Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh a realizace chytré domácnosti

Bc. Jan Hlubuček

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jan Hlubuček

Informatika

Název práce

Návrh a realizace chytré domácnosti

Název anglicky

Design and implementation of a smart home

Cíle práce

Cílem práce je návrh, vytvoření a konfigurace chytré domácnosti s využitím mikropočítače, mikrokontrolerů a dalších komponent. Dílčími cíli jsou pak analýza a srovnání možných řešení a technologií pro chytrou domácnost a vytvoření webového rozhraní, které umožní obsluhu jednotlivých funkcionalit domácnosti.

Metodika

V diplomové práci bude provedena analýza možných open-source i komerčních řešení pro chytré domácnosti. Na základě této analýzy bude navrženo konkrétní řešení, včetně konkrétních hardwarových komponent a vhodného software. Následně se bude práce věnovat implementaci tohoto návrhu ve fyzické domácnosti. Tedy propojením a konfigurací všech prvků chytré domácnosti. Pro usnadnění ovládání bude vytvořen webový portál, který umožní obsluhu jednotlivých prvků domácnosti.

Doporučený rozsah práce

60-80 stran

Klíčová slova

chytrá domácnost, mikrokontroler, mikropočítač, automatizace, Node-RED, smart assistant

---

Doporučené zdroje informací

BUCHANAN, Marlon. The Smart Home Manual: How to Automate Your Home to Keep Your Family Entertained, Comfortable, and Safe. HomeTechHacker, 2020. ISBN: 978-1735543000

KYAS, Othmar. How To Smart Home: A Step by Step Guide for Smart Homes & Building Automation. Key Concept Press, 2017. ISBN 978-3-944980-12-6.

PRŮCHA, Jan. Chytré bydlení: Inteligentní dům [online]. 2012 [cit. 2018-08-28]. Dostupné z: http://www.insighthome.eu/Chytre-bydleni/index.html.

UPTON, Eben a Gareth HALFACREE. Raspberry Pi: uživatelská příručka. 2., aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4819-8.

VALEŠ, Miroslav. Inteligentní dům. Brno: ERA, 2006. ISBN 80-7366-062-8.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 4. 11. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 31. 03. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Návrh a realizace chytré domácnosti" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2023 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce Ing. Marku Píckovi, Ph.D.

Návrh a realizace chytré domácnosti

Abstrakt

Tato práce se věnuje chytrým domácnostem založených na chytrých zařízeních a mikrokontrolerech a také automatizacím různých aspektů této chytré domácnosti. Může se jednat o úlohy jako je spínání světel, ovládání termostatu, nebo žaluzií a dalších.

V teoretické části se práce věnuje zkoumání technologií, služeb a hardware které lze použít pro tvorbu chytré domácnosti. Jsou popsány chytrá zařízení, které lze použít, existující systémy pro chytrou domácnost a protokoly které lze využít k propojení jednotlivých částí a zařízení domácnosti.

V praktické části se práce nejprve věnuje návrhu samotné domácnosti včetně požadavků, které by měla splňovat a následně návrhu webové aplikace kterou bude možné použít pro její ovládání. Řešení aplikace je postaveno na jazyce Java a frameworku Spring Boot. Chytrá domácnosti je potom založena na software Home Assistant. Pro tvorbu sensoru bylo použito ESP32. Na konci práce je výsledná implementace zhodnocena a je zkontrolováno že bylo dosáhnuto stanovených cílů a požadavků.

Klíčová slova: chytrá domácnost, IoT, Home Assistant, MQTT, ZigBee, Automatizace, Raspberry Pi, ESP32

Design and implementation of a smart home

Abstract

This work is dedicated to a smart home with smart devices and microcontrollers, as well as automation of various aspects of this smart home. It can be tasks such as switching on lights, controlling the thermostat or blinds and others.

In the theoretical part, the thesis examines technologies, services and hardware that can be used to create a smart home. Smart devices that can be used in such home are described as well as, existing smart home systems and protocols that can be used to connect individual parts and devices of such home.

In the practical part, the work first focuses on the design of the household itself, including the requirements it should meet, and then on the design of the web application that can be used to control it. The application solution is built with Java programing language and the Spring Boot framework. The smart home is then based on the Home Assistant software. An ESP32 was used to create the sensor. At the end of the work, the resulting implementation is evaluated and it is checked that the set goals and requirements have been achieved.

Key words: smart home, IoT, Home Assistant, MQTT, ZigBee, Automation, Raspberry Pi, ESP32

Obsah

1 Úvod

V posledních letech se stále více hovoří o konceptu chytré domácnosti a moderních technologiích, které jsou k jejímu vytvoření nezbytné. Tento trend ovlivňuje nejen průmyslové firmy, ale také běžné spotřebitele, kteří se snaží využívat moderní technologie ke zlepšení svého každodenního života.

Chytrá domácnost přináší mnoho výhod pro uživatele. Umožňuje centralizované ovládání různých zařízení v domácnosti a snadnou automatizaci různých procesů, což může přinést úsporu času a energie. Díky možnosti vzdáleného ovládání z mobilního zařízení lze chytrou domácnost ovládat odkudkoliv a kdykoliv. Chytré zařízení mohou také zlepšit bezpečnost domácnosti, například prostřednictvím chytrých kamer nebo detektorů kouře. Chytrá domácnost může také zlepšit energetickou účinnost a pomoci uživatelům snížit své náklady na energie.

Cílem této diplomové práce je popsat koncept chytré domácnosti a navrhnout konkrétní řešení, které bude implementováno v rámci reálné domácnosti. Součástí návrhu bude také vytvoření jednoduché webové aplikace, která umožní uživatelům ovládat jednotlivé prvky chytré domácnosti z jednoho místa.

V úvodu se práce bude věnovat přiblížení konceptu chytré domácnosti. V teoretické části pak popíše dostupná řešení, které lze k jejímu vytvoření použít. 5ešení budou zahrnovat jak již dostupné systémy, tak i dostupné technologie jako jsou komunikační protokoly tak i hardware či software, který lze v rámci domácnosti použít.

V rámci návrhu budou definovány nároky, které by mela výsledná chytrá domácnost splňovat. Následně pak bude se pak návrh bude zabývat propojením těchto dílčích částí do jednoho fungujícího systému který bude tyto požadavky splňovat. Součástí návrhu bude také návrh webové aplikace, kterou bude možné použít k ovládání některých prvků chytré domácnosti.

Po dokončení návrhu se se práce bude věnovat jeho implementaci do reálného světa a vytvoření chytré domácnosti.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je návrh, vytvoření, konfigurace a automatizace chytré domácnosti s využitím mikropočítače, mikrokontrolerů a dalších komponent. Dílčími cíli jsou pak

- analýza a srovnání možných řešení a technologií pro chytrou domácnost
- vytvoření webového rozhraní, které umožní obsluhu jednotlivých funkcionalit domácnosti.
- Přestavení možností automatizace a jejich implementace v rámci chytré domácnosti, které je předmětem této práce.

2.2 Metodika

V úvodní části práce bude provedena analýza možných technických řešení pro chytrou domáckost, a to jak z pohledu software či služeb které lze použít tak z hlediska protokolu pro komunikaci mezi zařízeními v rámci chytré domácnosti.

Tyto znalosti budou následně v další částí použity pro vytvoření návrhu chytré domácnosti. Pro tuto domácnosti bude zároveň navržen seznam požadavků, který by měl výsledný návrh splňovat. Spolu s návrhem domácnosti bude také navržena jednoduchá webová aplikace, které bude umožňovat ovládání jejích prvků.

V další fázi práce bude vytvořena zprovozněna a nakonfigurována reálná chytrá domácnost, podle návrhu v předchozí kapitole. Budou také vytvořeny jednotlivé automatizace, které umožňují a usnadňují používání chytré domácnosti. Spolu s chytrou domácnosti bude také podle návrhu vytvořena i webová aplikace pro její ovládání.

Poslední částí této práce bude závěr, který bude obsahovat shrnutí práce a bude zjištěno, jestli byli splněny všechny cíle této práce.

3 Pojmy

3.1 Domácnost

Pojem domácnost lze vymezit v oblasti právní a ekonomické.

Z právního hlediska do 31. prosince 2013 dle § 115 dřívějšího občanského zákoníku (Zákon č. 40/1964 Sb.) platilo, že domácnost tvoří fyzické osoby, které spolu trvale žijí a společně uhrazují náklady na své potřeby. Od 1. ledna 2014 roku 2014 tato definice už nemá oporu v právních předpisech [1]

Z ekonomického hlediska pak lze za domácnost považovat ekonomický subjekt který vstupuje na trh za účelem uspokojení svých potřeb. Na trhu produktu a služeb se jedná o skupinu kterou lze považovat za kupující nebo také spotřebitele naopak na trhu práce je tato skupina v roli poskytovatelů. Celý cyklus tedy funguje tak, že prodává své výrobní faktory firmám a za svou odměnu pořizuje jejich produkty a služby.[2]

3.2 Chytrá domácnost

V dnešní době zatím neexistuje přesná definice výrazu chytrá domácnost. Jsou proto používány různé termíny jako: smart home, chytrý dům, domácí automatizace, domotika, inteligentní elektroinstalace.

Pojem chytrá domácnost tak může být používána velmi volně a muže označovat například ukázkové domy projektované jako tzv. domy budoucnosti které jsou plné různých senzorů, a zařízení které spolu vzájemně komunikují. Stejně tak ovšem může tento pojem označovat dům, který je vybaven pouze bezpečnostními kamerami napojenými na počítačovou síť. [3]

Obecně lze říci že chytrá domácnost je taková domácnost, která obyvatelům v různých směrech usnadní fungovaní. Hlavní myšlenkou chytré domácnosti je vzájemná integrace jednotlivých zařízení v domácnosti. Krom usnadnění provádění různých činností nebo jejich automatizace muže chytrá domácnost přinášet úsporu z hlediska vytápění nebo ohřevu vody pomocí automatické regulace.[3]

3.3 Inteligentní budova

V předchozí kapitole je uvedeno několik definic chytré domácnosti. V poslední době se však také užívá termín inteligentní (nebo také chytrá budova). Proto je zapotřebí tyto pojmy rozlišit, ačkoliv se na první pohled může zdát, že se jedná o téměř identické pojmy. Chytrá domácnost se tedy vlastně skládá z nejrůznějších vybavení domácnosti, které spolu komunikují. Chytrá budova je budova, která je takto projektována již od začátku, cílem je pak nejčastěji snížení nákladu na její provoz a údržbu. [4]

4 Teoretická východiska

4.1 Komunikační protokoly

Jak již bylo řečeno chytrá domácnost se snaží usnadnit fungování svým obyvatelům pomocí automatizací. Komu aby bylo možné jednotlivá zařízení automatizovat je nutné jejich propojení do jednoho celku, aby automatizace mohly být používány napříč celím ekosystémem. Je tak nutné uvažovat o protokolech které budou v chytré domácnosti používány ke komunikaci mezi jednotlivými zařízeními.

4.1.1 Bluetooth

Bluetooth protokol je bezdrátový komunikační protokol, který umožňuje přenos dat mezi dvěma zařízeními v krátké vzdálenosti. Bluetooth využívá rádiové vlny a s kmitočtem 2,4 GHz. Byl vyvinut v roce 1994 společností Ericsson a jeho cílem bylo přinést bezdrátovou alternativou pro připojování periferních zařízení.[5]

Bluetooth je založen na tzv. Master-Slave architektuře. Jedno zařízení (Master) je zodpovědné za řízení komunikace a druhé zařízení (Slave) přijímá a odesílá data. Výhodou tohoto přístupu je, že lze přenášet data v obou směrech, aniž by došlo k narušení komunikace.[5], [6]

Bluetooth podporuje několik různých profilů, které umožňují přenos různých typů dat. Například profil Hands-Free umožňuje přenos hovorů mezi mobilním telefonem a sluchátky, zatímco profil File Transfer umožňuje přenos souborů mezi dvěma zařízeními.[6]

Bluetooth protokol se postupně vyvíjel a v současné době existuje několik jeho verzí. Bluetooth 1.0 a 1.0B byly první verze protokolu s pomalým přenosem dat a bezpečnostními nedostatky. Bluetooth 1.1 přinesl vylepšení v bezpečnosti a následovaly verze Bluetooth 2.0, 2.1, 3.0 a 4.0 s vylepšenou rychlostí přenosu dat, nižší spotřebou energie a podporou nových technologií jako NFC a BLE. Poslední verzí je Bluetooth 5.0 s ještě vyšší rychlostí přenosu dat, větším dosahem signálu, vylepšenou bezpečností a podporou mesh sítí. Každá verze Bluetooth protokolu přináší vylepšení oproti předchozí verzi a je kompatibilní se staršími verzemi.[7]

4.1.2 Wifi

Wi-Fi je sada bezdrátový sada protokol s označení IEEE802.11x pro přenos dat mezi zařízeními pomocí rádiových vln v bezlicenčních pásmech 2,4 GHz a 5 GHz. Wi-Fi je široce používaný pro bezdrátové připojení k internetu, a to zejména v domácnostech, kancelářích a veřejných prostorech.[8]

Wi-Fi protokol se skládá z několika vrstev, přičemž každá vrstva má své specifické funkce. Nejdůležitější vrstvou je fyzická vrstva, která definuje způsob, jakým se data přenášejí po bezdrátovém spojení. Další vrstvou je vrstva MAC (Media Access Control), která řídí přístup více zařízení k jednomu přístupovému bodu a zajišťuje bezpečnost komunikace. Poslední vrstvou je aplikační vrstva, která definuje způsob, jakým se aplikace připojují k internetu prostřednictvím Wi-Fi sítí.[9]

- Wi-Fi 802.11a: Tato verze byla uvedena v roce 1999 a pracuje v pásmech 5 GHz. Poskytuje rychlost přenosu dat až 54 Mb/s a má menší dosah signálu než následující verze.
- Wi-Fi 802.11b: Tato verze byla uvedena v roce 1999 a pracuje v pásmech 2,4 GHz. Poskytuje rychlost přenosu dat až 11 Mb/s a má delší dosah signálu než 802.11a.
- Wi-Fi 802.11g: Tato verze byla uvedena v roce 2003 a pracuje také v pásmech 2,4 GHz. Poskytuje rychlost přenosu dat až 54 Mb/s a má podobný dosah signálu jako 802.11b.
- Wi-Fi 802.11n: Tato verze byla uvedena v roce 2009 a pracuje v pásmech 2,4 GHz a 5 GHz. Poskytuje rychlost přenosu dat až 600 Mb/s a má větší dosah signálu než předchozí verze.
- Wi-Fi 802.11ac: Tato verze byla uvedena v roce 2013 a pracuje pouze v pásmech 5 GHz. Poskytuje rychlost přenosu dat až 1,3 Gb/s a má velmi malý dosah signálu, ale zároveň umožňuje vysokou rychlost přenosu dat.
- Wi-Fi 802.11ax: Tato verze byla uvedena v roce 2019 a pracuje v pásmech 2,4 GHz a 5 GHz. Poskytuje rychlost přenosu dat až 9,6 Gb/s a má větší dosah signálu než 802.11ac.

[10]

4.1.3 Zigbee

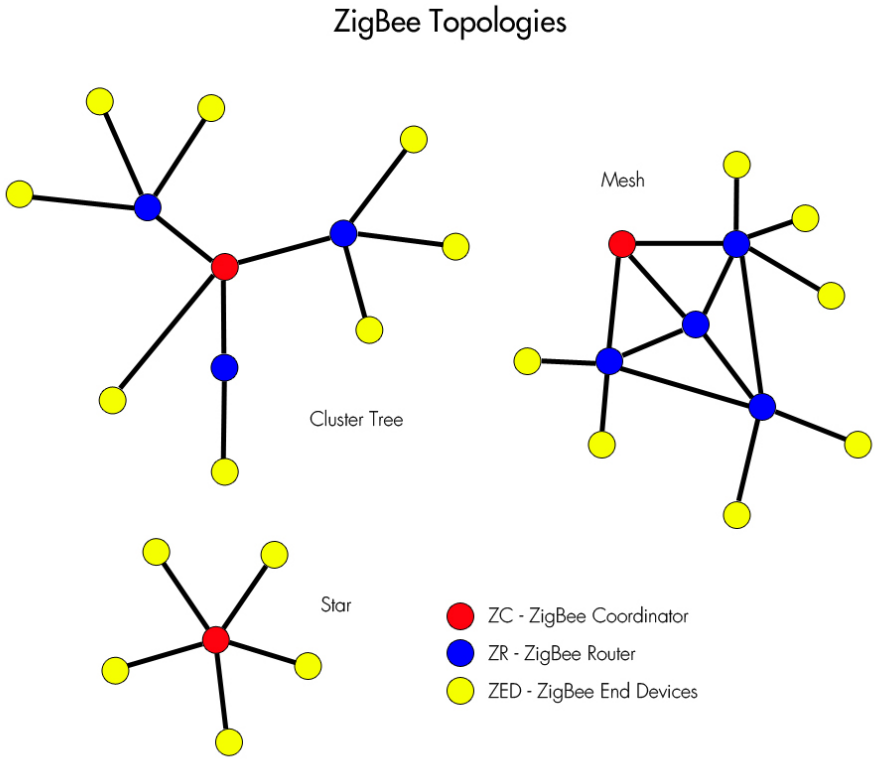
Zigbee začal vznikat na konci 90. let a jeho první verze byla vydána v roce 2005 století jako standard bezdrátové sítě pro zařízení s nízkou spotřebou, která vyžadují nízkou šířku pásma. Byl navržen tak, aby byl jednoduchý, levný a dobře se hodil pro použití v široké řadě aplikací, včetně senzorů, zařízení pro chytrou domácnost a dalších aplikací internetu věcí (IoT). Používá se k vytváření sítí typu WPAN (wireless private area network). Je založen na standardu IEEE 802.15.4 a pracuje ve frekvenčním pásmu 2,4 GHz, což je stejné pásmo, jaké používají jiné technologie, jako je Wi-Fi a Bluetooth.[11]

Zigbee byl vytvořen organizací Zigbee Alliance. Aliance byla založena v roce 2002 a je tvořena více než 400 členy společností z různých odvětví, mezi kterými se nachází například Amazon nebo Ikea. Hlavním cílem této organizace je podporovat rozšíření a používání standardu Zigbee v širokém spektru aplikací IoT zařízení a systémů. K dosažení tohoto cíle se aliance snaží zajistit, aby zařízení podporující Zigbee byla mezi sebou kompatibilní, což znamená, že mezi sebou mohou komunikovat bez větších problémů. Aliance také poskytuje nástroje, zdroje a podporu svým členům, aby jim pomohla vyvíjet a prodávat zařízení podporující Zigbee.[8], [11]

ZigBee sítě mohou být zapojeny ve třech různých topologiích a to strom, síť nebo hvězda. V každé síti se nachází právě jeden koordinátor, přičemž další zařízení mohou složit jako uzli v síti (v rámci Zigbee sítě zvaný router) nebo listy tedy koncová zařízení v rámci sítě. Nejčastěji používanou topologií je topologie mesh. V této topologii má každé zařízení schopnost komunikovat přímo s ostatními zařízeními v síti a může také fungovat jako opakovač, který přeposílá zprávy dalším zařízením a pomáhá tak rozšířit dosah sítě. To vytváří síť vzájemně propojených zařízení, což umožňuje více cest pro přenos dat

mezi zařízeními. Této topologie je, že umožňuje větší dosah a pokrytí, protože data lze předávat ze zařízení na zařízení, dokud nedosáhnou svého cíle. Umožňuje také větší spolehlivost, protože síť může nadále fungovat, i když jedno zařízení selže nebo je ze sítě odstraněno. Nevýhodou této topologie pak je, potenciálně větší spotřeba energie k jejímu provozu a může být složitější na nastavení a údržbu.

