BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF MICROELECTRONICS

ÚSTAV MIKROELEKTRONIKY

HOME AUTOMATIZATION SYSTEM BASED ON RASPBERRY PI

INTELIGENTNÍ DOMÁCNOST S VYUŽITÍM RASPBERRY PI

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Lukáš Lokajíček**
AUTOR PRÁCE

SUPERVISOR — **Ing. Michal Pavlík, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Lukáš Lokajíček ***ID:*** 154792
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Inteligentní domácnost s využitím Raspberry Pi

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte modulární systém inteligentní domácnosti založený na platformě Raspberry Pi. Součástí návrhu bude využití spolehlivé komunikační sběrnice, senzory a aktuátory, potřebné k realizaci inteligentní domácnosti. K ovládání, zobrazení stavu a nastavení inteligentní domácnosti vytvořte serverovou aplikaci využívající webové rozhraní. To bude realizováno s ohledem na kompatibilitu s mobilními zařízeními. Pro ověření funkčnosti návrhu realizujte prototyp zařízení.

DOPORUČENÁ LITERATURA:

According to recommendations of supervisor

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 25.5.2017

Vedoucí práce: Ing. Michal Pavlík, Ph.D.
Konzultant:

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

Abstrakt:

Předkládaná diplomová práce se zabývá návrhem systému pro inteligentní domácnost, který využívá výhod jednodeskového počítače Raspberry Pi. V teoretické části je provedena rešerše v oblasti inteligentních domů, která odhaluje slabé stránky v této oblasti. Práce si dává za cíl tyto slabiny odstranit a zohledňovat spolehlivost, rozšiřitelnost a nízké pořizovací náklady. Praktická část je uvedena návrhem jednotlivých modulů, který zahrnuje jak design hardwaru, tak i programové vybavení. Projekt je uzavřen spojením všech komponentů do jednoho funkčního univerzálního systému spolu s uvedením rozšiřitelnosti.

Abstract:

The master's thesis deals with the design of the Smart Home System (SHS), which takes advantage of the 'Raspberry Pi' single-board computer. Background research about the theoretical concept of SHS is carried out, which reveals weaknesses in that field. The aim of the thesis is elimination these weak points and takes into account reliability, extensibility and low acquisition price. The practical part is introduced by design of particular modules, which include both hardware design and software. The project is concluded with integration all components into the single functional universal system together with the extensibility presentation.

Klíčová slova:

Inteligentní domácnost, Raspberry Pi, RS-485, UART, TCP, openHAB.

Key words:

Smart Home System, Raspberry Pi, RS-485, UART, TCP, openHAB.

Bibliografická citace mé práce:

LOKAJÍČEK, L. *Inteligentní domácnost s využitím Raspberry Pi.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 72 s. Vedoucí diplomové práce Ing. Michal Pavlík, Ph.D..

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma **„Inteligentní domácnost s využitím Raspberry Pi“** jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 1. května 2017 .....
podpis autora

Poděkování

Kde jinde, než v této diplomové práci, uzavírající moji kariéru studenta, je místo pro poděkování zejména rodičům, kteří mne podporovali všemi možnými cestami po celou dobu studia, bez této pomoci bych jen obtížně ukončil studium, proto vám patří ten největší dík. Systém inteligentní domácnosti byl implementován do bytu mého kamaráda Ing. Marka Jakeše, kterému bych chtěl tímto poděkovat za projevení důvěry, trpělivosti a umožnění realizace v jeho vlastním bytu za finanční podpory. Během praktické realizace byla potřeba schopného člověka, kterým byl nejlepší kamarád Bc. Zdeněk Ptáček, jemuž děkuji za pomoc a věcné podněty k tomuto projektu. Rovněž stylistická korektura textu diplomové práce byla v jeho rukou. Vedoucímu diplomové práce Ing. Michalu Pavlíkovi, PhD. děkuji za cenné rady k formální částí této práce.

V Brně dne 1. května 2017 .....
podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czech Republic

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

Obsah

Úvod

V dnešní době se klade velký důraz na ekologii, úsporu energie a v neposlední řadě i na komfort. Kdo by nechtěl po náročném dni přijít z práce a užít si právě nabízenou pohodu svého domu. Pro většinu lidí je právě domov místem, kde si mohou odpočinout. Nabízí se tedy myšlenka upravení domu tak, aby nám co nejvíce zpříjemnil relaxování a zároveň uspokojil myšlenku úspory energie a tím ulehčil i přírodě. Nazvěme tuto myšlenku inteligentní domácností.

Pro řízení komplexnějších systémů je vhodné použít řídící prvek s vyšším výpočetním výkonem. Nejen proto, abychom minimalizovali dobu odezvy na jednotlivé podněty, ale také abychom měli vytvořenou rezervu výpočetního výkonu pro možnost přidávání dalších prvků do systému, které by později zatěžovaly vyhodnocovací jednotku. Zároveň by bylo vhodné vyhodnocovací jednotku jednoduše integrovat do systému. Nabízí se tedy řešení pomocí jednodeskového počítače Raspberry Pi (v této práci označován jako RPi). Výhodou tohoto počítače je přítomnost GPIO (General Purpose Input/Output), což umožňuje právě zmíněnou integraci. Mezi další výhody je dobrá dostupnost, kvalitní dokumentace, výborný poměr cena/výkon atd.

Jedním ze základních požadavků na systém inteligentní domácnosti (dále *SID*) je spolehlivost, na níž má signifikantní vliv komunikace. Průmyslové sběrnice se zejména vyznačují spolehlivostí vůči rušení. Proto byla inspirace čerpána z toho oboru a v ohledu na další požadavky, jakou jsou komunikace na dlouhou vzdálenost a připojení vícero zařízení, byl vybrán standard sériové komunikace RS-485.

Aktuátory a senzory jsou nezbytnou součástí systému inteligentní domácnosti. Dalo by se říct, že čím více těchto prvků systém obsahuje, tím více může být domácnost automatizována a vyznačovat se prvky inteligence. V reálné situaci inteligentních domácností je potřeba v každém projektu specifikovat, jaké prvky domácnosti mají být automatizovány. Podle těchto požadavků jsou voleny aktuátory a senzory. Systém by tedy měl nabízet širokou paletu aktuátorů a senzorů, z nichž by se jednotlivé projekty inteligentních domácností sestavovaly. Samozřejmě není možné mít natolik obsáhlou paletu možností, aby z ní bylo možné sestavovat jakékoliv projekty. Podíváme-li se na firmy, které se pohybují v oblasti inteligentních domácností na našem trhu, toto tvrzení se potvrdí. Potřeby zákazníků jsou velmi heterogenní, a proto se někdy stane, že je zákazník nucen vybavit svůj byt dvěma systémy, což je velmi nepraktické i v ohledu ovládání těchto dvou konkurenčních systémů. Proto jsou na nově vznikající systém kladeny nároky na možnost rozšíření a to ve smyslu rozšíření po instalování systému, tak i rozšíření nabízené palety o nové prvky již během projektování. Zavedením programu openHAB do SID bylo umožněno integrovat různé technologie do jednoho systému s jednotným ovládáním.

Inteligentní domácnost musí brát také ohledy na energetickou nenáročnost a úsporu energií. Energetickou nenáročnost systému lze ovlivnit zejména designem elektronických obvodů, napájecím napětím a také programem, který může přepínat prvky do úsporných režimů. Zvolený počítač Raspberry Pi také napomáhá uspořit energii systému, jelikož disponuje malým odebíraným výkonem, který se mění úměrně k využití CPU (Central processing unit – centrální procesorová jednotka). Jinými slovy, dobře navržený ovládací program přispívá ke zvýšení energetické nenáročnosti. Úspora energie musí být zajištěna inteligentní domácností. Největší úsporu pocítíme na osvětlení a vytápění. Již LED osvětlení pomáhá šetřit energii, a když přidáme regulaci jasu, úsporu znatelně pocítíme. Regulace topení a vypínání topení v době nepřítomnosti osob v domácnosti či při otevřeném oknu, je dalším klíčem k úspoře energie na topení.

1 Inteligentní domácnost obecně

V dnešní době slyšíme z každé strany o inteligentní domácnosti, jaké poskytuje výhody, komfort, šetření energií, zabezpečení a mnohé další užitečné vlastnosti. Ovšem co to vůbec je inteligentní domácnost? Je opravdu inteligentní? Jaké jsou její výhody a nevýhody? Kolik stojí pořízení takové domácnosti? Veškeré zmíněné otázky, a nejen ty, budou zodpovězeny v této kapitole.

1.1 Rysy inteligentní domácnosti

Inteligentní domácnost je schopna zajistit komfortní podmínky pro obyvatele, postarat se o minimalizaci energetických výdajů a zajistit bezpečí. Jedná se o relativně nový pojem, který není jednoznačně definován, proto jsou zde uvedeny hlavní rysy a ne definice. Domácnost nemusí splňovat veškeré zmíněné vlastnosti, aby ji bylo možné nazývat inteligentní. Obecně se dá říct, že stačí, aby splňovala alespoň jednu jmenovanou vlastnost, která dům automatizuje. O bližších informacích pojednává podkapitola 1.2 Management, kde jsou vyjmenovány nejčastější oblasti automatizace domácnosti.

1.1.1 Název „Inteligentní domácnost“

Narážíme na dva zmíněné pojmy: *automatizace a inteligence*, zajisté není možné mezi tyto pojmy vepsat rovnítko, přesto při mluvení o inteligentní domácnosti tomu tak je. V anglické literatuře je jedno z používaných slovních spojení „*Home automation“* neboli automatizace domácnosti, právě tento název je nejpřesnější. Obecná definice inteligence je umění se učit. Kdybychom tedy chtěli nazvat domácnost inteligentní, bylo by potřeba, aby byla schopna se učit od uživatelů, což většinou takovou schopnost nemá. Dalším anglickým názvem je „*Smart home“,* ze které nejspíše vzniklo označení „Inteligentní domácnost“. Závěrem tedy řekněme, že ač nazýváme automatizované domácnosti inteligentními, inteligentní jako takové rozhodně nejsou a budeme to chápat pouze jako název.

Když už jsme si konečně probrali první slovo, podívejme se na slov druhé a to „domácnost“. Někdy se použije slovo domácnost a jindy dům. Pokud je slovo správně použito, tak mezi těmito realizacemi je velký rozdíl. Budeme-li mluvit o domácnosti, tak se jedná čistě o automatizované domácnosti – řešení osvětlení, tepelný management, media, zabezpečení atd.. Mluvíme-li o inteligentním domu, zahrnujeme i pokročilé technologie stavební. Rozšiřujeme tedy projekt inteligentní domácnost o inteligentní stavbu, například pasivní domy a jiné pokročilejší stavební technologie.

1.2 Management

Management neboli řízení inteligentní domácnosti. Nejlepší systém by byl takový, který by se postaral o veškeré potřeby uživatele. V této podkapitole si uvedeme nejčastější prvky celkového managementu, které dohromady tvoří právě inteligentní dům. Když zahrneme veškeré prvky do jedné výkonné řídící jednotky, bude mít přehled o všech dějích v domácnosti a může tak reagovat na současné podněty a zároveň na podněty, které se již staly, ale pořád mají dopad na způsob řízení. Uveďme si příklad z tepleného managementu a nepřímý zásah z managementu bezpečnosti: Máme v místnosti nastavenou teplotu na 25 °C, ale teplota klesne na 23 °C. Budeme-li uvažovat pouze tepelný management, tak se nestane nic jiného, než se zapne vytápění, aby teplota opět vzrostla na požadovaných 25 °C. Využijeme-li komplexní management, použijeme data z bezpečnostního managementu a zjistíme, že v dané místnosti je otevřené okno. Systém tedy nebude zbytečně zapínat topení, aby se neplýtvalo energií, ale

upozorní na otevřené okno a vyčká do doby, než se okno zavře. Po zavření okna (nezvolí-li uživatel jinak) se aktivuje topení do doby, než bude v místnosti naše požadovaná teplota. Díky správnému řízení nám systém ušetří peníze a postará se o udržení komfortní teploty v domácnosti. K příkladu může být vznesena námitka na to, že hlídání stavu oken může být zahrnuto v tepelném managementu. Námitka je zcela oprávněna, v případě, kdy systém neobsahuje zabezpečovací management, a hlídání stavu oken je zahrnuto v tepleném řízení. Zde se ale nabízí uvést, že hlídání stavu oken již může být minimalistické řízení bezpečnosti, ale to bude záležet na úhlu pohledu.

1.2.1 Světelný management

Řadí se mezi dvě nejpoužívanější řízení, což ani nevzbuzuje údiv, když je světlo potřebné v každé domácnosti. S automatickým osvětlením se setkáváme již řadu let, a přesto o dané domácnosti nemluvíme jako o inteligentní. Jsou to právě (většinou pyroelektrické) senzory pohybu, které při vyhodnocení pohybu sepnout relé přímo u světla a po nějakou dobu jej nechá rozsvícené. Pokud nezaznamená další pohyb, tak jej zhasne. Takové řízení je vhodné pouze do prostor, které jsou určeny pouze pro průchod (například vstupní dveře), nebo prostory, které nejsou dlouho obývány (jako jsou kupříkladu toalety). Zcela nevhodné jsou tyto senzory pro místnosti, kde se moc pohybu nevykoná. Typickým příkladem může být pracovna, kde při sezení u počítače senzor nevyhodnotí žádný pohyb a světlo by po chvíli zhaslo, což je nežádoucí.

Jedním z řešení zmíněné problematiky může být takzvané počítání osob v místnostech. Světlo by se tedy rozsvítilo s příchodem první osoby a zhaslo při odchodu poslední osoby z místnosti. Nabízí se otázka, kdo bude počítat lidi v dané místnosti. Určitě nebude vhodné dát tlačítka ke dveřím na obě strany a při průchodu stisknout tlačítko, aby systém věděl, kolik v místnosti zbývá ještě osob. Taková tlačítka nahradíme optickou závorou, která bude na každé straně dveřních prostor a podle zaznamenání pohybu a pořadí aktivace závor algoritmus vyhodnotí, zda osoba do místnosti vešla či z ní odešla. Dalším zjišťování počtu osob, a to mnohem složitějším, může být pomocí kamer, které by sledovaly objekty. Velkou výhodou by bylo už jen to, kdyby kamera byla schopna rozlišit, i do které místnosti osoba míří, a dopředu by v místnosti rozsvítila.

