Implementace vlastního komplexního řídícího systému pro řízení rodinného domu

Bc. Lukáš Grolig

Diplomová práce
2015

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš Grolig**
Osobní číslo:	**A13479**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Implementace vlastního komplexního řídicího systému pro řízení rodinného domu**
Téma anglicky:	**An Implementation of a Comprehensive Control System for the Management of a Domestic House**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na téma "komplexní inteligentní řídicí systémy pro domácnost".**
2. **Proveďte analýzu typické domácnosti z hlediska možnosti řízení jednotlivých jejích součástí řídicím systémem (světla a jejich intenzita, vytápění, zabezpečení, apod.).**
3. **Navrhněte koncepci inteligentního řídicího systému – řídicí jednotka, jednotlivé zamýšlené moduly pro řízení i ovládání.**
4. **Realizujte celý řídicí systém hardwarově i softwarově.**
5. **Vytvořte manuál pro Vámi realizovaný systém.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Feynman, Richard P., Leighton, Robert B. a Sands, Mathew.Feynmanovy prednasky z fyziky : revidovane vydani s resenymi priklady. Praha : Fragment, 2013. Sv. 2. 978-80-253-1643-6.**
2. **Feynman, Richard P., Leighton, Robert B. a Sands, Mathew.. Feynmanovy prednasky z fyziky : revidovane vydani s resenymi priklady. Praha : Fragment, 2013. Sv. 1. 978-80-253-1642-9.**
3. **Scherz, Paul a Monk, Simon.Practical Electronics for Inventors, Third Edition. New York : McGraw-Hill, 2013. 978-0071771337.**
4. **Gasston, Peter.The Modern Web: Multi-Device Web Development with HTML5, CSS3, and JavaScript. San Francisco, CA : Na Starch Press, 2013. 978-1593274870.**
5. **Hintjens, Pieter.ZeroMQ: Messaging for Many Applications. S.l : O'Reilly Media, 2013. 978-1449334062.**
6. **Horvath, Joan.Mastering 3D Printing. New York : Apress, 2014. 978-1484200261.**
7. **Obe, Regina O. a Hsu, Leo S.PostgreSQL: Up and Running. 2. Sebastopol, CA : O'Reilly Media, 2015. 978-1449373191.**
8. **Wargo, John M.Apache Cordova 4 Programming (Mobile Programming). Upper Saddle River, NJ : Addison-Wesley, 2015. 978-0134048192.**

Vedoucí diplomové práce: **Ing. Tomáš Sysala, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **6. února 2015**

Termín odevzdání diplomové práce: **15. května 2015**

Ve Zlíně dne 6. února 2015

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

ABSTRAKT

Výpočetní technologie pronikají do všech oblastí života. Domácnost není výjimkou. Implementací inteligentního řízení a sledováním domu je možné ušetřit náklady na jeho provoz a zároveň zvýšit pohodlí uživatelů. Tato práce se zabývá stanovením požadavků, které musí systém splňovat, průzkumem dostupných řešení a základními aspekty implementace takového systému. Jsou diskutovány technické možnosti implementace, volba technologií a také základní moduly jako jsou řízení osvětlení, měření a ovládání teploty v místnostech, řízení okruhů vytápění, bezpečnostní a kamerový systém, a další funkce očekávané od takového systému. Mimo samotných řídích jednotek obsahuje systém i moderní grafickou ovládací konzoli, kterou je možné využívat na více platformách a zařízeních.

Klíčová slova: domácí automatizace, řízení osvětlení, řízení vytápění, měření teploty, bezpečnostní systém

ABSTRACT

Information technologies continue to get into all areas of life. Home is no exception. Implementing intelligent control and monitoring house is possible to save expenses for its upkeep and also increase comfort of the users. This work look into definition of aspects which are required from this type of the system. It discuss technical possibilities of implementation, choose of technologies and basic system modules like light control, temperature measurement and control, heating control, security and camera system, and other functions expected from this system. Beyond control units themselves system also contains modern graphic console which can used on multiple platforms and devices.

Keywords: home automation, light control, heating control, temperature measurement, security system

Děkuji Dr. Sysalovi, že se ujal dohledu nad touto diplomovou prací a nechal mi volnou ruku v jejím směrování.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tato práce se zabývá návrhem systému domácí automatizace, který by měl cenově dostupný a spolehlivý z dlouhodobého hlediska. Na začátek definuje, co to je systém domácí automatice a jaké výhody a nevýhody inteligentní domácnost přináší. Následně jsou diskutovány součásti inteligentního domu, funkční i nefunkční nároky.

Stanovením všech nároků na systém se dostává k analýze trhu s inteligentní domácností. Je důležité porovnat, co dělá výhody konkurenčním systémům, ale také hlavně se podívat na nedostatky, který by měl systém řešit, aby potenciální zájemci sáhli právě po něm.

Následuje prostor na volbu vhodné technologie. Správné implementační nástroje a technologie snižují náklady na vývoj a zároveň jej pomáhají urychlit. Pro modulární systém jako by měl být ten pro inteligentní domácnost, by technologie měla zároveň přinést výhodu vývojářům, kteří mají zájem systém rozšiřovat a upravovat.

Vhodné technologie a vývojové prostředí pomohly při vývoji open-source systému Housedroid. Některé jeho části jsou probrány v rámci praktické části práce. Jedná se hlavně o rozhodnutí technologického rázu, které pomáhají splnit požadavky na systém z hlediska rychlosti odezvy, spolehlivosti a rozšiřitelnosti.

Každý moderní systém umožňuje obsluhu z mobilní aplikace, a právě u inteligentního domu by mělo být samozřejmé ho monitorovat a ovládat zároveň právě přes aplikaci. Dobře rozvržené uživatelské prostředí a vzhled aplikace zpříjemňují práci se systémem. Proto je součástí této práce vytvoření webového klienta pro systém Housedroid s možností jeho budoucí úpravy na nativní mobilní aplikaci.

Pomocí responzivní aplikace bude systém ihned dostupný pro webový prohlížeč i pro většinu mobilních zařízení. Správnou volbou implementační technologie navíc bude transformace na nativní aplikaci velmi rychlá a jednoduchá.

I. TEORETICKÁ ČÁST

1 INTELIGENTNÍ DOMÁCNOST

Ve všech oblastech života dochází ke stále většímu pronikání informačních technologií. Co začalo počítačem, mobilem, tabletem se dál rozvíjí v tzv. internetu věcí (Internet of Things). Jedná se o síť propojených fyzických objektů s vestavěnou elektronikou. Ta je tvořena senzory, výpočetními jednotkami a komunikačními prostředky. Stále větší integrace vestavných zařízení umožňuje sledovat stále více aspektů našeho života. Získané informace mohou být využity na lepší poznaní vzájemného ovlivnění objektů.

1.1 Co je to domácí automatizace

Internet věcí je základem domácí automatizace. Jedná se o integraci různých senzorů a reakčních prvků v domě, které umožňují automaticky reagovat na různé situace bez potřeby zásahu obyvatel domu. Příkladem může být vytápění domácnosti. V každé místnosti se obvykle nachází zdroj tepla. Ten býval obvykle ovládaný jediným termostatem na celý dům. Takto fungující systém míval často za následek, že v místnosti s termostatem byla ideální teplota, ale zbylých místnostech v domě bylo příliš chladno nebo naopak zbytečně přetopeno. Přidáním teplotních senzorů do každé místnosti a upravení regulace tak, aby bylo možné spínat samostatně zdroje tepla v každé místnosti, podstatně zlepšuje tuto situaci.

Z dlouhodobého měření je možné vytvořit i o mnoho pokročilejší systém, který se učí, jak ovlivňuje topení v sousedních místnostech teplotu v místnosti sledované. Tím může zareagovat už dříve, než až se teploměr dostane na hraniční hodnotu pro vypnutí vytápění.

1.2 Hlavní výhody nasazení

Pokud by systém domácí automatizace nepřinesl výrazné výhody, nebylo by třeba o něm uvažovat. Tato sekce popisuje, co přinese systém domácí automatizace. Množství výhod je závislé na konkrétních prvních, které jsou v domě nasazené. Nejde ani poskytnout úplný seznam výhod, jelikož domácí automatizace patří k jedné z nejrychleji se rozvíjejících aplikací vestavných zařízení dneška a oblasti využití se rychle rozšiřují.

1.2.1 Řízení a monitorování spotřebičů

Integrací senzorů a akčních prvků do spotřebičů můžeme zvýšit pohodlí, které dům poskytuje svým uživatelům a stejně tak jejich pocit bezpečí. Běžná je situace, kdy se před dovolenou na poslední chvíli používá žehlička, a následně pak člověk po cestě přemýšlí, jestli náhodou nezůstala zapojená. Dalším hezkým příkladem je varná deska. I u ní se může stát podobná situace jako s žehličkou. Co třeba televize? Kolikrát se stane, že dítě ráno odchází z domu a nechá zapnutou televizi, která následně běží až do večera a neustále spotřebovává proud? Právě tyto případy, řeší možnost ovládání zásuvek případně širší možnosti vestavěné ve spotřebičích. V mobilní aplikaci jde lehce zkontrolovat, který spotřebič zůstal běžet a je možné ho vypnout.

1.2.2 Řízení osvětlení a nastavení atmosféry

Osvětlení je jedním z hlavních spotřebitelů elektrické energie v domě. Tato situace se sice s příchodem LED osvětlení mění, ale stále může být dosaženo výrazných úspor. Běžné použití znamená sepnutí světel pomocí vypínače a provoz na plný výkon. Plný provoz zůstává i v situaci, kdy v místnosti nikdo nezůstal, ale poslední odcházející zapomněl zhasnout v dobré víře, že se za chvíli vrátí. Chytrý dům sleduju pohyb svých obyvatel, sleduje úroveň světla v jednotlivých místnostech a je schopný světla sepnout nebo vypnout dle potřeby. Při využití světel, která jsou ztlumitelná, je možné i regulovat, jak intenzivně světlo svítí. Díky tomu je možné dosáhnout i mnohem příjemnější atmosféry.

1.2.3 Zabezpečení vstupu pomocí inteligentních zámků

Kolikrát se stane, že při cestě do práce se člověk přistihne s myšlenkou, jestli náhodou nezapomněl zamknout dveře? Co když dítě zapomene klíče od domu a nemá se jak dostat dovnitř? Pokud přidáme do dveří elektromechanická kování, můžeme odkudkoliv zkontrolovat stav zámků a případně zamknout nebo odemknout dveře.

1.2.4 Kontrola zavřených oken a dveří

Další běžnou situací je, jestli jsou skutečně zavřená okna nebo dveře. Nemusí to být jen z bezpečnostních důvodů, ale také pokud začíná pršet, každý chce vědět, kde má zavřít okno, aby potom nemuset utírat parapet. Dveřní senzory umožňují detekovat, zda je otevřeno nebo zavřeno a tuto informaci poskytnout svému uživateli. Pro větší pocit bezpečí je možné na sklo integrovat senzory vibrací, která alarmují, pokud došlo k rozbití skla.

1.2.5 Přehled o dění díky kamerovému systému

Kamerový systém v domě dává přehled o aktuálním dění. V kombinaci s nahrávacím zařízením může také pomoci s usvědčení pachatelů, ať už trestných činů, nebo domácích mazlíčků, kteří v nepřítomnosti „páníčka“, dělají zakázané věci.

1.2.6 Ovládání teploty a klimatu

Integrací senzorů sledujících teplotu, vlhkost a kvalitu vzduchu můžeme udržovat všude v domě příjemné klima. Pokud se nikdo přes den v domě nenachází, může být automaticky snížena teplota, a těsně před návratem zase opětovně zvýšena. Toto opatření vede nejen ke zvýšenému pohodlí, ale také ke snížení výdajů za energie.

Sledováním škodlivin ve vzduchu (např. CO) lze odhalit požár nebo obsah jedovatých plynů, a ihned varovat uživatele.

1.2.7 Ekonomické úspory inteligentnímu řízení a monitorování

Jak už bylo zmíněno u některé z předešlých výhod, je možné dosáhnout výrazných úspor za energie, protože se využívá pouze tolik, kolik je třeba. Výrazně klesnou náklady na vytápění a na spotřebu elektrické energie.

Zároveň je možné díky monitorovacímu systému sledovat spotřebu a omezit plýtvání.

1.2.8 Pocit jistoty

Díky možnostem sledování veškerých možností v domě jediným pohledem na aktuální stav můžeme mít klid v případě jakékoliv nejistoty. Pokud někoho často trápí situace, jestli na něco nezapomněl, investice do domácí automatizace mu ušetří starosti.

Tato kapitola je částečně založena na shrnutí výhod z (1).

1.3 Motivace k vytvoření systému

1.4 Cíle práce

Cílem této práce je navrhnout systém domácí automatizace, a připravit tak základy pro implementaci v rámci open source projektu Housedroid. Zároveň budou zhodnoceny a diskutovány možné implementační technologie. S využitím návrhů a vybraných technolo-

gií je provedena implementace konceptů modulů pro určení směru vývoje projektu a ověření realizovatelnosti.

1.5 Stanovení hypotéz

Předpoklady, kterých se práce snaží dosáhnout, jsou následujících:

Existují technologie umožňující rychlou implementaci systému na úrovni hardwaru a softwaru, vysokou spolehlivost, snadnou údržbu a rozšíření.

Je možné definovat a implementovat/navrhnout nejdůležitější moduly a zajistit jejich integraci pomocí zvolených technologií.

Poskytnout téměř okamžitou odezvu na operace požadované uživatelem.

Dosáhnout lepšího uživatelského zážitku z pohledu prostředí než jaký nabízí současné systémy.

Je možné dosáhnout poloviční ceny systému, než za jakou jsou nabízena řešení dostupná na trhu.

1.6 Struktura práce

Tato práce je rozdělena na teoretickou a praktickou část. V teoretické části je proveden průzkum dostupných řešení na trhu. Tato řešení jsou vzájemně porovnána s cílem najít jejich silné a slabé stránky. Tento průzkum poskytne náhled na to, jaké moduly jsou nabízeny uživatelům v rámci domácností. Na základě určení silných a slabých stránek jsou v následujících kapitolách definovány funkce, které by bylo vhodné vytvořit i v systému Housedroid a zároveň také, které věci by měli být oproti konkurenci vylepšeny.

Kromě stanovení těchto základů je také proveden průzkum dostupných technologií pro implementaci systému. Samostatně jsou diskutovány hardwarové platformy, které je možné použít pro řízení. Po zvolení výchozí platformy jsou dále zkoumána prostředí pro rychlé prototypování hardwarových modulů. Vybrané řešení bude použito pro návrh desek plošných spojů a přípravu podkladů pro výrobu.

Samotný je také průzkum dostupných jazyků, vývojových prostředí a knihoven, které by umožnily systém vytvořit. Příslušná kapitola opět definuje požadavky na softwarové prostření a hledá přes silné a slabé stránky nejhodnější prostředí pro následnou implementaci.

S definovanými požadavky na systém, vybranou základní platformou a vývojovým jazykem je proveden vysokoúrovňový návrh architektury. V rámci architektury jsou popsány i další zvažované alternativy řešení a vysvětlena rozhodnutí vedoucí k volbě konkrétního kroku.

V rámci praktické části je nejprve definován komunikační protokol, který bude použit napříč všemi moduly. Po vytvoření komunikačního protokolu je rozebrán návrh a implementace směrovacího modulu, který má v systému roli propojení všech prvků a tím tvoří páteř celého řešení. Tím je připraven základ pro ostatní moduly, které jsou rozebírány v navazujících kapitolách.

A v neposlední řadě je uživatelské rozhraní, které je tvořeno aplikací pro prostředí webu nebo možnou předělávku na nativní aplikaci. V rámci kapitoly věnované UI je navržen koncept responzivní aplikace a rozebrána její implementace.

2 PRŮZKUM TRHU

Tento průzkumu trhu zkoumá dostupná řešení pro domácí automatizaci s cílem analyzovat běžně dostupné moduly, řešení, uživatelské rozhraní a další součásti. Tyto informace budou dále použity k doplnění plánů modulů pro vytvoření konkurence schopného systému.

V rámci průzkumu budou zkoumána hlavní dostupná řešení na trhu. Tato řešení byla vybrána na základě trendů v rámci vyhledávacích nástrojů. Podle webů společností byl odhadnut počet partnerů realizujících řešení a také množství referenčních nasazení.

Pro každý z vybraných systému je možné v následujících podkapitolách nalézt jeho stručný popis, rozbor základních modulů a také analýzu silných a slabých stránek.

Právě využití analýzy silných a slabých míst by mělo umožnit následně vývoj kvalitního finálního řešení v rámci projektu Housedroid.

Cenové rozdíly budou uvažovány pro běžný rodinný dům velikosti 5 + 1, který bude mít nasazeno řízení osvětlení a topení.

2.1 Vyhledání dostupných řešení

Pro analýzu byla zvolena řešení zastupující několik kategorií. První kategorií jsou dostupná open source řešení. V rámci open source je poskytnut kompletní zdrojový kód softwaru a také návrhy hardwaru modulů a to včetně jejich boxů.

Druhou kategorii představují řešení od výrobců elektroinstalačních materiálů. Tato řešení rozšiřují portfolio výrobce původně „hloupých“ součástek, jakými jsou například jističe, o inteligentní řídící a měřící prvky.

Poslední zkoumanou kategorií jsou výrobci specializující se primárně na systémy domácí automatizace. Jejich řešení by měla být z teoretického hlediska ta nejpokročilejší a odvážnější než produkty běžných výrobců elektroinstalací a zároveň je úroveň zpracování očekávána na vyšší úrovni než jakou očekáváme od open source.

2.1.1 Open source řešení

V této kategorii je možné nalézt velké množství projektů, ale pouze jediné řešení je využíváno ve větším měřítku. Tímto řešením je Ninja Blocks. Projekt se nachází už ve druhé verzi, která je založena na poměrně netradičním řešení, které bude popsáno později v této kapitole.

2.1.2 Řešení od tradičních výrobců elektroinstalačních komponent

Pod tuto kategorii lze jednoduše zařadit výrobce, kteří jsou dodavateli sortimentu prodejců elektroinstalačních materiálů. Při prohlídce katalogů lze objevit výrobky od společností ABB, která je jedním z největších dodavatelů v ČR, od společností Legrand, Schneider, ale také více do šířky zaměřených společností jako je Siemens nebo Honeywell.

2.1.3 Společnosti se specializací na domácí automatizaci

V rámci úzce specializovaných firem se objevují hlavně 3 řešení s mírně odlišných cílovým segmentem. Jedná se o řešení Loxone, Fibaro a Control4. Vzhledem k očekáváním budou tyto systémy rozebrány s větším důrazem.

2.2 Ninja Blocks

Ninja Blocks byl uveden v lednu 2012 na Kickstarteru.[1] Jednalo se o levnou platformu, která měla spojovat jednotlivé bloky (moduly) pomocí principu „if this then that“, což můžeme přeložit na „pokud toto tak tamto“. Tento princip umožňoval definovat podmínky pro blok a vytvořit reakci na jiném bloku. Například pokud se teplota dostane pod 20 stupňů, spusť vytápění. Projekt získal podporů 102 935$. Počáteční požadovaný vklad se autorům podařilo získat za pouhých 72 hodin.

Systém využíval jako základ platforem Raspberry Pi nebo Beaglebone Black. Tyto platformy byly rozšířeny o bezdrátový komunikační modul v pásmu 433MHz. Jedná se o bezlicenční pásmo používané radioamatéry nebo zařízeními pro bezdrátové ovládání. Výhodou díky frekvenci je poměrně dlouhý dosah, ale problémy může způsobovat rušení a velké využití pásma. Je třeba mít na paměti, že se jedná o jeden z nejlevnějších způsobů bezdrátové komunikace! Většina systému byla vyvinuta v Javascriptu pomocí Node.js. Více informací o tomto jazyce lze najít v kapitole o implementačních technologiích pro software. Cena základní jednotky byla v řádu 120$.

Celý systém by lehce rozšiřitelný a objevilo se hodně (převážně asijských) společností, které pro něj začaly vytvářet bloky. Cena bloků začínala na 40$.

[1] Kickstarter je jedna z největších platforem na komunitní financování projektů.

Změna nastala v roce 2014, kdy byla oznámena nová podoba jménem Ninja Sphere. Raspberry a Beaglebone byly nahrazeny vlastní hardwarovou platformou, která byla umístěna v boxu v podobně vykrojené polokoule. Projekt odstartoval rovněž na Kickstarteru, ale tentokráte se podařilo vybrat 702 937$. Uvedení na trh znamenalo konec propagace staré verze platformy a jejích bloků. Zároveň byl následně ukončen veškerý prodej. Vše je sice dostupné na GitHubu, ale nikdo další nepokračuje s vývojem a výrobou. Z tohoto důvodu už nebude původní platforma dále porovnávána.

Technické řešení druhé verze je založené na vlastní hardwarové platformě a Sféře jako inovativním ovladači. Každá Sféra obsahuje moduly pro bezdrátovou komunikaci na bázi protokolu Bluetooth a také Zigbee. Náklady už jsou tedy vyšší, než to bylo u pásma 433MHz.

Jako implementační jazyk byl zvolen programovací jazyk Go, hostování v rámci Docker kontejnerů na operačním systému Ubuntu Core Snappy. Tyto technologie jsou popsány v kapitole o implementačních technologiích.

Z pohledu uživatele je důležité ovládání. Funkce ovladače je realizována detekcí pohybových gest nad přístrojem a vysvícení vzorů na vrcholu Sféry. Jde o nový koncept, který byl jedním z důvodů pro velký zájem ze strany uživatelů.