Jednou z hlavních výhod Zigbee je tak možnost pokrytí. není tak nutné, aby každé zařízení bylo v dosahu centrálního koordinátora. Stačí že jednotlivá zařízení budou navzájem v dosahu a díky vzájemnému propojení tak budou schopni s ním komunikovat.[11], [12]



Obrázek 1: ZigBee topologie [13]

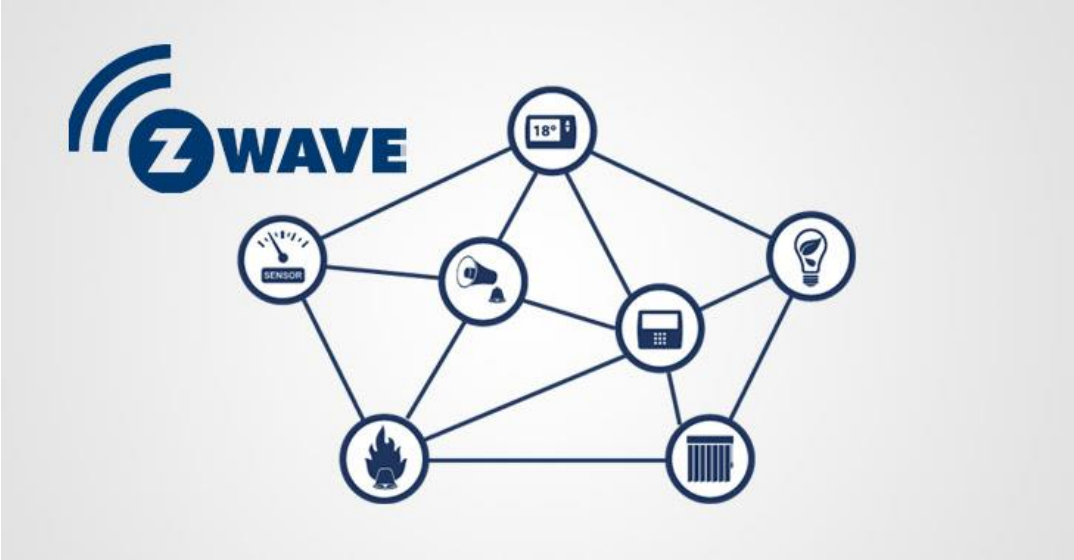
4.1.4 Z-wave

Z-Wave byl vyvinut společností Zensys, dánskou firmou založenou v roce 1999. První verze Z-Wave byla uvedena na trh v roce 2001 a od té doby prošla několika aktualizacemi a vylepšeními. Od roku 2007 byl Z-Wave vlastněn společností Sigma Designs, americkou firmou zabývající se vývojem integrovaných obvodů a bezdrátových technologií. V roce 2018 byla technologie odkoupena americkou společností Silicon Labs, která ji vlastní dodnes. [14], [15]

Jedná se o bezdrátový komunikační protokol, který se používá především v IoT a chytrých domácnostech. Pracuje v sub-gigahertzovém vích pásmech 868 MHz, 908 MHz nebo 2400 MHz. Díky tomu může přenášet data na relativně velkou vzdálenost a má také velmi dobrou prostupnost materiálem. Oproti protokolům, které využívají vyšší pásma o vyšších frekvencích přináší také nižší energetickou náročnost. Nevýhodou je pak velmi malá šířka pásma, přibližně 20 kbit/s. [14], [16]

Z-wave má obdobně jako Zigbee dva typy zařízení a to, řídící zařízení a jednotlivé uzli, které se k němu připojují. Řídicí zařízení zahájí komunikaci odesláním příkazů do sousedních uzlů ty pak předávají zprávy jiným uzlům nebo, pokud jsou zamýšlenými příjemci, odpovídají na přijaté příkazy a provádějí je. Řídicí zařízení mají úplnou směrovací tabulku sítě Z-Wave a jsou schopna komunikovat se všemi zařízeními v síti. Slave uzly nemohou nezávisle posílat přímé zprávy jiným uzlům, pokud nejsou nařízeny řídícími zařízeními. Pokud podřízený uzel obdrží příkaz, provede jej a poté odešle řídicímu zařízení odpověď s oznámením o úspěšném provedení příkazu. Pokud řídicí zařízení nepřijme potvrzovací zprávu, je rámec znovu vysílán s náhodným zpožděním, aby se zabránilo potenciální kolizi.[14], [17]

Hlavní rozdíl tedy oproti ZigBee tedy spočívá využívaní jiného pásma což muže pomoci zejména ve městech a zalidněných oblastech. Zde obvykle pásma 2,4 Ghz značně zaplněné, což může vést k nestabilitě nebo zpomalování sítě. Daní za tuto výhodu je ovšem nižší přenosová rychlost těchto pásem.[16]



Obrázek 2: Z-wave [18]

4.1.5 MQTT

MQTT (Message Queuing Telemetry Transport) je jednoduchý protokol pro zasílání zpráv. Protokol je navržen pro prostředí s nízkou šířkou pásma a vysokou latencí. Je tedy vhodný zejména pro použití v tzv. M2M (machine to machine) komunikaci a internetu věcí (IoT), kde je šířka pásma často omezená a síťová připojení mohou být nestabilní. Díky své jednoduchosti je také protokol vhodný pro použití v zařízeních s omezeným výkonem.[19]

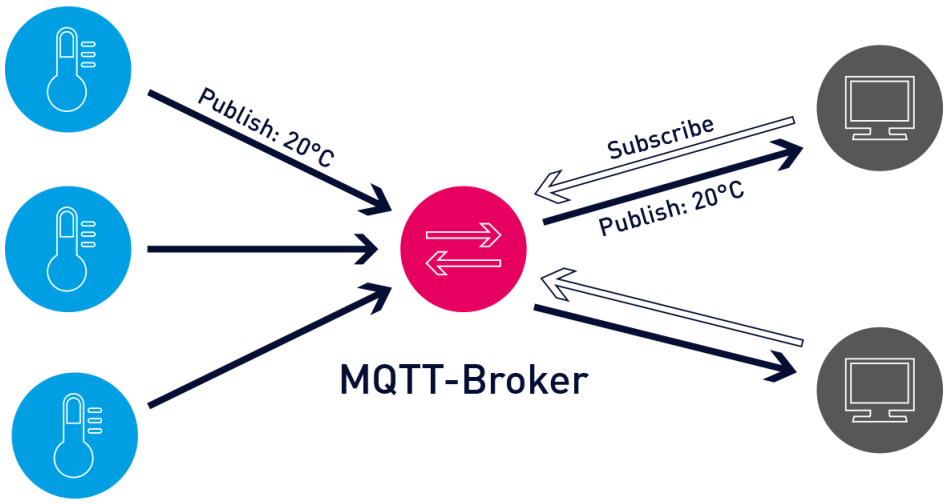
Protokol MQTT funguje přes TCP/IP protokol na principu publish/subscribe. To je princip, kdy odesílatelé (publishers) neposílají své zprávy přímo konkrétním příjemcům (subscribers). Místo toho jsou publikované zprávy přiřazeny k jednotlivý kanálům (topics), aniž by jejich odesílatelé věděli, jací konkrétně či pokud vůbec odběratelé existují. Podobně příjemci e přihlašují (subscribe) ke konkrétním tématům (opět bez jakýchkoliv znalostí o možných odesílatelích) a dostávají pouze zprávy, které náleží těmto tématům. Toto oddělení vydavatelů a předplatitelů škálovatelnost celé sítě[20], [21]

Protokol je založen na principu klient/server. Jelikož je tedy protokol navržen tak, že jednotlivý klienti (odesílatelé a příjemci) spolu vzájemně nekomunikují musí se v síti nacházet také tzv. broker. Jedná se o centrální server, který přijímá všechny zprávy od odesílatelů a poté zprávy směruje příjemcům na základě kanálu zprávy. Broker je zodpovědný za distribuci zpráv přihlášeným klientům a je základní součástí vzoru principu publish/subscribe v MQTT.[21]

Jednou z klíčových vlastností MQTT je také jeho podpora pro úrovně „kvality služeb“ (QoS), která klientům umožňují specifikovat úroveň záruky, že správy budou doručeny. Existují celkem tři úrovně QoS:

- QoS 0 (maximálně jednou): Zpráva je doručena maximálně jednou, bez záruky, že bude doručena.
- QoS 1 (alespoň jednou): Zpráva je doručena alespoň jednou, ale mohou se vyskytovat duplikáty (zpráva bude přijata vícekrát)
- QoS 2 (přesně jednou): Zpráva je doručena přesně jednou. Jedná se o nejvyšší úroveň záruky, ale je spojena s vyššími náklady.

[22]



Obrázek 3: MQTT broker [23]

Zprávy jsou v rámci MQTT publikování a přijímání prostřednictvím tzv. "témat" (topics). Topik v MQTT je název, pod kterým je publikována zpráva. Tyto názvy tvoří hierarchickou stromovou strukturu (obdobně jako například souborové systémy) a umožňují určit, které zařízení má přijímat specifickou zprávu. Zprávy mohou být filtrovány na základě tématu, což umožňuje efektivní distribuci informací mezi zařízeními v rámci sítě. Používání témat umožňuje zjednodušit komunikaci mezi zařízeními a usnadňuje správu informací v rámci sítě.[24]

Topiky v MQTT pak mohou vypadat následovně:

- "domácnost/kuchyně/teplota"
- "pokoje/ložnice/světlo"
- "venku/teplota"

V těchto příkladech, každý topic představuje specifickou informaci – teplotu v kuchyni, stav světla v ložnici a teplotu venku. Tyto informace mohou být publikovány a přijímány na základě tohoto tématu, což umožňuje zařízením komunikovat mezi sebou bez nutnosti přímého propojení.[24]

Při odebírání topiků lze používat speciální znaky. V MQTT, topik může obsahovat následující speciální znaky:

- / (lomítko) - slouží k oddělení jednotlivých částí tématu a tvoření stromové struktury
- + (plus) - používá se k označení jednoho nebo více podřízených témat, např. "domácnost/+/teplota" bude přijímat data pro všechna témata, které začínají na "domácnost/".
- # (hashtag) - používá se k označení všech témat, které začínají na daný prefix. Např. "domácnost/#" bude přijímat data pro všechna témata, které začínají na "domácnost/".

Je důležité si uvědomit, že tyto speciální znaky mají specifický význam a musí být použity správně, aby zajistily správnou funkčnost témat v rámci sítě MQTT.[24]

4.1.6 AMQP

AMQP (Advanced Message Queuing Protocol) je standardizovaný protokol pro zasílání zpráv s otevřeným zdrojovým kódem, který se široce používá pro zasílání zpráv mezi aplikacemi, službami a systémy. Poskytuje flexibilní, spolehlivý a efektivní způsob asynchronní komunikace pomocí front zpráv.[25]

Protokol umožňuje aplikacím odesílat a přijímat zprávy asynchronním způsobem, což znamená, že odesílatel a příjemce nemusí být současně připojeni do sítě. Díky tomu je užitečný pro vytváření distribuovaných systémů, které jsou odolné vůči poruchám a mohou se horizontálně škálovat.[26], [27]

AMQP poskytuje několik funkcionalit pro zajištění co nejspolehlivějšího doručování zpráv, včetně:

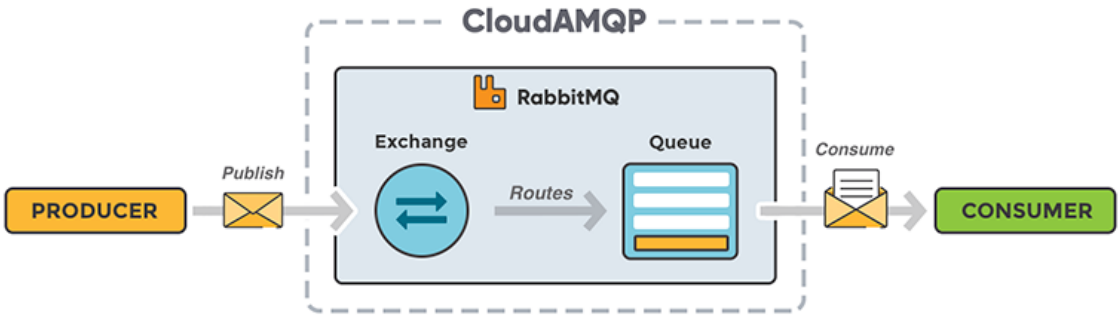
- Acknowledgement (potvrzení): Producenti si mohou vyžádat potvrzení od zprostředkovatele zpráv, když je zpráva doručena nebo odmítnuta.
- Transakce: Producenti mohou seskupit sérii zveřejněných zpráv do jediné transakce, která buď všechna uspěje, nebo všechna selžou.
- Persistency (trvanlivost): Zprávy mohou být uloženy v úložišti, takže se neztratí, pokud je zprostředkovatel není schopen přijmout.

[26], [27]

V AMQP jsou zprávy odesílány a přijímány z front zpráv, což jsou datové struktury obsahující seznam zpráv, které čekají na zpracování. Fronty zpráv jsou umístěny na zprostředkovateli zpráv, což je software, který spravuje fronty zpráv a směruje zprávy

mezi producenty (aplikace, které odesílají zprávy) a spotřebitele (aplikace, které přijímají zprávy).[27]

AMQP poskytuje řadu funkcí, díky kterým je užitečný pro zasílání zpráv na podnikové úrovni. Má podporu pro spolehlivé doručování zpráv, potvrzování zpráv, vypršení platnosti zprávy a stanovení priority zpráv. Má také podporu pro transakce, které umožňují odesílání a přijímání více zpráv atomickým způsobem, což zajišťuje, že buď budou doručeny všechny zprávy, nebo žádná.[27]



Obrázek 4: AMQP protokol [28]

4.1.7 HTTP

HTTP (HyperText Transfer Protocol) je protokol sloužící pro přenos dat mezi počítači. V rámci referenčního modelu ISO/OSI se pohybuje na sedmé tedy aplikační vrstvě. Protokol byl původně navržen Timem Bernersem-Lee v CERNu a ze značné čísti tím umožniv vznik dnešního internetu někdy nazývaného také www (World Wide Web). Dodnes je využíván k přístupu na většinu webových stránek. Protokol sám o sobě není nijak zabezpečený, z toho důvody byl vytvořen protokol HTTPS (HyperText Transfer Protocol Secure). Ten spojuje Protokol HTTP s protokolem TSL, který HTTP komunikaci šifruje pomocí asymetrická kryptografie.[29], [30]

Původní verze pojmenovaná 0.9 popisuje způsob výměny dat založeném na principu klient-server. Tato verze je dnes již zastaralá. Nejpoužívanější verzí je nyní HTTP/1.1.[30]

Protokol je navržen pro architekturu typ klient-server. Způsob komunikace tedy probíhá následovně. Komunikace musí být zahájena klientem, není možné, aby server navázal spojení s klientem. Po navázání spojení je komunikace založena na principu request-response (žádost-odpověď). Klient tedy odešle požadavek. Ta na serveru zpracována a následně je klientovy odeslána odpověď. Data tedy proudí oběma směry, protokol je tedy možné využít jak k vyžádání nějakých dat ze serveru, tak k zápisu nových dat na server (nebo jejich úpravě).[29], [31]

HTTP dotaz se skládá ze tří částí. První část požadavku, se skládá z metody požadavku, cesty požadavku a verze HTTP[31]

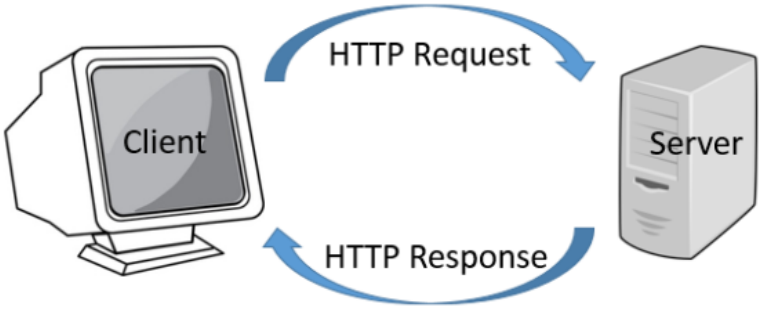
Druhou částí je hlavička. Ta obsahuje další informace o požadavku. Hlavičky požadavků mohou být použity k specifikaci formátu těla požadavku, k ověření klienta nebo například ke specifikaci preferovaného jazyka nebo kódování.[31]

Poslední částí je tělo požadavku. To obsahuje samotná data posílaná na server. Není nutné, aby požadavek tělo obsahoval. Některé typy dotazu dokonce ani neumožňují tělo k dotazu přidat[31]

Mezi záhladní typy http dotazů patří:

- GET – slouží k získávání dokumentů nebo dat. Používá se například k načítání webových stránek
- POST – slouží k zasílání nových dat na server. Používá se například po odeslání dat z webového formuláře
- PUT – slouží k úpravě již existujících dat nebo objektů na serveru
- DELETE – slouží k mazání již existujících dat nebo objektů na serveru

[32]



Obrázek 5: HTTP protokol [33]

4.1.8 COAP

CoAP je protokol založený na UDP (User Datagram Protocol). Specializuje se na především na sítě s malou šířkou pásma. Takové sítě nejsou vždy velmi spolehlivé a obecně mají nízkou datovou propustnost. Zařízení využívající tyto sítě mají obvykle velmi omezený výpočetní výkon. Protokol CoAP řeší tento problém tím, že umožní odesílání paketů přes UDP, který je jednodušší. Proto redukuje celkovou velikost paketů a zároveň snižuje fragmentaci5. Pro jednoduchost a snadnou interoperabilitu s Hypertext Transfer Protocol (HTTP) protokol CoAP podporuje REST architekturu (Representational State Transfer). Mezi další funkce patří podpora různých typů obsahu (Content-type), URI (Uniform Resource Identifier) a nízká složitost parsování paketů.[34], [35]

Zprávy používané v CoAP jsou rozděleny do několika typů:

- confirmable: Zpráva, která vyžaduje potvrzení od příjemce. Každá taková potvrzená zpráva má za následek samostatnou zprávu o potvrzení nebo resetování.
- non-confirmable: Zpráva, která nevyžaduje potvrzení při příjezdu. Používá se pro zprávy, které jsou přenášeny, ale nemusí být zaručeno jejich správné doručení.
- acknowledgement: Zpráva pro potvrzení, že dorazila konkrétní potvrzovatelná zpráva.
- reset: Typ zprávy, kterou lze odeslat pro resetování přenosu zprávy, pokud přijatou potvrzenou nebo nepotvrzenou zprávu nelze interpretovat nebo byly zapomenuty některé požadované informace, například pokud příjemce restartoval.

Prázdné zprávy (viz prázdná odpověď níže) lze použít k vyvolání resetovací zprávy ke kontrole online stavu koncového bodu (CoAP ping).

- piggyback: Odpověď obsažená v potvrzovací zprávě se nazývá zpětná odpověď.
- separate: Pokud je místo zpětné odpovědi potvrzovatelná zpráva potvrzena potvrzovací zprávou, odpověď odeslaná později, například proto, že odpověď ještě nebyla připravena na serveru, může být odeslána jako samostatná odpověď.
- empty: Zprávy odeslané bez obsahu a kódu 0,00 nejsou ani požadavkem, ani odpovědí. Obsahuje pouze 4bajtové záhlaví a lze jej použít k provedení pingu CoAP.

[35]

4.2 Koncová zařízení v chytré domácnosti

4.2.1 Osvětlení

Chytré osvětlení je obvyklou součástí chytrých domácností a umožňuje dálkové ovládání všech chytrých světel prostřednictvím aplikace nebo chytrého zařízení jako jsou Amazon Echo nebo Google Home. Osvětlení pak muže být ovládáno pomocí automatizací, například na základě času nebo dat ze sensorů (například sensor pohybu). Kromě klasické spínání obvykle tato zařízení umožňují měnit svou intenzitu nebo barvu podle preference uživatele. Zařízení také může být synchronizováno s dalšími zařízeními v domácnosti. Pomocí automatizací osvětlení v chytré domácnosti lze minimalizovat dobu svícení v domácnosti. Chytré osvětlení tak umožňuje úspory energie a výdajů za elektřinu. [36]

Obrázek 6: HUE lights [37]

4.2.2 Termostaty

Inteligentní termostaty jsou zařízení, která slouží ke řízení teploty v místnosti. Tyto termostaty mohou být ovládány pomocí mobilního telefonu nebo hlasových asistentů. Umějí si ukládat teplotní profily a na základě nich automaticky regulovat teplotu v místnosti. Inteligentní termostaty také mohou být nastaveny tak, aby se automaticky přizpůsobovaly vašemu životnímu stylu a plánům. Tyto termostaty mohou také analyzovat data a použít umělou inteligenci k vylepšení hospodaření s energií. Inteligentní termostaty také umožňují uživatelům nastavit teplotu v místnosti na dálku a sledovat spotřebu energie. Tyto zařízení mohou pomoci ušetřit peníze za energie a zlepšit komfort v domácnosti.[38]

Obrázek 7: chytrý termostat [38]

4.2.3 Čidla a sensory

Čidla a senzory v chytré domácnosti jsou zařízení, která slouží k měření a monitorování různých veličin v domácnosti. Tyto čidla a senzory mohou měřit různé hodnoty, jako je teplota, vlhkost, světlo, pohyb, tlak, hluk a další. Tyto zařízení mohou být integrovány s dalšími zařízeními v chytré domácnosti, jako jsou inteligentní termostaty, světla a bezpečnostní kamery, a mohou tak pomoci automatizovat a zlepšit funkce v domácnosti.

Existují také čidla jejich účelem je zvýšení bezpečnosti v domácnosti, například sensor vody pro detekci vytopení, který lze umístit například v blízkosti pračky nebo myčky. V případě že v domácnosti začne nečekaně vytékat vada (například vyteče pračka, nebo praskne vodovodní trubka). Sensor detekuje přítomnost vody a muže spustit automatizaci, která například odešle varovnou správu, ale může například také vypnou elektřinu v domě, aby se předešlo dalším škodám. Obdobně lze využít také například chytré detektory kouře nebo CO2[39]

Obrázek 8: senzor úniku vody [40]

4.2.4 Bezpečnostní zařízení v chytré domácnosti

Bezpečnostní zařízení v chytré domácnosti jsou určena k ochraně vašeho domova. Tyto zařízení lze ovládat a sledovat pomocí aplikace. Mezi bezpečnostní zařízení patří chytré zámky, bezpečnostní kamery, detektory pohybu a snímače dveří/okenních rámů. Tyto zařízení mohou detekovat nežádoucí pohyb a okamžitě vás upozornit prostřednictvím zpráv nebo hlasových příkazů. Díky bezpečnostním zařízením má uživatel kontrolu nad tím, co se děje ve vašem domově, i když tam není. Tyto zařízení mohou být nastavena tak, aby spolupracovala s jinými inteligentními zařízeními, jako jsou světla nebo termostaty. Bezpečnostní zařízení v chytré domácnosti vám přinášejí větší klid a jistotu, že domácnosti je chráněna. Tyto zařízení mohou být také propojena s profesionálními službami bezpečnosti, které zajistí rychlou pomoc v případě nouze.[41]

Obrázek 9: chytrý zámek [41]

4.2.5 Reproduktory a hlasový asistenti

Chytré reproduktory a asistenti jsou zařízení, která využívají umělé inteligence a technologie rozpoznávání hlasu k výkonu úkolů a poskytování informací prostřednictvím hlasových příkazů. Mohou být integrovány do systémů chytrých domácností, což vám umožňuje ovládat různé chytré zařízení a spotřebiče ve vašem domově, jako jsou světla, termostaty, bezpečnostní systémy a další, pouze prostřednictvím hlasu. Příklady populárních chytrých reproduktorů a asistentů zahrnují Amazon Echo s Alexa, Google Home s Google Assistant a Apple HomePod s Siri.[42]

Obrázek 10: hlasový asistent [43]

4.2.6 Chytré spotřebiče

Chytré spotřebiče jsou zařízení, která lze propojit s chytrým systémem domácnosti. Tyto spotřebiče poskytují pohodlí a kontrolu nad vašimi domácími činnostmi. Patří sem například chytré ledničky, pračky, sušičky nebo kávovary. Tyto spotřebiče mohou být obvykle ovládány pomocí aplikace nebo hlasových asistentů. Tyto zařízení mohou být také nastavena tak, aby spolupracovala s jinými inteligentními zařízeními v domácnosti v rámci automatizace. Chytré spotřebiče vám umožňují mít větší kontrolu nad domácími činnostmi a poskytují větší pohodlí v domácnosti.[44]

4.3 Existující systémy pro chytrou domácnost

Jedním z cílů chytré domácnosti je propojení všech zařízení do jednoho celku, tak aby její automatizace mohly probíhat napříč všemi zařízeními. Toto propojení obvykle probíhá pomocí HUBu. To je obvykle hardwarové zařízení, které funguje jako centrální bod domácnosti, ke kterému se připojují koncová zařízení jako jsou světla vypínače termostaty a podobné.

