchovými teplotami stěn. Součinitel pro takovou teplotu je k nalezení ve fyzikálních tabulkách.

Pro součinitel přestupu tepla α_1 se situace značně komplikuje, jelikož závisí jak na fyzikálních vlastnostech tekutiny, tvaru obtékaného tělesa, tak i směru a rychlosti proudění tekutiny. K vyjádření jeho hodnoty je proto potřeba množství komplikovaných kriterii.

V situaci systému jež popisuje tato práce, by vznikla rovnice o vícero neznámých tj. α_1, T_{s2} a q, proto je vhodné použít alternativní řešení, rozvinuto v kapitole 4.5.

2.5 Statistické metody

Statistické metody jsou metody zabývající se uspořádáním, analýzou, interpretací a prezentováním dat. Při aplikaci statistiky na nějaký problém se zpravidla vychází ze statistické populace (skupina objektů) nebo statistického modelu, který muže být rovněž využit pro předpověď budoucího vývoje dat, na základě kterých je sestaven. [30]

Klouzavý průměr (Moving average - MA)

Ve statistice je klouzavý průměr typem výpočtu pro analýzu řady bodů, pomocí vytvoření sérii průměrů podmnožin vytvořených z celé řady. A patři mezi typy filtru s konečnou impulzní odezvou. Pro danou série bodů a podmnožinu o statické velikosti, se první element MA získá vypočtením průměrné hodnoty pro první z řady podmnožin. Následně je podmnožina modifikovaná posunutím okna statické velikosti, kdy dojde k vyloučení prvního prvku podmnožiny a dodání následujícího prvku ze série dat.

Běžné využití klouzavého průměru je v kombinaci s časovou sérií dat pro vyhlazení krátkodobých výkyvů a zvýraznění dlouhodobého trendu nebo cyklu. Prah mezi krátkodobosti a dlouhodobosti závisí na konkrétní aplikaci a patřičně zvolených parametrech nebo typu klouzavého průměru.

Existuje vícero adaptací klouzavého průměru, ale zde jsou uvedeny pouze některé o nichž se pojednává v dalších částech práce.

Jednoduchý klouzavý průměr (Simple moving average - SMA) je neváženým průměrem předchozích k hodnot z série dat obsahující n bodů $p_1, p_2, p_3, ..., p_n$. Značící mírů nějaké veličiny v průběhu času. Průměr posledních k hodnot definujeme pak jako SMA_k vypočtený vzorcem

$$SMA_k = \frac{p_{n-k+1} + p_{n-k+2} + ... + p_n}{k} = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{2.9}$$

Charakteristickou vlastností SMA je, že pokud data mají periodické výkyvy a aplikujeme na ně SMA o stejné periodě, pak dojde k eliminaci této datové variace.

Exponenciální klouzavý průměr (Exponentila moving average - EMA) je filtrem prvního řádu nekonečné impulsivní odezvy aplikujícím váhový faktor, který klesá exponenciálně, to jest váha s každým starším datem exponenciálně klesá do nekonečna k nule. V technické analýze pro danou sérii dat se využívá následující formulace pro výpočet aktuální hodnoty EMA_{now}, přičemž lze pozorovat, jak EMA postupuje směrem k nejnovější hodnotě jenom pomocí vážené části rozdílu při každém kroku.

$$EMA_{now} = EMA_{prev} + \alpha(p_{now} - EMA_{prev}) \tag{2.10}$$

Kde p_{now} značí aktuální hodnotu data v bodě p a α hodnotu váhové funkce.

V případě použití posuvného okna velikost k, aby se omezil vliv starších hodnot vyskytujících se v řadě, pak váha α nabírá formy

$$\alpha = \frac{2}{N+1} \tag{2.11}$$

kde N je velikost posuvného okna k a hodnota 2 je vyhlazovací faktor.

Ze vzorce 2.10 je patrné, že pro výpočet nové EMA hodnoty je třeba znát její předchozí hodnotu, co představuje komplikaci výpočtu prvního kroku. Z toho důvodu ve většině případů je tato hodnota nahrazena klasickou SMA pro stejné časové okno, čím se zaručí start výpočtu EMA.

Charakteristickou vlastností EMA je silnější reakce na změny novějších datových hodnot nežli klasická SMA, co znamená rychlejší adaptaci směrem k aktuální hodnotě. Co může být, jak výhodou, tak nevýhodou této funkce, jelikož při krátké anomálii s extrémní hodnotou dojde k dočasné nepřesnosti, na straně druhé je rychlejší adaptace při změně trendu v časové sérii.

Mean absolute percentage error - MAPE)

Hodnota MAPE je mírou podle niž lze určit přesnost předpovědi vzniklé jako výsledek nějakého modelu v rámci statistiky. Z pravidla představující přesnost, jako poměr formulován jako

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \mid \frac{A_t - F_t}{A_t} \mid .100 \tag{2.12}$$

kde A_t je pravdivá hodnota a F_t je hodnota předpovědi. Absolutní hodnota ve výpočtu je sečtena pro každý predikovaný bod v čase a vydělena počtem testovaných bodů n. Celek vynásoben 100 pro převedení na procenta.

Kapitola 3

Možnosti existujících řešení

Tato kapitola se zaměřuje na existující řešení chytrého ohřevu vody a alternativy k hlavnímu tématu práce. Shrnuje jejich možnosti, účinnost během provozu a finanční náročnost. Taktéž se věnuje jejich nedostatkům a omezením plynoucích z provozu v chytré domácnosti. Dnešní možnosti chytrého ohřevu vody můžeme rozdělit na tři hlavní odvětvi: chytré bojlery, moduly k bojlerům a hobby řešení.

3.1 Chytré bojlery

Chytrý bojler je zařízení využívané pro ohřev užitkové vody v domácnosti. Od normálního bojleru se odlišuje softwarem umožňujícím širší spektrum kontroly ohřevu vody a určitým stupněm autonomie spočívající v možnosti učit se vzorcům spotřeby teplé vody domácnosti v závislosti na denní době. Cenově se takové zařízení pohybuje o pár tisíc výš než klasický bojler vizte tabulka 3.1. Data o efektivitě takového řešení demonstruji na konkrétních produktech.

Značka	Typ	Smart funkce	Cena
Ariston	VELIS WI-FI 80	Ano	8000,-
DZ Dražice	OKHE SMART 80	Ano	6799,-
Mora	EOM 80 PK	Ne	3290,-
DZ Dráždice	OKCE 80	Ne	5299,-

Tabulka 3.1: Tabulka cen bojlerů

OKHE SMART

Pro příklad použijeme chytrý bojler firmy *DZ Drážice*. Podle dokumentace výrobku [9] bojler nabízí několik důležitých vlastností. Poskytuje SMART režim, který podle výše zmíněného dokumentu, poskytne minimálně 10% úsporu energie oproti standardnímu režimu po dokončení učící fáze trvající 7 dní. Dále je zde možnost provozu chytrého režimu v kombinaci s *HDO*[1]. Taktéž je důležitou vlastností ovládání pomocí *bluetooth* prostřednictvím mobilní aplikace, která krom nastavování skýtá možnost plánování pomocí kalendáře a času např. výjimky v provozu jakou je dovolená.

[1] **HDO** - Hromadné dálkové ovládání je impulz v sítí indikující nízký tarif

Z výše uvedených příkladů je patrná jedna značná nevýhoda a to, že takové řešení postrádá možnost centrálního řízení tudíž i efektivního začlenění do smart home kompozice s centrálním řízením. V druhé řadě potom vysoké pořizovací náklady.

3.2 Dodatečné chytré moduly

Tyto moduly v sobě mají podobnou funkcionalitu, jako tomu je u chytrých bojlerů. Opět je zde přítomen nějaký vnitřní mechanismus obstarávající spínání na základě předpokladu spotřeby horké vody. Avšak oproti bojleru mají jednu značnou výhodu a to v nenáročnosti instalace, při poskytnutí velmi podobných výsledků, ale s nepatrným rozdílem v efektivitě. Cena modulů se vejde v rozmezí do pár tisíců.

Aquanta

Podle dokumentace [16] se jedná o přídavný modul pro bojlery fungující na základě 2 teplotních čidel umístěných na trubce s přívodem studené vody a TP ventilu (pojistný ventil). Bojler je pak připojený do sítě přes modul, čím je umožněno snímat také spotřebu. Na základě teplotních dat a dat o spotřebě se systém učí a odhaduje spotřebu teplé vody. Komunikace je prostřednictvím Wi-Fi sítě a grafová reprezentace dat, jak i možná nastavení jsou dostupná pomocí webového rozhraní nebo mobilní aplikace. Opět je umožněno plánování pomocí kalendáře. Systém nabízí i API pro možnou integraci do cloudu.

U této alternativy je patrná značná multifunkčnost systémů, a také lepší základ pro integraci do smart home kompozice. Avšak problémem je stále vysoká cena pořízení činící 149$, co je v přepočtu zhruba 3150 kč.

3.3 Hobby řešení

Hobby nebo také DIY (udělej si sám) řešení zahrnuje velice rozmanité spektrum řešení v rámci tématu chytrého ohřevu vody a vytápění. Ve většině případů se ale nejedná o zcela samostatný systém, který by byl schopen rozhodovat se na základě nějakých nabytých znalostí podložených daty o využití, nýbrž o regulační termostaty s možností plánování nebo samostatného přepínání vytápěcích okruhů. Ve valné většině jsou založeny na jedné z variací jednodeskového počítače *Arduino* nebo *RaspberryPi*, jako řídící jednotky a doplněny o přídavné komponenty jako třeba spínací relé, jednoduchý displej a bezdrátové moduly kombinované s teplotními čidly umístěnými na rozvodech tepla a nebo v místnostech. Pomocí čeho vytvoří autonomní řídící systém, ne však chytrý pracující na základě nasbíraných dat.

Příkladový systém

Pro příklad si rozebereme systém s použitím *RaspberryPi*, jako domácího kontroleru pro vytápění představeného na [31]. Přičemž cena takového vlastnoručně vyrobeného systému, v odvolání na popis, by neměla překročit 1500 korun.

Zmíněný systém je založen na dvou hlavních částech, kontrolní jednotce umístěné při centrálním vytápění a bojleru(technická místnost) a jednoho nebo více teplotních senzorů rozmístěných v domácnosti. Kontrolní jednotka sestává z 2 součástek RPi[2] a relé pro spínání

[2] **RPi** - RaspberryPi

obvodu centrálního vytápění. Podobně je tomu u teplotních senzoru, kde je relé zastoupeno jedním nebo více teplotními snímači. Zařízení komunikují mezi sebou pomocí Wi-Fi sítě.

Na kontrolní jednotku RPi je nasazených několik softwarových technologii pro zajištění funkcionality. Z ohledem na webové rozhraní pro interakci s uživatelem se využívá *apache2* web server, na který je nasazeny jednoduchý webový systém založený na kombinaci *php* a *MYSQL* serveru. Pro manipulaci s databází je zde přítomen rovněž *Python*.

Systém poskytuje celkem komplexní možnosti řízení ohřevu domácnosti založené na mnoha faktorech. Pomocí tlačítka na webovém rozhraní umožňuje automaticky najít a přidat teplotní senzory rozmístěné v domácnosti, připojené k lokální síti. Aktualizace senzorových dat probíhá každou minutu stejně tak sepnutí relé pokud jsou splněné podmínky. Plánování je umožněno pomocí rozvrhu editovaného na webu, kde aktivátorem k události může být jedna z možností nebo jejich kombinace a to specifický čas, den, mód, časovač nebo i zařízení které se právě připojilo k lokální síti. Zde příkladově může být rozpoznána *MAC adresa* Vašeho mobilního telefonu, co indikuje přítomnost v domácnosti a znamená potřebu zvýšení vnitřní teploty. Je také možné vytvořit konkrétní nastavení spouštěné manuálně prostřednictvím webového systému např. volný víkend, příchod hostů atd. kde každý z těchto modelů se jinak odrazí na vytápění. Je dostupné i rozšíření pro vzdálený přístup.

Ve shrnutí, tento DIY systém je velmi bohatý ohledně možností plánování a spouštění daných postupů s ohledem na vzniklé události, což je výhodné a žádané v chytré domácnosti. Avšak nezpochybnitelný mínus je absence učícího se algoritmu, který by reguloval domácnost sám od sebe na základě senzorů se spouštěči a plánovač události by sloužil pouze pro doplnění (pokrytí výjimek).

Kapitola 4

Návrh řešení

V této části je shrnut návrh řešení, který by měl najít efektivní východisko dané problematiky, kdy chceme při pomocí chytré zásuvky konvertovat klasický boler na chytrý. Jak bylo zmíněno v úvodu hlavním úskalím je absence univerzálního systému, který by se nasadil k stávajícímu zařízení v domácnosti. V kapitole nalezneme také vlastní řešení problémů a cestu k dosažení stanovených cílů.

4.1 Obecný návrh řešení

Návrh vlastního řešení systému pro chytrý bojler, je poměrně komplexní záležitost z ohledu na rozmanitost. Proto je třeba mít rozhled v různých odvětvích vývoje. Systém pro konverzi běžného bojleru na chytrý, by měl být hlavně univerzální. Pro návrh je porto potřeba nejprve vytvořit analýzu požadavků. Hlavní požadavky na sytém:

- **Pořizovací cena** - Cena kompletního systému i jednotlivých komponent.
- **Univerzálnost** - Možnost aplikovat systém na většině bojlerů ve většině domácností.
- **Efektivita** - Efektivita řídicího algoritmu a viditelné snížení energie spotřebované na ohřev vody.
- **Nízká výpočetní náročnost** - Využití co nejmenšího množství zdrojů a možnost aplikovat algoritmus např. na mikropočítači Arduino.

V části kapitoly 4.2 bude detailně popsán návrh **HW zařízení** pro snímání teploty a jeho řídicího systému. Dále v části 4.3 je možno nalézt popis, jak je snímána spotřeba energie. Část 4.4 pak popisuje celkovou architekturu systému, a jak jsou propojeny jeho prvky. Poslední a zároveň nejdůležitější část se zabývá samotným algoritmem řízení, zpracování dat a predikce spotřeby teplé vody s ohledem na úsporu energie.

4.2 Snímač teplot

Jedná se o zařízení jež je schopno reagovat na vstupy i akce a umožňuje generovat adekvátní výstup. Konceptuálně je snímač v tomto řešení souborem součástek spojených v rámci elektrického obvodu umístěného na prototypovací desce neboli *DPS*[1]. Hlavní komponentou je mikrokontroler, který obsluhuje všechny akce v rámci daného zařízení.