Co se změnilo, tak nová verze se nesnaží o transformaci existujících prvků domácnosti na inteligentní, ale spoléhá se na ovládání inteligentních prvků se zabudovaným podporovaným komunikačním protokolem. Například je umožněno ze sféry ovládat inteligentní žárovky Philips Hue, které umožňují řízení pomocí Bluetooth. To představuje velkou nevýhodu z pohledu pořizovací ceny systému. Kdy inteligentní žárovka stojí dvacetinásobek běžné. Tím jsou náklady na výměnu prvků v domácnosti velmi vysoké a to je ještě třeba přičíst pořizovací cenu systému v ceně asi 250$ za jednu Sféru. Těch se hodí mít samozřejmě více.

2.2.1 Silné stránky

1. Největší lákadlo je inovativní ovladač a řídící jednotka v jednom. To tvoří hlavní prodejní bod systému.
2. Integrované ovládání výrobků jiných společností umožňuje relativní volnost volby od více dodavatelů.
3. Očekávané uvolnění Sféry v rámci open source.

2.2.2 Slabé stránky

1. Chybí jakékoliv moduly pro konverzi stávajících řešení na inteligentní. Je třeba vyměnit všechny prvky, které mají být řiditelné. To je jedna z největších nevýhod pro využití na současném trhu.
2. Podporované prvky vyžadují Bluetooth nebo Zigbee a aplikační podporu. Tím jsou omezena řešení pouze na ty z vyšší cenové kategorie. Zvyšuje to náklady na pořízení systému.
3. Mladý produkt, se kterým zatím nejsou dlouhodobé zkušenosti a nejsou implementovány všechny podstatné moduly.

2.2.3 Cenová výhodnost

Tabulka 2.1 – Cenový odhad systému Ninja Sphere

Prvek	**Cena**
Řídící jednotky a ovládání	750$
Výměna svítidel	1800$
Řízení topení	650$
Celkem	**3200$ (ekv. 76 800Kč)**

2.3 ABB

Společnost ABB se zaměřuje na technologie pro energetiku a automatizaci. Jedná se o jednoho z hlavních dodavatelů elektroinstalačního materiálu v České republice. Pro zákazníky jsou známé hlavně produktové řady jako Tango, které představuje jeden z nejčastěji využívaných systémů ovladačů v domech.

Z pohledu inteligentních domů nabízí společnost dvě produktové řady – ABB Ego-n a ABB i-bus KNX. Tyto systémy se od sebe liší cílovým segmentem. Ego-n je určený pro instalaci v rezidenčních stavbách, zatímco i-bus KNX cílí na komerční stavby a luxusní rezidence. Tedy na vyšší cenovou hladinu.

Nejprve bude rozebrán systém Ego-n. Z technického hlediska se jedná o sběrnicový systém pro propojení 512 prvků. Sběrnice je vedena na čtyřech vodičích a propojuje všechny prv-

ky. Jedná se tedy o kabelový nikoliv bezdrátový systém. Z tohoto pohledu je vhodnější pro nasazení do novostaveb nebo rekonstrukcí než jako transformace stávajícího systému.

Možnosti modulů jsou poměrně rozsáhlé. Jedná se o regulaci osvětlení (spínání, stmívání, bezdrátové ovládání, světelné scény, simulace přítomnosti), ovládání rolet a žaluzií, spínání spotřebičů a zásuvek, detekce přítomnosti, měření spotřeby, teplot a hlásič kouře.

Z pohledu vzdáleného ovládání je k dispozici aplikace pro Android a iOS, a dále také ovládání přes webový prohlížeč.

2.3.1 Silné stránky

1. Systém nabízí velké množství modulů
2. Nabízí aplikace pro mobilní zařízení a webový prohlížeč
3. Je k dispozici více designových řad produktů

2.3.2 Slabé stránky

1. Jedná se o proprietární systém neumožňující rozšíření jiné než od výrobce
2. Nelze integrovat s inteligentními prvky (např. chytré žárovky)
3. Není možné lehce transformovat současnou instalaci
4. Aplikace pro mobilní zařízení působí přeplácaným a nepřehledným dojmem, kdy tvůrci chtěli dát pocit maximální kontroly místo komfortu a jednoduchosti

2.3.3 Cenový výhodnost Ego-n

Cena byla odhadnuta na základě vzorového návrhu dostupného na stránkách společnosti ABB.

Tabulka 2.2 – Cenový odhad systému ABB Ego-n

Prvek	**Cena**
Řídící jednotky a ovládání	48 000 Kč
Řízení osvětlení	20 000 Kč
Řízení topení	21 600 Kč
Kabeláž	3 000 Kč
Celkem	**92 600 Kč**

2.3.4 Rozdíly pro ABB i-bus KNX

Vzhledem k použití protokolu KNX lze očekávat, že systém bude podporovat prvky jiných výrobců fungující pod sběrnicí KNX. Tím je eliminována nevýhoda použití pouze prvků od jednoho výrobce. Ostatní výše zmíněné ovšem zůstává.

Co se mění tak je cena systému. Vzhledem k cílení na vyšší segment lze očekávat podstatné navýšení ceny.

2.3.5 Cenový výhodnost i-bus KNX

Cenový odhad byl stanoven stejně jako v přechozím případě podle vzorového rozpočtu na webu společnosti.

Tabulka 2.3 – Cenový odhad systému ABB i-bus KNX

Prvek	**Cena**
Řídící jednotky a ovládání	34 000 Kč
Řízení osvětlení	30 000 Kč
Řízení topení	8 000 Kč
Kabeláž a další prvky	10 000 Kč
Celkem	**93 000 Kč**

Při využití ABB-ComfortPanel, což je dotykový ovladač systému o velikosti 12“, se cena zvyšuje o 92 360 Kč. Tato částka představuje oproti tabletu s instalovanou aplikací neúměrný náklad. Zároveň se jedná o prvek, který je potřeba pro ovládání systému přes mobilní telefon.

2.4 LEGRAND

Legrand je francouzský výrobce elektrotechniky a mezi výrobci spínačů a zásuvek patří mezi největší na světě. I tato společnost nabízí své řešení pro inteligentní domácnost pojmenované My Home.

Systém je zaměřený na několik oblastí:

Atmosféra – řízení osvětlení, žaluzií a zvuku;

Úspory – řízení topení, klimatizace a sledování spotřeby;

Bezpečí – alarm, detektory úniku plynu či proudu, kamerový systém;

Ovládání – k dispozici jsou dotykové panely různých velikostí a aplikace pro tablet a PC.

Z pohledu funkcionality nabízí totéž, co varianta od ABB. Systém je k dispozici ve dvou verzích a to jako sběrnicový nebo bezdrátový systém.

Technické řešení sběrnicového systému je postavené na sběrnici SCS. Jako kabeláž je využita kroucená dvojlinka. Toto řešení využívá kromě Legrandu společnost Bticino, jinak nemá v rámci domácí automatizace větší rozšíření. Toto je limitujícím faktorem pro výběr dodavatelů modulů a pro možné rozšiřování systému.

Varianta postavené nad protokolem ZigBee využívá proprietární komunikační protokol, jedná se ovšem o systém nabízející snadnou transformaci stávajících řešení.

2.4.1 Silné stránky

1. Dostupná sběrnicová i bezdrátová varianta
2. Velké množství modulů pro automatizaci
3. Dostupná aplikace pro mobil, tablet, PC
4. Více designových řad
5. Přehledná aplikace

2.4.2 Slabé stránky

1. Protokol, který nevyužívá více výrobců a tím omezený výběr modulů od různých dodavatelů
2. Neumožňuje integraci s inteligentními prvky jiných výrobců
3. Design aplikace nepůsobí nejmoderněji

2.4.3 Cenová výhodnost

Cenový odhad byl proveden podle vzorového listu výrobce pro sběrnicové řešení s ovládacím systémem podle rozšířených scénářů.

Tabulka 2.4 – Cenový odhad systému Legrand My Home

Prvek	**Cena**
Řídící jednotky a ovládání	28 000 Kč

Řízení osvětlení	19 000 Kč
Řízení topení	24 000 Kč
Kabeláž a další prvky	3 000 Kč
Celkem	**74 000 Kč**

2.5 Schneider Electric

Schneider Electric je další francouzskou společností na trhu s elektrotechnickým materiálem. Pro domácí automatizaci nabízí systém sběrnicový systém KNX. Systém se chlubí certifikací asociace KONNEX a garantuje kompatibilitu s moduly jiných výrobců KNX systémů.

Hlavní cíle tohoto řešení jsou úspora nákladů, přizpůsobitelnost, rozšiřitelnost, garance a certifikace.

Základní moduly odpovídají předchozím systémům:

Jsou obsaženy moduly pro řízení osvětlení, ovládací tlačítka, dotykové displeje, čidla pohybu a přítomnosti, senzory povětrnostních podmínek, ovládání multimediálních systémů, zabezpečovací a kamerový systém, požární hlásič, přístupový systém, ovládání a časové řízení spotřebičů, regulace vytápění, chlazení a klimatizace, řízení stínící techniky.

2.5.1 Silné stránky

1. Systém postavený na sběrnici KNX
2. Velké množství modulů a to včetně modulů od jiných výrobců
3. K dispozici bezdrátový ovladač světel a žaluzií
4. Více designových řad
5. Dostupná mobilní aplikace

2.5.2 Slabé stránky

1. Mobilní aplikace obsahuje velké množství informací, chybí přehledné a jednoduché ovládání
2. Integraci inteligentních Bluetooth prvků je třeba řešit produkty jiného výrobce, ale opět řešení existuje

2.5.3 Cenová výhodnost

Cenový odhad byl proveden na základě vzorového scénáře na webu výrobce.

Tabulka 2.5 – Cenový odhad systému Schneider Electrics

Prvek	**Cena**
Řídící jednotky a ovládání	37 000 Kč
Řízení osvětlení	18 000 Kč
Řízení topení	7 000 Kč
Kabeláž a další prvky	3 000 Kč
Celkem	**65 000 Kč**

Pozn. v části řídící jednotky jsou zahrnuty spínače osvětlení a ovladače řízení topení.

2.6 Honeywell

Honeywell je americká společnost, která se zaměřuje na široké spektrum odvětví, jako jsou letecké a dopravní systémy, materiály ale také automatizace a řízení. Proto nabízí i řešení pro domácí automatizaci. Systém je postavený na protokolu Z-Wave, který lze považovat za konkurence ZigBee.

Z funkčního hlediska systém poskytuje funkcionalitu potřebou pro řízení osvětlení, regulaci topení čí chlazení, větrání, ovládání zámků a sledování kamerového systému.

2.6.1 Silné stránky

1. Systém je postavený na bezdrátovém protokolu, lze tedy využít i pro konverze.
2. Dostupné aplikace pro mobilní zařízení a webový prohlížeč

2.6.2 Slabé stránky

1. Pro systém není k dispozici takové spektrum modulů, jako nabízejí ostatní výrobci, je třeba doplnit produkty od jiných výrobců podporujících Z-Wave
2. Design výrobků je velmi zastaralý
3. Aplikace jsou „přeplácané“, designově zastaralé a ve spoustě míst neintuitivní

2.6.3 Cenová výhodnost

Tabulka 2.6 – Cenový odhad systému Honeywell

Prvek	**Cena**
Řídící jednotky a ovládání	35 000 Kč
Řízení osvětlení	Obsaženo v ovladači
Řízení topení	12 000 Kč
Celkem	**47 000 Kč**

2.7 Siemens

Společnost Siemens je jednou z nejvýznamnější německých společností v oblasti elektrotechniky a strojírenství. V nabídce má velké množství prvků pro elektroinstalace, řízení a automatizaci. Zaměření je ale spíše na komerční objekty a větší rezidentní budovy než na rodinné domy.

V nabídce má dva typy systémů a to systémy Gamma a systém Synco Living. Oba systémy jsou postavené nad sběrnicí KNX, ale také zároveň podporují její bezdrátovou variantu KNX RF.

Z funkčního hlediska, ale Siemens nabízí v těchto řadách pouze displeje, mobilní a webovou aplikaci a řízení topení. Ostatní akční prvky jako je regulace osvětlení je doporučena doplnit dalšího německého výrobce Hager.

2.7.1 Silné stránky

1. Systém postavený na sběrnici KNX a i její bezdrátové variantě KNX RF
2. Dostupná aplikace pro mobil, tablet a webový prohlížeč.
3. Moderní a intuitivní mobilní aplikace

2.7.2 Slabé stránky

1. Design některých výrobků (série Synco Living) je zastaralý
2. Nutné doplnit prvky jiných výrobců
3. Nemožnost integrace inteligentních Bluetooth prvků

2.7.3 Cenová výhodnost

Cenový odhad vychází z akční nabídky produktové řady Synco Living s odebráním termoelektrických hlav, který nebyly započítány i předchozích odhadech s doplněním prvků pro řízení osvětlení od společnosti Hager.

Tabulka 2.7 – Cenový odhad systému Siemens

Prvek	**Cena**
Řídící jednotky a ovládání	34 000 Kč
Řízení osvětlení	24 000 Kč
Řízení topení	16 500 Kč
Kabeláž a další prvky	3 000 Kč
Celkem	**77 500 Kč**

2.8 Loxone

Loxone je rakouská společnost založená v roce 2008, která si dala za cíl vytvořit moderní systém domácí automatizace. To je také její jediná specializace.

Z pohledu funkcionality je systém prezentován v několika kategoriích:

Bezpečnost – tlačítko strachu, která rozsvítí světla, vytáhne žaluzie a volá v o pomoc. Podobnou funkci má i jako alarm. Dále má detektory vody, kouře a také simulaci přítomnosti.

Komfort – řízení osvětlení včetně jeho barvy a intenzity, ovládání multimedií, vidovrátník a automatické vytváření scén.

Úspora energie – monitorování spotřeby elektrické energie, vody, fotovoltaiky, přepínání zařízení do pohotovostního režimu, regulaci vytápění, větrání, ovládání žaluzií.

Z technického hlediska je hlavní řízení řešeno centrální jednotkou, která využívá modulů nazvaných extension (rozšíření). Využit je proprietární protokol Loxone Link a také na centrální jednotce je k dispozici konektor pro připojení na sběrnici KNX.

2.8.1 Silné stránky

1. Systém podporuje protokol KNX, ale také i velké množství jiných protokolů pomocí rozšíření – 1Wire, 0-10V, Modbus, RS232 a RS485
2. Velké množství modulů nabízených od výrobce, připojení ještě většího množství od jiných výrobců
3. Aplikace pro mobil, tablet a web
4. Moderní a povedený design aplikace
5. Přehlednost aplikace

2.8.2 Slabé stránky

1. Potřeba rozšíření k základní jednotce pro podporu jiných protokolů
2. Vzhledem k rozložení aplikace potřeba skrolování

2.8.3 Cenová výhodnost

Tabulka 2.8 – Cenový odhad systému Loxone

Prvek	**Cena**
Řídící jednotky a ovládání	24 500 Kč
Řízení osvětlení	22 600 Kč
Řízení topení	12 000 Kč
Kabeláž a další prvky	3 000 Kč
Celkem	**62 100 Kč**

2.9 Fibaro

Fibaro je další společnost, která se zaměřuje výhradně na systémy domácí automatizace. Jako ostatní systémy nabízí funkce monitorování spotřeby energií, senzory pohybu, regulaci světel a vytápění, alarm, detektor kouře nebo úniku vody.

Systém je postavený nad protokolem Z-Wave. Je tedy bezdrátový a měl by umožňovat spolupráci s některými dalšími zařízeními pracujícími na tomto protokolu. Za zmínku stojí použití procesoru Intel Atom v rámci centrální jednotky. Žádná podstatná inovativních řešení v tomto systému nejsou použita.

2.9.1 Silné stránky

1. Systém je postavený na bezdrátovém protokolu Z-Wave a lze jej použít i pro snadnou transformaci stávajících řešení
2. Dostupné moduly pro většinu požadavků
3. Dostupná aplikace pro mobil, tablet a webový prohlížeč

2.9.2 Slabé stránky

1. Aplikace působí „přeplácaným“ dojmem. Obsahují příliš mnoho voleb v rámci jedné obrazovky
2. Horší design aplikace
3. Prezentace produktu, obtížné dohledat podstatné informace

2.9.3 Cenová výhodnost

Tabulka 2.9 – Cenový odhad systému Fibaro

Prvek	**Cena**
Řídící jednotky a ovládání	44 600 Kč
Řízení osvětlení	10 000 Kč
Řízení topení	10 000 Kč
Celkem	**64 600 Kč**

3 STANOVENÍ CÍLŮ SYSTÉMU A ANALÝZA DŮLEŽITÝCH MODULŮ

Z předchozí kapitoly jde vidět, že se základní funkcionalita ve většině případů opakuje, a výrobci pouze občas přidávají některé další moduly pro získání konkurenční výhody. Zároveň minulá kapitola shrnula u každého systému jeho silné a slabé stránky.

V této kapitole bude proveden rozbor všech modulů, které jsou nutné pro vytvoření použitelného a konkurence schopného systému, která neztrácí oproti současným řešením.

Zároveň také budou stanoveny parametry systému podle silných a slabých stránek současných řešení, která systém musí řešit, aby i z tohoto pohledu byl pro uživatele atraktivnější než konkurence.

Na závěr kapitoly bude provedena SWOT analýza navrhovaného řešení. SWOT analýza je metoda, jejíž pomocí je možno identifikovat silné (ang: Strengths) a slabé (ang: Weaknesses) stránky, příležitosti (ang: Opportunities) a hrozby (ang: Threats), spojené s určitým projektem nebo podnikatelským záměrem. Bude zároveň sloužit pro zhodnocení, zda se vyplatí projekt realizovat.

3.1 Obecné požadavky na systém

Systém domácí automatizace musí poskytnout uživateli jednoduché a přehledné uživatelské rozhraní, které odpovídá moderním trendům a umožňuje případnou pozdější změnu designu. Uživatelské rozhraní bude rozdělené na dvě části.

Jednou částí je aplikaci, která musí být použitelná na všech typech zařízení: mobilní telefon, tablet, počítač (webový prohlížeč) a také dotykový ovládací panel (například postavený na základě tabletu).

Druhou část tvoří samostatné spínače a regulátory, které můžou nahradit klasické spínače elektroinstalace. Tyto spínače je třeba rozdělit na skupinu, která je jednoúčelová a víceúčelová. Jednoúčelové spínače a regulátory poskytuje operace pouze pro jednu činnost. Například spínač osvětlení umožňuje zapnout a vypnout osvětlení; regulátor topení umožňuje nastavit pomocí otočného tlačítka teplotu v místnosti.

Dalším požadavkem je rychlost reakce systému. Z pohledu uživatele musí být požadovaná operace provedena okamžitě. Za okamžitou reakci je považována doba 100 ms. Toto je rozebráno v kapitole o architektuře a návrhu systému.

Zavedení systému musí také zvýšit komfort užívání domu. Tomu napomáhá jak dobře provedené ovládání, tak možnosti nastavení atmosféry barvou osvětlení, nebo automatizace potřebných úkonu. Mezi takové patří automatické větrání v případě potřeby nebo stažení rolet či žaluzií.

Dalším požadavkem je zvýšení pohodlí při užívání domu. Zpětný okruh teplé vody, kdy při příchodu do místnosti, čerpadlo zareaguje při nízké teplotě vody na cílovém místě spuštěním okruhu a přísunem teplé vody na místo, kde by ji člověk mohl použít. Tím odpadá potřeba odpouštět vodu, a šetří se náklady jak za vodu, tak za odpad. Dále lze inteligentním řízením vytápění ušetřit náklady na energie.

Zvýšení bezpečí je dalším důležitým aspektem inteligentního domu. Dům musí být schopen chránit a upozornit svého majitele v případě požáru nebo v situaci, kdy praskne potrubí u pračky či myčky. Musí nabízet možnosti kamerového systému, detektory rozbití oken a alarm proti nezvaným návštěvníkům.

Posledním důležitým aspektem je poskytnout uživateli možnost monitorování a řízení spotřeby. Poučený uživatel může sledovat, jaké jsou jeho náklady v kterou dobu a zároveň by měl pomoci spotřebu snižovat přepínáním spotřebičů do stand-by režimu nebo úplném vypnutí zásuvky.

3.2 Požadované moduly pro systém inteligentního domu

Tato kapitola rozebere uživatelské nároky na jednotlivé moduly, které musí mít systém inteligentního domu. Velké množství modulů je založené na analýze modulů rozebraných v předchozí kapitole.

3.2.1 Centrální jednotka

Z uživatelského pohledu je vhodné mít centrální jednotku, která poskytuje možnost vzdáleného přístupu a poskytuje webové rozhraní aplikace. Tato jednotka zároveň slouží pro koordinaci nebo konfiguraci ostatních jednotek, které jsou součástí systému.

Zároveň využití centrální jednotky umožňuje vytvoření modulů pro rozšíření systému o komunikační jednotky pro spolupráci s jinými systémy – např. na sběrnici KNX. Tím je možné lehce rozšířit systém i o cizí moduly a zajistit spolupráci v rámci již existujících systémů.

3.2.2 Stmívač

Pro regulaci osvětlení je třeba mít k dispozici modul stmívače. Stmívač by měl být vytvořen jako samostatná jednotky – tj. pro jeden okruh světel. Současná řešení nabízí vícekanálové stmívače. Toto může být řešeno instalací několika samostatných stmívačů v rámci jednoho boxu.

Základní funkce stmívače je možnost zapnutí/vypnutí světel a řízení intenzity jejich osvětlení. Pro pokročilejší variantu je možné přidat i nastavení barvy osvětlení pro LED pásky.

Modul by měl fungovat pro jakýkoliv typ zátěže, a to kapacitní, induktivní a odporovou.

Maximální očekáváný výkon pro jeden okruh byl stanoven na 600W. Což je využití 10 kusů 60W žárovek s wolframovým vláknem. Pro LED takový okruh zvládne podstatně více svítidel.

3.2.3 Relé

Dalším kusem by měl být modul relé. Relé umožňuje sepnout nebo vypnout okruh podle potřeby. Takový modul se dá využít pro řízení větví podlahového topení systémem ON/OFF. Pro plynule nastavitelný systém se hodí spíše stmívač.