Problémem velké části těchto koncových zařízení pro chytrou domácnost je, že jsou často určeny k použití v rámci vlastního ekosystému daného výrobce. Například Smart Home od Ikea neumožnuje přidání zařízení od jiného výrobce. HUE od Philips umožnuje

přidání zařízení jiných výrobců, ale má omezené možnosti nastavení, které jsou zaměřené primárně na osvětlení, na které je HUE ekosystém zaměřený.[45]

Pokud se rozhodneme používat zařízení od různých výrobců či vlastnoručně vyrobená zařízení, například pomocí mikrokontrolerů. Je vhodné vybrat takový HUB, který umožní párovaní veškerých zařízení, či protokolů které plánujeme v domácnosti používat.

4.3.1 OpenHAB

OpenHAB je open-source softwarová platforma pro automatizaci domácnosti postavená na jazyce JAVA. Jejím cílem je umožnit uživatelům ovládání a monitoring jejich chytré domácnosti. Umožnuje propojení s řadou chytrých zařízení a protokolů. Tím umožňuje uživatelům ovládání veškerých chytrých zařízení v domácnosti (se kterými je propojen) prostřednictvím webového rozhraní nebo mobilní aplikace. Platforma je založena na modulární architektuře a umožňuje uživatelům rozšířit její možnosti pomocí pluginů, skriptů a pravidel. [46]

OpenHAB je založen na modulární architektuře. To umožnuje uživatelům rozšiřovat platformu pomocí různých zásuvných modulů a skriptů. Díky toto modularitě je možné OpenHAB doplnit o pluginy určené k integraci systémů a služeb třetích stran, jako jsou hlasoví asistenti, zařízení pro chytrou domácnost. Skripty se používají k automatizaci určitých úkolů, typicky se jedná o spouštění akcí při splnění určitých podmínek. Pravidla se používají k definování podmínek a akcí, k jejichž provedení dojde při splnění těchto podmínek.[46], [47]

Automatizace v rámci OpenHAB je založena na tzv plavidlovém engine (rule engine). Ten umožnuje snadné vytváření automatizací, pomocí definování pravidel. Jejich spouštěčem mohou být konkrétní události, specifický čas nebo například data ze senzorů. Při splnění těchto podmínek potom mohou spouštět různé akce, jako je odesílání oznámení, změna stavu (zapnutí/vypnutí) zařízení nebo odesílání specifický příkazů konkrétním zařízením. [48]

Pro složitější automatizace OpenHAB podporuje vytváření automatizačních scriptů. Pro tvorbu těchto skriptů je podporováno více jazyků. Groovy, JavaScript, Jruby a Jpython. [48]

OpenHab lze instalovat buďto jako samostatnou aplikací na dané zařízení, alternativně jak pak takém možní jej provozovat v docker kontejneru.[49]

4.3.2 Home Assistant

Home Assistant je open-source softwarová platforma pro domácí automatizaci, která umožňuje uživatelům ovládat a automatizovat různé zařízení a systémy ve svém domě, jako jsou inteligentní žárovky, kamery, termostaty atd. [50]

Home Assistant podporuje širokou škálu protokolů a API, včetně Zigbee, Z-Wave, Bluetooth, MQTT, HTTP a dalších, což umožňuje snadnou integraci s velkou škálou zařízení. Home Assistant také podporuje automatickou detekci zařízení, která umožňuje snadnou instalaci nových zařízení bez nutnosti manuální konfigurace.[51]

Umožnuje také implementaci nejrůznějších automatizací napříč všemi zařízeními či služby miktere jsou s Home Assistem propojeny. Uživatelé mohou buďto vytvářet vlastní automatizační scénáře pasných v jazyce YAML, nebo pomocí integrovaného editoru tvořit jednu. Jednoduché automatizace bez nutnosti psaní kódu. Tyto scénáře mohou reagovat na různé události, data ze sensorů nebo například čas a spouštět různé akce, například ovládání světel, změnu teploty nebo provolání nějaké služby.

Home Assistant také nabízí rozsáhlé možnosti zobrazování dat a informací, včetně grafů a tabulek. Uživatelé tak mohou snadno vytvářet vlastní dashboardy pro sledování a analýzu dat ze senzorů a jiných zařízení. Tyto dashboardy mohou být také použity k zobrazení stavu různých zařízení a automatizačních scénářů v reálném čase. V poslední době byl Home Assistant rozšířen i možnost integrace s virtuálními asistenty jako Google Assistant a Amazon Alexa.[51], [52]

Pro Home Assistant existuje několik základních druhu instalací:

Home Assistant Core: Tento způsob instalace představuje provoz Home Assistan v prostředí již nainstalované OS, který obsahuje veškeré požadované knihovny. Jedna se a nativní instalaci aplikace Home Assistant. Výhodou tohoto řešení je, že zařízení, na kterém bude Home Assistant nainstalován není nutné dedikovat pouze provozu této aplikace ale lze jej využívat k provazu dalších aplikací. Zároveň je ovšem nutné pamatovat na to že uživatel musí sám aplikaci udržovat, například je nutné se starat o průběžnou instalaci updatů.[53]

Home Assistant Container: Tento způsob instalace umožňuje nainstalovat Home Assistant jako Docker kontejner. Jedná se o obdobný způsob instalace jako v předchozím případě. To umožňuje snadnou správu a aktualizaci aplikace v rámci kontejneru. Výhodou také je že aplikace je provozována v izolované kontejneru, není tak nutné řešit problémy jako nekompatibilní požadované verze knihoven více různých SW.[53]

Home Assistant Supervised: Tento způsob instalace zahrnuje Home Assistant Core (který je ovšem provozován v docker kontejneru) spolu se supervizor. Supervizor je program, který se stará o správu Home Assistant, například jej updatuje nebo umožňuje instalovat doplňky. Instalace doplňků probíhá tak, že supervizor naistaluje doplněk do samostatného docker kontejneru na OS, kde běží a ve webovém prostředí Home Assistant pak mužní jeho správu a konfiguraci. Jedná se o nejsložitější způsob instalace, který je doporučen pouze pro uživatele s dobrou znalostí OS Linux, Docker, a počítačových sítí.[53]

Home Assistant OS: Tento způsob instalace zahrnuje celý operační systém (OS) včetně Home Assistant. Jedno se o uživatelsky nejjednodušší způsob instalace, uživatel totiž po instalici OS nemusí instalovat žádné další programy. Tento způsob instalace je vhodný pro uživatele, kteří chtějí snadnou a rychlou instalaci s minimálním nastavením. Hlavní nevýhodou pak je že Home Assistant OS není určen pro instalaci jakýchkoliv dalších programu mimo Home Assistant. Zařízení, které bude k provozu použito tak již nelze využívat za žádným jiným účelem.[53]

	OS	Container	Core	Supervised
Automations	✓	✓	✓	✓
Dashboards	✓	✓	✓	✓
Integrations	✓	✓	✓	✓
Blueprints	✓	✓	✓	✓
Uses container	✓	✓	✗	✓
Supervisor	✓	✗	✗	✓
Add-ons	✓	✗	✗	✓
Backups	✓	✓[1]	✓[1]	✓
Managed OS	✓	✗	✗	✗

Obrázek 11: Home Assistant - druhy instalace [53]

4.3.3 SmartThings

SmartThings je platforma pro chytrou domácnost vyvinutá společností Samsung, nejedná se tedy již o open-source řešení ale o řešení komerční. Tato platforma umožňuje majitelům domů ovládat a automatizovat svá připojená zařízení z jednoho centrálního místa. Platforma je navržena tak, aby byla vysoce škálovatelná. Ovládání pak probíhá pomocí aplikace, kterou lze nainstalovat do mobilu či tabletu nebo pomocí webového rozhraní. [54]

SmartThings je cloudově založený. Ačkoliv jsou tedy jednotlivá zařízení spárovaná s fyzickým HUBem který se nachází v domácnosti. Samotná data z chytrých zařízení jsou tedy ukládána a zpracovávána na vzdálených serverech. To umožňuje uživatelům ovládat své zařízení z libovonného místa pouze pomocí internetu. Také tím odpadá nutnost časté aktualizace HUBů v domácnostech, jelikož klíčové procesy se odehrávají na serverech Samsugu aktualizace tak systému tak obstarává Samsung a pro koncové uživatele tato starost odpadá. Nevýhodou tohoto řešení je závislost na připojení k internetu, bez kterého automatizace přestávají fungovat. Také to do celého systému přináší časové prodlevy, protože příkazy musí putovat mezi cloudem a domácností místo toho, aby byly vytvářeny lokálně. Tyto prodlevy se zvyšují s klesající kvalitou připojení domácnosti k internetu. [54], [55]

SmartThings podporuje řadu protokolů pro propojení a komunikaci s inteligentními zařízeními, včetně: Zigbee, Z-Wave, WiFi, MQTT, AMQP.V rámci sítových protokolů pak SmartThings podporu automatické objevení a spárování podporovaných chytrých zařízení. Některá zařízení, které lze připojit k internetu podporují přímé napojení na

SmartThings cloud, čímž odpadá nutnost lokálního HUBu (typicky se jedná například o televize značky Samsung). Pro zařízení používající lokální protokoly jako ZigBee, Z-wave, MQTT a podobné je nutné pořídit HUB kompatibilní se SmartThings. Cena tohoto HUBu na v době psaní práce byla 2712 Kč. [56]

Obdobně jako u předchozích platforem lze pak v rámci SmartThings nastavovat automatizace. J zde možnost nastavit jednodušší automatizace jako kombinace podmínek při jejichž splnění budou provedeny předem definované akce. Složitější automatizace je pak možné realizovat pomocí skriptů psaných v jazyce Groovy.[55]

4.3.4 Fibaro

Fibaro je komerční domácí automatizační systém od společnosti Fibaro. Ten umožňuje uživatelům ovládat a monitorovat všechny elektronické zařízení v jejich domácnosti pomocí jednoho centrálního bodu. Jedná se o Hardwarové řešení, to znamená že obsahuje konkrétní (hardwarovou) centrální jednotku, kterou je nutné instalovat do domácnosti. Fibaro poskytuje vlastní koncová zařízení, jako jsou světla, termostaty či zámky. Není ale striktně kompatibilní pouze se zařízeními stejného výrobce ale umožnuje připojení koncových zařízení jiných výrobců, které používají standardizované protokoly jako ZigBee, Z-wave, MQTT.[57], [58]

Výhodou Fibaro je, že se jedná o komerční řešení. Za stabilitu celého systému tak ručí výrobce a zařízení tak obvykle funguje bez větších problému, oproti tomu open-source nemusejí vždy být vždy plně stabilní, jelikož nikdo neručí za jejich funkčnost. S tím se váže a hlavní nevýhoda celého systému, a to jsou počáteční náklady. Jelikož se jedná o Hardwarové řešení je nutnou zakoupit centrální jednotku. Její cena v době psaní této práce je 13 090 Kč.[58], [59]

Fibaro podporuje širokou škálu protokolů, včetně Zigbee, Z-Wave, Bluetooth, MQTT, HTTP a dalších, což umožňuje snadnou integraci s velkou škálou zařízení. Fibaro také podporuje funkci automatického párování zařízení. Tato funkce umožňuje snadné a rychlé párování nových zařízení s centrálním panelem pomocí komunikačních protokolů. Stačí jednoduše připojit nové zařízení do sítě a centrální jednotka zařízení automaticky rozpozná a párování provede automaticky.[57]

Obdobně jako předchozí platformy i Fibaro podporuje aromatizaci pomocí jednouchých pravidel na principu spouštěcích podmínek a prováděných akcí. Pro složitější automatizace lze opět použít skripty. Pro ty se v ekosystému ekosystému Figaro používá jazyk Lua [57]

4.3.5 Srovnání

Všechny čtyři zmíněné platformy nabízejí stejnou konektivitu lze je párovat se záři ženěním pomocí protokolů ZigBee, Z-wave. Lze je také napojit na MQTT nebo AMQP brokery, síťovou konektivitu pak lze obstarat pomocí Wifi nebo síťového kabelu.
Z hlediska konektivity tak nelze upřednostnit konkrétní řešení.

Obdobně z hlediska nastavení automatizací není patrné velké množství rozdílů, všechny platformy umožnují jednoduché nastavení automatizace prostřednícím zadání spouštěcích podmínek, při jejichž splnění se spustí definované akce. Pro pokročilé automatizace pak

všechny platformy obsahují možnost tvorby skriptů, přičemž hlavním rozdílem je použitý jazyk. OpenHAB podporuje jazyky Groovy, JavaScript, Jruby a Jpython. Smart Assistant podporuje skrypty psané v jazyce YAML. SmartThings podporuje jazyk Groovy. Fibaro automatizační skrypty jsou psané v jazyce Lua.

Hlavním rozdílem pak je, jestli se jedná o softwarové, hardwarové nebo cloudové řešení. Hlavní výhodou je přenesení veškeré logiky na vzdálené servery (nebo také tzv. do cloudu). Veškerá obsluha systému pak není závislá na připojení uživatele do domácí sítě. Stačí pouze připojení k internetu. Zároveň tím odpadá i starost o upravování případného software, protože i komu dochází v rámci cloudu. Z toho plynoucí nevýhody jsou pak nutnost připojení domácnosti k internetu, bez kterého domácnost přestává fungovat. Komunikace mezi domácností a internetem také přináší prodlevu mezi zadáním příkazu, či splněním nějakých podmínek k automatizaci a provedením očekávaných činností. Teoreticky by bylo možné provozovat Cloudovou platformu bez lokálního hubu, pokud by všechny zařízení v rámci domácnosti byla schopna komunikovat přímo s cloudem. Prakticky je ovšem nutné mít lokální hub pro připojení zařízení s protokoly jako ZigBee nebo Z-Wave. Tento hub se následně propojí s cloudem čímž zprostředkovává jeho propojení s těmito zařízeními.

Oproti tomu Hardwarové řešení funguje čistě v rámci domácí sítě. Domácnost tedy ke svému fungování nevyžaduje připojení k internetu. Komunikace a přenést informací Probíhá pouze v rámci lokálních sítí domácnosti, provádění příkazů je tedy výrazně rychlejší, jelikož příkazy nikdy nemusí opustit lokální síť. S tím již zmíněná nevýhoda je nutnost lokálního HUBu který je obvykle dobře pravidelně aktualizovat, zároveň pokud to to daná platforma neumožňuje, není vždy možné ovládat domácnost na dálku, bez přístupu do lokální sítě domácnosti.

Softwarové řešení je pak velmi podobné hardwarovému. Teoreticky by bylo možné provozovat tento software v cloudu. To se ovšem v praxi prakticky nevyužívá, mimo jiné z důvodu problematického připojení zařízení komunikující pomocí lokálních protokolu jako ZigBee nebo Z-Wave. Reálně je tak software nainstalovaný na server, který běží uvnitř lokální sítě domácnosti čímž vzniká řešení velmi podobné Hardwarovému.

Dalším podstatným rozdílem pak může být, jestli se jedná o open source nebo komerční řešení. Hlavní výhodou komerční řešení je jejich stabilita. Za komerčním řešením zpravidla stojí společnost, která jej vyvíjí a zodpovídá za fungování systému. Zároveň je ale nutno vzít v podraz že se jedná zpoplatněný produkt což sebou oproti open-source řešení nese větší pořizovací náklady. V případě open-source platformám nikdo takový neexistuje. To vede k tomu, že ne vždy je taková platforma stabilní a muže docházet k problémům kdy nové aktualizace přináší bugy nebo přímo rozbijí některé stávající funkcionality.

4.4 Docker

Docker je platforma pro kontejnerizaci aplikací, která umožňuje vytvářet izolované kontejnery. Ty obsahují všechny potřebné závislosti, kód a konfiguraci pro spuštění aplikace. Lze je také snadno přenášet mezi různými prostředími, což usnadňuje nasazení aplikací a zajištění konzistentního provozu bez nutnosti instalace závislostí nebo konfigurace.

Docker umožňuje jednoduché a efektivní řízení těchto kontejnerů. Díky tomu je Docker ideální pro vertikální škálování, kdy stačí spustit více kontejneru s požadovanou aplikací nebo službou. Díku tomu se hodí pro provozování cloudových aplikací. Docker poskytuje také řadu nástrojů pro řízení kontejnerů, například Docker Compose pro správu více kontejnerů najednou a Docker Swarm pro správu velkých clusterů.[49]

Kontejner je izolované běhové prostředí, které umožňuje spustit aplikaci s vlastními závislostmi, kódem a konfigurací. Jedná se o způsob virtualizace, který je odlišný od tradiční virtualizace pomocí virtuálních strojů, protože kontejnery jsou mnohem lehčí a efektivnější.[60]

Kontejner obsahuje vše, co je potřebné pro spuštění aplikace, včetně operačního systému, běhových knihoven a aplikace samotné. Kontejnery také nabízejí řadu výhod, jako je například izolace mezi aplikacemi, konzistentní prostředí, snadná správa a škálovatelnost.

Jedna z největších výhod kontejnerů je jejich efektivita. Kontejnery používají sdílené jádro operačního systému hostitele a izolují pouze procesy aplikace, což znamená, že jsou mnohem lehčí než virtuální stroje, které musí mít vlastní operační systém. Kontejnery také nabízejí rychlejší spuštění a snadnou správu, protože lze kontejnery rychle vytvářet, spravovat a přenášet mezi různými prostředími.[60]

Kontejnery nabízejí řadu výhod, jako je například:

- Konzistentní prostředí: Kontejnery zajistí, že aplikace běží v konzistentním prostředí bez nutnosti instalace závislostí na konkrétním systému.
- Škálovatelnost: Díky snadnému vytváření a správě kontejnerů lze aplikace rychle škálovat, a to i v cloudu.
- Izolace: Kontejnery poskytují izolaci mezi aplikacemi, což snižuje riziko vlivu jedné aplikace na druhou.
- Efektivita: Kontejnery jsou velmi lehké a efektivní, což zajišťuje rychlé spuštění a snadnou správu.

Docker a kontejnery jsou v dnešní době stále více využívány v různých odvětvích, jako jsou například vývoj aplikací, testování, nasazení a správa serverů.[61]

4.5 Node-RED

Node-RED je vizuální nástroj pro tvorbu toků dat a automatizaci, který umožňuje uživatelům snadno propojovat a řídit různé aplikace, služby a zařízení pomocí grafického uživatelského rozhraní. Nástroj je založen na Node.js, a umožňuje uživatelům vytvářet složité Automatizace. Ty se v Node-RED se skládají z různých uzlů. Ty jsou obvykle použity pro sběr dat, filtraci, transformaci a odesílání dat do jiných systémů. Node-RED také umožňuje používat různé protokoly, jako jsou HTTP, MQTT, WebSocket, TCP a další, aby bylo možné komunikovat s různými zařízeními a systémy.[62]

Pomocí Node-RED tak lze využít pro vytváření automatizací pro mnoho různých užití. Například sledování senzorů v reálném čase a řízení zařízení na základě hodnot těchto senzorů, automatizace v domácnostech, nebo i automatizace v průmyslu.

Node-RED také umožňuje snadnou integraci s různými službami a platformami, jako jsou například Google Cloud, Amazon Web Services, IBM Cloud a další, což usnadňuje vytváření automatizací a umožnuje zapojení těchto systému do samotných automatizací. Node-RED je open-source projekt.

Node-RED je široce používaný pro vytváření automatizací v IoT (Internet of Things) projektech, jako je například ovládání osvětlení, topení, alarmů a mnoho dalších aplikací. Díky své jednoduchosti a flexibilitě se Node-RED stal oblíbeným nástrojem pro vývojáře IoT, právě pro svou schopnost snadno vytvářet i komplexnější automatizace.[62], [63]

4.6 Spring Boot

Spring Boot je open-source framework určený pro vývoj aplikací v jazyce Java. Jeho hlavním cílem je usnadnit webových aplikací a zjednodušit jeho konfiguraci.

Mezi významnou výhodu Spring Bootu patří jeho schopnost automaticky konfigurovat většinu aspektů aplikace, vývojáři tak nemusí strávit hodiny nastavováním a laděním konkrétních konfigurací. Framework dodržuje standardní konvence a základní přednastavení, která usnadňují vývoj.

další výhodou je jednoduchá integrace s ostatními nástroji a technologiemi, jako jsou databáze, cache, webové servery a další. Spring Boot poskytuje mnoho užitečných nástrojů pro řešení problémů které je nutno u aplikací řešit, Například logování, testování a zabezpečení aplikace. [64]

4.7 Eclipse Mosquitto

Eclipse Mosquitto je open-source MQTT broker, který slouží k vytváření distribuovaných IoT komunikačních sítí. Protokol MQTT je určen primárně pro IoT zařízení a umožňuje komunikaci v reálném čase.[65]

Mosquitto podporuje verze MQTT 5.0, 3.1 a 3.1.1 a poskytuje publikování a přijímání zpráv s různou kvalitou služeb (QoS). Broker také umožňuje šifrování pomocí TLS/SSL a autentizaci pomocí uživatelských jmen a hesel. Mosquitto lze nakonfigurovat pro použití autorizačních pluginů pro ověřování uživatelů a dalších bezpečnostních funkcí.[65]

Mosquitto běží na různých operačních systémech, jako jsou Linux, Windows a macOS. Broker poskytuje nástroje pro monitorování provozu a sledování stavu spojení mezi klienty a brokerem. V případě výpadku brokeru mohou být zprávy přeposlány na jiný broker pomocí funkcí "bridge", které umožňují vytvoření spojení mezi různými MQTT sítěmi.[66]

Celkově je Eclipse Mosquitto MQTT broker jednoduchý a flexibilní nástroj pro vytváření komunikačních sítí IoT zařízení. Broker má široké využití v průmyslových aplikacích a projektech IoT.[66]

4.8 Grafana

Grafana je open-source nástroj pro vizualizaci dat, který umožňuje uživatelům sledovat a analyzovat data pomocí interaktivních grafů, panelů a dashboardů. Tento nástroj se zaměřuje na rychlé a efektivní zobrazení dat a poskytuje uživatelům mnoho možností pro přizpůsobení vizualizace a analýzy dat podle potřeb uživatele.

