Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Inteligentní řízení domu se vzdáleným přístupem

Intelligent House with Remote Control

2014 Luboš Matejčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Luboš Matejčík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Inteligentní řízení domu se vzdáleným přístupem
Intelligent House with Remote Control

Zásady pro vypracování:

Navrhněte nízkorozpočtové inteligentní řízení domu pro jeho hlavní silové rozvody, řízení topení a rolet. Systém navrhněte s dostatečnou modularitou, aby jej bylo možno dále rozšiřovat. Pro konfiguraci systému navrhněte vhodné WWW rozhraní.

1. Seznamte se se systémy inteligentního domu, jejich technickými možnostmi a možnostmi řízení.
2. Navrhněte vlastní nízkonákladovu řídící jednotku domu pro řízení a monitorování jeho vybraných částí.
3. Vyberte vhodné součástky a navrhněte potřebné elektronické obvody a DPS.
4. Napište potřebné programové vybavení pro mikropočítače a hlavní řídící počítač s WWW rozhraním.
5. Otestujte spolehlivost a stabilitu navrženého systému a pokuste se vyhodnotit jeho efektivitu.

Seznam doporučené odborné literatury:

1. http://ad.iqdum.cz/inteligentnidum
2. Datové listy vybraných elektronických komponenet

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Rád bych na tomto místě poděkoval panu Ing. Petru Olivkovi za jeho cenné rady a velice užitečnou pomoc. Dále bych chtěl poděkovat všem lidem, kteří mi pomohli s vytvářením této práce.

Abstrakt

Tato práce pojednává o tom, jak vytvořit levný inteligentní dům, který se vyrovná kvalitním systémům inteligentních domů, které jsou na trhu.

Klíčová slova: inteligentní dům, Raspberry Pi, ovládání, monitorování, I2C, 1-Wire, PCF8574, DS18B20, vzdálené ovládání

Abstract

This paper discusses how to create a smart home, which parallels the quality of smart home systems that are on the market.

Keywords: smart home, Raspberry Pi, control, monitoring, I2C, 1-Wire, PCF8574, DS18B20, remote control

Seznam použitých zkratek a symbolů

I2C	–	Inter-Integrated Circuit – multi-masterová počítačová sériová sběrnice vyvinutá společností Philips
SDA	–	Serial Data Line – Vodič přenášející data u sběrnice I2C
SCL	–	Serial Clock Line – Vodič přenášející hodinový signál u sběrnice I2C
SSH	–	Secure Shell
EAGLE	–	Easily Applicable Graphical Layout Editor
CRC	–	Cyclic redundancy check – Cyklický redundantní součet
ASP	–	Active Server Pages
JSP	–	JavaServer Pages
HTTP	–	Hypertext Transfer Protocol
URL	–	Uniform Resource Locator
AJAX	–	Asynchronous JavaScript and XML
LED	–	Light-Emitting Diode – dioda emitující světlo
LCD	–	Liquid Crystal Display – displej z tekutých krystalů
HTML	–	Hyper Text Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

Úvod

V technickém pokroku dnešní doby, kdy je velké množství zařízení ovládáno počítači, se tento trend pomalu dostává i do běžných domácností. Zařízení z nedávných sci-fi filmů, která byla stvořena z fantastických myšlenek scénáristy, jsou dnes záležitosti, které jsou realizovatelné. Myšlenka inteligentního ovládání zařízení v domě, kde s Vámi mluvil Váš dům, zavíral okna, či Vám dokonce naladil Vaši oblíbenou stanici, při dnešních technologiích není nemožné vytvořit. Navíc realizace takového projektu by při vhodném výběru technologií nebyla ani tak cenově nedostupná.

Cílem mé bakalářské práce je vytvořit nízkorozpočtový systém řízení zařízení v domě, jinými slovy inteligentní dům. Systém bude umožňovat pohodlné ovládání silových částí domu, monitorování vstupních zařízení, například teplotních čidel, vypínačů, magnetických kontaktů a v podstatě veškerých dvoustavových zařízení. Výsledek práce bude obsahovat hlavní řídící jednotku, v podobě miniaturního počítače Raspberry Pi, který bude sloužit jako server pro veškeré ovládání systému, dále bude systém obsahovat veškerý hardware obstarávající spínání silových části, monitorování vstupních zařízení a další prvky pro úpravu sběrnic.

Systém inteligentního domu chci vytvořit, protože bych si přál, aby byl součástí mého bydlení. Nemohu si jej dovolit koupit, ale můžu jej vytvořit za menší částku, než za kterou se nabízí. Navíc si myslím, že k vytvoření takového systému bude potřeba propojit mnoho odvětví techniky a vědy. Tento systém by také mohl usnadnit bydlení tělesně postiženým lidem s omezenými možnostmi pohybu a vzhledem k nízké ceně systému by si jej mohli dovolit.

V první části Vás seznámím se systémy inteligentních domů, respektive s možnostmi řízení, monitorování a technologiemi s nimi spojenými. V této kapitole budu také vycházet z doporučené literatury Inteligentní dům od Miroslava Valeše [1].

Další část bude pojednávat o mém návrhu řídící jednotky pro řízení a monitorování vybraných částí. Popíši topologii a vysvětlím proč jsem zvolil dané technologie.

Dále popíši návrhy zařízení, do kterých se připojí silové části domu a ty, do kterých se připojí monitorovaná zařízení. Ukážu schémata zapojení, návrhy desek plošných spojů a schémata celého propojení systému.

V neposlední řadě Vám popíši programové vybavení řídící jednotky a předvedu webové rozhraní, tvář celého systému, projdu funkce webového rozhraní a představím použité technologie.

Jako poslední část bude testování systému, kde bych rád ukázal, ve srovnání k jiným systémům, jak si můj systém povede a pokusím se vyhodnotit efektivitu.

1 Systémy inteligentního domu

V této kapitole se budu zabývat systémy inteligentního domu, jaké existují možnosti řízení a budu probírat komunikaci uživatele s domem.

1.1 Co je to inteligentní dům

„Inteligentní dům v nejširším možném smyslu slova je budova vybavená počítačovou a komunikační technikou, která předvídá a reaguje na potřeby obyvatel s cílem zvýšit jejich komfort, pohodlí, snížit spotřebu energií, poskytnout jim bezpečí a zábavu pomocí řízení všech technologií v domě a jejich interakcí s vnějším světem." [1]

Inteligentní dům by měl zpříjemnit bydlení uživatelům a zajistit komfort. Jako základní prvky inteligentního domu bychom mohli označit termostaty pro řízení topení, ovládání světel, ale také zabezpečovací systém a kamerový systém. Ozvučení místností a domácí kino by rovněž v takovém domě nemělo chybět. Podobný systém dokážeme vytvořit koupením těchto prvků v obchodě s elektronickými zařízeními a spotřebiči, ale tyto prvky nebudou spolu komunikovat a nebude je možno ovládat z jednoho centrálního rozhraní.

Důležitá věc, která dělá z domu inteligentní dům je tedy propojení těchto výše zmíněných prvků do jednoho systému, kde bude existovat možnost ovládání připojených prvků. Tímto propojením bychom zajistili i automatizaci.

Díky tomuto propojení a vytvoření jednotného systému není potřeba mít velké množství ovladačů. Také není potřeba mít v každé místnosti klávesnici pro ovládání klimatizace, nebo zabezpečovacího systému, což zasahuje i do odvětví designu interiéru v kladném mínění. Nevypadalo by pěkně, kdyby na stěně bylo připevněno několik druhů ovladačů a každý by byl jinak designově zpracován.

Automatizace může zajistit komfort v podobě zavření střešních oken při dešti, nebo při zapomenutém zhasnutí světel může tuto práci udělat za nás. V podstatě toto propojení všech zařízení do jednoho systému dělá z myšlenek velice jednoduše vytvořitelné záležitosti. Připojení nového, nebo třeba již nainstalovaného zabezpečovacího systému do inteligentního domu vede k dalším výhodám. Například rozsvícení všech světel při poplachu a přivolání záchranné služby, nebo zamčení dveří v místnosti kde se narušitel nachází. [1]

1.2 Prvky inteligentního domu

Tato kapitola ukáže jaké prvky by měly být těmi základními pro intelignentí dům. Může se jednat o:

- prvky zajišťující ovládání osvětlení,
- zařízení obstarávající regulaci topení,
- prvky na řízení tepelných ztrát a zisků.

1.2.1 Osvětlení

V inteligentním domě bychom měli být schopni nejen vypnout a zapnout osvětlení, ale také regulovat jeho intenzitu. Ta by mohla být i závislá na denním světle kvůli úsporám za energii. Regulace svitu osvětlení také předejde oslňování v noci. S některými světly, například ve sklepě, nebo venku na zahradě, není třeba svítit dlouhodobě. Z toho důvodu bychom mohli využít detektory pohybu. Díky detektorům docílíme dalších energetických úspor.

V rámci úspor je dobré zvážit, do kterých místností by bylo vhodné nainstalovat úsporné žárovky, a do kterých místností LED osvětlení viz obrázek 1.1. Vzhledem ke spotřebě energie žárovek by bylo vhodné jimi osvětlovat místa, která svítí pár hodin denně a LED osvětlení použít na více využívaných místech. Sytému inteligentního domu by nemělo chybět nastavení scén osvětlení a jednoduché přepínání mezi nimi. Zajímavé rozšíření by mohlo být i nastavení scén v závislosti na slunečním svitu, nebo času.

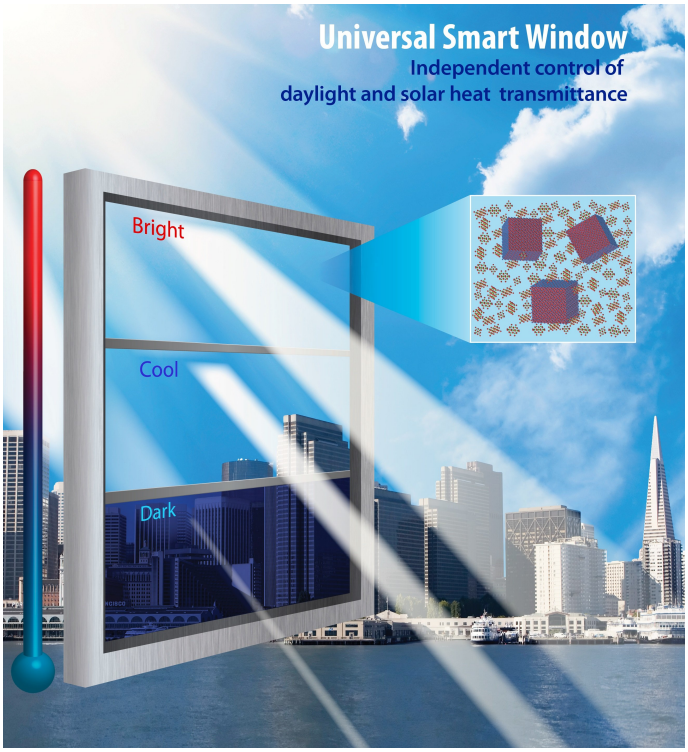
Obrázek 1.1: LED žárovka 5050 4W příkon

1.2.2 Stínění

Mezi způsoby zastínění můžeme uvést například rolety, žaluzie, ale také celkem nestandartní stínící skla nazývané Universal Smart Window viz obrázek 1.2, které se chovají obdobně jako LCD displeje, při přivedení elektrického proudu sklem neprochází světlo. Intenzita stínění je regulovatelná velikostí protékajícího proudu.

Z hlediska úspory energie v létě je výhodnější zastínit okna na sluneční straně a snížit využívání energeticky náročné klimatizace. O toto by se opět měla postarat automatizace inteligentního domu. Je-li teplota vyšší než stanovená hranice, stínící systém zastíní okna na slunečné straně. Pokud bude slunce delší dobu schované za mraky, systém zastínění

odstraní. Zatažené žaluzie nebo rolety také sníží teplotní únik v noci. Rovněž lze využít identifikaci uživatele, zajistit tak pro něj nastavení jeho oblíbeného režimu stínění oken a díky tomu mu dopřát světelnou intenzitu, na kterou je zvyklý. [1]



Obrázek 1.2: Ukázka stínícího okna Universal Smart Window

1.2.3 Topení

Dnešní nejběžnější způsob vytápění je za použití jednoho teplotního čidla umístěném v referenční místnosti a podle této teploty se řídí termostat, který ovládá topení v celém domě. V inteligentním domě by neměla chybět možnost regulace termostatů pro každou místnost zvlášť. Z hlediska úspory energie je toto řešení výhodnější.

Mezi nejčastěji využívané topící soustavy patří elektrické topení a vytápění oběhem vody. U elektrického topení je řešení z hlediska ovládání nejjednodušší. Taková soustava vytápění má zpravidla dva stavy a to zapnuto nebo vypnuto. U soustavy vytápění oběhem vody je potřeba ohřát vodu na požadovanou teplotu plynovým, či elektrickým kotlem a navíc ovládat regulační prvky soustavy u tepelných výměníků, respektive radiátorů. Pro regulaci soustavy se využívají servomotory připevněné k regulačním prvkům soustavy – elektronické hlavice se servopohonem viz obrázek 1.3.

Inteligentní dům měřené teploty ukládá s možností zobrazení grafů teplot v závislosti na čase.

Obrázek 1.3: Ukázka standartní elektronické hlavice se servopohonem

1.2.4 Další prvky

Mezi další prvky bychom mohli zařadit

- klimatizace,
- ventilace,
- spotřebiče,
- ozvučení,
- brány.

1.3 Komunikace s domem

Inteligentní dům by měl poskytovat jednoduché a intuitivní ovládání veškeré techniky v domě pomocí jednoduchého ovladače. Většinou je toto ovládání zajištěno dotykovým panelem, nebo grafickou klávesnicí (obrázky 1.4 a 1.5). Dotykové panely by měly nejen umožnit ovládání prvků v domě, ale také zajistit konfiguraci. Například může se změnit význam vypínače v obývacím pokoji. Tento vypínač by nadále nemusel ovládat osvětlení v obývacím pokoji, ale třeba by mohl zapínat televizor. [1]

Další z možností by měla být konfigurace scén a přiřazení těchto scén k hodinám dne, ročnímu období, nebo by scény mohl měnit jiný spouštěč, například světelný senzor. Takto nakonfigurovaný dům plní jeden ze svých úkolů, a to komfort.

Jako přepínač scén může také sloužit předvolba „místnost bez lidí". Tato předvolba může snížit spotřebu energie a plnit druhý význam domu, kterým je snižování energetické spotřeby.

Obrázek 1.4: Hlavní obrazovka dotykového panelu

Obrázek 1.5: Grafická klávesnice bezpečnostního systému

Ovládaní inteligentního domu by ovšem mělo být přístupné i na dálku. Pomocí internetového připojení lze komunikovat s domem a ovládat tak zařízení odkukoliv. Uživatel takového domu potom může na dálku zapnout topení a po příjezdu domů bude mít vytopeno na požadovanou teplotu.

2 Návrh inteligentního domu

Tato sekce bude pojednávat o mém návrhu levného inteligentního domu.

2.1 Hlavní řídící jednotka

V prvotním konceptu inteligentního domu jsem chtěl vytvořit řídící jednotku použitím mikrokontroléru s dostatečnou rychlostí a množstvím vstupů/výstupů. Ačkoliv po zvážení možností mikrokontrolérů, ceny a času stráveném na výrobě takové jednotky jsem musel zvolit jiný prostředek, nejlépe sestavu stolního počítače s operačním systémem. Hledal jsem sestavy mini počítačů, které by vyhovovaly pro řídící jednotku, ale jejich cena nebyla příliš přijatelná. Když jsem se dozvěděl o miniaturním jednodeskovém počítači Raspberry Pi a jeho výhodách, tak jsem jej zvolil jako hlavní řídící jednotku.

2.1.1 Raspberry Pi

Raspberry Pi pro mě byla jednoznačná volba, nejen z důvodu velké rychlosti procesoru ve srovnání s mikrokontroléry, ale také tím, že má přímo na desce vyvedené vstupně výstupní piny. Takže v podstatě nahrazuje mikrokontrolér na desce s dvěma USB, připojením ethernetu, HDMI výstupem a zvukovým výstupem, viz Tabulka 2.1. Pro Raspberry Pi bylo vytvořeno několik distribucí operačních systémů. Zvolil jsem Raspbian, což je klon operačního systému Debian, s tím rozdílem, že je napsán pro ARM architekturu a přizpůsoben Raspberry Pi. [6]

Díky Raspberry Pi je programování řídící jednotky na úrovni Linuxového operačního systému. Navíc není potřeba vytvářet složité desky plošných spojů pro vytvoření řídící jednotky s mikrokontrolérem.

2.1.2 Funkce řídící jednotky

Na řídící jednotce bude spuštěn serverový program, který bude monitorovat vstupní zařízení a ovládat výstupní zařízení. Veškerá automatizace, logika všech zařízení, všechna data a konfigurace budou umístěny v této jednotce. Díky připojení jednotky do počítačové sítě je možno přistupovat k celému systému ze sítě. K jednotce se bude možno připojit pomocí protokolu SSH a celý systém tak spravovat. Řídící jednotka bude také plnit úlohu webového serveru a tímto způsobem bude poskytovat ovládací rozhraní pro uživatele.

2.2 Zařízení

Z pohledu mého návrhu inteligentního domu chápejme tyto zařízení, dále fyzická zařízení, jako prvky, které jsou připojeny k řídící jednotce, nikoliv osvětlení, topení, klimatizace apod. Prvky domu jsou připojeny až k fyzickým zařízením. Tato zařízení budou propojena sběrnicí podle obrázku A.1.

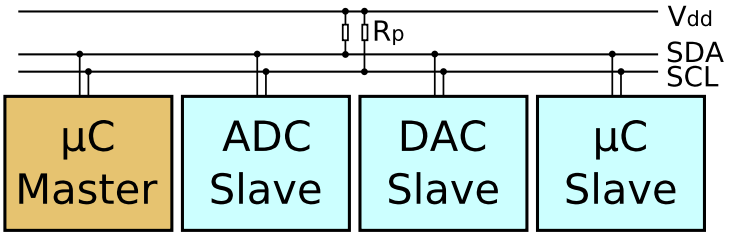
Procesor	ARMv6 700 MHz
Grafický procesor	VideoCore IV
Paměť RAM	512 MB
Paměť	SD karta
HDMI	Ano
Zvukový výstup	Ano
USB	2
Ethernet	Ano
I2C	Implementováno v jádře OS
1-Wire	Modul jádra OS
Napájení	5V 1.2 A
Příkon	3W

Tabulka 2.1: Parametry Raspberry Pi

2.2.1 Komunikační sběrnice

Jak již bylo zmíněno Raspberry Pi poskytuje mnoho komunikačních možností. Například I2C sběrnicí. „Sběrnice I2C byla navržena firmou Philips Semiconductors pro komunikaci jednočipových mikropočítačů s dalšími číslicovými obvody." [2]

Tato sběrnice komunikuje po dvou linkách a je obousměrná. Linka SDA slouží pro přenos dat a adresy. Linka SCL pro přenos hodinových impulzů. Uspořádání systému je takové, že musí existovat alespoň jeden účastník master, který ovládá průběh komunikace a dále alespoň jeden účastník slave, který vždy pouze odpovídá, je-li vyzván ke komunikaci viz obrázek 2.1. Každý slave na sběrnici má svou adresu. Zpravidla část této adresy je zarezervovaná čipu a zbylou část můžeme nastavit. [2] [4]



Obrázek 2.1: Struktura I2C sběrnice

Jako další komunikační možnost Raspberry Pi, ačkoliv není přímo implementována do jádra operačního systému Raspbian, je sběrnice 1-Wire. Tato sběrnice komunikuje po jedné lince a to obousměrně. Rozdíl oproti sběrnici I2C je ten, že neobsahuje linku s přenosem hodinových impulzů. Sběrnice umožňuje provoz bez napájení. Využívá se nabití kondenzátoru umístěném v čipu a následného vybití do integrovaného obvodu a odeslání dat zpět po sběrnici. Tento operativní mód není příliš vhodné využívat, pokud to okolnosti dovolí, vzhledem k chybám, ke kterým může na sběrnici dojít. Opět má každé zařízení svou adresu a ke komunikaci vyzývá master. S touto sběrnicí určitě přichází do styku mnoho lidí. Například iButton (obrázek 2.2) čipy pro otevření dveří na relativně nových zvonkových tablech. [7]

Obrázek 2.2: Ukázka čipu iButton od firmy Dallas

2.2.2 Výstupní expandér

Výstupní expandér je fyzické zařízení, které bude pomocí elektromagnetických relé ovládat elektrické silové části domu. Expandér bude prvek, který je složen z jednoduchých elektronických obvodů, a to z důvodů zajištění spolehlivosti a co nejmenší ceny. Veškerá logika tohoto zařízení bude naprogramována v řídící jednotce. Zařízení bude komunikovat po sběrnici I2C jako zařízení I2C slave. Důvodem je jednoduchost sběrnice a kompatibilita s velkým množstvím integrovaných obvodů. Ovládání bude zajišťeno převodem logické informace na sepnutí/rozepnutí elektromagnetického relé. Tato relé budou zapínat nebo vypínat zařízení k nim připojená. Výstupní expandéry se budou nacházet v elektrickém rozvaděči.