Aby bylo dosaženo, co nejvyšší životnosti, neměla by být využita elektromagnetická relé, ale součástky na polovodičové bázi.

U základního systému relé bude stačit výkon okolo 600W. Pro náročnější systémy by měla být k dispozici i verze s podporou 2500W. Tato varianta by mohla být využita například v chytrých zásuvkách.

3.2.4 Senzory

Senzory je možné řešit v rámci systému dvěma způsoby. Prvním způsobem je vytvoření modulu, který bude pracovat na existujícím protokolu čidla. V tomto režimu podporovaným systémem by měla být sběrnice 1Wire. Na sběrnici 1Wire existuje velké množství levných senzorů, která sníží počáteční náklady na vytvoření senzorových modulů a umožní poskytnutí očekávaných součástí hned ze začátku.

Dalšími uvažovanými moduly pro protokoly by měly být sběrnice Modubus, která se používá většinou pro čítače; RS232 a RS485.

Druhý způsob umožňuje vytvoření pokročilejších a kombinovaných systému senzorů založených na vlastním protokolu. Stejně jako celý automatizační systém i komunikační protokol by měl být otevřený pro možnost využití dalšími výrobci.

V příkladech takových senzorů můžeme uvést senzory kvality ovzduší s integrovanými detektory kouře, snímači intenzity osvětlení a senzory přítomnosti. Takovýto systém senzorů typu „vše v jednom“ umožňuje snížit náklady na výrobu, pořízení a zvýšit rychlost instalace.

3.2.5 Bezpečnostní moduly

V rámci bezpečnosti je třeba poskytnout běžně nabízené moduly. Základem je kamerový systém pro sledování okolí s detekcí pohybu. Kamerový systém by měl umožňovat zaznamenávání obrazu. Díky detekci pohybu může být měněna kvalita ukládání a zároveň urychleno nalezení míst s pohybem osob.

Doplňkové moduly jsou senzory otevřených a zavřených oken, senzory rozbití skla a elektronické ovládání zámků s možností automatického zamčení.

3.2.6 Ovládání

U ovládání systému je třeba se dívat na několik částí. První částí je spojení s dalšími prvky systému. Pro ovládání je velmi vhodné umožnit bezdrátové spojení ovladače a přijímače (buďto v podobě modulu nebo součást centrální jednotky). To umožní usnadnit transformaci stávajících systémů bez nutnosti rekonstrukce rozvodů a také pozdější doplňování ovladačů. Pro rekonstrukce nebo novostavby by měl být k dispozici kabelový systém, který obecně dosahuje lepší spolehlivosti a nižší odezvy.

Základem systému je ovládání pomocí aplikace pro mobilní telefony, tablet a webový prohlížeč. Aplikace musí být co nejjednodušší i za cenu méně voleb než má konkurence. Zároveň rozhraní musí být velmi přehledné a lehce ovladatelné. Na mobilních telefonech a tabletech musí být UI jednoobrazovkové – tj. bez skrolování dolů.

Stejná mobilní aplikace by měla být nasaditelná do dotykových panelů, které budou jedním z možných stacionárních ovladačů. Tyto panely mohou být vytvořeny na základě tabletu s jiným rámečkem, případně může být použit tablet s nástěnou dokovací stanicí.

Další alternativní způsob ovládání představují samostatné ovladače. Funkce ovladači může být konfigurovatelná. Jednoduchý dvojpolohový ovladač může sloužit pro sepnutí nebo

vypnutí světel. Více polohový ovladač může být využit pro nastavení scén (např. barva a intenzita osvětlení).

Pro další aplikace je možné vytvoření i otočného ovladače. Takovýto ovladač může sloužit pro regulaci intenzity osvětlení nebo pro nastavení teploty v místnosti.

Budoucí uvažované rozšíření nabídky jsou dálkové ovladače.

3.3 SWOT analýza navrženého řešení

3.3.1 Silné stránky

1. Možnost spolupráce s jinými protokoly jako KNX nebo 1Wire. Tím je možné využít nabídku senzorů a prvků od jiných společností
2. Bezdrátová varianta ovládání usnadňuje transformaci současných řešení a umožňuje i budoucí přidání ovladačů
3. Velké množství dostupných modulů
4. Aplikace pro mobilní telefon, tablet a webový prohlížeč
5. Jednoduchá, přívětivá a moderní aplikace
6. Design prvků s možností změny krytu
7. Cena systému

3.3.2 Slabé stránky

V této fázi práce nejsou očekávané slabé stránky

3.3.3 Příležitosti

1. Rychle se rozvíjející trh
2. Jednoduchá konverze stávajícího systému na inteligentní
3. Poloviční cílová cena oproti konkurenci
4. Design prvků
5. Jednoduchost a komfort ovládání

3.3.4 Hrozby

1. Z počátku nedostatečné množství modulů
2. Certifikace produktů a příprava na trhy
3. Zajištění výroby o vysoké kvalitě a nízké ceně
4. Nalezení partnerů poskytujících instalaci

4 VOLBA IMPLEMENTAČNÍCH TECHNOLOGIÍ

Volba implementační technologie ovlivňuje jak architekturu systému, tak i dobu a náročnost implementace. Pro systém inteligentního domu je třeba vybrat vhodné prostředky pro implementaci hardwarové částí a softwaru, který s nimi bude pracovat.

Ne každý jazyk umožňuje pracovat s hardwarem. Proto budou porovnány pouze jazyky, které toto umožňují a z nic bude vybrán ten, který umožní rychlý a spolehlivý výsledek.

4.1 Hardware

V rámci hardwaru existují možnosti buďto navržení všech součástí od základu nebo využití některé z dostupných vývojových platforem pro vestavná zařízení.

Cesta kompletního vlastního řešení má výhodu, že výsledek je přesně vytvořený na základě požadavků. Je možné dosáhnout menších rozměrů, nižší spotřeby a jsou integrovány všechny požadované funkce. Nevýhodou naopak je výrazně delší doba vývoje a vyšší náklady na výrobu oproti využití existujícího řešení.

V případě volby vývojové platformy musíme nejprve vybrat řešení, které poskytne nejjednodušší vývoj. Ne každá platforma poskytuje stejnou funkcionalitu. U využití vývojové platformy je třeba počítat s většími rozměry výsledného produktu a nutnosti propojení platformy a vlastních rozšiřujících modulů (běžně označované jako cape).

Jelikož jedním z hlavních požadavků je řešení, která je otevřené rozšířením, rychlý vývoj a nízké výrobní náklady, je zvolena cesta využití vývojové platformy.

V rámci projektů pro Internet of Things jsou v současnosti nejvíce využívány platformy Arduino, Beaglebone a Raspberry Pi. Pro tento systém bylo Arduino vyřazeno, jelikož je jako platforma nejjednodušší, výkon procesoru je velmi nízký a bylo by potřeba řešit podstatně více vlastních rozšíření.

V následujících kapitolách budou porovnány mezi sebou platformy Beaglebone a Raspberry.

4.1.1 Beaglebone Black

Beaglebone Black je nízkonákladová komunitní vývojová platforma pro vývojáře a hobby. Dokáže nabootovat Linux pod 10 sekund a je možné zahájit vývoj během pěti minut po připojení USB kabelu.

4.1.1.1 Hardwarové specifikace

Procesor: AM335x 1GHz ARM® Cortex-A8

Paměť: 512MB DDR3 RAM

Perzistentní úložiště: 4GB 8-bit eMMC

Obsahuje 3D grafický akcelerátor

Obsahuje akcelerátor výpočtů v plovoucí řádové čárce NEON

Obsahuje dva 32-bit PRU mikrokontroléry

4.1.1.2 Podporované operační systémy

Debian, Ubuntu, Angstrom, FreeBSD, Gentoo, ArchLinux, LinuxCNC, Minix, Android

4.1.1.3 Zhodnocení platformy

V roce 2014 měla platforma velkou výhodu proti jiným nízkonákladovým platformám v podobě výkonnějšího procesoru a podpory novějších verzí Linuxu. S nástupem Raspberry Pi 2 a alternativních platforem tuto výhodu již ztratila.

Operační paměť obsažená na platformě stačí pro většinu běžných aplikací. Nebyla by vhodná pouze na centrální jednotku, pro kterou se hodí více paměti.

Perzistentní úložiště je paměť kategorie SSD. 4GB jsou dostatečné pro instalaci operačního systému, ale nemusí již stačit, pokud aplikace potřebuje lokálně ukládat data. Pro takový účel by bylo lepší mít k dispozici 8GB a více.

Z pohledu napájení Beaglebone vyžaduje 210-460 mA při 5V. To odpovídá přibližně 1-2,5W. Při přičtení spotřeby rozšíření je tedy možné vytvořit řídící platformu, která má spotřebu cca 5W, což odpovídá spotřebě mnoha spotřebičů ve stand-by režimu.

Z pohledu vývojářského platforma má k dispozici 69 GPIO portů. GPIO je zkratka pro general purpose input output (vstup/výstup pro obecní použití) a také porty pro komunikaci na protokolech SPI a I2C. Další užitečnou funkcí jsou dva PWM moduly. Díky nim je možné generovat PWM (pulse width module – v češtině používáme výraz pulzně šířková modulace) bez závislosti na procesoru. Tím je možné ušetřit procesorový čas pro jiné úlohy a snížit spotřebu. Podobné je to i u 4 hardwarových časovačů.

Síťová karta na Beaglebonu je pouze o rychlosti 10/100 Mbps, což je v kategorii nízkonákladových modulů obvyklé.

Podporované jsou jakékoliv vývojářské jazyky, které jsou dostupné na Linuxu. Jsou předpřipravené knihovny pro C/C++, Python, Javascript (Node.js) a Go. Ostatní jazyky mohou k portům přistupovat pomocí Linuxového device tree. Tento přístup je ovšem řádově výrazně pomalejší a má vyšší režii. Pro dosažení nejlepšího možného výkonu na portech se hodí sáhnout po předpřipravených knihovnách. Zde se mezi jazyky také výkon výrazně liší. Pro nejlepší možný výkon při práci se vstupy a výstupy je vhodné zvolit jazyk umožňující přímý přístup do paměti.

4.1.1.4 Shrnutí

Silné stránky:

1. Velké množství GPIO portů
2. PWM výstupy
3. Velké množství dostupných distribucí
4. Široký výběr programovacích jazyků
5. Open source projekt, všechny návrhy dostupné

Slabé stránky:

1. Starší jednojádrový procesor s nižším výkonem
2. Méně dynamické a perzistentní paměti
3. Pouze 1 USB port

4.1.2 Raspberry Pi 2

Raspberry Pi je nízkonákladový počítač velikosti kreditní karty. Vznikla v rámci Raspberry Pi Foundation jako charitativní vzdělávací projekt, který má umožnit přistup k počítači všem. Vzhledem ke svým možnostem se Raspberry stalo, ale také vyhledávanou platformou vývojářů, kteří nad ním vytváří své projekty pro IoT.

V rámci toho hodnocení bude rozebrána pouze platforma Raspberry Pi 2, když na začátku prací nebyla ještě dostupná. Zmíněny ovšem budou některé rozdíly, které nové Pi přineslo oproti své první verzi.

4.1.2.1 Hardwarové specifikace

Procesor: Broadcom BCM2836, čtyřjádrový 900MHz ARM® Cortex-A7

Paměť: 1GB RAM LPDDR2

Perzistentní úložiště: chybí/nutné využít SD karty

Obsahuje 3D grafický akcelerátor VideoCoreIV

Obsahuje akcelerátor výpočtů v plovoucí řádové čárce NEON

4.1.2.2 Podporované operační systémy

Raspbian, Debian, Ubuntu, OPENELEC, OSMC, Pidora, Risc OS, Windows for IoT

4.1.2.3 Zhodnocení platformy

Raspberry Pi 2 přišlo v roce 2015 jako náhrada za první verzi B+. Hlavní rozdíly jsou v použití lepšího procesoru a více paměti. Původní procesor měl pouze jedno jádro taktované na 700 MHz s architekturou ARM6. Problém této architektury byl, že nebyla podporována novějšími Linuxovými distribucemi jako je Ubuntu.

Se čtyřmi jádry novými jádry procesorový výkon násobně poskočil. Je tedy možné zpracovat mnohem více operací a to i paralelně.

Operační paměť je staršího standardu DDR2, který nemá tak vysoké takty a proto lze očekávat nižší propustnost paměti. Ovšem pro tento procesor by paměť neměla být limitem. Použita je hlavně varianta LP, což je low power (nízký příkon), který podstatným způsobem snižuje spotřebu energie.

Z první verze zůstala nevýhoda/výhoda chybějícího perzistentního úložiště. To je řešeno slotem na SD karty. Proto je třeba vždy hledět na to, jaký výkon pořizovaná karta nabízí. Lze se poté teoreticky dostat na výkon vestavného úložiště a poskytuje možnost rozšíření kapacity v budoucnu. Zároveň možnost výměny karet může pomoci při vývoji, kdy na jednom Pi můžeme rychle vystřídat různé systémy a konfigurace pro různé aplikace.

Napájení je řešeno 5V konektorem. Spotřeba se pohybuje od 230mA (ve stavu idle) do 420mA (při využití všech 4 jader). To nám dává přibližně 1-2W. Procesor má inteligentní řízení spotřeby, takže při nevyužití, je jádro CPU deaktivováno a odpovídajícím způsobem klesá spotřeba.

Z pohledu vývojáře máme k dispozici 40 pinů - z toho je 26 pinů určeno pro GPIO. Mimo GPIO jsou podporovány i SPI a I2C, jejichž piny jsou sdíleny i pro GPIO. Naopak chybí hardwarová podpora pro PWM. Platforma také nemá obvod pro hodiny reálného času. Je třeba proto vždy čas synchronizovat ze sítě.

Toto je kompenzováno více USB porty a samostatným konektorem pro kameru. Pi se tak dá využít pro vytvoření inteligentních kamer. Dostupné jsou 1080p kamery i v NoIR verzi (snímá infračervené záření a lze ji použít pro noční vidění).

Programovací jazyk jde opět použít jakýkoliv podporovaný Linuxem nebo Windows for IoT. Jsou dostupné knihovny pro většinu jazyků (většina knihoven je komunitních). Pro nejvyšší výkon při práci s GPIO platí výhoda nativních jazyků.

4.1.2.4 Shrnutí

Silné stránky:

1. Velký výkon CPU
2. 1GB operační paměti
3. Velké množství dostupných distribucí
4. Široký výběr programovacích jazyků
5. Rozsáhlá komunita
6. 4 USB porty
7. Podporuje také Windows for IoT

Slabé stránky:

1. Chybí obvod pro hodiny reálného času
2. Menší propustnost paměti

4.2 Srovnání Rasberry Pi a Beaglebone Black

Nejvýznamnější rozdíl mezi Raspberry Pi 2 a Beaglebone Black je v procesoru. Ač označením vypadá A7 horší než A8, opak je pravdou. A7 je novější generace, která je plně kompatibilní s její výkonnější verzí A15.

Oba procesory mají in-order zpracování instrukcí, takže v tomto ohledu není v architektuře výrazný rozdíl. Liší se délka pipeline. A7 má 8 stupňů zatímco A8 má 13 stupňů. Při více stupních je možné dosáhnout více IPC (instrukcí za cyklus), ale méně stupňů zase snižuje penalizaci při špatné predikci větvení.

V ohledu větvení má A7 novější prediktor a poskytuje výrazně lepší výsledky.

S tím souvisí také lepší algoritmy prefetcheru (slouží přednačtení dat do cache). Zároveň byla snížena latence L2 cache (na 10 cyklů).

Na závěr novější výrobní proces (40nm/28nm) poskytuje proti staršímu (65nm/45nm) nižší spotřebu a možnost vyšších taktů.

Porovnání procesorů je založeno na zdrojích (2), (3) a (4).

Z pohledu procesoru tedy výrazně vítězí Raspberry Pi 2.

Další rozdíly v úložišti, paměti, počtu USB portů a podpoře Windows for IoT hovoří pro Pi.

Jediné výhody BeagleBone jsou v PWM modulu, což lze řešit samostatným čipem a získat tak více PWM výstupů, nebo v hodinách reálného času.

Pro vývoj tedy zvítězila platforma Raspberry Pi.

4.3 Software

Kapitola o software bude rozdělena na dvě části. V první části bude diskutován výběr implementačního jazyka a v druhé části bude stručně popsáno vývojové prostředí pro implementaci systému.

4.3.1 Požadavky na programovací jazyk

Na programovací jazyk je kladeno několik technologicky protichůdných požadavků. Prvním požadavkem je, aby byl vývoj co nejrychlejší. Pod rychlost vývoje je možné zařadit náročnost nastavení vývojových nástrojů a potom samotnou tvorbu kódu. Obvykle platí, že čím méně kódu je potřeba pro vytvoření funkcionality, tím je rychlost vývoje vyšší. Není to ale jen o počtu řádků kódu.

Programovací jazyk by měl mít přehlednou syntaxi, ve které se ihned vyzná i vývojář, který vidí jazyk poprvé. Vhodné jsou i jazyky nebo doplňkové nástroje, které jsou schopné ohlídat čistotu kódu z pohledu pojmenování proměnných, velikosti funkcí, členění a pojmenování souborů aj.

Mimo to je vhodné mít pokročilé integrované vývojové prostředí (IDE), které nabízí našeptavač funkcí (Intellisense), integruje debugger, hlídá syntaxi už ve fázi psaní kódu a nabízí případně další doplňkové funkce. Toto dokáže podstatně urychlit vývoj.

Mimo produktivity při psaní kódu a vzhledem k požadavku na co nejnižší odezvu systému, je vhodné zvolit jazyk, který je pokud možno nejrychlejší. Z pohledu rychlosti výsledného kód obvykle platí, že vedou jazyky kompilované do nativního kódu, následované jazyky s virtuálním strojem a just-in-time kompilací a na posledním místě jsou interpretované jazyky. Pro účely rychlého porovnání jazyků z hlediska rychlosti budou využity výsledky z (5).

Pro srovnání byly zvoleny jazyky, které jsou na platformě Pi využity nejčastěji (toto lze zjistit dle aktivity projektů na Githubu), a to C++, C#, Javascript, Python a také byl navíc zařazen jazyk, která získává v poslední době rychle na popularitě – Go.

4.3.2 C++

Jazyk C++ vznikl již v roce 1983, jako rozšíření jazyka C. Jedná se o objektově orientovaný jazyk s manuální správou paměti. Manuální správa paměti je jeho výhodou i nevýhodou. Pro vývojáře je to více práce muset spravovat paměť a je třeba pečlivosti, aby to bylo provedeno správně a nedocházelo k únikům. Ty mohou být pro aplikaci běží non-stop poměrně kritické.

Výhoda tohoto vyspělého jazyka je v jeho rychlosti. Jedná se o nejrychlejší jazyk současnosti z pohledu výsledného kódu.

Pro C++ jsou k dispozici knihovny urychlující vývoj na Pi. Jde dosáhnout i nejvyšší rychlosti práce s GPIO piny (nejvyšší frekvence).

Nevýhodou je výrazně vyšší množství kódu nutné pro vytvoření aplikace. Ač na úrovni hardware se může jednat o skvělou volbu, API implementovaných modulů by zabralo velké množství času.

Existuje velké množství vývojových prostředí, která pomáhají s debugováním kódu a urychlují vývoj. Bohužel práce v žádném z nich, není tak komfortní jako pro ostatní jazyky.

4.3.3 C#

C# je alternativou pro C++ a pro Javu od Microsoftu. Jedná se o jazyk běžící na virtuálním stroji s automatickou správou paměti pomocí garbage collectoru (sběrač odpadků). Jedná se o proces na pozadí, který v případě potřeby odstraní z paměti objekty, který už nejsou dále využívány. Více informací o funkci garbage kolekce a jazyku lze nalézt v (6).

V jazyce není třeba ručně spravovat paměť. Lze očekávat méně problému už v rané fázi vývoje aplikace. Výsledný kód je přibližně o 20% pomalejší než C++.

Pro vývoj je k dispozici opět velké množství knihoven. Pro běh na Linuxu je možné využít framework Mono, na platformě Windows for IoT je možné použít .NET Framework od Microsoftu.

Oproti C++ je práce s GPIO pomalejší ve většině případu asi o 60%, což je poměrně výrazná ztráta daná přístupem na úrovni zápisu do souboru.

Množství kódu je oproti C++ potřeba asi poloviční. Což výrazně urychluje vývoj aplikace. Pro tvorbu API je k dispozici knihovna WebAPI, která umožňuje velmi rychlý vývoj aplikačního rozhraní na bázi REST služeb.

Z vývojových prostředí je nejpoužitelnější Visual Studio, které patří mezi nejlepší vývojová prostředí, a umožňuje i přímé propojení na Raspberry a tím více urychluje vývoj.

4.3.4 Javascript

Javascript vznikl na bázi ECMAScriptu (norma). Původně byl uveden jako interpretovaných jazyk pro skriptování webových stránek. Tato věta ale přestala platit s rozvojem interaktivních webů a nástupem moderních prohlížečů. Jedna z hlavních implementací v rámci projektu V8 (javascriptové jádro prohlížeče chrom) je implementace běžící na virtuálním stroji s just-in-time kompilací.

Díky V8 byl javascript také nakonec přenesen i mimo prostředí prohlížeče a umožňuje tak multiplatformní vývoj. Projekty, které využívají jádra V8 jsou Node.js a io.js. Oba dva jsou vzájemně kompatibilní.

Správa paměti je v Javascriptu automatická, řešená pomocí garbage collectoru. Mimo automatickou správu paměti má Javascript ještě dynamické typování. Tato kombinace může velmi urychlit psaní kódu, ale také přináší určitá rizika a klade nároky na znalosti vývojáře a jeho pečlivost.

Z hlediska rychlosti kódu patří mezi rychlé jazyky, ale je přibližně 2-3x pomalejší než C++. Při vývoji ale odpadá nutnost kompilace kódu před spuštěním aplikace.