[1] **DPS** - Deska plošných spojů

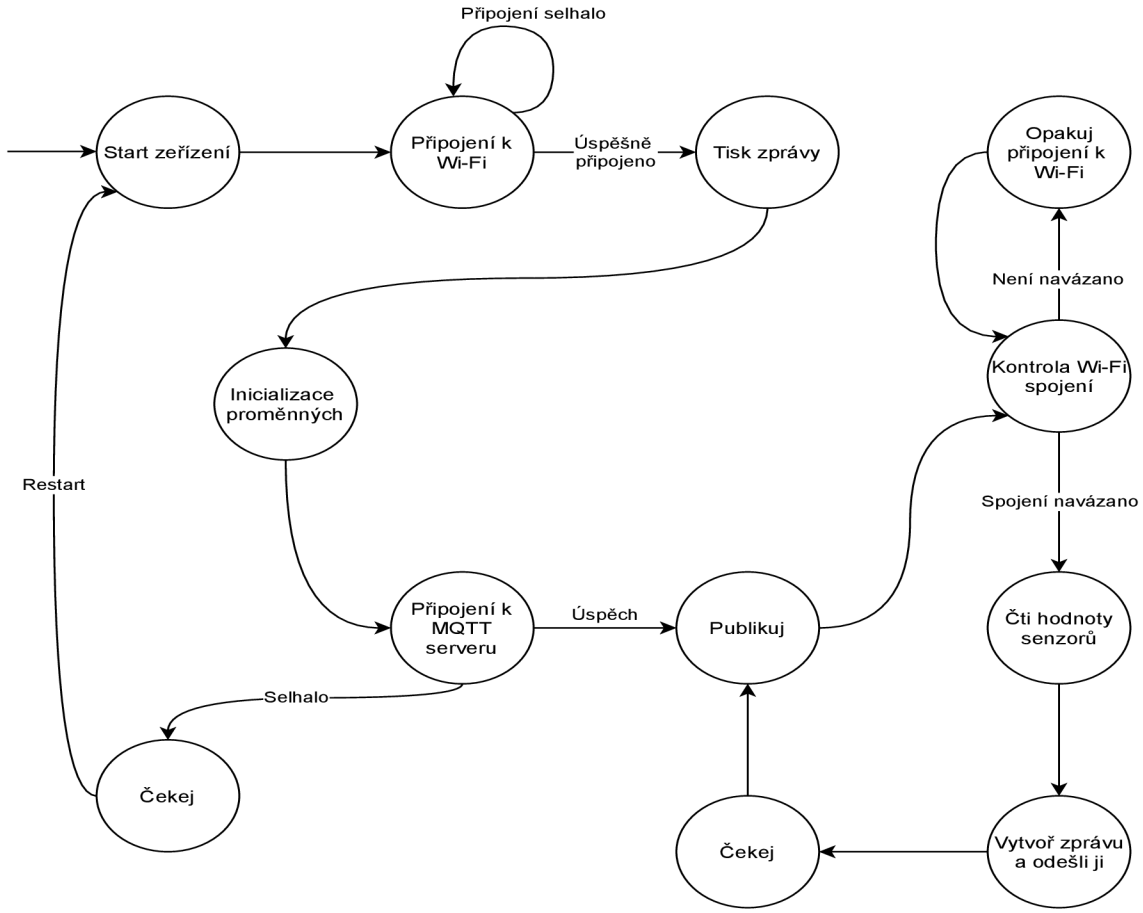
Mikrokontroler snímacího zařízení se stará o připojení I/O periferii (v rámci této práce se jedná o teplotní snímače a bezdrátový modul pro komunikaci s databází a uschování dat).

V této práci by se mělo jednat o energeticky nenáročné zařízení, napájené ze sítě pomocí široce dostupného 5V zdroje pro mobilní telefony, připojeného pomocí *microUSB* rozhraní. Nabízela by se zde i možnost použití akumulátoru, avšak v tomto konceptu, kde se předpokládá konstantní provoz, by akumulátor představoval riziko možného výpadku, při zanedbání jeho nabití. Mělo by rovněž skýtat možnost připojení dvou teplotních senzorů (včetně zajištění napájení). Z toho důvodů je vhodná i přítomnost bezdrátového *Wi-fi* modulu aby byla zajištěna komunikace s centrální databází. Pro komunikaci je vhodné využít protokolu *MQTT*(viz. kapitola 2.2).

Snímání teploty (nádrž a přívod studené vody) a její zápis do databáze je třeba provádět s dostatečnou frekvencí, aby byla zaznamenaná spotřeba i u krátkých činností, jako je mytí rukou. Proto jsem stanovil snímací interval na *5 sekund*.

Softwarová funkcionalita

Z pohledu softwarového by měl mikrokontroler umožňovat nahrání a spuštění vlastního kódu. Optimálně rozděleného na 2 části, kde první, hned po spuštění zajistí připojení do bezdrátové sítě a druhá, hlavní, obstará inicializaci potřebných *pinů* pro snímání teploty, připojení k serveru a zasílání teplotních dat, implementovaná pomocí automatu s několika stavy(viz. Obrázek 4.1). Všechny tyto informace o teplotě musí pak být cyklicky, co několik sekund, čteny a publikovány do databáze.



Obrázek 4.1: **Stavový automat.** Automat znázorňuje chování MCU pomocí přechodu mezi jednotlivými stavy.

4.3 Snímaní spotřeby a spínání bojleru

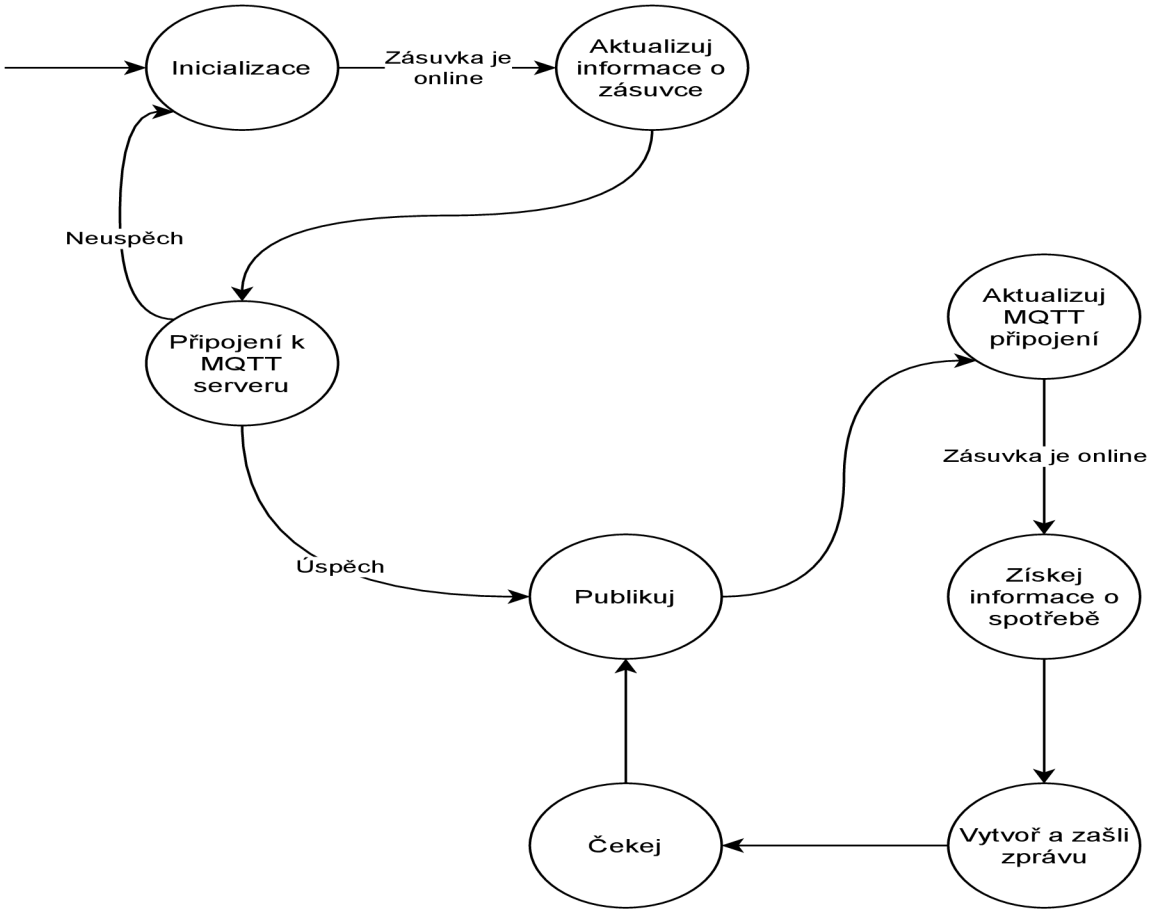
Elektrický okruh bojleru je potřeba spínat na základě odhadovaných metrik spotřeby teplé vody, aby ji byl připraven dostatek s potřebným předstihem. Proto připadá v úvahu chytrá zásuvka s možností dálkové kontroly alespoň v rámci lokální sítě. Což je potřeba pro efektivní řízení pomocí algoritmu. Druhým požadavkem na toto zařízení je monitorování aktuální spotřeby, přičemž výhodou by rovněž bylo monitorování celkového součtu spotřebované energie v Wh s možnosti resetování tohoto čítače.

Jelikož toto zařízení bude spínat ohřívač vody s vysokým odběrem energie, musí splňovat normu pro použití v rámci 16A elektrického okruhu. Taktéž veškerá omezení v rámci výkonu i s patřičnou rezervou plynoucí z výkonové špičky, která se muže objevit při spouštění ohřívače (pravděpodobně velmi malá, jelikož se jedná o odporový spotřebič).

Z ohledu na jednoduchost, by byla ideální určitá forma vložky do klasické zásuvky a volně dostupná aplikace nebo skript pro asistenci při jejím nastavení (přesně připojení k síti a případné přejmenování zařízení).

Softwarová funkcionalita

Sběr dat o spotřebě a jejich uložení v centrální databází, obstarává jednoduchý několika stavový skript viz. 4.2 spuštěný, jako služba na centrálním domácím serveru. Tento skript každých 5 sekund zažádá o energetická data a uloží je do databáze, při použití protokolu *MQTT*(viz. kapitola 2.2),za předpokladu že je zařízení aktivní.



Obrázek 4.2: **Stavový automat.** Automat znázorňuje chování skriptu pro odečet spotřeby ze zásuvky pomocí přechodu mezi jednotlivými stavy.

4.4 Architektura systému

Z pohledu vývoje systému, je tato část práce jedna z nejzásadnějších a podstatnějších. Při vývoji je třeba mít dobrý přehled o technologiích, návrhových vzorech a mnoho jiných věcech, aby bylo možné vhodně systém postavit. Architektura by měla být sestavena přehledně a takovým stylem, aby se dala jednoduše modifikovat nebo v případě nutnosti nahradit její část komplementární možností.

Architektura udává, jak jsou jednotlivé komponenty požadovaného sytému se sebou spjaty a propojeny. Stejně tak rozdělují sytém na jeho jednotlivé dílčí části, kde se každá z nich zaměřuje na specifickou úlohu a tím klade nároky na to, jakým způsobem je nutné ji realizovat.

Architekturu v rámci zadání(viz. Obrázek 4.3) lze z hlediska hardwaru rozdělit na 2 hlavní části, server a periferní zařízení (snímač, zásuvka), kde každá z nich má své specifické potřeby popsané v této kapitole, zásuvka vizte. část 4.3 a snímač vizte. 4.2.

Server tvoří platformu, na které běží všechny potřebné procesy pro zajištění požadavku zasílaných periferiemi. Dále se také zabývá samotnou logikou, komunikace mezi všemi částmi, vykonáváním požadavku na databázi, poskytnutím výpočetní síly pro komplexnější operace a hlavně poskytuje zdroje pro běh řídícího algoritmu, podrobněji popsáného v části 4.5.

Pro zaručení chodu systému je proto potřeba, aby na serveru běžela databáze se zaměřením na práci s senzorovými hodnotami z vícero zdrojů a jednoduchou možností napojení aplikace pro reprezentaci těchto dat uživateli. Přičemž nebude vystavena komplikovaným databázovým operacím jen jednoduchým *SELECT* dotazům. Její hlavní roli je uschování historických dat pro další zpracováni (predikce spotřeby). Jako optimální se jeví použití *InfluxDB*.

Dále je nezbytné z hlediska na využití *MQTT* protokolu(více viz. 2.2) pro komunikaci s periferiemi, aby byl v rámci serveru trvale v chodu *MQTT broker*. Realizován formou serverové služby např. pomocí *Eclipse Mosquitto*.

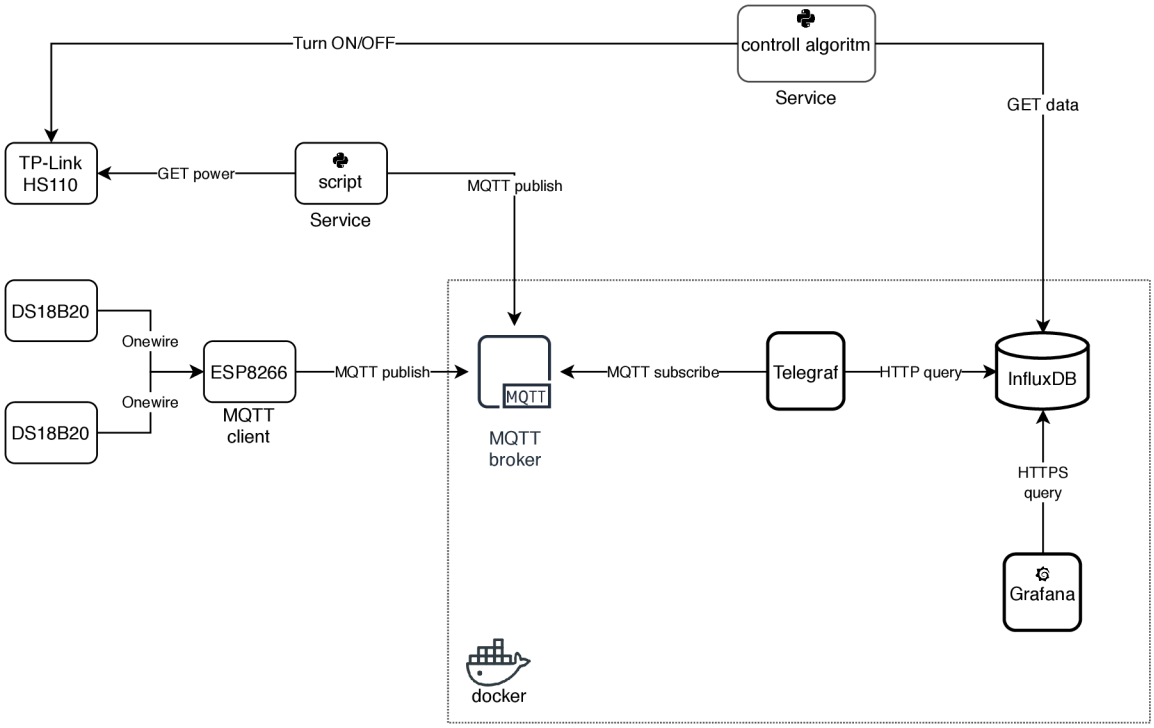
Samotná databáze a *MQTT broker* však netvoří nezávislý funkční celek, jelikož je potřeba ještě mezi nimi zajistit komunikaci a v tom i správné formátovaní sebraných dat, aby mohlo dojít k jejich uložení. Z toho důvodu je potřeba nasadit agenta pro provedení tohoto typu propojení sběru metrik a jejich předáni. Zde se nabízí třeba využití open source softwaru *Telegraf*, který po přihlášení k brokeru zasílá vybraná data formou *Queries* prostřednictvím *HTTP* protokolu do specifikované databáze.

Grafická prezentace dat s možností modifikace zobrazení a rozmanitou škálou agregačních funkci, jak též možnosti nastavení výstrah, je velmi dobře zpracovatelná pomocí open source webové aplikace *Grafana*. Tato aplikace vyhovuje rovněž proto, že je přístupná prostřednictvím jakéhokoliv aktuálního webového prohlížeče z libovolného zařízení ve stejné síti. To umožní uživateli jednoduchý náhled na aktuální stav bojleru (aktuální teplota, spotřebovaná energie, průběh vývoje teploty). Připojení k databázi je zajištěno pomocí *HTTPS* protokolu.