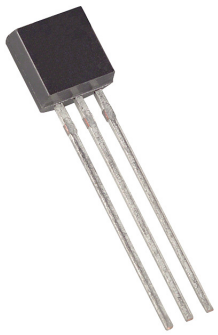
2.2.3 Vstupní expandér

Jedná se o fyzické zařízení, které bude monitorovat spínací vstupní prvky domu. Mezi vstupní prvky můžeme zařadit vypínače, tlačítka, magnetické kontakty, pohybové čidla a v podstatě všechny prvky, které nějakým způsobem spojí dva kontakty. Zařízení bude sestrojeno, opět jako u výstupního expandéru, z jednoduchých elektronických součástek a integrovaného obvodu. U komunikační sběrnice také nedojde ke změně. Toto zařízení nebude umístěno v elektrickém rozvaděči, ale ve zdi v elektroinstalační krabici KU pod vypínačem, nebo v místech, kde bude zapotřebí. Aby jej bylo možno umístit do KU

krabice je nutné zajistit malé rozměry tohoto zařízení. Opět veškerou logiku expandéru zajistí řídící jednotka.

2.2.4 Teplotní čidlo

Existují levné a k těmto účelům dostatečně přesné digitální teploměry. Tyto teploměry (obrázek 2.3) komunikují po sběrnici 1-Wire. Součástka malých rozměrů je ideální volba pro zajištění měření teploty. V každé místnosti bude umístěno jedno teplotní čidlo a může být schováno ve vypínači. Čidlo by se nemělo nacházet v blízkosti topení nebo u stropu. [7]



Obrázek 2.3: Digitální teploměr DS18B20 od společnosti Dallas

2.3 Ovládací panel

Ve chvíli, kdy je veškeré ovládání řešeno za pomocí webového rozhraní, je dle mého názoru zbytečné navrhovat a vytvářet složité obvody pro dotykové panely nebo ovládací klávesnice. Jednoduše lze použít tablet s operačním systémem Android, řádně nastavit tak, aby se vždy místo standartní aplikace Launcher pro zobrazení ikon na ploše, spustil webový prohlížeč s adresou hlavní řídící jednotky.

Navíc by bylo efektivní využít kameru k rozpoznávání pohybu v místě, kde se tablet nachází a podle toho zhasínat displej. Když jsem srovnal náklady za různé již hotové klávesnice nebo dotykové panely a náklady za tablet, jednoznačně vítězí tablet. Za cenu dotykového panelu nebo klávesnice lze pořídit tři levné tablety, které jsou pro tyto účely naprosto dostačující.

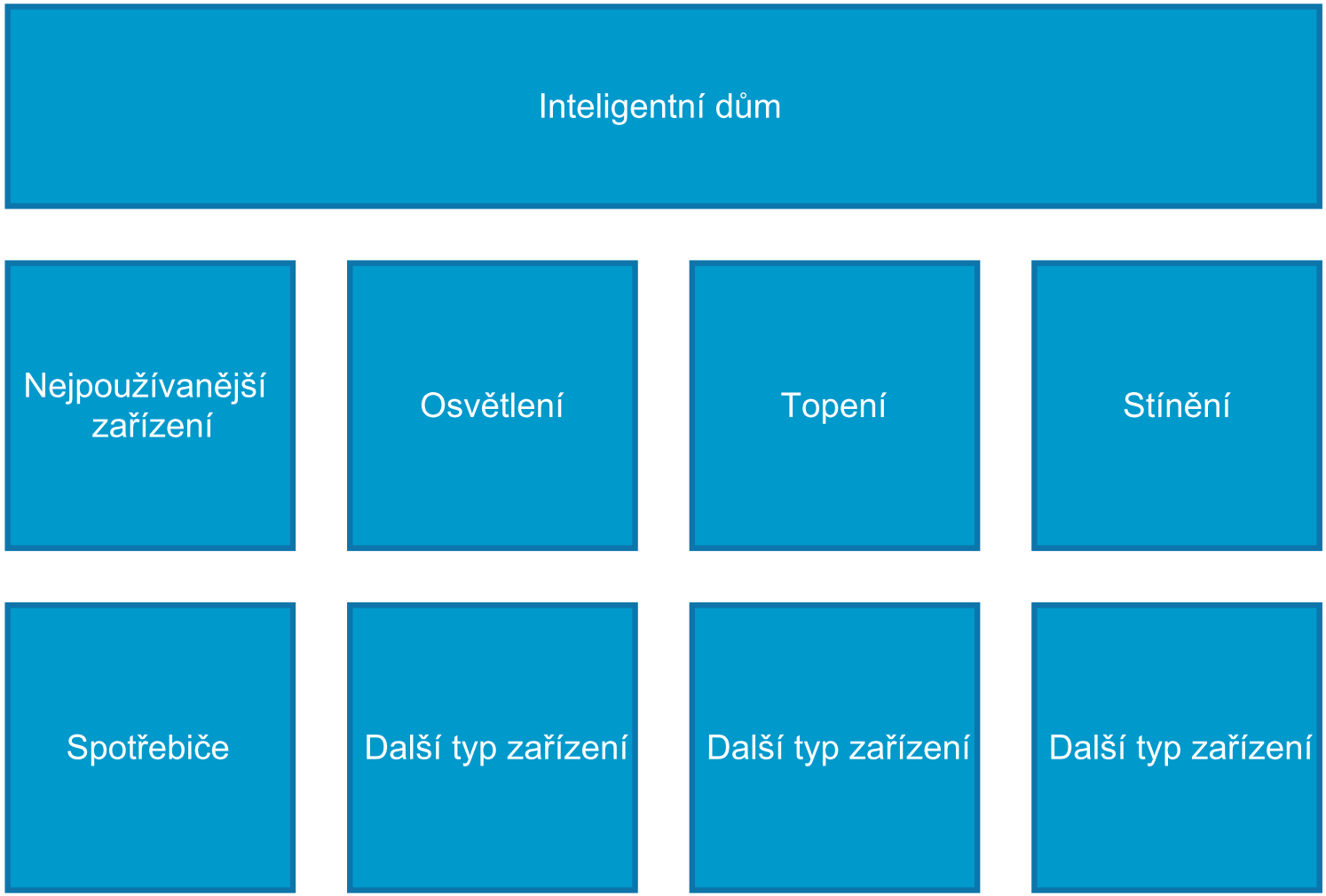
Na ovládacím panelu, respektive v ovládacím rozhraní, by mělo být docílení uživatelova požadavku velice rychlé. Bylo by nepohodlné proklikávat se několika obrazovkami, aby uživatel například rozsvítil světlo. Z toho důvodu je vhodné, aby uživatel odpověděl maximálně na dvě otázky, které jsou jaký typ zařízení chce ovládat a co s tímto zařízením chcete udělat.

Úvodní obrazovka bude obsahovat typy všech zařízení a také výčet nejpoužívanějších zařízení. Tímto odpovíme jedním dotykem na otázku jakého typu jsou zařízení obrázek 2.4. Po zvolení typu zařízení se zobrazí výčet všech zařízení daného typu s možnostmi

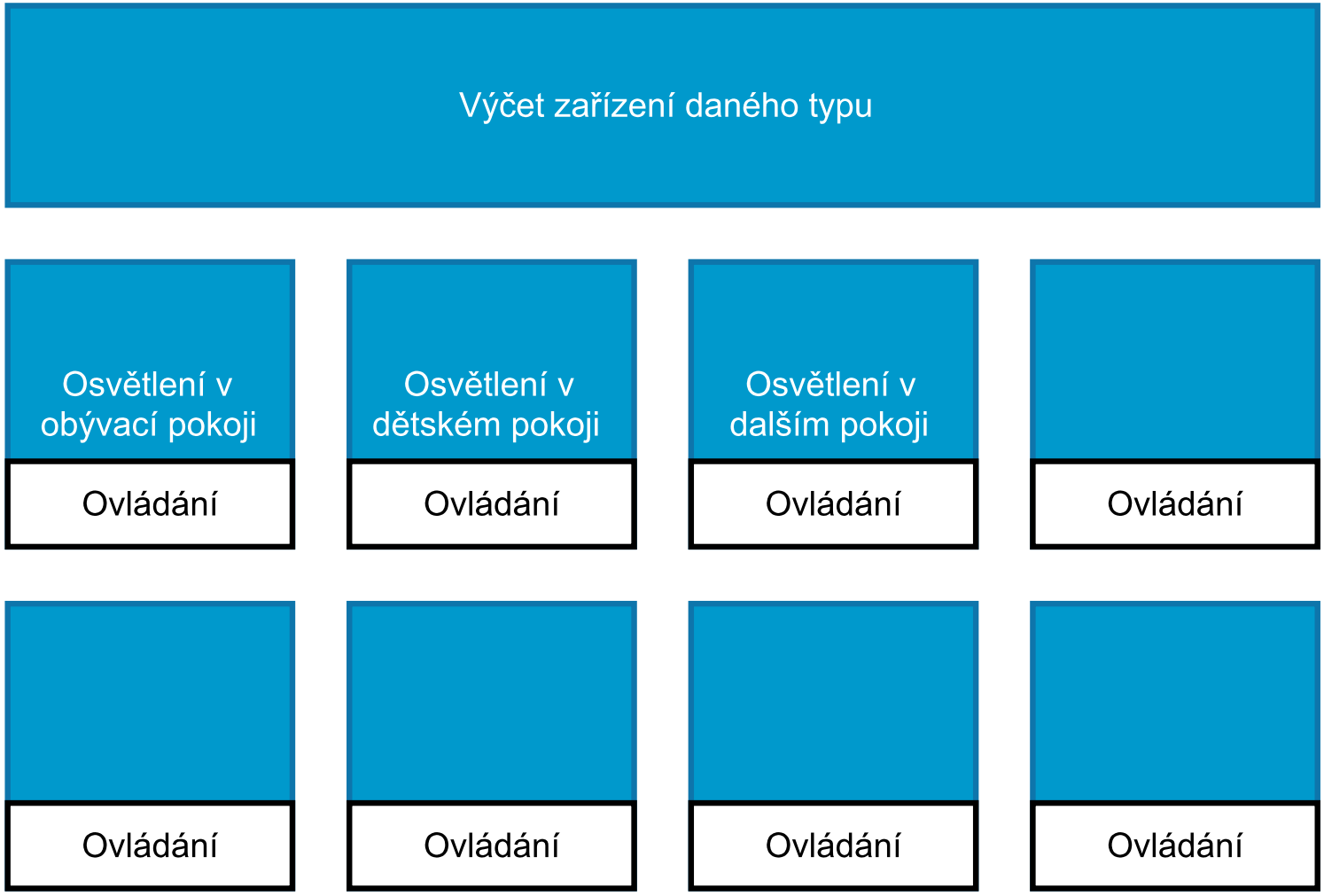
ovládání. Ukázka na obrázku 2.5. V této obrazovce může uživatel systému říci, co chce se zařízením udělat.

Samozřejmě, že uživateli nic nebrání v otevření ovládacího webového rozhraní na svém mobilním telefonu, nebo dokonce na televizi s multimediálním centrem a připojením k síti.

Způsob ovládání pomocí webového rozhraní má své výhody, ačkoliv bychom měli zvážit otázku bezpečnosti. V momentě, kdy je systém přístupný ze sítě, mají k němu přístup i ostatní. Z toho důvodu je provoz webového rozhraní bez přihlašování pro vybrané IP adresy a ostatní se musí řádně přihlásit svými přihlašovacími údaji.



Obrázek 2.4: Náčrt uživatelského rozhraní hlavní obrazovky ovládacího panelu



Obrázek 2.5: Náčrt uživatelského rozhraní obrazovky s výčtem zařízení

3 Řešení inteligentního domu

Nyní si ukážeme všechna zařízení, která tvoří inteligentní dům, popíšeme jejich zapojení a funkčnost. Bude se jednat o fyzická zařízení zmíněná v sekci Zařízení 2.2.

3.1 Základna

3.1.1 Délka sběrnice

Problémem použití sběrnic 1-Wire a I2C bez úpravy napěťových úrovní je parazitní kapacita vedení. Díky tomu, že se vyskytuje tato kapacita na vedení je digitální signál deformován. Napěťové úrovně sběrnic jsou 5V nebo 3,3V.

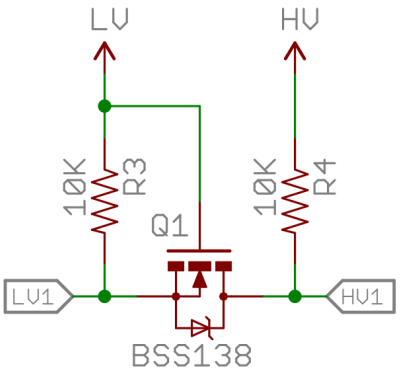
Výstupní proud číslicových obvodů je zpravidla v řádech mikroampérů. Pro zvednutí proudu na sběrnici se využívají tzv. pull-up rezistory. Pull-up rezistor je připojen k napájecímu napětí a druhý vývod je připojen ke sběrnici. Napětí pull-up rezistoru musí být stejné nebo menší jako na sběrnici. Pokud by nebylo, proud by tekl směrem do zařízení, které vysílá logickou jedničku. To by mohlo zapříčinit zničení vnitřních součástek číslicového obvodu. Úlohou pull-up rezistoru je zvýšení proudu při napěťové úrovni logické jedničky. V situaci kdy master vysílá a slave naslouchá, teče proud z jednoho číslicového obvodu do druhého. Zároveň se k němu přičítá proud dodávaný ze zdroje omezený pull-up rezistorem.

Na vedení vzniká parazitní kapacita a tento jev omezuje délku vedení na krátké vzdálenosti při vyšších frekvencích přenášeného signálu. Jednoduché řešení tohoto problému je zvýšení napěťových úrovní a proudu protékajícím sběrnicí. Číslicové obvody využívající I2C sběrnici mají zpravidla limitovaný proud sběrnice, který může téct do obvodu při logické nule na desítky miliampér. Proto řešení pomocí pull-up rezistoru můžeme využívat pouze částečně. [3]

Pro shrnutí je potřeba, aby:

- do číslicových obvodů tekl proud, který nepřesáhne desítky miliampér,
- sběrnice mohla dosahovat delších rozměrů v řádu kilometrů,
- nenarušit obousměrnost sběrnice,
- frekvence sběrnice nebyla příliš nízká, určeme spodní hranici na 5kHz.

Je třeba sestavit jednoduchý obvod na zvýšení napěťových úrovní, zvýšení proudu a zachovat obousměrnost sběrnice. Lze toho docílit využitím unipolárních tranzistorů typu N s ochrannou diodou a překlápěcím napětí odpovídající TTL úrovním. Obrázek 3.1 ukazuje schéma obousměrného převodníku napěťových úrovní. Na svorce LV1 jsou napěťové úrovně nižší a na svorce HV1 vyšší. Velikost napěťových úrovní určuje napětí LV a HV. Při logické jedničce na LV může strana HV „shodit" sběrnici do logické nuly díky ochranné diodě. Při logické jedničce na straně HV může LV sběrnici přenést do stavu logické nuly díky tomu, že je na bráně tranzistoru přivedené napětí a proud protéká přes tranzistor



Obrázek 3.1: Schéma obousměrného převodníku napěťových úrovní

z elektrody source do elektrody drain. Použitím dvou těchto obvodů zajistíme zvednutí napěťových úrovní a následné snížení.

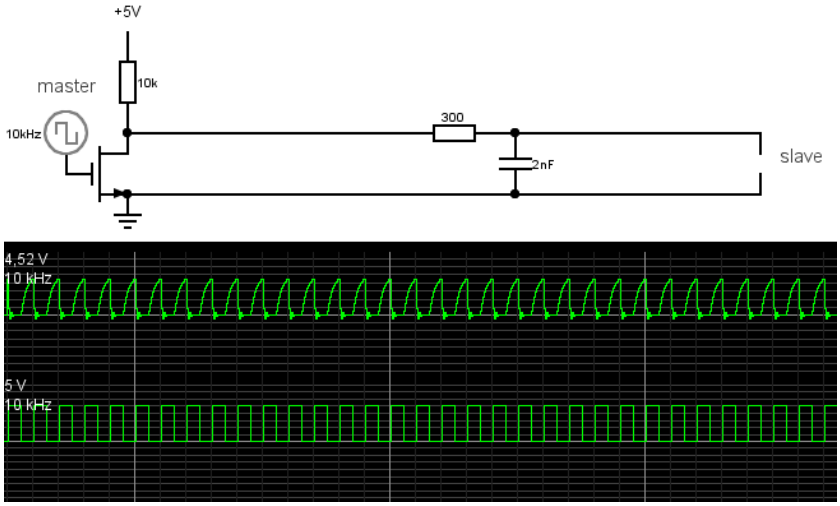
Simulací lze zjistit jaký je limit délky vedení při použití výše zmíněného převodníku. Vzhledem k použití kabelu UTP CAT5 počítejme s průměrnou kapacitní nerovnováhou 330pF na 500 metrů a s měrným elektrickým odporem vodiče 100Ω na 1km. Pro vodič můžeme tedy najít náhradní schéma ve formě integračního článku, kde je velikost kondenzátoru úměrná délce vedení. Řekněme, že se v domě využije vedení dlouhé až 3km. Potom je kapacita takového vedení 1980pF a odpor 300Ω.

Provedl jsem simulaci obvodu v programu Circuit Simulator [14], abych viděl jak bude vypadat přenesený signál po sběrnici. Hodnoty součástek a frekvence sběrnice jsou uzpůsobeny případu délky vedení 3km. Na obrázku 3.2 je dole průběh vstupního signálu sběrnice a nad ním průběh signálu na konci sběrnice. Jak je vidět je deformace za hranicí čitelnosti z hlediska číslicového obvodu. Tranzistor, rezistor a generátor tvoří výstup číslicového obvodu. Kondenzátor mění svou impedanci, respektive reaktanci v závislosti na frekvenci a to tak, že čím vyšší frekvence, tím menší impedance. Proto je zvolená frekvence 10kHz, aby byla délka vedení co největší. [14]

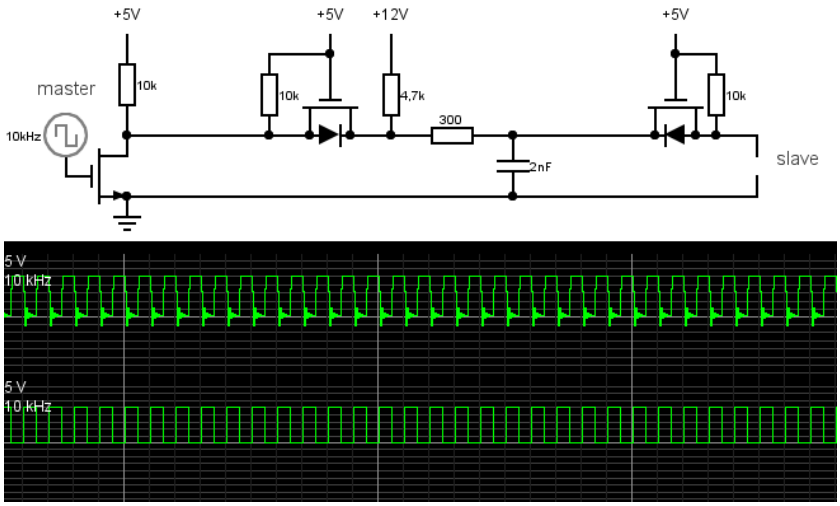
U simulace s převodníkem (obrázek 3.3) signál není tak zdeformovaný. Jediný problém, který by mohl působit na přenos jsou zákmity při sestupné hraně signálu, ale tyto zákmity dosahují maximálně 1V a jsou zcela zanedbatelné v tomto případě.

3.1.2 Schéma a návrh desky plošných spojů

Z výše uvedených simulací a určení součástek jsem vytvořil schéma zapojení součástek a desku plošných spojů v programu EAGLE. Na obrázku A.2 a A.3 jsou schémata a deska plošných spojů. Ve schématu je konektor pro připojení GPIO pinů Raspberry Pi označen SV1 a vidlice do dps pro nasazení šroubovacích svorek. Základna (obrázek A.4) je v podstatě tvořena pouze konektory a třemi převodníky napěťových úrovní, které převádí úrovně z 3,3V na 12V a zpět. Tři převodníky jsou z toho důvodu, že pro I2C sběrnici jsou



Obrázek 3.2: Simulace obvodu bez napěťového převodníku[1]



Obrázek 3.3: Simulace obvodu s napěťovým převodníkem[1]

využity dva a to pro přenos linky SDA a SCL. Třetí převodník je určen pro sběrnici 1-Wire. Z této základny se připojí všechny další zařízení, se kterými bude potřeba komunikovat.