Doposud bylo zmíněno dvouúrovňové ovládání světla, tedy svítí či nesvítí. Většině svítidel lze plynule měnit intenzitu svitu a to různými způsoby, které se liší v závislosti na typu svítidla. Těmto svítidlům se říká stmívatelná, v angličtině *Dimmable light*. Tato vlastnost je s výhodou využita při automatickém hlídání úrovně osvětlení. Aby bylo možné hlídat úroveň osvětlení v místnosti, je třeba použít snímače intenzity osvětlení a pomocí řídící jednotky regulovat intenzitu světla svítidel tak, aby bylo dosáhnuto nastavené hodnoty. Tato schopnost je využita zejména přes den, kdy je intenzita slunečního svitu proměnlivá a pomocí tohoto systému je možné doplňovat sluneční svit o světlo umělé.

Je možné také uvažovat prudké světlo z venku pronikající skrz okno dovnitř, které je až oslňující. V tomto případě nám nebude stačit možnost ovládání jasu osvětlení, ale budeme muset přidat možnost elektronického ovládání závěsů či, ještě líp, žaluzií. Výhodou elektronického ovládání žaluzií (či závěsů) můžeme využít i v tepleném řízení, kterému je věnovaná další podkapitola. Uplatnění také koneckonců najde v bezpečnostním managementu, kde žaluzie (či závěsy) mohou simulovat přítomnost obyvatel, což zmate potenciálního zloděje.

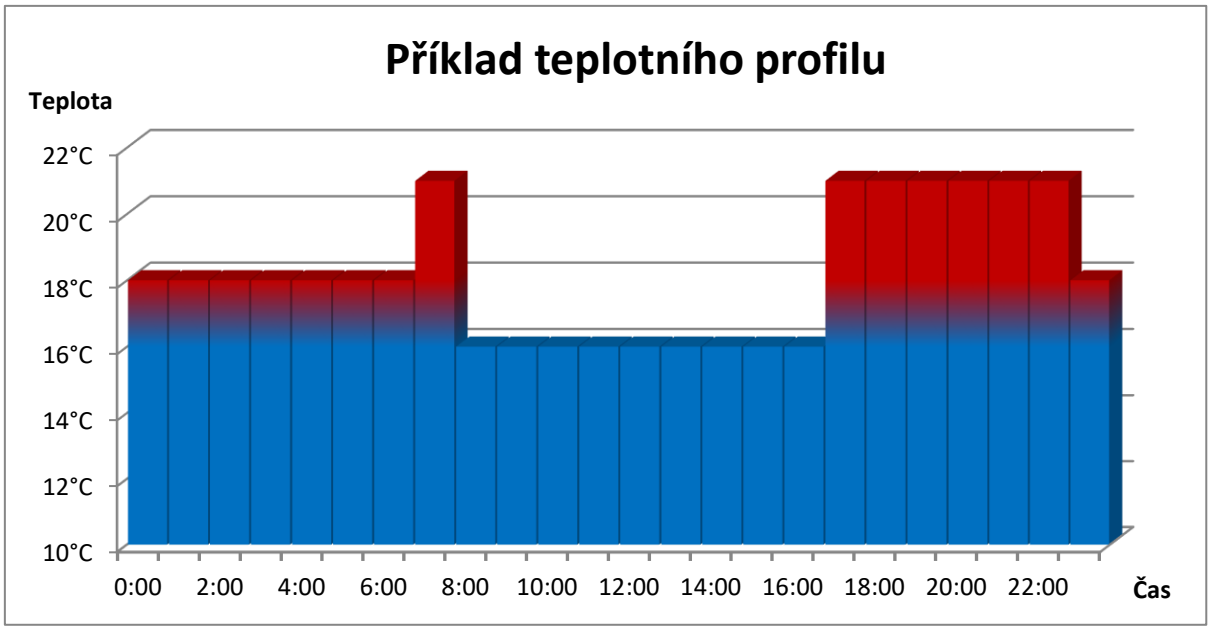
1.2.2 Tepelný management

Díky správnému tepelnému managementu můžeme ušetřit poměrnou část peněz a přitom zvýšit komfortnost své domácnosti. Málokdo si hlídá vypínání topení, když v zimě otevírá okno. „Přece je to pouze na chvíli a stejně bych to topení zapomněl zapnout“ říká si většina uživatelů, nebo je dokonce ani nenapadne, že by měli takové záležitosti řešit. Přitom v době, kdy je otevřené okno a topení je umístěno přímo pod oknem, jak tomu z praktických důvodů bývá, tak dochází k velmi rychlému ochlazování topení a vzduchu. Za chvíli by termostat vyhodnotil, že teplota klesla pod nastavenou úroveň, a zapnul by topení. Ovšem nestane se nic jiného, než to, že vyzářené teplo jde rovnou ven a tedy veškerá energie přichází vniveč. V případě implementace SID, centrální jednotka vyhodnotí pokles teploty, ale než začne topit, ověří si, zda není okno otevřené. Automatizovaný systém nezapne topení do doby, než se okno zavře. Samozřejmě není problém, aby si uživatel zapnul topení sám, má-li k tomu důvod. Právě popsaná situace pomáhá ušetřit energii při větrání oknem.

Druhým, ale signifikantnějším šetřením energie je snížení teploty v době nepřítomnosti uživatele. Právě tímto způsobem se ušetří podstatné množství energie. Čtenáři může vyvstat myšlenka, že bude snížen komfort dobou čekání na vytopení bytu po příchodu do bytu. Zajisté by tomu tak bylo, kdyby nebyla možnost si topení zapnout přes internet nebo dopředu nastavit, v kolik hodin má být byt již vytopený. Některé také může napadnout myšlenka šetření energie během večera, kdy se spí a není potřeba, aby teplota v domě byla příliš vysoká. Zde znovu přichází do popředí možnost nastavení si dané teploty pro danou hodinu podle potřeb uživatele domu. Příklad nastaveného teplotního profilu pro lepší pochopení zobrazuje obr. spolu s vysvětlující tab. 1. Teplotní profil je uvedený pouze pro celý byt, což ovšem není podmínkou. Byt může být rozdělen do jednotlivých zón, podle přání zákazníka, a tím si volit teplotní profil pro jednotlivé zóny. Teplotní management musí obecně počítat s několika možnostmi ovládání teploty pro zóny, či místnosti, chcete-li. Ne každý uživatel si bude chtít měnit teplotní profily, v případě, kdy jeho harmonogram dnů je spíše nepravidelný. Pro takového uživatele bude nejlepší, když si chvíli před příjezdem domů zvýší teplotu domácnosti přes Smartphone. Poslední zmíněnou možností, která sice zvyšuje komfort ovládání, ale naopak snižuje komfort teplotní, je nastavení pouze dvou teplot, a to v případě, kdy nikdo není doma a kdy je domov obýván. Teplotní profil by byl pak proměnlivý a závislý na příchodu a odchodu uživatelů domácnosti. Tímto způsobem ovšem bude vykoupen teplotní komfort, kdy na vytopení do požadované teploty od doby příchodu bude potřeba určitá doba.

tab. 1: Vysvětlení časových intervalů pro příklad teplotního profilu (obr.)

Od	Do	Teplota	Vysvětlení
0:00	7:00	18 °C	Doba spánku. Je lepší spát v chladnější místnosti.
7:00	8:00	22 °C	Doba vstávání. Vstávání je mnohem příjemnější do vyhřáté místnosti.
8:00	17:00	16 °C	Doba trávená v práci. Není potřeba vytápět byt, stačí pouze udržovat minimální teplotu.
17:00	23:00	22 °C	Doba trávená doma. Udržování komfortní teploty
23:00	24:00	18 °C	Doba spánku.



obr. 1: Příklad teplotního profilu během dne

Dalšími prvky teplotního managementu mohou být sofistikovanější techniky, než je pouze topení a větrání okny. Trh nabízí moderní tepelné výměníky, které se zejména používají v novějších stavbách a přednostně v pasivních domech. Mají tu skvělou vlastnost, že do domu přichází čerstvý vzduch z venku, který se ovšem ohřeje o vzduch, který z domu naopak vychází, jako vydýchaný. Jedná se o opravdu brilantní myšlenku, díky níž je šetřena energie, a uživatel domu zbavuje nutnosti větrání pomocí oken. Trh obecně nabízí mnoho možností, z nichž zbývá ty správné vybrat a vhodně zakomponovat do celkového systému.

1.2.3 Bezpečnostní management

Neméně důležitou součástí služeb nabízejících domem je právě bezpečnost. Jedná se vlastně o nejpoužívanější službu v domě a ve firmách v současné době. Dalo by se říct, že právě bezpečnost je velkou částí skupiny zákazníků upřednostňována před komfortem či úsporou energie. Je pochopitelné, že firmy mají větší zájem o bezpečnost už jen proto, že v budově mají povětšinou uložené zboží, či výrobky, v nichž mají část kapitálu.

Prvky pro hlídání bezpečnosti mohou být různé a různě kombinované. V základu se objevují senzory pohybu spolu se sirénou. Dále je možné přidat rozšíření o GSM (*Globální Systém pro Mobilní komunikaci*), který informuje uživatele o porušení klidových zón pohybem v době, kdy v místnosti nikdo být nemá. Kamery jsou velmi účinným prostředkem pro zvýšení bezpečí, už jen i pro poskytnutí důkazů v případě nepovoleného vniku do bytu. Také mohou být přidány prvky pro hlídání otevření oken a dveří, hlídání přítomnosti kouře, úniku plynu, úniku vody a další prvky.

1.2.4 IoT – Internet of Things

Poslední podkapitola nastiňuje možnost využití takzvaných „Internet of Things“ neboli „Internet věcí“. Tento pojem zahrnuje věci, které jsou schopny se připojit k internetu (intranetu), přes nějž jsou ovládány. IoT otevírají dveře novým možnostem. Velkou výhodou je propojení všech věcí do jednoho systému. Mít přehled o všech sbíraných informacích a ovládat všechna zařízení přes internet.

Příkladem rozšířené funkcionality je spárování kávovaru s budíkem. Jak ještě víc je možné zvýšit komfort, než ráno vstát a dát si čerstvě uvařenou kávu. Také je příjemné si nachystat kávu jen kliknutím na mobilu. Už jeden prvek IoT dokáže opravdu významně zvýšit komfort, již jen závisí na schopnostech designera systému implementovat zařízení.

1.3 Současný pohled

Možnosti, které jsou nabízeny inteligentními domácnostmi, jsou široké a pro většinu lidí dobře znějící. Nevýhodou těchto inteligentních domácností je pořizovací cena, která je obecně vyšší. Proto některé domácnosti vznikly skoupením modulů a vytvoření vlastního systému inteligentní domácnosti, jak uvádí Studie od firmy Microsoft za spolupráce Washingtonské univerzity [1].

Tato studie si vybrala 14 inteligentních, nebo chcete-li automatizovaných, domácností, které podrobili jejich výzkumu. Tyto domácnosti byly rozděleny na dvě části, kde první skupina byla takzvaně „DIY – Do-It-Yourself“ neboli udělej si sám a druhá skupina „Outsourced“, která zahrnovala profesionálně nainstalované inteligentní domácnosti specializovanou firmou. Obě skupiny se shodly na tom, že významnými problémy v SID je vysoká pořizovací cena, nedostatek flexibility nabízených možností, špatná ovladatelnost a těžko dosažitelná bezpečnost. Aby bylo možné tyto domy nabídnout široké veřejnosti, je potřeba, překonat tyto bariéry. Zmíněné bariéry nejspíše budou postupně zmírňovány až odstraněny díky konkurenčnímu boji.

Zajímavou tabulkou výzkumu [1] je porovnání pořizovacích cen a kombinací značek komponentů pro SID viz tab. 2.

tab. 2: Přibližné pořizovací ceny a použité značky SID [1]

ID		DOBA POUŽÍVÁNÍ [ROKY]	ZNAČKA	~ CENA [USD]	
DIY	D1	4	Elk M1	$	5 000
	D2	2	Elk M1, Charmed Quark	$	10 000
	D3	1,5	mControl, Leviton	$	5 000
	D4	1	X10	$	200
	D5	2	Lorax, BayWeb	$	14 500
	D6	2	Control4	$	50 000
	D7	5	X10, Active Home	$	300
	D8	2	Lagotek	$	5 000
	D9	10	ISY-99i, Insteon, X10	$	3 000
OUT-SOURCED	O1	3	HAI, neznámá		Neznámá
	O2	6	Creston	$	60 000
	O3	2,5	Control4	$	120 000
	O4	10	EIB Instabus, KNX	$	13 500
	O5	2	Lagotek, AudioQuest	$	20 000

Ve výše uvedené tab. 2 si můžeme všimnout nejen cen, které ve své podstatě uvádí skutečnost, že inteligentní domácnosti „DIY“ – vytvořené pomocí svých sil, jsou spíše levnější. Podíváme-li se na sloupec značek použitých komponent, tak nás musí zarazit pospojování více značek dohromady. Zmíněné spojování značek dohromady nesvědčí kompatibilitě jednotlivých komponent a nutí uživatele mít nainstalováno více ovládacích prvků. Ovládací prvky mohou být jak softwarové, tak i hardwarové, ale jednotlivé ovládací prvky jsou dedikovány pro jeden daný systém stejné značky. V této kapitole již byl zmíněn problém nepružnosti systémů inteligentních domácností, což může být právě příčina instalování systémů od různých výrobců. Zmiňovaný výzkum [1] také říká, že SID typu „DIY“ v porovnání s „Outsourced“ disponovaly vyšší funkcionalitou a rozšiřitelností. Systémy inteligentní domácnosti instalované firmami již nebylo možné rozšířit a byly tedy statickými.