Veškeré skripty obstarávající požadavky periferií a výpočetní úlohy se zajistí formou serverových služeb, které budou spuštěny automaticky po naběhnutí systému a vyhovění jejich požadavkům na *dependencies* pro správnost běhu. Tím se míní skript pro snímání spotřeby viz. 4.3 a řídící algoritmus včetně všech jeho podpůrných částí.

Z ohledem na univerzálnost se jeví, jako správná volba využití *kontejnerizace*[2], jelikož představuje snadný způsob zajištění běhu všech potřebných části, stejně jako jejich případnou údržbu ve formě, kde by postačilo vyměnit pouze potřebný kontejner. Přičemž všechny potřebné kontejnery mohou byt jednoduše sestaveny a uvedeny do provozu pomocí *compose* souboru.

[2]**kontejnerizace** - Enkapsulace nebo také zabalení softwarového kódu a všech jeho závislostí, tak aby ten mohl být spuštěn jednotně a konzistentně v odlišných prostředích.



Obrázek 4.3: **Schéma systému.** Jedna se o schematické rozkreslení systému z hlediska na jeho části a jejich propojení.

Celý tento systém včetně všech jeho prvků, by měl být dostupný pouze v rámci lokální sítě, aby se tím předešlo nutností použití složitých autentizačních mechanik spjatých s interakcí a komunikace pomocí internetových služeb. Z důvodu zajištění bezpečné komunikace a tím zabránění kompromitace systému. Takové komunikace není zapotřebí uvážíme li stanovené cíle a oblast aplikace sytému.

Pro realizaci programového řešení je vybrán jazyk *Python* a jeho odvětví *MicroPython* optimalizováno pro provoz na mikrokontrolerech. Je velmi populární v sekci zpracování větších objemů dat a provádění výpočetních operací pro aproximaci budoucích hodnot. Taktéž je vhodný z hlediska mnoha dostupných knihoven, vytvořených právě pro tyto účely, co pro programátora představuje zjednodušení v rámci implementace dané problematiky.

4.5 Řídící algoritmus

Řídící algoritmus stanoví jádro celého systému, které je umístěno na serveru. Právě on obstarává veškeré výpočty nutné k jeho řádnému běhu. Vytváří dotazy na databázi a následně zpracovává data, reprezentující vývoj sledované veličiny v průběhu času. Zde konkrétně vývoj teplot v jednotlivých dnech. Vše aby se docílilo, co nejpřesnější předpovědi a umožnilo tak naplánovat další dění v systému. Jedná se proto o programově nejkomplexnější část systému, která musí být vytvořena s dostatečnou robustností aby mohla spolehlivě provádět komplikované výpočty a případně reagovat na náhlou změnu oproti predikovanému vývoji. Přičemž by neměla nijak zásadně ovlivnit náročnost na výpočetní zdroje.

Jelikož se jedná o složitější celek, je vhodné jej rozdělit na sérii podproblémů.

V první řadě získání vybraných hodnot z databáze.

V řadě druhé jejich následná diskretizace, to jest převedení série hodnot na interval reprezentován jednou konkrétní hodnotou.

Za třetí předání těchto diskretizovaných hodnot v patřičném formátu statistickému modelu, který na jejich základu predikuje budoucí vývoj a hodnoty. Dále je třeba tyto nové hodnoty opět převést (litry na stupně Celsia), aby mohlo dojít k výpočtu doby potřebné pro ohřev a naplánování sepnutí ohřevu. V poslední řadě je třeba udržovat aktuální přehled o teplotě, aby došlo k včasnému odhalení případné odchylky oproti předpovědi a tím zabránilo vzniku nedostatku teplé vody v zásobníku, předčasným spuštěním ohřevu. Tyto problémy jsou podrobně popsány v následujících částech.

Získání dat

Data o teplotě zachycené senzory jsou umístěny v databázi (InfluxDB). V první řadě proto skript musí vypracovat patřičný dotaz na databázi pro jejich získáni.

V tomto případě mají data o teplotě vody posloužit k vypracování teplotního průběhu pro následující den. Proto nemůže dojít k výběru všech dostupných dat, a potom na jejich základu k vypracování budoucího průběhu. Důvodem je vysoká variabilita mezi jednotlivými dny týdne, kde ku příkladu už jen víkend je z hlediska uspořádaní dne značně odlišný od pracovních dnů.

Proto je potřeba separovat jednotlivé dny a tím omezit výběr dat. Taktéž není nutné aby obsahoval zbytečně stará data, jelikož můžou mít v sobě otisknuté vzorce chování, které podlehlo změnám. Stejně tak se může měnit trend z hlediska na měsíc v roce a roční období.

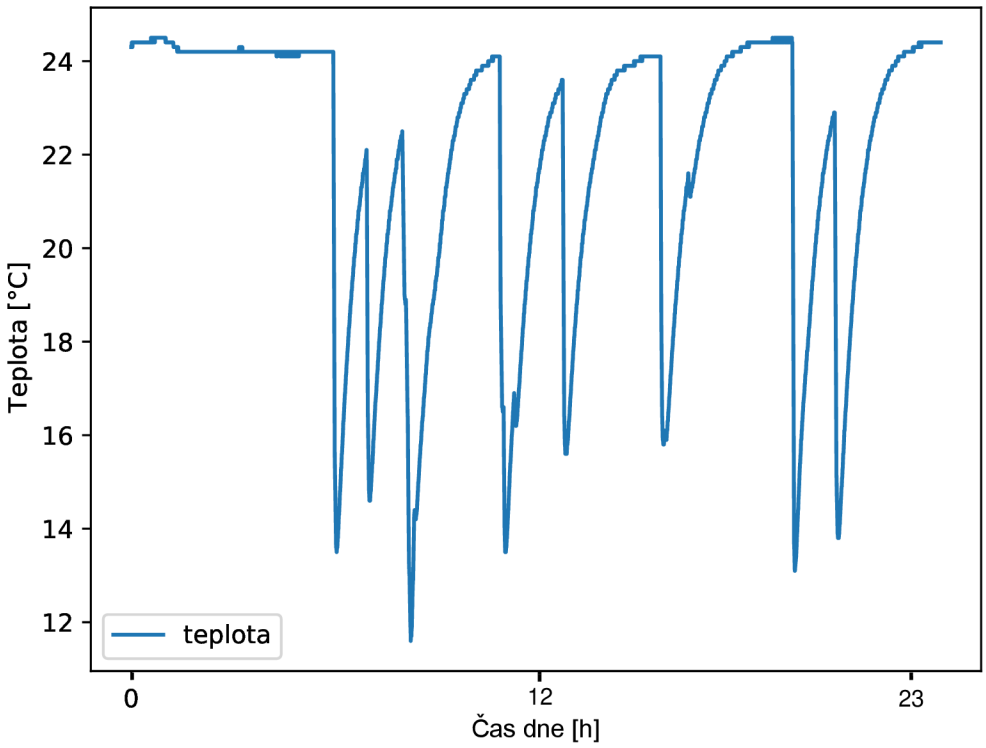
Z toho důvodu se vhodně jeví vybírat pouze data pro konkrétní den týdne s omezením na čtyři předchozí stejné dny (měsíční okno), aby byla zachována dynamičnost změn chování. Na to později, při predikování, navazuje aplikace exponenciálního klouzavého průměru, aby starší data měla menší váhu.

Dotazy na databází, tak musí být dynamicky měněny v závislosti na nadcházejícím dnu týdne a musí být dopočítáno rovněž datum pro čtveřici minulých stejných dnů. Teprve taková data je možné použít v dalších výpočtech.

Zpracování

Aby mohlo dojít k předání dat funkci predikující jejich průběh, je nejprve potřeba je před připravit, to jest náležitě přefiltrovat, transformovat z teploty na objem a následně diskretizovat do 15 minutových intervalů.

Filtrace teplotního průběhu je nutná, jelikož v zájmu je detekovat pouze klesající série teplotních hodnot, neboť právě ony indikují přítok čerstvé studené vody do zásobníku, co svědčí o vzniklé spotřebě. Z pohledu detekování zde ale existuje problém spjatý s samovolným pozvolným poklesem teploty v důsledku tepelných ztrát plynoucích z fyzikálních zákonů o sdílení tepla a základu termodynamiky (viz. Kapitola 2.4 a 2.3). Proto jsou použity 2 tepelné snímače, kde jeden snímá teplotu nádoby (umístěný v šachtě pro analogový teploměr) a druhý teplotu přívodové trubky se studenou vodou. Jelikož právě na ní jsou díky sdílení tepla prouděním a vedením velmi dobře viditelné teplotní výkyvy v momentě, kdy vznikne nebo zanikne vodní průtok viz. obrázek 4.4.



Obrázek 4.4: Ukázka teplotních výkyvů při vzniknutí průtoku na přívodu studené vody.

A právě korelace mezi oběma teplotními veličinami kombinovaná se stejným časovým razítkem, umožňuje odlišit, kdy došlo k poklesu teploty z důvodu odběru, a kdy se jedná o samovolný pokles způsobený rozdílem teplot vůči okolnímu prostředí. Proto pro určení rozdílu teploty nádrže bude použito časové okno definováno dobou začátku klesání teploty na přívodové trubce a ukončeno v momentě zahájení její růstu. Eventualitou je rovněž možnost, kde časové okno ukončí konec 15 minutového intervalu. V tomto případě bude začátek následujícího intervalu rovněž začátkem okna pro klesající teplotu.

$$i_t < i_{t-1} \wedge i_t < i_{t-2} \text{ pak je klesající}$$
$$i_t > i_{t-1} \wedge i_t > i_{t-2} \text{ pak je rostoucí}$$

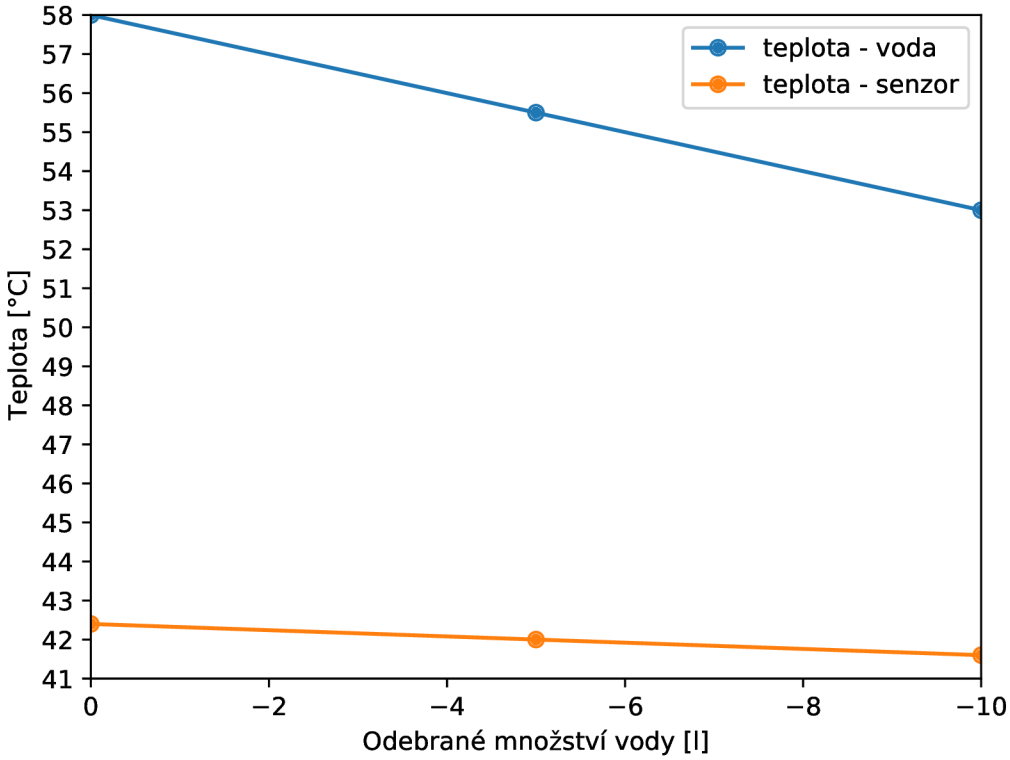
Kde i_t, je teplota v čase t a i_{t-n} je n-ta předchozí hodnota.

Výsledkem této operace tak může být série teplotních dvojic(`0 až N`), stanovujících rozdíl teploty nádrže před a po odčerpání vody.

U bojleru postrádajícího šachtu pro analogový teploměr, co platí i pro ten, na kterém je prováděn vývoj. Jsou tyto teploty nádrže nahrazeny teplotami pláště (ocelová nádoba pod izolací), ty jsou ale závislé na mnoha faktorech(viz. Kapitola 2.4). Z toho důvodu je třeba hodnoty dodatečně přepočítat na skutečnou teplotu vody.

Toho lze docílit dvěma různými způsoby přičemž první využívající znalosti a rovnice o sdílení tepla 2.4, co v tomto případě nelze aplikovat z hlediska na nemožnost řešit soustavu

jedné rovnice o vícero neznámých (viz. rovnice 2.8). Kde je neznámou jak teplota tekutiny uvnitř, tak hodnota α i q. Proto se využije druhé metody založené na experimentálním přístupu, která umožní určit vztah mezi teplotou na čidle a teplotou vody uvnitř. Toho se docílí jednoduchým experimentem. Zaznamená se teplotu na čidle a odebere testovací množství vody pro zjištění její teploty. Načež se odebere ze zásobníku 5 litrů vody a opět provede záznam a to opakovaně. Tímto způsobem se získají hodnoty pro obě veličiny a na jejich základu stanoví vztah pro přepočet teploty čidla na teplotu vody. Průběh viz graf 4.5. Z toho vyplývá, že pokles teploty na čidle o **0.1** stupně znamená pokles teploty vody o **0.625** stupně, co se použije jako konstanta pro přepočet.



Obrázek 4.5: **Průběhy teplot.** Závislost mezi teplotou na čidle a teplotou vody při odebírání vody po 5 litrech $y_{water} = [58, 55.5, 53]$ a $y_{sensor} = [42.4, 42, 41.6]$

S výslednou reálnou teplotou pak můžeme určit, při použití základních znalostí o termodynamice (viz. Kapitola 2.3), množství odčerpané horké vody pro dvojici. To provedeme použitím kalorimetrické rovnice 2.4 pro teplotu ustáleného systému. Ta v tomto konkrétním případě má tvar:

$$T = \frac{c_1 m_1 T_1 + c_2 m_2 T_2}{c_1 m_1 + m_2 c_2} \tag{4.1}$$

Kde c_1, m_1, T_1 jsou hodnoty reprezentující ohřátou vodu v zásobníku a c_2, m_2, T_2 jsou hodnoty vody studené přitékající do zásobníku po odběru (pro T_2 bude použita průměrná

teplota studené vody 8.7 °C [33]). Hodnota T pak reprezentuje teplotu vody v nádrži po odběru horké vody a promísení zbytku ohřáté vody s čerstvou studenou vodou.

Jelikož tento konkrétní systém nevyžaduje velmi vysokou přesnost odhadovaného objemu (setiny litru), respektive je nemožné ji dosáhnout už z hlediska přepočtu reálné teploty. Můžeme tedy předpokládat, že teplo je vyměňováno pouze v rámci dvou tekutin bez ohledu na ocelový plášť nádrže a izolaci za ním. Dále jak vyplývá z fyzikálních tabulek vizte Tabulka 4.1, je patrné že měřená tepelná kapacita má v rozmezí teplot, ve kterém se pohybujeme minimální variaci, proto můžeme hodnotu c_i považovat za konstantu.