3.2 Výstupní expandér

Hlavní součástka výstupního expandéru je integrovaný obvod PCF8574. Je to I2C osmibitový expandér, z čehož vyplývá, že bude možno spínat osm zařízení. Jeho výstupy jsou připojeny do báze bipolárního tranzistoru typu PNP přes bázový rezistor. PNP tranzistor

[1]Unipolární tranzistor N kanál je značen nestandartním způsobem.

spíná elektromagnetické relé a LED diodu pro signalizaci. Vzhledem k tomu, že je zde PNP tranzistor, sepnutý stav je při logické nule. Je to z toho důvodu, že PCF8574 je po vnitřním resetu uveden do stavu, kdy jsou všechny výstupy v logické jedničce. Kdyby se spínalo logickou jedničkou, před resetováním by na dobu trvání resetu byla relé sepnuta. Výstupní expandér také obsahuje napěťový převodník, aby převedl 12V úrovně sběrnice I2C na 5V. U výstupního expandéru jsou tři malé přepínače v řadě pro volbu tří bitové části adresy. Z toho vyplývá, že je systém omezen počtem výstupních expandéru na osm. Schéma[1] a deska plošných spojů jsou na obrázcích A.5 a A.6. Hotový výstupní expandér je na obrázku A.7. [5]

Toto zařízení bude umístěno v krabičce na DIN lištu a upevněno do elektrického rozvaděče. K němu budou připojeny všechny silové spínací prvky, jako jsou okruhy osvětlení, topení, rolety apod.

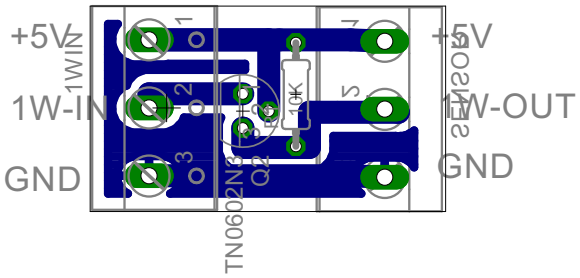
3.3 Vstupní expandér

Vstupní expandér slouží k monitorování vstupních zařízení, což jsou přepínače, vypínače, tlačítka a kontakty. Hlavní součástka vstupního expandéru (obrázky A.8 a A.9) je stejná jako u výstupního, jen má jinou část adresy danou výrobcem. PCF8574(A) obvod, jak jsem zmínil u výstupního expandéru, má po resetu na vývodech P1-P7 logické jedničky. Tyto vývody jsou přes ochrannou diodu přivedeny na svorky, ke kterým se připojí spínací vstupní prvky. Tyto prvky tedy spínají vývody expandéru se zemí. Řídící jednotka poté detekuje logickou nulu jako sepnuto. Deska plošných spojů je malých rozměrů, aby ji bylo možno umístit do KU krabice ve zdi pod vypínač, nebo na místa, kde bude za potřebí. Z expandéru se poté vyvede 8 vstupních zařízení, jak je znázorněno na obrázku A.1. Vypínač, pod kterým je schován vstupní expandér označuji jako master vypínač. Další vypínače, které jsou připojeny k tomuto master vypínači, označuji jako slave. Osazený vstupní expandér je na obrázku A.10 a A.11. [5]

3.4 Teplotní čidlo

Digitální teploměry DS18B20 (obrázek 2.3), využívající sběrnici 1-Wire pro komunikaci, jsou velice malé a pro tyto účely mají dostačující přesnost. Proto jsem zvolil tyto teploměry jako nástroje pro měření teplot. Teploměry budou připevněny do svorek převodníku. Součástí teploměru musí být tedy převodník napěťových úrovní. Do převodníku (obrázek 3.4) bude připojena 12V sběrnice 1-Wire, která začíná v řídící jednotce a směrem z převodníku bude připojen do svorek teploměr. Každý digitální teploměr DS18B20 má svou osmibajtovou adresu, která je dána výrobcem. Pro rozeznání o jaký typ zařízení se jedná, výrobce určil tzv. kód typové řady (family code). U DS18B20 je family code $28h$. Tento family code je součástí adresy. Adresa může vypadat například takto $28\ 00\ 00\ 05\ 5A\ E1\ 47\ FA$. První bajt adresy je family code, tudíž můžeme říci, že se jedná o DS18B20. Poslední bajt adresy je kontrolní součet adresy, CRC. [7]

[1] Značky součástek nemusí odpovídat. Například tranzistor BS108 ve schématu nemá ochrannou diodu, ačkoliv ji opravdu obsahuje.



Obrázek 3.4: Deska plošných spojů převodníku pro teploměr DS18B20

4 Software řídící jednotky

Jedna z možností volby programového prostředí byla použít Linuxový shell a všechna data ze zařízení přečíst a uložit do databáze. Prostřednictvím webového rozhraní by tedy uživatel pouze upravoval data v databázi a server, který je napsaný v Linuxovém shellu, by na základě těchto dat vykonával úlohy.

V takovém případě by bylo programování webového rozhraní velice komplikované a systém by ztrácel na modularitě. Navíc by byl rozdělen do částí shell, databáze a webové rozhraní. Detekce problému by poté byla velice zdlouhavá.

Další z možných řešení bylo navrhnout podobný systém a to vytvořit ASP webový server. Toto by spojilo všechny části systému do jednoho. Databáze by byla součástí serveru, ASP by zařizovalo webové rozhraní a služba by se starala o přenos dat. Bohužel platforma .NET není příliš podporována operačním systémem Raspbian, ačkoliv provoz .NET aplikací na platformě mono je dnes běžná věc. Navíc knihovny pro práci s GPIO na Raspberry Pi psané v .NETu nejsou příliš vyvíjeny.

Poslední a nakonec zvolenou možností bylo využívat technologii Java. Jediný problém, který jsem na této technologii viděl byl náročný webový server. Existuje projekt Jetty, který umožňuje vytvořit webový server s podporou servletů v Java aplikaci. Navíc Jetty není výpočetně náročný. Navíc je mnoho projektů v Javě pro ovládání GPIO pro Raspberry Pi. Nejrozšířenější z nich je Pi4J. [12]

Po testování provozu databáze na Raspberry Pi, jsem se rozhodl využívat jiné datové úložiště, a tak docílit co nejmenšího vytížení procesoru a paměti RAM.

4.1 Hlavní aplikace

Všechna fyzická zařízení připojena k Raspberry Pi přes základnu budou ovládána právě touto aplikací. Také tato aplikace bude poskytovat webové rozhraní a ukládat nastavení do datového úložiště. Tuto aplikaci budu dále označovat jako server. Server bude spouštět Linuxová služba, která jej po ukončení znovu nastartuje. [8]

4.2 Funkce serveru

Ve spouštěcím bodě, který je v singletonu Daemon, se spustí vlákno Daemonu a vlákno webového serveru Jetty viz Zdrojový kód 1. V tomto vlákně se vytvoří objekt z třídy DevicePool a zavolá na něj funkci Load, která načte z datového úložiště (xml soubory) informace o všech zařízeních a jejich konfiguraci. Také se načítají informace o tom, v jaké místnosti se zařízení nacházejí. [13]

Jak jsem výše zmínil, fyzická zařízení jsou připojena na sběrnici a dále do Raspberry Pi. Takže Raspberry Pi ovládá pouze fyzická zařízení. Dále jsou označovány jako virtuální zařízení ta, která jsou připojena do fyzických zařízení, tzn. zařízení, která jsou ovládány fyzickými zařízeními. Například výstupní expandér je fyzické zařízení, do kterého jsou připojeny světla, elektrické topení, nebo další spínací prvky a tyto zařízení jsou virtuální. Raspberry Pi ovládá pouze fyzická zařízení, která ovládají virtuální, takže Raspberry Pi ovládá nepřímo virtuální zařízení.

Každý typ zařízení má svou třídu. Z těchto tříd se vytváří objekty na základě načtených informací z datového úložiště. Třídy, které symbolizují vstupní zařízení spouští své vlákno pro čtení dat ze sběrnic, aby bylo možno spouštět události, na které jsou připojeny posluchači.

Každé zařízení má své identifikační číslo. Pod tímto číslem jsou vedena všechna zařízení uložena v hešovacích tabulkách. Tyto tabulky jsou ve třídě DevicePool. Prostřednictvím DevicePool objektu se tedy přistupuje ke všem načteným zařízením, jak k virtuálním, tak k fyzickým.

Po načtení těchto objektů a spuštění jejich vláken začne spouštěcí vlákno programu čekat na vstup příkazů pro debugování. Tento proces je spuštěn ve třídě Daemon, která je singleton. Prostřednictvím singletonu Daemon lze přistupovat k DevicePool objektu, ve kterém jsou uloženy všechny prvky inteligentního domu. Tímto je zajištěno vrstvení systému. Monitorování zařizují vlákna každého vstupního zařízení a ovládání vyplývá z těchto pozorovaných událostí. Také lze ovládat prostřednictvím webového rozhraní.

Na jaké události vstupních zařízení, a která výstupní zařízení mají reagovat, určuje konfigurační xml soubor. Například o světle, které je v obývacím pokoji a má být rozsvicováno vypínačem, je v konfiguračním xml souboru uložena informace o tomto propojení. DevicePool se postará o propojení posluchačů na události. V uvedeném příkladě by tedy DevicePool připojilo posluchače světla v obývacím pokoji na událost, kterou vyvolává vypínač.

Po načtení všech zařízení se vytvoří webový server Jetty. Odešlou se GET požadavky na všechny stránky, aby se spustila kompilace JSP souborů, která zabírá delší čas. Díky tomu uživatel nepocítí dlouhé načítání webového rozhraní.

```
public class Daemon implements Runnable {
    public static boolean debug = true;
    private static Daemon daemon;

    public static Daemon getDaemon() {
        if (daemon == null)
            daemon = new Daemon();
        return daemon;
    }

    private Thread mainThread;
    private boolean running;

    public static void main(String[] args) {
        daemon = getDaemon();

        WebAppContext context = new WebAppContext();
        context.setDefaultsDescriptor(" ./src/com/autohome/web/WEB-INF/webdefault.xml");
        context.setDescriptor(" ./src/com/autohome/web/WEB-INF/web.xml");
        context.setResourceBase("./src/com/autohome/web/");
        context.setContextPath("/");
        context.setTempDirectory(new File("./tmp/"));

        Server server = new Server(8080);
        server.setHandler(context);
```

```
            daemon.start();

            try {
                server.start();
            } catch (Exception e) {
                e.printStackTrace();
                System.exit(-1);
            }

            try {
                daemon.join();
                server.stop();
            } catch (InterruptedException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        ...
    }
```

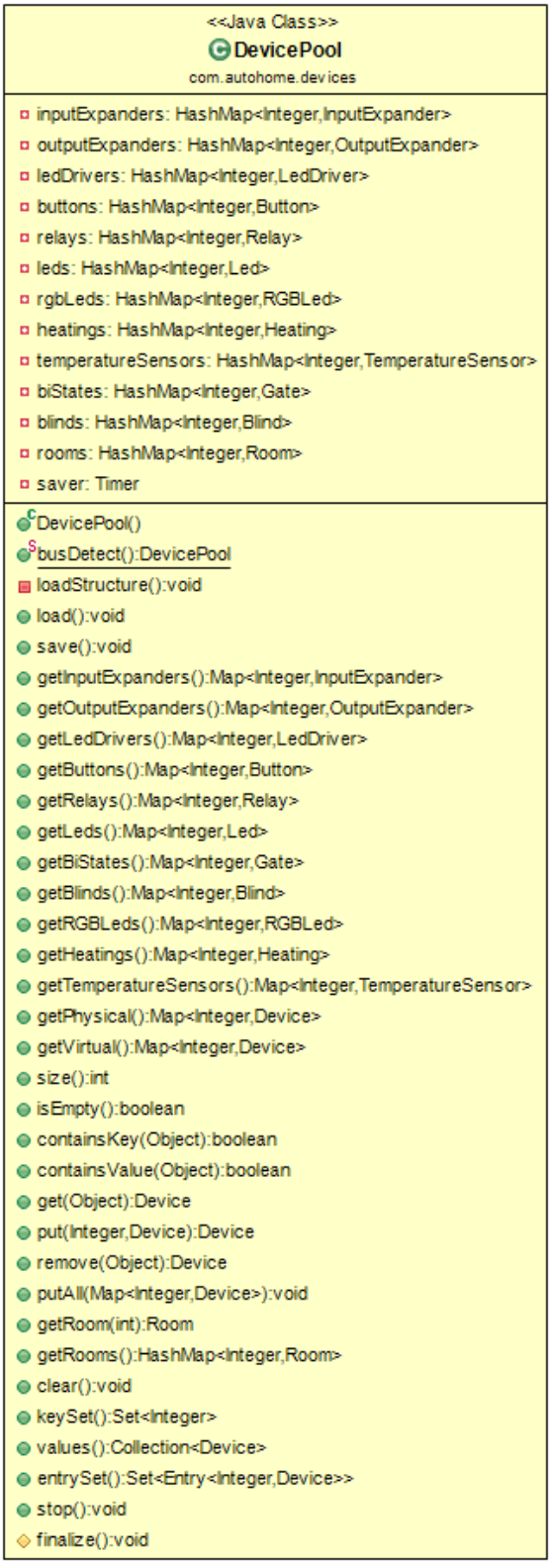
Výpis 1: Daemon singleton

4.3 DevicePool

Tato třída (obrázek 4.1) uchovává všechna zařízení roztřízená do hešovacích tabulek podle typu a implementuje rozhraní Map<Integer, Device>. Z této třídy lze přistupovat ke všem zařízením pomocí identifikačního čísla zařízení. Nebo si lze vyžádat zařízení ze zvolené místnosti. Aby bylo jednoduché přistupování k zařízením z místnosti, funkce pro získání zařízení ze zvolené místnosti vrací objekt typu DevicePool. V podstatě by se dalo říci, že se jedná o odrůdu návrhového vzoru kompozit.

Instance této třídy je centrum celého inteligentního domu. Zde se přistupuje k veškerému dění. Webové rozhraní využívá jako svou controller komponentu pro přístup k datům právě tuto instanci třídy. Úpravou funkcí Load a Save třídy DevicePool lze změnit způsob ukládání dat.

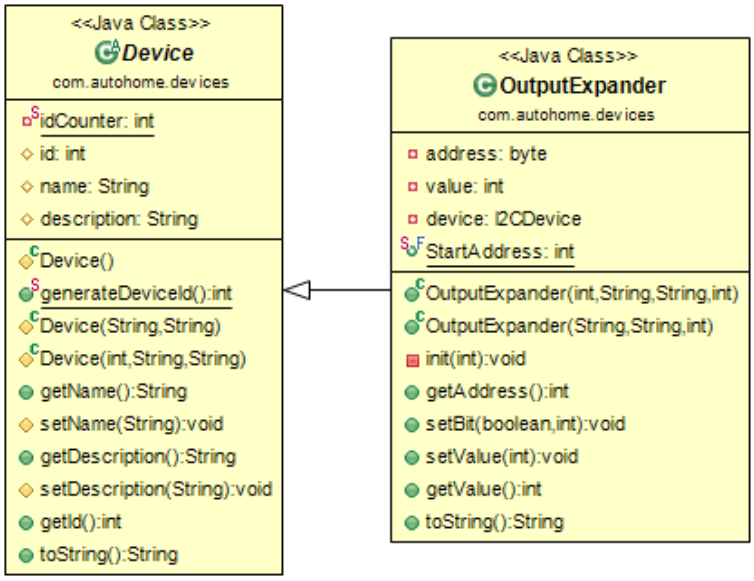
Úkolem této třídy je také uložení aktuálního stavu všech zařízení zpět do datového úložiště. Toto ukládání probíhá periodicky.



Obrázek 4.1: Třídní diagram třídy DevicePool

4.4 OutputExpander

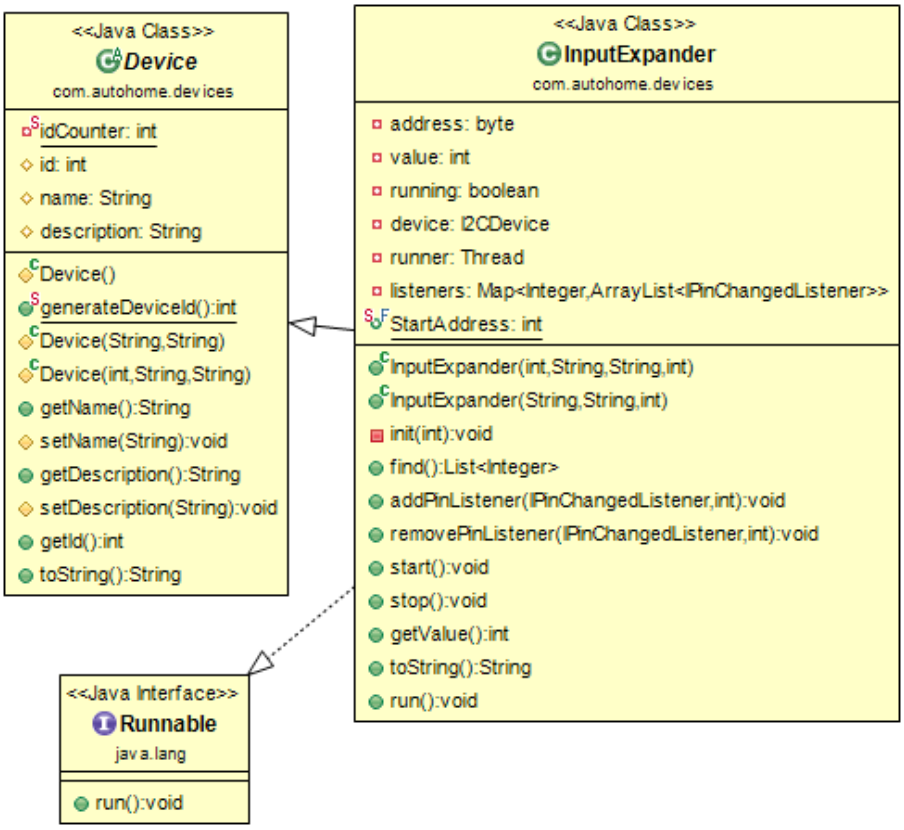
Instance této třídy (obrázek 4.2) vyjadřuje výstupní expandér (obrázek A.7). Třída dědí z třídy Device a uchovává informace o I2C adrese, aktuální nastavenou hodnotu výstupní brány I2C expandéru. Třídní proměnná device typu I2CDevice je z knihovny Pi4J se stará o komunikaci po I2C sběrnici se zařízením. Obsahuje funkce pro nastavení bitu expandéru nebo celého bajtu. Parametry konstruktorů jsou identifikační číslo, jméno, popis a adresa. Identifikační číslo je nepovinný parametr. Pokud tento parametr není předán, vytvoří se zařízení a přiřadí se mu nově vygenerované identifikační číslo.



Obrázek 4.2: Třídní diagram třídy OutputExpander

4.5 InputExpander

Objekt vytvořený z této třídy symbolizuje vstupní expandér 3.3. Třída (obrázek 4.3) dědí z třídy Device a uchovává stejné informace jako třída OutputExpander 4.4. Objekt přijímá data z I2C expandéru ve vlákně runner a to tak, že v cyklu využívá proměnnou device typu I2CDevice z knihovny Pi4J. V tomto vlákně se detekuje změna hodnoty a spouštějí se posluchači na změnu stavu vstupního pinu. Proměnná listeners typu Map<Integer, ArrayList<IPinChangedListener>> uchovává posluchače změny jednoho bitu vstupní brány expandéru.



Obrázek 4.3: Třídní diagram třídy InputExpander

4.6 TemperatureSensor

Vzhledem k tomu, že je teplotní čidlo vstupní zařízení, tak tato třída implementuje Runnable a spouští své vlákno. V tomto vlákně se čte teplota ze sběrnice 1-Wire.

Existují moduly jádra pro Raspberry Pi, které umožňuji na GPIO4 komunikovat se sběrnicí 1-Wire a také moduly, které čtou teploty z DS18B20. Použitím tohoto modulu lze poté jednoduše číst teplotu zařízení a také detekovat, zda čtení proběhlo správně. [8]

Po zavedení modulu do systému se ve složce /sys/bus/w1/devices/ objeví složky, jejichž jména jsou adresy zařízení. První dvojčíslí je family code, zbytek je adresa – bez kontrolního součtu (CRC). Ve složce s adresou zařízení se nachází soubory pro ovládání nebo čtení teploměru. Soubor w1_slave obsahuje přenesená data z paměti zařízení, což je vlastně naměřená teplota a kontrolní součet. Tento modul také kontroluje kontrolní součet.