Grafana podporuje mnoho datových zdrojů, jako jsou databáze, cloudové služby, protokoly IoT a mnoho dalších. Tento nástroj umožňuje uživatelům propojit data z různých zdrojů a vytvořit z nich komplexní dashboardy pro sledování klíčových metrik a indikátorů výkonnosti.[67]

Grafana poskytuje také rozsáhlou knihovnu pluginů a rozšíření, které umožňují uživatelům rozšířit funkcionalitu tohoto nástroje a přizpůsobit si jej podle svých potřeb. Díky své flexibilitě, jednoduchosti použití a široké podpoře datových zdrojů se Grafana stala velmi populárním nástrojem v oblasti vizualizace dat a monitoringu v reálném čase.[67]

4.9 MVC a Thymeleaf

MVC (Model-View-Controller) je návrhový vzor, který odděluje logiku aplikace od uživatelského rozhraní. V Spring Boot frameworku se používá společně s Thymeleaf šablonovacím enginem pro tvorbu webových aplikací.[68], [69]

Model reprezentuje datovou část aplikace, která je uložena v databázi nebo jiném datovém úložišti. Ta může být ve Spring Bootu reprezentována jako třída nebo rozhraní s metodami pro přístup k datům.

View reprezentuje uživatelské rozhraní a zobrazuje data z modelu. V Spring Bootu se obvykle používá Thymeleaf šablonovací engine, který umožňuje snadnou tvorbu uživatelského rozhraní pomocí HTML, CSS a JavaScriptu.[68]

Controller zpracovává uživatelské vstupy a provádí potřebné operace nad modelem. V Spring Bootu může být Controller reprezentován jako třída s metodami, které jsou označeny jako požadavky na určitou URL adresu.[68]

Díky použití MVC architektury v kombinaci se Spring Boot frameworkem a Thymeleaf šablonovacím enginem lze rychle a snadno vytvořit moderní webové aplikace s oddělenou logikou a uživatelským rozhraním.[68], [69]

4.10 ESPresence

ESPresense je opensource projekt zaměřený na detekci přítomnosti založený na skenování dostupných Bluetooth zařízení. Na stránkách projektu je popsán jako: „Uzel detekce přítomnosti založený na ESP32 pro použití s komponentou Home Assistant mqtt room pro lokalizovanou detekci přítomnosti zařízení, založený na projektu ESP32-mqtt-room“. Software lze velmi jednoduše nahrát na mikrokontroler ESP32, pomocí online pomocí webového instalátoru. Ten se umí prostřednictvím prohlížeče připojit přes usb port ke zvolenému mikrokontroleru. ESPresense následně skenuje všechna Bluetooth

zařízení v okolí a na MQTT server odesílá informaci o zachycených zařízení, včetně parametrů jako je MAC adresa zařízení, síla signálu a odhadovaná vzdálenost. [70]

Home Assistan obsahuje připravenou komponentu pro ESPresence, integrace této služby je tedy velmi jednoduchá. [70]

4.11 Hardware

4.11.1 Raspberry PI

Raspberry Pi 4 je jednodeskový počítač, který byl vydán v roce 2019. I přes své malé rozměry se jedná o plnohodnotný počítač, na který lze instalovat operační systémy. Je vybaven čtyřjádrovým procesorem s frekvencí až 1,5 GHz a až 8 GB operační paměti (k dispozici jsou varianty 2GB, 4GB a 8GB). Je tak schopen zvládat i náročné ukoly, jako je práce se 3D grafikou nebo provoz virtuální stroje.[71], [72]

Raspberry Pi 4 disponuje čtyřmi porty USB, Gigabit Ethernetem, Wi-Fi a Bluetooth. Kromě toho má také dva HDMI výstupy pro připojení dvou monitorů a 3,5mm jack pro připojení reproduktorů nebo sluchátek. Tyto porty umožňují snadné připojení většiny periferií, jako jsou klávesnice, myši, monitory a další.[72]

Raspberry Pi 4 běží na operačním systému Raspbian, který je založen na Linuxu, ale lze na něm spustit i jiné operační systémy, jako je Windows 10 IoT nebo Ubuntu. Díky tomu může být Raspberry Pi 4 využit pro různé účely, jako je například kancelářská stanice a provoz kancelářských aplikací, nebo jako malí webový server.[71]

Jedná se o cenově velmi dostupný počítač, což jej dělá ideálním pro různé projekty a experimenty. Ceny se liší v závislosti na verzi a velikosti paměti, ale zůstávají cenově dostupné pro většinu lidí. To umožňuje vývojářům, studentům a nadšencům využívat Raspberry Pi 4 pro své projekty bez nutnosti investovat velké množství peněz do dražších počítačů.[71], [72]

4.11.2 ESP32

ESP32 je mikrokontrolér, který byl vyvinut společností Espressif Systems a je oblíbený pro mnoho IoT projektů díky své vysoké výkonosti a nízké ceně. Je založený na svém předchudci ESP8266, oproti kterému disponuje například větším počtem GPIO pinů a větší RAM. ESP32 obsahuje dva jádra s frekvencí až 240 MHz, což umožňuje rychlé zpracování dat a efektivní multitasking. Tento mikrokontrolér také obsahuje vestavěné Wi-Fi a Bluetooth moduly, což umožňuje snadné a bezdrátové připojení k internetu nebo dalším zařízením. [73]

Další funkcí ESP32 je podpora mnoha periferií, jako jsou čidla, kamery, reproduktory a další. Tento mikrokontrolér také podporuje různé protokoly a standardy, jako jsou I2C, SPI, UART a další. To znamená, že lze snadno integrovat ESP32 s dalšími zařízeními a senzory. [73]

ESP32 má také nízkou spotřebu energie, což je ideální pro projekty, které mají být napájeny baterií. Tento mikrokontrolér podporuje mnoho funkcí pro snížení spotřeby

energie, jako je tzv. "deep sleep" režim, kdy je mikrokontrolér uspán a spotřebuje pouze velmi malé množství energie.[73]

Celkově lze říci, že ESP32 je malý, výkonný a cenově dostupný mikrokontrolér, který je ideální pro IoT projekty a aplikace s nízkou spotřebou energie. Jeho vysoká výkonnost, široká škála funkcí a snadná integrace s dalšími zařízeními a senzory jej dělají populární volbou pro IoT projekty.[73]

4.11.3 Osvětlení TRÅDFRI

TRÅDFRI je systém chytrého osvětlení od společnosti IKEA. Systém se skládá z různých druhů osvětlení jako jsou žárovky a led pásky či panely spolu s ovládacími prvky jako jsou bezdrátové ovladače nebo například sensor pohybu. Kromě osvětlení pak také systém nabízí například chytré rolety nebo reproduktory. Ikea je členem ZigBee Aliance, a tak i její výrobky využívají pro konektivitu protokol ZigBee. [74], [75]

Součástí systému je aplikace, které slouží k jeho ovládání. aplikace umožnuje uživatelům ovládat intenzitu světla, nastavit jeho teplotu čí barvu. Uživatelé mohou také vytvářet různé scény podle svých potřeb či nastavovat časovače pro konkrétní akce. [74]

V rámci návrhu chytré domácnosti budeme využívat žárovky a ovladače z tohoto systému. Nebudeme využívat aplikace určenou k ovládání celého TRÅDFRI systému nýbrž jednotlivá zařízení zapojíme do vlastní ZigBee sítě.[74]

4.11.4 Sonoff SNZB-03

Sonoff ZigBee Motion Sensor SNZB-03 je malý bezdrátový senzor pohybu, který komunikuje pomocí protokolu Zigbee a je napájen pomocí jedné baterie CR2450, přičemž výrobcem uvadávána výdrž je až dva roky. Senzor je vybaven infračerveným senzorem pohybu a dokáže detekovat pohyb v rozsahu až 6 metrů. V rámci komunikace pak sensor vysílá dva typy signálu. První indikuje zahájení detekce pohybu a druhý pak její ukončení. Sensor tedy neodesílá zprávu při každé detekci naopak podle vlastní logiky rozhodne, kdy již pohyb přestal být detekován a odešle zprávu z touto informací. Výrobce neuvádí, podle kterých parametrů sensor rozhoduje, že žádný pohyb již není detekován. [76]

4.11.5 Sonoff SNZB-04

Sonoff SNZB-03 je bezdrátový senzor dveří a oken, který umožňuje sledovat stav dveří, oken nebo jiných vstupních bodů v domě, bytě nebo kanceláři. Tento senzor používá bezdrátovou komunikaci protokolu Zigbee, což znamená, že ho lze snadno spárovat s jinými Zigbee zařízeními, jako jsou například chytré žárovky, vypínače nebo termostaty.[77]

Senzor dveří Sonoff SNZB-03 se skládá z dvou částí – magnetického čidla a báze. Magnetické čidlo se umístí na dveře nebo okně a báze se umístí na rámu dveří nebo okna. Když jsou dveře nebo okna uzavřeny, magnetické čidlo je v blízkosti báze, což způsobuje, že senzor ukazuje, že jsou dveře nebo okna uzavřeny. Pokud jsou dveře nebo

okna otevřeny, magnetické čidlo se vzdálí od báze, což způsobí, že senzor ukazuje, že jsou dveře nebo okna otevřené.[77]

4.11.6 MQ-9

MQ-9 je plynový senzor, určený k detekci oxidu uhelnatého (CO), methanu (CH4) a dalších hořlavých plynů. Senzor má malé rozměry a nízkou spotřebu energie, což ho dělá vhodným pro použití v rámci IoT, hodí se tak pro monitorování nadměrného výskytu CO2 nebo kvality ovzduší.[78]

MQ-9 senzor obsahuje snímač elektrochemického typu, který reaguje na změny v koncentraci, CO a CH4 v okolním prostředí. Tento senzor je citlivý na koncentrace plynů v rozsahu 10 až 1000 ppm (parts per million) a může být použit v kombinaci s mikrokontroléry nebo jinými zařízeními pro sběr dat.[78]

4.11.7 DHT11

DHT11 je digitální teplotní a vlhkostní senzor. Tento senzor se skládá z elektrochemického senzoru vlhkosti a teplotního čidla, které jsou zasazeny do malého plastového krytu. Díky svému malému rozměru a nízké ceně a energetické náročnosti je DHT11 senzor vhodný pro použití v rámci IoT.[79]

DHT11 senzor měří teplotu v rozsahu 0 až 50 stupňů Celsius s přesností +/- 2 stupně Celsia a relativní vlhkost v rozsahu 20 až 90% s přesností +/- 5%. Senzor komunikuje pomocí jednoduchého digitálního signálu, který je snadno čitelný mikrokontroléry nebo jinými zařízeními pro sběr dat.[79]

4.11.8 Zigbee 3.0 USB Dongle Plus

Zigbee 3.0 USB Dongle Plus je adaptér, založený na chpsetu CC2656P, který umožňuje komunikaci pomocí protokolu ZigBee. Tento adaptér je vybaven nejnovější verzí standardu Zigbee 3.0, což zajišťuje kompatibilitu s širokou škálou zařízení. Adaptér používá software vyvíjený společností Texas Industries, která je zároveň jeho výrobcem. V případě potřeby je možné firmware adapterů updatovat pomocí tzv. flashe.[80]

Adaptér se připojuje k USB portu počítače nebo jiného zařízení a komunikuje pomocí ZigBee protokolu s ostatními ZigBee zařízeními. Zařízení, které tento adaptér používá muže být použit jako koordinátor i koncové zařízení sítě.[80]

5 Návrh řešení

Ačkoliv výsledný návrh bude implementován v rámci konkrétní domácnosti, cílem tohoto není vytvořit chytrou domácnost na míru pro konkrétní domácnost. Výsledný návrhy by měl být aplikovatelný do libovolné domácnosti, bez nutnosti návrh příliš upravovat. Zároveň je pak možné implementoval pouze část návrhu a vynechat části, o jejichž služby nemá uživatel zájem. Pokud například uživatel nebude vyžadovat složité

automatizace je možné vynechat Node-Red, nebo pokud nebude zapotřebí analyzovat data ze sensoru nebude zapotřebí Grafana.

5.1 Požadavky

5.1.1 Funkční požadavky

- Ovládání zařízení
    - Systém by měl umožnit ovládání různých zařízení v domě, jako jsou osvětlení, topení, klimatizace, zabezpečovací systémy a další, z jednoho místa, například z mobilního telefonu nebo tabletu.
- Možnost integrace ZigBee zařízení
    - Systém by měl umožňovat integraci nových ZigBee zařízení do centrálního bodu systému a jejich následné využití v automatizacích,
- Možnost integrace MQTT
    - Systém by měl umožňovat integraci nových zařízení pomocí MQTT protokolu a jejich následné využití v automatizacích,
- Plánování a automatizace
    - Systém by měl umožnit programování automatizovaných činností v domě, jako je zapínání a vypínání osvětlení a topení, závlaha zahrady a podobně, aby uživatelé mohli ušetřit energii a peníze.
- Detekce přítomnosti
    - V rámci automatizací by chytrá domácnost měla umět rozpoznat přítomnost člověka. Detekce přítomnosti pak může být použita jak pro zpuštění automatizací, tak pro upravení jejich chování.
- Monitorování CO_2
    - Systém by měl umožnit monitorování úrovně CO_2
- Odesílání notifikací
    - Systém by měl být schopen posílat uživatelům upozornění na různé události, může se jednat například o detekci zvýšeného množství CO^2, Požáru nebo úniku vody.

5.1.2 Nefunkční požadavky

- Integrace do jednoho centrálního bodu
    - Integrací je myšleno propojení jednotlivých zařízení v chytré domácnosti do jediného rozhraní nebo aplikace. Integrace je pro chytrý dům klíčová, protože umožňuje bezproblémovou komunikaci mezi zařízeními umožňuje automatizaci napříč všemi zařízeními v domácnosti. Například chytrý termostat se může integrovat se chytrým osvětlením, aby automaticky vypnul světla, když termostat zjistí, že nikdo není doma.
- Ovládání přes webové rozhraní
    - Vybraná zařízení chytré domácnosti by měla být ovladatelné pomocí jednoduchého webového rozhraní.
- Kompatibilita
    - Kompatibilita se vztahuje na schopnost chytrých zařízení pracovat s jinými zařízeními nebo platformami. Kompatibilita umožňuje uživatelům

používat různá zařízení od různých výrobců a integrovat je do svého chytrého domu. Kompatibilita je důležitá, protože umožňuje uživatelům využívat různá zařízení bez omezení výběru.

- Škálovatelnost
    - Škálovatelnost se vztahuje na schopnost chytrého domu přizpůsobit se potřebám uživatele v budoucnosti. Chytrý dům by měl být navržen tak, aby bylo možné přidávat další zařízení a funkce v průběhu času, bez nutnosti kompletního přepracování systému. Škálovatelnost je důležitá, protože umožňuje uživatelům přizpůsobovat si svůj chytrý dům podle svých budoucích potřeb a rozšiřovat ho v průběhu času.

5.2 Návrh řešení

Je zapotřebí zvolit centrální bod systému, do kterého bude možné integrovat veškerá zařízení. Jako tento bod byl zvolen software Home Assistant. Ten tedy umožňuje integraci všech chytrých zařízení, která budou v rámci chytré domácnosti použita a poskytuje uživatelsky přívětivé rozhraní pro správu a monitorování chytrých zařízení. Podporuje širokou škálu zařízení a služeb a umožňuje vytváření vlastních automatizací a skriptů pro ovládání zařízení a reakci na události.

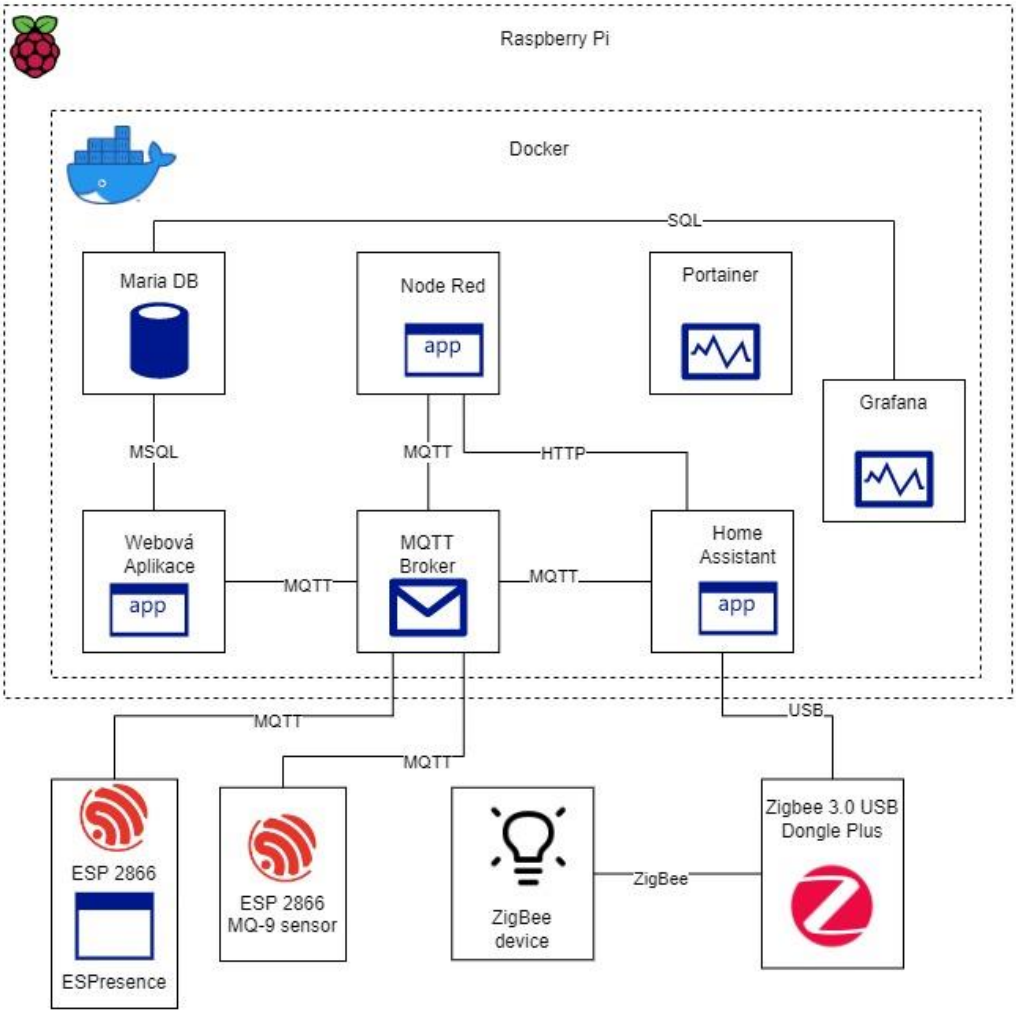
Jednodušší automatizace lze snadno vytvářet v systému homeassistant. Pro pokročilejší automatizace byl zvolen již zmíněný node Node-RED. Ten poskytuje vizuální rozhraní pro vytváření komplexních pracovních postupů a integraci více zařízení a služeb. Podporuje širokou škálu protokolů a rozhraní, což z něj činí všestranný nástroj pro vytváření vlastních automatizací.

Pro připojení většiny chytrých zařízení do systému bude použit Protokol Zigbee, který poskytuje spolehlivou a bezpečnou komunikaci mezi zařízeními. Do systému bude také připojení několik sensoru. Tyto zařízení budou založeny na mikrokontroléru ESP32. Pro integraci bude využito WiFi rozhraní tohoto mikrokontroléru spolu s sprotokolem MQTT.

Pro jednoduché ovládání systému bude vytvořen jednoduchý web založený na frameworku Spring Boot ten bude sloužit jako zjednodušené grafické rozhraní mezi Home Assistant a uživatelem. Ke komunikaci s Home Assistant bude web využívat MQTT, což bude uživatelům umožňovat jednoduché ovládání zařízení a přístup k datům ze senzorů v reálném čase. Přistup k datum ze sensorů je z webové aplikace omezený. V rámci každého sensoru jsou zobrazena pouze data za posledních dvacet čtyři hodin. Pro detailnější přístup k datum tak bude použita Grafana. Ta nám umožní detailnější analýzu a zobrazení dat v případě potřeby.

Architektura systému chytré domácnosti je navržena jako modulární a škálovatelná, což umožňuje snadné přidávání nových zařízení a služeb podle potřeby. Použití kontejnerů Docker umožňuje efektivní využití zdrojů a snadné nasazení různých komponent. Celkově je systém navržen jako flexibilní a adaptabilní řešení pro správu a automatizaci chytrých zařízení v domácím prostředí.

Pro zjednodušení správy a monitorování jednotlivých kontejneru s aplikacemi bude využita služba Portainer. Ta umožňuje uživatelům snadno vytvářet, spouštět a spravovat kontejnery a jejich síťové konfigurace. Konfigurace a správa kontejneru probíhá pomocí jednoduchého grafického rozhraní. Mimo to Portainer také poskytuje nástroje pro sledování výkonu, správu uživatelů a řízení přístupu. Tyto nástroje ovšem nebudou v rámci navrhované chytré domácnosti zapotřebí.