Teplota T (°C)	Měřená tepelná kapacita c ($J.kg^{-1}.K^{-1}$)
T = 10	4,195
T = 30	4,176
T = 50	4,178
T = 70	4,187
T = 90	4,202

Tabulka 4.1: Měřená tepelná kapacita vody v závislosti na teplotě. Převzato z [22].

Z toho předpokladu plyne možnost zjednodušení rovnice 4.1 na tvar 4.2, který má po provedení úprav formu 4.3 reprezentující hmotnost přiteklé vody v ***kg***, což je komplementární s objemem v jednotkách ***l***. Avšak ten představuje rovnici o dvou neznámých, jelikož nevíme kolik vody zbylo v nádrži a kolik přiteklo. Jejich součet 4.4 můžeme ale nahradit známým objemem nádrže (dále m_3) a pak využít substituce, co vede na druhou rovnici 4.5 z níž je možné pak získat množství spotřebované vody m_2 jelikož platí $pritok = spotreba$

$$T = \frac{m_1 T_1 + m_2 T_2}{m_1 + m_2} \tag{4.2}$$

$$m_2 = \frac{T(m_1 + m_2) - T_1 m_1}{T_2} \tag{4.3}$$

$$m_3 = (m_1 + m_2) \tag{4.4}$$

$$m_2 = \frac{T m_3 - T m_1}{T_2} \tag{4.5}$$

Ve finále, tak díky vyjádření m_1 a upravě obdržíme rovnici 4.6 pro množství spotřebované vody, které se přičte k souhrnu daného 15 minutového intervalu. Souhrny v další části umožní výpočet predikce.

$$m_2 = \frac{m_3(T - T_1)}{T_2 - T_1} \tag{4.6}$$

Souhrn spotřeby je z podstaty věci závislý na teplotě vody vztahem, *čím teplejší voda v nádrži, tím menší spotřeba.* Proto je nutno jej normalizovat na spotřebu vody o konstantní

teplotě (směs studené vody z vodovodu a horké z bojleru). Za teplotní konstantu zvolíme teplotu těla, to jest $37°C$ a použitím transformované rovnice 4.2 v kombinaci s rovnicí 4.4 dostaneme normalizované množství použité vody ve tvaru 4.7 jelikož známe hodnoty pro T, T_1, T_2, m_1. Na základě této hodnoty leze již bezpečně vypočítat předpověď.

$$m_3 = m_1 + \frac{m_1(T_1 - T)}{T - T_2} \tag{4.7}$$

Predikce

Predikci se rozumí, co nepřesnější výpočet budoucích hodnot, na základu historických záznamů. V případě tohoto systému bude prováděna predikce spotřebovaného množství vody, v objemové jednotce litr, na následujících 24 hodin.

Pro výpočet jsou k dispozici data ve formátu, 96 patnácti minutových intervalů (dále jako i_1 *až* i_{96}) s hodnotou spotřeby s pro každý interval a to pro 4 předchozí stejné dny týdne (dále jako d_{t-1} *až* d_{t-4}). Dohromady činící časovou sérii 384 hodnot. Vezmeme li v potaz poměr výkonu ohřevu a objemu bojlerů (zde konkrétně 2400W k 80l co znamená cca 2:20 pro dosažení 65°C), pak můžeme oněch 384 hodnot redukovat na 96 sečtením čtyř intervalu do jednoho, protože to nijak zásadně neovlivní přesnost předpovědi ani řízení ohřevu a zároveň dojde k zjednodušení výpočtu z ohledu na počet hodnot.

Data stanoví jednorozměrnou časovou sérii, bod v čase a hodnota. Na jejich základu musí být vypracovaná krátkodobá předpověď (to jest z předchozích 48 až 96 hodin na následujících 24). Při pohledu na data přes toto krátké časové okno, můžeme uvažovat, že jsou stacionární tudíž jejich hodnota osciluje kolem určitého průměru (strop je objem nádrže a minimum 0) a zároveň v nich existuje jistá cykličnost odpovídající lidskému chování v dané denní době. Tyto informace lze následně aplikovat při výběru metody pro předpověď a omezit tak škálu možností.

Dobře se zde prezentuje metoda postavena na základu *klouzavého průměru* (dále jen *MA*[3]) jmenovitě metoda *exponenciálního klouzavého průměru* (dále jen *EMA*[4]), přičemž se aplikuje specifickým způsobem popsaným níže. Bližší vysvětlení funkcionality, výpočtu a specifikace těchto metod vizte kapitola 2.5.

Využitím klouzavého průměru *EMA*, který se sice rychleji přizpůsobuje změnám v porovnání s jednoduchým klouzavým průměrem, by v momentě jeho aplikace pro vytvoření denního průběhu na bázi předchozích hodnot sice vznikl nový denní průběh, ale byl by mírně opožděn oproti realitě. A tato prodleva by pak způsobovala chyby v předpovědi.

Proto místo celodenní kontinuální aplikace, by byl pomocí EMA předpovídán pouze každý z jednotlivých 24 intervalu i_n zvlášť. To znamená, že by nejdříve byl vypočten průměr ze dvou až čtyř intervalu i_1 pro následný start výpočtu *EMA* na intervalu i_1 v dni d_{t-4}, dále d_{t-3} až do d_{t-1}. Hodnota *EMA* pro i_1 v dni d_{t-1} by pak sloužila, jako předpověď spotřeby s v intervalu i_1 pro následující den. Obdobně by se vypočetlo aproximaci spotřeby s pro zbývající intervaly počínaje i_2 do i_24 a tím dosáhlo plné 24 hodinové předpovědi.

Takto vypočtené hodnoty následně slouží pro naplánování doby a času spuštění ohřevu.

Na historických hodnotách lze poté otestovat vývoj přesnosti předpovědi v závislosti na velikosti okna využitého pro výpočet *EMA* v různých fázích systému odvíjejících se od

[3] **MA** - z angličtiny Moving average
[4] **EMA** - z angličtiny Exponential moving average

doby, jaká uplynula od započetí provozu. Pro určení přesnost se použije funkce *MAPE*[5] pro jednotlivé velikosti okna (2, 3, 4).

Plánování ohřevu

Plánování ohřevu je poslední důležitou části řídicího algoritmu, zabývající se následujícími úkoly: monitorování aktuální teploty a srovnávání vůči předpovědi, kontrola minimální teploty nádrže a plánování času sepnutí ohřevu s dostatečným předstihem.

Minimální teplotu nádrže stanovíme na 40°C, co je o 3 stupně víc než předpokládána teplota horké vody během užívání. Čím se zamezí vzniku situace, kdy by byla k dispozici pouze voda vnímána jako studená. Proto musí docházet k cyklickému ověřování teploty vůči této hranici a bude-li překročena, pak k automatickému sepnutí ohřevu. Přičemž je dostačující, provádět tuto činnost jednou za minutu.

Cyklicky je rovněž vhodné konfrontovat momentální teplotu nádrže s předpovědí spotřeby, kde je ale už zapotřebí provést dodatečné výpočty opět na bázi *Termodynamiky* 2.3. To jest vyvodit z aktuální teploty nádrže maximální produkční kapacitu vody o normované teplotě 37°C a tu porovnat s sumou předpokládané spotřeby před dalším ohřevem. S použitím rovnice 2.3 a 4.4 dosáhneme vztahu,

$$m_3 = m_1 + \frac{m_1(T_1 - T_3)}{T_3 - T_2} \tag{4.8}$$

kde m_1 je objem bojleru, T_1 teplota vody v nádrži, T_2 teplota studené vody, T_3 teplota požadované vody a m_3 je její maximální množství, které je možné obdržet. Přičemž m_2 je vyjádřeno formou zlomku.

Hodnota m_3 je pak porovnána z sumou spotřeby před dalším ohřevem a pokud ta je menší, pak není nutné plánovat předčasný ohřev vody. Vyhodnocení tohoto typu, tak musí být provedeno na konci každého hodinového intervalu aby byla odchylka odhalena s dostatečným předstihem.

Hlavní část plánování se musí provádět krátce po dokončení předpovědi, což je těsně po začátku nového dne a to hlavně z ohledu na fakt, aby nebylo potřeba dopočítávat časový posun v rámci data a také jelikož nízký cenový tarif za elektřinu se z veliké části nachází právě v nočních hodinách. Rovněž z důvodu tepelných ztrát je třeba cílit na to, aby byl ohřev dokončen před první znatelnou denní spotřebou. Proto je nutno plánování provést následujícím způsobem.

V momentě dokončení předpovědi se vyhledá první interval i_n s nenulovou spotřebou s a od tohoto bodu se sečte spotřeba všech následujících intervalu po interval i_m nacházející se v rozmezí odpoledních hodin (12:00 až 16:00) a zároveň mající nejnižší spotřebu s právě v tomto rozmezí. Rozmezí je opět zvoleno z ohledem na výskyt nízkého tarifu v domácnosti, kde je systém nainstalován. Suma spotřeby $\sum s$ od intervalu i_n po interval i_m mimo něj, pak slouží pro výpočet potřebné teploty nádrže aby byla uspokojeny požadavky na spotřebu. Minimální teplotu nádrže pak určíme soustavou dvou rovnic 2.3 a 4.4 (měřená tepelná kapacita c je vynechána ze stejného důvodu jaký byl uveden výše viz tabulka 4.1)

$$m_3 = m_1 + m_2 \tag{4.9}$$

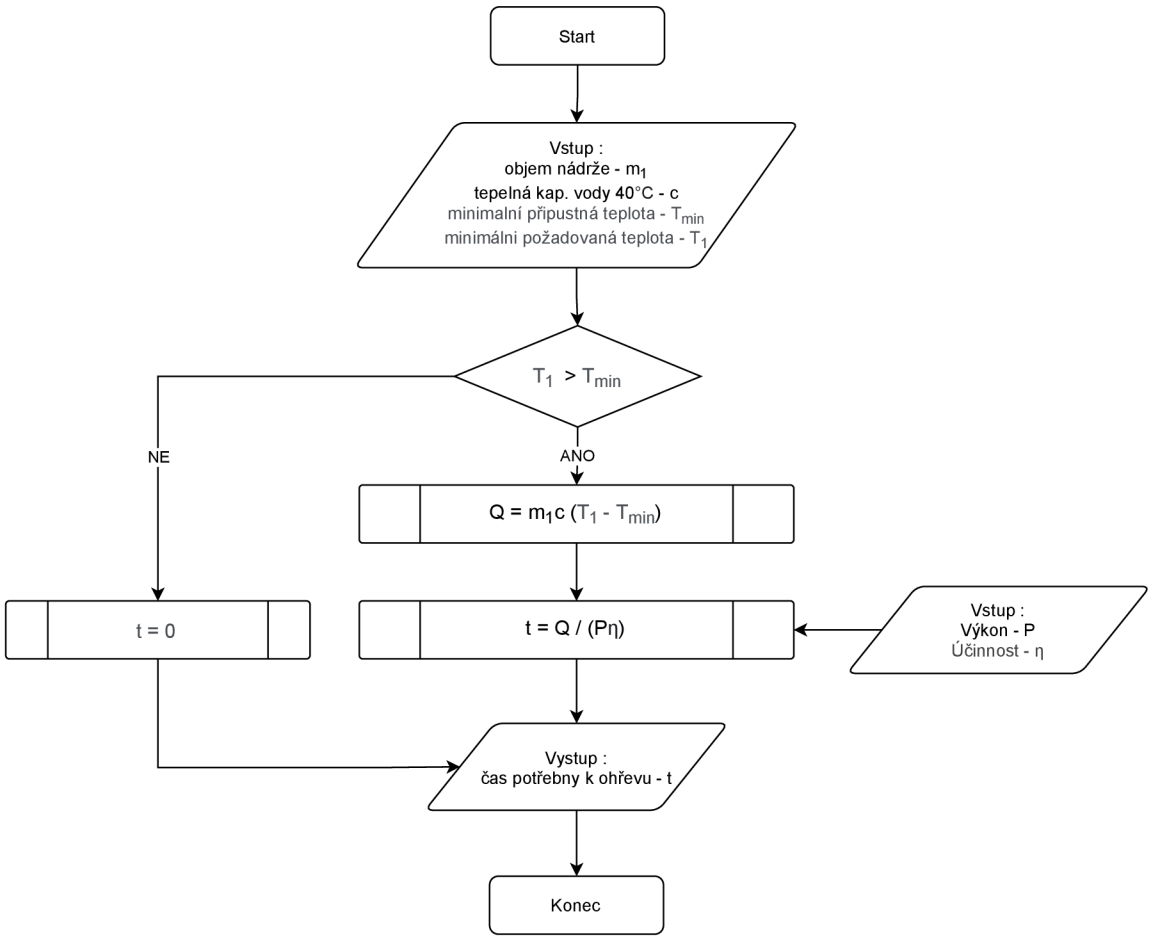
$$T_3 m_3 = T_1 m_1 + T_2 m_2 \tag{4.10}$$

[5] **MAPE** - z angličtiny Mean absolute percentage error

kde z první získáme m_1 po dosazení spotřeby za m_2 a objemu nádrže za m_3 co se následně využije ve druhé, kterou pak převedeme na tvar 4.11

$$T_1 = \frac{T_3 m_3 - T_2 m_2}{m_1} \tag{4.11}$$

kde T_1 značí minimální teplotu nádrže, m_1 objem nádrže po odečtení sumy spotřeby $\sum s$, T_2 teplotu studené vody, m_2 objem čerstvé studené vody (rovný objemu spotřeby), T_3 minimální přípustnou teplotu vody v nádrži (40°C) a m_3 objem nádrže.



Obrázek 4.6: Vývojový diagram postupu pro určení času t potřebného pro ohřáti vody na požadovanou teplotu T_1.

Vypočtená minimální teplota vody v nádrži se poté využije k určeni množství tepla Q potřebného pro dosažení teploty T_1 a to pomocí rovnice 2.5. Při tomto výpočtu se vychází z rozdílu nejnižší možné přípustné teploty (40°C) a teploty T_1, jenž je reprezentován jako ΔT, dále měřené tepelné kapacity c pro vodu o teplotě 40°C a objemu nádrže m_1.

Tak získané teplo Q je nezbytné pro stanovení doby potřebné pro ohřev vody za využití rovnice 2.6, kde za P je dosazen výkon konkrétního zařízení (zde 2,4kW) a za η účinnost elektrických ohřívačů vody činící 98% podle článku [21].

Tím se získá čas potřebný pro ohřev (výše popsaný postup získání ilustrován obrázkem 4.6), který je následně odečten od intervalu i_n, co stanoví přesnou hodinu, kdy je nutné začít ohřívat.

Obdobně se postupuje i v případě přípravy vody od konce intervalu i_m až do konce dne. Z rozdílem, že se vynechá část s výpočtem počátku ohřevu a přejde se neprodleně k sepnutí ohřívání, které je vypnuto po dosažení minimální teploty vody v nádrži T_1 vypočtené opět na základu rovnice 4.11 s jediným rozdílem a to, že za sumu spotřeby $\sum s$ je dosazena nová hodnota odpovídající sumě spotřeby po intervalu i_m až do konce dne.

Takovýmto postupem se řídí část algoritmu zajišťující ohřev na určitou teplotu a spínání chytré zásuvky.

Kapitola 5

Použité technologie

V této kapitole jsou popsány technologie, komponenty a služby využívané pro implementaci návrhu řešení sytému pro chytré řízení ohřevu bojleru. Využití těchto prvku bylo pečlivě uváženo se snahou využít nejlepší možné dobře dostupné technologie a techniky, co bych svým pohledem označil za úspěšné.