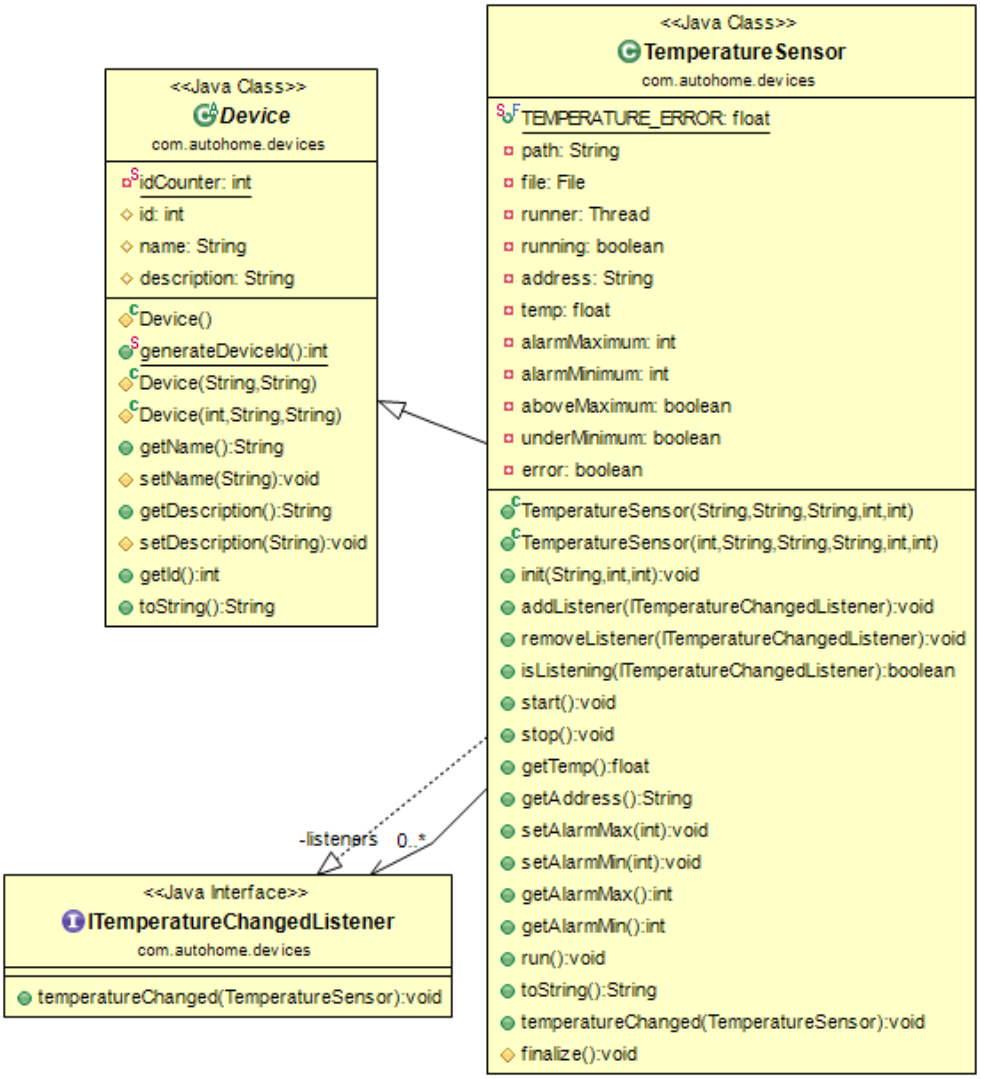
Jak je vidět z Výpisu 2, komunikace s digitálním teploměrem s adresou $28\,00\,00\,05\,5A\,E1\,47$ proběhla správně, protože CRC součet souhlasí a teplota je $22,187\,°\mathrm{C}$. Tímto způsobem lze tedy jednoduše naprogramovat čtení teplot, jako čtení textového souboru a následnou analýzou textu získat informace jako jsou teplota a zda komunikace proběhla správně. [7] [8]

```
$ cat /sys/bus/w1/devices/28-0000055ae147/w1_slave
63 01 4b 46 7f ff 0d 10 15 : crc=15 YES
63 01 4b 46 7f ff 0d 10 15 t=22187
```

Výpis 2: Čtení teploty z DS18B20 na Raspberry Pi

Třída také detekuje zda je teplota v uživatelem stanoveném intervalu. Pokud není teplota v tomto intervalu nebo dochází k chybám na teploměru odešle varovný email.

TemperatureSensor třída (obrázek 4.4) drží odkazy na ITemperatureChangedListener a spouští vždy funkci temperatureChanged všech posluchačů při změně teploty. Také implementuje toto rozhraní a zapojí se do naslouchání sama sebe, aby mohla reagovat na změny a odesílat emaily, pokud nastane chyba nebo teplota nebude v nastaveném pracovním intervalu.

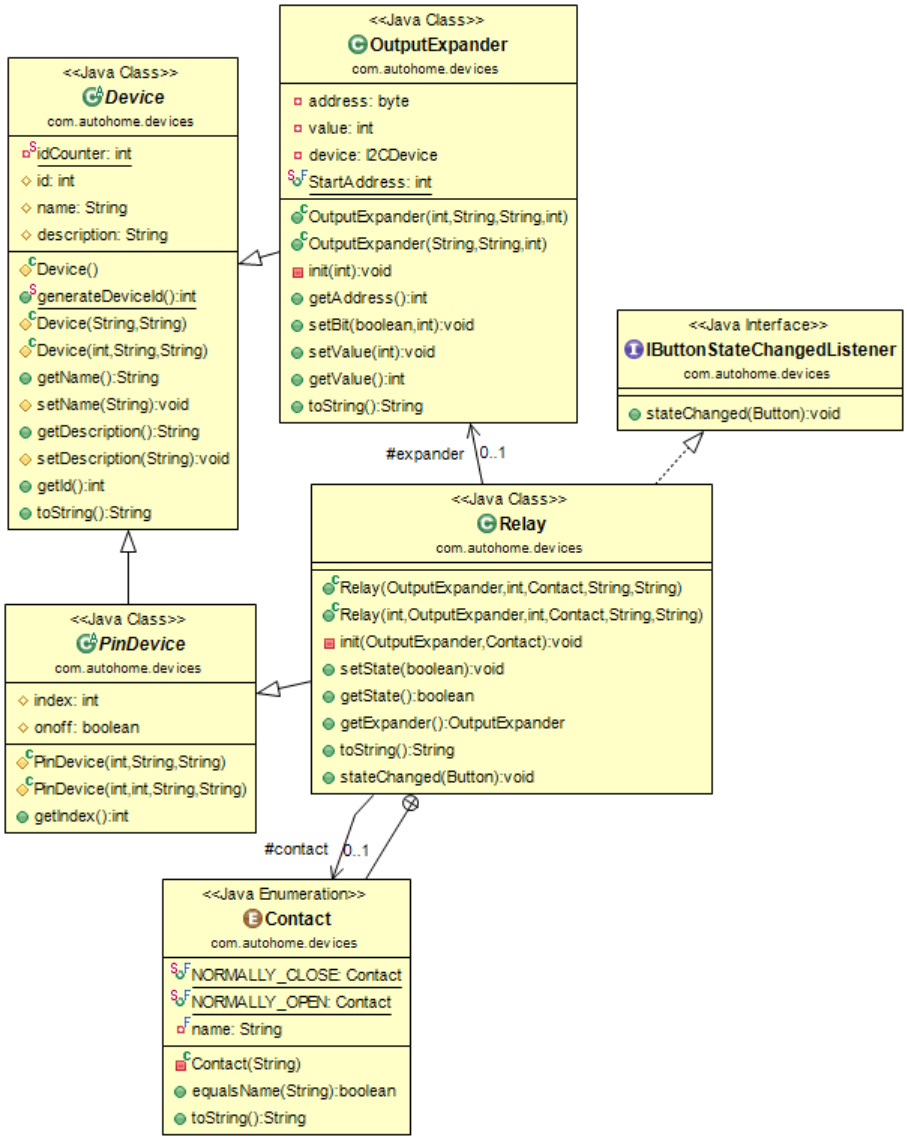


Obrázek 4.4: Třídní diagram třídy TemperatureSensor

4.7 Relay

Třída Relay (obrázek 4.5) symbolizuje jedno elektromagnetické relé, které je na výstupním expandéru. Dědí ze třídy PinDevice, ve které jsou členské proměnné index a onoff. Index je pozice pinu na zařízení, na kterém se nachází toto PinDevice zařízení. Vzhledem k tomu, že PinDevice dědí z Device má Relay také jméno, popis a identifikační číslo.

Relay implementuje rozhraní IButtonStateChangedListener. Je tedy posluchač na změnu stavu vypínače. Uchovává také OutputExpander, na kterém se nachází toto relé. Relé může být nakonfigurováno způsoby v klidovém stavu sepnuto nebo v klidovém stavu rozepnuto.



Obrázek 4.5: Třídní diagram třídy Relay

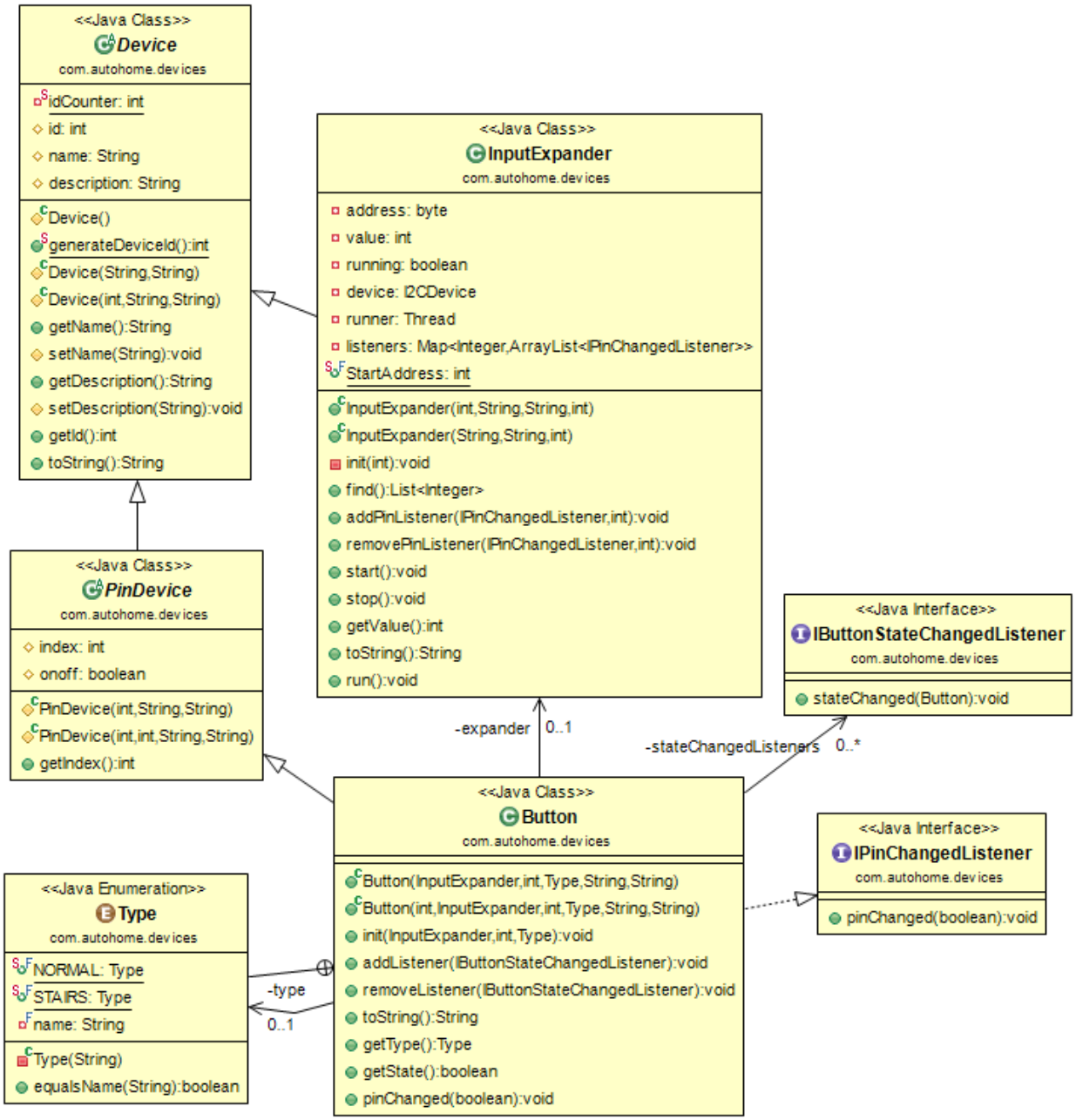
4.8 Button

Button (obrázek 4.6) je vyjádření vypínače. Button dědí z třídy PinDevice, jako to bylo u Relay 4.7, a uchovává InputExpander 4.5, ke kterému je tento vypínač připojen. Button implementuje IPinChangedListener, takže naslouchá změně pinu na vstupním expandéru. Také je držitelem seznamu IButtonStateChangedListener, aby mohl informovat posluchače na změnu stavu vypínače.

Vypínač definuje dva typy vypínačů. První je typ NORMAL. Jedná se o klasický vypínač. Funkce schodišťového vypínače zajišťuje typ STAIRS. Prvky, které naslouchají změně stavu, by měli počítat s tímto typem a je na jejich implementanci, jak se budou chovat.

Například Relay 4.7 je posluchač IButtonStateChangedListener a v implementaci funkce stateChanged se rozhoduje podle typu vypínače, zda se má nastavit tomuto relé stav sepnuto[2], pokud je vypínač typu NORMAL sepnut, nebo se se má při jakékoliv změně stavu vypínače typu STAIRS negovat stav relé. Druhý případ ukazuje realizaci schodišťových vypínačů.

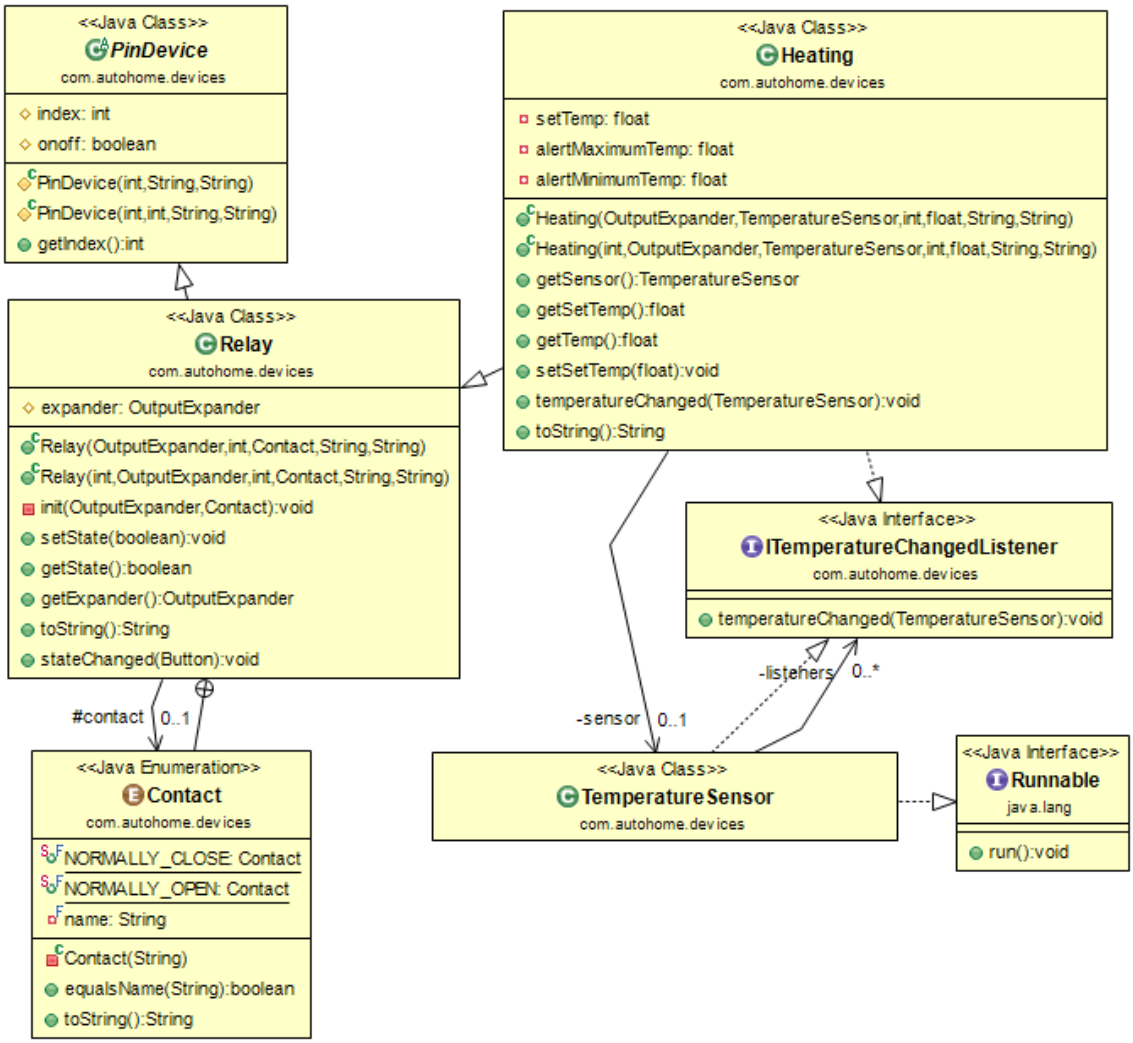
[2]Pokud je relé nastaveno tak, že v klidovém stavu je rozepnuto.



Obrázek 4.6: Třídní diagram třídy Button

4.9 Heating

Třída Heating (obrázek 4.7) je vyjádření elektromagnetického relé, které spíná topení. Heating dědí z Relay 4.7 a implementuje ITemperatureChangedListener. Také drží TemperatureSensor 4.6 a je posluchač změny teploty. Když se změní teplota digitálního teploměru, objekt typu TemperatureSensor 4.6 informuje objekt typu Heating, který je posluchačem tohoto teploměru a Heating sepne či rozepne topení v závislosti na nastavené teplotě.



Obrázek 4.7: Třídní diagram třídy Heating

4.10 Blind

Blind třída vyjadřuje stínící zařízení – interiérové rolety. Na těchto roletách jsou umístěny dva magnetické kontakty. První kontakt detekuje, zda jsou rolety zatažené a druhý dete-

kuje zda jsou úplně vytažené. Mechanická část rolet má dva kontakty. Při sepnutí prvního se rolety zatahují a při sepnutí druhého se vytahují.

Magnetické kontakty jsou zapojeny do vstupního expandéru 2.2.3 a mechanická část rolet je připojena do výstupního expandéru 2.2.2. Tímto můžeme rolety úplně zavřít, nebo úplně otevřít.

Implementace třídy Blind, ale umožňuje také polohovat procentuálně. A to tak, že se spočítá čas, za jak dlouho se rolety dostanou ze zataženého stavu do vytaženého. Díky pamatování tohoto času můžeme vypočítat, jak dlouho se mají rolety vytahovat/zatahovat, aby se dostaly do zvolené polohy. Ačkoliv tento systém není příliš bezpečný z hlediska nelinearity průběhu vytahování/zatahování rolet, zvolil jsem jej proto, že je jedním z nejjednodušších a nejlevnějších.