Pro Javascript existuje asi nejvíce knihoven na vývoj pod Pi. Výběr je tedy opravdu velký. Jazyk je podporovaný jak na Linuxu, tak na Windows for IoT.

Rychlost práce s GPIO piny odpovídá C# a platí zde přesně totéž.

Množství kódu se oproti C# děle o něco zmenšuje, ale rozdíl již není tak výrazný. Vzhledem k dostupných knihovnám lze velice rychle tvořit nejen aplikace, ale také API.

Vývojových prostředí je velké množství - mezi nejlepší patří WebStorm a Visual Studio. Při využití Visual Studio je opět možnost rychlého nasazení přímo na Pi.

4.3.5 Python

Programovací jazyk Python je z roku 1991 a vznikl jako mix iterativního, objektového a funkcionálního jazyka. Jedná se o dynamický interpretovaný jazyk. Automatická správa paměti je pro interpretované jazyky téměř pravidlem. Jazyk nemá virtuální stroj, takže lze očekávat výrazné dopady na rychlost výsledného kódu oproti ostatním srovnávaným jazykům.

Rychlost výsledného kódu je 20-40x pomalejší než pro kód C++. Toto je opravdu velká daň za využití Pythonu, se kterou bohužel nejde příliš udělat.

Knihoven existuje velké množství. Pro starší verzi Pi byl Python nejpoužívanějším jazykem. Existuje i podpora pro Windows for IoT.

Rychlost práce s GPIO dosahuje frekvence asi 10x nižší než při využití C++. Není tedy příliš možné vyvinout extrémně rychlé aplikace.

Množství napsaného kódu je ale nejmenší ze všech jazyků. Úspora je dalších 30% oproti Javascriptu. Je potřeba ovšem zhodnotit, zda se tato úspora vzhledem k rychlosti výsledné aplikace vyplatí. Pro nasazení v cloudu, kde se platí za čas procesoru, bude cena úměrně vyšší.

Z vývojových prostředí lze zmínit Netbeans a Visual Studio (je třeba nainstalovat Python Tools for Visual Studio). Ve Visual Studiu je dostupná možnost přímého nasazení na Pi.

4.3.6 Go

Go nejmladším jazykem v porovnání (rok 2007). Jde se o staticky typovaný jazyk odvozený z C. Provedené změny usnadňují práci, zjednodušují kód a zvyšují bezpečnost. Syntaxe je velmi podobná interpretovaným jazykům, ale jedná se o jazyk s kompilací do nativního kódu.

Správa paměti v Go byla vždy automatická. Nejprve byl využit automatický reference counting a nakonec tvůrci implementovali garbage collector.

Rychlost je jedním z nejdůležitějších aspektů Go. Překlad je prakticky instantní, což rozhodně neplatí pro ostatní nativní jazyky a v rychlosti kódu zaostává oproti C++ o 10%, ale v některých situacích dokáže být i rychlejší.

Už v tuto chvíli existují knihovny pro práci s Pi. Není jich takové množství jako v ostatních jazycích, ale dokáží výrazně urychlit práci.

Rychlost práce s GPIO zaostává oproti C++ o 10%, takže v tom ohledu patří mezi nejrychlejší.

Množství kódu odpovídá jazyku Python. Je třeba si ovšem zvyknout na jiný způsob programování. Některé chybějící koncepty (z důvodu jednoduchosti jazyka) vedou občas i k více kódu pro danou část.

Vývojová prostředí jsou prozatím jednodušší. Nejlepší je v době psaní podpora v rámci WebStorm. Visual Studio podporu nemá. Nasazení aplikace do Pi musí být řešeno skriptem.

4.3.7 Výběr jazyka

Vzhledem ke shrnutým výhodám byl pro vývoj hlavních částí zvolen jazyk Go. S tím, že architektura bude vytvořena tak, aby umožnila použít i jiné jazyky. Go bude využit hlavně pro práci s hardwarem a pro vytvoření služeb.

Pro aplikaci bude využit hlavně Javascript, který je multiplatformní a má lepší knihovny pro tvorbu frontend aplikací.

4.3.8 Vývojové prostředí pro Go a Javascript

Pro vývoj v Go a Javascriptu bylo vybráno prostředí WebStorm. Obsahuje pokročilý editor kódu, debugger, automatizaci testů, generátory projektů a podporu pro velké množství jazyků.

Go je do prostředí třeba přidat pomocí pluginu, který je aktivně vyvíjen. Občas se v něm objeví chyby, které jsou ale během jednoho nebo dvou dní opraveny.

Pro verzování kódu byl zvolen Git s hostováním na Githubu. Projekty jsou dostupné pod organizací Housedroid.

Dále byl vývojový proces doplněn o systém kontinuální integrace a to konkrétně drone.io.

4.3.9 Vývojové prostředí pro návrh obvodů

Pro projekt nejprve byly otestovány prostředí KiCAD, Upverter a Circuitlab.

Circuitlab se dal využít pouze na vysokoúrovňový návrh a jako simulátor obvodů a nedal se tedy využít pro návrh desek plošných spojů. Z tohoto důvodu byl vyřazen.

KiCAD a Upverter jsou oba dostupné zdarma a umožňují návrh DPS. Po vyzkoušení obou dvou bylo jasné, že prostředí Upverteru je modernější jednodušší a přívětivější. Práce v něm je navíc mnohem rychlejší. Proto byl zvolen jako výchozí editor schémat a DPS pro projekt.

V následujících odstavcích budou popsány základní funkce Upverteru.

Upverter je editor schémat a DPS, který běží v prohlížeči. Tj. bez stahování a instalace. Na učení je velmi jednoduchý. Obsahuje i vestavěného výukového průvodce, kde se uživatel naučí ovládat program při tvorbě Bluetooth komunikátoru.

Pro snížení počtu chyb při vývoji obsahuje real-time kontrolu designových chyb, nástroje na real-time spolupráci a revize.

Obsahuje rozsáhlou knihovnu součástí a zároveň umožňuje pro chybějící součástí jejich rychlé vytvoření. Všechny informace o součástce jsou vyhledány v databázi octoparts a je připraven návrh pro manuální dokončení.

Každá změna ve schématu se automaticky promítá do návrhu DPS a zároveň jsou všechna spojení na DPS hlídána proti schématu.

Editor umožňuje automatické verzování a návrat k původním verzím.

Výsledné soubory s návrhem je možné odeslat ihned do několika různých výroben nebo je stáhnout a zajistit výrobu osobně.

5 ARCHITEKTONICKÉ PRINCIPY A POSTUPY

V této kapitole budou rozebrána rozhodnutí a důvody vytvoření architektury systému.

5.1 Aplikační prostředí

Systém bude postaven na odlehčené verzi Linuxu s možností převodu na Windows for IoT. Při zahájení implementace Windows for IoT nebyl dostupný a nebylo tedy možné zvážit jeho výhody.

Distribuce, která existují v odlehčené verzi jsou CoreOS, Atomic a Ubuntu Snappy. Vzhledem k podpoře Ubuntu Snappy na Raspberry byla proto zvolena právě tato distribuce.

Nad operačním systémem budou vytvořeny aplikační kontejnery pomocí technologie Docker, které usnadňují nasazení a aktualizaci aplikací, a poskytují jednotné prostředí pro vývoj a produkční nasazení.

Více informace o odlehčených operačních systémech a práci s Dockerem lze nalézt na (7).

5.2 Mikroslužby

V posledních letech se začal objevovat pojem mikroslužby. Jedná se o metodu návrhu architektury systému, tak že se vytvoří služby nezávislé na dalších částech aplikace, které se také nezávisle nasazují.

Základním principem je, že jedna aplikace se rozdělí do několika služeb, přičemž každá služba představuje samostatný proces. Tyto služby používají pro komunikaci odlehčené komunikační mechanismy - obvykle komunikují přes HTTP pomocí definovaného rozhraní (např. REST API).

Pro porovnání s klasickým způsobem tvorby aplikace - monolitická aplikace je sestavená jako jedna jednotka. V případě enterprise aplikace se jedná většinou o vrstvou uživatelského rozhraní, business logiky a datové vrstvy. U monolitické aplikace můžeme tyto části rozdělit mezi serverem a klientem. S nástupem cloudu se ale začaly objevovat problémy se škálováním a nasazením monolitických aplikací. Problémem v případě rozsáhlého nasazení je provedení změn v aplikaci.

Každá změna nebo úprava vyžaduje, aby celá aplikace byla znovu sestavena a nasazena. Ve spoustě případů je třeba ji znovu nasadit i na desítky serverů, což může vést ke snížení dostupnosti nebo k ovlivnění rychlosti odezvy aplikace.

A právě tyto problémy vedou k rozvoji mikroslužeb. Mikroslužba je samostatně nasaditelná škálovatelná jednotka, která poskytuje jednoznačné API pro práci se konkrétním business problémem. Tím, že řeší pouze jeden problém, má přesně stanovené hranice modulu a umožňuje tak, aby jednotlivé služby byly psané v různých jazycích s požitím různých technologií, dokud podporují společný vysokoúrovňový způsob komunikace. Často jsou také jednotlivé služby tvořeny různými týmy. Tento přístup není ničím novým. Vyskytuje se už od dob Unixu.

V současné chvíli není žádná formální definice mikroslužby a nejsou přesně dané požadavky, co taková služba musí splňovat. Pro názornost bude shrnuto pár základních aspektů, které splňuje většina mikroslužeb.

5.2.1 Rozdělení na komponenty pomocí služeb

Už od počátku vývoje softwaru se vývojáři snažili rozdělit software do komponent, které jsou samostatně udržované a případně samostatně aktualizovatelné. Této podobě komponent se v monolitických aplikacích říká knihovny. Knihovny jsou slinkováné s programem a volány využití jejich funkcí.

U mikroslužeb tyto komponenty nepředstavují knihovny, ale služby samotné. Vzájemné propojení aplikačních celků se poté realizuje buďto pomocí requestu nebo volání vzdálených procedur.

Jak již bylo zmíněno, hlavní výhodou je, že takto vytvořené komponenty je možno samostatně nasazovat a aktualizovat. Nemusí se tedy nasazovat celá aplikace. Tím pádem změna vede k nasazení pouze jedné služby.

Zároveň ale toto rozčlenění má určité nevýhody - konkrétně vzdálené volání procedur nebo http komunikace jsou dražší, než když voláme funkci v rámci programu.

5.2.2 Návrh endpointu

V případě návrhu mikroslužby je cílem, aby se samotná služba chovala v podstatě jako linuxová pipe. Má dobře definované vstupní a výstupní rozhraní a dělá pouze jednu věc. A tu věc musí umět dělat dobře.

Jako vstupní API se obvykle používá REST rozhraní. Dále hodně prosazovaný způsob, je využití message queue. Konkrétním příkladem message queue je ZeroMQ a RabbitMQ. Ve většina případů platí, že čím lehčí varianta tím lépe.

U vytvořených endpointů dochází k situaci, že se obvykle zpracovává větší blok, než jaký by byl zpracován při klasickém volání funkcí. Předávají se rozsáhlejší datové struktury, než jaké jsou použity v parametrech volání funkcí.

5.2.3 Decentralizovaná data a přístup

Každá služba má svůj vlastní kontext. Tento kontext nejčastěji používá samostatnou databázi a nikoli jednu společnou jako u monolitických aplikací. Takový návrh je tedy vázaný na kontext. Opět díky tomu můžeme využít pro každou službu jinou technologii, která se hodí pro daný účel. Vzhledem ke škálovatelnosti se využívá moderních noSQL databází.

5.2.4 Návrh předcházející chybám a vysoká dostupnost

Rámci návrhu služby je třeba myslet na to, že kdykoliv může dojít k výpadků služby. Vhodné je tedy využívat redundance, automatického detekování stavu služby a automatickému zotavení. V případě výpadků jedné instance služby, by mělo ihned dojít k využití instance druhé. Dále by měly existovat procesy, které automatizují detekci chyby a zajistí zotavení. Tento přístup vede k vytvoření sofistikovaných monitorovacích systémů.

5.2.5 Evoluce služeb

Problémem u mikroslužeb je, že služba je nasazována nezávisle na ostatních. To způsobuje, že není zaručeno, že všichni využívají stejnou verzi komunikačního protokolu. Každá služba by měla poskytovat verzování jejího API již od samého začátku. Bez něj by nebylo možné dosáhnout zpětné kompatibility. Zároveň poté odpadají starosti o to, zda všichni konzumenti podporují stejnou verzi API.

5.2.6 Vliv na složení týmu

U monolitických aplikací se často využívá přístup, že jednotlivé části aplikace jsou vyvíjeny různými specializovanými týmy - odděleně týmem designérů uživatelských rozhraní, vývojáři business logiky, databázovými specialisty.

Mikroslužby jsou ale dodávány jako jednotlivý byznys celek. Proto je třeba, aby týmy byly složené ze všech možných specilizací. To samozřejmě může být i v případě monolitických aplikací, ale běžně to není pravidlem.

5.3 Komunikace a propojení služeb

Jedním ze základních pilířů mikroslužeb je správné propojení a komunikace mezi jednotlivými službami. Z podstaty systémů je vyžadováno, aby komunikace probíhala v reálném čase. Pro některé částí je vyžadována garantovaná odezva do určitého času (hard realtime systém), zatímco jinde je možné, aby operace proběhla, jakmile to bude možné (soft realtime systém).

Vzhledem k ceně a nejjednodušší cestě vývojářů je vhodné využít některý z běžných síťových protokolů spíše než průmyslové a sběrnicové systémy. Garantované odezva vyžaduje protokol, který vždy zajistí doručení zpráv a je třeba implementovat i mechanismy pro opakování komunikace, pokud nebyla zpráva doručena nebo akce neproběhla v dostatečném čase.

Časový horizont pro provedení akce a i případná opakování včetně výpočetní částí a komunikace je stanoven na 100ms. Tato hodnota je vědecky ověřená doba, po kterou člověk vnímá událost jako instantní. (1) Cokoliv nad tuto hranici je vnímáno uživatelem jako událost s prodlevou a je v rámci systému netolerovatelná, zatímco vše pod tuto hranici je vnímáno prakticky stejně.

Pokud nedojde k úspěšnému provedení akce, je třeba akci opakovat. Základní nastavení systému jsou 3 opakování. Pokud dojde k vyčerpání těchto opakování, spustí se automaticky diagnostika uzlu, detekce chyby, případné zotavení a informování správce či uživatele v závislosti na závažnosti závady.

Pokud je třeba ve 100ms provést akci a zároveň zabezpečit 3 možná opakování zůstává tedy na jedno opakování akce 25ms. Tím je v drtivé většině případů, kdy nedojde k vážné závadě zajištěno provedení akce na hranici vnímání.

U operací, kde není třeba dosáhnout vždy instantního provedení je stanoven čas na provedení akce 100ms. Tím bude v běžné situaci dosaženo instantní odezvy a jen v případě mírné závady bude vnímáno určité zpoždění.

Tyto časy umožňují použít běžně dostupné komunikační systémy známé z PC. Základní propojení komponent bude realizováno pomocí technologie Ethernet případně pomocí Wifi 802.11n.

Tím je možné využít zároveň protokolu TCP a protokolů vyšší vrstvy nad ním postavených. V rámci systému budou využívány protokoly vlastní implementace v rámci Message Queue, ale také webové komunikační protokoly jako http a WebSockets.

Důležitou součástí komunikace mezi jednotlivými jednotkami, ale také mezi službami samotnými je zabezpečení. Vzhledem ke snadnosti implementace, spolehlivosti a zároveň dospělosti je uvažováno hlavně zabezpečení pomocí protokolu SSL/TSL. Tím je možné dosáhnout bezpečné komunikace s minimálním nárokem na implementační cenu a zároveň poskytuje uživateli dostatečnou úroveň bezpečnosti.

5.4 Škálovatelné protokoly a Mangos

Cílem škálovatelných protokolů je poskytnout vývojáři abstrakci nad použitou komunikační technologií. Dříve museli být aplikace psané na míru komunikačnímu systému (např. pro IP sockety).

S rozvojem SOA (Service Oriented Architecture) se dostávaly do popředí systémy pro zasílání zpráv, které dokázaly pracovat s různými technologiemi současně. Nejprve nacházely uplatnění v Enteprise systémech jako samostatné serverové aplikace, kde bylo potřeba vynaložit určité prostředky na správu a údržbu takového řešení.

S nástupem mikroslužeb bylo žádoucí využít komunikační metody podporující různé protokoly, ale vyhnout se režii, které představovaly Enterprise systémy. To vedlo ke vzniku ZeroMQ – počin autorů AMQP (základ pro enterpise message bus), kteří chtěli poskytnout jednodušší a robustnější řešení.

Základem je jednoduchá knihovna, která pouze vytváří abstrakci nad komunikačními protokoly a poskytuje vývojáři systém podobný BSD socketům. Nad ním si vývojář obvykle implementuje vlastní komunikační protokol.

Zároveň všechny potřebné prostředky a mechanismy jsou distribuovány s aplikací. Tím odpadá nutnost instalovat na server dedikovanou aplikaci pro komunikaci. Tento způsob přináší velké množství výhod a různorodosti použití.

Vzhledem ke sporům ohledně budoucnosti vývoje ZeroMQ, kterými bylo v první řadě zabezpečení[2] a následně také nízké množství podporovaných protokolů, vedlo k odštěpení části vývojářů a vytvoření systémů nanomsg. (2)

Nanomsq je statická knihovna, napsaná v jazyce C. Celé API je koncipováno jako tzv. zero-copy[3]. Kombinace jazyka C a zero-copy API poskytují extrémně nízké latence na transportní vrstvě. Součástí je široká nabídka knihoven pro jiné jazyky, tak aby mohli pracovat s nanomsg bez jakýchkoliv problémů při vývoji nebo nasazení.

S rozšiřováním jazyka Go pro vývoj backendu (pro připomenutí hlavními výhodami jsou rychlost, jednoduchost vývoje konkurentních aplikací a čitelnost kódu) se hledala vhodná knihovna pro tvorbu spolehlivých škálovatelných aplikací. Go samotné obsahuje API pro práci se sockety, ale právě pro škálovatelné aplikace je třeba vyvinout velkého úsilí pro vytvoření takové aplikace. A to právě usnadňují škálovatelné protokoly jako ZeroMQ a nanomsg, a komunikační vzory které poskytují.

Pro Go jsou samozřejmě k dispozici knihovny, které umožňují využití obou technologií, ale bohužel nesplňují jeden z konceptů používaných při vývoji pod Go. Vývojář by se měl snažit vyhnout se načtení nativních knihoven, pokud je to možné. Odměnou za tento přístup je jednodušší ladění a nasazení jediného binárního souboru oproti více. Proto vznikla reimplementace nanomsg v čistém Go. Jmenuje se Mangos. (3)

Mangos nabízí idiomatickou implementaci a rozhraní v Go. Používá zároveň stejné komunikační vzory jako nanomsg, se kterým je kompatibilní. Jmenovitě se jedná o pub/sub, pár, req/rep, pipeline, sběrnice, survey. Podporuje také stejný model zásuvných transportních protokolů umožňující jednoduché rozšiřování (v základu je podporování TCP, IPS, inproc, websocket a TSL). (4)

[2]ZeroMQ neposkytuje možnost bezpečné komunikace. Je nutno řešit na úrovni aplikace.
[3]Zero-copy znamená, že se předávají pouze reference na objekt a nedochází k jeho překopírování.

Jednou z výhod implementace základního protokolu byla, že nevyužívá garbage collector[4]. Nicméně toto neplatí v rámci Go a je nutné za to zaplatit určitou cenu. Mangos využívá object pooly, aby omezil spuštění garbage collectoru. Ten není možné vypnout, proto jsou využity postupy, aby jeho využití bylo minimalizováno. Je to důležitý základ pro systémy vyžadující vysokou propustnost. V tom je Mangos srovnatelný s nanomsg. Oba poskytují velmi dobé výsledky při testech propustnosti systému.

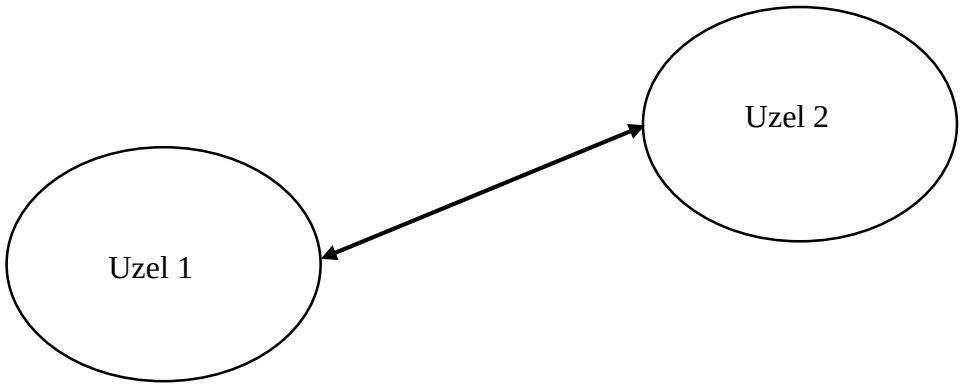
Mangos má rozhraní odpovídající socketům. Díky tomu je rychlost osvojení pro vývojáře znalé socketům velmi vysoká.

5.4.1 Komunikační vzory

Tato kapitola nabízí základní přehled komunikačních vzorů, které jsou dostupné v rámci Mangos a nanomsg.

5.4.1.1 Pár (obousměrná komunikace)

Pár (anglicky pair) představuje základní vzor poskytující obousměrnou komunikaci mezi dvěma koncovými body (jeden jednomu). Komunikace je asynchronní neblokující.