Obrázek 12: Schéma chytré domácnosti [zdroj: autor]

5.3 Webová aplikace

Součástí řešení bude také webová stránka určená k jeho ovládání. Tato kapitola se zabývá jejím návrhem.

5.3.1 Požadavky

- Ovládání světel
  - aplikace by měla umožnit jednoduché ovládání světel. Vypnutí, zapnutí, změna jasu a změna barvy světla.

- Zobrazení dat ze sensorů
    - aplikace by měla umět zobrazovat data ze sensoru, aplikace by mela ukazovat jak aktuální data, tak i graf s historickými daty.
- Přidání světla do služby
    - vytvoření nového světla, které bude možné webovou službou ovládat. Nejedná se o integraci nového světla do systému chytré domácnosti ale o vytvoření ovládacího prvku v rámci aplikace, který bude takové světlo schopný ovládat.
- Přidání sensoru do služby
    - vytvoření nového sensoru, tak aby byla aplikace schopna načítat jeho data a zobrazovat je. Nejedná se o integraci nového sensoru do systému chytré domácnosti ale o vytvoření prvku v rámci aplikace, který bude takové schopný zobrazovat data takového sensoru.
- Rozdělení prvku do skupin.
    - Jednotlivé ovládací prvky by mělo být možné rozdělit do jednotlivých skupin, například podle místností, ve kterých se nachází.

5.3.2 Návrh aplikace

Hlavním prvkem aplikace budou dashboard (nástěnky). Těch bude možné vytvářet libovolné množství a budou složit ke zobrazovaní prvků s ovládáním světla či daty ze sensorů. Zároveň budou dashboardy sloužit k seskupení těchto prvku do skupin, například podle místností.

Pro ovládání světel bude nutné vytvořit samostatný prvek, který bude možné přidat na libovolný dashboard. Ten to prvek by měl obsahovat, jak možnost světlo vypnout nebo zapnout tak i možnost pro nastavení jasu. Tento prvek bude k ovládání světla používat protokol MQTT bude tedy nutné k danému prvku přiřadit topiky pro stav a jas světla. Kromě topiku bude také zapotřebí znát maximální a minimální hodnoty jasu daného světla. Může se totiž například jednat o hodnoty 0 až 100 nebo třeba 0 až 255. V rámci každého ovládacího prvku pro světlo tedy bude nutno uchovávat tyto informace. Zároveň by měla byt ke každému prvku uchována informace o posledním známém stavu a jasu, aby bylo možné ovládací prvek nastavit na tyto hodnoty při jeho načtení.

Druhým prvkem bude prvek, který slouží pro zobrazení dat ze sensoru. Pro tento prvek bude nutné znát topik, do kterého jsou pravidelně publikovaná data, které sensor naměřil.

Aplikace by měla automaticky odebírat topiky všech těchto prvku (světla a sensory), které byly v aplikaci vytvořeny. Příchozí data pro světla se potom budou ukládat jako poslední známa hodnota. Pro data ze sensoru bude nutné držet historii všech naměřených dat, aby se výstup ze sensoru dal zobrazit jako graf.

5.3.3 Wireframes

Pro návrh webové aplikace byly využity wireframes. Aplikace pracuje se dvěma druhy prvků, světli a sensory. Aby byla aplikace přehledná. Je možné vytvářet jednotlivé nástěnky. Na ty lze přidáváte jednotlivé prvky (světla a sensory). Lze tedy vytvářet nástěnky pro jednotlivé místnosti anebo například nástěnku ztužující důležitá světli či sensory. Výsledné wireframes se nacházejí v příloze B.

Prvek pro ovládání světla bude obsahovat vypínač, kterým bude možné světlo vypnout nebo zapnout. Dále bude obsahovat posuvník, kterým bude možné nastavit jas světla a název světla, aby bylo možné identifikovat které světlo tento element ovládá. Druhým prvkem, který bude možné přidat na dashboard prvek pro zobrazení dat ze sensoru. Ten se bude skládat z názvu sensoru grafu který obsahuje data ze sensoru za poslední den a zobrazení poslední známé hodnoty.

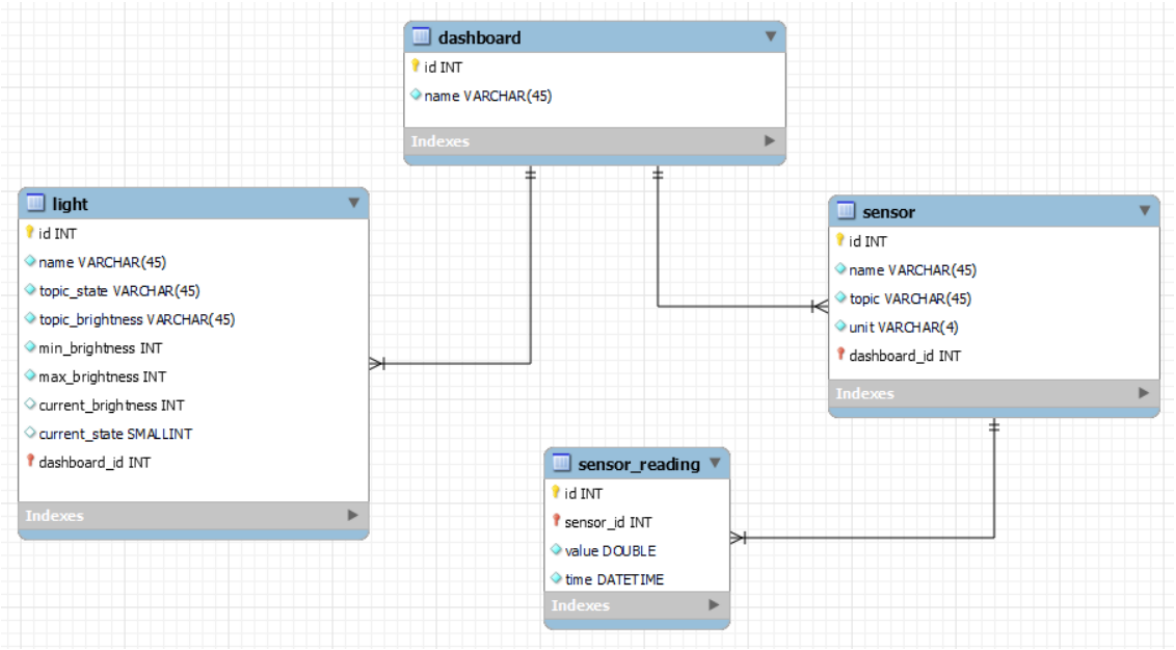
Následně bude aplikace obsahovat formuláře pro vytváření či editaci jednotlivých prvků (sensor, světlo, dashboard).

5.3.4 Databáze

V databázi bude nutné udržovat všechny výše zmíněné entity, tedy dashboard, světlo a sensor. První tabulkou je dashboard, ten se skládá pouze z názvu a id, které databáze generuje sama pomocí autoincrementu. A reprezentuje dashboardy vytvořené v aplikaci

Další vytvořenou tabulkou je světlo. Jednotlivé řádky tabulky reprezentují ovládací prvky světel vytvořené v aplikaci. Každý ovládací prvek náleží nějakému dashboardu, mezi světlem a dashboardem je tedy vazba 1:n reprezentovaná pomocí cizího klíče v tabulce světlo. Dále pak tabulka obsahuje řádky pro topiky se stavem a jasem světla, jeho název a minimální a maximální jas. Všechny tyto prvky jsou povinné. Dále pak v tabulce existují dva nepovinné sloupce pro poslední známou hodnotu stavu a jasu. Nepovinné jsou z toho důvodu, že při vytváření ovládacího prvku není jejich stav známí.

Prvky pro zobrazení dat sensoru jsou také uloženy ve vlastní tabulce. Tabulka obsahuje název sensoru, topik, kam jsou publikovány jeho data a jednotka ve které jsou data měřena. Stejně jako ovládací prvek pro světlo potom obsahuje 1:n vazbu na dashboard reprezentovanou cizím klíčem. K sensorům je také nutno ukládat naměřená data. Jelikož chceme od naměřených dat udržovat historii. Byla pro data sensoru vytvořena samostatná tabulka. Ta obsahuje cizí klíč s vazbou na konkrétní sensor, kterému data náleží, čas měření a naměřenou hodnotu.



Obrázek 13: databázový model [zdroj: autor]

5.4 Automatizace

Z hlediska automatizace se nabízejí dva možné postupy, které lze kombinovat. Pro jednoduché automatizace jako je propojení vypínačů a světel použijeme automatizace které lze vytvářet přímo v prostředí služby Home Assistant. Tyto automatizace je možné vyvážet buďto pomocí jednoduchého grafického prostředí anebo pomocí sterilizačního jazyka YAML. Grafické prostředí je vlastně jen jakousi uživatelskou nadstavbou, které výsledek převádí do YAML souboru.

Pro složitější automatizace pak bude použit nástroj Node-RED. Ten umožňuje vytvářet složité automatizace pomocí jednoduchého drag-and-drop rozhraní, které je vhodné i pro uživatele bez programátorských zkušeností. Pomocí Node-RED lze vytvářet automatizace na základě více podmínek a akcí, které jsou navzájem snadno propojovat jako jednotlivé uzly. To usnadňuje tvorbu složitějších automatizací, které zahrnují vetší množství podmínek, nebo automatizací které mají složitější průběh, například obsahují nějaký časovač.

5.5 Škálování

Jedním z požadavků je možnost škálování systému, pokud se systém dostane do stavu, kdy nebude jeho výkon dostatečný. Hlavní možnosti škálování je vertikální nebo horizontální škálování. Vertikální škálování je založené vytvoření více instancí jednotlivých služeb a vytvořením balancingu mezi jednotlivými instancemi. Tento postup se používá spíše pro služby, které řeší zpracování velkého množství požadavků najednou a příliš se nehodí pro chytrou domácnost. Druhou možností je pak vertikální škálování, aplikace stále poběží pouze v jedné instanci, ale bude přenesena na stroj s většími hardwarovými zdroji. Tento přesun mezi servery nám výrazně usnadňuje fakt, že veškeré služby provozované jsou provozované v docker kontejnerech. Kontejnery vytvářejí tzv volume což je souborový systém vytvořený v rámci služby docker. Jednou možností je pak přenesení těchto volume mezi jednotlivými servery a následné spuštění kontejnerů

s těmito volume a původním image. Alternativně také docker umožnuje export kontejneru [1] a jejich spuštění na jiném stroji.

Případné škálování služeb tedy pude probíhat vertikálně pomocí přesunu jednotlivých kontejnerů mezi stroji.

5.6 Detekce přítomnosti

Dalším požadavkem je detekce přítomnosti lidí. Ta bude řešena pomocí tří různých senzorů. Prvním sensorem je sensor pohybu popsaný v kapitole 4.11.4 , který je schopný detekovat pohyb na vzdálenost šesti metrů. Druhým z nich je senzor na dveře popsaný v kapitole 4.11.5. Tento senzor se hodí zejména pro spouštění automatizací, nevýhodou je že na základě dat z tohoto sensoru nelze určit, jestli člověk místnost opouští nebo do ní naopak vstupuje. Jako poslední sensor pak slouží ESP32 se službou ESPresence. Tato služba zaznamenává viditelná Bluetooth zařízení v okolí včetně jejich MAC adresy. Pokud zjistíme adresy zařízení jednotlivých členů domácnosti, lze tyto data použít i identifikaci přítomných osob.

[1]Export docker kontejnerů: https://docs.docker.com/engine/reference/commandline/export/

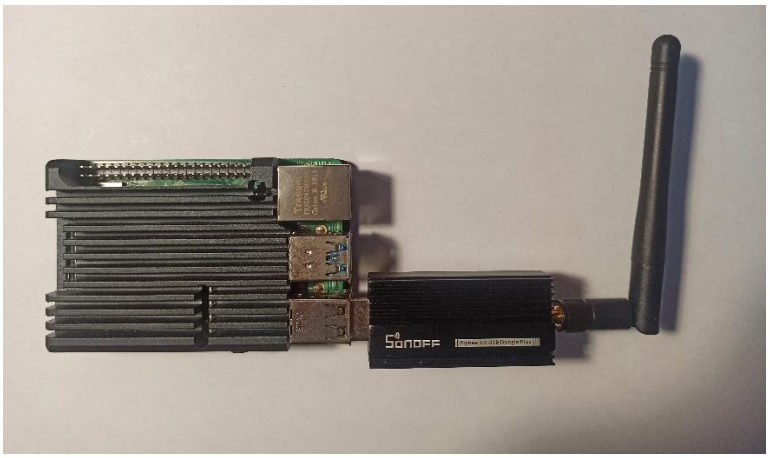
6 Realizace

Tato kapitola se věnuje realizace řešení podle návrhu uvedeného v předchozí kapitole.

6.1 Server

Jako server pro chytrou domácnost byl zvolen jednodeskový počítač Raspberry Pi 4. Prvním nutným krokem je instalace operačního systému. Jako OS byl zvolen Raspberry Pi OS Lite (64 bit). Jedná se o operační systému určený přímo pro Raspbbery Pi. OS je linuxová distribuce založena na Debianu. Lite verze znamená, že se jedná o verzi, která neobsahuje žádné grafické uživatelské prostředí. Tato verze se tak hodí pro užití, kdy k přístupu a užívaní nebudeme užívat GUI ale použijeme například konzoli. Díky tomu že OS neobsahuje GUI nezabírá tolik místa na uložišti a zároveň méně vytěžuje procesor.

Pro instalaci OS byl použit program Rufus pro vytvoření bootovacího disku z micro SD karty. Po jejím zasunutí do Raspbbery Pi se OS automaticky načetl a zařízení bylo připraveno k použití.



Obrázek 14: použité Raspberry Pi 4 [zdroj: autor]

6.1.1 instalace zvoleného SW

Zvolený software je na serveru provozován v docker kontejnerech. Díky nim lze pak software snadno udržovat a případně updatovat, zároveň jednotlivé programy běží v izolovaném prostředí což minimalizuje nechtěné interakce mezi jednotlivými programy.

Pro toto řešení je tedy nutné nainstalovat docker. Při instalaci jo možné vycházet z oficiálního návodu pro Debian[2], na kterém je Raspberry Pi OS založen. Pro usnadnění používaní Dockeru byl také nainstalován doplněk Compose. Ten umožnuje spouštění a správu kontejneru na základě yaml skriptů. Usnadňuje tak psaní potenciálně dlouhých a

[2] Raspberry Pi instalace Docker: https://docs.docker.com/engine/install/debian/

komplikovaných parametrů které jsou nahrazeny právě těmito skripty. Obdobně jako při instalaci Docker je možné při instalaci vycházet z oficiální dokumentace[3].

Instalace zvolených SW probíhá velmi podobně. Pro každou službu je zvolen image, ze kterého který se bude používat pro docker kontejner. Ke všem zvoleným službám existují již vytvořené docker image, které jsou udržovány a jsou vydávány jejich nové verze. Všechny tyto image jsou zdarma dostupné ve veřejných repositářích. Pokud by pro některou ze služeb nebyl žádný dostupný image, bylo by nutné vytvořit Dockerfile skript. Ten slouží k vytvoření nového image. V rámci tohoto skriptu je nutné zvolit vhodný image ze kterého budeme vycházet, následně do doinstalovat veškeré potřebné knihovny a závislosti, v poslední řadě pak naistalovat samotnou službu. Kromě výchozího image pak také bylo nutné namapovat porty. Použité docker compose soubory jsou dostupné v příloze.

Většina služeb potřebuje být přístupná z vnějšího prostředí, jelikož docker kontejnery jsou izolované, je nutné namapovat porty kontejneru na porty hostující zařízení[4] čímž je učiníme přístupné v rámci sítě na které je hostující zařízení připojeno (v tomto případě je hostujícím zařízením Raspberry Pi).

Služby si také udržují různé soubory, které potřebují ke svému provozu. Jedná se obvykle o konfigurační soubory nebo data která potřebují uchovávat. Docker vytváří volumes, kde jsou tato data uchovávána. Problémem ovšem je, že pokud by Docker nebo celé Raspberry Pi nečekaně spadlo, je možné že budou tato data ztracena. Řešením tohoto problému je obdobně jako u portů mapovaní na hostující stroj. Podstatné části adresářového sytému kontejneru lze namapovat na adresářový systém hosta, V případě nečekaného ukončení jsou tak data uchována v adresářovém systému hosta a při opětovném spuštění kontejneru z něj budou znovu načtena.

Následující část se týká Kontejneru, ve kterém běží Home Assistant. Ten potřebuje přístup k USB Zigbee adaptéru, který je připojen k Raspberry Pi. A obdobně jako v předchozích případech nastává problém, kdy kontejner, díky který je izolovaný nemá defaultně k připojeným zařízením přístup. I v tomto případě je problém řešen mapování. Tentokrát se jedná o mapovaní zařízení připojených k hostu na konkrétní kontejner.

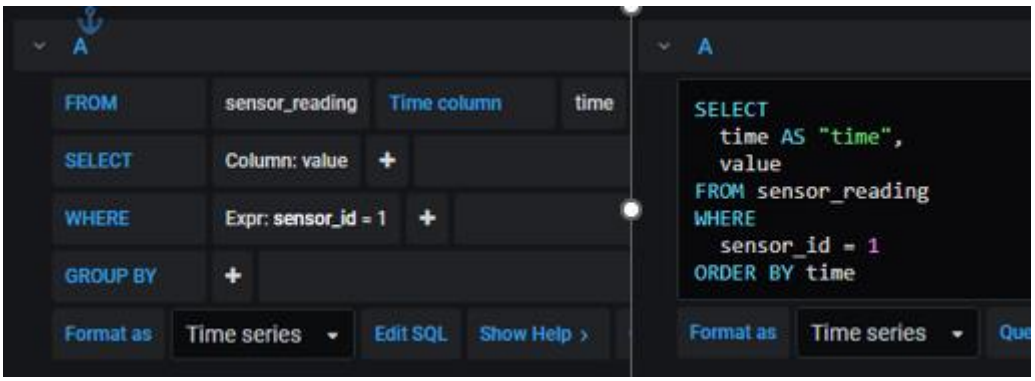
[3] Instalace Docker Compose: https://docs.docker.com/compose/install/linux/

[4] Alternativně by bylo možné použít docker networking, ale to pouze v případě že by jednotlivé kontejnery potřebovali komunikovat pouze mezi sebou. Pro komunikace směrem do kontejneru mimo docker by je nutné použít mapování portů.

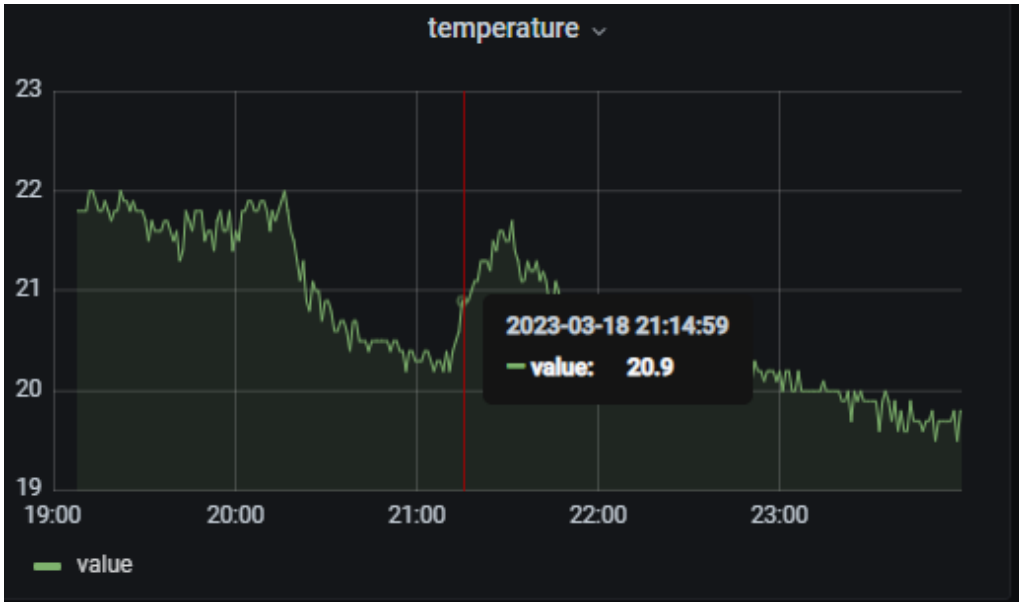
```yaml
version: '3.0'

services:
  homeassistant:
    image: ghcr.io/home-assistant/home-assistant:stable
    container_name: homeassistant
    network_mode: host
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/London
    volumes:
      - ./config:/config
    devices:
      - /dev/ttyACM0:/dev/ttyACM0
    restart: unless-stopped
```

Obrázek 15: Docker compose pro službu home assistant [zdroj: autor]

Nastroj Grafana stejně jako ostatní software zprovozníme pomocí docker kontejneru. Po jeho vytvoření je nutno vytvořit dashboard kde bude možné pozorovat data ze sensorů. Prvně je třeba vytvořit zdroj dat, tím je v tomto případě databáze. Dále pak již vytvoříme nový dashboard a specifikujeme použitý zdroj dat a dotaz kterým lze data získat. To lze buďto pomocí sql query nebo pomocí query builderu. Následně nastavíme že se data mají zobrazovat jako časová řada. Výsledkem je přehledný graf, se kterým můžu uživatel pohodlně pracovat a například si vybrat konkrétní časová období která ho zajímají.