Neméně důležitou otázkou je i to, co vlastník inteligentního domu požaduje. Dá se předpokládat, že základní rysy, jako je ovládání osvětlení, medií, prostředí (teplota) a zajištění bezpečnosti, jsou požadovány většinou vlastníků. Zmiňovaný Microsoft výzkum [1] provedl podrobnější šetření i v oblasti požadovaných vlastností inteligentního domu, jehož výsledek je v tab. 3.

tab. 3: Odpovědi účastníků, zda aplikaci již mají instalovanou, zakoupili by ji či nemají zájem

	Aplikace	Instalováno	Zakoupil bych	Nezájem
Ovládání	Nastavení "scén"	24	5	0
	Centralizované ovládání systému	24	4	1
Média	Poslání obrazu počítače na TV	19	5	5
	Sledování nahrané TV na jakékoliv TV v domácnosti	16	11	2
	Poslání obrazu počítače na Smartphone	4	20	5
	Poslání obrazu Smartphone na TV	3	13	13
	Přeposlání video-hovorů mezi zařízeními	0	13	15
Zabezpečení & Monitoring	Vzdálený přístup k domácím kamerám	16	10	3
	Automatické upozornění (např. zapnutá kamna)	9	18	2
	Vzdáleně otevřít vchodové dveře	4	18	7
	Záznam užívání zařízení	3	9	17
	Časové omezení používání zařízení	1	15	12
	Sledování dětského počítače na TV	0	17	12
Prostředí	Termostat, který se učí rutiny	16	10	3
	Sledování energetické náročnosti domu	3	23	3
	Zapínání zařízení dle přítomnosti	3	20	6
	Automatické nastavení oken a clon k udržení komfortu	1	24	4

Tato podkapitola nastínila vyvstávající problémy v oblasti inteligentních domácností a v jakých oblastech managementu se inteligentní domácnosti nejčastěji pohybují. Tab. 2 nastínila přibližné pořizovací ceny inteligentních domácností a reprezentovala reálný problém spojení více značek modulů, které musí být pospolu propojeny, aby splňovaly přání zákazníka. Tab. 3 se pak pustila do bližšího průzkumu ohledně požadovaných funkcí. Některé funkce jsou pomalu brány jako povinné, například centralizované ovládání systému. Najdou se ovšem funkce, které jsou spíše nezajímavé. Mezi takové, podle tab. 3, patří záznam užívání zařízení inteligentní domácnosti.

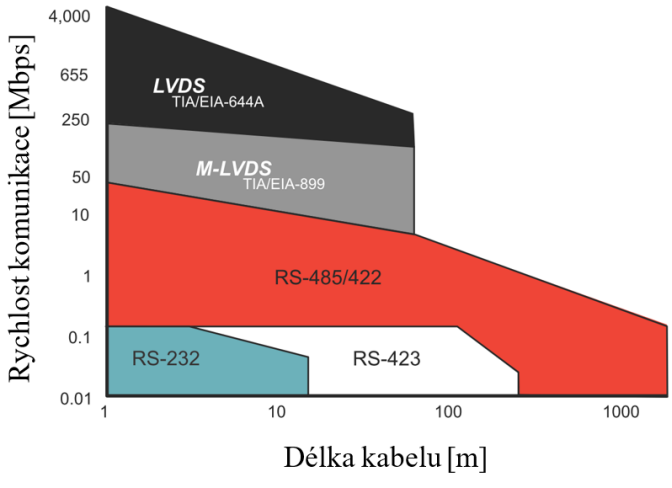
2 Síť a komunikační rozhraní inteligentní domácnosti

Důležitým úkolem sítě je spolehlivé propojení jednotlivých zařízení, které jsou díky ní schopny komunikace. Druhů sítě je mnoho a jejich dělení rozsáhlé. Důležitým požadavkem sítě pro inteligentní domácnost je spolehlivost. Proto se dále zaměříme na průmyslové sítě, jež se vyznačují vysokou spolehlivostí. Jaké jsou vlastně další požadavky na síť? Druhým požadavkem je komunikace na dlouhou vzdálenost. I když se bude jednat o malou domácnost, může disponovat větším počtem zařízení, jejichž propojení může být spletité, a tím délka sítě značně naroste. Také nenáročnost na management komunikace, finanční nenáročnost a omezení nabídky sběrnic podporovány mikrokontroléry se odrazí ve výběru té optimální sítě a komunikačního rozhraní.

Vzhledem k předchozím znalostem rodiny AVR mikrokontrolérů, zejména typů ATmega, firmy Atmel, které již od základních verzí mikrokontroléru podporují univerzální asynchronní sériovou komunikaci (UART - *Universal Asynchronous Receiver/Transmitter*), byla vybrána právě tato komunikace. Čtenář by mohl navrhnout také komunikační sběrnici CAN (Controller Area Network), která je známá spíše z automobilového průmyslu. CAN sběrnice by byla také vhodná pro implementaci do inteligentního domu. I tato možnost byla zvažována, ale nižší a levnější řady mikrokontroléru ATmega nepodporují CAN sběrnici. Dalším aspektem byla cena převodníku z logické úrovně vůči potenciálu nuly na logickou úroveň reprezentovanou diferenčním výstupem. Drivery pro CAN diferenční výstup byly někdy až 5 krát dražší.

2.1 Standard RS-485

Právě UART je spojován s komunikačním standardem RS-485, který je doposud používaný v průmyslu pro jeho vysokou spolehlivost. Jedná se snad o nejuniverzálnější komunikační standard, který propojuje zařízení napřímo bez potřeby modemu. Důležitou vlastností je možnost vytvoření sítě, na kterou je možno připojit daný počet zařízení (až 256), který je udávaný linkovým převodníkem (transceiverem). Výhodná je vzdálenost, na kterou dokáží moduly komunikovat, a to až na 1200 m. Rychlost komunikace je také nezanedbatelná, jejíž maximum je přibližně 50 Mbps (= 1 Mega bit per second → 1 milión bitů za sekundu). Rychlost komunikace jde proti vzdálenosti, na kterou je komunikace spolehlivá. Na přesnější údaje se podíváme v následující podkapitole 2.1.1. [2]

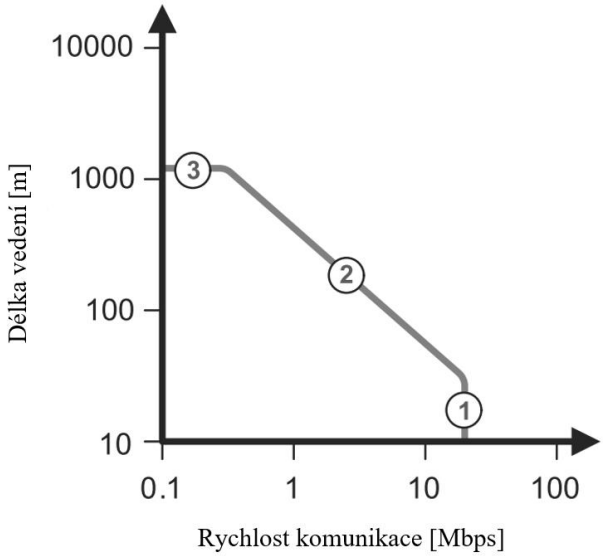


obr. 2: Porovnávní jednotlivých standardů kominukace [2]

Grafické porovnání délky kabelu a rychlosti komunikace u jednotlivých standardů je znázorněno na obr. 2. Na první pohled lze vidět, že standard RS-485 nabízí největší vzdálenost komunikace a s více než dostačující rychlostí komunikace. Nejvyšší rychlost komunikace nabízí rozhraní LVDS (Low-Voltage Differential Signalling – nízkonapěťová diferenční „signalizace“), která se velmi často používá pro komunikaci u displejů, kde velká vzdálenost není na místě.

2.1.1 Rychlost komunikace vs. délka kabelu

Každý kabel má parazitní vlastnosti, které omezují rychlost komunikace. Většinou jsou parazitní vlastnosti udávané na jeden metr kabelu, z čehož můžeme usoudit, že čím delší kabel máme, tím výraznější jsou parazitní vlastnosti na celé délce kabelu. Právě parazitní vlastnosti kabelu nám snižují maximální možnou rychlost komunikace. Většinou se jedná o kapacitu vodiče, která se spolu s reálným odporem vodiče chová jako dolní propust, což se zpočátku projeví nízkou rychlostí přeběhu náběžné a sestupné hrany. Grafické znázornění této závislosti lze vidět na následujícím obr. 3:



obr. 3: Závislost délky vedení na rychlosti komunikace [3]

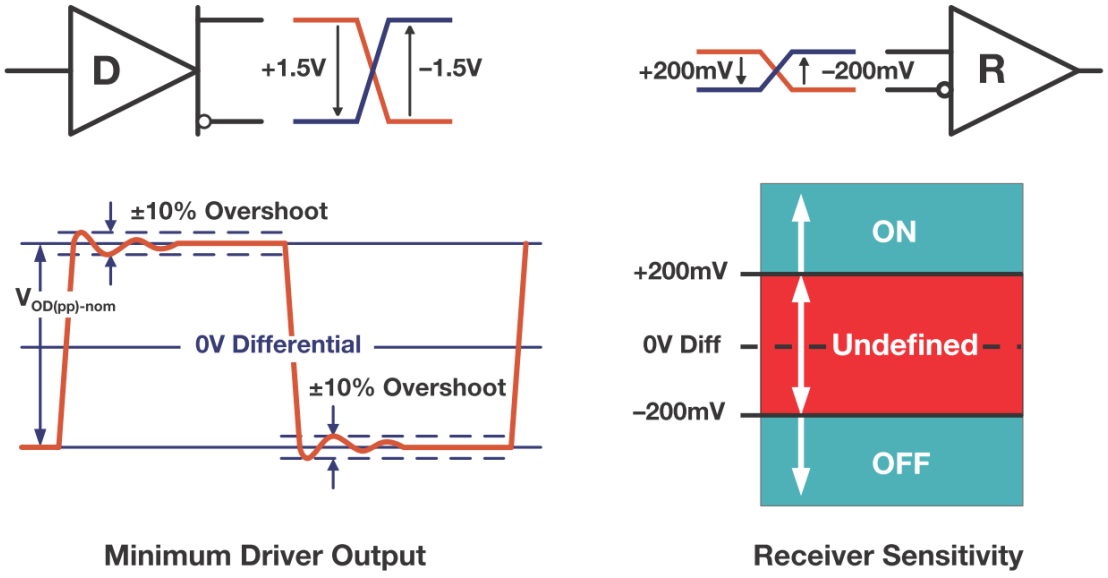
Popis jednotlivých částí v grafu (obr. 3) [3]:

1. Část 1 v grafu reprezentuje oblast vysoké rychlosti komunikace skrze krátký kabel. Ztráty na kabelu můžeme zanedbat díky jeho krátkosti. Rychlost komunikace je hlavně dána linkovými drivery a jejich schopnosti generovat logické úrovně s velkou rychlostí náběžné hrany. I když standard doporučuje 10 Mbps, dnešní rychlé obvody mohou komunikovat až na rychlosti 40 Mbps.
2. Část 2 grafu ukazuje přenos dat od krátkých až po dlouhé komunikační linky. Ztráty na komunikační lince již musí být brány v potaz, proto s rostoucí délkou kabelu musí být rychlost komunikace snižována. Odhad od oka říká: „vynásobí-li se délka kabelu v metrech s rychlostí komunikace v Mbps, hodnota by měla být menší než 10^7“. Toto pravidlo je mnohem konzervativnější než dovolují dnešní kabely, proto při rozumném překročení zobrazené „limity“ bude komunikace stále fungovat.

3. Část 3 zastupuje komunikaci na nízkých komunikačních rychlostech, tedy na nízkých frekvencích. V této části již nejsme omezeni kapacitami vodiče, které by snižovaly rychlost náběžných a sestupných hran. Projevuje se čistý odpor vodiče, který se hodnotou blíží hodnotě zakončovacích rezistorů (120 Ω) a stává se z této konfigurace dělič napětí 1:1. Reprezentace v decibelech je -6 dB. Pro kabel 22 AWG, 120 Ω, UTP, tento jev nastává přibližně na 1 200 metrech.

2.1.2 Úrovně signálu definované standardem

Díky vhodnému definování výstupních a vstupních napětí dle normy RS-485 je komunikace spolehlivá i na velké vzdálenosti. Výstupní napětí z driveru je definováno pro ±1,5 V, mezitímco vstupní napětí je definováno již od napětí ±200 mV. Právě tato velká napěťové vůle počítá s tím, že na dlouhých kabelech bude útlum napětí. Tento předpoklad zajistí spolehlivost i na velké komunikační vzdálenosti. Následující obrázek (obr. 4) graficky znázorňuje úrovně napětí, jak jsou definovány standardem.



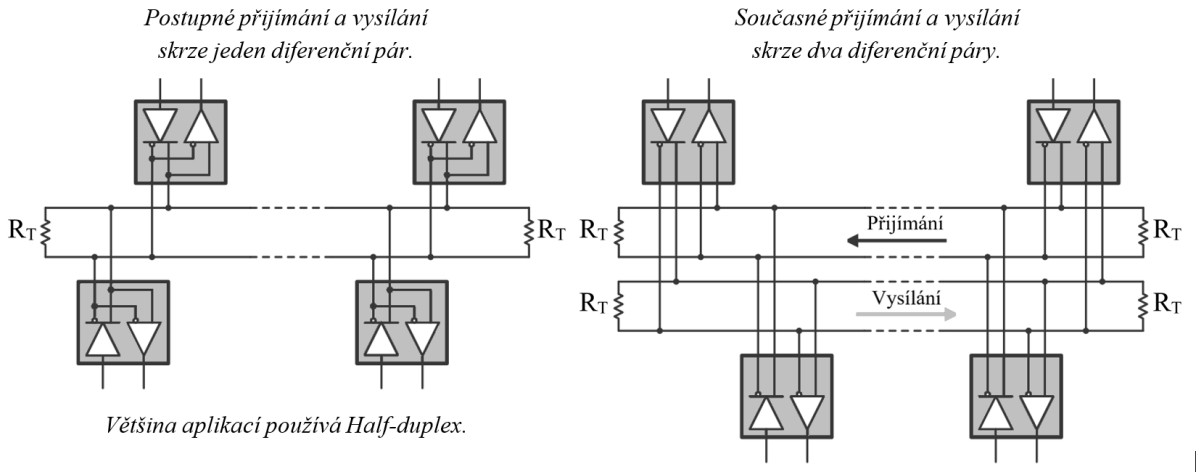
obr. 4: Napěťové úrovně definovány standardem RS-485 [2]

Rozdělíme-li si obr. 4 na levou a pravou část, všimneme si, že se levá půlka zabývá výstupními napětími z driverů, mezitímco se pravá půlka zabývá vstupními napětími. Druhý kvadrant zobrazuje vysílač a první kvadrant přijímač, a jejich napěťové povahy. Třetí kvadrant upozorňuje na nutnost počítání s překmity (na obrázku označeno jako Overshoot), které by neměly přesáhnout 10 %. Čtvrtý kvadrant pak graficky reprezentuje definici logických úrovní na vstupu. Ještě malá poznámka k obrázku, který je převzat z literatury [2], šipky znázorňující orientaci napětí jsou (pro obrázek v 1. a 2. kvadrantu) špatně, když vezmeme v potaz, že jsou i opačná znaménka. Správně by tedy bylo: Ponechat šipky v originálním směru, a zapsat stejná znaménka (buď plus či mínus). Druhý způsob opravy by byl v ponechání znamének a šipky zakreslit do stejných směrů.