Po vytvoření a objektu typu Blind se spustí kalibrace (viz Výpis 3), která spočítá tento čas.

```
public void calibrate () {
    Log.getLog().println ("Blind: Starting calibration process.%s", this.toString () );
    outputExpander.setBit(true, outputIndexUp); // stop
    outputExpander.setBit(true, outputIndexDown); // stop
    outputExpander.setBit(false, outputIndexUp); // lets lift to top
    Log.getLog().println ("Blind: Calibration : Moving up to top to reach zero percent value.%s",
        this.toString());
    try {
        synchronized(calibrationTop) {
            calibrationTop.wait() ; // contact on top will wake us
        }
        if (!settingThreadRunning) return;
    } catch (InterruptedException e) { e.printStackTrace() ; }
    outputExpander.setBit(true, outputIndexUp); // stop, we are on top
    try {
        Thread.sleep(500);
    } catch (InterruptedException e1) { e1.printStackTrace() ; }
    Log.getLog().println ("Blind: Calibration : Blind is on top. Lifting down to measure time.%s"
        , this.toString());
    Date start = new Date();
    outputExpander.setBit(false, outputIndexDown); // lets lift to bottom
    try {
        synchronized(calibrationBottom) {
            calibrationBottom.wait() ; // contact on bottom will wake us
        }
        if (!settingThreadRunning)
            return;
    } catch (InterruptedException e) {
        e.printStackTrace() ;
    }
    outputExpander.setBit(true, outputIndexDown); // stop
    calibrationMs = getDateDiff( start , new Date(), TimeUnit.MILLISECONDS);
    Log.getLog().println ("Blind: Calibration : Completed. Calibration time is %d ms.%s",
        calibrationMs, this.toString());
}
```

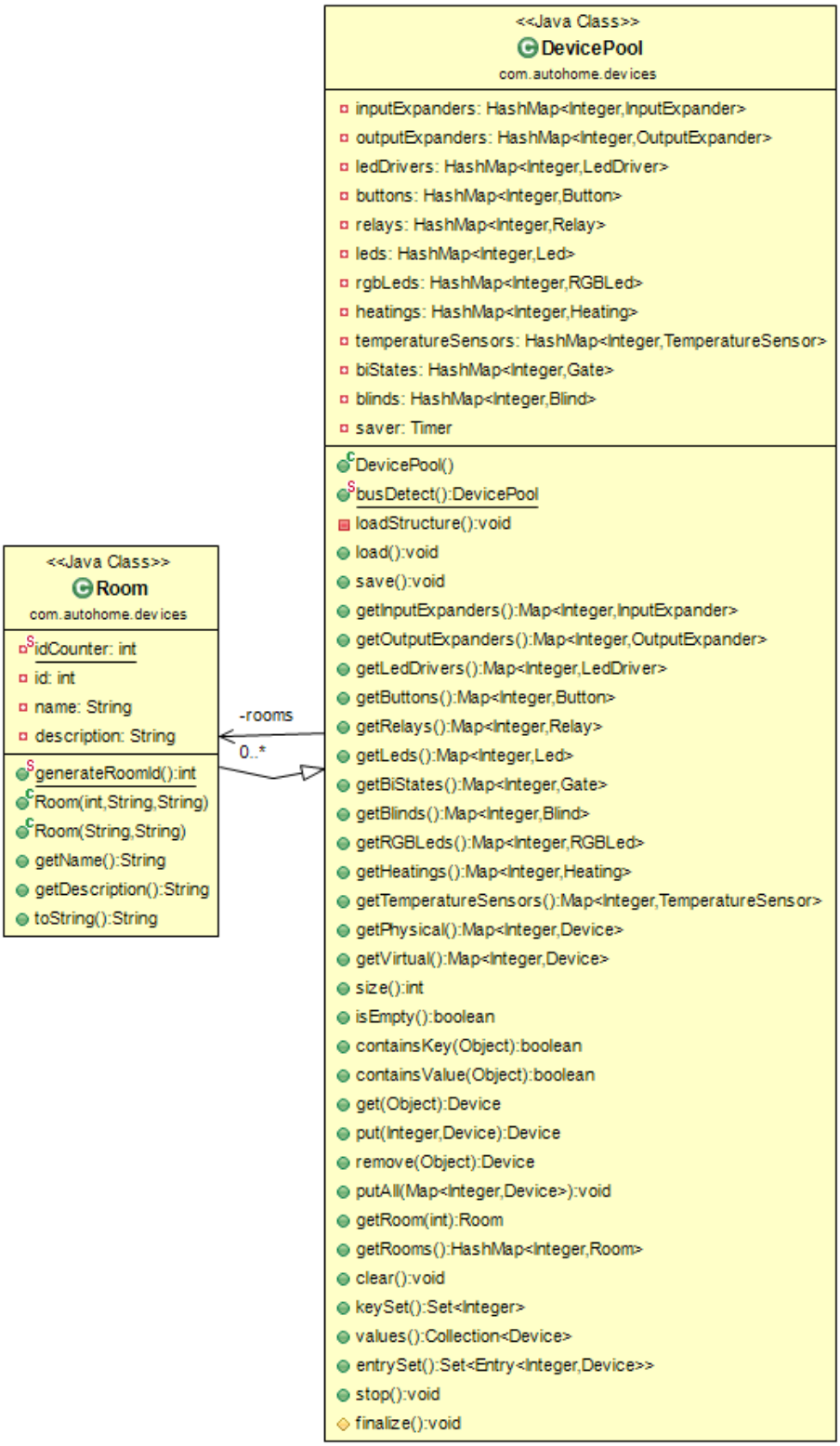
Výpis 3: Kalibrace rolet

4.11 Room

Tato třída (obrázek 4.8) vyjadřuje místnost domu. Uchovává název místnosti, její popis a všechna zařízení, která se v místnosti nachází. Dědí z třídy DevicePool 4.3, aby bylo zajištěno jednoduché vybírání zařízení z místností.

DevicePool 4.3 objekt načte informace o zařízeních z datového úložiště a propojí jejich posluchače. Poté načte místnosti, vytvoří objekty typu Room a přiřadí těmto objektům zařízení, která se nachází v dané místnosti.

Díky tomu, že Room dědí z DevicePool 4.3 lze poté jednoduchým způsobem vyžádat zařízení z místnosti podle typu nebo identifikátoru jak je tomu u DevicePool objektu.



Obrázek 4.8: Třídní diagram třídy Room

4.12 Log

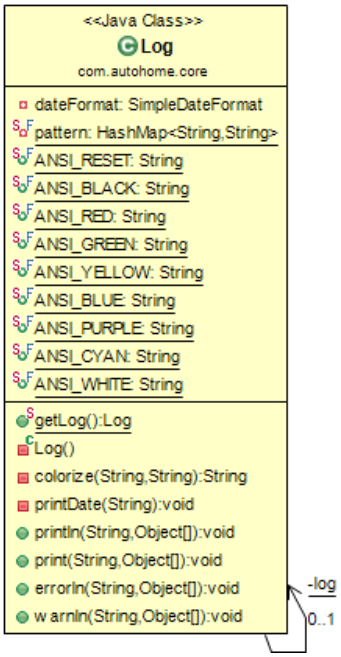
Jedná se o singleton (obrázek 4.9), který zapisuje vzniklé události do souboru a obarvuje názvy při výpisu na standartní výstup.

Jeho funkce jsou println pro výpis informace, warnln pro varování a errorln pro oznámení chyby. Každá tato funkce používá jinou barvu pro výpis na standartní výstup.

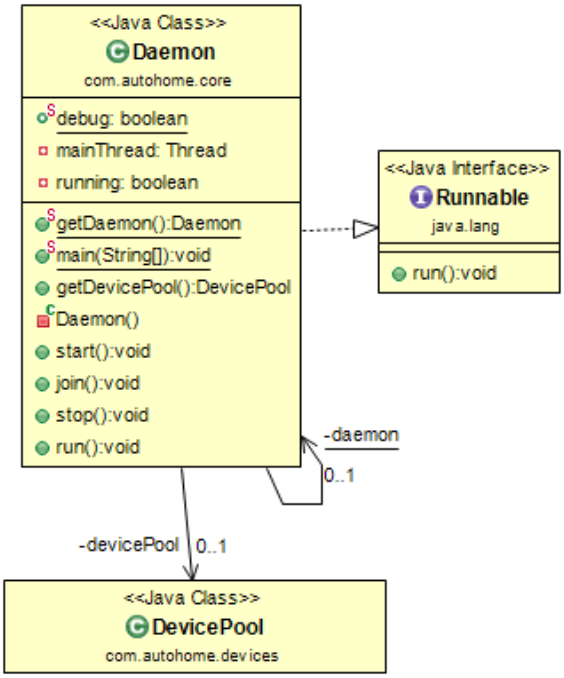
4.13 Daemon

V tomto singletonu (obrázek 4.10), jak jsem zmínil výše, je spouštěcí bod programu. Ve spouštěcím bodě se vytvoří hlavní vlákno a vlákno webového serveru Jetty. Tyto vlákna se spustí a hlavní spouštěcí bod čeká na ukončení hlavního vlákna a poté ukončí vlákno webového serveru Jetty.

Hlavní vlákno vytváří objekt typu DevicePool a dá povel objektu, aby načetl zařízení a informace o místnostech z datového úložiště a aby spustil automatické ukládání zařízení zpět do datového úložiště. Poté spustí všechna Runnable zařízení a čeká na ladící příkazy.



Obrázek 4.9: Třídní diagram třídy Log



Obrázek 4.10: Třídní diagram třídy Daemon

4.14 Webové rozhraní

Jak jsem výše zmínil webový server obstarává Jetty. Je to odlehčená verze Java webového serveru s podporou Java serveltů. Odpověď na otázku proč jsem zvolil Jetty je, že Jetty lze vestavět do aplikace. Aplikace poté může celý server zapínat (viz Ukázka zdrojového kódu 4) a ovládat.

```
public class Daemon {
    ...
    public static void main(String[] args) {
        WebAppContext context = new WebAppContext();
        context.setDefaultsDescriptor(" ./ src/com/autohome/web/WEB-INF/webdefault.xml");
        context.setDescriptor(" ./ src/com/autohome/web/WEB-INF/web.xml");
        context.setResourceBase("./src/com/autohome/web/");
        context.setContextPath("/");
        context.setTempDirectory(new File("./tmp/"));

        Server server = new Server(8080);
        server.setHandler(context);

        try {
            server.start();
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }
    ...
}
```

Výpis 4: Spuštění webového serveru Jetty

K tomu, aby se spustil Jetty webový server, je mu potřeba sdělit informace o konfiguraci serveru a také informaci o tom, kde se nachází kořenový adresář webového rozhraní. Soubor /WEB-INF/webdefault.xml obsahuje základní nastavení serveru. Po načtení webdefault.xml se načte /WEB-INF/web.xml (viz Výpis 5), který je v podstatě rozšíření základní konfigurace. V případě webového rozhraní inteligentního domu jsou v souboru web.xml nastavení tzv. welcome souborů, které při nezmínění souboru v URL zobrazí klientovi welcome soubor. Také jsou zde nastaveny stránky, které se zobrazí při HTTP chybě. V neposlední řadě je zakázán přístup do složky /pages/ a použit filtr UrlRewriteFilter pro úpravu URL adres. [13]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.
    com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>
  < filter >
      < filter -name>UrlRewriteFilter</filter-name>
      < filter -class>org.tuckey.web.filters. urlrewrite . UrlRewriteFilter </ filter -class>
  </ filter >
  < filter -mapping>
      < filter -name>UrlRewriteFilter</filter-name>
      <url-pattern>/*</url-pattern>
      <dispatcher>REQUEST</dispatcher>
      <dispatcher>FORWARD</dispatcher>
  </ filter -mapping>
  <error-page>
    <error-code>403</error-code>
    <location>/403.html</location>
  </error-page>
  <error-page>
    <error-code>404</error-code>
    <location>/404.html</location>
  </error-page>
  <security-constraint>
      <display-name>Restrict direct access to certain folders</display-name>
      <web-resource-collection>
          <web-resource-name>Restricted folders</web-resource-name>
          <url-pattern>/pages/*</url-pattern>
      </web-resource-collection>
      <auth-constraint />
  </security-constraint>
</web-app>
```

Výpis 5: Soubor web.xml

UrlRewriteFilter filtr načte ze souboru /WEB-INF/urlrewrite.xml regulérní výrazy a k nim jejich „podstrčenou" URL. Filtrem je docíleno „hezkých" URL. Například URL server[3]/main nezobrazí soubor main, ale vzhledem k definovaným pravidlům filtru v souboru urlrewrite.xml (viz Výpis 6) podstrčí adresu server/index.jsp?page=main.

[3] server ve smyslu adresy nebo doménového jméno serveru

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE urlrewrite PUBLIC "-//tuckey.org//DTD UrlRewrite 4.0//EN" "http://www.tuckey.org/res
    /dtds/urlrewrite4.0.dtd">
<urlrewrite>
    <rule>
        <from>/main</from>
      <to>/index.jsp?page=main</to>
    </rule>
   <rule>
      <from>^/lights$</from>
      <to>/index.jsp?page=lights</to>
   </rule>
   <rule>
      <from>/test</from>
      <to>/index.jsp?page=test</to>
   </rule>
   <rule>
      <from>^/thermo$</from>
      <to>/index.jsp?page=thermo</to>
   </rule>
   <rule>
      <from>^/gates$</from>
      <to>/index.jsp?page=gates</to>
   </rule>
</urlrewrite>
```

Výpis 6: Soubor urlrewrite.xml

V souboru index.jsp se sestavuje webová stránka z několika částí. Tyto části jsou uloženy v adresáři /pages/. První část je /pages/header.jsp, kde se vytváří HTML hlavička. Další část je závislá na URL adrese. Například URL adresa server/main, která díky UrlRewriteFilter filtru podstrčí server/index.jsp?page=main použije jako druhou část /pages/main.jsp. Soubor, který má být použit jako druhá část, není přímo text GET proměnné main. Poslední částí je footer.jsp. Kvůli tomuto rozdělení na části, je zakázán přístup k souborům v adresáři /pages/. V podstatě uživatel aniž by věděl, navštěvuje pouze stránku index.jsp, ikdyž se jeho URL adresa mění.

Než se sestaví webová odpověď z částí, ověří se uživatelova identita. Pokud není uživatel přihlášen, je mu zobrazena přihlašovací obrazovka a jsou zakázány veškeré interakce se systémem inteligentního domu. Také v souboru index.jsp je kód, který zpracovává všechny AJAX požadavky.

Díky tomu, že existuje webový server v aplikaci, je možné přistupovat ze servletů k třídám aplikace. Tohoto přístupu je využito ve všech JSP souborech. Například lights.jsp (viz Výpis 7) vypíše do HTTP odpovědi seznam všech světel a to tak, že si data o světlech načte z objektu typu DevicePool, který získá ze singletonu Daemon. Jeden z možných výstupů souboru lights.jsp je zobrazen ve Výpise 8.

```
<%@page contentType="text/html; charset=UTF-8" %>
<%@page import="com.autohome.devices.Room"%>
<%@page import="com.autohome.devices.Relay"%>
<%@page import="com.autohome.core.Daemon"%>
<%@page import="java.util.Map.Entry"%>
    <div id="listView" class="sublist">
      <% for(Entry<Integer, Room> room : Daemon.getDaemon().getDevicePool().getRooms().
          entrySet()) { %>
      <div class="separator"><%=room.getValue().getName() %></div>
      <% for(Entry<Integer, Relay> r : room.getValue().getRelays().entrySet()) { %>
      <a href="#" title ="" data-id="<%=r.getValue().getId()%>" class="relay-control subitem
          light-<%=(r.getValue().getState() ? "on" : "off")%>" data-state="<%=(r.getValue().
          getState() ? "on" : "off")%>">
       <div class="middle-wrapper light-wrapper">
         <p><%=r.getValue().getName() %></p>
       </div>
      </a>
      <% } %>
      <% } %>
    </div>
```

Výpis 7: Soubor lights.jsp

```
    <div id="listView" class="sublist">
      <div class="separator">Obyvaci pokoj</div>
      <a href="#" title="" data-id="106" class="relay-control subitem light-off" data-state="
          off">
       <div class="middle-wrapper light-wrapper">
         <p>Zarovka 2</p>
       </div>
      </a>
      <a href="#" title="" data-id="104" class="relay-control subitem light-off" data-state="
          off">
       <div class="middle-wrapper light-wrapper">
         <p>Zarovka 0</p>
       </div>
      </a>
      <a href="#" title="" data-id="105" class="relay-control subitem light-off" data-state="
          off">
       <div class="middle-wrapper light-wrapper">
         <p>Zarovka 1</p>
       </div>
      </a>
    </div>
```

Výpis 8: Možný výstup souboru lights.jsp

Tímto způsobem jsou řešena zobrazení všech typů zařízení. Veškeré další ovládání je ponecháno javascriptu.

Aby uživatel viděl, co se děje se zařízeními v reálném čase, je každý element, který symbolizuje zařízení, rozšířen o parametr data-id a svázán k jQuery události devicesLoaded. Každých 250ms zajišťuje jQuery načtení JSON objektu, který obsahuje informace

o všech zařízeních, která existují v tomto inteligentním domě. Po načtení tohoto objektu je vyvolána událost devicesLoaded a tento objekt předán. Všechny elementy, kterým tato událost spustí funkci, si vyžádají ze získaného JSON objektu zařízení, které symbolizují podle atributu data-id. Tento atribut symbolizuje unikátní identifikační číslo zařízení. Díky knihovně google-gson lze jednoduchým způsobem převést objekty z programovacího jazyka Java na JSON. Anotací @Expose lze označit členské proměnné tříd, které se zahrnou[4] do JSON objektu. [10] [9]

Například (viz Ukázka kódu 9) div s třídou relay-control si při načtení JSON objektu ze serveru změní pozadí v závislosti na tom, zda je zařízení zapnuto, či vypnuto. URL server/index.jsp?ajax=getDevices (viz Výpis 10) vypíše JSON objekt, který reprezentuje pole, kde jsou klíče identifikátory zařízení a hodnoty objekty reprezentující zařízení. Na obrázku A.12 je zobrazen pohled na prvky relay-control, respektive je zobrazena stránka server/lights. [9]

```
var loaderDisabled = false;

function startDevicesLoader() {
    if (!loaderDisabled)
        $.ajax({
            url: "/?ajax=getDevices",
            timeout: 5000, // timeout 5s
            success: function(data) {
                $("*").trigger({
                    type: "devicesLoaded",
                    devices: JSON.parse(data)
                });
                setTimeout(function() { startDevicesLoader(); }, 250);
            },
            error: function(x, t, m) {
                location.reload();
            }
        });
}

$(".relay-control").bind("devicesLoaded", function(e) {
    $(this).attr("data-state", e.devices[$(this).attr("data-id")].onoff ? "on" : "off");
    if ($(this).attr("data-state") == "on") {
        $(this).css("background-image", "url(./images/light-on.png)");
 } else if ($(this).attr("data-state") == "off") {
        $(this).css("background-image", "url(./images/light-off.png)");
 }
    $("body").css("cursor", "");
});
```

Výpis 9: Svázání elementu s událostí devicesLoaded

[4] Zda se zahrnou, či vyloučí závisí na nastavení převodu

```
{
  "106": {
    "contact": "NORMALLY_OPEN",
    "index": 2,
    "onoff": false,
    "id": 106,
    "name": "Zarovka 2",
    "description": "Zarovka v tretim pokoji"
  }
}
```

Výpis 10: Možná odpověď z URL server/index.jsp?ajax

Pro změnu stavu zařízení (viz Ukázka kódu 11) typu Relay slouží jednoduché „navštívení" stránky server/index.jsp?ajax=setRelay&id=ID&value=HODNOTA. Po kliknutí na element s třídou relay-control se pošle GET požadavek a předají se potřebné parametry souboru index.jsp. V souboru index.jsp poté dojde k nastavení příslušného zařízení na požadovanou hodnotu (viz Ukázka kódu 12) a to tak, že se zažádá o dané zařízení z objektu typu DevicePool, který je získán ze singletonu Daemon a nastaví se mu příslušná hodnota.

Aktualizace zobrazeného zařízení na stránce zaručí událost devicesLoaded, nikoliv událost, která nastane při kliknutí na prvek. Díky tomu je zajištěno asynchronní čekání na odpověď serveru. [10]

```
$(".relay-control").click(function() {
    $("body").css("cursor", "progress");
    setRelay($(this).attr("data-id"), $(this).attr("data-state") == "on" ? "0" : "1");
});
function setRelay(id, value) {
    $.ajax({
        url: "/?ajax=setRelay&id=" + id + "&value=" + value
    });
}
```

Výpis 11: Změna stavu zařízení typu Relay

```
<%
User user = (User)session.getValue("user");
String ajax = request.getParameter("ajax");
if (ajax != null) {
    if (user != null) {
        if (ajax.equals("setRelay") && request.getParameterMap().containsKey("id") && request.
            getParameterMap().containsKey("value")) {
            Relay r = (Relay)Daemon.getDaemon().getDevicePool().get(Integer.parseInt(request.
                getParameter("id")));
            r.setState(Integer.parseInt(request.getParameter("value")) != 0);
            Gson gson = new GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
            out.clear();
            out.print(gson.toJson(r));
            return;
        }
        if (ajax.equals("getDevices")) {
            Gson gson = new GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
            out.clear();
            out.print(gson.toJson(Daemon.getDaemon().getDevicePool().getVirtual()));
            return;
        }
    }
    ...
    else
        return;
}
...
%>
```

Výpis 12: Zpracování AJAX požadavku setRelay a getDevices

Tato neustálá aktualizace všech zobrazených zařízení zajišťuje synchronizaci mezi všemi připojenými klienty. Každý klient webového rozhraní vidí aktuální stav systému, respektive stav zařízení a změna vyvolaná klientem je ihned promítnuta všem připojeným klientům.