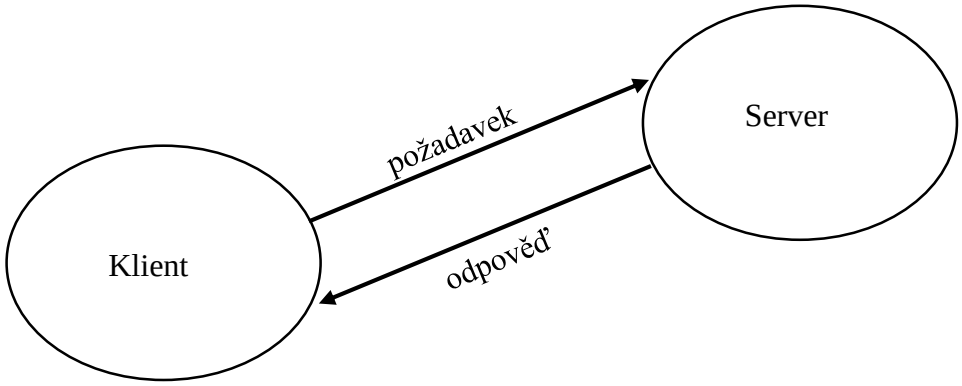


Obrázek 5.1 – komunikační protokol „pár“

[4] Sběrač odpadků, automatický správce a uklízeč paměti

5.4.2 REQREP (klient požaduje, server poskytuje)

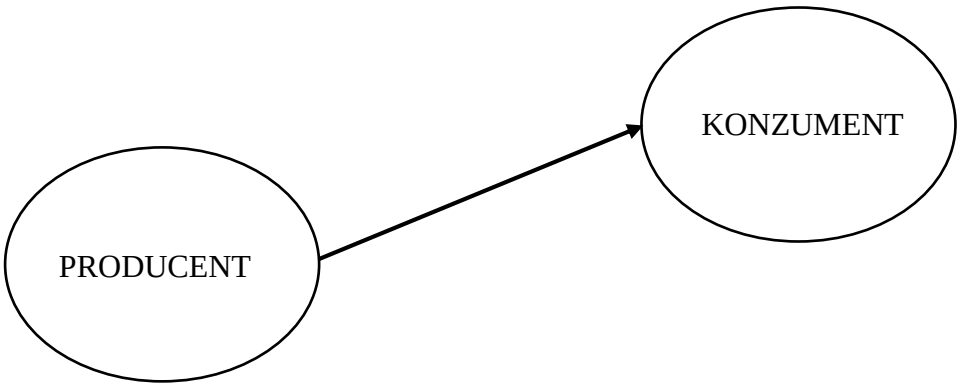
Tento vzor je obvykle použit pro tvorbu bezstavových služeb pro zpracování požadavků. Klient odešle požadavek na server. Server požadavek přijme, zpracuje a odešle zpět odpověď. Komunikace je velmi podobná základnímu požadavku protokolu http.



Obrázek 5.2 – komunikační protokol „reqrep“

5.4.3 Pipeline

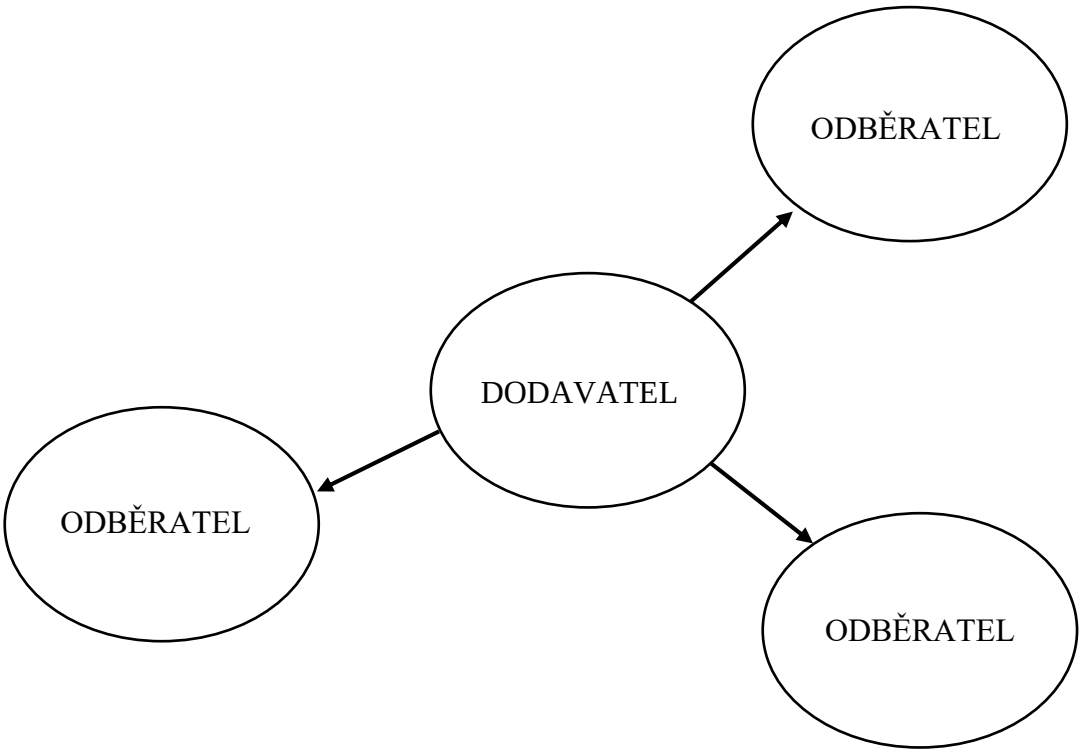
Pipeline, neboli roura, je vzor určující jednosměrný tok dat. Podobně jako roury v Unixu umožňuje zřetězit několik různých služeb (procesů) pro zpracování požadavku. Producent odešle zprávu, která je dále zpracována konzumentem. A takto může být vybudován celý řetěz generující cílovou odpověď.



Obrázek 5.3 – komunikační protokol „pipeline“

5.4.4 PUBSUB (Hromadné zasílání zpráv odběratelům)

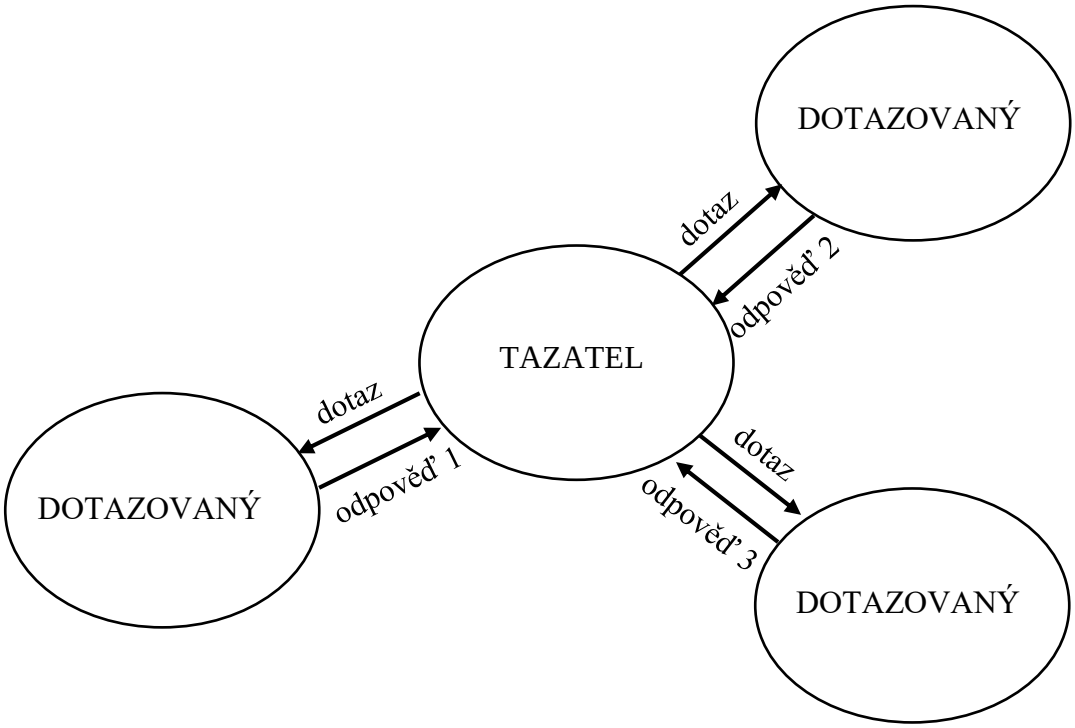
PUBSUB umožňuje producentům/dodavatelům zaslat stejnou zprávu nula a více odběratelům. Odběratel se přihlásí na odběr konkrétního tématu, což mu umožňuje dostávat pouze komunikaci, která je pro něj relevantní. Zároveň má každý odběratel možnost přihlásit se k více dodavatelům pro příjem zpráv.



Obrázek 5.4 – komunikační protokol „pubsub“

5.4.5 SURVEY (průzkum)

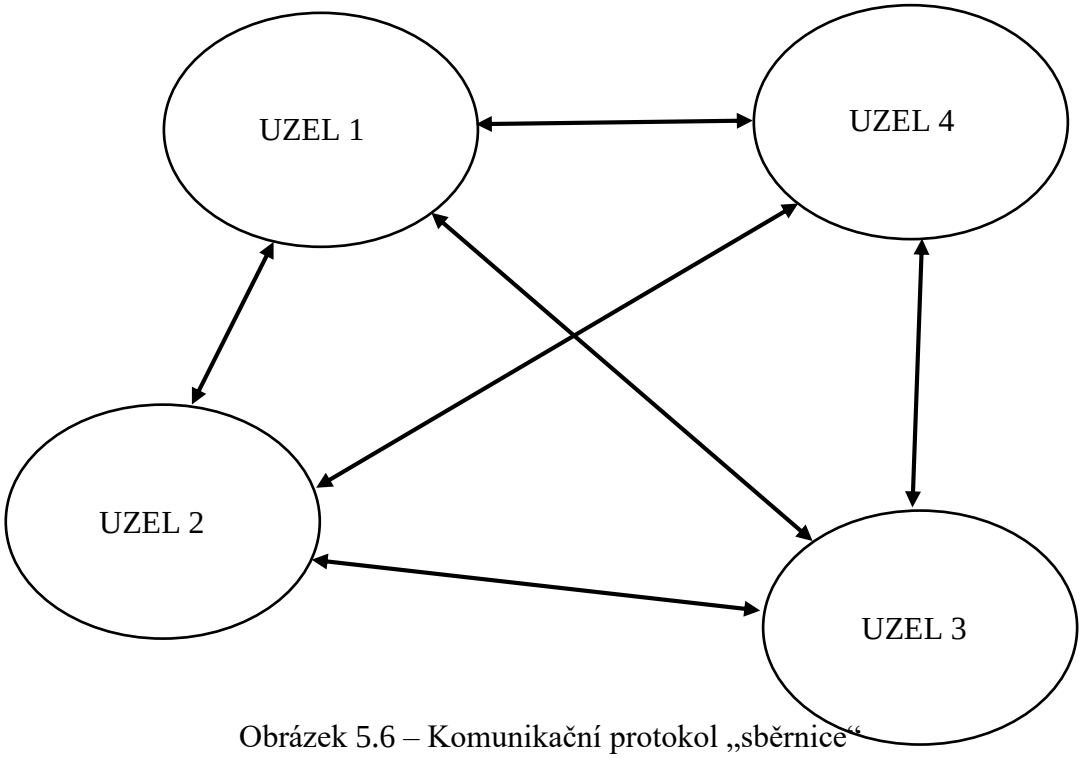
Tento vzor je jedním z nejužitečnějších a přitom se nevyskytuje v příliš mnoho message queue knihovných (např. ZeroMQ tento vzor nepodporuje). Je velmi podobný vzoru PUBSUB. V rámci průzkumů je odeslána zpráva tazatele všem respondentům/dotazovaným ve skupině. Každý respondent zprávu zpracuje, což může být i různým způsobem – pro různé služby různá implementace reakce, a odešle svoji odpověď zpět tazateli. Tento vzor umožňuje rychlou tvorbu služeb, kde je třeba se dotázat na stejnou věc několik různých služeb a následně agregovat výsledky. Velkou výhodou je také možnost stanovení časového okna, do kdy musí být zaslána odpověď.



Obrázek 5.5 – komunikační protokol „survey“

5.4.6 BUS (sběrnice)

Posledním použitelným komunikačním vzorem je sběrnice. Ta umožňuje, aby zpráva odeslaná kterýmkoliv uzlem, byla doručena všem zbývajícím uzlům. To umožňuje tvořit aplikace, které potřebují vzájemně sdílet svůj stav. Příkladem může být generování sítě dostupných sousedů.



Obrázek 5.6 – Komunikační protokol „sběrnice“

5.5 Uložení dat

Většina dat systému bude držena v paměti a je možné aktuální informace rychle sestavit podle čtení ze senzorů. Jsou však i místa, kde se očekává nutnost perzistentního uložení dat. Jde o centrální uložení konfigurace modulů (konfigurace bude moci být uložena i lokálně), uložení monitorovacích dat a logovacích zpráv o stavu systému, a o uložení záznamů z kamerového systému.

5.6 Rozdíly použití relačních databází a noSQL

SQL databáze se používají již od 80 let jako datové úložiště. Jsou postaveny nad relačním modelem a tabulkovém schématu. Právě tabulkové schéma je třeba na začátku vytvořit a následně diktuje a hlídá podobu dat.

Oproti tomu noSQL databáze jsou tzv. schema less. Není potřeba vytvářet žádnou předpřipravenou strukturu a je možné přímo ukládat data. NoSQL databáze navíc ve spoutě příkladů umí skvěle pracovat s JSONem, který bude rozebrán v následující kapitole a bude použit jako základ systému.

Jedním z rozdílů je, že SQL databáze jsou obvykle nasazené na jednom výkonném stroji. NoSQL databáze naopak je rozdělena pomocí shardu na několik méně výkonných strojů.

Většina noSQL nesplňuje základ SQL databází – ACID (atomicita, konzistence, izolovanost, trvanlivost). To ovšem pro systém domácí automatizace není vyžadováno vzhledem k jednoduchosti dat, která je třeba ukládat.

Porovnávány mezi sebou byly databáze Postgres, který představuje relační databázi, která má i funkce noSQL databáze, umí ukládat JSON a pracovat s ním, a MongoDB jakožto jeden z nejčastěji používaných zástupců noSQL databází.

5.7 JSON

JSON je zkratka pro Javascript Object Notion. Je velmi výhodný pro komunikační protokol, který je lehce parsovatelný (strojově zpracovatelný) a i lehce čitelný člověkem například při debugování. Je založený na podmnožině programovacího jazyku Javasript podle standardu z prosince 1999. JSON je nezávislý na jazyku – lze ho využít v C, C++, C#, Javě, Javascriptu, Perlu, Pythonu a dalších.

JSON je postaven na dvojici struktur:

1. Na kolekci párů jméno/hodnota. To jde v různých jazycích realizovat jako objekt, záznam, struktura, slovník, hashovací tabulka, list s přístupem dle klíče nebo asociativní pole.
2. Seřazený seznam hodnot. Ve většině jazyků jde o pole, vektor, seznam nebo sekvenci.

Toto jsou univerzální datové struktury, která jsou poskytovány téměř všemi dostupnými jazyky. (12)

II. PRAKTICKÁ ČÁST

6 KOMUNIKAČNÍ PROTOKOL

Komunikace bude postavená na bázi protokolu JSON. Základní tělo zprávy bude shodné pro všechny zprávy a bude mít vyhrazenou část pro data specifická pro modul.

Základním atributem zprávy je identifikace odesilatele. Každý modul bude mít přidělené své ID, které je v rámci systému jednoznačné. ID je přidělitelné při výrobě modulu a je na modulu vytištěno například QR kódem, což umožňuje později zrychlit konfiguraci celého systému mobilním telefonem.

Každá zpráva má svůj čas vytvoření. Časovou známkou se předejde situaci, kdy by novější data byla přepsána staršími, která z nějakého důvodu byla zpožděna.

Pro možnost propagace zpráv a rychlé nasměrování do modulu, který má zájem o danou zprávu, je dále definován typ zprávy. Typ zprávy může být „čtení ze senzoru“. Tím je možné odlišit zprávy, kdy je třeba nastavit cílové hodnoty a kdy se pouze čte aktuální stav.

V rámci modulu je třeba mít vždy označenou oblast, kde se modul nachází. Pro specifikaci místa bude použita stromová struktura určující oblast a její podoblasti. Měla by být v podobě cesty, která se používá i v souborovém systému. Tím bude lehce a rychle parsovatelná.

Ikdyž je možné podle ID vždy dohledat typ modulu v centrálním úložišti konfigurace, je typ modulu vždy uveden ve zprávě. Tím se opět zrychlí směrování zpráv za cenu pouze menšího navýšení velikosti zprávy.

Vzorová zpráva:

```
{
        "area" : "/house/living-room",
        "created" : "30-12-2014T13:50",
        "message-type" : "sensor-reading",
        "module-id" : "adf465fs",
        "module-type" : " sensor/temperature",
        "data" : {
                "action" : "temperature-reading",
                "value" : "22.00"
```

```
        }
}
```

Rozbor atributů zprávy a hodnot pro vzorový případ:

area	Představuje místo, kde bylo čtení provedeno. Hodnota značí, že modul je umístěný v domě v obývacím pokoji.
created	Reprezentuje časovou známku ve formátu UTC.
message-type	Označuje typ zprávy. V tomto případě jde o čtení ze senzoru.
module-id	Jednoznačná identifikace modulu. V tomto případě jsou použity hashe.
module-type	Označení o jaký druh modulu se jedná – pro vzorovou zprávu teplotní senzor
data	Objekt nesoucí data ve specifickém formátu modulu.

Protokol samotný nemá verzování. Všechny atributy musí být rozpoznatelné všemi moduly kromě objektu nesoucího konkrétní data.

7 MODUL ROUTER

Páteří celého systému je modul router. Tento modul slouží ke směrování zpráv mezi moduly. Každý modul připojený do systému se nejprve registruje routeru, a označí, jaké typy zpráv bude odebírat. Modul má možnost routeru zaslat jakoukoliv zprávu. Pokud není pro daný typ zprávy určený žádný příjemce, je zpráva zahozena.

Pro zasílání zpráv je použit komunikační vzor pár.

7.1 Stromové struktury

Router si při registraci modulu, na základě definice jeho oblasti, sestavuje stromovou strukturu, která na úrovni uzlu obsahuje typy zpráv a moduly registrované pro jejich odběr.

Pokud daný uzel při registraci neexistuje, je vytvořen. Pokud uzel už existuje, je pouze přidán do seznamu odběratelů nový odběratel.

Při implementaci byla využita abstraktní datová struktura typu strom. Výsledný strom je nevyvážený. Každý uzel má svého rodiče (kromě kořene) a může mít 0-N potomků. V rámci uzlu jsou uloženy odkazy na moduly a případně metody, které mají o daný typ zprávy zájem. Uložení je ve formě hashovací tabulky, která používá typ zprávy jako klíč a hodnotou je seznam registrovaných příjemců zprávy.

7.2 Registrace prostředků

Pro registraci prostředku zasílá modul následující zprávu:

{

"area" : "/house/living-room",

"created" : "30-12-2014T13:50",

"message-type" : "module/registration ",

"module-id" : "adf4dffs",

"module-type" : "dimmer",

"data" : {

"accepted-area" : "/house/living-room ",

"accepted-events" : "light/intensity, light/switch",

```
        }
}
```

Ze vzorové zprávy jde vidět, že probíhá registrace modulu „dimmer“ – „stmívač“. Podle atributu accepted-area se modul registruje pouze pro obývací pokoj a podle accepted-events přijímá pouze zprávy týkající se intenzity světla a případně přepnutí světla.

7.3 Zpracování události

V případě přijetí události proběhne následující proces:

1. Zpráva je přijata a zparsována
2. Na základě oblasti jsou vyhledáni příjemci
3. Pokud příjemce akceptuje daný typ zprávy, je mu zpráva předána
4. Pokračuj dalším příjemcem

Proces vyhledání příjemce:

1. Vezmi kořen jako aktuální uzel
2. Pokud aktuální uzel obsahuje potomka, který je definován cestou, pokračuj bodem 3, jinak vrať aktuální uzel
3. Nastav potomka jako aktuální uzel
4. Pokračuj bodem 2

7.4 Generování událostí

Vytvoření události probíhá vždy v rámci modulu. Jsou vyplněny všechny atributy, doplněna data a zpráva je zaslána na router.

Všechny zprávy jsou registrovaným příjemcům zasílány samostatně. To je z důvodu šetření prostředků komunikační linky. Při využití vzoru „pubsub“ by někteří z příjemců museli zahazovat zprávy. Proto byl využit právě vzor „pár“.

7.5 Bublání zpráv

Daný typ zprávy může být doručen více příjemcům. Zasílání probíhá od nejkonkrétnějšího příjemce až po toho nejobecnějšího. Proces přepínání mezi aktuálními příjemcem je pojmenován jako bublání zpráv.

Nejkonkrétnější příjemce je vždy ten, který přesně odpovídá cestou oblasti a přijatou zprávou. Tomu je zpráva předána nejdříve. Následně je vybrán příjemce ve stejném uzlu (stejné oblasti), který ale přijímá obecnější typ zprávy. Například pro typ zprávy „light/switch“ je zpráva nejprve předána příjemci, který je registrovaný pro událost „light/switch“, následně příjemci, který je registrovaný pro událost „light“ a nakonec pro příjemce, který přijímá všechny události v daném uzlu „*“.

Stejným způsobem se proces opakuje na úrovni rodičovského uzlu od nejspecifičtější události po tu nejobecnější. Tímto způsobem se pokračuje až do kořenového uzlu.

8 MODUL REACTOR

Reactor je základní modul, který poskytuje možnost reakce na změnu stavu systému na základě přijaté zprávy. Je použit jako základ, pro odvození specializovaných modulů reakce na událost. Spolu s routerem se jedná o základní modul systému.

Základní modul funguje na principu „if this than that“. Pro modul jdou nakonfigurovat podmínky, na které má reagovat a pro dané podmínky přiřazena reakce ve formě nové zprávy. Konkrétní případ může být následující – pokud bylo stisknuto tlačítko vypínače a světla jsou vypnuta, přepni všechna světla v místnosti do stavu zapnuto.