Obrázek 16: SQL vs query builde v nástroji grafana [zdroj: autor]



Obrázek 17: výsledný graf v nástroji Grafana [zdroj: autor]

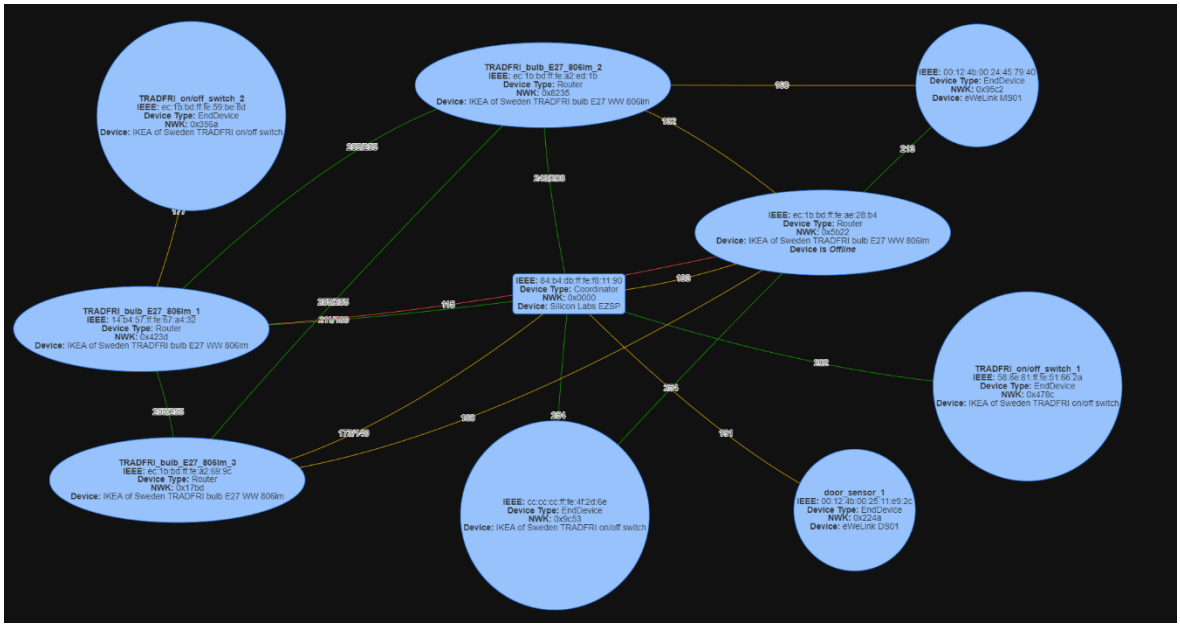
6.1.2 Provoz a monitorování

Pro monitorování byla zvolena již zmíněná služba Porteiner. Ten je možný provozovat v dockerovém image a umožnuje snadné monitorování dockeru. Pro nás je podstatné monitorovaní běžících kontejnerů. Porteiner také obsahuje nástroje na správu kontejneru. Lze je například zastavit restartovat či pozastavit jejich běh.

6.1.3 Integrace koncových zařízení

Většina použitých zařízení protokol ZigBee. Prvně je nutné vytvořit ZigBee koordinátora ke kterému bude možné jednotlivá zařízení připojit. Home Assitent obsahuje rozšíření Zigbee Home Automation. Při nastavení se pak vybere zařízení které bude toto rozšíření používat jako koordinátora. Konkrétně se jedná o zařízení 4.11.8 , které je připojeno na adrese „/dev/ttyACM0“ viz. Obrázek 15.

Dále je pak jen nutné přepnout koordinátora do režimu discovery a následně všechna zařízení resetovat. Při resetu zařízení zapomenout předchozí síť a obvykle se připojí ke koordinátorovi, pokud je v režimu discovery. Výsledkem je pak ZigBee sít zobrazená na následujícím obrázku.



Obrázek 18: výsledná ZigBee síť [zdroj: autor]

Zařízení komunikující pomocí MQTT protokolu lze do Home Assitant integrovat pomocí jednoduché úpravy konfigurace. V konfiguračním souboru se nastaví název zařízení, topik, ze kterého zařízení (respektive jeho reprezentace v Home Assitant) čerpá data a případné další parametry[5]. Na základě této konfigurace je potom v rámci Home Assistant vytvořeno zařízení, jehož stav bude aktualizován podle poslední hodnoty, která byla publikována v daném topiku.

```yaml
mqtt:
  sensor:
    - name: "bedroom_temperature"
      state_topic: "/jhlubucek/sensor/temp"
      unit_of_measurement: "°C"
```

Obrázek 19: konfigurace pro přidání MQTT sensoru

Obrázek 20: MQTT sensor v prostředí Home Assistant [zdroj: autor]

6.2 ESP32

V chytré domácnosti se nacházejí dvě ESP32. První slouží k monitorování domácnosti pomocí senzorů DHT11 MQ-9. Pro čtení dat ze sensorů a jejich odesílání pomocí MQTT byl v rámci práce vytvořen vlastní kód. Druhé ESP32 je pak použito k provozování služby ESPresence.

[5] Konfigurace MQTT sensoru v HA: https://www.home-assistant.io/integrations/sensor.mqtt/

6.2.1 ESPresence

Instalace ESPresence je velmi jednoduchá. Instalace probíhá pomocí webového instalátoru[6]. Stačí tedy ESP32 připojit k počítači pomocí USB kabelu, zvolit port na kterém je zařízení připojeno a instalátor se i falšovaní programu postará automaticky.

Po instalaci ESPresence vytvoří stejnojmennou WiFi síť, na tu je nutné se připojit, aby bylo možné zařízení konfigurovat. Po připojení do sítě najdeme formulář s konfigurací na adrese ESPresence 192.168.4.1. Zde lze nastavit připojení k WiFi síti, MQTT brokerovy a název místnosti, ve které se ESP32 bude nacházet. Nachází se zde více možných nastavení[7] ale výše uvedené stačí ke zprovoznění základních funkcionality.

Následně lze pomocí MQTT klienta sledovat data o zařízeních, které ESPresence detekovalo. Data jsou odesílána do topiku „espresense/rooms/bedroom“ bedroom je název místnosti v konfiguraci, a bude se tedy měnit na základě nakonfigurované místnosti dané instance ESPresence. Na následujícím obrázku je vidět ukázka dat, které ESPresence odesílá o zpozorovaných zařízeních.


```
{
  "id": "sd:0xfe9f",
  "idType": 15,
  "rssi@1m": -71,
  "rssi": -62,
  "raw": 0.55,
  "distance": 0.55,
  "speed": 0,
  "mac": "7434347fe419",
  "interval": 670
}
```


Obrázek 21: ukázka odpovědi z ESPresence

Zde je pro nás podstatné ID daného zařízení, podle kterého jej lze obvykle identifikovat. Užitečná pak může být také odhadovaná vzdálenost zařízení.

Home Assistant je schopen sám rozpoznat ESPresence skrze MQTT protokol a službu sám nakonfiguruje. Následně je pak nutné v konfiguraci Home Assistant vytvořit jednotlivé entity, jejichž status bude Home Assistant sledovat.

[6] Webový instalátor ESPresence: https://espresense.com/firmware

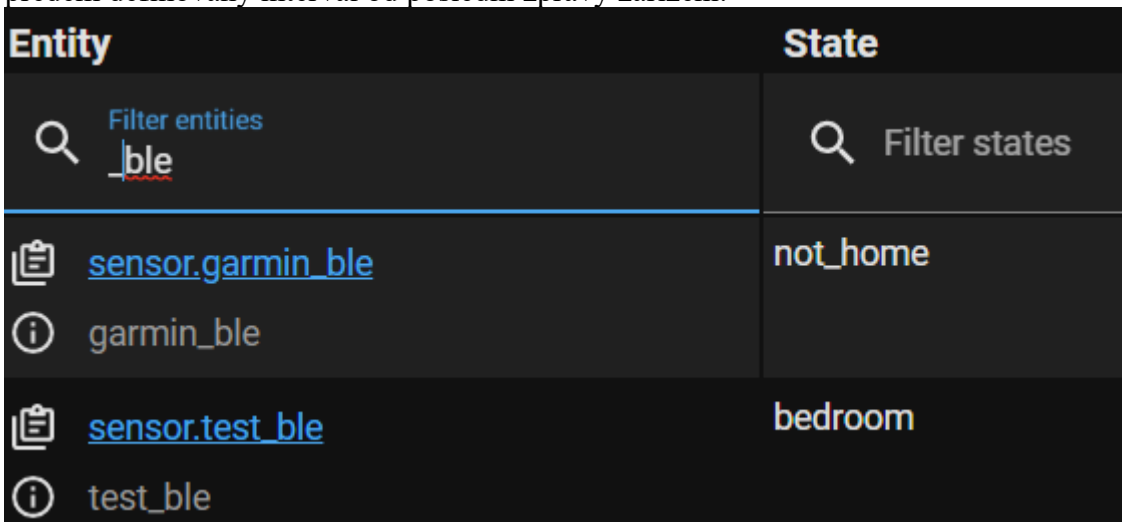
[7] Konfigurace ESPresence: https://espresense.com/configuration/settings

```yaml
sensor:
  - platform: mqtt_room
    name: "garmin_ble"
    device_id: "known:90f15720e618"
    state_topic: "espresense/rooms"
    timeout: 30
    away_timeout: 120
  - platform: mqtt_room
    name: "test_ble"
    device_id: "nut:702845633080"
    state_topic: "espresense/rooms"
    timeout: 10
    away_timeout: 120
```

Obrázek 22: konfigurace ESPresence detekovaných zařízení v Home Assistant [zdroj: autor]

- name
  - Název zařízení, pod kterým ho bude možné v rámci Home Assistant identifikovat.
- device_id
  - Reprezentuje id zařízení, proměnná id z příchozích dat z ESPresence
- state_topic je
  - Název topiku na kterém Home Assistant očekává informa o daném zařízení.
- timeout
  - Značí předpokládanou prodlevu mezi jednotlivými zprávami jedná se v podstatě o TTL dané zprávy (v sekundách), přičemž po jejím uplynutí je zpráva považována za starou.
- away_timeout
  - Prodleva (v sekundách) pokud po jejím uplynutí nepřijde žádná zprava od daného zařízení nastaví se jeho status jako „not_home“

Na základě této konfigurace se v Home Assistant vytvoří entity, jejichž status se aktualizuje na základě příchozích zpráv. Statusem těchto entit je název místnosti, ve které se nacházejí, lze tedy detekovat přechod mezi jednotlivými místnostmi, existuje v rámci domácnosti více instancí ESPresence. Alternativně pak „not_home“ pokud uplynul předem definovaný interval od poslední zprávy zařízení.



Jako problém se ukázalo používaní detekce Bluetooth ve spojitosti s některými zařízeními. V tomto případě se jednalo konkrétně o hodinky Garmin Venu 2. Po jejich integraci do systému byly detekovány pouze nahodile a po krátké časové úseky. Při bližším zkoumání byla nalezena spojitost s tím, zda jsou hodinky připojeny k mobilnímu telefonu. Hodinky bylo pomoci Bluetooth skenu možné objevit pouze v případě že nebyli zrovna připojeny k žádnému zařízení. V dokumentaci zařízení se nepodařilo objevit vysvětlení pro toto chování. Zároveň nutno podotknout, že se nejedná běžné užití tohoto zařízení. Na základě online diskuzí[8] se lze dočíst že se nejedná o ojedinělí problém ale pravděpodobné o běžné chování tohoto zařízení. Při výběru zařízení, které bude použito pro detekci je tedy nutno pomatovat na to, že ne všechna zařízení mohou být detekovatelná pomocí Bluetooth skenu. Jako zařízení pro detekci pomocí ESPresence byl nakonec použit mobilní telefon s operační systémem Android.

6.2.2 MQ-9 a DHT11 sensory

Druhé ESP32 je použito k načítání dat ze sensoru a jejich publikování pomocí MQTT. Tato kapitola se věnuje programu, který byl za tímto účelem vytvořen.

Jako programovací jazy pro ESP32 byla zvolená upravená verze C++, kterou používá platforma Arduino. Pro sestavení prototypu pak bylo využito nepájivé pole.

Program obsahuje konfigurační soubor, kde lze definovat všechny podstatné proměnné, jako například použité piny pro dané sensory, nebo konstanty použité při měření.

```
22  //dhtsensor
23  #define DHTPIN 14
24  #define DHTTYPE DHT11
25  //MQ9 sensor
26  #define MQ9PIN 35
27  #define MQ9R0 2.55
```

Obrázek 23: Konfigurace ESP32 s DHT11 sensorem [zdroj: autor]

Platforma Arduino používá dvě hlavní funkce z nich je setup(). Ta je volána při na začátku běhu programu. Následně pak funkce loop(). Ta, jak již název naznačuje běží v nekonečné smyčce a obsahuje hlavní část programu.

[8] Problematika ESPresence a Garmin zařízení: https://community.home-assistant.io/t/smartwatch-detection/316676

```
ConnectionService connectionService;
DHT dht(DHTPIN, DHTTYPE);
MQ9 mq9(MQ9PIN, MQ9R0);


void setup() {
  Serial.begin(9600);
  connectionService.connect();
  pinMode (LED_BUILTIN, OUTPUT);
}
```

Obrázek 24: implementace setup funkce v programu pro ESP32 [zdroj: autor]

Pro komunikaci se sensorem DHT11 je použita již existující knihovna[9]. Nebylo tedy nutné vyvíjet komponentu, která by se sensorem komunikovala. Oproti tomu za účelem měření pomocí sensoru MQ-9 byla vytvořena samostatná třída. Ta ke své inicializaci potřebuje pin, přes který načítá data ze sensoru a konstantu R0. Tato konstanta se používá ke zpřesnění výsledků měření. MQ-9 jsou sensory s poměrně velkou odchylkou proto se zpravidla před jejich použitím sensor kalibruje. Výsledkem této kalibrace je tato konstanta, jejímž účelem je minimalizovat odchylku daného sensoru. Pro nalezení konstanty byl použit kód[10] který na svých stránkách uvádí prodejce sensoru.

Samotné měření pak probíhá načtením hodnoty ze sensoru. Pro snížení odchylek proběhne sto měření během jedné vteřiny, přičemž výsledná hodnota je průměrem těchto měření. Tato hodnota je potom přepočtena na výsledné napětí (výsledkem měření je hodnota od 0 do 1024, přičemž 0 reprezentuje 0V a 1024 reprezentuje 3.3V). Na základě napětí je pak vypočten odpor sensoru a na jeho základě je pomocí konstanty R0 aproximována hodnota ppm (parts per milion) pro CO_2.

```
void MQ9::updateMesurements(){
  sensor_value = 0;
  for (int x = 0 ; x < 100 ; x++) {
    sensor_value = sensor_value + analogRead(pin);
    delay(10);
  }
  sensor_value = sensor_value / 100.0;
  measured_volt = sensor_value / 1024 * 3.3;
  RS_gas = (5.0 - measured_volt) / measured_volt;
  ratio = RS_gas / R0;
  if ( ratio < 2 ) {
    // přepočet načtených dat, z rozsahu 0-700
    // na koncentraci 200-10000
    ppm = map(ratio*100, 150, 79, 200, 1000);
  }else{
    ppm=0;
  }
};
```

V hlavní smyčce programu je potom volaná funkce, která se stará o připojení k WiFi a MQTT brokerovy. Funkce zkontroluje, jestli jsou obě připojení aktivní. Pokud tomu tak není snaží se vytvořit nové spojení. Následně rozsvítí diodu, která indikuje, že dochází

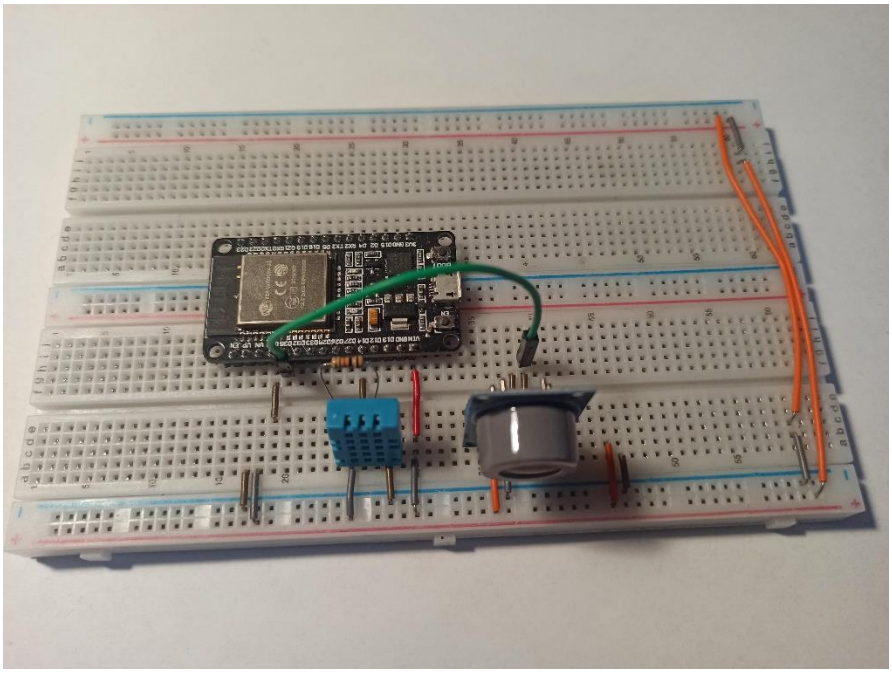
[9] https://github.com/adafruit/DHT-sensor-library

[10] https://navody.dratek.cz/navody-k-produktum/senzor-oxidu-uhelnateho-mq-9.html

k měření, provede měření a výsledná data publikuje do příslušných MQTT topiků. Následně již jen čeká šedesát vteřin a pote se celá funkce spouští znovu.

```
void loop() {
  //pripojení na WIFI a MQTT brokera
  connectionService.connect();
  //led on
  digitalWrite(LED_BUILTIN, HIGH);
  delay(300);

  //update sensor mesurements
  mq9.updateMesurements();

  //publish data
  connectionService.publishData("/jhlubucek/sensor/temp", String(dht.readTemperature()).c_str());
  connectionService.publishData("/jhlubucek/sensor/co2", String(mq9.getPpm()).c_str());
  connectionService.publishData("/jhlubucek/sensor/volt", String(mq9.getSensorVolt()).c_str());
  connectionService.publishData("/jhlubucek/sensor/value", String(mq9.getSensorValue()).c_str());
  connectionService.publishData("/jhlubucek/sensor/ratio", String(mq9.getRatio()).c_str());

  //led off
  digitalWrite(LED_BUILTIN, LOW);
  delay(60000);
}
```

Obrázek 25: implementace loop funkce v programu pro ESP32 [zdroj: autor]



Obrázek 26: ESP32 se sensory

6.3 Webová aplikace

Webová aplikace je psaná v jazyce Java za pomoci frameworku Spring. Pro tvorbu webové stránky byl použít návrhový vzor MVC. Kromě Modelu, view a kontroleru pak aplikace obsahuje ještě další část, kterou je komponenta sloužící k napojení na MQTT brokera, který podle dat uložených v databází odebírá zprávy z MQTT brokera a ukládá je do databáze.

Databázová vrstva (či v rámci MVC modelu modelová vrstva) aplikace je řešená pomocí DTO (data transfer object) tříd. Tyto třídy reprezentují jednotlivé tabulky v databázi. Pro vytváření dotazů je využita knihovna JDBC, ta přináší abstrakci nad konkrétním zdrojem dat a umožnuje vytváření dotazu a funkcí bez nutnosti znát konkrétní zdroj dat (například druh a verzi databáze). JDBC knihovna pomocí vrací list DTO objektů. Kdy každý objekt reprezentuje jeden řádek odpovědi.

```
public List<Light> getAllLightsFromDashboard(int dashboardId){
    List<Light> lights = jdbcTemplate.query(
            sql: "SELECT * FROM light WHERE dashboard_id = ?",
            lightRowMapper,
            dashboardId);
    return lights;
}
```

Obrázek 27: Funkce v datové vrstvě pro vrácení všech existujících světel [zdroj: autor]

View vrstva, tedy grafický výstup pro uživatele je řešena pomocí knihovny Thymeleaf. Thymeleaf je šablonovací systém pro vývoj webových aplikací s využitím frameworku Spring. Jeho hlavním úkolem je umožnit snadné propojení dat z backendové části aplikace s uživatelským rozhraním v HTML formátu. Je založený na generovaní HTML stránek na základě šablon, které jsou naplněny daty z databáze

```
<div class="container parent">
  <div th:each="sensor : ${sensors}" class="temperature-graph child">
    <div th:id="'sensor-canvas-' + ${sensor.id}" ></div>
  </div>
</div>
```

Obrázek 28: ukázka thymeleaf šablony generující prvky s daty sensorů [zdroj: autor]

Poslední vrstvu na základě MVC vzoru představuje controller. Ten spojuje model a view do jednoho celku. Jednotlivé endpointy které aplikace poskytuje jsou zprostředkovány právě pomocí controlleru. Funkce, které jsou použity ke zpracování http dotazů v anotaci obsahují informaci, pro který endpoint mají být volány. Následně funkce pomocí modelu připraví potřebná data ke zpracování dotazu. Tato data předá view vrstvě a vrátí výsledný view (obvykle HTML dokument). Také se zde nachází několik endpointů, které nevrací HTML odpověd ale pouze plaintext. Jedná se o endpointy které zpracovávají POST a DELETE dotazy. Jejich účel je obvykle zpracování HTML formuláře nebo smazání či upravení nějaké entity (například smazání světla či upravení jeho jasu)

Poslední důležitou částí je pak napojení aplikace na MQTT brokera. Pro komunikaci pře MQTT byla zvolena knihovna Paho eclipse a byla vytvořena servise, která umožnuje jak publikování zpráv, tak odebírání jednotlivých topiků. V rámci odebírání konkrétních topiku je třeba dopředu definovat, jak bude zpracovaná příchozí správa pro daný topik. Proto jsou vytvořeny zvlášť funkce pro odebírání druhu topiku a jeho zpracování. Například pro data ze sensorů je třeba uchovávat historii, zatímco u dat ze světla je zapotřebí uchovávat pouze poslední hodnotu. Aplikace pak periodicky kontroluje objekty vytvořené v databází a přihlašuje se k odebírání všech potřebných topičů. Zároveň je MQTT využito i v rámci kontrolerů, kdy například endpoint pro rozsvícení světla odešle zprávu do daného topiku s informací, že se světlo má rozsvítit.