2.1.3 Duplexní versus polo-duplexní komunikace

Standard RS-485 nabízí dva způsoby komunikace a to Full-Duplex a Half-Duplex. Do češtiny tento pojem překládáme jako duplexní a polo-duplexní komunikace. Jak je již z názvu patrné, jedná-li se o duplexní komunikaci, probíhá komunikace současně obousměrně. Naopak tomu je při polo-duplexní komunikaci, kdy je umožněna jen jednosměrná komunikace, ale je možné mezi jednotlivými směry přepínat, tedy přepínat mezi přijímáním a vysíláním dat.

Výhodou duplexní komunikace je dvojnásobná prostupnost dat oproti polo-duplexní. Druhou nespornou výhodou duplexní komunikace je vyšší, spolehlivost. Ač se to zprvu nezdá, je možné při chybě na přijímací straně (z pohledu řídící jednotky), kdy je komunikace zablokována vysílací jednotkou (například senzorem), která z důvodu chyby drží logickou úroveň v jedné hodnotě, donutit řídící jednotkou restartovat zařízení na komunikační sběrnici zprávou „*restart*“, a tím odstranit (pouze softwarovou) chybu. V případě, kdy komunikace není zcela řízena řídící jednotkou, kvůli zvýšení reakční rychlosti, a senzory mohou začít komunikovat bez vyzvání, tak je při duplexní komunikaci nižší pravděpodobnost střetu dvou zároveň vysílaných zpráv.



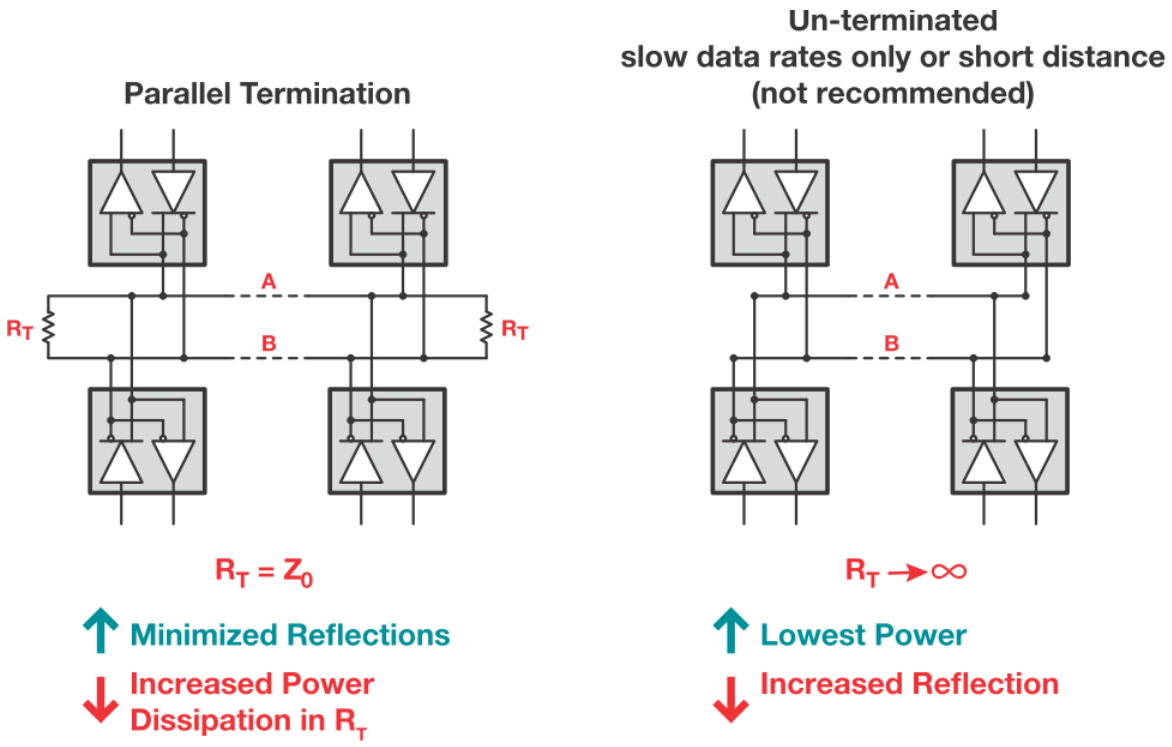
obr. 5: Duplexní a poloduplexní komunikace [2]

Vezmeme-li v potaz stránku ekonomickou, tak bude duplexní komunikace náročnější a také bude mít větší odběr energie při komunikaci. Jelikož je potřeba v duplexní komunikaci vést kabely jak pro přijímání, tak i pro vysílání zpráv, je nutné počítat s tím, že v kabelu budeme potřebovat o dvě žíly víc (jeden pár), než pro komunikaci polo-duplexní. Odběr energie je zvýšen také tím, že je potřeba použít dvakrát více integrovaných driverů. To ovšem záleží na tom, jaké drivery si vybereme. Na trhu jsou jak drivery, které mají vyvedeny zvlášť diferenční páry pro příjem a pro vyslání, tak i drivery, které mají výstup jen jeden a jsou spíše určeny pro polo-duplexní režim komunikace. Při výběru driveru jste ovlivňováni spoustou dalších parametrů a tak se může stát, že jste nuceni využít drivery prioritně určené pro polo-duplexní komunikaci, ovšem využijete dva, abyste dosáhli plnohodnotné duplexní komunikace.

Poznámka k polo-duplexní a simplexní komunikaci. Simplexní komunikace je taková komunikace, která umožňuje pouze jeden směr komunikace, bez jeho změny. Polo-duplexní, jak již bylo zmíněno výše, také poskytuje jednosměrnou komunikaci v jednu dobu, ale lze přepínat mezi směry. Jelikož simplexní komunikace jako taková se téměř nepoužívá, tak je název polo-duplexní často nahrazován za simplexní.

2.1.4 Terminace sběrnice

Terminace, neboli zakončení sběrnice, je prováděno pomocí zakončovacích rezistorů o definované hodnotě. Nejdůležitějším úkolem zakončovacích rezistorů je impedanční přizpůsobení konců vodiče, a tím zamezit odrazům ve vedení. Na druhou stranu se objevuje nežádoucí efekt a to zvýšení spotřeby během komunikace, jelikož drivery dodávají proud do zátěže, která je přibližně určena paralelní kombinací zakončovacích rezistorů. Hodnota zakončovacích rezistorů je daná charakteristickou impedancí komunikačního kabelu. Standard RS-485 doporučuje charakteristickou impedanci vedení 120 Ω, tudíž terminování by mělo být provedeno rezistorem o nominální hodnotě 120 Ω. Terminování je prováděno vždy na obou koncích vedení.



obr. 6: Terminace vedení [2]

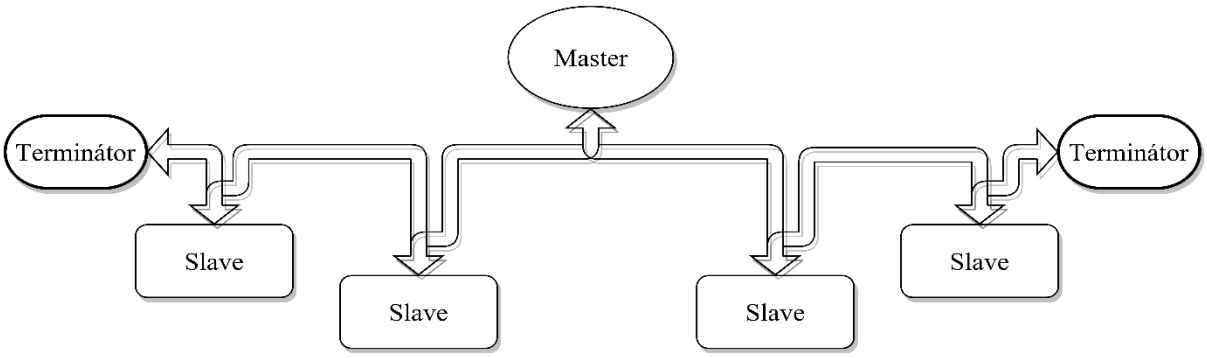
Díky předchozí kapitole (**Chyba! Nenalezen zdroj odkazů. Chyba! Nenalezen zdroj odkazů.**) poznáme, že obr. 6 zobrazuje komunikační sběrnici s typem polo-duplexní komunikace, ovšem teď se zaměříme na zakončovací rezistory. Levá část obrázku ukazuje ten lepší přístup, kdy použijeme zakončovací rezistory s odporem charakteristické impedance vedení pro minimalizaci reflexí ovšem na úkor větší spotřeby. Pravá část nám sice nabízí výhodu menší spotřeby, ovšem je možné ji použít pouze pro nízké komunikační rychlosti či vzdálenosti.

Obecně neterminované vedení není doporučováno a po praktickém odzkoušení je možno pouze potvrdit, že terminace mnohonásobně zvýší spolehlivost komunikace. Přibližně na vzdálenost 120 m s asi 23 připojenými moduly a komunikační rychlostí ~143 kbps je komunikace téměř neproveditelná bez těchto zakončovacích rezistorů. Velmi často se stalo, že zpráva ani nedorazila, natož aby se v ní objevily pouze chybné bity. Po připojení jednoho zakončovacího rezistoru se komunikace zlepšila natolik, že při třech pokusech o odeslání se většinou odeslání zdařilo. Připojením obou zakončovacích rezistorů se stala komunikace již bezeztrátovou.

2.2 Topologie sítě a komunikační kabel

Standard RS-485 doporučuje zapojení jednotlivých jednotek do sériové topologie, sběrnice, nebo také jinak řečeno zřetězené topologie. Jelikož se jedná jen o různá pojmenování jedné a té samé topologie, nadále bude tato topologie označena jako sběrnice. Hlavním rysem sběrnice je připojení všech jednotek k jednomu hlavnímu kabelu. Sběrnice není nijak větvena, jedná se o dlouhou lineární sběrnici, která má pouze malé odbočky k jednotlivým modulům na síti. I když některé topologie by byly výhodnější z pohledu spotřeby kabelu a složitosti propojování modulů, je potřeba se držet standardu. Sběrnice má přirozeně pouze jeden začátek a jeden konec, které jsou zakončeny rezistory, viz předchozí kapitola 2.1.4. Na obr. 7 je zobrazena ideální topologie sítě v inteligentním domu. Obrázek trochu předbíhá kapitolu, jelikož již rozlišuje jednotky Master a Slave, které budou rozebrány později.

Literatura [2], také zmiňuje topologii hlavního řetězce s odbočkami, ovšem u ní je poznámka, že tato síť „funguje“ (ale není nejlepší volbou). Topologie sběrnice je zde uvedena jako nejlepším řešením, a proto tato topologie bude i využita v této práci. Nesmíme opomenout vyšší nároky na komunikaci po síti, jako je komunikace na větší vzdálenost spolu s vyšší rychlostí komunikace a hlavně spolehlivostí.



obr. 7: Ideální topologie sběrnice (RS-485)

Standard RS-485 využívá výhod diferenčního vstupu a výstupu, a proto je záhodno využít kabel, který má v sobě kroucené dvojlinky. Nabízí se i koaxiální kabel, ale ten je drahý, mechanicky náchylný a nemá vícero žil, proto je tento kabel nevhodný. Už když se řekne kabel s kroucenými dvojlinkami, tak by nás hned mohl napadnout kabel UTP (*Unshield Twisted Pair*) nebo STP (*Shield Twisted Pair*). Je výhodný tím, že je bezproblémově dostupný, díky velkým odběrům je levný a má v sobě čtyři kroucené dvojlinky. Už se nabízí jen otázka, jakou má charakteristickou impedanci, abychom mohli rozhodnout, zda je UTP kabel vhodný k použití. Kabel UTP kategorie 5 má v charakteristických vlastnostech uvedenou impedanci 100 Ω (±10 %), což je velmi blízko požadované hodnotě 120 Ω. Praktická zkouška potvrdila, že tento kabel je vhodný pro účely komunikace inteligentního domu.

UTP nabízí 4 páry kroucené dvojlinky, jak je tedy využijeme? Z předchozích kapitol víme, že bude použita duplexní komunikace a také proč. Zbývají nám tedy ještě dva páry a otázka napájení modulů připojených na komunikační síti. Abychom si objasnili myšlenku napájení připojených modulů, trochu předběhneme kapitolu a řekneme si něco k modulům. Z pohledu napájení modulů můžeme rozlišit dva typy modulů a to pasivní a aktivní z pohledu napájení. Aktivní moduly budou schopny dodávat energii do sítě a pasivní ji naopak z ní brát. Síť bude tedy navrhnuta tak, aby na ní bylo dostatek vhodně

umístěných aktivních modulů, které budou schopny napájet moduly pasivní, bez značného úbytku napětí na vedení. Pro napájení je potřeba pouze dvou vodičů, jeden pár, a to s nulovým potenciálem a s kladným napětím. Jelikož jsou volné dva páry, použijeme oba páry na napájení.