8.1 Seznamy a slovníkové tabulky

Pro uložení typů událostí, na které modul umí reagovat, se využívá slovníku/hashovací tabulky. Principem hashovací tabulky je nalezení takové funkce H, která po získání v parametru určité množiny dat vrátí index do tabulky T, kde se taková data nachází. (13)

V rámci tabulky je uložen jako hodnota seznam párů – kritéria a reakční funkce. Reakční funkce představuje odkaz na funkce, která je volána při splnění kritérií.

Kritéria jsou reprezentovány stromovou strukturou a seznamy obsahující trojice ve formátu atribut, operátor porovnání, hodnota. Kritéria mohou být vzájemně spojena pomocí operátorů AND a OR. Tímto způsobem lze vytvořit komplexní strukturu pro definici kritérií.

8.2 Systém reakcí

8.2.1 Registrace reakce na událost

Každá reakce na událost musí být nejprve registrována. Pro registraci události musí být vytvořeno kritérium (což je možné provést v administračním rozhraní) a vybrat funkci, která má být při splnění kritérií zavolána.

Takto vytvořená kritéria jsou uložena v konfiguračním úložišti, a jsou pomocí konfigurační služby načtena Reactorem při jeho spuštění. Při změně konfigurace je vygenerována zpráva informující o změně konfigurace a modul si novou konfiguraci načte znovu z konfigurační služby.

Reakční funkce v základním modulu umožňuje pouze vytvoření nové zprávy se zadanou oblastí, typem zprávy a datovými hodnotami[5]. Pokud je potřeba komplexnější logiky nebo jiných operací, je třeba vždy vytvořit nový modul odvozený od Reactoru.

8.2.2 Vyhledání reakce podle příchozí události

V případě přijetí zprávy je nejprve vyhledána v hashovací tabulce informace o tom, jaká má být volána funkce.

Vyhledávání probíhá na základě typu zprávy. Pokud byla přijata zpráva „light/intensity“, je vyhledán pouze klíč „light/intensity“ a nikoli i pro „light“.

Vedle sebe přitom můžou v tabulce existovat oba dva záznamy. Pro každý je doručena samostatná zpráva a není tedy třeba dále jednu zprávu dělit pro více událostí.

8.2.3 Reakce na událost

Ve chvíli, kdy byl získán seznam reakčních funkcí a kritérií, probíhá postupně pro každou položku seznamu vyhodnocení kritérií. Vyhodnocení probíhá na základě porovnávání atributů zprávy s podmínkami definovanými ve stromě. Výsledkem je pro každý uzel stromu hodnota true/false. Na závěr probíhá redukce vyhodnocených podmínek a je získána výsledná hodnota pro celý strom.

Pokud výsledná hodnota byla true, je zavolána reakční funkce s příslušnými parametry. Obvykle se jedná o celou zprávu, ze které si funkce sama vytáhne potřebná data.

[5] Datové hodnoty určují, jakou operaci je třeba udělat. Např. nastavit intenzitu světel na 100%

9 KONFIGURAČNÍ MODUL

Konfigurační modul slouží k jako centrální úložiště konfigurace. Pro uložení využívá databázi MongoDB a pro tvorbu konfigurace se spoléhá na externí aplikaci. Každý modul připojený do systému, může načíst svoje konfiguraci pomocí svého ID.

9.1 Vytvoření konfigurace

Pro vytvoření konfigurace modulu se využívá externí aplikace (např. správcovský modul). Při instalaci modulů, by vždy měl být zanesen záznam do konfigurační databáze. Tento záznam obsahuje informace o ID modulu, jeho typ, umístění a případně další informace (např. který pin ovládá osvětlení v konkrétní místnosti).

Pro tvorbu konfigurace je určený samostatný endpoint API: /v1/config/

Endpoint je vytvořen jako REST API. Pro vytvoření konfigurace stačí zaslat datovou zprávu metodou POST a po ověření je konfigurace vytvořena.

Dalším způsobem přístupu/vytvoření konfigurace je využití modulu pro přijaté zprávy, který volá funkce REST rozhraní přímo z podfunkce konfiguračního modulu.

9.2 Získání konfigurace

Pro získání konfigurace moduly používají následující zprávu:

```
{
        "area" : "unknown",
        "created" : "30-12-2014T13:50",
        "message-type" : "config/request ",
        "module-id" : "adf4dffs",
        "module-type" : "unknown",
}
```

Ve zprávě je důležité pouze ID modulu a typ zprávy „config/request“. Atributy area a module-type nejsou důležité, protože modulem nemusí být v době žádosti o konfiguraci známé.

Podle ID modulu je v databázi dohledána konfigurace modulu a odeslána žadateli. Žadatel následně konfiguraci uloží i lokálně. Při případném restartu modulu je napřed načtena lokální konfigurace a poslána žádost o rekonfiguraci.

9.3 Rekonfigurace

Rekonfigurace může proběhnout dvěma způsoby. Prvním nejčastějším případem je změna konfigurace ze správcovské aplikace. V takovém případě je vytvořena zpráva pro modul, který je upozorněn, že proběhly změny. Modul následně vyšle zprávu konfiguračnímu modulu a žádostí o novou konfiguraci. Žádost je stejná jako v případě nové žádosti o konfiguraci.

Pro upozornění na rekonfiguraci se využívá následující zpráva:

```
{
        "area" : "/house ",
        "created" : "30-12-2014T13:50",
        "message-type" : "config/changed ",
        "module-id" : "cdredffs",
        "module-type" : "config",
        "receiver-id" : " adf4dffs "
}
```

V tomto případě je navíc obsažen atribut receiver-id, který umožňuje routeru odeslat zprávu přímo na konkrétní zařízení. Tím je modul informován o tom, že si má požádat o novou konfiguraci.

Druhým způsobem je restart modulu nebo jeho zapojení po době nečinnosti nebo poruše. V takovém případě modul nejprve využije lokálně uloženou konfiguraci a zároveň si zažádá o rekonfiguraci. Každá konfigurace má označení revize, podle které je určeno, jestli je třeba přijatou konfiguraci aplikovat.

10 UŽIVATELSKÁ KONZOLE

Uživatelská konzole je aplikace, která je určená pro běh ve webovým prohlížeči a případně i jako aplikace pro tablet, dotykový panel nebo mobil. Aby splnila všechny tyto požadavky, musí být vytvořena jako responzivní.

Responzivní aplikace přizpůsobuje podle rozlišení a typu zařízení své rozložení. Obvykle využívá sloupcové rozložení, když pro různá zařízení je počet sloupců na šířku různý. Responzivní aplikaci jde vytvořit pomocí media queries jazyka CSS.

V této kapitole bude rozebrána podoba konzole a způsob její implementace.

10.1 Požadavky na ovládací konzoli

Základní požadavky na podobu ovládací konzole jsou následující:

1. Responzivní aplikace
2. Komunikující se systémem v reálném čase oběma směry
3. Okamžitá odezva aplikace na provedenou akci
4. Okamžité vykreslení při změně
5. Grafická úroveň v závislosti na výkonu zařízení (např. animace jsou dostupné pouze na tabletu nebo ve webovém prohlížeči)
6. V případě běhu na dotykovém panelu (stacionární ovladač) může být aplikace spuštěna pouze za přítomnosti uživatele

10.2 Komunikace pomocí Websockets

Aby bylo možné dosáhnout real-time komunikace mezi aplikací a systémem, je třeba zvolit vhodný komunikační protokol. Jelikož je požadavek na běh aplikace ve webovém prohlížeči, nabízí se požití technologie WebSockets.

WebSocket je protokol poskytující obousměrný komunikační kanál přes jedno TCP spojení. Protokol je standardizován organizací IETF jako RFC 6455 a WebSocket API je standardizováno ve Web IDL organizací W3C.

WebSocket je implementován na straně webového prohlížeče a webového server, ale může být i použit jinou klientskou nebo serverovou aplikací. Protokol umožňuje vytvořit více interakce mezi serverem a prohlížečem a umožnit serveru, vždy aktualizovat živý obsah

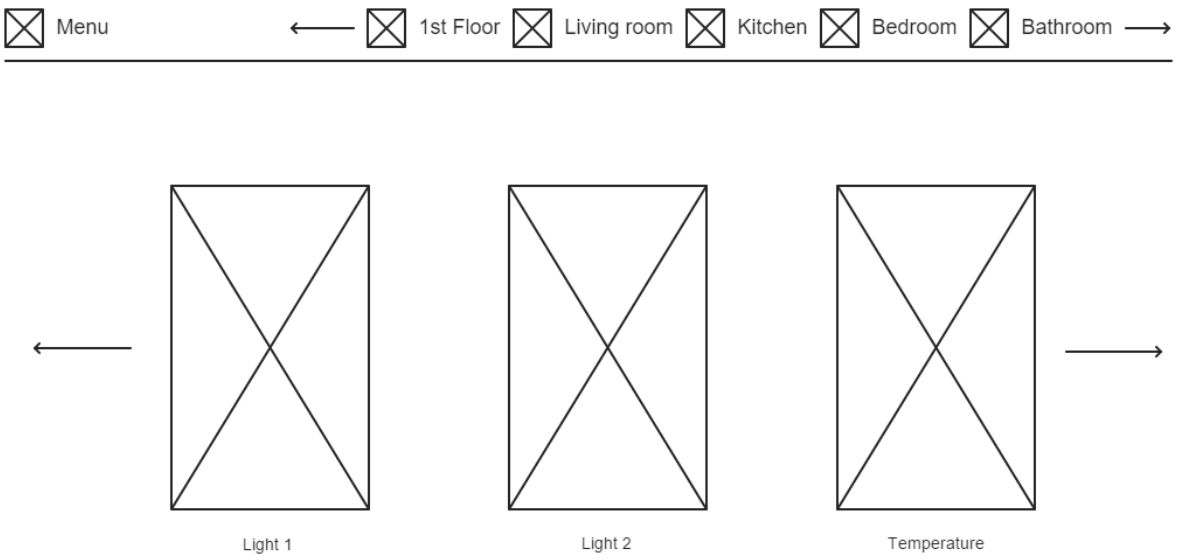
nebo poskytnout základ pro síťové online hry. Server umí zaslat prohlížeči obsah, aniž by byl předem vyžádán. (14)

Pro implementace je využita na straně serveru knihovna pro práci z WebSockets, která je součástí jazyka Go a pro klienta JavaScriptový objekt pro práci s WebSockets.

Vzhledem k rozšířenosti WebSockets není implementován žádný fallback na jinou technologii.

10.3 Wireframe

Wireframe poskytuje základní informace o podobě aplikace a o rozložení ovládacích prvků. Je využit dále pro tvorbu designu a implementaci. Samotné prvky wireframu budou rozebrány v navazujících podkapitolách.



Obrázek 10.1 – Wireframe uživatelské aplikace

10.4 Nabídka místností

V rámci aplikace, nehledě na zařízení, je možné se přepínat mezi jednotlivými místnostmi domu. Výběr místností je součástí menu v horní části obrazovky. Vždy je zobrazen pouze omezený výběr místností. Konkrétní počat závisí na velikosti zařízení a jeho rozlišení.

Místnosti mohou být v případě větších domů seskupeny – například po patrech. Pro menší domy je možné vypsat přímo všechny místnosti. Posuv probíhá tahem z jedné strany na druhou – dle toho jakým směrem se má nabídka posunout. Při ovládání myší jsou zobrazena pomocná tlačítka pro posun.

V závislosti na typu zařízení, jeho umístění a na uživateli nemusí být zobrazeny všechny volby. Toto umožňuje nasazení systému i ve větších domech nebo komerčních objektech, kde je umožněno uživatelům ovládání pouze jejich vlastního bytu či jedné místnosti.

10.5 Výchozí zobrazení

Aplikace při svém spuštění vybere výchozí místnost. Pro stacionární dotykové panely je vždy nakonfigurována výchozí místnost a na tuto místnost se aplikace přepne zpět i po určité době nečinnosti.

Pro mobilní zařízení dochází k pokusu o detekci aktuální místnosti. Přesnost detekce závisí na lokální Wifi sítí a případně na Bluetooth prvcích v okolí. V případě, že se detekce nezdaří, aplikace se přepne na výchozí nastavenou místnost případně na místnost naposledy použitou.

10.6 Zobrazení modulů

Hlavní část plochy aplikace zabírají moduly. Modul je definovaná skupina zařízení, kterou je možné ze zobrazovacího modulu sledovat nebo nastavovat. Příkladem může být modul symbolu žárovky, který po stisknutí rozsvítí nebo zhasne světla.

Množství zobrazených modulů je stejně jako u místností limitováno dle typu zařízení. Pro mobilní telefon je zobrazen pouze jediný modul, zatímco pro tablet jsou to 3-4 moduly v závislosti na velikosti. Pro počítač může být zobrazeno až 5 modulů na jedné obrazovce. Posuv mezi moduly je řešen tahem ze strany obrazovky v části s moduly.

10.7 Využití SVG v uživatelských rozhraních

Pro všechny grafické prvky v aplikace je využíváno SVG. SVG je vektorový formát s tvary definovanými pomocí XML tagů se speciálním významem. Je podporovaný všemi moderními prohlížeči.

Jeho hlavní výhodou je neustálá ostrost při jakémkoliv rozlišení, protože vektorová grafika jde libovolně zmenšovat a zvětšovat. Další výhody jsou možnosti stylování pomocí CSS,

což velmi usnadňuje tvorbu barevných schémat aplikace nebo vytváření animaci nad jednotlivými prvky.

S každým atributem SVG je také možné manipulovat pomocí Javascriptu a nebo k němu přiřadit EventListenery, která zachytávají události nad prvkem a volají související kód.

Pro rychlejší načítání závislostí jsou menší obrázky spojeny do tzv. sprites. Sprites jsou obrázkové mapy, kde konkrétní obrázek je vybrán nastavením výšky, šířky a pozice v mapě. V SVG je možné každému podobrázku přiřadit ID nebo třídu, a odkázat se na ně místo na pozici.

10.8 Stylování

Stylování je provedeno pomocí jazyka SASS. SASS je rozšíření CSS o proměnné, dědičnost, funkce a další chybějící koncepty. Díky tomu je možné vytvářet jednodušeji rozsáhlejší soubory stylů s menším výsledným souborem.

Pro stylování byly použity koncepty BEM – block, element, modifier. Více informací o tomto konceptu lze nalézt na (15).

Pro každý blok je vytvořen samostatný soubor, ve kterém jsou definované příslušné elementy a jejich modifikátory. V principu je v HTML atributu definována třída bloku, pro element je vybrána definice dle elementu a doplněna poslední třída pro modifikátor. Blok a element nastavují základní a společné atributy pro prvky daného typu. Modifikátor je použit pro specifická nastavení. Například blok definuje formulář a element tlačítko. Modifikátorem lze nastavit barvu tlačítka. Tento princip pomáhá vytvořit udržovatelný systém stylů a také jednoduchý redesign.

Pro redesing se využívají změny souborů s definicemi proměnných. Každá barva použitá v stylu by měla být definovaná ve vlastní proměnné, a zároveň by měli být vytvořeny zástupné proměnné pojmenované podle elementů a účel. Např. proměnná green nastavena na hodnotu #00FF00 a proměnná button-active-color nastavená na proměnnou green.

Soubory jsou vždy při změně přeloženy překladačem jazyka SASS, použita automatická prefixace pro zajištění kompatibility se staršími prohlížeči, spojeny do jednoho výstupního souboru, minifikovány a následně zapsány na cílové místo s mapou pro debugování.

10.9 React.js komponenty

React.js je Javascriptová knihovna od Facebooku. Nejedná se o MVC (model-view-controller) framework, ale pouze o knihovnu pro tvorbu komponent. Tyto komponenty jsou tedy vhodné pouze pro view (zobrazovací část aplikace).

V rámci Reactu se nepoužívají templaty (šablony) nebo directivy. Místo toho je celé rozhraní tvořeno rozdělením do komponent. Komponenty jsou tvořeny pomocí jazyka JSX, což je jazyk, který je překládán do Javascriptu.

React je založen na principu virtuálním DOM (document object model). Nativní implementace je zatížená velkým množstvím funkcí a atributů (několik set). Implementací virtuálního DOM v Javascriptu jen implementována jen minimální potřebná podmonožina. Tím jde docílit nižších nároků na paměť a výpočetní prostředky.

Velká výhoda proti jiným frameworkům také spočívá v možnosti renderovat celý DOM na serveru a na klientovi pouze doplnit o dynamickou funkcionalitu. Díky kombinaci těchto prvků je možné docílit načtení aplikace v řádu 100ms a všechny operace vypadají jako instantní. Tím jde pouhým výběrem frameworku docílit mnohem lepšího uživatelského zážitku.

Pro potřeby projektu byl projekt rozdělen na komponenty následovně:

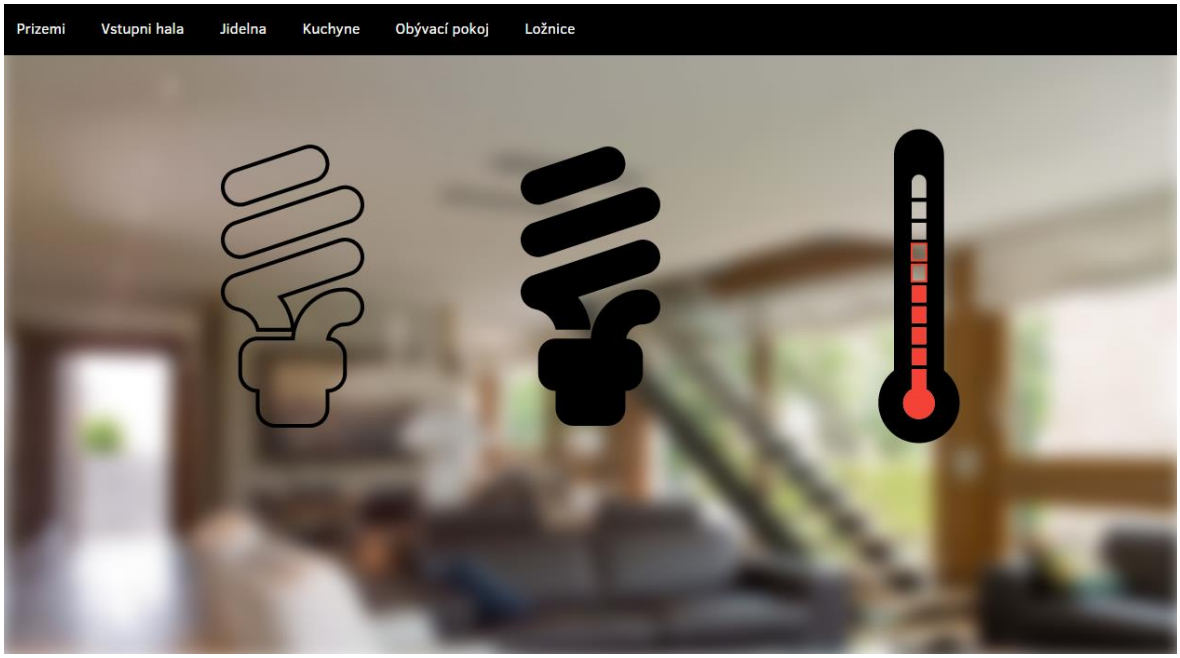
Celá aplikace tvoří jednu komponentu, která je složena z několika podkomponent. Jedná se o menu, plátno s moduly a stavovou řádku.

Aplikační menu obsahuje dvě komponenty. Vertikální menu pro konfiguraci a horizontální menu s místnostmi a skupinami. To je rozděleno ještě na komponentu skupiny a na komponentu položka (místnost).

Plátno tvoří samostatnou komponentu. Ta zodpovídá za načtení a vykreslení modulů vztahujících se k místnosti. Moduly mají společný základ, ale každý modul představuje samostatnou komponentu, která nese zároveň veškerou jeho funkcionalitu – načtení stavu ze systému, zasílání informací o změnách, animace, reakce na události (dotyk, kliknutí).

V nedávné době byla vydána i verze knihovny react-native pro tvorbu nativních mobilních aplikací. Pro ještě lepší výkon a uživatelský zážitek se nabízí úprava aplikace speciálně pro mobilní platformy pod touto knihovnou.

10.10 Výsledná podoba aplikace



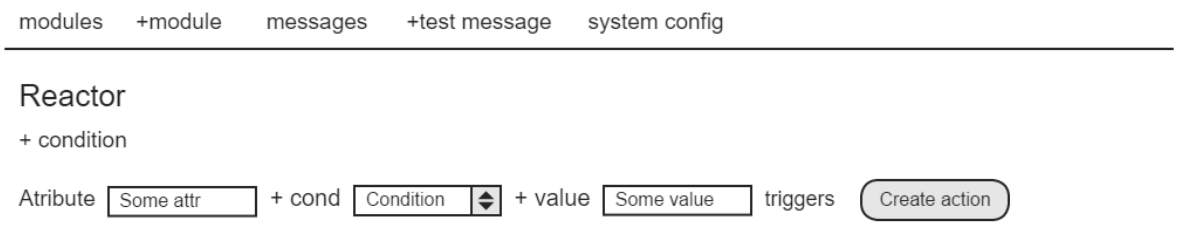
Obrázek 10.2 – Výsledná podoba aplikace pro tablet

Zdrojové kódy jsou dostupné na adrese https://github.com/housedroid.

11 SPRÁVCOVSKÁ KONZOLE

Správcovské konzole je v rámci systému samostatná aplikace, která slouží pro počáteční registraci a konfiguraci, a později pro nastavení podmínek modulů. Grafické zpracování aplikace nemusí být tak dokonalé jako u ovládací konzole, protože bude použito převážně techniky nebo velmi pokročilými uživateli. I tak ale rozhraní musí být přehledné a jednoduché.