V podkapitole 5.1 jsou popsány technologie, které byly použity na implementaci serverové části, podkapitola 5.2 pak obsahuje technologie a komponenty použité pro implementaci hardwarové části tj. periferních zařízení projektu.

5.1 Serverová část

Jak už bylo řečeno části věnující se návrhu architektury systému řízení bojleru (více informací viz. 4.4), serverová část v sobě skýtá nejkomplikovanější implementační partie z celého vývoje řešení. Pro potřeby vývoje serverové části (spotřeba energie, řídíci algoritmus) byl využit jazyk *Python* a doplňující open source knihovny (detailněji popsané níže v této části). Pro vystavění prostředí, nutného pro běh systému byl využit nástroj *Docker*, pod kterým běží databáze *Influxdb*, mqtt server *mosquitto*, agent pro sběr metrik senzorů a jejich zápis *Telegraf*, jak rovněž platforma vizualizující shromážděné metriky *Grafana*.

Influxdb

Obrázek 5.1: Převzato z [17].

Influxdb je open source databází pro časové série dat (TSDB[1]) vyvíjenou společností InfluxData. Napsána je v jazyce *Go* a optimalizovaná pro rychlé, široce přístupné uschování a vyhledávání v rámci dat o charakteru časové řady. To pro použití v odvětvích, jako monitorování procesů, pozorování metrik aplikací, analytické operace v reálném čase nebo monitorování *IoT* senzorů. Nabízí rovněž rozsáhlou podporu co se týče jiných frameworků a volně dostupné klientské knihovny k většině programovacích jazyků pro umožnění jednoduchého vývoje aplikací.

Databáze je rovněž multiplatformní aby bylo možné její využití nezávisle na operačním systému. Vydaná je pod licencí *MIT*, tudíž ji lze svobodně distribuovat a měnit.[19]

Má za sebou přes 8 let vývoje a výtečnou reputaci v oblasti záznamu senzorových dat, rovněž z hlediska vývojáře představuje jednoduchou aplikaci a vysokou spolehlivost. Proto byla databáze *Influxdb* vybrána i pro použití v implementaci systému řízení ohřevu bojleru.

Telegraf

Obrázek 5.2: Převzato z [18].

Telegraf je serverový agent řízený zásuvnými moduly pro shromažďování a odesílání metrik a událostí z databází, systémů a senzorů *IoT* zařízení. Má rovněž mnoho pluginu ke zdrojům různých metrik přímo ze systému, na kterém běží nebo může získávat metriky z API třetích stran. Případně i naslouchat metrikám prostřednictvím statistik nebo jiných služeb. Je napsán v jazyce *Go* a kompiluje se do jediného binárního souboru bez externích závislostí a také vyžaduje velmi malou paměťovou stopu.

[1] **TSDB** - Time series database

Stejně jako je tomu u zdrojů metrika, co se tyče množství pluginu, tak platí i pro výstup, to jest posílání metrik. Tím pádem je schopen zapisovat do široké škály dalších databázi a služeb např. Graphite, Librato, MQTT, *Influxdb* aj. [20]

Pro tyto vlastnosti je zvolen, jako prostředník i v implementaci systému řízení ohřevu bojleru, kde je nakonfigurován pro sběr metrik ze senzoru a následně jejich odeslání do *Influxdb* v patřičném formátu.

Mosquitto

Obrázek 5.3: Převzato z [28].

Eclipse Mosquitto je open source zprostředkovatelem zpráv, který implementuje protokol MQTT verze 5.0, 3.1.1 a 3.1, nacházející se pod licencí EPL/EDL.

Přičemž jeho implementace serveru protokolu MQTT, napsáná v jazyce *C* je velmi nenáročná na zdroje a proto je vhodná pro všechny situace od strojů s vysokým výkonem až po vestavěné systémy, a stroje s nízkou energetickou náročností. Senzory, které jsou často zdrojem jak i cílem zpráv MQTT, mohou být malé a postrádat vysoký výkon. To platí také pro vestavěné systémy, ke kterým jsou připojeny, což může představovat místo běhu služby Mosquitto. Současná implementace Mosquitto využívá spustitelný soubor pohybující se řádově kolem 120 kB, který spotřebovává přibližně 3 MB RAM při připojení 1000 klientů.[11]

Právě jelikož se i jeden z požadavků na systém řízení ohřevu bojleru týká nízké náročnosti na zdroje, bylo využito této serverové implementace MQTT protokolu.

Grafana

Obrázek 5.4: Převzato z [23].

Grafana je volně šiřitelný multiplatformní webová aplikace, používána k analytice a interaktivní vizualizaci dat. Uživateli poskytuje množství grafů, schémat a také upozornění ohledně připojených datových zdrojů v případě překročení nastavené podmínky. Je dostupná kromě enterprise verze formou samoobslužné instalace a je rozšiřitelná pomocí plugin sytému. Nasazená lokálně, pomocí zmíněné samoobslužně instalace je pak dostupná prostřednictvím libovolného internetového prohlížeče na adrese *http://localhost:3000* na portu číslo 3000.

Koncovému uživateli umožňuje komplexní panely pro monitorování různých veličin a dějů při využití interaktivních nástrojů pro sestavování dotazů. Jako vizualizační nástroj

je *Grafana* populárni komponentou využívanou při monitorování real-time kolekcí dat v kombinací s *TSDB* takovými jak *Influxdb*, Graphite aj. [24]

Vydaná je pod licencí *Apache 2.0*, čím umožňuje uživateli volné používání, šíření, modifikaci a šíření modifikované verze při zachování stejné licence.

O použití této aplikace bylo rozhodnuto pro její jednoduchost, s vysokým stupněm možnosti modifikace zobrazení sledovaných hodnot a uživatelskou přívětivost. Stejně tak pro osvědčené použití ve vizualizací časových sérii dat a vysokou kompatibilitu s *Influxdb*.

Docker

Obrázek 5.5: Převzato z [8].

Docker je sada PaaS[2] produktů, které používají virtualizaci na úrovni OS k doručení softwaru ve formě balíčků nazývaných kontejnery, které jsou navzájem volně izolované. Izolace a zabezpečení umožňují provozovat na daném hostitelském systému více kontejnerů současně. Kontejnery samy o sobě jsou ne až tak náročné na výkon, jelikož všechny sdílejí služby jednoho jádra operačního systému, používají tak méně prostředků než virtuální stroje.

Obsahují rovněž vše potřebné ke spuštění dané aplikace, takže není třeba se spoléhat na to, co je aktuálně k dispozici na hostitelském systému. Přitom je volně k dispozici široká škála už hotových obrazů kontjnerizovaných aplikací, co ještě víc minimalizuje náročnost přípravy prostředí vyžadovaného pro chod konkrétní aplikace. [7]

Tyto výhody jsou hlavním důvodem využití produktu *Docker* i v implementaci systému popisovaného v této práci, kde se ho využívá právě pro zajištění prostředí nutného pro správný chod celého systému. Zastřešuje služby *Eclipse Mosquitt*, *Telegraf*, *Influxdb* a *Grafana*, čím se minimalizují komplikace při jejich instalaci.

Python-kasa

Python-kasa je knihovna určená jazyku Python pro bezdrátovou kontrolu TP-Link smart home přístrojů takových jak zásuvky, přepínače, svítící pásky nebo žárovky. Pro komunikaci se zařízeními se využívá asynchronních dotazů na jejich API.[6]

Knihovna je využívaná z důvodu komunikace s chytrou zásuvkou TP-Link HS110.

APScheduler

APScheduler (Advanced Python Scheduler) je Python knihovna umožňující plánování času provedení dávky kódu, a to pouze jednou nebo periodicky s nějakým intervalem. Nabízí možnost přidávání nových nebo odebírání starých úloh za běhu programu v libovolném momentě. V případě uložení úloh do databáze zůstanou zachovány i pokud dojde k restartu plánovače, včetně jejich aktuálního stavu.

[2] **PaaS** - Platform as a Service

Mimo jiné rovněž poskytuje bohatou sadu spouštěčů aby bylo možno definovat specifickou událost, kdy má dojít k provedení nějaké speciální činnosti. Mimo prováděné činnost nenarušuje nijak běh aplikace.[13]

Tato knihovna byla zvolena na základě požadavků, některých synchronizačních úloh jež je nutno provádět s předem neznámými dynamicky se měnícími intervaly, spjatými s dynamickým charakterem systému řízení ohřevu bojleru. Rovněž z ohledem na kladnou uživatelskou zpětnou vazbu.

Paho

Obrázek 5.6: Převzato z [12].

Paho projekt byl vytvořen pro poskytnutí škálovatelných open-source implementací otevřených a standardních komunikačních protokolů zaměřených na nové, existující a nově se objevující aplikace pro M2M[3] a IoT prostředí. Reflektuje inherentní fyzická a nákladová omezení připojení zařízení. Cíli na efektivní úroveň oddělení zařízení a aplikací, navrženou pro podporu rychlého růstu škálovatelného webového a podnikového middlewaru a aplikací. Primární zaměření tohoto projektu je implementace publish/subscribe klienta MQTT protokolu určena pro vestavěné systémy.[10]

Zde je Paho knihovna pro jazyk Python používána v rámci skriptu obsluhujícího chytrou zásuvku k zasílání dat o spotřebě energie na MQTT server.

5.2 Hardwarová část

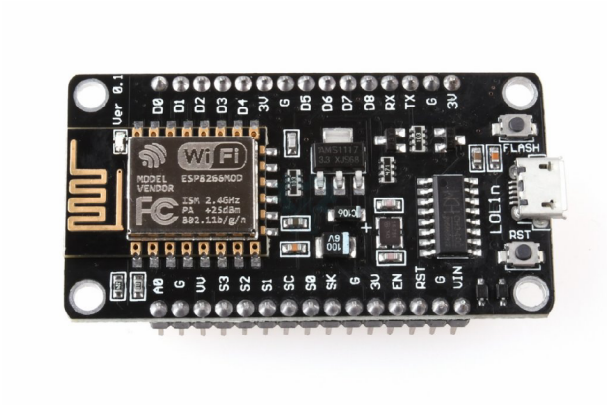
Tato podkapitola obsahuje moduly, které jsou použité v daném řešení chytrého řízení ohřevu bojleru. Konkrétní zařízení sestaveno v kontextu této práce je MCU využívající modul *ESP8266* doplněné o dvojici teplotních senzorů *DS18B20* pro snímání teplot nádrže. Dále pak chytrá zásuvka *TP-Link HS110* pro monitorování spotřeby a spínání ohřevu.

Mikrokontroler obsahující čip ESP8266

ESP8266 je dobře dostupný a levný Wi-Fi mikročip, který v kombinaci s několika komponenty tvoří výtečný mikrokontroler. Mikrokotrolery využívající právě tento mikročip, jsou velmi oblíbené v oblasti chytrých zařízení a to díky své stabilitě, výkonu, přijatelné velikosti a dobré ceně. Na trhu je dostupných mnoho mikrokontrolérů obsahujících tento mikročip.

K nejoblíbenějším patří *Wemos D1 mini* a *NodeMCU*. Tato práce využívá právě mikrokontroleru *NodeMCU* pro jeho spolehlivost, širší možnosti připojení periferii nízkou cenu pohybující se kolem 150kč k dni 12.4 2021.

[3] **M2M** - Machine to Machine



Obrázek 5.7: NodeMCU mikrokontroler. Převzato z [25].

NodeMCU je open source firmware založeny na Lua kombinovaný s vývojovou deskou se zaměřením na *IoT* aplikace. Zahrnuje firmvare provozovaný na ESP8266 Wi-Fi SoC[4] od Espressif System a hardware který je založen na modulu ESP-12. [2]

Mezi jeho klíčově vlastnosti patří:

- Napájení pomocí microUSB
- Až 16 DIO pinů
- 4MB flash memory
- 3x 3.3V piny a 4x GND piny
- Rozměry 58mm x 32mm

Teplotní čidlo DS18B20

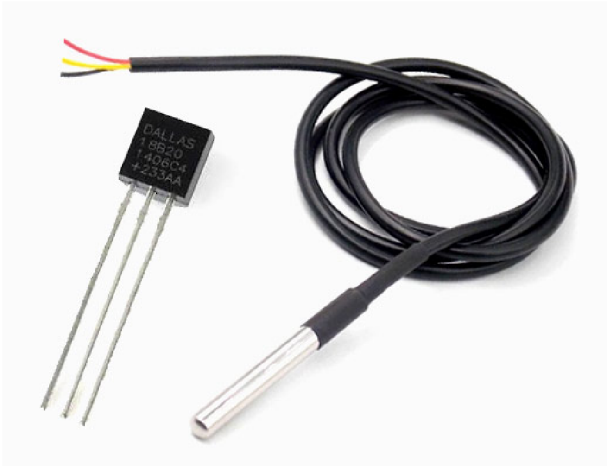
DS18B20 je 1-vodičový programovatelný teplotní senzor od společnosti Maxim Integrated, široce používán k měření teplot v náročných prostředích. Konstrukce senzoru je robustní a lze ji rovněž dostat ve vodotěsném provedení. Měří v širokém rozsahu teplot od -55°C do +125°C s přesností ±0.5°C. Každý senzor má jedinečnou adresu a k přenosu vyžaduje pouze jeden pin, co je výhodné v situaci kdy se vyžaduje úspora pinů. [1]

Cenově se tento senzor pohybuje kolem 80kč k dni 12.4 2021.

Mezi jeho klíčově vlastnosti patří:

- Operační napětí 3V - 5V
- Komunikace prostřednictvím 1 pinu
- Přesnost 0.5 °C
- Rozsah teplot -55°C do +125°C
- Vodě odolné provedení
- Čas převodu na 12 bitů 750ms

[4] **SoC** - System on Chip



Obrázek 5.8: DS18B20 senzory (vodotěsná verze na pravé straně). Převzato z [1].

TP-Link HS110

Obrázek 5.9: TP-Link HS110 chytrá zásuvka. Převzato z [26].

HS110 je chytrá zásuvka firmy TP-Link s ovládáním prostřednictvím Wi-Fi sítě a měřením spotřeby, jak aktuální tak i celkové. Skýtá též možnost nastavování různých časovačů a ovládání pomocí mobilní aplikace. Z hlediska teto práce je však podstatné dálkové spínání a měření spotřeby energie, co je podporováno v PC implementaci výše zmíněnou knihovnou python-kasa. [34]

Mezi další pro práci klíčově vlastnosti patří:

- Připojení Wi-Fi 2.4GHz, IEEE 802.11b/g/n
- Výstupní napětí 100V - 240V
- Maximální zátěž 16A
- Maximální výkon 3680W

Kapitola 6

Implementace

V této kapitole je zahrnuto, jakým způsobem bylo dané řešení konverze standardního bojleru na chytrý implementováno. Návrh řešení a použité technologie byly již představeny v předchozích kapitolách (Návrh kapitol 4, Technologie 5). Návrhová část plní roli směrodatné specifikace, obsahující potřebné znalosti k implementaci daného řešení. V kapitole pojednávající o technologické stránce byly pak zmíněny důvody výběru jednotlivých technologií.

Systém konverze klasického bojleru na chytrý je tématem, které v sobě zahrnuje různorodá prostředí. Lze hovořit o kompozici několika části serverové do niž spadá prostředí jak rovněž skripty řízení a hardwarová zařízení. Během implementace jednotlivých části docházelo k drobným modifikacím jejich chování, případně struktury z ohledu na vzniklé potřeby. Z pohledu funkčnosti jako celku, ale tyto změny nijak neovlivnily finální produkt představující spolehlivý systém řídící ohřev vody dle předpokládané spotřeby.