6.3.1 Dockerizace

```
FROM maven:3.8.3-openjdk-17 AS builder
WORKDIR /home/app
COPY . .

RUN mvn clean package -B

FROM arm64v8/openjdk:20-ea-17-jdk as production

MAINTAINER jan hlubucek


#Add Spring Boot app.jar to Container
COPY --from=builder /home/app/target/smartControls.jar app.jar
RUN echo $(ls -1 /tmp)

# Fire up our Spring Boot app by default
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -jar /app.jar" ]
```

Obrázek 29: dockerfile soubor webové aplikace [zdroj: autor]

6.4 Automatizace

Cílem této kapitoly je na praktických příkladech popsat, jakým způsobem byly v rámci chytré domácnosti vytvořeny automatizace. Pro tvorbu automatizací byly využity jak automatizace Home Assistant tak automatizace vytvořené pomocí služby Node-RED.

6.4.1 Automatizace pomocí Home Assistant

V rámci Home Assistanta lze vytvářet automatizace. Tvorba probíhá pomocí sterilizačního jazyk a YAML. Alternativně lze pak použít i grafické prostředí pro tvorbu automatizace. Automatizace v rámci Home Assistant mají pevně danou strukturu. První část je trigger tedy spouštěcí akce, může se jednat o akci nějakého zařízení (například stisk tlačítka), příchozí zprávu (např. MQTT), specifický čas a mnoho dalších. Po spouštěcí akci následuje podmínka, ta není povinná a jedná se o nějakou podmínku která musí být splněna, aby proběhla akce automatizace. Mezi podmínky obvykle patří nějaký stav, například teplota je vyšší než 21 °C nebo již zapadlo slunce. Podmínka se dají libovolně kombinovat, například pomocí logických spojek jako je AND a OR. Posledním prvkem automatice je samotná akce, zde se muže být provedena celá škála činností, jako je rozsvícení/zhasnutí světla, odeslání zprávy. Tyto činnosti lze mezi sebou různě

kombinovat, lze je spouštět sekvenčně nebo paralelně (najednou) či spouště specifické akce na základě podmínky if.

```yaml
alias: switch_2_automation
description: ""
trigger:
  - device_id: ab169edefefef296d1e9956bf71ccaa4
    domain: zha
    platform: device
    type: remote_button_short_press
    subtype: turn_off
    id: "1"
  - device_id: ab169edefefef296d1e9956bf71ccaa4
    domain: zha
    platform: device
    type: remote_button_short_press
    subtype: turn_on
    id: "2"
condition: []
action:
  - if:
      - condition: trigger
        id: "1"
    then:
      - if:
          - condition: state
            entity_id: light.bedroom
            state: "on"
        then:
          - service: light.turn_off
            data: {}
            target:
              entity_id: light.bedroom
        else:
          - service: light.turn_on
            data:
              brightness_pct: 45
            target:
              entity_id: light.bedroom
    else: []
  - if:
      - condition: trigger
        id: "2"
    then:
      - service: light.turn_on
        data:
          brightness_pct: 90
        target:
          entity_id: light.bedroom
mode: single

```

Obrázek 30: ukázka automatizace v YAML [zdroj: autor]

Uvedený příklad je reprezentací jednoduché automatizace pro interakci mezi spínačem a žárovkou (respektive skupinou žárovek). První část definuje spouštěče (trigger), jedná se o dvě možné akce, stisknutí on nebo off tlačítka na spínači. Obě těmto akcím je přiřazeno id, aby je bylo možné později odlišit. Podmínky jsou v tomto případě prázdné a

automatizace se spustí vždy při spouštěcí akci. Následuje podmínka if, která rozlišuje, zda bylo stisknuto tlačítko off nebo on. Pokud bylo stisknuto tlačítko off následuje další podmínka, která zkontroluje jestli jsou světla zapnuta. Pokud ano tak se vypnou, v opačném případě se zapnou na padesáti procentní jas. To umožňuje uživateli velmi jednoduše nastavit světla na poloviční jas. Pokud byla stisknuto tlačítko on světla se zapnou na devadesáti procentní jas. Automatizace tedy umožnuje zapnutí a vynutí světel, popřípadě jejich nastavení na poloviční jas (stisknutím tlačítka off ve chvíli, kdy jsou světla vypnuta).

6.4.2 Automatizace pomoci Node-RED

Pro pokročilejší automatizace byl použit nástroj Node-RED. Jeho výhodou je, že na rozdíl od Home Assistant nemusí automatizace dodržovat předem danou strukturu. Nástroj je založen na tzv. "flow-based programming" přístupu. To znamená, že automatizace se v Node-REDu skládají z jednotlivých bloků, nazývaných uzly (nodes), které jsou propojeny pomocí "cest" (wires) a tvoří tak celkový "flow" (tok). Každý uzel má vstup, výstup, nebo obojí. Jednotlivé cesty potom představují tok zpráv z výstupu jednoho uzlu na vstup druhého.

Počáteční uzly mají definovaný nějaký spouštěč, například příchozí práva nebo dotaz (MQTT, AMQP, HTTP atd.) pokud je na jejich výstup připojen nějaký uzel, pošlo při této události na připojená uzel zprávu. Pokud mají uzly vstup i výstup, tak při obdržení zprávy provedou jejich akci a poté pošlou zprávu na další napojený uzel (pokud takový existuje). Některé uzli pak mají pouze vstup a nelze na ně připojit další uzly.

Existuje mnoho různých uzlů pro různé účely, ať už se jedná o uzly pro zpracování dat, komunikaci s jinými systémy nebo ovládání hardwaru. Tyto uzly jsou k dispozici ve výchozí instalaci Node-REDu, ale existují i další, které si lze nainstalovat pomocí rozšíření (tzv. "nodes modules").

Následující příklad popisuje automatizaci světel. Cílem této automatizace je automaticky rozsvěcet a zhasínat světla na základě, sensoru pohybu, sensoru otevřených dveří a okolních Bluetooth zařízení. Zároveň chceme, aby se automatizace spouštěla pouze v určitém čase a šlo ji jednoduše vypnout. Zároveň pokud bude docházet k automatickému zhasnutí světel. Měl by na to být uživatel upozorněn (například ztlumením jasu světla, než dojde k úplnému zhasnutí), abychom předešli náhlému zhasnutí v případě, kdy není detekován pohyb, ačkoliv v místnosti někdo je.

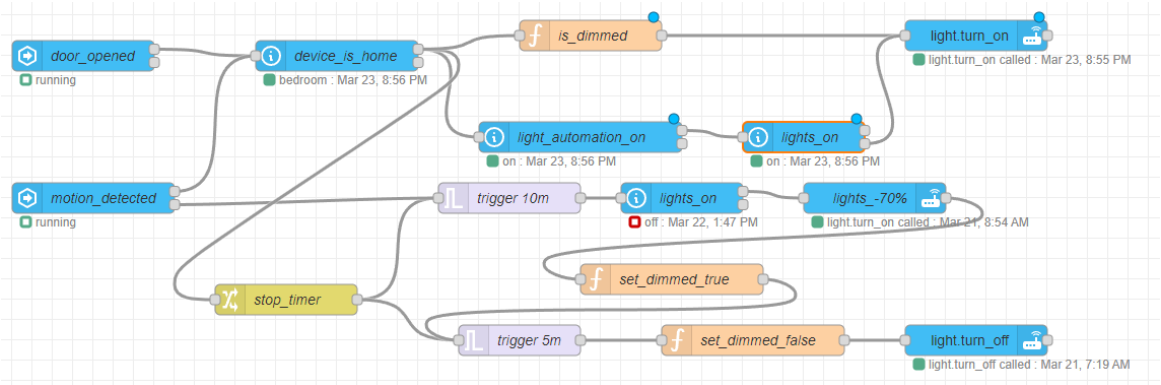
Pro splnění tohoto cíle je tedy zapotřebí rozeznat kdy v pokoji někdo je a naopak, kdy v pokoji již nikdo není. Za detekci přítomnosti je považováno otevření dveří nebo detekování pohybu. Za tímto účelem jsou použity dva vstupní uzly, které sledují změnu stavu sensoru pohybu a sensoru dveří. Pokud je detekována přítomnost, následující uzel zkontrolujeme že je v dosahu předem definované zařízení. Uzel, který je k tomu použit pošle zkontroluje stav Bluetooth zařízení v Home Assistant a zprávu pošle dál pouze v případě, že je zařízení v dosahu. V této části je již tedy jasné, že v místnosti byla detekována přítomnost a zároveň je detekováno Bluetooth zařízení a lze tedy předpokládat že je přítomen majitel tohoto zařízení. Následně je potřeba zjistit, jestli se má automatizace v daném čase spustit.

Za tímto účelem byla v Home Assistant vytvořená proměnná „light_automation“, jejíž stav je binární, on nebo off. Stav této proměnné udržuje jednoduchá automatizace v Home Assistant, která ji zapne při západu slunce a v deset hodin večer ji zase vypne. Alternativně ji pak lze manuálně vypnout či zapnout. Další uzel tedy zkontroluje její stav a správu odešle dál pouze v případě, že je ve stavu on.

Následuje ještě uzel, který zkontroluje že jsou světla zhasnutá, abychom neměnily nastavený jas, pokud jsou již zapnutá. Pokud jsou světla zhasnutá následující uzel je rozsvítí na předem nastavený jas.

Tato část automatizace řeší automatické rozsvícení světel. Další část řeší jejich zhasnutí. Pro detekci nepřítomnosti nelze použít dveře, protože nelze určit, jestli osoba vstoupila anebo naopak místnost opustila. Jak vstup je, tak použit pouze signál sensoru pohybu, značící, že již není detekován pohyb. Není chtěné, aby světla byla zhasnuta okamžitě jelikož muže dojít k situaci kdy po nějakou dobu není detekován pohyb (například protože uživatel sedí a příliš se nehýbe, aniž by opustil místnost.) Při detekci pohybu je tak spuštěn uzel, který spustí desetiminutový odpočet a teprve po jeho uplynutí pošle zprávu dál. Následně se zkontroluje, jestli jsou světla rozsvícena. Pokud tomu tak je dojde ke ztlumení světel, aby byl uživatel upozorněn, že již není detekován pohyb a dojde k jejich vypnutí. Pomocí nodu function si uložíme proměnou s informací, že došlo ke ztlumení světel a je spuštěn další uzel s odpočtem. Tentokrát na pět minut. Po uplynutí tohoto odpočtu si pomocí dalšího uzlu function přepíšeme hodnotu proměnné, jelikož světla již nejsou ztlumena. Poslední uzel pak světla vypne.

Tím je vyřešeno rozsvěcení a vypínání světel. Poslední část automatizace má za úkol, vypnutí prvního časovačů, pokud bude za jejich běhu detekován pohyb a případné opětovné rozsvícení světel, pokud došlo k jejich ztlumení. Za tímto účelem je při detekci pohybu provolán i uzel pojmenovaný „stop_timer“. Tento uzel odešle zprávu s předem definovanou hodnotou do obou uzlu s časovačem. Ty jsou nastaveny tak aby tato zprava oba časovače vypla v případě, že běží. Tíme je efektivně zastaven proces vypínání světel. Spolu s uzlem pro zastavení časovače je pak provolán také další uzel function, jehož účelem je zkontrolovat jestli je v proměnné uložena informace, že světla jsou ztlumena. Pokud tomu tak je pošle zprávu do následujícího uzlu, který ztlumená světla opět rozsvítí.



Obrázek 31: automatizace v Node-Red rozsvěcení a zhasínání světel [zdroj: autor]

7 Výsledky

7.1 Hodnocení vytvořeného systému

Cílem práce byl návrh a následné vytvoření systému pro chytrou domácnosti. Dílčím cílem potom byl návrh a realizace webové aplikace pro ovládání prvků výsledné chytré domácnosti. Pro vytvoření systému byly použity služby Home Assistant, Node-RED, Mosquitto Broker a portainer. Webová aplikace je potom vytvořena v jazyce Java a založena na frameworku Spring Boot. Pro vytvoření frontendové časti byl použit návrhový vzor MVC a šablonovací engine Thymeleaf.

Výsledkem práce je funkční systém pro chytrou domácnost a jeho návrh. Výsledný systém splňuje jak funkční tak nefunkční požadavky.

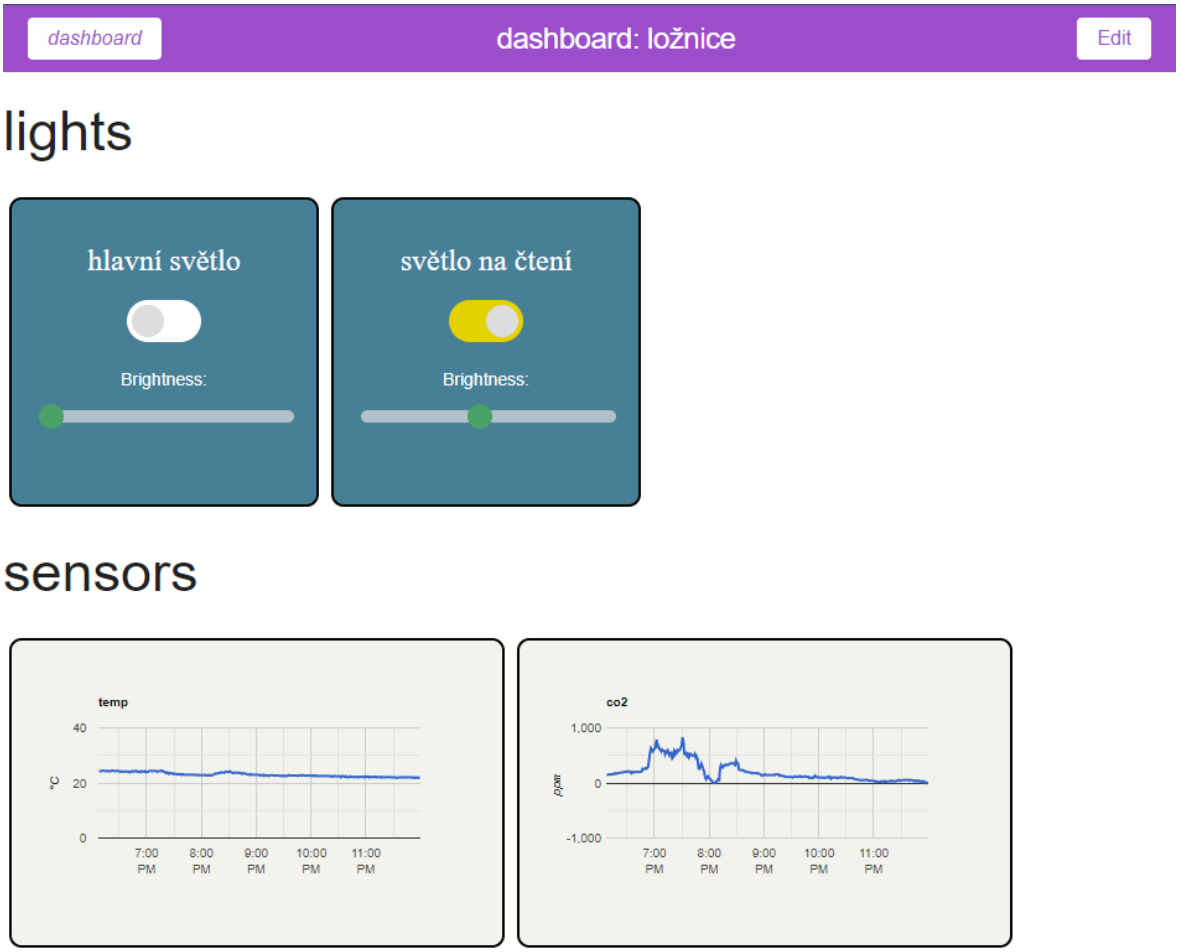
7.2 Klady

Systém chytré domácnosti navržený v textu má několik předností. Zaprvé, centrálním bodem systému je software Home Assistant, který umožňuje integraci všech chytrých zařízení a poskytuje uživatelsky přívětivé rozhraní pro správu a monitorování těchto zařízení. Dále, systém podporuje širokou škálu zařízení a služeb a umožňuje vytváření vlastních automatizací a skriptů pro ovládání zařízení a reakci na události. Kromě toho, systém je navržen jako modulární a škálovatelný, což umožňuje snadné přidávání nových zařízení a služeb podle potřeby. Použití kontejnerů Docker umožňuje efektivní využití zdrojů a snadné nasazení různých komponent. Celkově je systém navržen jako flexibilní a adaptabilní řešení pro správu a automatizaci chytrých zařízení v domácím prostředí. Výhodou webové aplikace pak je že umožnuje průběžně přidávat další světla či sensory. Díky tomu bude možné přidávat do chytré domácnosti další zařízení bez nutnosti úpravy kódu aplikace.

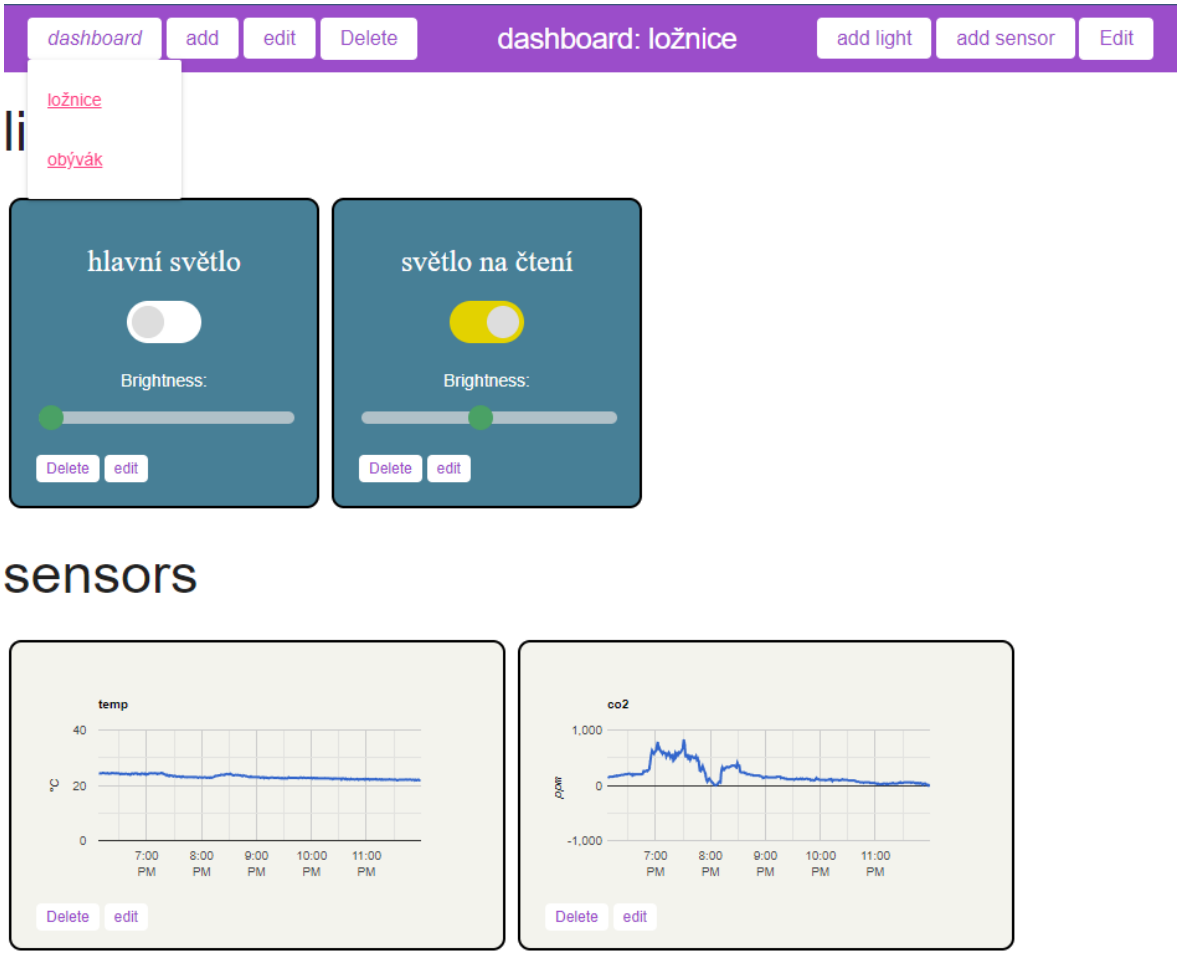
7.3 Zápory

Mezi nedostatky tohoto systému lze zařadit zpoždění reakce při ovládání skrze webovou aplikaci, což je pomalejší než u klasického spínače. Dále nelze pomocí vypínačů nastavovat jas světel granulárně, lze použít pouze předem definované hodnoty v rámci některých automatizací. Pro nastavení specifického jasu je tak potřeba použít webovou aplikaci nebo webové rozhraní Home Assistant. Dalším problémem je pak absence nastavení barvy či teploty konkrétního světla ve webové aplikaci. Chytrá domácnost neobsahuje světla s touto možností. Pokud boudou v budoucnosti do chytré domácnosti integrovány světla s touto možností, bude vhodné tuto možnost do aplikace přidat.