2.3 Vrstvy komunikačního modelu

Model komunikačního protokolu se uspořádává do vrstev, díky kterým je pak možné jednotlivé vrstvy modifikovat, bez toho, aniž by se muselo zasahovat do ostatních vrstev. Samozřejmě, dojde-li k větším změnám v jedné vrstvě, může vyvstat potřeba mírně předělat i vrstvu vyšší, ale pouze na její straně blíže k předělávané vrstvě. Existuje Referenční model ISO/OSI, o kterém se více dozvíme v následující podkapitole. Ve zkratce je zmiňován z důvodu částečné implementace modelu do komunikace.

2.3.1 Referenční model ISO/OSI

Referenční model ISO/OSI (*Open System Interconnection*) je zde uváděn pro objasnění úkolů jednotlivých vrstev v komunikačním protokolu. Vznikl již v roce 1984 v mezinárodní organizaci pro normalizace ISO (*International Organization for Standardization*). Využívá se při komunikaci mezi otevřenými datovými systémy, což se odráží v jeho názvu RM OSI (*Reference Model of Open Systems Interconnection*) [4]. Otevřené datové systémy lze chápat jako systémy nezávislé na konkrétním výrobci. RM ISO/OSI je jakousi ucelenou představou o tom, jak by měly být koncipovány počítačové sítě. Při dodržení všech podmínek definovaných tímto modelem máme jistotu, že budou různí účastníci sériového přenosu bezchybně mezi sebou komunikovat.

Sedmivrstvý model definuje funkci a podporu služby jednotlivým vrstvám, které jsou zmíněny a ve zkratce popsány v následujících řádcích. Jmenování jednotlivých vrstev je postupné a začíná na vrstvě nejnižší [4]:

1. **Fyzická vrstva** (angl. *Physical layer*) – Definuje veškeré elektrické a fyzikální vlastnosti připojených zařízení. Říká, jaké napěťové úrovně jsou použity, jaké je rozložení pinů, doporučuje typ kabelu, definuje význam změn a délky trvání logických úrovní. Stará se o navázání a ukončení komunikace s adresovaným zařízením. Další funkcí jsou konverze digitálních úrovní na signály používané pro komunikaci.
2. **Spojová vrstva** (angl. *Data link layer*) – Uspořádává data z fyzické vrstvy do logických celků, v této práci nazývanými zprávami. Nastavuje parametry pro fyzickou vrstvu a hlásí neopravitelné chyby. Formátuje fyzické rámce a přidává k nim fyzickou adresu.
3. **Síťová vrstva** (angl. *Network layer*) – Jejím úkolem je směrování zpráv v sítí a síťové adresování. Zajišťuje překlenutí rozdílných technologií v přenosových sítích.
4. **Transportní vrstva** (angl. *Transport layer*) – Obstarává přenos zpráv mezi koncovými uzly. Poskytuje určitou kvalitu přenosu, která je žádána protokoly nad touto vrstvou. Vrstva disponuje spojově (TCP) a nespojově (UDP) orientovanými protokoly.
5. **Relační vrstva** (angl. *Session layer*) – Organizuje a synchronizuje dialog mezi spolupracujícími relačními vrstvami obou systémů a řídit výměnu dat mezi nimi. Stará se o vytváření a ukončování relačních spojení, jeho synchronizaci a obnovení.

6. **Prezentační vrstva** (angl. *Presentation layer*) – Transformuje data do formátu, jež používají aplikace, jako je šifrování, konverze a komprimace. Formát dat se může lišit mezi oběma komunikujícími systémy, také se data transformují pro účely přenosu dat nižšími vrstvami.
7. **Aplikační vrstva** (angl. *Application layer*) – Poskytuje aplikacím přístup ke komunikačnímu systému a umožňuje jim mezi sebou spolupracovat.

Dělení do sedmi vrstev je pro komplexní komunikaci, kterou není třeba implementovat do sítě inteligentního bytu. Je potřeba se držet jednoduchého a spolehlivého návrhu, proto sedm vrstev bude redukováno do vrstev tří, a to fyzické, spojové a aplikační. Postupně si je představíme v následujících třech podkapitolách.

2.3.2 Fyzická vrstva

Většina fyzické vrstvy je již popsaná v předchozích kapitolách, jelikož se řídí standardem RS-485. Jsou tedy řešeny napěťové úrovně (diferenční napětí), kabeláž (UTP), topologie (sběrnice) a typ komunikace (UART). Pro úplnost popisu fyzické vrstvy je ještě potřeba zvolit konektor, rozložení jeho pinů a bližší specifikace UART komunikace (počet bitů, paritní bity, baud-rate, atd.).

Konektor – již se využívá UTP kabel, tak se nabízí otázka, proč nevyužít i konektor, který se nejčastěji používá pro UTP kabely a to konektory s označením RJ-45. Přesněji se jedná o konektor RJ-45 8p8c (8 positions, 8 contacts = 8 pozic, 8 kontaktů). Existuje standard T586A a T586B, který popisuje, jak mají být rozmístěny barevné vodiče v konektoru. Ovšem tímto standardem se již zapojení neřídí a je uděláno jednodušeji, bez přeskakování párů. Dva vývody jednoho páru jsou vždy umístěny v přímé blízkosti, jak lze vidět v (tab. 4).

tab. 4: Rozložení barev žil UTP kabelu

#	BARVA ŽÍLY
1	Oranžová s bílými proužky
2	Oranžová
3	Zelená s bílými proužky
4	Zelená
5	Modrá s bílými proužky
6	Modrá
7	Hnědá s bílými proužky
8	Hnědá

Již v předchozí kapitole byl zmíněn kabel UTP a náznak rozdělení funkcí jednotlivých párů. Blíže se na rozdělení funkcí podíváme v následující tab. 5.

tab. 5: Funkce diferenčních párů v UTP kabelu

Pár	Funkce
1-2	Napájení: +5 V → Oranžová s bílými proužky | GND → Oranžová
3-4	Datová diferenční linka směrující ze Slave do Master modulu
5-6	Datová diferenční linka směrující z Master do Slave modulu
7-8	Napájení: +5 V → Hnědá s bílými proužky | GND → Hnědá

UART – Mikrokontrolér ATmega podporuje mód multiprocesorové komunikace. Tento mód umožňuje filtrování příchozích rámců přímo v hardware USART (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*) přijímači. Rámce, které neobsahují informaci o adrese, jsou ignorovány a nejsou ukládány do vstupního zásobníku. Tento přístup efektivně redukuje počet příchozích rámců, které by jinak musely být obsluhovány CPU (*Central Processor Unit*), v systému s vícero mikrokontroléry, které komunikují skrze stejnou sériovou sběrnici. Vysílací hardwarová jednotka není nijak ovlivněna nastavením toho typu komunikace, ale je nutné ji používat odlišně právě tehdy, když je součástí systému používající multiprocesorový komunikační mód.

Když je přijímač nastaven pro přijímání rámce, který obsahuje 5 až 8 bitů, potom první stop bit indikuje, zda rámec obsahuje informaci o datech či adrese. Pokud je přijímač nastavený pro přijímání rámce s devíti bity, potom devátý bit je užitý pro determinaci typu rámce (adresový/datový). Je-li poslední bit (v případě devítibitové komunikace bit devátý, jinak první stop bit) jedničkový, potom rámec je adresový. V opačném případě se jedná o datovou zprávu. [5]

Mód multiprocesorové komunikace umožňuje přijímat zprávy několika Slave mikrokontroléry od Master mikrokontroléru (dále označován jako Slave (modul), resp. Master). Popsaná vlastnost je zajištěna porovnáním přijaté adresy Slavem s jeho nastavenou. Pokud porovnání adres je pozitivní, tak se přepne na přijímání i datových rámců a data přijme. Pokud výsledek porovnávání je negativní, potom daný Slave ignoruje datové rámce, do doby než přijde po sběrnici další adresový rámec.

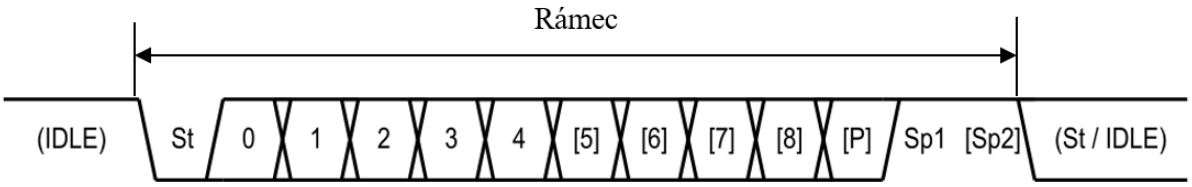
Datový/Adresový rámec

Sériový rámec je definován jako jeden znak složený z datových bitů, synchronizačních bitů (start a stop bity) a volitelného bitu paritního. Hardwarová implementace USART v mikrokontroléru ATmega8 akceptuje všech 30 kombinací vytvořených z následujících možností formátu:

- 1 start bit (neměnný)
- 5 – 9 datových bitů 5 možností
- žádná, sudá či lichá parita 3 možnosti
- jeden či dva stop bity 2 možnosti

Rámec začíná start bitem následovaným LSB (*Least Significant Bit* = nejméně významný bit). Potom následují datové bity, kterých je až devět, až po MSB (*Most Significant bit* = nejvíce významný bit). Pokud je povolena parita, je do rámce vložen paritní bit a to mezi datové bity a stop bit/y. Stop bit může být jeden či dva, dle nastavení komunikace. Když je odeslaný celý rámec, může být okamžitě vysílaný další rámec začínající znovu start bitem. Pokud není potřeba vysílání dalšího rámce, přechází

sběrnice do nečinného stavu (angl. Idle). Tento stav má logickou úroveň jedna. Následuje obrázek (obr. 8), který graficky znázorňuje podobu rámce. Bity v hranatých závorkách jsou volitelné.



obr. 8: Formát rámce UART [5]

Vysvětlení zkratek použitých v obr. 8:

- **St** – start bit
- **n** – datové bity (0 až 8)
- **P** – paritní bit
- **Sp** – Stop bit, vždy v logické jedničce
- **IDLE** – nečinný stav = neprobíhá ani přijímání ani vysílání. Vždy v logické jedničce

Po představení celého rámce si jen uvedeme, jaké jsou parametry komunikace po sběrnici SID, koneckonců to nás nejvíc zajímá.

- Počet datových bitů: 9 (poslední udává typ rámce)
- Paritní bit: žádný
- Počet stop bitů: 1
- Komunikační rychlost: ~143 kbps

Výhodně je tedy v komunikaci využito devět bitů, díky nimž posíláme zprávu o délce jednoho bytu a přitom jsme pořád schopni determinovat typ rámce (datový/adresový). Přináší to ale i svá úskalí v případě, kdybychom chtěli použít pouze převodník napěťových úrovní a vést tuto komunikaci přímo do počítače, který by ji vyhodnocoval, neboli choval se jako jednotka Master. Většina počítačových COM portů podporuje nanejvýše 8 bitovou komunikaci. Nehledě na nenormovanou komunikační rychlost (Baud Rate), jež je přibližně 143 kbps. Problém se zvolenou rychlostí by se dal řešit nalezením vhodnější komunikační rychlosti, která by byla blíž normované hodnotě. Nejvhodnějším způsobem by bylo vytvoření jednoduchého převodníku z komunikace na sběrnici na komunikaci pro počítač. Komunikace s počítačem nemusí být již nutně typu UART. Bližší informace budou rozebrány v další kapitole.

2.3.3 Spojová vrstva

Příchozí byty z fyzické vrstvy jsou analyzovány spojovou vrstvou. Ta nalézá v příchozích bytech zprávy neboli logické celky, a to díky jednoduchému formátu zprávy, která obsahuje v prvním datovém rámci (bytu) informaci o zprávě – typ zprávy a její délku. Jakmile spojová vrstva zná délku zprávy, ví přesně, kolik ještě datových rámců musí přijmout. O ukončení přijímání již nemusí spojová, ani žádná další vrstva, rozhodovat, jelikož se o to postará fyzická vrstva, která si také přečte první informační byte.

Kontrola chyb je také na seznamu úkolů spojové vrstvy. V případě využití ATmegy pro komunikaci přes UART komponentu je implementace kontroly paritní chyby již realizována v hardwarové podobě. Na nás je pouze možnost, zda tuto kontrolu chyb využijeme, ovšem uděláme-li tak, zahrneme kontrolu parity již do vrstvy fyzické. Kontrola pomocí paritního bitu nebyla zvolena, a tím tedy zachováváme myšlenku Referenčního modelu ISO/OSI, kdy kontrolu chyb řeší spojová vrstva. Samozřejmě tento fakt nebyl argumentem pro nevyužití paritního bitu. Byla dána přednost CRC8 kontrole, neboli 8bit cyklického redundantního součtu (angl. *8bits Cyclic redundancy check*). Čím širší je redundantní kontrolní součet, myšleno více bytový, tím je větší pravděpodobnost odhalení chyby ve zprávě. Nevýhodou širších CRC kódu je potřebný výpočetní výkon, kterým mikrokontrolér nižší řady ATmega nedisponuje. Zajisté by tento mikrokontrolér byl schopný spočíst širší redundantní součet, ale docházelo by k prodlevám mezi odesláním zprávy a přijetím kontrolního potvrzení, čímž by se prodloužila celá komunikace, což není žádoucí. Právě proto nebyla zvolena větší šířka CRC. Každá zpráva na svém konci obsahuje CRC8 byte. Jakmile se přijme celá zpráva do zásobníku, funkce se postará o to, aby byl spočítán CRC8 kód příchozí zprávy, který se potom porovná s přijatým CRC8 kódem. Jsou-li cyklické redundantní součty shodné, je posláno potvrzení Masterovi o přijetí zprávy. Zpráva je následně poslána o vrstvu výš, do vrstvy aplikační. Obsahuje-li zpráva chybu, cyklické redundantní součty nejsou shodné, neodesílá se potvrzovací zpráva Masterovi. Chybná zpráva není již odesílána do vyšší vrstvy, jedná-li se o jednotku Slave. Posílání zprávy do vyšší vrstvy by nemělo smysl, jelikož stejně nemůže být nadále vyhodnocována. Master chybnou zprávu odesílá do aplikační vrstvy, jelikož zde je vrstva složitější, a ukládá si chybné zprávy pouze pro účely logování – záznamy o činnosti programu a komunikace.