Z důvodů, že implementace systému obsahuje několik různých aspektů, je tato kapitola rozdělena na dvě hlavní podkapitoly věnující se jednotlivým částem. V podkapitole 6.1 je představena funkcionalita serverové části z hlediska prostředí a nejpodstatnějších části systému řízení. Podkapitola 6.2 se věnuje HW zařízením, jejich rolím, struktuře a funkcionalitě v rámci celého systému.

6.1 Server

Tato podkapitola obsahuje implementaci serverové části sytému, jejíž hlavním těžištěm je skript řídící spínání ohřevu na základě vypočtené předpovědi. V další řadě rovněž prezentuje implementaci prostředí požadovaného k zajištění běhu řídícího skriptu, tím se myslí i jeho celá koncepce včetně skriptu. Druhá podkapitola se zaměřuje na periferní zařízení zakomponované do systému a jejich specifické řešení s funkcionalitou.

Z konceptuálního hlediska se jedná o systém, který po montáži periferii (zásuvka a snímač teplot) využívá sérii prvků instalovaných a spuštěných na serveru v domácnosti tak, aby s jejich pomocí ovládal ohřev vody v bojleru. Z tohoto pohledu je nutné dodržet separaci jednotlivých prvků ze strany běhu a zajistit mezi nimi jenom nutnou komunikaci, aby byla zachována robustnost celého systému, jak rovněž zjednodušená oprava v případě výpadku způsobeného např. výpadkem ze strany hardwarů. Po úspěšné instalaci, by tento systém měl běžet na pozadí každodenního života a aktivně se přizpůsobovat změnám v oblasti spotřeby vody, prakticky bez nutnosti interakce s uživatelem.

K takovémuto ideálnímu konceptu směřuje i jeho implementace vedená systémovým návrhem.

Serverové prostředí

Server musí zajišťovat prostředí pro neustály běh systému. Proto jsou všechny prvky systému, databáze, agenti pro komunikace s periferiemi, vizualizační prostředí i řídící skript implementovány, jako série služeb (dále jen *SBservices*) běžících na pozadí operačního systému serveru. Přitom se u některých z nich, jako je databáze, MQTT server, agent pro komunikaci databáze s MQTT serverem a vizualizační aplikace, využívá kontejnerizační aplikace *Docker*. Kdy tento způsob implementace je velmi vhodný, jelikož zmenšuje náročnost samotné instalace a hlavně velmi efektivně odděluje jednotlivé prvky od sebe, čímž zjednodušuje opravy.

Aby bylo zajištěno jejich automatického spuštění v případě výpadku např. elektřiny, tak jsou implementovány pomocí takzvaných *unit files* tj. souborů s příponou *.service*, které jsou strukturálně rozděleny na tři části.

V první sekci se nachází popis služby která je jím zastřešena a její závislosti. Sekce druhá pak definuje její cíl, to jest co a jakým způsobem má být spuštěno. Poslední sekce stanovuje, kdy se služba spustí.

Zajištění běhu všech *SBsevices* je implementováno stejným způsobem, pouze s rozdíly v rámci *.service* souborů vyžadovanými specifickým charakterem služby.

Veškeré chyby, které můžou vzniknout za běhu těchto služeb jsou zaznamenávaný do logovacího souboru operačního systému, odkud je lze v případě potřeby vyčíst.

Z pohledu závislostí mezi prvky je implementace provedena následujícím způsobem. Databáze stanoví část na které závisí nejvíc prvků, proto je jediná již je přiřazena vlastní paměťový svazek v rámci kontejneru. To je provedeno z důvodu persistence dat při výpadku a aby tím způsoben nedošlo k jejich ztrátě, co by vedlo k změně chování řídicího algoritmu (objasnění v následující podkapitole).

Řídící skript

Je těžištěm celého systému a jeho implementace by se měla shodovat co se postupu výpočtu týče s návrhovou části této práce viz. kapitola 4.5, která se zaobírá právě veškerými výpočty a postupy nutnými pro správné fungování. Čistě z implementačního a funkcionálního hlediska jej lze rozdělit na několik částí. Kde každá z nich zahrnuje funkce zaobírající se konkrétní problematikou.

První z nich se věnuje výpočtům v oblasti datum a jejich využití při vytváření dotazu na databází. Ke kterým se využívá patřičná uživatelská knihovna ulehčující jejich formulaci ve skriptu.

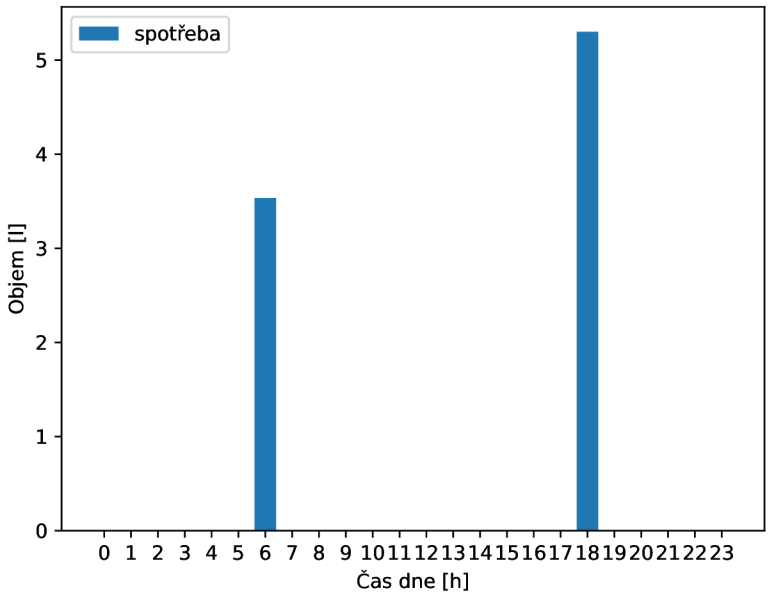
V části druhé je implementace veškerých výpočtu v oblasti termodynamiky a výkonu, které se využívají k vyjádření časových, teplotních a objemových veličin na nichž závisí právě, jak přesnost predikce, tak v pozdější fází i čas, využívaný plánovací části algoritmu.

V další části je implementována filtrace dat získaných pomocí dotazů, které účelem je vyhledání podstatných sekcí z hlediska předpovídání, aby nebylo třeba pracovat s nadbytečně rozsáhlými daty. Zde stojí za zmínění funkce implementující vyhledávání padající sekvence teplot podle návrhu 4.5, která byla doplněna o kalibrační konstantu 0.2 stupně z důvodu vlivu změny teploty okolního vzduchu viz. výpis 6.1,

```
if float(actual) < float(prev) and float(actual) < float(prev_prev) and (not falling):
    if (float(prev_prev) - float(actual)) >= 0.2:  # Calibration to not detect falling air
    temperature
        index_list.append(i - 2)
        falling = True
elif float(actual) > float(prev) and float(actual) > float(prev_prev) and falling:
    index_list.append(i)
    falling = False
```

Výpis 6.1: Část funkce implementující detekci klesající řady.

jak rovněž proměnnou vyjadřující zda li se již nacházíme v padající sekvenci aby nedocházelo k zaznamenání bodů, kdy se teplota zvedla ale nenachází se v započaté padající sérii. Co zaručuje značně přesnější detekci a odstraňuje nedostatek návrhu.



Obrázek 6.1: Spotřeba vody v průběhu dne.

V části čtvrté se tak nachází funkce zpracovávající získané data nejprve do formy řady 24 intervalů, které znázorňují spotřebu vody v rámci jednoho dne. Takovou řadu vidíme na obrázku 6.1 zobrazenou formou histogramu spotřeby. Jak rovněž je zde implementovaná funkce pro výpočet předpovědi budoucího vývoje, které výsledkem je stejně vyhlížející řada 24 intervalů. S rozdílem v tom, že je vytvořena pomocí 2 až 4 historických řad. Její vypracovaní zajišťuje funkce z výpisu 6.2.

```
def predict(list_of_usage_lists):   # list of lists [oldest data, -> ,newest data]

    p = []
    n = len(list_of_usage_lists)

    if n == 2:
        u1 = list_of_usage_lists[1]
        u2 = list_of_usage_lists[0]
        for i in range(0, 24):
            p.append(round(ema([u2[i], u1[i]], n), 2))
    elif n == 3:
        u1 = list_of_usage_lists[2]
        u2 = list_of_usage_lists[1]
        u3 = list_of_usage_lists[0]
        for i in range(0, 24):
            p.append(round(ema([u3[i], u2[i], u1[i]], n), 2))
    elif n == 4:
        u1 = list_of_usage_lists[3]
        u2 = list_of_usage_lists[2]
        u3 = list_of_usage_lists[1]
        u4 = list_of_usage_lists[0]
        for i in range(0, 24):
            p.append(round(ema([u4[i], u3[i], u2[i], u1[i]], n), 2))

    return p
```

Výpis 6.2: Funkce implementující predikci budoucích hodnot s využitím EMA a sérii historických řad.

Kde můžeme pozorovat, že v závislosti na množství historických dat o spotřebě N jsou patřičně dosazeny parametry funkce *ema* zajišťující výpočet hodnoty pro konkrétní denní interval.

Předposlední část pak stanoví řadu logických funkcí, které implementují chování algoritmu v jeho jednotlivých fázích. Co znamená, že volají v dané posloupnosti jiné funkce a to tak, aby mohlo dojít k bezpečnému výpočtu předpovědi pokud je k dispozici dostatečné množství dat. To jest alespoň 14 denní historie spotřeby. V případě, kdy je systém v provozu méně než 14 dnů, pak se nachází v fázi kolekce dat a jeho chování je řízeno jednoduchým statickým denním modelem definovaným na výpisu 6.3.

```
def baseSwitching(sched):
    d = datetime.date(datetime.now())

    t_on = str(d) + " " + "01:00:00"
    t_off = str(d) + " " + "06:00:00"
    t_on2 = str(d) + " " + "13:00:00"
    t_off2 = str(d) + " " + "14:00:00"

    sched.add_job(func=turnOn, trigger='date', next_run_time=t_on)
    sched.add_job(func=turnOff, trigger='date', next_run_time=t_off)
    sched.add_job(func=turnOn, trigger='date', next_run_time=t_on2)
    sched.add_job(func=turnOff, trigger='date', next_run_time=t_off2)
```

Výpis 6.3: Funkce implementující chování systému v případě kdy není k dispozici předpověď.

Kde je ohřev v provozu během nočních hodin a potenciálního nízkého tarifu a následně jednu hodinu odpoledne pro doplnění zásoby teplé vody. Tento model je využitý i v případě výpadku snímače teplotních dat. Co by zapříčinilo absenci hodnot spotřeby v době výpadku a tím vneslo chyby do předpovědi.

Hodiny jsou také voleny z ohledem na čas, ve kterém je spotřeba vody nejnižší. Jelikož současný ohřev a odběr způsobuje nepřesnosti v budoucích přepočtech rozdílu teplot na objem. Přesněji ohřev během spotřeby způsobí menší rozdíl teplot, co vede na menší objem spotřebované vody po přepočtu.

Hlavní rozhodovací část je implementována ve funkci *makeForecast*, která provede všechny dílčí části nutné k vypracovaní předpovědi a následně na jejím základu zajistí naplánování dob spínání ohřevu pomocí zásuvky a časového plánovače *APScheduler*. V případě, kdy není možné zajistit přesnou předpověď, ať už z důvodu chyby nebo nedostatku dat, je využito funkce z obrázku 6.3.

Poslední část algoritmu je tvořena inicializací potřebných proměnných, plánováním základních funkcí, které je nutné provádět periodicky a nekonečným cyklem zajišťujícím konstantní běh skriptu. Jak je patrné z obrázku 6.4, tak 2 hlavní funkce stanoví právě výše zmíněná *makeForecast* a *checkLimitTemp*, která obstarává aby teplota vody v nádrži neklesla pod minimální dovolenou teplotu.

```
client = influxdb.InfluxDBClient(host='localhost', port=8086, username='telegraf', password
    ='telegraf',database='sensors')

plug = kasa.SmartPlug(plugIP)

scheduler = BackgroundScheduler()
scheduler.start()
scheduler.add_job(func=makeForecast, args=[scheduler], trigger='cron', hour='0', minute='15
    ')
scheduler.add_job(func=checkLimitTemp, args=[scheduler], trigger='interval', minutes=5)

while True:
    time.sleep(1)
```

Výpis 6.4: Hlavní část skriptu.

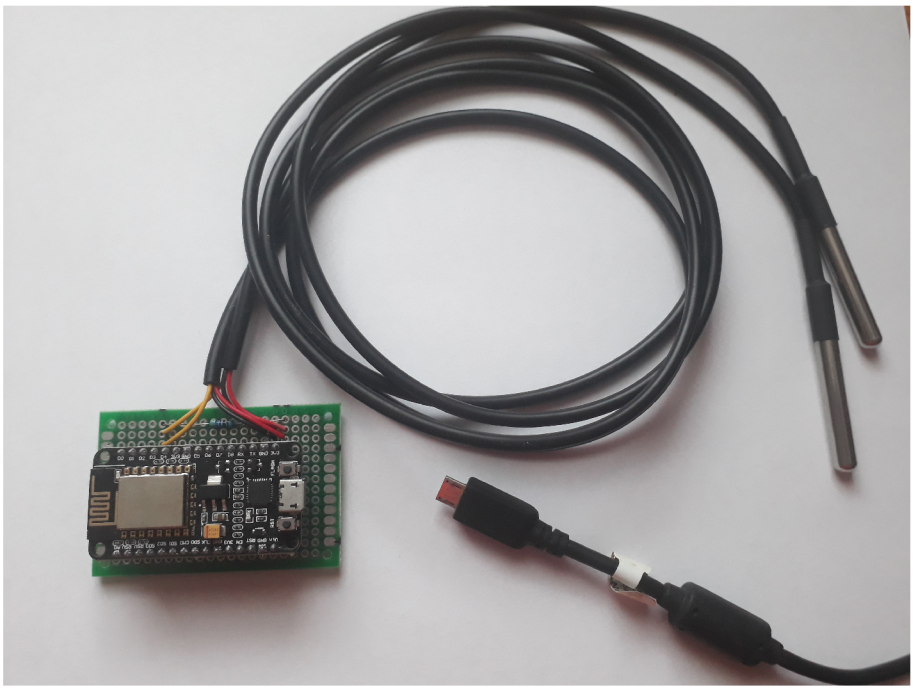
Tímto postupem je zajištěna funkcionalita řídícího algoritmu a tím i dostatek teplé vody ze strany uživatele systému.

6.2 Periferní zařízení

Implementace periferií je nezbytnou součásti této práce. Jelikož právě jejich prostřednictvím jsou sbírány informace o stavu systému, stejně jako je zapínán okruh ohřevu. Z pohledu spínání, nebylo třeba implementovat nějaké nové speciální zařízení, ale využilo se již existujících možností (chytrá zásuvka TP-Link HS110). Jediné, co bylo třeba implementovat je skript pro odečet spotřeby elektrické energie, aby pomocí něj mohlo dojít k zhodnocení efektivity celého systému. To je provedeno formou nekonečného cyklu implementovaného podle návrhu viz. kapitola 4.3, kde se v 5 sekundových intervalech zasílá do databáze informace o spotřebě, ty kromě zhodnocení efektivity mohou být zobrazeny uživateli formou grafu v aplikaci Grafana.