U nastavování teploty termostatu, může být na dotykovém zařízení obtížné psát čísla. Proto je použita knihovna jQueryUI a je využit prvek slider – posuvník. Za použití této knihovny lze jednoduše implementovat posuvník (viz Ukázka kódu 13) s nastaveným rozsahem teplot a při změně hodnoty posuvníku, respektive při puštění posuvníku, oznámit nastavenou teplotu. Oznámení je v podstatě odeslání GET požadavku na index.jsp. [11]

Vzhledem k neustálé aktualizaci by mohly nastat komplikace, kdy při chycení posuvníku by posuvník „odskočil" na jinou pozici. Řešením tohoto problému je vyřazení načítání při chycení posuvníku a opětovné zapnutí načítání při puštění, respektive při odeslání nastavené teploty.

```
$(".thermo-slider").slider({ min: -10, max: 40, step: 1,
    change: function(event, ui) {
        $(this).children(".thermo-slider-fill").css("width", $(this).children(".ui-slider-handle").
            css("left"));
        // ajax send set temperature
        var id = $(this).parent().attr("data-id");
        if (!thermoSliderDisabled) {
            loaderDisabled = true;
            $.ajax({
                url: "/?ajax=setHeatingTemperature&id=" + id + "&value=" + ui.value,
                success: function(data) {
                    loaderDisabled = false;
                },
                error: function(x, t, m) {
                    loaderDisabled = false;
                }
            });
        }
    }, slide: function(event, ui) {
        $(this).parent().children(".thermo-input-wrapper").children(".thermo-input").val(ui.value.
            toString() + ""); // degree in brackets
        $(this).children(".thermo-slider-fill").css("width", $(this).children(".ui-slider-handle").
            css("left"));
    }, create: function(event, ui) {
        $(this).slider("value", $(this).attr("data-value"));
        $(this).children(".thermo-slider-fill").css("width", $(this).children(".ui-slider-handle").
            css("left"));
    }
});
```

Výpis 13: Odeslání teploty na server pomocí GET požadavku

U termostatů (obrázek A.13), je také možnost nastavit varovné teploty. Když teplota bude v intervalech nastavených varovných teplot, tak se odešle informativní email o této anomálii. Tyto teploty se nastavují ve formuláři, který se zobrazí po kliknutí na ozubené kolečko v dlaždici termostatu.

```
<% for(Entry<Integer, Blind> b : Daemon.getDaemon().getDevicePool().getBlinds().entrySet()) {
    %>
<div class="subitem blind" data-id="<%=b.getValue().getId() %>">
  <div>
      <div class="blind-icon"></div>
      <div class="blind-slider" data-value="<%=b.getValue().getValue() %>">
          <div class="blind-slider-fill"></div>
      </div>
      <div class="middle-wrapper">
          <p><%=b.getValue().getName() %> (<span><%=b.getValue().getValue() %></span
              >%)</p>
      </div></div></div>
<% } %>
```

Výpis 14: Výpis rolet v souboru blinds.jsp

Ovládání rolet (obrázek A.14) je řešeno podobným způsobem jako u termostatů. Posuvníkem se nastavuje hodnota rolet. Hodnota ve smyslu kolik procent rolet stíní okno. Problém, který může nastat u ovládání rolet je například nastavení hodnoty u rolety, která se teprve přesouvá z jednoho bodu do druhého. Než motor uvede rolety do nastaveného stavu je zablokována změna stavu. O vypsání HTML struktury kódu se stará soubor blinds.jsp (viz Ukázka kódu 14).

```
var blindUpdateDisabled = false;
var blindUpdating = true;
$(".blind-slider").slider({ min: 0, max: 100, step: 5,
    change: function(event, ui) {
        blindUpdateDisabled = true;
        $(this).children(".blind-slider- fill ").css("width", $(this).children(".ui-slider-handle").
            css("left"));
        // ajax send set temperature
        var id = $(this).parent().parent().attr("data-id");
        if (blindUpdating) {
            $.ajax({
                url: "/?ajax=setBlind&id=" + id + "&value=" + ui.value,
                success: function(data) {
                    blindUpdateDisabled = false;
                },
                error: function(x, t, m) {
                    blindUpdateDisabled = false;
     }
            });
        } else {
            blindUpdateDisabled = false;
        }
    }, slide: function(event, ui) {
        blindUpdateDisabled = true;
        blindUpdating = true;
        $(this).parent().children(".middle-wrapper").children("p").children("span").html(ui.value.
            toString());
        $(this).children(".blind-slider- fill ").css("width", (parseInt($(this).children(".ui-slider-
            handle").css("left"))+5).toString());
    }, create: function(event, ui) {
        $(this).slider("value", $(this).attr("data-value"));
        $(this).children(".blind-slider- fill ").css("width", (parseInt($(this).children(".ui-slider-
            handle").css("left"))+5).toString());
});});
$(".blind").bind("devicesLoaded", function(e) {
    if (!blindUpdateDisabled) {
        blindUpdating = false;
        var blind = e.devices[$(this).attr("data-id")];
        $(this).children("div").children(".blind-slider").slider("value", blind.setPercentValue);
        $(this).children("div").children(".middle-wrapper").children("p").children("span").html(
            blind.setPercentValue);
        blindUpdating = true;
    }
});
```

Výpis 15: Realizace posuvníku u rolet

Nastavení hodnoty opět zajišťuje odeslání GET požadavku na stránku server/index.jsp?ajax=setBlind&id=ID&value=HODNOTA. V místě zpracování požadavku se vyhledá z DevicePool objektu zařízení podle identifikátoru a nastaví se mu odeslaná hodnota viz Ukázka kódu 16.

```
...
<%
User user = (User)session.getValue("user");
String ajax = request.getParameter("ajax");
if (ajax != null) {
    if (user != null) {
        if (ajax.equals("setBlind") && request.getParameterMap().containsKey("id") && request.
            getParameterMap().containsKey("value")) {
            Blind b = (Blind)Daemon.getDaemon().getDevicePool().get(Integer.parseInt(request.
                getParameter("id")));
            b.setValue(Integer.parseInt(request.getParameter("value")));
            out.clear();
            return;
        }
        ...
    }
}
...
```

Výpis 16: Ukázka zpracování požadavku na změnu hodnoty rolet

5 Testování

Celková doba testování byla 240 hodin. V testovacím provoze jsem se snažil vyvolávat rušení sběrnic, simulovat napěťové poklesy a generovat nesmylsné požadavky na systém.

Například jsem zkoušel zapínat vysavač blízko sběrnice. Vzhledem k rušení vzniklému při rozběhu elektrického motoru, jsem očekával, že se objeví nějaké chyby. Rušení se projevilo minimálně na sběrnicích a zařízení k nim připojená neprojevila žádné známky chyb.

Napěťové poklesy u napájení fyzických zařízení zapříčinily akorát vypnutí zařízení a řídící jednotka zaznamenala chybu. Systém je navržen tak, aby se z těchto anomálií rychle dostal do stavu v jakém byl před nimi. Takže po ustálení napájecího napětí se systém uvedl do správného chodu. Navíc pokud zvážím, že k výpadkům napájení bude docházet globálně a nemá smysl ovládat zařízení, která nemají zdroj elektrického proudu není řešení záložní baterií na místě.

Po použití jednoho z nejlepších nástrojů testování, malého dítěte, jsem přišel na to, že po neustálém zapínání a vypínaní vypínače se může tento vypínač dostat do neurčitého stavu. U zjišťování (viz Ukázka kódu 17) změny stavu pinu vstupního expandéru se využívá logické operace XOR. Operandy jsou předchozí hodnota vstupního expandéru a nynější přečtená. Díky tomu lze detekovat, na kterém pinu došlo ke změně a oznámit to posluchačům. Pokud testovací nástroj neustále přepíná stavy vypínače, tak se také spouštějí posluchači, kteří poslouchají tuto změnu stavu. Pokud je posluchačem pin výstupního expandéru (elektromagnetické relé) je několikrát po sobě oznámeno, že se změnil stav tlačítka. Díky tomu, že jsou tyto události oznamovány paralelně vzhledem k výstupnímu expandéru dochází k těmto anomáliím. Řešením tohoto problému je zajistit nějakým synchronizačním nástrojem řešení souběhu těchto několika vláken. Konkrétně použít zámek ve funkci setValue, která je ve třídě OutputExpander a zamknout objekt device typu I2CDevice.

```
int oldVal;
while(running) {
    try {
        oldVal = this.value;
        this.value = device.read();
        if (oldVal == this.value)
            continue;
        byte bitwise = (byte)(oldVal ^ this.value);
        for(Entry<Integer, ArrayList<IPinChangedListener>> l : listeners.entrySet()) {
            boolean v = (this.value & (1 << l.getKey())) == 0; // logical 0 = connected
            if ((bitwise & (1 << l.getKey())) != 0) {
                for(IPinChangedListener listener : l.getValue()) {
                    listener.pinChanged(v);
                }
            }
        }
    } ...
}
```

Výpis 17: Implementace čtecího vlákna vstupního expandéru

Sběrnice 1-Wire (sběrnice na které jsou umístěny teplotní čidla) projevila svou chybovost v noci. Měřené teploty náhle stouply o desítky stupňů. Nedokázal jsem přijít na to, proč v noci dochází přibližně jednou za dvě hodiny k anomáliím na sběrnici. Vzhledem k malému výskytu těchto chyb jsou tyto anomálie zanedbatelné. Implementace čtení také zajišťuje, že pokud nastane teplotní skok nebo propad o více jak $8\,^{\circ}C$ je tato teplota ignorována.

Paměťová náročnost programu řídící jednotky se po dobu testování nijak razantně neměnila. Využití procesoru při čtyřech připojených klientech bylo přibližně v polovině maxima. K žádným opožděným reakcím systému nedocházelo, ani po 240 hodinách.

Při využití maximálního potenciálu systému, při připojení 8 vstupních, 8 výstupních expandérů a 32 teplotních čidel, je paměť Raspberry Pi stále dostačující. Celkem je tedy možno použít 64 vstupů a 64 výstupů. Teplotní čidla jsou limitovány možnostmi sběrnice 1-Wire.

Na trhu je řada systémů inteligentího domu. Ceny systému inteligentního domu, který je srovnatelý, s tímto se pohybují přibližně okolo 50000 Kč. V tomto systému je umožněno v každém pokoji ovládat dvě světla, jedno topení a jedny žaluzie. Počet pokojů je omezen na tři, takže je umožněno ovládání šesti světel, třech topení a třech žaluzií. Pro srovnání řekněme, že systém, který by umožňoval ovládat stejný nebo větší počet zařízení, musí být složen ze dvou výstupních expandérů, jednoho vstupního, základny, tabletu a řídící jednotky. Náklady jsou u výstupního expandéru 1000 Kč, vstupního expandéru 500 Kč, základny 250 Kč, tabletu 2000 Kč a u Raspberry Pi 800 Kč. Také je potřeba přičíst náklady za napájecí zdroje a krabičky. Součet těchto položek vychází přibližně na 7550 Kč. Řekněme, že marže na tomto produktu bude 200% z celkových nákladů. Celkový systém by na trhu stál 22650 Kč[5] a to je polovina z ceny, za kterou je tento systém nabízen.

[5] Všechny ceny jsou uvedeny bez DPH a nejsou započteny náklady na instalaci.

Závěr

Cílem mé bakalářské práce bylo navrhnout a vytvořit levný a modulární systém inteligentního domu. Toho jsem také docílil a vzhledem k modularitě systému budu ve vývoji pokračovat.

Průběh tvorby práce probíhal celkem skepticky. Vzhledem k tomu, že jsem se opravdu snažil docílit největší kvality, nejmenší ceny a především modularity systému jsem vývoj asi třikrát zastavil a začal od začátku.

Začínal jsem s návrhem řídící jednotky. První podoba byla napsána v programovacím jazyce C# a spouštěna pomocí mono. Tento jazyk nebyl příliš výhodný vzhledem k operačnímu systému Raspbian, a tak jsem začal od začátku. Tentokrát jsem začal vyvíjet v Javě. Po vyvinutí alpha verze programu pro řídící jednotku jsem začal s návrhem fyzických zařízení.

První byl výstupní expandér. U vývoje výstupního expandéru jsem vytvořil asi šest verzí celého expandéru. Za tu dobu jsem se naučil skvěle využívat program EAGLE. První verze výstupního expandéru obsahovala velké množství chyb z hlediska elektroniky. Například, díky tomu, že jsem si těchto chyb nevšiml, se odpálilo všech osm signalizačních LED diod a cestou vzal proud i NPN tranzistory. Nevšiml jsem si, že jsou některé cesty na desce spojeny a to vedlo k vyřazení bázových rezistorů a rezistorů připojených k LED diodě. Druhá verze výstupního expandéru už „neodpalovala" diody a tranzistory, ale její návrh byl nešikovně zpracován a byl jsem donucen použít asi tři propoje. To by nebyl až takový problém, ale tento návrh skončil u testování. Zjistil jsem totiž, že obvodům PCF8574 chviličku trvá, než se resetují a během této doby jsou na výstupní bráně jedničky. Vzhledem k tomu, že jsem použil NPN tranzistory pro ovládání relé se po zapnutí systému všechna relátka sepla na dobu resetu a to vedlo k sepnutí všech zařízení v domě po restartu systému. Tuto krátkodobou špičku by nevydržely, jak kontakty relé, tak jističe. Takto vývoj pokračoval až do šesté verze výstupního expandéru.

Z těchto chyb jsem se poučil a výsledný vstupní expandér byl po třech pokusech hotov. V první verzi chyběly ochranné diody.

Po dobu vytváření vstupního a výstupního expandéru jsem nijak nepřemýšlel nad délkou vedení sběrnic I2C a 1-Wire. Po dalším zkušebním provozu jsem tedy zjistil, že by to s 3.3V úrovněmi nešlo a to vedlo k vytvoření základny, která měla za úkol upravit výkon sběrnic a umožnit jednoduchou instalaci kabeláže. Vytvoření základny vedlo tedy k předělání vstupního a výstupního expandéru. Musel jsem přidat do těchto zařízení napěťové převodníky sběrnic. Nakonec zbývalo vytvoření poslední jednoduché desky a to napěťového převodníku pro DS18B20 teploměr.

Po vytvoření těchto zařízení a naprogramování aplikace pro řídící jednotku zbývalo vytvoření webového rozhraní. Snažil jsem se držet pořekadla „V jednoduchosti je síla" a na základě toho vytvořit přehledný, na pohled jednoduchý a uživatelsky přívětivý grafický návrh webového rozhraní. Myslím, že jsem toho docílil. Po testování webového rozhraní jsem byl mile překvapen, jak i lidé, kteří nepracují s počítačem dokázali ovládat systém.

Luboš Matejčík

Literatura

[1] VALEŠ, Miroslav. *Inteligentní dům.* 1. vyd. Brno: ERA, 2006, 123 s., il. (část barev.). ISBN 80-736-6062-8

[2] OLIVKA, Petr a David SEIDL. *Návody do cvičení pro předmět Architektury počítačů a paralelních systémů.* [online]. [cit. 2014-04-28]. Dostupné z: `http://poli.cs.vsb.cz/edu/apps/lab/apps-cvic.pdf`

[3] BRANDŠTETTER, Pavel. *Elektronika.* Ostrava: Vysoká škola báňská - Technická univerzita, 2007, 1 CD-R. ISBN 978-80-248-1481-0.

[4] MATOUŠEK, David. *Udělejte si z PC - generátor, čítač, převodník, programátor..: Měření, řízení a regulace pomocí sériového portu PC a sběrnice I2C.* 1. vyd. Praha: BEN - technická literatura, 2001, 175 s. ISBN 80-730-0036-9.

[5] Datové listy k obvodu PCF8574. *PCF8574: Remote 8-bit I/O expander for I2C-bus* [online]. Nizozemsko: Koninklijke Philips Electronics N.V., 2002. [cit. 2014-04-28]. Dostupné z: `http://www.nxp.com/documents/data_sheet/PCF8574.pdf`

[6] Datové listy k GPIO Raspberry Pi. *BCM2835 ARM Peripherals* [online]. Europe: Broadcom Corporation, 2012. [cit. 2014-04-28]. Dostupné z: `http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf`

[7] Datové listy k obvodu DS18B20. *DS18B20 Programmable Resolution 1-Wire Digital Thermometer* [online]. USA: Maxim Integrated, 2008. [cit. 2014-04-28]. Dostupné z: `http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf`

[8] *Manuálové soubory systému Linux.*

[9] W3C Candidate Recommendation. *HTML5: A vocabulary and associated APIs for HTML and XHTML* [online]. 2013 [cit. 2014-04-28]. Dostupné z: `http://www.w3.org/TR/2014/CR-html5-20140204/single-page.html`

[10] Dokumentace knihovny jQuery. *JQuery API Documentation* [online]. [cit. 2014-04-28]. Dostupné z: `http://api.jquery.com/`

[11] Dokumentace knihovny jQuery UI. *JQuery UI API Documentation* [online]. [cit. 2014-04-28]. Dostupné z: `https://api.jqueryui.com/`

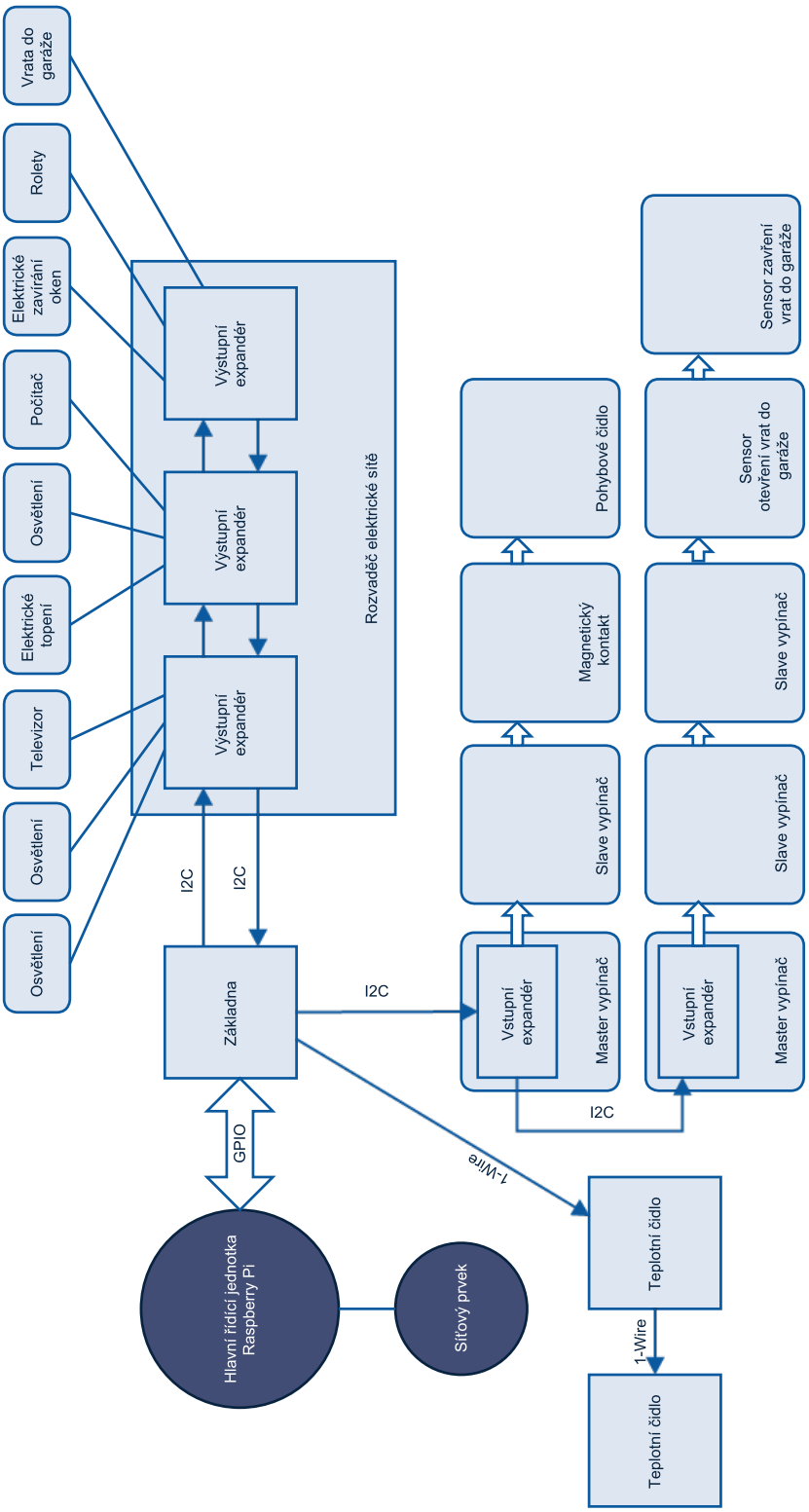
[12] Java SE 7 dokumentace. *Overview (Java Platform SE 7)* [online]. 1993 [cit. 2014-04-28]. Dostupné z: `http://docs.oracle.com/javase/7/docs/api/`

[13] Dokumentace Jetty. *Jetty - Documentation* [online]. [cit. 2014-04-28]. Dostupné z: `http://www.eclipse.org/jetty/documentation/`