Spojová vrstva má také na starost adresování a to jak kontrolování příchozí adresy, tak i její přidávání ke zprávě. Všechny jednotky jsou připojeny na jednu sběrnici, a komunikují pouze s Masterem. Ten ovšem nemá ponětí, od jakého Slave modulu zpráva přišla, jelikož Slave zahájí komunikaci bez výzvy Mastera. Proto všechny zprávy obsahují i informaci o adrese Slave modulu. Právě tuto adresu vysílajícího Slave modulu zařazuje do zprávy Spojová vrstva komunikačního modelu. Co se týče kontroly adresy, tak ta je jednoduše kontrolována při přijetí adresového rámce. V případě, kdy adresy jsou shodné, nastaví nižší (fyzickou) vrstvu tak, aby přijala i datové rámce. V tomto pohledu, adresování na straně příchozí, spojová vrstva plní úkol, jak jí předkládá Referenčním modelem ISO/OSI. Spojová vrstva také může přidělovat zprávě adresu, na kterou je zpráva posílána. Ovšem tenhle aspekt, na straně Master, není již zcela splněn. Adresa, na kterou je posílána zpráva, je již obsažena ve zprávě, která je odesílána. Jediné co spojová vrstva udělá, je to, že ji rozpozná, podle formátu zprávy, a nastaví vysílanému bytu devátý, adresový, bit do jedničky. Devátý bit značí pro fyzickou vrstvu, že se jedná o adresový rámec. Slave mikrokontrolér vždy komunikuje pouze s Master mikrokontrolérem, proto není třeba, aby aplikační vrstva rozhodovala, na kterou adresu má být zpráva vyslána. Slave mikrokontrolér tedy přiřazuje adresu až ve spojové vrstvě.

2.3.4 Aplikační vrstva

Jednoduchost komunikace umožnila spojit zbytek vrstev komunikačního modelu do aplikační vrstvy. Zprávy nejsou nijak kódovány, ani opatřeny dalšími daty, kterými by bylo třeba opatřit vysílanou zprávu. Přímo v aplikační vrstvě se vygeneruje z části předpřipravená zpráva, kterou je třeba vyslat po sběrnici, v závislosti na jejím účelu. Jak již bylo zmíněno v předchozí kapitole, spojová vrstva jen opatří zprávu o CRC8 a popřípadě i adresu (jedná-li se o Slave jednotku).

Podíváme-li se na chování aplikační vrstvy při příjmu zprávy, nejedná se o nic jiného, než postupného analyzování byte po bytu v jedné zprávě, až zpráva postupně probublá ke správné funkci a zbytek zprávy je (většinou) použit pouze už jako jednotlivé parametry pro danou vykonávanou funkci. Pokud se objeví ve zprávě byte, který neodkazuje na žádnou funkci, tak je zpráva prostě ignorována. Na tuto analýzu se můžeme podívat i jako na filtrování zpráv.

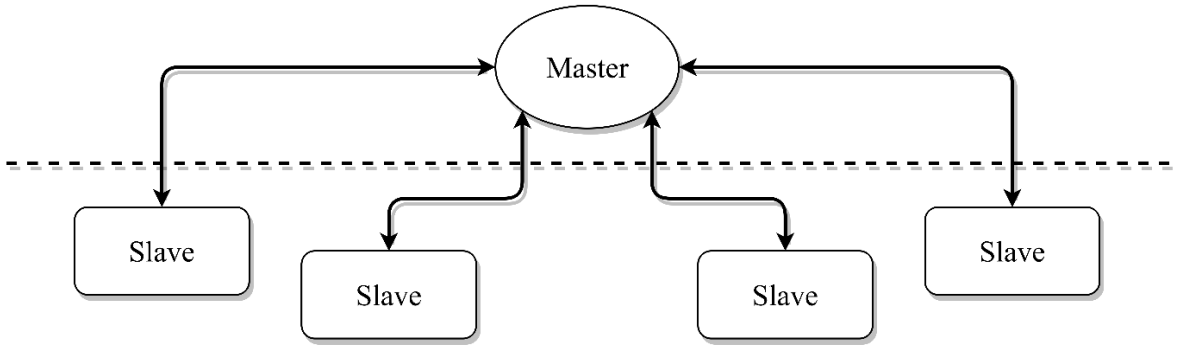
Stane-li se taková chyba v komunikaci, že by náhodou pro špatně přijatou zprávu byl spočítán stejný CRC8 kód, což nenabývá vysoké pravděpodobnosti, je možné ještě doufat v to, že neprojde analýzou (filtrováním) zprávy. Kdyby nastala stejná chyba, ale pouze pro byte, který je považován za argument funkce, filtrování by nijak nepomohlo a například světlo by se mohlo místo rozsvícení na úroveň 10 % rozsvítit na 25 %.

2.4 Komunikační protokol

Komunikační protokol zahrnuje v sobě všechny potřebné informace o způsobu komunikace. Jedná se tedy o specifikaci syntaxe a význam jednotlivých zpráv. V této podkapitole si postupně projdeme způsob výměny informací mezi jednotlivými moduly.

2.4.1 Koncept komunikace

Jak již bývá zvykem, a ne jen v komunikacích, máme nadřízené a podřízené jednotky. V případě naší popisované sítě máme pouze jeden nadřízený modul, který se stará o celou komunikaci, centralizované ovládání a zároveň propojení s dalšími systémy. Komunikace probíhá pouze mezi Masterem a jedním z modulů Slave, nikdy se nemůže stát, že by komunikoval Slave spolu se Slave modulem, a to i z fyzických důvodů. Pro jednoduchost je zde uveden obrázek hierarchie z pohledu komunikace, viz obr. 9, nejedná se o topologii sítě!



obr. 9: Hierarchie komunikace

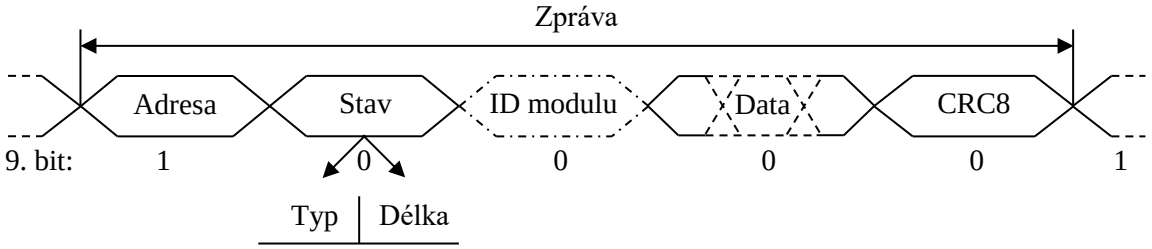
Na obr. 9 vidíme, že komunikace mezi Masterem a Slave modulem je vždy oddělená a přímá. Hierarchie komunikace je pouze jednoúrovňová, není tedy možné, aby se modul Slave stal prostředníkem pro přeposlání zprávy. Tato skutečnost naopak vychází z topologie sítě, která je sběrnicová (viz obr. 7) – všechny Slave moduly jsou v přímém spojení s Masterem.

Iniciace komunikace není závislá pouze na Masterovi. Slave může zahájit komunikaci bez výzvy mastera. Děje se tomu tak pro snížení latence. Kdyby Slave mohl vyslat informace pouze až po vyžádání od Mastera, který může mít připojených až 255 Slave modulů, tak maximální doba latence by mohla být již pocítěna. Už jen při stisku tlačítka pro rozsvícení světla je potřeba, aby se světlo okamžitě rozsvítilo. Ovšem dobu latence způsobenou vyzíváním Slave modulů k poslání dat si můžeme přibližně vypočítat. Vezměme v úvahu 143 tisíc bitů za sekundu. Jeden rámec se skládá z jednoho start bitu, devíti datových bitů a jednoho stop bitu. Máme tedy 11 bitů na jeden rámec. Pošleme tedy 13 tisíc rámců za sekundu. Kdyby Master postupně vyzíval jednotlivé Slave moduly k poslání informací, potřeboval by vyslat alespoň dva rámce, adresa a data znamenající výzvu komunikace, Slave by také musel odpovědět alespoň dvěma rámci. Dobu vyhodnocení zanedbáme a budeme brát 4 rámce jako jeden dotaz. Počet dotazů za jednu sekundu by bylo přibližně 3 tisíce (zaokrouhleno z 13/4). Vezmeme v potaz nejhorší případ a to 255 Slave jednotek. Přibližný čas čekání by mohl být až 85 ms. Tato doba by nebyla tolik omezující, ovšem připočítává se do celkového zpoždění. Ale co je určitě nepřehlédnutelné, je spotřeba takto fungujícího systému. Jelikož by komunikace probíhala neustále, spotřeba by mnohonásobně vzrostla, což je absolutně nežádoucí. Hlavně kvůli aspektu spotřeby energie nebyl volen tento způsob iniciace komunikace Masterem.

Nevýhodou započetí komunikace bez výzvy Masterem s sebou přináší problém možnosti současného vysílání zprávy dvou a více Slave modulů. Tomuto jevu se někdy také říká v anglické literatuře *Crosstalk* (překládaný jako přeslech). Slovo *Crosstalk* se vykládá v závislosti na použití v daném oboru. Jeden z výkladů, v případě bezdrátové komunikace, znamená právě současné vysílání dvou vysílačů na jednom kanálu, což popisuje naši problematiku, jen se nejedná o bezdrátový přenos. Ať si tento problém pojmenujeme jakkoliv, je potřeba najít řešení. Jelikož obě zprávy v případě *Crosstalk* budou ztraceny, nepřijde tedy ACK, je potřeba je znovu vyslat. Tyto zprávy nesmí být znovu vyslány se stejnou periodou, jinak by neustále docházelo k chybě přenosu. Aby se tomu tak nedělo, je využito generování pseudonáhodné doby pro další vyslání. Téměř na nulu se sníží pravděpodobnost vyslání zpráv ve stejný čas.

2.4.2 Formát zpráv

Formát zprávy definuje, význam jednotlivých bytů ve zprávě. Díky formátu zpráv víme, jak máme zprávy analyzovat, sestavovat a jak jim vlastně rozumět. Jedná se tedy o jazyk, jakým si mezi sebou jednotlivé moduly vyměňují informace. Bez znalosti tohoto formátu jsou příchozí zprávy bezcenné. Můžeme říct, že čím složitější je formát zpráv, a čím více má výjimek, tím je složitější pro vyhodnocování. Většinou čím je složitější formát zpráv, tím má více výjimek a tím je i složitější vymyslet, jak správně vytvořit zprávu. Po celou dobu je potřeba udržet jednoduchý, ale efektivní koncept celého projektu, proto bude tato myšlenka aplikována i ve tvoření formátu zpráv. Grafická interpretace formátu zprávy je na obr. 10.



obr. 10: Formát zprávy

Formát zprávy, jak zobrazuje obr. 10, začíná *adresovým* rámcem, který je odlišný devátým bitem. Všechny ostatní rámce jsou již datové, proto jsou i deváté bity nulové. Jelikož adresa je jednobytová, můžeme mít tedy až 256 jednotek v jedné síti, které je možno adresovat na jedné sběrnici. Tento počet je dostačující už jen proto, že více jednotek na jedné sběrnici nám nepovoluje ani standard RS-485, respektive linkové převodníky na diferenciální signály.

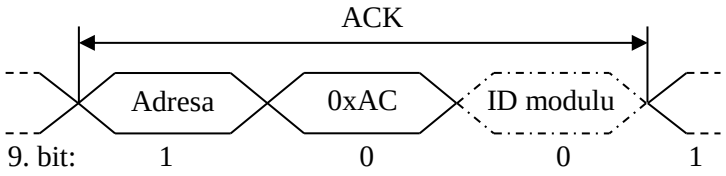
Druhým rámcem v pořadí je *stavový* rámec, byl tak nazván, protože vypovídá o stavu zprávy – jejího typu a délky. Stav je tedy byte rozdělený na dva nibbly. První nibble vypovídá o typu zprávy, který je závislý na komponentě, s níž se komunikuje. Nejedná se tedy o ustálené označení typů zprávy. Většinou typ zprávy odpovídá vyvolávané funkci v dané komponentě. Neplatí to vždy, jelikož bylo snahou odesílat co nejkratší zprávy s co nejvíce informacemi, neboli zprávy s co nejvyšší hustotou informace. Může se tedy stát, že někdy 16 typů zprávy nestačí a je potřeba vytvořit další typy, ale současně nechceme zvyšovat typový nibble na typový byte. Není tedy problém v náročnější komponentě použít jeden typový nibble ke dvěma zprávám, které budou mít rozdílnou délku zprávy. Pro představu mějme příklad, kdy můžeme jedním modulem rozsvěcovat šest stmívatelných světel. Chtěli bychom vytvořit nový typ zprávy, který by rozsvěcoval místo jednoho světla zároveň světla dvě. Řekněme, že už všech 16 typů zprávy je vyčerpaných a přesto bychom chtěli přidat popsaný typ. Je možné využít typ zprávy, který rozsvěcuje jedno světlo a rozšířit ho o další třídění. Bude-li zpráva obsahovat informaci o pozici světla a intenzitě, bude nejspíše dlouhá 2 byty. Přidá se tedy při vyhodnocování tohoto typu zprávy i rozhodování podle délky zprávy, která bude dlouhá již 4 byty (pozice 1. světla a jeho intenzita (2 byty) + pozice 2. světla a jeho intenzita (2 byty)). Tímto úkonem byla zvýšena náročnost pro vyhodnocení příchozí zprávy, ale byl zachován formát zprávy s dobrou hustotou informace. Druhým nibblem ve stavovém bytu je informace o délce zprávy. Není to přesně celková délka zprávy, ale počet datových bytů, bezprostředně následujících za stavovým bytem. CRC8 kód se do toho čísla již nezahrnuje.