11.1 Wireframe



Obrázek 11.1 – wireframe správcovské aplikace

11.2 Funkce aplikace

Základním prvkem aplikace je menu, které umožňuje správu modulů a jejich přidání. V případě kliknutí na moduly se zobrazí dlaždicový výběr existujících modulů. Rychlé přidání modulu spustí formulář s možností naskenování základních informací o modulu z QR kódu nebo ruční zadání všech potřebných polí.

Další položkou jsou zprávy. Ty slouží pro dotazování z logovací služby na historická data nebo umožní zobrazit události z routeru v reálném čase. To usnadňuje ladění systému v případě problému.

Zároveň je možné generovat testovací zprávy, které jsou následně přímo zaslány na router. Je to další z funkcí pro usnadnění ladění a otestování situací, kdy nemusí být některá ovládací součást dostupná nebo se předpokládá její nefunkčnost.

Poslední volnou je obecné nastavení. Cokoliv, co není součástí nastavení modulu a vztahuje se k celému systému, je právě pod touto položkou.

12 STMÍVAČ

Stmívač je jednou ze základních hardwarových komponent systému inteligentního domu. Může být použit v systému regulace intenzity osvětlení, tak i v systémech plynulé regulace topení.

Základem funkce stmívače je úprava průběhu sinusového signálu střídavého napětí, kdy se po určitou část periody drží napětí v nule.

Základním polovodičovým prvkem, který něco takového umožňuje je tyristor. Nyní bude tyristor a z něj odvozené polovodičové součástky popsány blíže.

12.1 Součástky pro konstrukci stmívače

12.1.1 Tyristor

Tyristor je dvou až 4 vývodové zařízení, které se chová jako přepínač. Na rozdíl od tranzistorů nezesiluje signál. 3 vývodové tyristory se používají pro malá napětí, která se aplikují na jeden z jejích vývodů na ovládání mnohem větších napětí a toku proudu přes druhé dva vývody.

Dvou vývodové tyristory naopak nepoužívají řídící vodič, ale místo toho jsou navrženy tak, aby spínaly ve chvíli, kdy napětí dosáhne určité úrovně. To znamená, že do dosažení této úrovně je tyristor vypnutý.

Proč je lepší použít tyristor než tranzistor pro stmívač?

Samozřejmě je možné použít i tranzistor, ale problém tranzistoru je, že nemá pouze stav sepnuto/vypnuto, ale zároveň zesiluje vstupní signál. Bylo by potřeba nastavit přesnou úroveň řídícího napětí a řídícího proudu, aby se choval pouze jako spínač. Pokud by se nacházel mezi stavem zapnutou a vypnuto, zesiloval by vstupní signál. Naproti tomu tyristor je pouze ve stavu sepnuto/vypnuto a není třeba řešit problém se zesílením.

Tyristory se používají běžně napříč zařízeními v celém elektrotechnickém průmyslu. Uplatněný nacházejí v obvodech pro řízení rychlosti, spínání napětí, jako náhrada relé, levné časovače, okruhy pro ovládání fáze, invertory, ovládání rychlosti motoru nebo právě pro řízení intenzity osvětlení.

12.1.2 Hlavní druhy tyristorů

12.1.2.1 SCR

SCR se normálně nachází ve stavu vypnuto. Přivedením malého proudu na hradlo, uvedeme SCR do stavu sepnuto. Proud poté teče od anody ke katodě. SSR zůstává sepnutý i po odebrání řídícího proudu. Aby byl uveden zpět do stavu vypnuto, musí se zároveň odstranit napětí mezi anodou a katodou nebo se musí na anodou přivést záporné napětí vyšší než které je na katodě. Proud teče pouze v jednom směru.

12.1.2.2 SCS

Konstrukci odpovídá SCR, ale liší se dalším vývodem, který představuje anodové hradlo. Spíná ve chvíli, kdy přivedeme pulz právě na tento hradlový okruh.

12.1.2.3 Triak

Jak funguje obdobně jako v SCR, ale na rozdíl od něj propouští proud v obou směrech. To znamená, že může být použit jak pro spínání stejnosměrných, tak zároveň střídavých proudů. Triak zůstala ve stavu sepnuto pouze, pokud je na hradlo přiveden řídící proud. Tzn., že pro vypnutí triaku, stačí odstranit proud přitékající do hradla.

12.1.2.4 4 vrstva dioda

Má pouze dva vývody a používá se mezi dvěma body obvodu. Chová se jako spínač citlivý na napětí. Dokud úroveň napětí přesahuje spínací napětí, proud protéká. Vypíná se ve chvíli, kdy se napětí dostane pod tuto úroveň. Jako klasická dioda propouští pouze v jednom směru.

12.1.2.5 Diak

Funguje stejně jako 4 vrstva dioda, ale na rozdíl od ní propouští v obou směrech. Může se tedy využít jak pro spínání stejnosměrných tak i střídavých obvodů.

12.1.2.6 Výběr vhodného typu

Pro potřeby řídicích systémů domácí automatizace budou využity hlavně prvky SCR a triak. Nevýhoda SCR je propouštění právě jedním směrem. Aby se toto omezení ve střídavých obvodech obešlo, je nutné použít dva SCR zapojené opačně. Toto zapojení přináší i

výhodu v podobě spínání větších proudů než které zvládnete triak. Obě varianty se často používají jako náhrada mechanických relé. Oproti nim přináší výhodu ve vyšší životnosti.

Výhoda triaku spočívá v použití pouze jedné součástky oproti SCR. Výsledný obvod je jednodušší a je poloviční potřeba prvků. Jde tedy dosáhnout výrazně nižší ceny.

12.1.3 Rezistory

Rezistory se rozdělují podle způsobu použití nebo podle materiálu, ze kterého jsou vyrobeny. Když se vybírá rezistor, musí se rozhodnout, jestli je potřeba precizní rezistor, rezistor pro obecné využití nebo výkonný rezistor.

Pracovní rezistory mají malé napětí a nízký výkonnostní koeficient, mají skvělou tepelnou a časovou stabilitu zároveň nízký šum a velmi nízkou reaktanci. Tyto rezistory jsou obvykle dostupné jako rezistory s metalickým filmem nebo drátové konstrukci. Běžně se používají v obvodech, kde je vyžadována velmi blízká rezistence návrhové hodnoty.

Semi precizní jsou menší, než pracovní rezistory používají se převážně při nastavení proudu a při snížení napětí. Mají dlouhodobou teplotní stabilitu.

Rezistory pro obecné využití jsou využívány v obvodech, kde se nevyžaduje tak přesně odpovídající velikost rezistence jako bylo návrhu. Tzn., že tolerance se pohybuje obvykle mezi 5 až 20%. Často zároveň tyto rezistory mají vysoké hodnoty šumu. Nicméně se také rezistory tvořené metalickým filmem používají také jako obecné rezistory vzhledem ke své nízké ceně.

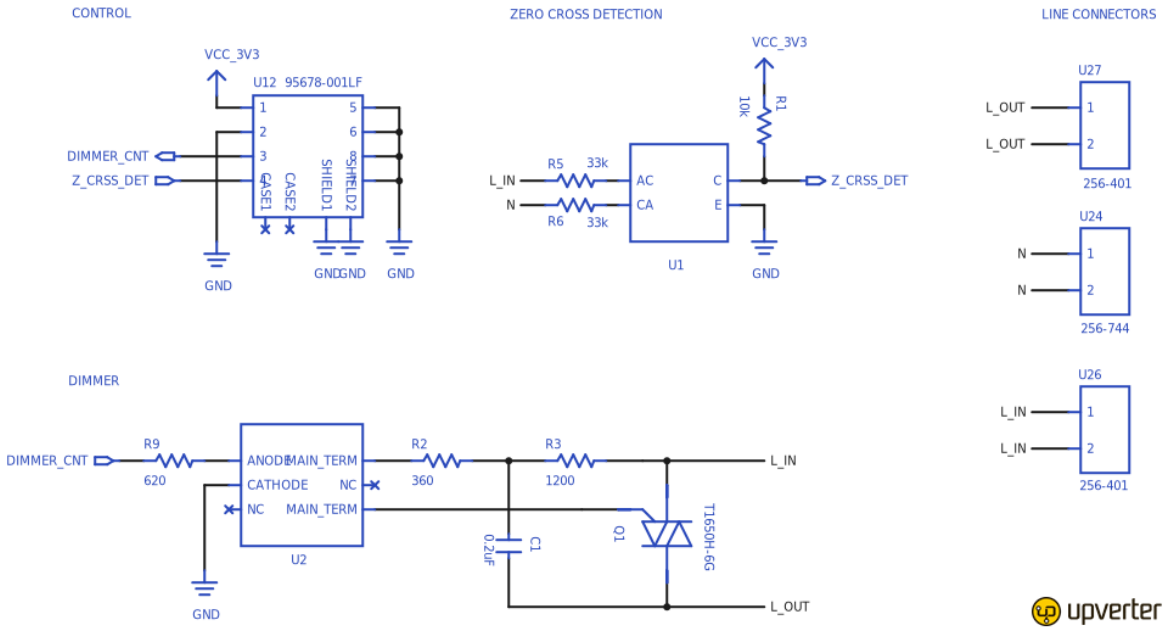
Výkonové rezistory se používají se strojích na řídících okruzích anebo jako děliče napětí. V rámci této aplikace je stabilita okolo 5% akceptovatelná. Výkonné rezistory jsou často dostupné ve formě stočného drátu nebo fólie. Fóliové výkonné tranzistory mají výhodu stability na vysokých frekvencích a mají vyšší hodnoty resistence než rezistory tvořené stočným drátem.

12.1.4 Kondenzátory

Pro aplikace se střídavým napětím, je třeba vybrat kondenzátory určené pro AC aplikace. Tyto kondenzátory se podle bezpečnosti dělí na X a Y kondenzátory. Obvykle jsou keramické nebo fóliového typu.

Popis elektronických součástek byl zpracován podle (16).

12.2 Schéma stmívacího prvku



Obrázek 12.1 – schéma stmívacího prvku

Obvod se dá rozdělit na dvě základní části. První částí je detekce fáze a průchodu nulou. K tomu je využit optotranzistor se dvěma diodami (U1), který lze využít přímo pro střídavá napětí. Běžně používanou alternativní konstrukcí je využití usměrňovacího můstku na vstupu a jedné diody v optotranzistoru. Oproti tomuto řešení je sníženo množství součástek, zjednodušen návrh a s tím i cena.

Pomocí pull-up rezistoru je výstup držen v jedničce na hodnotě 3,3V. V případě průchodu nulou je puštěn proud do báze a následkem toho začne protékat proud mezi kolektorem a emitorem. Tím se výstup dostává do nuly. Výstup Z_CRSS_DET je veden do Pi, kde v případě změny hodnoty na nulu je spuštěn časovač.

V Evropě je v elektrické síti frekvence 50Hz. Tím víme, že sinusoida se zopakuje 50x. Jedno opakování je tedy dlouhé 20ms. Půlvlna trvá 10ms. Při detekci nuly je nastaveno napětí na výstup DIMMER_CNT na nulu. To trvá po dobu 0-10ms v závislosti na zátěži a nastavené intenzitě osvětlení.

Druhá část obvodu je tvořena samotným stmívačem, který je tvořen optoizolátorem (U2) a triakem (Q1). Vstupem optoizolátoru je DIMMER_CNT, který je v 0, pokud je napětí drženo v 0, jinak v 1. Optoizolátor vyžaduje pro svoji činnost proud minimálně 5mA. Na to

je třeba myslet při napájení z Pi. Výstup z Pi je 3,3V. Pro dosažení 5mA byl doplněn rezistor R6, vypočtena jeho hodnota a použita nejbližší menší dostupná z katalogu. Mírné překročení 5mA se hodí v průběhu životnosti komponenty, kdy dochází k degradaci optoizolátoru a je potřeba mírně vyšší proud.

Raspberry Pi má 50mA regulátor pro napájení pinů. Maximální proud na pin je 16mA. Z toho důvodu musel být vybrán úsporný optoizolátor a zároveň nesmí být ovládáno více než 10 stmívacích členů současně. Pro navýšení počtu je třeba pro každý výstup z Pi využít externí tranzistor a externí napájení například pomocí spínaného regulátoru poskytujícího odpovídající proud.

Optoizolátor je dále chráněn rezistory R2, R3 a kondenzátorem C1. Výpočet jejich hodnoty je možné najít v application guide (16).

Použitý triak umožňuje spínání pro proudy až o velikosti 8A. Je v provedení tzv. snubberless. Snubber je obvod (obvykle sériově zapojený rezistor a kondenzátor) pro snížení napěťových a proudových špiček a rezonance. Toto provedení je velmi odolné a přidání snubberu nevyžaduje.

V příloze lze najít i schéma desky plošných spojů. Obvod je dostupný v knihovně Upverteru na (17).

Pro vícekanálový modul může být využito několik stmívačů společně v jednom boxu. Pokud bude v boxu společně umístěno i Pi, je vhodné vyřešit společné napájení pomocí spínaných regulátorů odpovídajícího výkonu, které by umožnily vstup 12/24V a nabízely 5V pro Pi a 3,3V pro zbytek obvodu.

12.3 Výsledná cena pro 4 kanálový stmívač

Cena DPS při výrobě ve Seeedstudio ..... 160 Kč

Cena součástek při malé objednávce ..... 400 Kč

Raspberry Pi..... 800 Kč

Box..... 200 Kč

Výsledná cena..... 1560 Kč

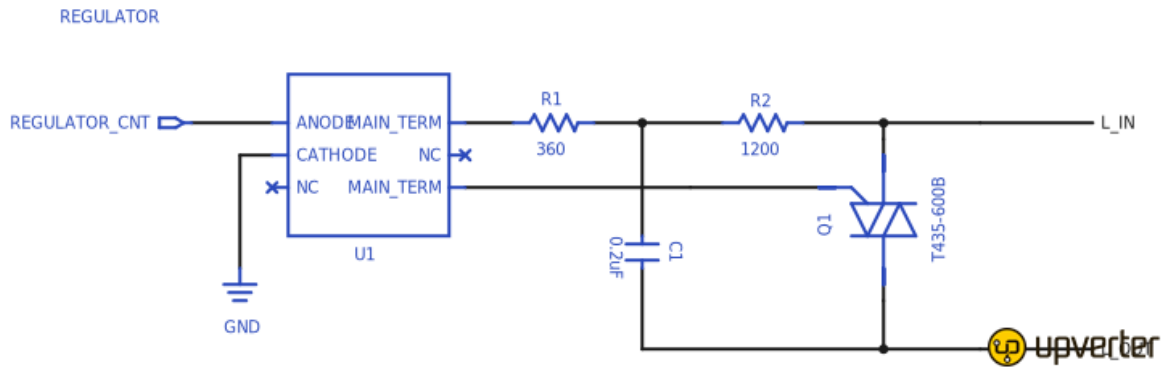
13 MODUL RELÉ

Modul relé má spoustu využití. Jeho základní funkcí je sepnutí nebo vypnutí obvodu. To jde využít pro řízení topení typu „ON/OFF“, zapnutí a vypnutí světel, ovládání žaluzií aj. Modul by měl být schopný spínat rezistivní, kapacitní i induktivní zátěž.

Na rozdíl od stmívače nemusí sledovat průběh sinusového signálu a průchod nulou. Stačí pouze v případě aktivního signálu průchod zapnout, jinak zůstává vypnutý.

Součástky pro využití v relé byly popsány již v minulé kapitole.

13.1 Schéma relé



Obrázek 13.1 – schéma relé

Základem je optoizolátor s triakem, který z důvodů lepší stability se aktivuje v nule (U1). Tento optoizolátor vyžaduje ke svému běhu opět 5mA. Tento optočlen chrání řídící jednotku (Pi) před poškozením vysokým napětím ze sítě.

Optoizolátor je stejně jako v případě stmívače chráněn rezistory R1, R2 a kondenzátorem C1. Společně slouží jako dělička napětí pro ochranu báze triaku, a R2 s C1 tvoří snubber chránící optoizolátor před špičkami. Triak použitý v obvodu je typu snubber-less a je tedy vhodný i pro induktivní zátěž bez snubberu.

Výhoda řešení relé přes triak je v mnohonásobně delší životnosti než při použití elektromagnetických relé, která mají mechanickou životnost pouze cca 10 000 sepnutí. Toto omezení polovodičového řešení odpadá.

V příloze lze najít i schéma desky plošných spojů. Obvod je dostupný v knihovně Upverteru na (18).

U finálního produktu se počítá se zapojením velkého počtu relé do jednoho boxu. Například systém Loxone má modul se 14 relé. Vzhledem k použití by i modul v rámci projektu Housedroid měl mít 12 a více relé. Pro napájení už je v této situaci nutné mít spínaný regulátor a síť tranzistorů pro zesílení proudů jdoucích z Raspberry.

13.2 Výsledná cena pro 14 kanálové relé

Cena DPS při výrobě ve Seeedstudio ..... 440 Kč

Cena součástek při malé objednávce ..... 1100 Kč

Raspberry Pi..... 800 Kč

Box..... 200 Kč

Výsledná cena..... 2540 Kč

14 MODUL REGULACE OSVĚTLENÍ

Jak již bylo zmíněno v teoretické části, modul pro řízení osvětlení slouží k zapnutí, vypnutí světel, nastavení intenzity osvětlení nebo případně k nastavení barvy světla.

S připravenými modulu – stmívačem a relé v aktuální podobně není možné barvu světla změnit a je podporována pouze intenzita, zapnutí a vypnutí. V případě že není potřeba regulace intenzity osvětlení, stačí pouze použít modul relé. Pro nastavení intenzity osvětlení je vždy nutné využít stmívač.

14.1 Model událostí a hlavní atributy

Událost o změně světla je generována na vypínači nebo v uživatelské konzoli (např. na dotykovém panelu). Tato zpráva je směrována do modulu pro řízení světel odvozeného od reactoru – je možné použít i reaktor s nastavenými podmínkami na ovládání relé v případě, že není nutné podporovat intenzitu. Jinak odvozený modul poskytuje více možností.

Následně je zpráva zpracována případně upraveny hodnoty dle podmínek v reactoru a je poslána na cílový stmívač.

Vzorová zpráva z ovladače:

```
{
        "area" : "/house/living-room",
        "created" : "30-12-2014T13:50",
        "message-type" : "light/intensity",
        "module-id" : "adf465fs",
        "module-type" : "controller/light",
        "data" : {
                "action" : "set-intensity",
                "value" : "50.00"
        }
}
```

Typ zprávy (message-type) může být nastaven na light/intensity pro nastavení stmívače, případně na light/switch pro zapnutí či vypnutí světel. Při nastavování stmívače musí být doplněna akce „set-intenzity“ a hodnota v rozmezí 0-100.

Vzorová zpráva z reactoru:


```
{
        "area" : "/house/living-room",
        "created" : "30-12-2014T13:50",
        "message-type" : "dimmer/intensity",
        "module-id" : "adf465fs",
        "module-type" : "reactor/light",
        "data" : {
                "action" : "set-intensity",
                "value" : "50.00"
        }
}
```


14.2 Ovládací modul

K ovládání světel byl vytvořen modul v uživatelské aplikaci. Základem je tvar žárovky, která je ve vypnutém stavu viditelná pouze se zvýrazněným okrajem. V případě zapnutého světla plné intenzity jsou všechny částí s plnou výplní. Pokud uživatel chce nastavit intenzitu, může tahem po světle nastavit intenzitu a vyplní se pouze odpovídající část.

15 MODUL PRO REGULACI TEPLOTY

Regulace teploty jde řešit dvěma systémy. První možností je regulace typu ON/OFF, kde pomocí termoelektrické hlavy se topný okruh plně uzavře nebo otevře. Pro toto použití se hodí využít modul relé.

Druhou možností je využití hlavic s plynulou regulací. Tento typ se používá hlavně u radiátorů. Pro tento systém je naopak vhodné použít stmívač.

Mimo akčních prvků je nutné využít i senzorů. V každé sledované místnosti by měl být umístěn teplotní senzor, který poskytuje aktuální hodnoty o teplotě. Nejlepší možné umístění je přibližně ve výšce hrudi dospělého člověka.

Pro pokročilejší systémy by bylo vhodné přidat i měření teplot na vstupu/výstupu okruhu a teplotu na zdroji tepla. Tyto funkce nebyly zatím navrženy.

15.1 Workflow regulátoru

Pro každou místnost je samostatně nastavena teplota pomocí ovladače nebo aplikace. O dosažení této teploty se dále stará systém. Systém získává aktuální teplotu čtením ze senzoru každých 10 sekund. To vytvoří událost o aktuální teplotě, která je přijata modulem odvozeným od reactoru určeným pro regulaci teploty. Modul byl odvozen hlavně z důvodu budoucího rozšiřování, jinak by stačilo použití podmíněných příkazů.

V případě, že teplota dosáhne nastavený teploty + 0,5°C, je daný okruh topení pomocí vypnut. Pokud teplota teplota klesne pod nastavenou teplotu, je okruh opět otevřen.

15.2 Model událostí a hlavní atributy

Vzorová zpráva z ovladače:

{

"area" : "/house/living-room",

"created" : "30-12-2014T13:50",

"message-type" : "temperature",

"module-id" : "dsfwerdg",

"module-type" : "controller/temperature",

"data" : {

```
        "action" : "set-temperature",
        "value" : "22.00"
    }
}
```


V tomto případě je důležité označení místnosti, pro kterou má být nastavena teplota, nastavení akce na „set-temperature“ a doplnění cílové teploty.

Vzorová zpráva z reactoru:


```
{
    "area" : "/house/living-room",
    "created" : "30-12-2014T13:50",
    "message-type" : "temperature",
    "module-id" : "dsfwerdg",
    "module-type" : " reactor/temperature",
    "data" : {
        "action" : "turn-on"
    }
}
```


Pro řízení je důležité nastavení akce. To je možné v případě zapnutí nastavit na „turn-on“ a v případě vypnutí na „turn-off“.