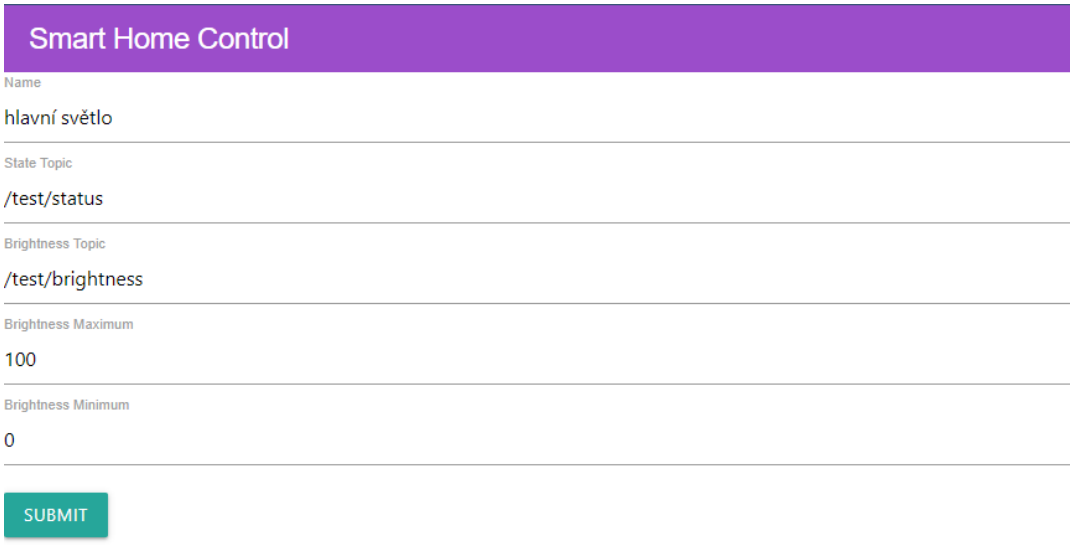
7.4 Podoba webového rozhraní



Obrázek 32: dashboard ve webové aplikace [zdroj: autor]



Obrázek 33: editace dashboardu ve webové aplikaci [zdroj: autor]



Obrázek 34: vytváření/editace prvku k ovládání světla

8 Závěr

Cílem této diplomové práce bylo navrhnout a vytvořit chytrou domácnost s využitím mikropočítače, mikrokontrolerů a dalších komponent, včetně webového rozhraní a automatizací. Práce byla rozdělena do několika částí, které zahrnovaly analýzu a srovnání možných řešení a technologií pro chytrou domácnost, návrh chytré domácnosti a webové aplikace pro její ovládání, implementaci reálné chytré domácnosti s automatizacemi a nakonfigurováním, a nakonec závěr, který shrnul všechny výsledky a zhodnotil splnění cílů práce.

V úvodní části práce byla provedena analýza možných technických řešení pro chytrou domácnost, včetně software a služeb, které lze použít, a protokolů pro komunikaci mezi zařízeními. Tyto znalosti byly poté použity pro vytvoření návrhu chytré domácnosti a webové aplikace, která umožňuje ovládání prvků domácnosti.

V práci byl navržen seznam požadavků pro chytrou domácnost a vytvořena webová aplikace, která umožňuje ovládání prvků domácnosti. Následně byl vytvořen návrh chytré domácnosti a všech jejích částí tak, aby byly splněny veškeré funkční i nefunkční požadavky. Následně byla podle tohoto návrhu vytvořena reálná chytrá domácnost. Byly také vytvořeny jednotlivé automatizace, které usnadňují používání chytré domácnosti, a webová aplikace pro její ovládání.

Lze tedy konstatovat že byl splněn jak hlavní cíl pro vytvoření chytré domácnosti. Stejně tak lze konstatovat i splnění dílčích cílů: vytvoření webové aplikace pro ovládání domácnosti, analýza možných řešení a technologií a přiblížení vytváření automatizací v chytré domácnosti.

Na výsledném řešení by se dalo stále upravovat. Již v kapitole 7. výsledky je zmíněno několik nevýhod ovládací aplikace, které by bolo dobré v případné další verzi aplikace odstranit. Zároveň nutno podotknout že se příliš nezaobírá kybernetickou bezpečností této domácnosti, ačkoliv bezpečnost je častým tématem právě v kombinaci s IoT nebo chytrými domácnostmi. Řešení je navržena k provozu pouze po LAN což by vyžadovalo, aby byl uživatel fyzicky přítomný v domácnosti a byl schopen se připojit do sítě. Alternativně pak pomocí útoku na Wi-Fi síť dané domácnosti. Pokud by k tomu došlo útočným by měl přístup k potenciálně citlivým datům o chodu uvnitř domácnosti.

9 Seznam použitých zdrojů

[1] A. C.- info@aion.cz, “40/1964 Sb. Občanský zákoník (1964-2013),” *Zákony pro lidi*. https://www.zakonyprolidi.cz/cs/1964-40 (accessed Mar. 31, 2023).

[2] itbiz, “Domácnost,” *ITBiz.cz*, Sep. 13, 2011. https://www.itbiz.cz/slovnik/ekonomie/domacnost/ (accessed Mar. 31, 2023).

[3] “Co je Smart Home? Domov podle vašich potřeb – smarteon.cz.” https://smarteon.cz/co-je-smart-home/ (accessed Mar. 31, 2023).

[4] “Inteligentní budovy.” http://www.inteligentni-budovy.cz/ (accessed Mar. 31, 2023).

[5] K. DAVID, “Bezpečnost Bluetooth.” Dec. 18, 2011. [Online]. Available: https://dsn.felk.cvut.cz/wiki/_media/vyuka/cviceni/x36mti/prezentace2007/dolezr1-doc.pdf

[6] “Bluetooth® Technology Website – The official website for the Bluetooth wireless technology. Get up to date specifications, news, and development info.” https://www.bluetooth.com/ (accessed Mar. 15, 2023).

[7] “Bluetooth 1.0 vs 2.0 vs 3.0 vs 4.0 vs 5.0 - How They Compare | Symmetry Blog,” *Symmetry Electronics*. https://www.symmetryelectronics.com/blog/bluetooth-1-0-vs-2-0-vs-3-0-vs-4-0-vs-5-0-how-they-compare-symmetry-blog/ (accessed Mar. 15, 2023).

[8] “Bezdrátové sítě (WiFi, Bluetooth, ZigBee) a možnosti jejich implementace. - PDF Stažení zdarma.” http://docplayer.cz/12704159-Bezdratove-site-wifi-bluetooth-zigbee-a-moznosti-jejich-implementace.html (accessed Mar. 15, 2023).

[9] “What Is Wi-Fi? - Definition and Types,” *Cisco*. https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html (accessed Mar. 15, 2023).

[10] “802.11 WiFi Standards Explained,” *Lifewire*. https://www.lifewire.com/wireless-standards-802-11a-802-11b-g-n-and-802-11ac-816553 (accessed Mar. 31, 2023).

[11] “ZigBee | Computerworld.” https://www.computerworld.com/article/2554497/zigbee.html?page=2 (accessed Mar. 15, 2023).

[12] Connectivity Standart Aliance, “ZigBee Specifications,” *CSA-IOT*. https://csa-iot.org/developer-resource/specifications-download-request/ (accessed Mar. 15, 2023).

[13] M. Macharla, “ZigBee and its Importance in the Internet of Things,” *IoTEDU*, Sep. 28, 2020. https://iot4beginners.com/zigbee-and-its-importance-in-the-internet-of-things/ (accessed Mar. 31, 2023).

[14] P. Carlos, “Z-Wave Explained: What It Is and How It Works - DIY Smart Home Solutions.” https://www.diysmarthomesolutions.com/z-wave-wireless-networking/ (accessed Mar. 15, 2023).

[15] S. D. Silicon Labs Inc., “Silicon Labs Completes Acquisition of Sigma Designs’ Z-Wave Business,” *Silicon Labs Newsroom*. https://news.silabs.com/2018-04-18-Silicon-Labs-Completes-Acquisition-of-Sigma-Designs-Z-Wave-Business (accessed Mar. 15, 2023).

[16] S. Sharon, “What is Z-Wave? | Definition from TechTarget,” *IoT Agenda*. https://www.techtarget.com/iotagenda/definition/Z-Wave (accessed Mar. 15, 2023).

[17] Z-Wave Alliance, “Learn about Z-Wave,” *Z-Wave*. https://www.z-wave.com/learn (accessed Mar. 15, 2023).

[18] “Z-Wave Wireless Technology | IHomeFuture,” Jan. 01, 2022. https://ihomefuture.com/z-wave-wireless-technology/ (accessed Mar. 31, 2023).

[19] M. Malý, “Protokol MQTT: komunikační standard pro IoT,” *Root.cz*. https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/ (accessed Mar. 15, 2023).

[20] iPC2U s.r.o., “Co je MQTT a k čemu slouží ve IIoT? Popis protokolu MQTT,” *iPC2U s.r.o.*, 04:23 100AD. https://ipc2u.cz/blogs/news/mqtt-protokol (accessed Mar. 15, 2023).

[21] The HiveMQ, “Publish & Subscribe - MQTT Essentials: Part 2.” https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/ (accessed Mar. 15, 2023).

[22] The HiveMQ, “Quality of Service (QoS) 0,1, & 2 MQTT Essentials: Part 6.” https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/ (accessed Mar. 15, 2023).

[23] “MQTT architecture, features and implementation with bOS! - ComfortClick.” https://www.comfortclick.com/News/Show/93 (accessed Mar. 31, 2023).

[24] The HiveMQ, “MQTT Topics, Wildcards, & Best Practices - MQTT Essentials: Part 5.” https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/ (accessed Mar. 15, 2023).

[25] L. Johansson, “FAQ: What is AMQP and why is it used in RabbitMQ?,” *CloudAMQP*, Nov. 21, 2019. https://www.cloudamqp.com/blog/what-is-amqp-and-why-is-it-used-in-rabbitmq.html (accessed Mar. 15, 2023).

[26] SmartBear Software, “AMQP Testing | ReadyAPI Documentation.” https://support.smartbear.com/readyapi/docs/testing/amqp.html (accessed Mar. 15, 2023).

[27] VMware, Inc, “AMQP 0-9-1 Model Explained — RabbitMQ.” https://www.rabbitmq.com/tutorials/amqp-concepts.html (accessed Mar. 15, 2023).

[28] N. Engineering, “RabbitMQ: Messaging that just works,” *Naukri Engineering*, Oct. 12, 2017. https://medium.com/naukri-engineering/naukriengineering-rabbitmq-messaging-that-just-works-dafcd0cfd194 (accessed Mar. 31, 2023).

[29] Cloudflare, Inc., “What is HTTP?,” *Cloudflare*. https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/ (accessed Mar. 15, 2023).

[30] Mozilla Foundation, “Evolution of HTTP - HTTP | MDN,” Mar. 03, 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP (accessed Mar. 15, 2023).

[31] IBM Corporation, “HTTP requests,” Oct. 13, 2021. https://www.ibm.com/docs/en/cics-ts/5.3?topic=protocol-http-requests (accessed Mar. 15, 2023).

[32] Refsnes Data, “HTTP Methods GET vs POST.” https://www.w3schools.com/tags/ref_httpmethods.asp (accessed Mar. 15, 2023).

[33] “Understand The game of HTTP Request and Response,” Dec. 23, 2022. https://bugbountyguide.org/2022/12/23/understand-the-game-of-http-request-and-response/ (accessed Mar. 31, 2023).

[34] B. Eric, “Introducing the Constrained Application Protocol (CoAP) for Java.” https://blogs.oracle.com/javamagazine/post/java-coap-constrained-application-protocol-introduction (accessed Mar. 15, 2023).

[35] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” Internet Engineering Task Force, Request for Comments RFC 7252, Jun. 2014. doi: 10.17487/RFC7252.

[36] R. Crist, “The complete guide to Philips Hue,” *CNET*, Jun. 03, 2020. https://www.cnet.com/home/energy-and-utilities/the-complete-guide-to-philips-hue/ (accessed Mar. 15, 2023).

[37] C. services s.r.o, “Philips Hue White Ambiance 8W E27 3-set starter kit,” *Voltio.cz*. https://www.voltio.cz/en/starter-pack/585-philips-hue-white-ambiance-8w-e27-3-set-starter-kit-8719514291232.html (accessed Mar. 31, 2023).

[38] TESLA Solar, s.r.o., “TESLA Smart Thermostatic Valve Style,” *TESLA*. https://www.teslasmart.com/tesla-smart-thermostatic-valve-style (accessed Mar. 15, 2023).

[39] TESLA Solar, s.r.o., “TESLA Smart Sensor Water,” *TESLA*. https://www.teslasmart.com/tesla-smart-sensor-water (accessed Mar. 15, 2023).

[40] “Aqara Water Leak Sensor - Xiaomi Store.” https://xiaomi-store.cz/en/smart-home/2038-aqara-water-leak-sensor-6970504212534.html (accessed Mar. 31, 2023).

[41] Tuya Inc., “Smart Lock Solutions | Tuya’s smart lock solutions guard your home and enhance your smart environment | Tuya Smart.” https://www.tuya.com/solution/hardware/smart-lock (accessed Mar. 15, 2023).

[42] B. Mona, “AI Faceoff: Siri vs. Cortana vs. Google Assistant vs. Alexa - businessnewsdaily.com,” *Business News Daily*. https://www.businessnewsdaily.com/10315-siri-cortana-google-assistant-amazon-alexa-face-off.html (accessed Mar. 15, 2023).

[43] M. cz a.s, “Google Nest Mini 2.gen. černá,” *Mironet.cz*. https://www.mironet.cz/google-nest-mini-2gen-cerna-smart-reproduktor-bluetooth-wlan-360-zvuk-google-asistatn+dp462342/ (accessed Mar. 31, 2023).

[44] Residential IoT Services GmbH, “Smart home appliances: let your home work for you.” https://www.home-connect-plus.com/global/en/works-with/smart-home-appliances/ (accessed Mar. 15, 2023).

[45] “Here’s how to add Ikea Trådfri lights to the Philips Hue Bridge hub,” *Android Authority*, Mar. 22, 2023. https://www.androidauthority.com/add-ikea-tradfri-lights-philips-hue-1211149/ (accessed Mar. 29, 2023).

[46] A. Brice, “Best of open source smart home: Home Assistant vs OpenHAB,” *Smart Home University*, Feb. 28, 2018. https://smarthome.university/your-smart-home-platform-home-assistant-vs-openhab/ (accessed Mar. 28, 2023).

[47] “OpenHAB,” *Moderní domek*. https://chytrydumsvepomoci.cz/blog/openhab (accessed Mar. 28, 2023).

[48] A. Brice, “Best of open source smart home: Home Assistant vs OpenHAB,” *Smart Home University*, Feb. 28, 2018. https://smarthome.university/your-smart-home-platform-home-assistant-vs-openhab/ (accessed Mar. 28, 2023).

[49] “Docker.” https://www.openhab.org/docs/installation/docker.html (accessed Mar. 29, 2023).

[50] “Home Assistant: Learn how to Automize your Home,” *DEV Community*, Mar. 09, 2023. https://dev.to/admantium/home-assistant-learn-how-to-automize-your-home-15a3 (accessed Mar. 28, 2023).

[51] H. Assistant, “Configuration.yaml,” *Home Assistant*. https://www.home-assistant.io/docs/configuration/ (accessed Mar. 28, 2023).

[52] “Home Assistant & Raspberry Pi 4: How to Get Started | All3DP.” https://all3dp.com/2/home-assistant-raspberry-pi-tutorial/ (accessed Mar. 28, 2023).

[53] H. Assistant, “Installation,” *Home Assistant*. https://www.home-assistant.io/installation/ (accessed Mar. 29, 2023).

[54] J. P. Tuohy, “Samsung’s new SmartThings Station is a do-it-all smart home hub,” *The Verge*, Feb. 05, 2023. https://www.theverge.com/23584507/samsung-smartthings-station-review-smart-home-hub-matter-thread (accessed Mar. 29, 2023).

[55] “SmartThings Developer Documentation — SmartThings Documentation 1.0 documentation.” https://stdavedemo.readthedocs.io/en/latest/index.html (accessed Mar. 29, 2023).

[56] “SmartThings Řídící jednotka - Aeotec Smart Home Hub - Works as a SmartThings Hub - EU | MALL.CZ.” https://www.mall.cz/gadgety/smartthings-ridici-jednotka-aeotec-smart-100043428419 (accessed Mar. 29, 2023).

[57] F. home automation, “FIBARO | Home Automation - Smart Home,” *fibaro.com*. https://www.fibaro.com/en/ (accessed Mar. 29, 2023).

[58] Advertorial, “What is FIBARO and what devices make up the FIBARO smart home system?,” *CEPRO*, Mar. 13, 2018. https://www.cepro.com/networking/what_is_fibaro_and_what_devices_make_up_the_fibaro_smart_home_system/ (accessed Mar. 29, 2023).

[59] A. a.s, “FIBARO Home Center 3 - Centrální jednotka | Alza.cz,” *Alza*. https://www.alza.cz/fibaro-home-center-3-d5772165.htm (accessed Mar. 29, 2023).

[60] “What is Docker? | IBM.” https://www.ibm.com/topics/docker (accessed Mar. 31, 2023).

[61] M. Augustýn, “Proč používat Docker,” *Zdroják*, Apr. 11, 2017. https://zdrojak.cz/clanky/proc-pouzivat-docker/ (accessed Mar. 31, 2023).

[62] “Node-RED | Unipi.” https://www.unipi.technology/products/node-red-66 (accessed Mar. 31, 2023).

[63] “Node-RED.” https://nodered.org/ (accessed Mar. 31, 2023).

[64] “What is Java Spring Boot? | IBM.” https://www.ibm.com/topics/java-spring-boot (accessed Mar. 29, 2023).

[65] “Eclipse Mosquitto,” *Eclipse Mosquitto*, Jan. 08, 2018. https://mosquitto.org/ (accessed Mar. 31, 2023).

[66] “What is Eclipse Mosquitto (MQTT)?,” *Educative: Interactive Courses for Software Developers*. https://www.educative.io/answers/what-is-eclipse-mosquitto-mqtt?ranMID=47764&ranEAID=PPkX79%2Fc*b0&ranSiteID=PPkX79_c.b0-EKD0r31pGCrhvn79NO.PCw&aff=Vprj (accessed Mar. 31, 2023).

[67] “What is Grafana?” https://www.redhat.com/en/topics/data-services/what-is-grafana (accessed Mar. 31, 2023).

[68] “Thymeleaf.” https://www.thymeleaf.org/ (accessed Mar. 31, 2023).

[69] “MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN,” Feb. 21, 2023. https://developer.mozilla.org/en-US/docs/Glossary/MVC (accessed Mar. 31, 2023).

[70] “Better Presence Detection with Home Assistant and ESPresense,” *James Ridgway*, Aug. 09, 2022. https://www.jamesridgway.co.uk/better-presence-detection-with-home-assistant-and-espresense/ (accessed Mar. 31, 2023).

[71] “Raspberry PI – hračka s budoucností.” https://www.dps-az.cz/vyvoj/id:11987/raspberry-pi-hracka-s-budoucnosti (accessed Mar. 31, 2023).

[72] G. Halfacree, “Benchmarking the Raspberry Pi 4,” *Medium*, Nov. 22, 2019. https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b (accessed Mar. 31, 2023).

[73] R. Teja, “Getting Started with ESP32 | Introduction to ESP32,” *Electronics Hub*, Feb. 17, 2021. https://www.electronicshub.org/getting-started-with-esp32/ (accessed Mar. 31, 2023).

[74] “Začínáme s IKEA Home Smart | SIEGER.cz - Váš partner pro chytré elektro.” http://drbi.sieger.cz/zaciname-s-ikea-home-smart/, http://drbi.sieger.cz/zaciname-s-ikea-home-smart/ (accessed Mar. 31, 2023).

[75] “IKEA Joins Zigbee Alliance Board of Directors - News.” https://eepower.com/news/ikea-joins-zigbee-alliance-board-of-directors/ (accessed Mar. 31, 2023).

[76] C. services s.r.o, “SONOFF SNZB-03 - ZigBee pohybový senzor,” *Voltio.cz*. https://www.voltio.cz/cs/senzory-a-snimace/540-sonoff-snzb-03-zigbee-pohybovy-sensor-6920075776119.html (accessed Mar. 31, 2023).

[77] “SONOFF SNZB-04P Zigbee Door/Window Sensor User Manual - Manuals+.” https://manuals.plus/sonoff/snzb-04p-zigbee-doorwindow-sensor-manual#axzz7xYMl0RYk (accessed Mar. 31, 2023).

[78] “MQ-9 Gas Sensor Module for Carbon Monoxide, Methane and LPG,” *QuartzComponents*. https://quartzcomponents.com/products/mq-9-gas-sensor-module-for-carbon-monoxide-methane-and-lpg (accessed Mar. 31, 2023).

[79] U. Gajera, “Basics of Interfacing DHT11/DHT22 Humidity and Temperature Sensor with MCU,” *OCFreaks!*, Nov. 06, 2017. http://www.ocfreaks.com/basics-interfacing-dht11-dht22-humidity-temperature-sensor-mcu/ (accessed Mar. 31, 2023).

[80] M. Zolnierczyk, “Getting started with Sonoff ZIGBEE 3.0 USB DONGLE PLUS,” *NotEnoughTech*, Oct. 06, 2021. https://notenoughtech.com/home-automation/sonoff-zigbee-3-0-usb-dongle-plus/ (accessed Mar. 31, 2023).

[81] BUCHANAN, Marlon. The Smart Home Manual: How to Automate Your Home to Keep Your Family Entertained, Comfortable, and Safe. HomeTechHacker, 2020. ISBN: 978-1735543000

[82] KYAS, Othmar. How To Smart Home: A Step by Step Guide for Smart Homes & Building AutomaƟon. Key Concept Press, 2017. ISBN 978-3-944980-12-6.

[83] PRŮCHA, Jan. Chytré bydlení: Inteligentní dům http://www.insighthome.eu/Chytre-bydleni/index.html. (accessed Mar. 31, 2023)

[84] UPTON, Eben a Gareth HALFACREE. Raspberry Pi: uživatelská příručka. 2., aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4819-8

[85] VALEŠ, Miroslav. Inteligentní dům. Brno: ERA, 2006. ISBN 80-7366-062-8.

10 Seznam obrázků, tabulek, grafů a zkratek

10.1 Seznam obrázků

Přílohy

Příloha A: Zdrojové kódy

Zdrojové kódy jsou dostupné jako samostatná příloha této práce

Pro lepší dostupnost jsou také umístěny na stránce https://github.com/

zdrojové kódy webové aplikace: https://github.com/hluj00/smart-constrols

docker-compose soubory: https://github.com/hluj00/smart-home-docker-compose

Příloha B: Wireframes

Obrázek 35: Wireframe dashboard [zdroj: autor]

Obrázek 36: Wireframe přidaní prvku pro ovládání světla [zdroj: autor]

Obrázek 37: Wireframe přidaní prvku pro sensoru [zdroj: autor]