ID modulu je identické s adresou modulu. Použití čerchované čáry je záměrné, jelikož tento rámec je povinný pouze pro Slave modul. V případě kdy se jedná o zprávu vysílanou Slave modulem, tak ID modulu je přiloženo do zprávy, jelikož je potřeba aby přijímací Master modul věděl, který Slave modul mu odeslal zprávu. Jak již bylo zmíněno, zprávy nejsou odesílány až po požadavku od Mastera, ale v době, kdy je potřeba zprávu odeslat. Tím se zkrátí latence komunikace, ale přichází s tímto přístupem další problémy. V případě posílání zprávy Masterem směrem k modulu Slave, není potřeba zanášet informaci o adrese Mastera, jelikož je jediný a od nikoho jiného nemůže Slave modulu přijít zpráva. Formát zprávy je tedy odlišný v závislosti na odesílateli.

Poslední byte je *CRC8* kód. Jak již bylo dříve naznačeno, jedná se cyklický redundantní součet, který pomáhá odhalovat chyby v přenosu zpráv. Jeho hodnota je vždy počítána z celé zprávy, tedy z adresového rámce, stavového bytu a datových bitů. Při přijetí zprávy je CRC8 spočítaný komponentou, která zprávu přijala a porovnaný s přijatým CRC8 kódem. V případě shody je ověřeno, na určité hladině spolehlivosti, že zpráva byla přijata správně. Jak již bylo dříve předesláno, je důležité, aby byl tento SID spolehlivý, proto při správném přijetí zprávy je komponentou, které tato zpráva přišla a byla komponentou označena za ověřenou, odeslána potvrzovací zpráva. Bližší informace o tomto kódu se dozvíte v jedné z následujících podkapitol.

2.4.3 Formát ACK

Zkratka ACK vychází z anglického slova *acknowledgement*, což v souvislosti s komunikací znamená potvrzení. Zde mluvíme o potvrzení o přijetí zprávy. Jedná se tedy o zpětnou vazbu pro vysílací modul, který ví, že zpráva v pořádku dorazila. V případě, kdyby vysílací modul nedostal tuto zpětnou vazbu, komunikace by nebyla natolik spolehlivá vzhledem k možnosti ztrátě dat, což je nepřípustné. Formát je graficky zobrazen na následujícím obrázku (obr. 11).



obr. 11: Formát ACK

Každá zpráva začíná *adresovým rámcem*, aby bylo určeno, jaké jednotce má přijít potvrzení o přijetí. Druhý rámec nastavuje statickou hodnotu, a to $(AC)_{16}$ (v šestnáctkové soustavě), která přijímací jednotce ulehčuje rozpoznání ACK. Jelikož je toto číslo na stejné pozici jako pozice stavového rámce ve formátu zprávy (viz obr. 10), je nutné se vyhnout použití zprávy typu $(A)_{16}$ a zároveň délky zprávy 12 znaků. Jelikož stavový rámec není generován automaticky, ale pevně dán již při návrhu jednotlivých zpráv pro daný modul, musí na něj brát ohled autor zpráv. Ovšem nikdo není neomylný, a může se stát, že by autor přesto vytvořil zprávu, která je typu $(A)_{16}$ o délce 12 znaků. Slave modul totiž testuje, zda se jedná o ACK zprávu jen v případě, když očekává ACK. V opačném případě zpráva není testována na konstantu ve spojové vrstvě a je rovnou, po ověření shody CRC8, předána aplikační vrstvě.

Posledním rámcem v ACK zprávě je *ID modulu*, které je identické s adresou modulu. Znovu je naznačeno čerchovaně, jelikož jeho používání je rozdílné pro zprávy posílané Masterem a zprávy posílanými Slave moduly. Master je pouze jeden, tudíž není třeba, aby bylo posíláno jeho ID, jelikož každý Slave modul ví, že zprávy jim mohou být poslány pouze Master modulem. ID modulu je tedy přidáváno pouze v případě, když Slave posílá ACK Masterovi. Master si nikdy nedovolí poslat dvě zprávy za sebou jednomu Slave modulu, jelikož by nebyl schopný rozlišit, pro kterou zprávu dostal potvrzení o jejím korektním přijetí.

2.4.4 Cyklický redundantní součet CRC8

CRC8 je zkratkou anglického výrazu *Cyclic Redundancy Check*, do češtiny překládaný jako cyklický redundantní součet. Osmička na konci značí, že se jedná o osmibitový kód. Tento kód slouží k detekci chyb na přenosu informací, ovšem chybu odhaluje jen s určitou pravděpodobností, která se zvyšuješ spolu s šířkou kódu. Velkou výhodou je jednoduchost výpočtu a zejména hardwarové implementace. Celá myšlenka je přidání ke zprávě vypočítaný součet podle určité formule. Na straně příjemce je spočítán CRC8 kód podle stejné formule a porovnám se stávajícím kódem ve zprávě. Jsou-li kódy shodné, je možné tvrdit, na určité hladině spolehlivosti, že přijatá zpráva byla přijata správně. Při neshodě kódů zprávu předpokládáme jako chybnou. V některých případech může CRC posloužit při opravě přijaté zprávy. Tuto vlastnost ovšem implementovaná komunikace nevyužívá, tudíž nebude dále rozebírána.

Základní myšlenkou CRC algoritmu je jednoduše považovat zprávu jako obrovské binární číslo, které vydělíme dalším konstantním binárním číslem, a zbytek po dělení považujeme za kontrolní součet. Dělení je ovšem velmi náročné a proto myšlenka vytvoření CRC kódu jde dál, jehož vysvětlení je zdlouhavé a je vysvětleno v literatuře [6]. My se podíváme na praktickou ukázku jednoduchého výpočtu CRC kódu v jazyce C inspirován kódy v literatuře [6].

Na počátku kódu se definuje polynom, kterým je děleno binární číslo, reprezentující celou zprávu. Podíváme-li se blíže na kód popsaný v jazyku C(99), žádné přímé dělení tam nevidíme. Je to tím, že přímá operace dělení by byla příliš náročná, navíc nás zajímá pouze zbytek po dělení. Složitá operace dělení je zde přepsána pouze na operaci XOR spolu s bitovým posunem. Operace XOR se provádí pouze v případě, kdy je MSB bit *crc* proměnné (mezivýsledku) roven jedničce. V případě, kdy není, tak nenastane operace XOR a mezivýsledek *crc* se pouze posune o jeden bit doleva. Tento postup je prováděný až do posledního bitu celé zprávy. Na konci těchto cyklů (první procházející byty ve zprávě, druhý bity v bytu) se vrací číslo, odpovídající CRC8 kódu. Také si můžeme povšimnout inicializace proměnné *crc* na počátku výpočtu CRC. Tato inicializace na určitou konstantu není povinná, může být nulová, potom první operace XOR neudělá nic jiného, než pouze přiřadí do proměnné *crc* první byte zprávy. Nastavíme-li inicializaci proměnné *crc*, musí tomu tak být pro všechny jednotky, aby došly ke stejnému číslu při výpočtu CRC kódu.

```
#define CRC8POLY 0x16 //0x16 ~ x^4 + x^2 + x (polynomial)
#define CRCINIT 0x2A

uint8_t crc8(uint8_t *message, uint8_t length) {
     uint8_t crc = CRCINIT;
for (uint8_t i = 0; i < length; i++) {
           crc ^= message[i];
           for (uint8_t n = 0; n < 8; n++) {
                if (crc & 0x80)
                     crc = (crc << 1) ^ CRC8POLY;
                else
                crc << 1;
           }
     }
     return crc;
}
```

3 Moduly inteligentní domácnosti

Právě čtená kapitola nabízí pohled na hardwarovou realizaci modulů inteligentní domácnosti. Nejprve bude představen základní koncept opírající se o návrh a realizaci jednotlivých modulů. V předchozí kapitole bylo definováno, jak si budou moduly vyměňovat informace, navrhneme tedy připojení komunikačních kabelů a převod signálů srozumitelných pro mikrokontrolér.

3.1 Základní koncept

Moduly můžeme rozdělit do dvou skupin z pohledu zpracování informací. První skupinu označíme senzory, které budou do naší sítě přivádět informace o dění v domácnosti. Aktuátory budou patřit do skupiny druhé, jež budou vykonávat definovanou činnost na základě vyhodnocení posbíraných dat ze senzorů. Veškeré moduly budou mít některé rysy společné, jako je napětí pro mikrokontrolér ATmega 5 V, linkové převodníky z diferenčního výstupu na napětí proti zemi pro komunikaci, možnost přeprogramování modulu na vyhrazených pinech.

Napájení modulů je zajištěno po sběrnici, která tedy zajišťuje jak přenos komunikace, tak i energie. Ovšem není možné na jedné straně napájet komunikaci pěti volty a na druhé straně, která může být vzdálena až jeden kilometr odebírat proud. I v případě, kdyby proud nebyl nikterak vysoký, docházelo by ke ztrátám napětí na dlouhém vedení. Tento problém byl vyřešen elegantně, a to tak, že moduly byly rozděleny z pohledu napájení na aktivní a pasivní. Aktivní moduly jsou takové, jež jsou schopny dodávat do sběrnice napětí. Jedná se zejména o moduly, které ovládají LED osvětlení. Využívají potřebné napětí na rozsvícení LED pásků, tedy 12 V, jež je pomocí DC-DC měničů sníženo na pět voltů. Těchto převedených 5 V slouží jak pro napájení modulu, tak i pro napájení sběrnice. Pasivní moduly jsou takové, které pouze odebírají proud ze sítě, ovšem není nikterak významný. Obecně veškeré moduly jsou navrhovány na co nejnižší spotřebu, což platí o celém systému. V případě, kdy by bylo málo bodů s aktivními moduly, není problémem přidat napájecí bod sběrnice, který by byl řešen zdrojem například do DIN lišty (nosná lišta ve tvaru U).

Mikrokontrolér, používaný pro většinu modulů, není nárokován na vysoký výpočetní výkon, proto byl využit základní mikrokontrolér ATmega8 taktovaný na frekvenci 8 MHz. Tato frekvence je pro potřeby některých modulů zbytečně příliš vysoká, ovšem není možné ji snížit kvůli komunikaci. Komunikace je nastavena na relativně vysokou rychlost, a to 143 kbps. Mimo jiné je také potřeba rychle spočítat CRC kód, aby se zbytečně nezdržovala komunikace s modulem čekáním na ACK. Samotné moduly jsou navrhovány také modulově, nebo lépe řečeno po blocích. Do všech modulů je přidán blok komunikace, který je povětšinou stejný. Dalším stavebním blokem modulu je mikrokontrolér a jeho obvodové zapojení, potřebné pro jeho funkci, jedná se o pasivní součástky, jako jsou kapacity, krystal a rezistory. Opakovaným obvodovým řešením jsou i ochranné prvky, jakou jsou polykrystalické pojistky na straně napájení, aby při poruše jednoho modulu se nestalo to, že by způsobil disfunkci celé sítě.

3.1.1 Uživatelské vstupy

Čím více přístupových bodů s různými možnostmi SID nabízí, tím jednodušší je ovládání ze strany uživatele, jelikož si každý může vybrat pro něj to nejlepší. Jednotlivé přístupové body by měly umožňovat co nejširší možnosti ovládání bytu, ale současně by neměly ztrácet jednoduchost a přehlednost ovládání. Co si lze představit pod pojmem přístupový bod? V této práci je přístupovým bodem pro ovládání SID myšlen ovládací prvek daného druhu, kterým je možno ovládat danou část systému, či systém celý. Příkladem jsou vypínače na zdi, dotykové ovládací panely na zdi, mobilní aplikace, počítačová aplikace, webové rozhraní a jiné. V následujících řádcích jsou uvedeny uvažované možnosti uživatelských vstupů.

Vypínače na zdi, či *tlačítka na zdi*, patří ke konvenčním ovládacím prvkům domácnosti, a to jak s obyčejnou elektroinstalací, tak i se SID. Jedná se o natolik standardní ovládací prvek, že již není možné ho vyřadit navrhovaným systémem, protože by každému uživateli chyběl. Snad jedině velmi liberální fanoušek inteligentních domácností by upřednostnil pouze alternativní ovládací prvky. Navrhovaný systém zcela zajisté bude pracovat s tlačítky na stěně. Výhodnější je ovšem použití tlačítek (jedna stabilní poloha), nežli využití spínačů (dvě stabilní polohy). Výhodou je možnost přidání funkcionality tohoto tlačítka, kdy můžeme rozlišit krátké, či dlouhé stlačení. U spínače by to bylo také možné, ovšem s menším komfortem ovládání pro uživatele. Další funkcionalitu by bylo možné zvýšit kombinací stisknutých tlačítek zároveň, ovšem je potřeba, aby nebyla ztracena jednoduchost systému. Představa toho, že by uživatel stál u tlačítek a přemýšlel, pod jakou kombinací tlačítek například vypne všechna světla v místnosti, již většinu uživatelů opravdu vyděsí. Vždy je potřeba se zamyslet nad použitelností a opravdovou využitelností aplikace či funkce.

Na zeď můžeme také umístit *dotykový panel* pro ovládání inteligentní domácnosti, jehož výhodou je využití piktogramů a tím zvýšit přehlednost ovládání. Nejedná se pouze o piktogramy, které je ostatně možno použít i na tlačítka, ale hlavně o dotykové posuvníky (angl. slider). Další výhodou je možnost podsvícení ovládacího panelu, nejlépe skleněného, tak, aby při jemném dotyku se rozsvítil a bylo možné vidět jednotlivé piktogramy i ve tmě.

Krokem výš od dotykových panelů jsou *dotykové obrazovky*. Trh nabízí inteligentní dotykové obrazovky, k nimž se dodává i program, ve kterém je možné si navrhnout motiv obrazovky. Každému grafickému prvku na inteligentním displeji je možno přiřadit kód, který je následně, při dotyku obrazovky v místě symbolu, poslán po sběrnici do mikrokontroléru umístěného za tímto displejem. Mezi výhody toho ovládání je příjemný grafický výstup, snadno měnitelný, může v sobě zahrnovat i přehled o teplotách v domácnosti a jiné. Jednodušeji řečeno výhodou je jeho univerzálnost. Nevýhodou je opět jeho vyšší spotřeba, kterou ovšem můžeme omezit vypínání této dotykové obrazovky, která by se aktivovala až při stisku.