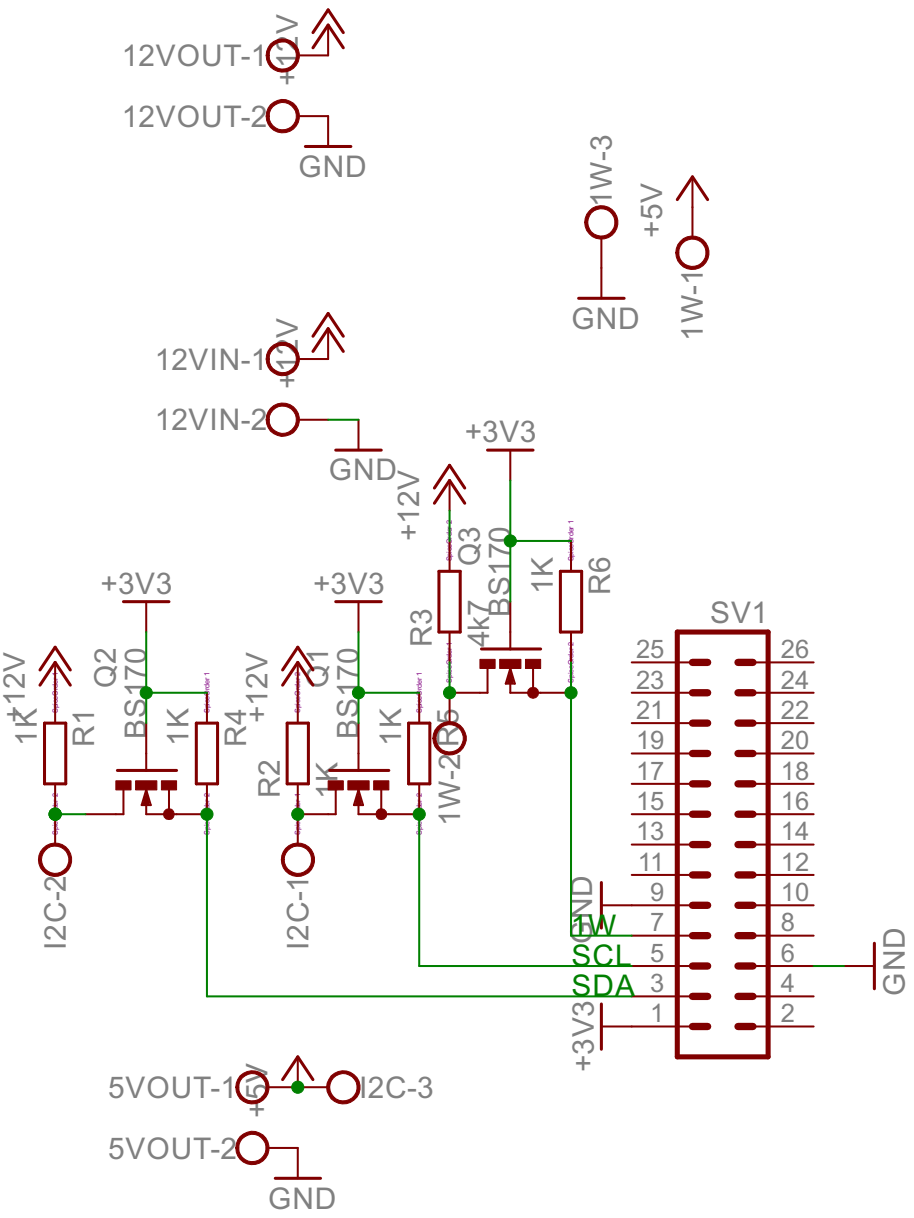
[14] Circuit Simulator. *Circuit Simulator Applet* [online]. [cit. 2014-04-28]. Dostupné z: `http://falstad.com/circuit/`

A Schémata, diagramy a fotografie

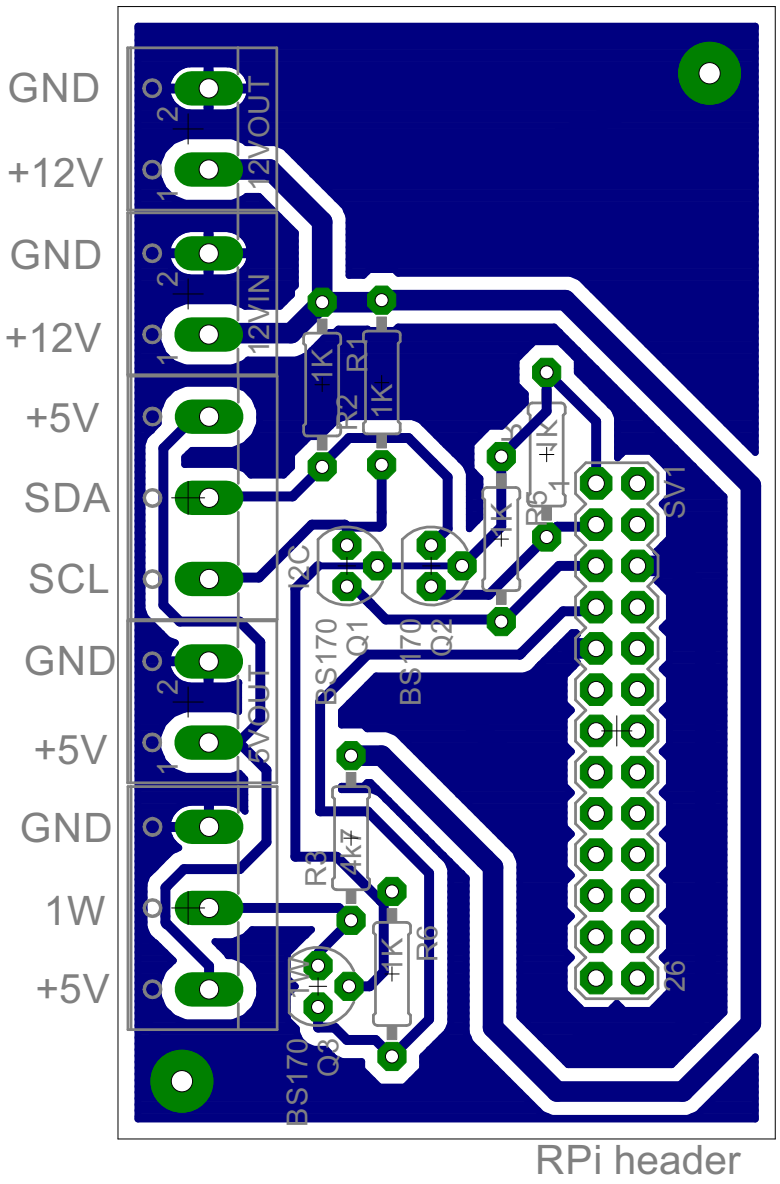
V této sekci naleznete veškerá schémata, diagramy zapojení, fotografie hotových výrobků a obrázky webového rozhraní.



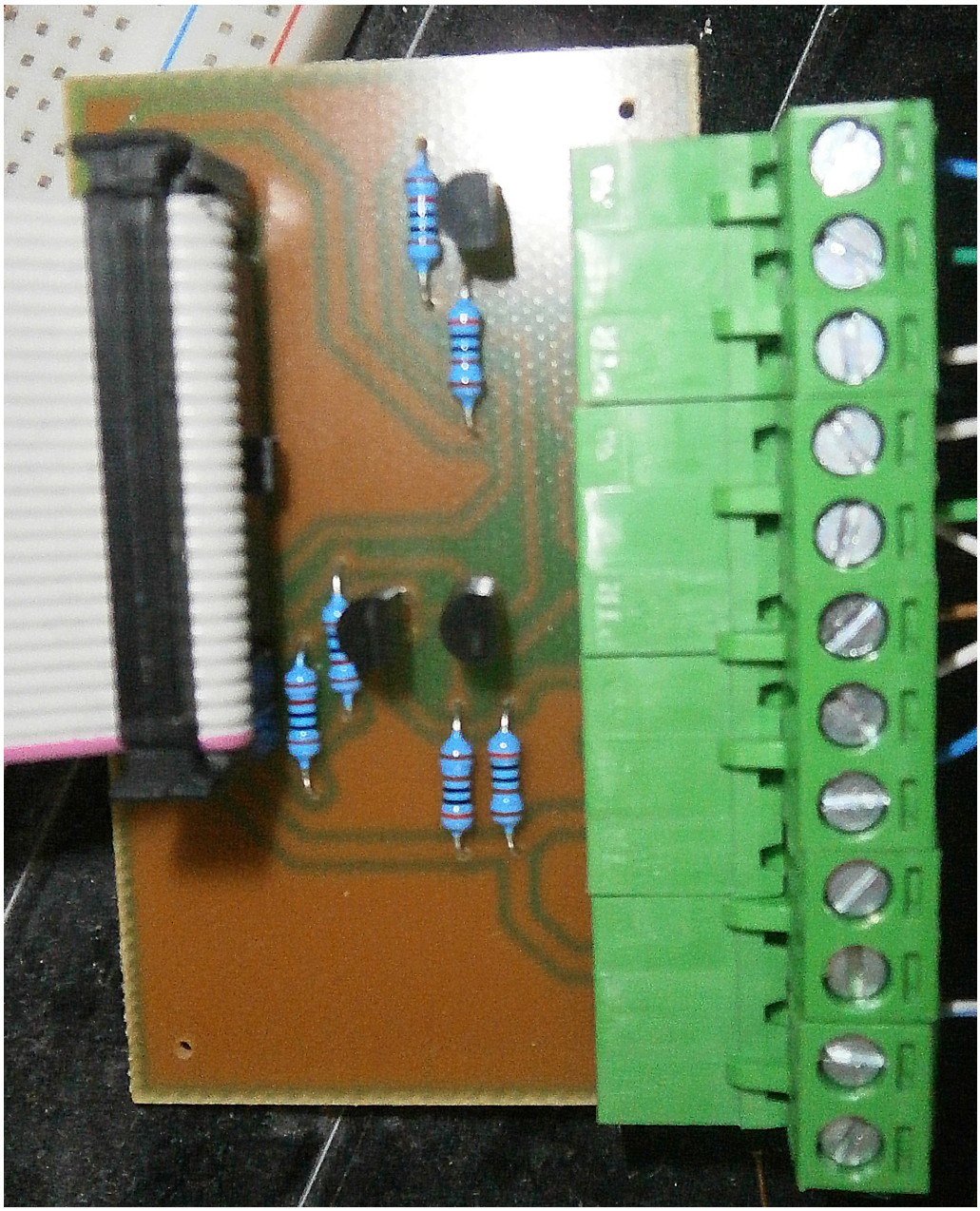
Obrázek A.1: Diagram zapojení fyzických zařízen



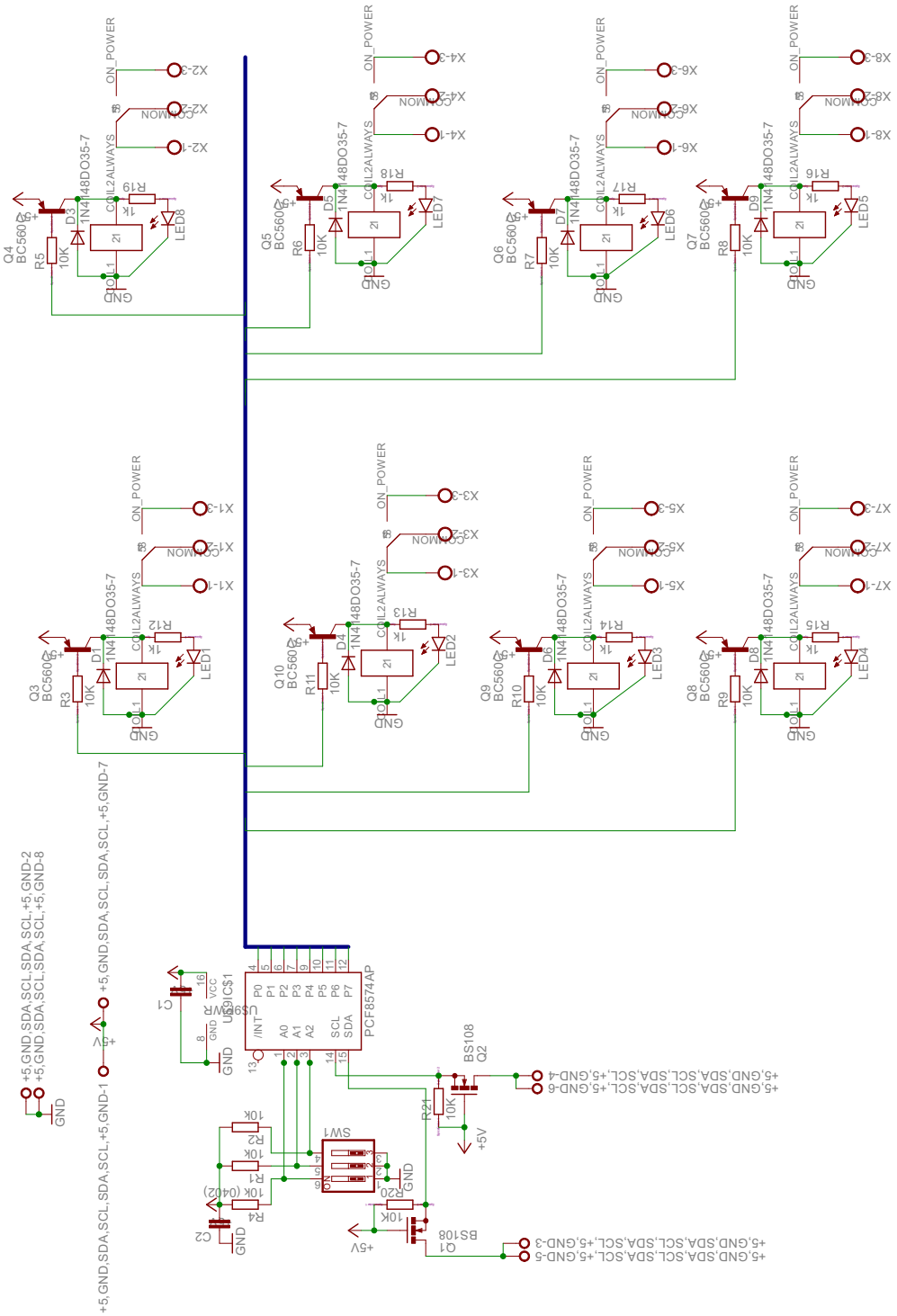
Obrázek A.2: Schéma zapojení součástek základny



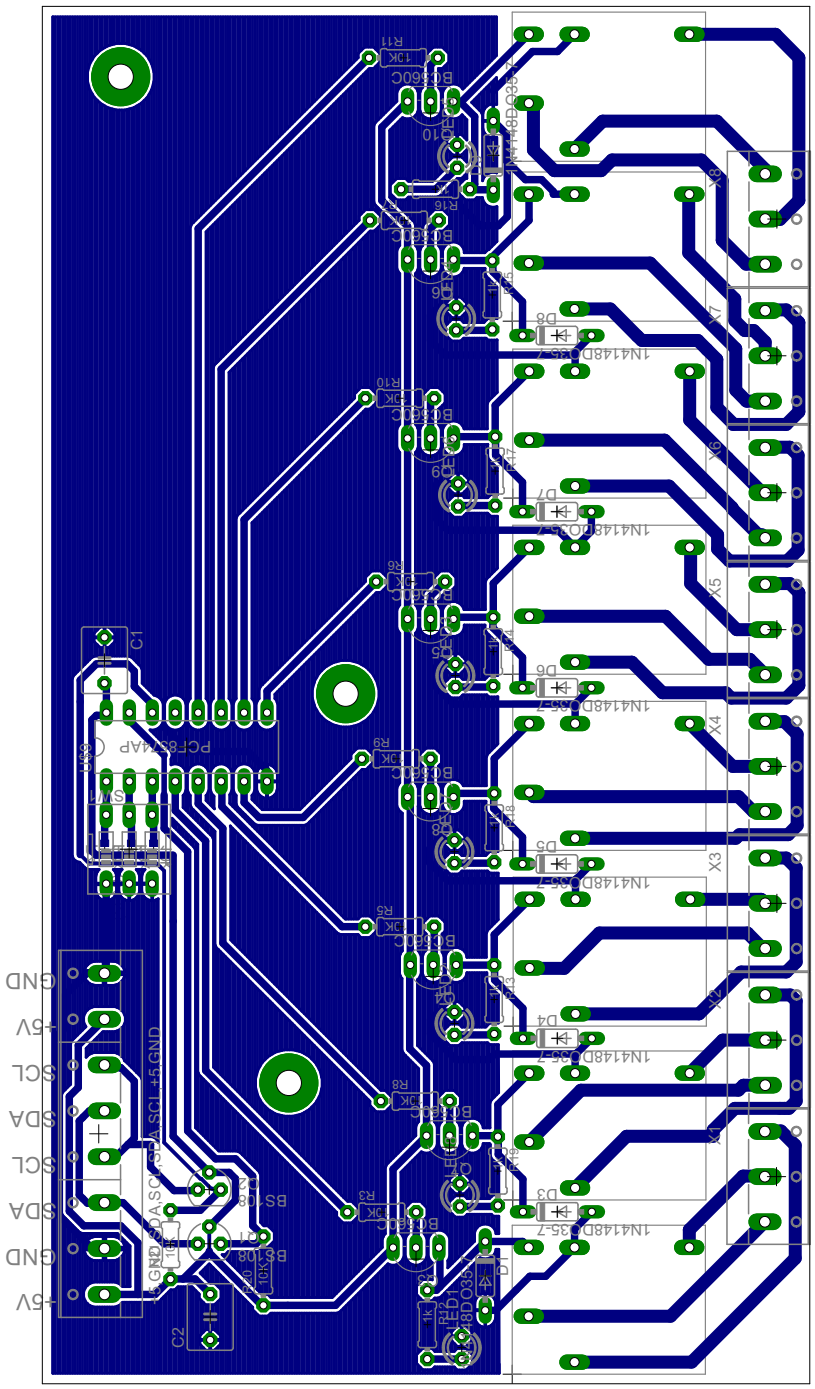
Obrázek A.3: Deska plošných spojů základny



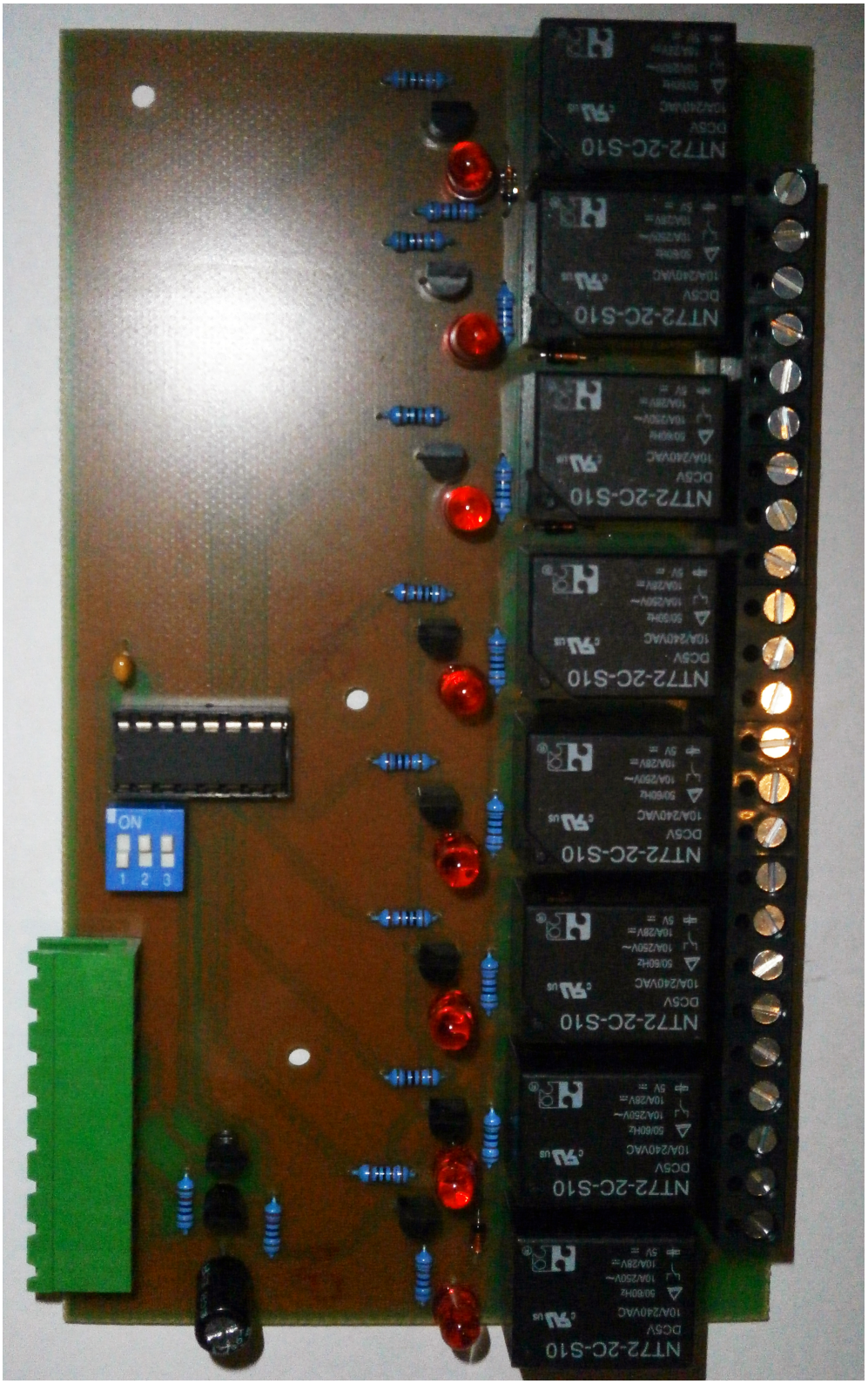
Obrázek A.4: Osazená deska plošných spojů základny