15.3 Sledování teploty

Pro sledování teploty stačí využít libovolný senzor teploty. Vhodný je senzor, který zároveň umí měřit i vlhkost vzduchu, jako je DHT22.

Takový senzor musí být zvlášť napojen na Raspberry Pi, používá přenos dat na jednom vodiči, ale bohužel se nejedná o protokol 1Wire. Musí tedy mít vlastní ovladač pro obsluhu měření.

Vzorová zpráva měření teploty:

```
{
        "area" : "/house/living-room",
        "created" : "30-12-2014T13:50",
        "message-type" : "temperature/reading",
        "module-id" : "dsfwerdg",
        "module-type" : " sensor/temperature",
        "data" : {
                "value" : "21.00"
        }
}
```


Ve zprávě jde vidět nastavený typ zprávy na čtení teploty, typ senzoru a následně přečtenou hodnotu ve °C.

15.4 Ovládací modul

Ovládací modul pro aplikaci je tvořený symbolem teploměru. Hladina teploměru je držena celou dobu na středu. Každý stupeň na teploměru představuje půl stupně nastavení. Při tahu směrem nahoru je teplota přidána. Výplň stupňů je zbarvena do červena, což symbolizuje vytápění. Další pole jsou zvýrazněna červeným obrysem a znázorňují cílovou teplotu a kolik do ní ještě chybí.

Při tahu směrem dolu se jedná o odpovídající akci pro chlazení. Výplň stupnice je zbarvena do modra a cílové stupně od středu jsem transparentní s modrým obrysem.

Modul má nastavenou hranici mezi 18-26°C.

16 MODUL PRO LOGOVÁNÍ

Modul pro logování slouží k dlouhodobému ukládání informací pro monitorování nebo pro ladění v případě potřeby. Princip jeho činnosti je velmi jednoduchý. Modul se přihlásí routeru pro odběr zpráv z určitých oblastí, nebo pro odběr všech zpráv z celého systému. Tím jsou mu přeposílány žádané zprávy.

Po přijetí je zpráva uložena do databáze MongoDB, která slouží jako perzistentní úložiště. Konfigurace databáze počítá s nastavením většího počtu zápisů než čtení. Data jsou uloženy v BSONu (binární podoba JSONu).

Nad službou je možné provádět dotazy pomocí REST rozhraní.

16.1 API pro dotazování

/v1/module/{module-id}/	Vrací všechny informace týkající se modulu
/v1/area/{area}	Vrací všechny zprávy z oblasti
/v1/area/{area}/message/{msg-type}	Vrací všechny zprávy z oblasti pro daný typ zprávy
/v1/area/{area}/module/{module-type}	Vrací všechny zprávy z oblasti pro daný typ modulu

Toto je základní API pro dotazování nad modulem. Pro pokročilejší dotazy je možné zadat jako parametry pro API hledaný atribut zprávy a jeho hodnotu. Takto je možné pro vyhledávání zkombinovat i více atributů.

17 BEZPEČNOSTNÍ MODUL

Bezpečnostní modul se dá rozdělit na část detekční a monitorovací, a část aktivní.

V rámci detekce je základem několik senzorů. Prvním senzorem, který se dá při nasazení využít je okenní senzor. Jedná se o vodič a magnet. Senzor je ve dvou provedení – normálně otevřený a normálně zavřený. V rámci aplikace stačí připojit senzor mezi dva GPIO piny (případně může být využit i 3,3V pin) spolu se sériově zapojeným rezistorem pro omezení průchodu proudu a ochranu pinů. Při změně hodnoty na pinu je detekováno otevření nebo zavření okna. Stejný senzor a princip se použije i na detekci otevřených dveří.

Druhým senzorem, který pomáhá zvýšit bezpečnost je senzor vibrací. Při aplikaci na sklo umožňuje zjistit, zda došlo k rozbití skla - stačí nastavit správnou citlivost senzoru a při detekci vibrací detekovat délku vibrace.

Třetím typem senzoru je detektor kouře (stejné informace platí i pro senzor plynu). Běžně nabízené senzory poskytují jako výstup 0-5V podle koncentrace CO v ovzduší. Pro připojení na Raspberry je nutné přidat dělič napětí, nebo použít externí AD převodník připojený přes SPI akceptující napětí 5V.

Mimo senzorů lze bezpečnostní systém doplnit kamerovým systémem. Pro testování bylo použito Raspberry Pi s NoIR kamerou uzavřené v boxu od falešné kamery. Tímto způsobem lze vytvořit levnou FullHD bezpečnostní kameru. Pi dokáže poskytovat živý přenos pomocí MPEG-DASH přímo do prohlížeče, a zajistit uložení úseků videa na síťové úložiště. Uložené video je možné přehrát v prohlížeči, když je do stránky (aplikace) vložené pomocí video tagu.

V rámci aktivní části jde o použití elektromotorický nebo elektromagnetických zámků. Pro zámky je, z navržených modulů, jediné využitelné relé. Toto ale není dostačující pro většinu moderních zámků a je doporučené využít externí ústředny.

ZÁVĚR

V rámci této práce byla provedena analýza aktuální situace na trhu automatizačních systému pro inteligentní domy. Pro vybrané systémy byly zkoumány jejích dostupné moduly, zjištěny jejich silné a slabé stránky a stanoven cenový odhad pro nasazení daného systému. Na základě této analýzy byly vybrány moduly, která je vhodné implementovat pro konkurence schopný systém, vybrány silní stránky, které je vhodné zachovat a naopak, které slabé časti současných systému se zaměřit.

V rámce práce byly vyhodnoceny dostupné technologie, který by umožnili rychlou a jednoduchou implementaci systému jak z pohledu softwaru, tak hardwaru. Tím se potvrdila hypotéza o vhodných dostupných technologiích.

Na základě toho byl navržen systém pro implementaci v rámci projektu Housedroid a provedena zkušební implementace některých částí. Potvrdila se možnost pomocí vybraných technologií realizovat všechny důležité části systému. Se zvolenými technologiemi a návrhem je možné dosáhnout i požadovaných parametrů jako je doba odezvy aplikace. Tím byl splněn další ze stanovených cílů práce.

Z hlediska uživatelské prostředí bylo provedeno několik iterací návrhu a implementace. Uživatelské rozhraní prozatím nesplňuje požadavek na dokonalý uživatelský zážitek, a potřebuje pro finální produkt ještě několik iterací ideálně ve spolupráci s profesionálním designérem. Navržená koncept se ale ukazuje jako vhodný začátek cesty.

Posledním důležitou hypotézou byla možnost dosažení poloviční ceny řešení. V rámci ceny řešení se podařilo dosáhnout ceny pro referenční projekt čtvrtinové oproti nejlevnějšímu analyzovanému systému a šestinové ceny oproti ostatním řešením. To dává při případném vstupu na trh možnost cenových úprav. Zároveň při větší výrobě než je kusová by se podařilo náklady na výrobu stlačit ještě výrazně níže.

Výsledek této práce jde považovat pouze za základ systému. Vývoj bude dále pokračovat, aby se doplnila chybějící funkcionalita a optimalizovaly se moduly a aplikace. Na konci léta 2015 budou provedena dvě referenční nasazení systému v rodinných domech.

SEZNAM POUŽITÉ LITERATURY

1. **Delgado, Rick.** Top 10 Benefits of Automating Your Home. *Freshome.* [Online] http://freshome.com/2013/01/17/top-10-benefits-of-automating-your-home/.

2. **Shimpi, Anand Lal.** ARM's Cotex A7: Bringing Cheaper Dual-Core & More Power Efficient Eigh-End devices. *AnandTech.* [Online] http://www.anandtech.com/show/4991/arms-cortex-a7-bringing-cheaper-dualcore-more-power-efficient-highend-devices.

3. **Specout.** ARM Cortex A7 MPCore vs ARM Cortex A8. *Specout.* [Online] http://system-on-a-chip.specout.com/compare/37-50/ARM-Cortex-A7-MPCore-vs-ARM-Cortex-A8.

4. **ARM Holdings.** Cortex A. *ARM.* [Online] http://www.arm.com/products/processors/cortex-a/.

5. **Debian Comunity.** The Computer Language Benchmarks Game. *Debian.* [Online] http://benchmarksgame.alioth.debian.org/.

6. **Troelsen, Andrew.** *Pro C# 5.0 and the .NET 4.5 Framework.* Berkeley, Calif. : Apress, 2012. 978-1430242338.

7. **Salvatore, Peter.** The new minimalist operating systems. *Docker Blog.* [Online] Docker. https://blog.docker.com/2015/02/the-new-minimalist-operating-systems/.

8. **Card, Stuart K., Robertson, George G. a Mackilay, Jock D.** *The information visualizer: An information workspace.* New Orleans : ACM CHI'91 Conf., 1991. stránky 181-188. 0-89791-383-3.

9. **Sustrik, Martin.** [Online] http://nanomsg.org/documentation-zeromq.html.

10. **Treat, Tyler.** Fast, Scalable Networking in Go with Mangos. *Brave New Geek.* [Online] http://bravenewgeek.com/fast-scalable-networking-in-go-with-mangos/.

11. **D'Amore, Garrett.** Repoztář Mangos. [Online] https://github.com/gdamore/mangos.

12. **ECMA.** ECMA-404 The JSON Data Interchange Standard. *JSON.* [Online] http://www.json.org/.

13. **Wróblewski, Piotr.** *Algoritmy, datové struktury a programovací techniky.* Brno : Cpress, 2004. 80-251-0343-9.

14. **Wikipedia.** WebSocket. *Wikipedia.* [Online] Wikipedia. http://en.wikipedia.org/wiki/WebSocket.

15. **Harisov, Vitaly, Berezhnoy, Sergey a Belov, Sergey.** Methodology. *BEM.* [Online] https://en.bem.info/.

16. **Scherz, Paul a Monk, Simon.** *Practical Electronics for Inventors.* New York : McGraw-Hill, 2013. 978-0071771337.

17. **Fairchild Semiconductors.** MOC30xx Application Guide. *Fairchild Semiconductors.* [Online] https://www.fairchildsemi.com/datasheets/MO/MOC3023M.pdf.

18. **Grolig, Lukáš.** Light Dimmer. *Upverter.* [Online] https://upverter.com/vexfalard/7dba16bf7d9939d4/Light-Dimmer/.

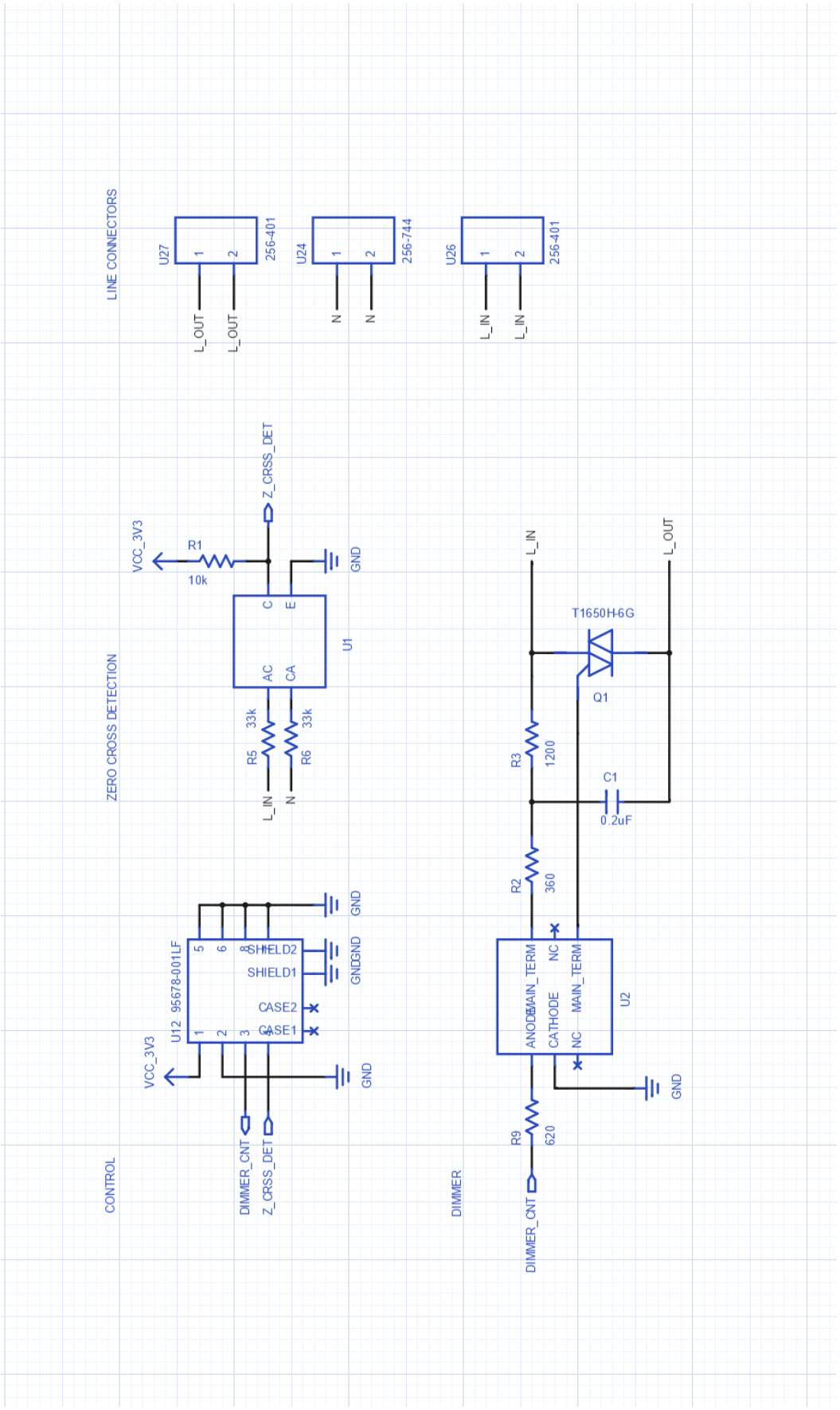
19. —. Regulator Module. *Upverter.* [Online] https://upverter.com/vexfalard/17aeed03144f3d19/Regulator-Module/.

SEZNAM OBRÁZKŮ

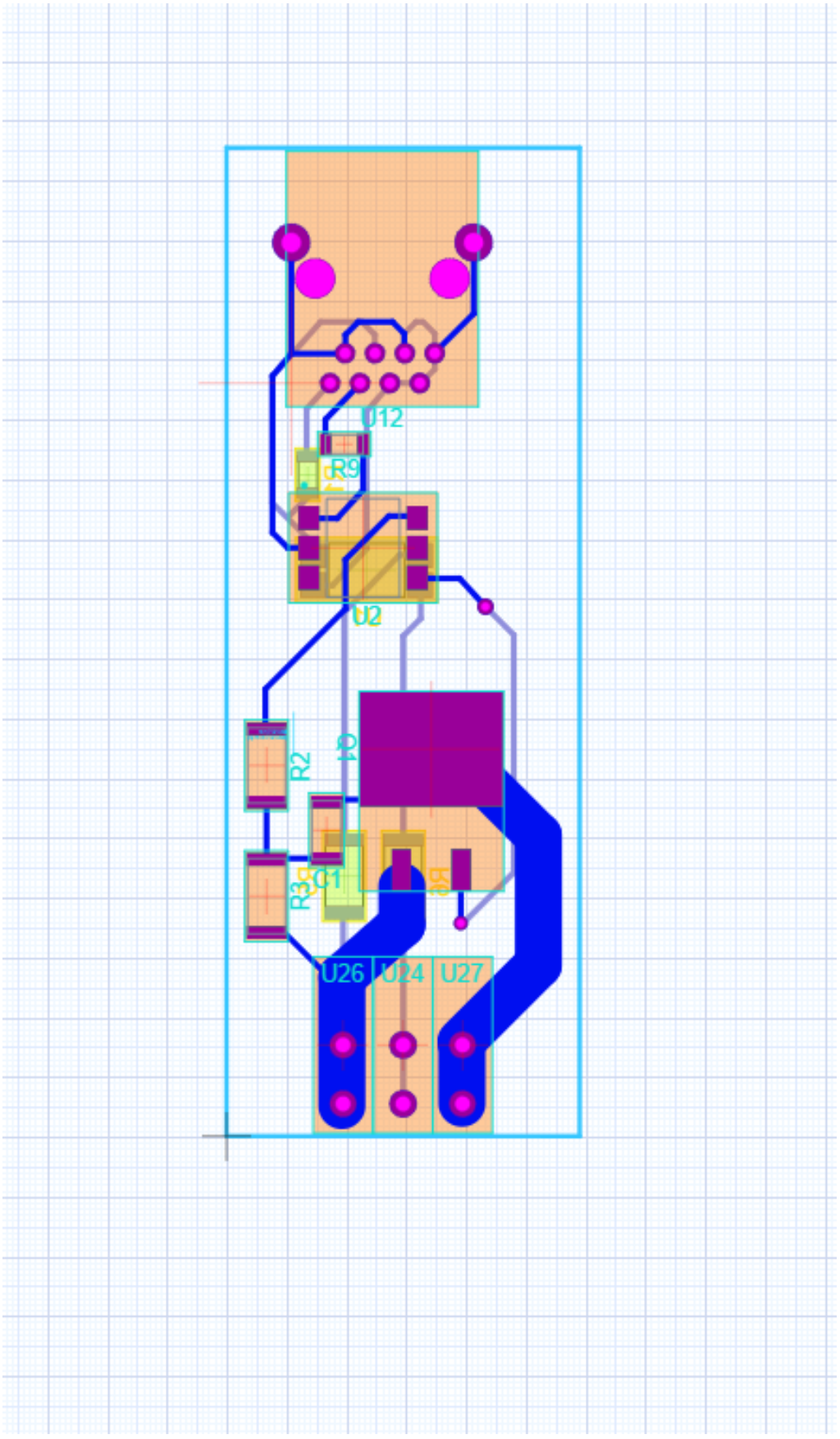
SEZNAM TABULEK

SEZNAM PŘÍLOH

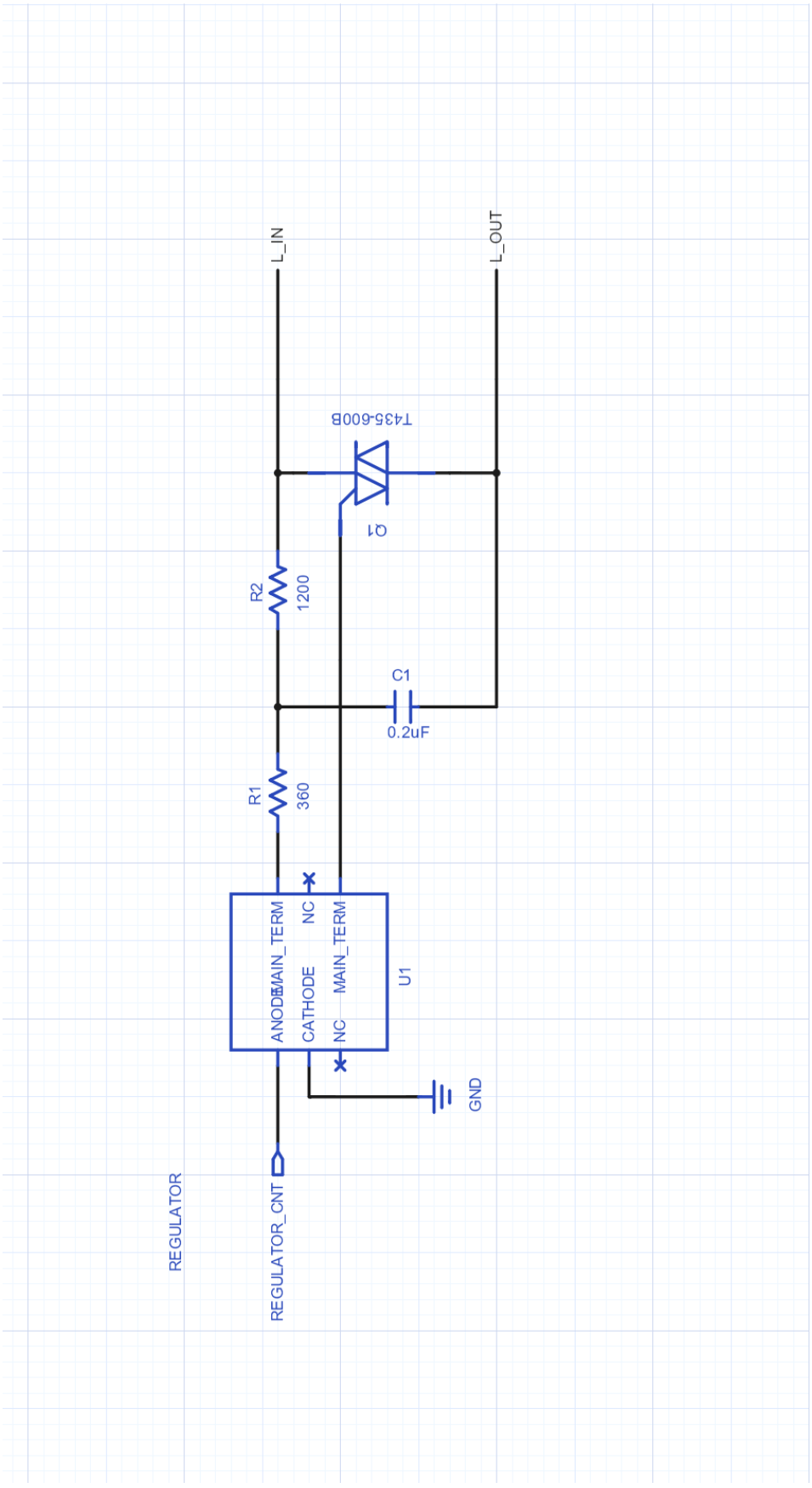
PŘÍLOHA P I: STMÍVAČ SCHÉMA



PŘÍLOHA P II: STMÍVAČ DPS



PŘÍLOHA P III: RELÉ SCHÉMA



PŘÍLOHA P IV: RELÉ PCB