V rámci periferií, tak významnější část stanoví sestavení a programová implementace mikrokontroleru pro snímání teplot a jejich zasílání do databáze. Z pohledu sestavení hardware byl využit mikrokontroler *NodeMCU ESP8266* jako základní platforma. Ke které se připojuje dvojice teplotních senzorů *DS18B20* ve vodě odolném provedení, které je vhodnější z hlediska prostředí v nímž jsou senzory umístěny. Zejména senzor na trubce studené vody, jelikož právě ta je vystavena vysokému tepelnému rozdílu zapřičiňujícímu kondenzaci vzdušné vlhkosti a to i na povrchu senzoru.

K propojení *NodeMCU ESP8266* a dvojice *DS18B20* senzorů se využívá prototypovací desky plošných spojů, na kterou je naletován mikrokontroler se senzory a $4.7k\Omega$ rezistorem podle schématu zapojení nacházejícího se v Příloze A. Výsledné zařízení prezentuje obrázek 6.2 Díky tomu se dosáhne pevné, trvalé konstrukce, kterou je možno umístit dodatečně do ochranného pouzdra.



Obrázek 6.2: Finální podoba zařízení snímajícího teplotu nádrže a trubky se studenou vodou.

Jak je patrné z obrázku, k napájeni tohoto zařízení se využívá jeho microUSB rozhraní, co je velmi výhodné konstrukčně i uživatelsky. A v případě, kdyby byl osazen v dodatečném pouzdře, pak by nebyl problém fixovat jej přímo na bojler pomocí oboustranných lepek.

Sestavený mikrokontrolér, je pak rozšířen o *microPython* interpret, ve kterém je implementována jeho softwarová část. Ta je rozdělena na 2 části, *boot.py* a *main.py*. První z nich je spuštěna už při bootování zařízení a setrvá se v ní až do jejího úspěšného provedení, které je podmíněno připojením k Wi-Fi síti, jelikož bez ní nelze odesílat data o teplotě.

Část druhá dále implementuje samotné čtení hodnot ze senzoru a jejich zasílání na MQTT server. Zde je pozornosti hodná zejména implementace funkce čtení hodnot ze senzoru viz. výpis 6.5.

```
def readSensor():
    try:
        roms = ds_sensor.scan()
        ds_sensor.convert_temp()
        time.sleep_ms(750)
        sensors_temperatures = []

        for rom in roms:
            sensor_temp = ds_sensor.read_temp(rom)

            if isinstance(sensor_temp, float) or (isinstance(sensor_temp, int)):
                reformat_temp = ('{0:3.1f}'.format(sensor_temp))
                sensors_temperatures.append([decodeByteArray(byte_arr=rom), reformat_temp])
            else:
                print("Invalid sensor readings format.")
                return []

        return sensors_temperatures
    except OSError:
        print("Failed to read sensor.")
        restartAndReconnect()
```

Výpis 6.5: Funkce pro čtení tepelných hodnot ze senzoru DS18B20.

Kde si lze povšimnout právě nutnosti čekání po dobu 750 ms z důvodu čtení teploty v 12 bitovém rozlišení, které se následně uloží pro další zpracování a odeslání na server.

Implementace chování celého mikrokontroleru se řídí chováním automatu nacházejícímu se v návrhové části této práce a konkrétně obrázek 4.1 v kapitole 4.2.

Nasazení

Typický případ použití takto implementovaného systému, si lze představit v domácnosti, která jako zdroj teplé vody používá bojler ohřívající vodu i mimo dobu nízkého elektrického tarifu. Přičemž rovněž skýtá alespoň jeden počítač, který je neustále v provozu bez ohledu na přítomnost osob. V takové situaci je systém po správně provedené instalaci schopen nabídnout řešení snižující spotřebu energie na ohřev vody a tím i cenu provozu bez negativního efektu omezujícího komfort uživatele, co se týče dostupnosti teplé vody. Hlavním pozitivním rysem by měl být fakt, že systém po úspěšné instalaci nevyžaduje další interakci s jeho uživatelem, čím nepředstavuje ani žádnou citelnou změnu oproti klasickému bojleru.

Kapitola 7

Testování

Obsah této kapitoly je zaměřen na testování celého systému pro řízení ohřevu vody v bojleru, jeho efektivity a přesnosti předpovědi spotřeby. Důležitým faktorem daného řešení byla efektivnost a přesnost předpovědi spotřeby teplé vody. Kde by s rostoucí přesností mělo docházet k postupnému snižování spotřebované energie.

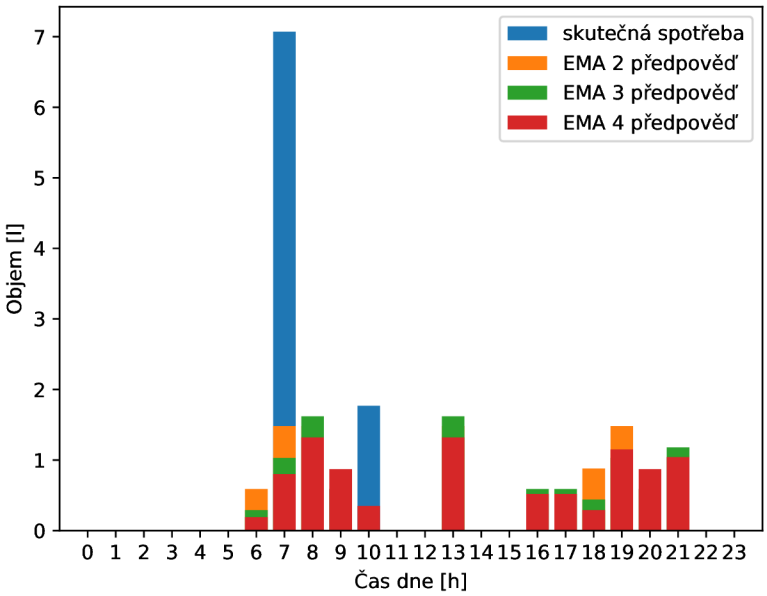
Podkapitola 7.1 se věnuje testování přesnosti předpovědi v závislosti na množství nasbíraných dat a srovnáním energetické spotřeby systému v běžné sestavě a v sestavě doplněné o produkt vytvořený v rámci této práce. V podkapitole 7.2 je obsah věnován využití systému v praxi s popisem konkrétního prostředí v němž byl nasazen. Zahrnuje rovněž hodnocení jeho stability, evaluaci požadavků a zhodnocení jeho finanční stránky. Poslední podkapitola pak zahrnuje reflexe na téma systému a nápady pro jeho vylepšení jenž vyplynuly během provozu.

7.1 Přesnost funkce předpovědi a dosažená úspora

Funkce predikující vývoj spotřeby teplé vody v domácnosti, plní velmi důležitou roli v celém systému inteligentního řízení bojleru. Právě díky její přesnosti předpovědi se zajišťuje ušetření energie, která se využívá pro ohřev vody. Přesněji je voda ohřáta pouze na takovou teplotu aby po vzniklé spotřebě X litrů, její teplota klesla k 40°C, bez podkročení. To znamená, že se voda neohřívá na příkladových 70°C stanovených termostatem, kde je udržována, ale místo toho je ohřáta pouze na 48°C, protože tato teplota vystačí k uspokojení spotřeby v daném dni.

Hlavním požadavkem vůči predikující spotřebu byla tudíž stabilní přesnost předpovědi hodnot, jelikož právě taková předpověď zajišťuje dobrou úsporu z dlouhodobého hlediska. Tyto vlastnosti byly testovány na shromážděných datech o provozu za poslední 2 měsíce.

Testy byly prováděny formou série předpovědí pro jednotlivé dny týdne, kde se pozorovalo chování předpovídaných hodnot pomocí funkce *EMA(2)*, *EMA(3)* a *EMA(4)* ve srovnání s reálnou spotřebou, která nastala v předpovídaném dni. Přičemž číslo ve funkci *EMA()* znamená počet dnů použitých pro vytvoření předpovědi spotřeby v jednotlivých časových bodech. Výsledky poskytly velmi zajímavý pohled na chování jednotlivých předpovědí.



Obrázek 7.1: Srovnání předpovědi vypočtených pomocí EMA různé délky (oranžová, zelená, červená) vůči reálné spotřebě vzniklé v predikovaném dni (modrá). Přičemž reálná spotřeba měla zde konkrétně charakter anomálie (kumulace celého běžného dne v dopoledních hodinách).

Na obrázku 7.1 můžeme pozorovat typický tvar rozložení předpovídaných hodnot ve srovnání s reálnou spotřebou. Stanovící vhodný příklad chování jednotlivých funkcí *EMA(x)*.

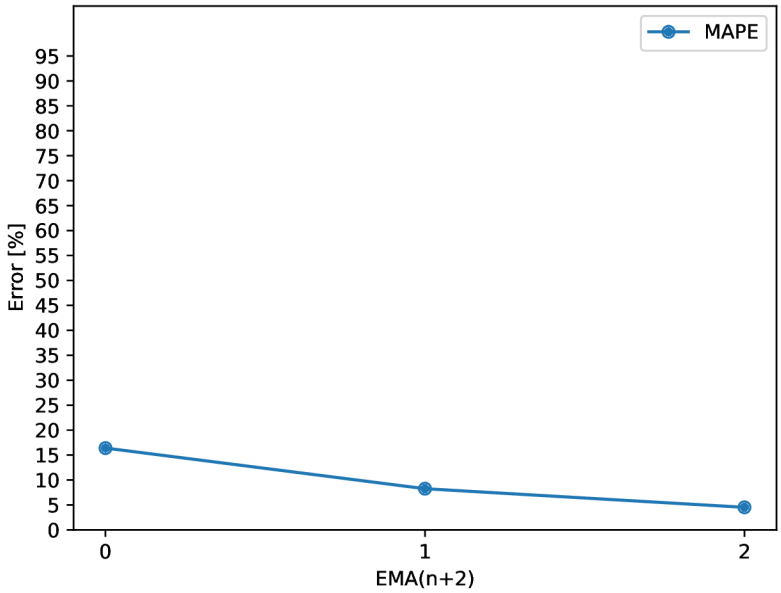
Toto chování *EMA(x)* v závislosti na velikosti x se charakterizuje následujícími znaky, jak vyplynulo z testování. Při použití x velikosti 2 předpovídané hodnoty občas prokazovaly vyšší přesnost předpovědi nežli tomu bylo u x velikosti 3 a 4, ale to za doprovodu citelně vyšší variace rozsahu. Co se odráží na stabilitě předpovědi a značně vyšší potřebě dodatečného ohřívání vody v neplánovaných dobách z důvodu poklesu pod spodní teplotní hranici nádrže. To jelikož suma předpovídané spotřeby byla menší než reálná spotřeba v daném časovém úseku.

Po zvýšení x na hodnotu 3, došlo k zlepšení těchto nevhodných vlastností souvisejících z kolísáním přesnosti předpovědi, co souvisí s širším rozložením spotřeby v průběhu času namísto sdružení větších hodnot do menšího počtu intervalů, jak je patrné i na obrázku 7.1. To v souvislosti s delší historií spotřeby. Opět je ale viditelné, že nebyly pokryty všechny intervaly, kde se vyskytla reálná spotřeba.

Při použití *EMA(x)*, kde x nabývá hodnoty 4, už lze pozorovat předpověď spotřeby, která se v jistém smyslu podobá pravděpodobnostnímu rozdělení výskytu spotřeby v daném časovém bodě, co u většiny předpovědi vedlo na plné pokrytí intervalu s reálnou spotřebou s intervaly kde byla spotřeba předpovídaná. Taková skutečnost sebou váže výhodu, která se může projevit zejména v dopoledních hodinách, kdy je voda připravena s před první spotřebou, co nemusí být splněno v případě použití *EMA(2)*, kdy se spotřeba vyskytne před očekávanou dobou. A jelikož je příprava teplé vody rozdělena na 2 období, před prvním

použitím a dohřáti v odpoledních hodinách, tak je charakter předpovědi *EMA(4)* velmi žádoucí.

To se odráží i na průměrné procentuální chybovosti jednotlivých předpovědí ve srovnání s realitou, jak je vidět na obrázku 7.2, zobrazujícímu *MAPE* hodnotu pro tentýž příklad s pohledu na sumu spotřeby.



Obrázek 7.2: Průměrné procentuální chybovost při EMA využívající různý počet historických dat.

Kde lze pozorovat pokles chyby předpovědi s rostoucím množstvím dat pomocí kterých byla vypočtena. Opět je nutno zmínit, že takový průběh, jak je reprezentován na grafu není garantovaný vždy, kdy *MAPE* hodnota pro *EMA(2)* je občasně nižší než li *EMA(4)* nebo *EMA(3)*. Stejně tak pro *EMA(3)*. Avšak z dlouhodobého hlediska kde se uváží všechny předpovědi, tak lze ztvrdit, že hodnota *MAPE* pro *EMA(4)* dosahuje nejlepších hodnot.

Úspory

Tím se dostáváme k druhému tématu testování kterým je úspora dosažená, jako důsledek nasazení systému. Ze shromážděných dat vyplynulo, že denní průměrná spotřeba energie bojleru před nasazením systému konverzujícího klasický bojler na chytrý činí **2609 Wh**. Po nasazení systému vytvořeného touto prací došlo k jejímu poklesu na denní průměr činící **2175 Wh**. To znamená úsporu ve výši **16.63%**. Která při ceně elektřiny 4.76 kč za 1 kWh činí denní úsporu ve výši **2.06 kč** v případě využití vysokého tarifu. V případě nízkého tarifu 2.82 kč za 1 kWh pak tato úspora klesne na **1.22 kč** za den.[14]

7.2 Systém v praxi

Systém byl do provozu nasazen v dvoučlenné domácnosti, tvořené osobami staršími 60-ti let, kde je každodenně přítomen aspoň jeden člen po celou denní dobu. Nainstalován byl na bojler *Wterm AQUASTIC AQ 80* s příkonem 2400 W a objemem 80l. Kde tento typ bojleru postrádá šachtu pro analogový teploměr.

Během provozu nevznikla situace, kdy by došlo k nedostatku teplé vody způsobeným běžným používáním.

Jako server byl použit notebook značky ASUS model F5VL s procesorem *Intel Pentium Dual CPU 1.73GHz*, *2 GB DDR2* operační pamětí, 160 GB pevným diskem a operačním systémem *Ubuntu 20.04 LTS*.

Po stránce finančních nákladu implementace systému pro danou domácnost, byla nejdražší chytrá zásuvka *TP-Link HS110*, které cena činila přibližně 620 kč. U zařízení pro snímaní teplot největší investici stanovil mikrokontroler *NodeMCU ESP8266*, kterého cena byla ve výší 140 kč, pak dvojice vodě odolných teploměrů *DS18B20* za 80 kč kus, následně deska plošných spojů v ceně 20 kč a $4.7k\Omega$ rezistor za 1 kč. Celková cena implementace systému pro konverzi klasického bojleru na chytrý tak vyšplhala na **941 kč**, při pořízení všech součástek v rámci České republiky. Při pořízení součástek ze zahraničí, by tak mohlo dojít k redukci této částky.

7.3 Možná budoucí vylepšení

Není žádný nový systém, který by neposkytoval možnosti na vylepšení nebo vyřešení jisté problematiky jinak, tak aby bylo dosaženo lepší efektivity. Během provozu a testování se odhalilo několik možnosti, jak by tento systém mohl být do budoucna vylepšen. Tato podkapitola je věnována právě jim a je člení se do několika části, kde každá je věnována rozdílné modifikaci systému.