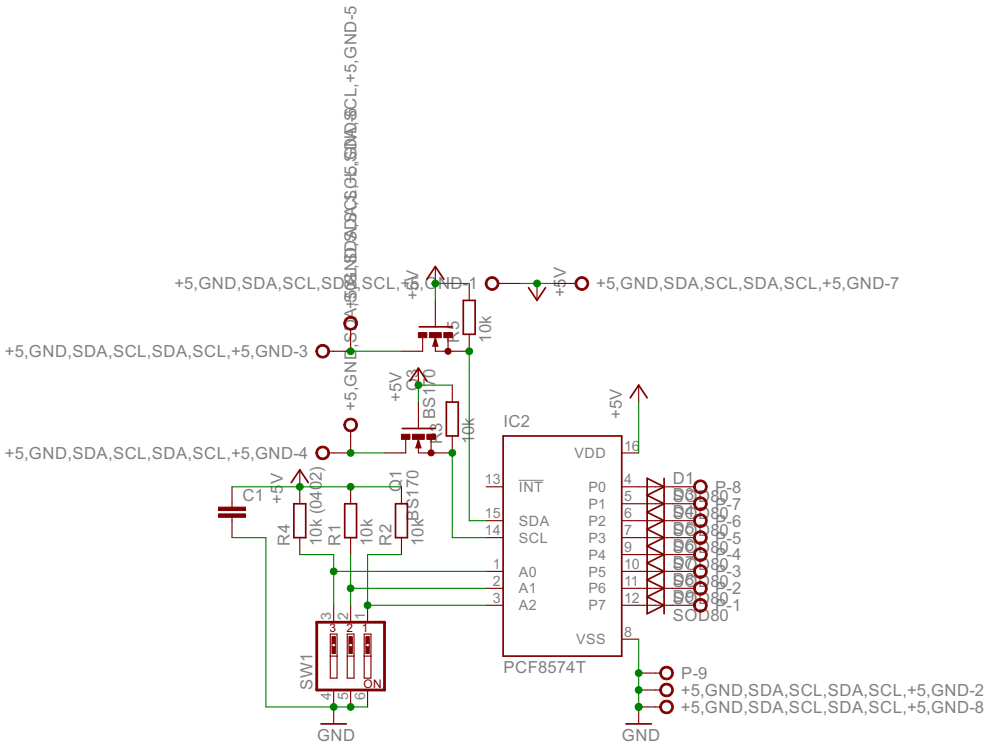
Obrázek A.5: Schéma zapojení součástek výstupního expandéru



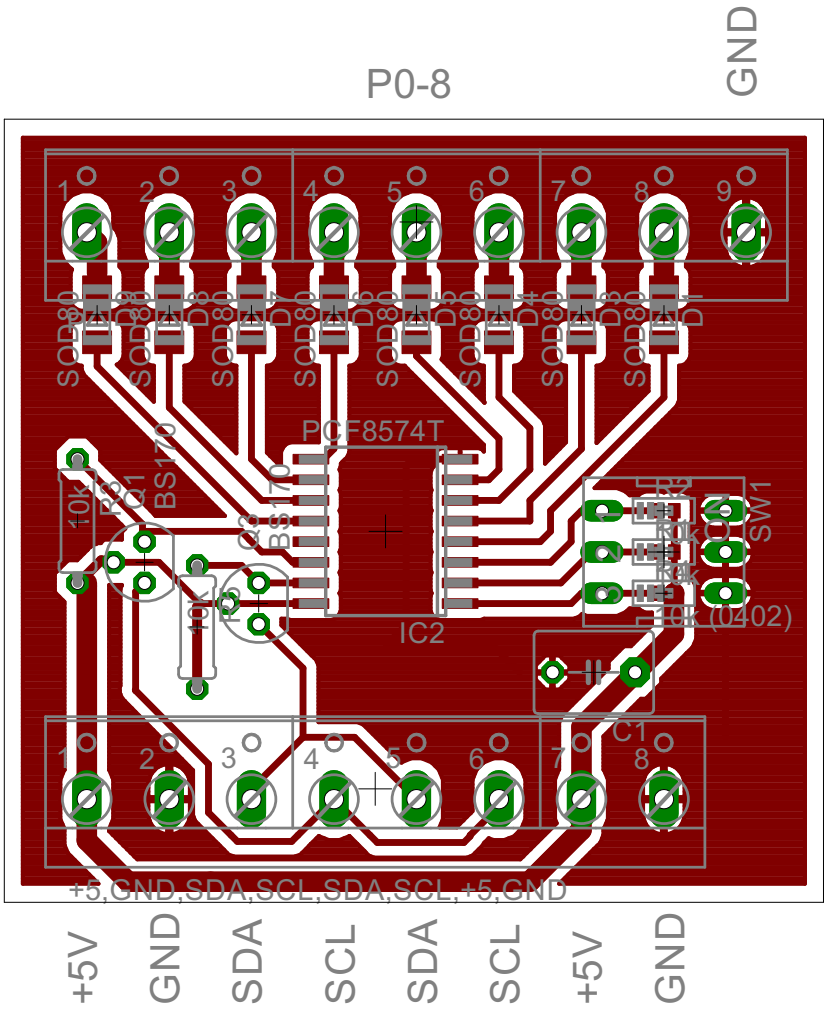
Obrázek A.6: Deska plošných spojů výstupního expandéru



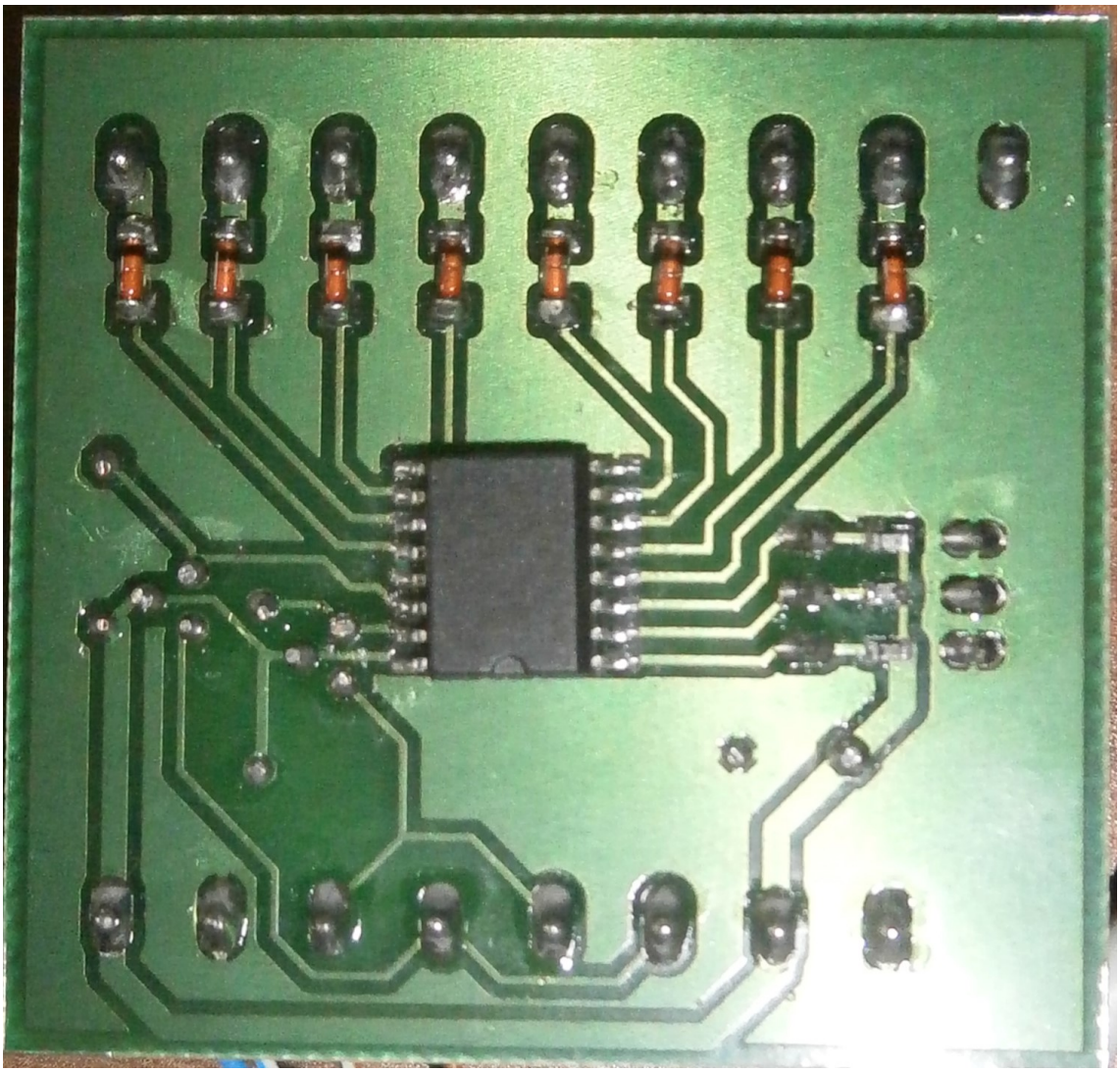
Obrázek A.7: Osazený výstupní expandér



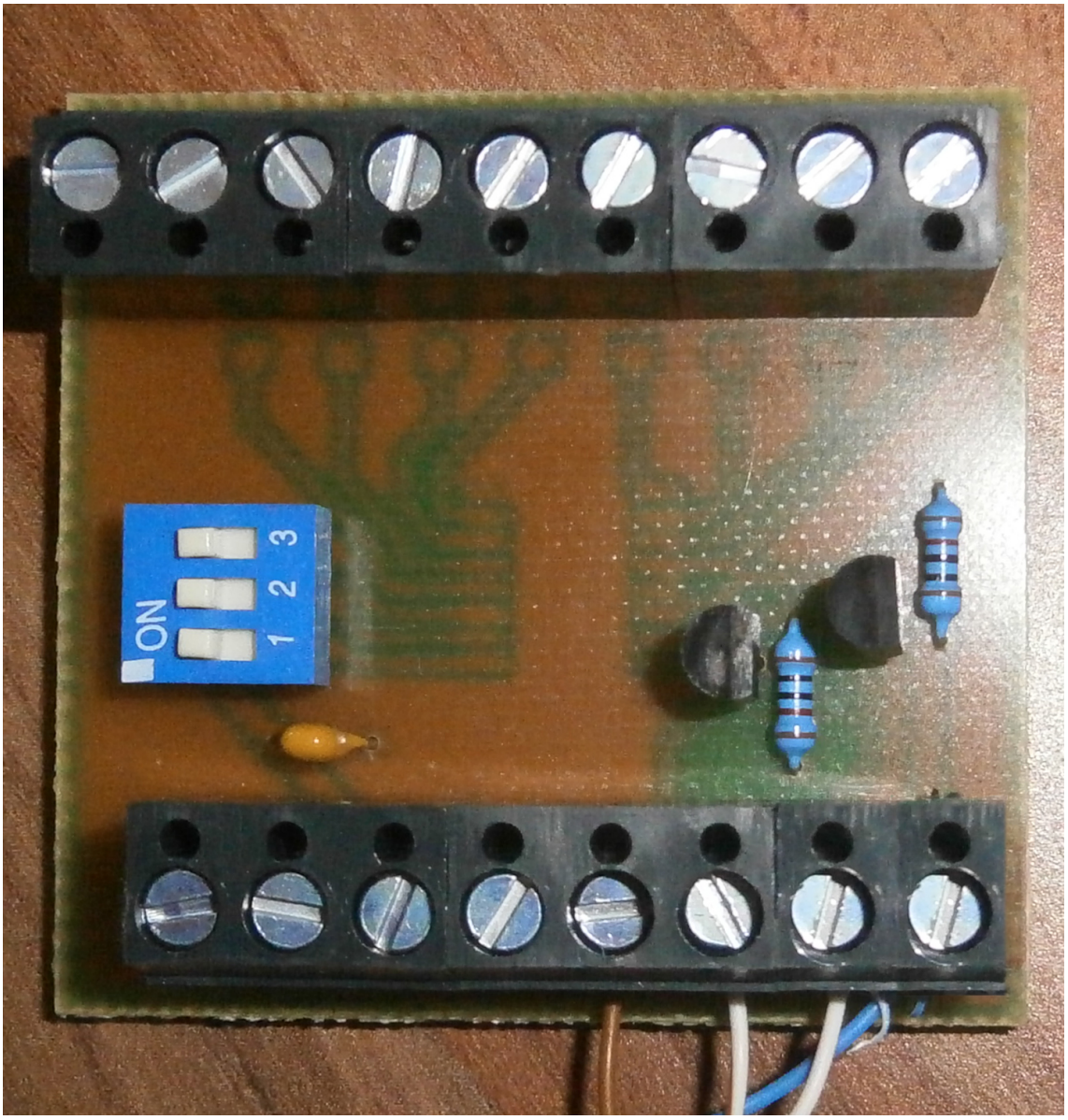
Obrázek A.8: Schéma zapojení součástek vstupního expandéru



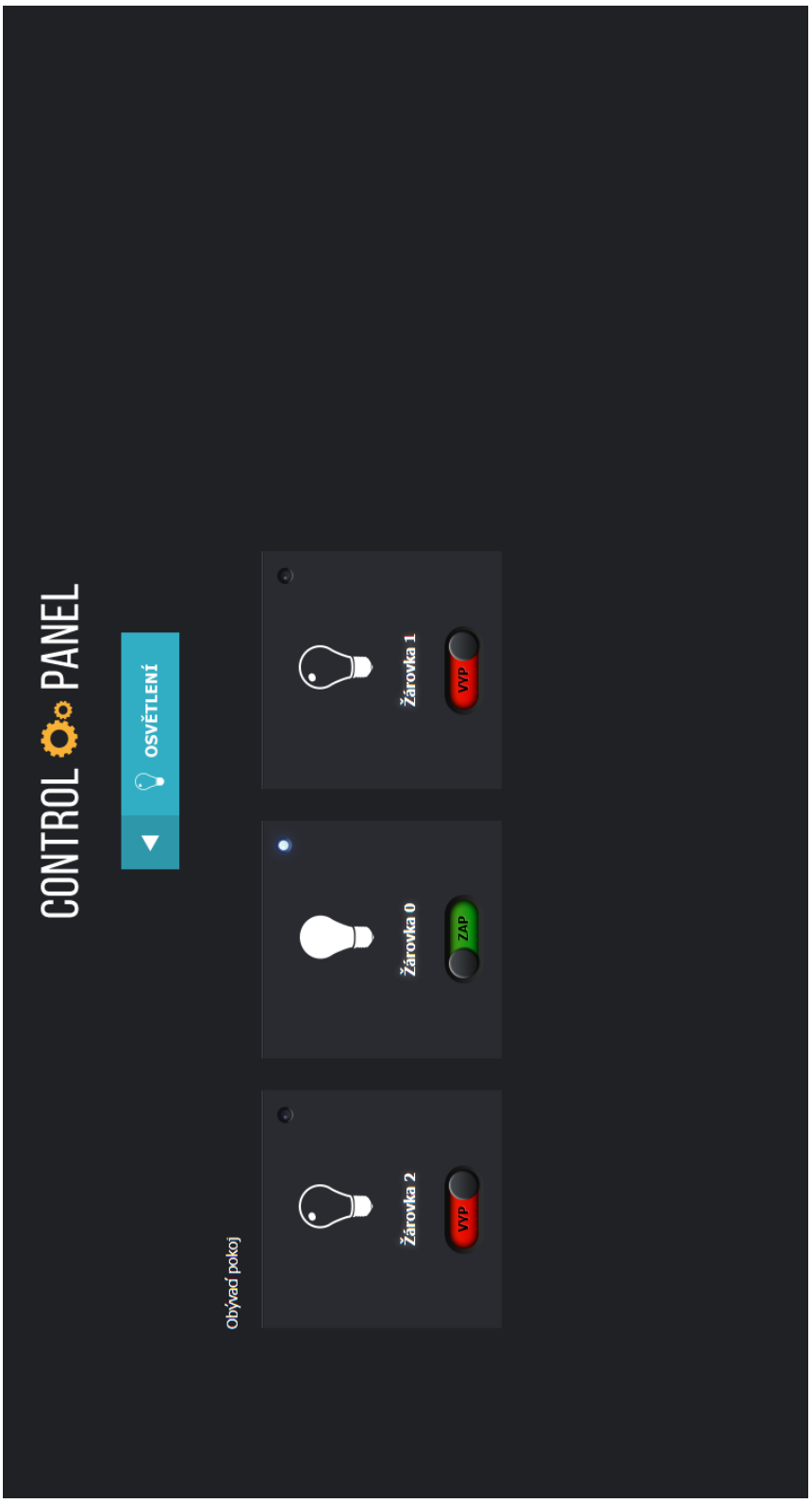
Obrázek A.9: Deska plošných spojů vstupního expandéru



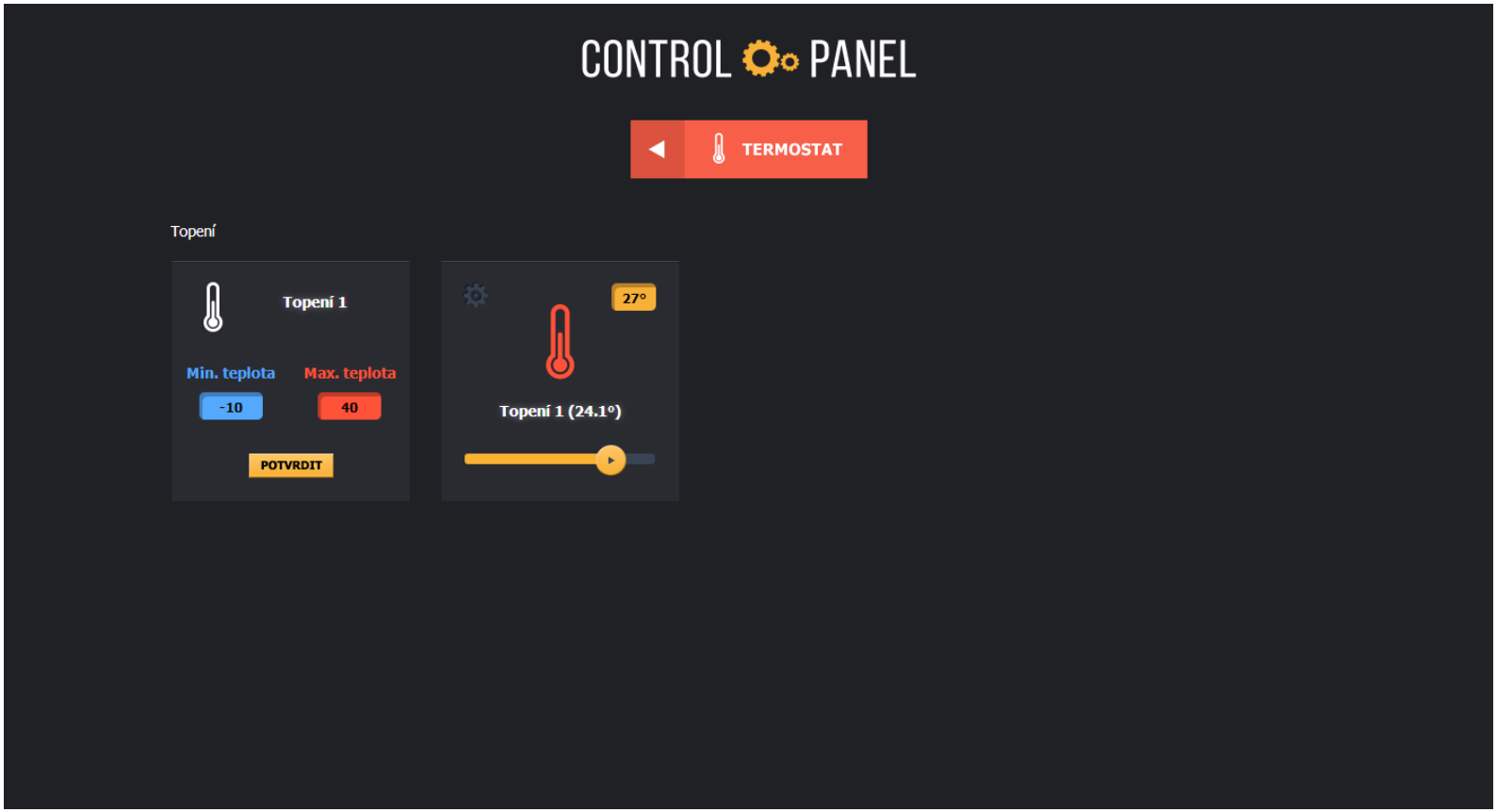
Obrázek A.10: Osazený vstupní expandér zespod



Obrázek A.11: Osazený vstupní expandér zvrchu



Obrázek A.12: Pohled na stránku server/lights



Obrázek A.13: Pohled na stránku server/thermo

Obrázek A.14: Pohled na stránku server/blinds