Serverová implementace

Ze strany serveru se jeví, jako vhodné budoucí vylepšení vyniklé z provozu a nasazení, implementace aplikace nebo nějakého instalačního klienta, který by řadou dotazů umožnil jednoduchou instalaci na server probíhající v duchu klasického instalování nového programu. Co by bylo nezbytné pokud by mělo jít o komerční plug and play řešení.

Dalším možným vylepšením je optimalizace metody pro předpovídání spotřeby vody, která by umožnila dosáhnout ještě přesnějšího odhadu, čím by se dosáhlo rovněž dalšího zvýšení dosažených úspor. Z toho hlediska by bylo třeba provést širší testování a srovnání vhodných metod předpovědi např. z řady autoregresivních klouzavých průměru.

Hardware

Z pohledu hardwaru, hraje značnou roli chytrá zásuvka použitá v systému pro monitorování spotřeby, jak i spínáni ohřevu. Proto by ona mohla být dalším možným vylepšením, například ze strany maximálního výkonu, kde by se použilo zařízení, které nemá definovanou jeho maximální hodnotu. Ačkoliv většina typů bojleru používaných v domácnosti nepřesahuje hranici výkonu 3500W.

Další možností vylepšení je náhrada teplotního čidla, umístěného na trubce pro přívod čerstvé vody do nádrže, průtokoměrem např. model YF-S201 jehož cena se pohybuje kolem

130 kč. Ve výsledku by tedy byla dosažena maximální možná přesnost při záznamu použité vody, jelikož by nebylo třeba dopočítávat ji na bázi rozdílu teplot, co sebou nese jistou míru nepřesnosti, a tak i malý vliv na výslednou úsporu.

Eventualitou průtokoměru může rovněž být snímač vibrací, neboť při průtoku vody trubkou vznikají mírné vibrace šířící se jejím povrchem. Snímač vibrací by ale měl stejnou nevýhodu, jako teplotní čidlo, jelikož by opět bylo třeba dopočítat objem spotřebované vody.

Zbylé postřehy

Další možné změny vyplynuly z reakcí a chování uživatelů a potenciálních uživatelů. Prvním z nich je minimální potřeba interakce ze systémem, to jest potřeba využívání grafické reprezentace teplotních průběhů atd. Kde jedinou položkou, která představovala eventuální bod zájmu byla hodnota spotřebované energie sporadicky rovněž aktuální teplota. Proto by do budoucna bylo možné podstatně redukovat hodnoty zobrazované systémem např. do formy jedné webové stránky s dvěma ukazateli.

Druhou je rozšířeni systému o funkcionalitu typu prázdniny, kde by uživatel pomocí jednoho přepínače mohl aktivovat tento speciální režim, čím by přepl systém do jakéhosi režimu spánku, ve kterém by setrval do doby opětovné změny stavu tohoto přepínače.

Také se nabízí přidání funkce, která by zajistila, v jistém časovém rozmezí, ohřev vody v nádrži na teplotu přesahující 50°C, aby tím došlo k zahubení mikroorganizmu, které by se mohly v nádrži vyskytnout.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit sytém řešící konverzi standardního bojleru na chytrý při využití chytré zásuvky. Tento záměr byl splněn, celý systém je funkční a byl úspěšně nasazen do reálného provozu.

Důležitým faktorem práce, bylo dosáhnutí nízké pořizovací ceny kompletního systému, aby řešení stanovilo levnou alternativu místo pořízení chytrého bojleru. Čeho bylo dosaženo pomocí adekvátní volby hardwarových komponentů. Přitom bylo potřeba zachovat univerzálnost použití tohoto řešení, pro jeho aplikaci u různých typu bojlerů. Velmi významným rysem práce je rovněž efektivita celého systému, prokázána výrazným snížením spotřeby elektrické energie při ohřevu vody, po jeho instalaci.

Náklady na pořízení tohoto systému činí přibližně 940 kč, což představuje asi jednu sedminu ceny nového chytrého bojleru. Přičemž to při úsporách v rámci spotřebované energie dosahujících téměř 17% znamená, že v průběhu necelých 2 let dojde k vrácení jeho pořizovacích nákladu. To je dosaženo díky využití historického záznamu o spotřebě, na jejíž bázi je vypracovávána předpověď pomocí exponenciálního klouzavého průměru používaná pro efektivní řízení ohřevu vody prostřednictvím chytré zásuvky.

V práci bych chtěl pokračovat, protože si myslím, že chytrá zařízení tohoto typu jsou cesta k dosažení efektivnějšího využití energie a prostřednictvím toho i pozitivního efektu na životní prostředí. Proto bych se dále zaobíral možností zvýšení efektivity tohoto systému cestou přesnějších metod pro předpověď. Aktuální stav projektu dostupný na odkazu[1].

Na tuto práci by rovněž mohl někdo navázat například navržením speciálního zařízení integrujícího všechny části tohoto systému na jednom místě, čím by vzniklo jedno zařízení, které by po montáži obstarávalo celé řízení bojleru bez přítomnosti serveru.

Práce mi dala spoustu zkušeností z různých sfér IT a naučila mě, jak může být občas obtížné navrhnout a realizovat kvalitní systém čerpající z různých odvětví vědy, které jsou u chytrých zařízení nezbytné. Jelikož právě ty stanoví trend, kterým směřuje vývoj věcí každodenního užitku.

[1] https://github.com/xmitur01/SmartBoiler

Literatura

[1] 101, C. DS18B20 Temperature Sensor. *Components 101* [online]. Květen 2018 [cit. 2021-4-14]. Dostupné z: https://components101.com/sensors/ds18b20-temperature-sensor.

[2] 101, C. NodeMCU ESP8266. *Components 101* [online]. Duben 202O [cit. 2021-4-14]. Dostupné z: https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet.

[3] Contributors, W. Internet of Things. *Wikipedia* [online]. 2020 [cit. 2020-11-05]. Dostupné z: https://en.wikipedia.org/wiki/Internet_of_things.

[4] Contributors, W. MQTT. *Wikipedia* [online]. 2020 [cit. 2020-12-18]. Dostupné z: https://en.wikipedia.org/wiki/MQTT.

[5] Corporation, S. *MQTT* [online]. 2020 [cit. 2020-12-18]. Dostupné z: https://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT%20Control%20Packet%20format.htm.

[6] developers python-kasa. python-kasa. *Read the docs* [online]. 2020 [cit. 2021-4-12]. Dostupné z: https://python-kasa.readthedocs.io/en/latest/index.html.

[7] Docker, I. Docker docs. *Wikimedia commos* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://docs.docker.com/get-started/overview/.

[8] dotCloud, I. Docker (container engine) logo.png. *Wikimedia commos* [online]. 2013 [cit. 2021-4-12]. Dostupné z: https://commons.wikimedia.org/wiki/File:Docker_(container_engine)_logo.pn.

[9] dzd.cz. *NÁVOD K OBSLUZE A INSTALACI OKHE 80,100,125,160 - SMART.* Benátky nad Jizerou: Družstevní závody Dražice - strojírna s.r.o., březen 2020 [cit. 2021-4-14].

[10] Foundation, E. Paho. *Eclipse Foundation* [online]. 2015 [cit. 2021-4-12]. Dostupné z: https://wiki.eclipse.org/Paho.

[11] Foundation, E. Eclipse Mosquitto. *Eclipse Foundation* [online]. 2020 [cit. 2021-4-12]. Dostupné z: https://projects.eclipse.org/projects/iot.mosquitto.

[12] Foundation, E. Paho. *Eclipse Foundation* [online]. 2020 [cit. 2021-4-12]. Dostupné z: https://www.eclipse.org/paho/.

[13] Grönholm, A. Advanced Python Scheduler. *Read the docs* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://apscheduler.readthedocs.io/en/stable/.

[14] Hamalčíková, K. Cena elektřiny za kWh opět zdražila. V roce 2020 stojí 4,76 Kč. *Elektřina.cz* [online], 9. března 2020 [cit. 2021-4-21]. Dostupné z: https://www.elektrina.cz/cena-elektriny-za-kwh-2020-cez-eon-pre-bohemia-centropol-a-dalsi.

[15] Ilkovič, D. *Fyzika*. 3. vyd. Praha 1: Státní nakladatelství technické literatury, 1961. ISBN 63-056-62.

[16] Inc., A. What If Water Heaters Were Smart? *Aquanta* [online]. 2021 [cit. 2021-4-14]. Dostupné z: https://aquanta.io/utilities/.

[17] InfluxData. Influxdb logo.svg. *Wikimedia commons* [online]. 2016 [cit. 2021-4-12]. Dostupné z: https://commons.wikimedia.org/wiki/File:Influxdb_logo.svg.

[18] InfluxData. Monitoring JVM Metrics using Grafana, Elasticsearch, Telegraf. *Kishara Buddika* [online]. 2020 [cit. 2021-4-12]. Dostupné z: https://medium.com/@KisharaBuddika/monitoring-jvm-metrics-using-grafana-elasticsearch-telegraf-a543b0bbdb8.

[19] InfluxData. influxdata. *InfluxDB* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://www.influxdata.com/products/influxdb/.

[20] InfluxData. influxdata. *Telegraf* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://www.influxdata.com/time-series-platform/telegraf/.

[21] Ing. Stanislav Števo, P. Energetika ohřevu vody. *Abs-portal* [online], 1. prosince 2017 [cit. 2021-4-1]. Dostupné z: https://www.asb-portal.cz/stavebnictvi/technicka-zarizeni-budov/energie/energetika-ohrevu-vody.

[22] Karel Laboutka, T. S. Tepelné vlastnosti vody při tlaku nasycení. *Tzbinfo* [online]. [cit. 2021-4-1]. ISSN 1801-4399. Literatura: RAŽNJEVIČ, K.: Termodynamické tabulky, Bratislava 1984. Dostupné z: https://www.tzb-info.cz/tabulky-a-vypocty/7-tepelne-vlastnosti-vody-pri-tlaku-nasyceni.

[23] Labs, G. Grafana logo. *PNG&SVG Download, Logo, Icons, Clipart* [online]. 2020 [cit. 2021-4-12]. Dostupné z: https://www.freelogovectors.net/grafana-logo/.

[24] Labs, G. Thousands love Grafana®, read why. *Grafana Labs* [online]. 2021 [cit. 2021-4-12]. Dostupné z: https://grafana.com/grafana/.

[25] laskaarduino. IoT ESP8266 Lua NodeMcu V3 WIFI modul. *Laskaarduino* [online]. [cit. 2021-4-14]. Dostupné z: https://www.laskarduino.cz/iot-esp8266-lua-nodemcu-v3-wifi-modul--tcp-ip/.

[26] Limited, T.-L. C. HS110. *Tp-Link* [online]. 2021 [cit. 2021-4-14]. Dostupné z: https://www.tp-link.com/cz/home-networking/smart-plug/hs110/.

[27] Miroslava Široká, E. S. a Karel Bartuška a Milan Bednařík a Oldřich Lepil a. *Přehled středoškolské fyziky*. 2. vyd. Praha 1: Prometheus, 1996. ISBN 80-7196-006-3.

[28] Mosquitto, E. Eclipse Mosquitto™ An open source MQTT broker. *Mosquitto* [online]. Prosinec 2020 [cit. 2021-4-12]. Dostupné z: https://mosquitto.org/.

[29] Nuwanthilaka, I. *Get into MQTT* [online]. 2018 [cit. 2020-12-18]. Dostupné z: https://medium.com/@isurunuwanthilaka/get-into-mqtt-in-2-minutes-python-dcker-5d4e8b55cf1c.

[30] Pal, A. a Prakash, P. *Practical Time Series Analysis*. 1. vyd. Birmingham: Packt Publishing Ltd., 2017. ISBN 978-1-78829-022-7.

[31] Pwll, J. *Raspberry-Pi Home Heating Controller* [online]. 2020 [cit. 2020-11-09]. Dostupné z: https://www.instructables.com/Raspberry-Pi-Home-Heating-Controller/.

[32] Sazima, M. *Sdílení tepla*. 1. vyd. Praha 1: Nakladatelství technické literatury, 1993. ISBN 80-85341-42-5.

[33] Schwarz, I. K. Teplo z domovního vodovodu pro tepelné čerpadlo - 1. část. *Topenářství instalace* [online], 14. června 2019 [cit. 2021-4-1]. Dostupné z: https://www.topin.cz/clanky/teplo-z-domovniho-vodovodu-pro-tepelne-cerpadlo-1-cast-detail-6691.

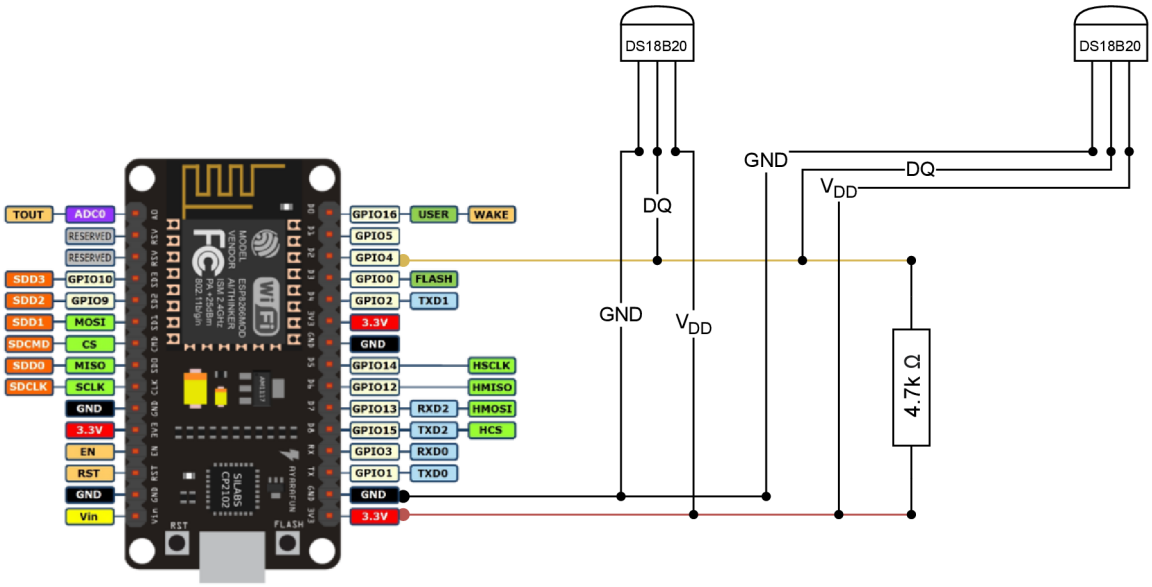
[34] TP Link. *User's Manual Wi-Fi Smart Plug with Energy Monitoring HS110*. REV 2.0.1. TP-Link Corporation Limited, 2017 [cit. 2021-4-14].

[35] Úřad Český statistický. Domácnosti podle používaných paliv a energií na ohřev vody. *Český statistický úřad* [online]. 2015 [cit. 2021-4-27]. Dostupné z: https://www.czso.cz/documents/10180/50619982/15018916020802.pdf/e8366302-e704-41cb-bbaa-781f315bf28a?version=1.0.

Příloha A

NodeMCU schéma zapojení

Následující obrázek představuje schematické zapojení dvojice senzorů DS18B20, pracujících na napětí 3.3V, k mikrokontroleru NodeMCU ESP8266 při využití jednoho GPIO pinu pro komunikaci, v kombinaci s $4.7k\Omega$ pull up rezistorem.



Obrázek A.1: **Schéma zapojení MCU.** Zapojení teplotních senzorů **DS18B20** k **NodeMCU ESP8266** použité v projektu.

Plakát

Obrázek A.2: Plakát k projektu.