Luxusnější variantou je umístění *dokovací stanice pro tablet* přímo na stěnu. V tomto tabletu, u něhož není potřeba vysokého výpočetního výkonu, je nahrána aplikace pro ovládání a přehled inteligentního domu. Výhodou tabletu je jeho mobilita. Mezi nevýhody je možno zařadit vyšší spotřebu tabletu, vyšší latenci reakce na podnět a nižší viditelnost obrazu na displeji na přímém slunci. Také pořizovací cena není zanedbatelná. Navíc při častém odpojování a připojování tabletu do nástěnné dokovací stanice bude baterka namáhána častým nabíjením a vybíjením, čímž se výrazně sníží její životnost.

Posledním jmenovaným hardwarovým prvkem je *dálkové ovládání*. Tímto prvkem již narážíme na ovládání bezdrátovým modulem, nepočítáme-li tablet, který komunikuje přes Wi-Fi, ovšem ze strany konstruktéra toho systému je bezdrátová komunikace vyřešena. Tento modul by bylo výhodné realizovat tak, aby dálkové ovládání bylo přenositelné mezi jednotlivými místnostmi a ovládalo místnost, ve které uživatel dálkové ovládání používá. Realizace toho dálkového ovládání nebude součástí realizace této diplomové práce, ovšem systém bude navržen tak, že umožňoval implementaci tohoto dálkového ovládání.

Webové rozhraní nabízí uživateli nejvíce funkcí a nastavení systému inteligentního domu. Pro jednoduchost je potřeba rozlišit dva typy zobrazení. První zobrazení je pro každodenní použití, v němž se jednoduše rozsvítí světlo, zapne ventilátor, zatáhnou rolety a další jednoduché úkony. Druhým zobrazením je pak podrobnější nastavení inteligentní domácnosti, jako může být kupříkladu teplotní profil vytápění, či přehled o stavu inteligentní domácnosti.

Mobilní či počítačová aplikace již může být považována za redundantní, jelikož webové rozhraní bude přístupno pro mobily (Smartphone), počítače, tablety a další prvky s přístupem na internet. Dále s webovým rozhraním není potřeba řešit kompatibilitu pro každý operační systém zvlášť. I když je možné program vytvořit v multiplatformním jazyku, jako může být například Java, tak je zde neustále potřeba upravit grafické výstupy pro různé platformy a velikosti displejů. Byly vyjmenovány pouze nevýhody využití aplikace, ale jsou zde i výhody, jako je odstranění potřeby neustále načítání webového rozhraní pro aplikaci, a pro malé rychlosti internetu dlouhé odezvy v přecházení mezi jednotlivými menu. Pro nevýhody těchto aplikací nebudou s největší pravděpodobností realizovány.

3.1.2 Prvky ovládané systémem inteligentní domácnosti

Tato kapitola nastíní moduly, které ovládají v inteligentním domu prvky zvyšující komfort, snižují energetickou spotřebu, zajišťují bezpečnost a jiné. Většina uživatelů má podobné požadavky na prvky, které mají být implementovány do SID, ovšem jsou zde i požadavky, které nezapadají do skupiny standardních prvků. Dále budou jmenovány standardní prvky spolu s náznaky na okruh prvků nestandardních.

Jako první jmenovaný prvek snad s nejvyšší prioritou je *osvětlení*. Nejspíše nebude mít nikdo zájem o to, aby jeho domácnost byla automatizována, ale nemohl na dálku vypnout světla, či se pouze ujistit, zda vypnutá jsou. Výhodou je možnost stmívání světel (přizpůsobených k tomuto jevu), díky níž je možné si pouze přisvítit, když nestačí sluneční svit pronikající do domu. Nebo když večer vstanete a nechcete naplno rozsvítit místnost, jen podržíte tlačítko a světlo se začne postupně rozsvěcovat, při uvolnění tlačítka zůstane světlo mírně rozsvíceno na zvolené hodnotě a vy nejste oslepeni silný světlem. Další výhodou stmívatelného světla je pouze přisvícení při sledování televize, nebo dokreslení světelné scény. V dnešní době se s výhodou využívají LED pásky, jejichž svit je jednoduše regulovatelný PWM signálem. Nabízí se tedy vytvoření modulu, který generuje výkonový PWM signál na základě poslané zprávy od Mastera. Další možností modulu generující výkonové PWM signály je možnost ovládání barvy RGB LED pásků, popřípadě LED pásků, jež nabízí ovládání teplotu barvy bílého světla. Při využití zdrojů světla se setrvačností dosvitu, jako jsou kupříkladu žárovky, se naopak využije regulace tyristorem, která bude do žárovky dodávat poměrnou část výkonu ze sítě v závislosti na úhlu otevření α tyristoru. Nakonec není vždy ani potřeba ovládání intenzity světla, a potom se nabízí využití pouze spínání síťového napětí přes relé. Relé modul je s výhodou možno použít i při ovládání ventilátorů.

Spínání silnoproudých obvodů pomocí *relé modulů* rozšíří možnosti automatizace inteligentního domu. Již byla zmíněna možnost rozsvěcování světel připnutím síťového napětí, či zapínání ventilátorů. Výhodou použití spínání ventilátorů pomocí relé je možnost časování, za jak dlouho bude ventilátor vypnut, tak i zjišťování jeho stavu – zda je sepnut časovaně, či je sepnut na dobu, než se rozhodne uživatel jej vypnout. Relé moduly také mohou aktivovat a deaktivovat síťové okruhy, tedy přivádět napětí do jednotlivých okruhů elektrických zásuvek a tím mít jistotu neaktivitu všech spotřebičů, které jsou připojeny do daného síťového okruhu. Dále se také mohou postarat o přivedení napětí do motorů, které pohání rolety, žaluzie, závěsy a jiné.

Ovládání elektronických termohlavic je příhodné zejména v přítomnosti ústředního topení, kde pomocí termohlavice je ovlivněn průtok horké vody a tím teplota topení. Na trhu jsou (hrubě děleno) dva typy těchto hlavic, některé hlavice jsou ovládány síťovým napětím (mohou být i jiná) a mají pouze dvě polohy, zcela zavřeno a naplno otevřeno. Druhé hlavice jsou ovládány napětím (např. 0 - 10 V), jež jsou otevřeny dle úrovně přiváděného napětí.

Modul pro řízení kotlů s výhodou využívá možnosti jejich externího řízení. Varianty řízení kotlů z externích zdrojů jsou různé. Jeden z příkladů je ovládání úrovně výkonu vytápějícího kotle pomocí úrovně napětí například od nuly do dvaceti čtyř voltů.

Modul pro otevírání dveří se může postarat o zvýšení pohodlí, například při potřebě otevřít dveře návštěvě vcházející do domu na dálku. Také implementace otevírání dveří na kartu je mnohem snazší s tímto modulem. Modul se může upravit i tak, aby bylo možno s ním otevírat i garážová vrata a jiné.

Obecně lze navrhnout modul pro zařízení, které je uzpůsobeno k externímu řízení, stačí znát specifikace pro jeho řízení a následně už není natolik složité navržení řízení. Jsou i zařízení, která umožňují komunikaci i pomocí speciálních sběrnic, ovšem v dnešní době je snahou ovládat moderní zařízení prostřednictvím intranetu (viz kapitola 1.2.4). V případě možnosti ovládání zařízení pomocí intranetu se již o jeho management stará čistě software umístěný na serveru, nejlépe implementovaný do řídícího programu pro celou inteligentní domácnost.

3.1.3 Implementované snímače

Každý správně navržený snímač by měl do systému inteligentní domácnosti přinést hodnotnou informaci, díky níž je možné zlepšit a zvýšit možnosti automatizování. V této podkapitole budou uvedeny ty snímače, které korespondují s moduly popsanými v předchozích dvou podkapitolách: 3.1.1 a 3.1.2.

Snímání intenzity osvětlení umožňuje zlepšit světelný management tím, že uživatel nenastaví hodnotu intenzity světla například LED pásku, ale nastaví, kolik luxů bude chtít v pokoji. Tento přístup zvýší komfort zejména v případě, kdy venkovní světlo často mění intenzitu, hlavně při oblačném počasí, nebo k večeru, kdy zapadá Slunce. Samozřejmě aby bylo možné využít tohoto snímače, je potřeba mít k dispozici světla s ovladatelnou intenzitou světla.

Snímače pohybu mohou být také využity při řízení osvětlení. Buď se bude jednat o jednoduché využití snímání pohybu pro světlo například na záchodě, nebo sofistikovanější počítání osob v dané místnosti tím, že se zjišťuje počet lidí prošlých rámem dveří a směr jejich chůze.

Teploměr a vlhkoměr se podílejí na řízení teploty v místnosti. Ač se to zprvu nezdá, je možné využít i vlhkoměr, jelikož vlhkost ovlivňuje naše vnímání teploty. Díky této kombinaci je možné si nastavit ne absolutní teplotu v pokoji, ale pocitovou teplotu, která je počítána z absolutní hodnoty teploty a relativní vlhkosti. V případě, kdy uživatel chce nastavit absolutní teplotu v pokoji, údaj z vlhkoměru bude mít pouze informativní účel.

Pro bezpečnost může sloužit *senzor plynů*, který je vhodný v případě, kdy je do bytu zaveden plyn a pomocí tohoto senzoru je možno měřit, zda plyn neuniká. V případě, kdy senzor umí vyhodnotit i kouř, pak se může jednat o ochranný požárnický prvek.

Senzor hladiny vody může ujistit uživatele domu, že neuniká voda z pračky, jiného zařízení, či potrubí, v době, kdy je uživatel mimo domov, kupříkladu na dovolené.

Kamery můžeme označit jako optické senzory a mohou mít hned několik funkcí. Mohou zastupovat prvek bezpečnostní, kdy v případě zaznamenaného pohybu je kamera spuštěna a je možno ukládat videozáznam na server. Při zazvonění na domovní zvonek může uživatel pomocí online přenosu záznamu z kamery umístěné před hlavními dveřmi zjistit, zda se jedná o očekávanou návštěvu, a na dálku otevřít dveře. Nabízí se také možnost sledování osob v domácnosti a na základě získaných informací ovládat osvětlení v jednotlivých místnostech. Ovšem popsaná funkce je složitějšího charakteru a při prokázání dostatečné funkčnosti senzorů pohybu nebude tato funkce uvažována. Navíc by tato funkce vyžadovala kameru v každé místnosti, čímž by se navyšovala pořizovací cena systému.

Snímače polohy dveří a oken se uplatní v bezpečnostním managementu, ale stejně tak dobře i v řízení vytápění domu. Snímání polohy oken či dveří není potřebné, aby bylo kontinuální, ale postačí binární úrovně, tedy stav otevřeno a zavřeno. Tato informace je vhodná i pro přehled uživatele domu. Určitě každý zná to, že vyjde z bytu a přemýšlí, zda všechna okna jsou zavřená (o dveřích by již nemělo být pochyb, ale informaci tak i tak zjistí z přehledu v systému). I na tento problém myslí inteligentní domácnost a umožňuje tak zobrazení stavů všech dveří a oken.

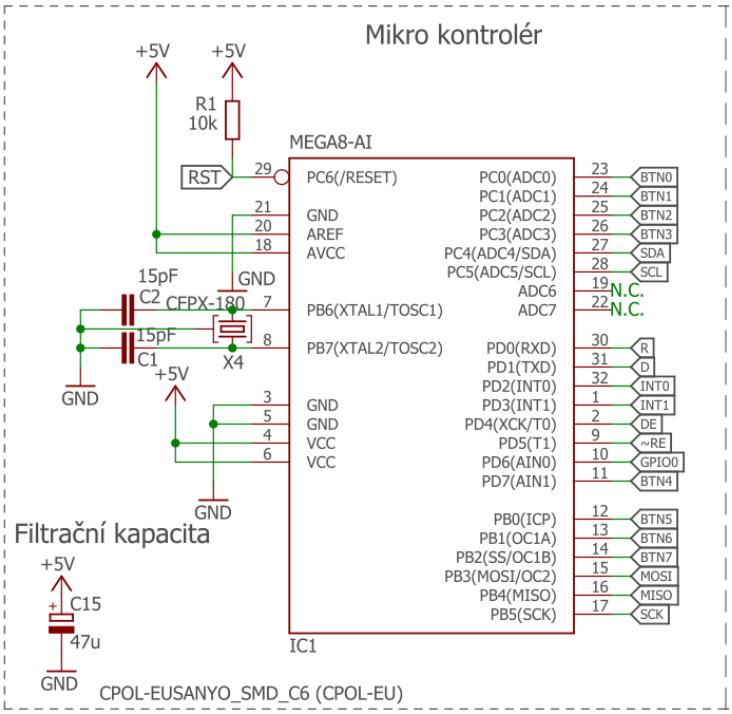
3.2 Podpůrné obvody

Většina následujících modulů se skládá z funkčních bloků jednoduchých obvodů, z nichž se některé opakují. Jedná se zejména o obvod s mikrokontrolérem ATmega8, rozhraní pro komunikaci po sběrnici a vyvedení programovacích pinů.

3.2.1 Mikrokontrolér ATmega

Mozkem všech modulů je mikrokontrolér (µPC) ATmega8 či ATmega328. Výhodou použití těchto dvou µPC je ta, že mají stejné pouzdro a stejné rozložení pinů. ATmega328 je novější a má tedy lepší parametry. Záleží tedy již na daném modulu, který z těchto µPC je nutný pro použití. Většinou moduly nejsou nijak náročné na výpočetní výkon, a proto dostačuje ATmega8.

Na obr. 12 je schéma zapojení µPC, které vychází z doporučení od výrobce v datasheetu [5]. Zapojení disponuje krystalem, jehož nominální frekvence je 16 Mhz. Kondenzátory, přímo připojeny ke krystalu, slouží k tlumení kmitů. Jejich hodnota vychází z doporučení výše zmiňovaného datasheetu. Rezistor R_1 funguje jako pull-up rezistor, který udržuje na invertovaném vstupu resetu v logické jedničce, tedy zajišťuje µPC v normálním běhu.



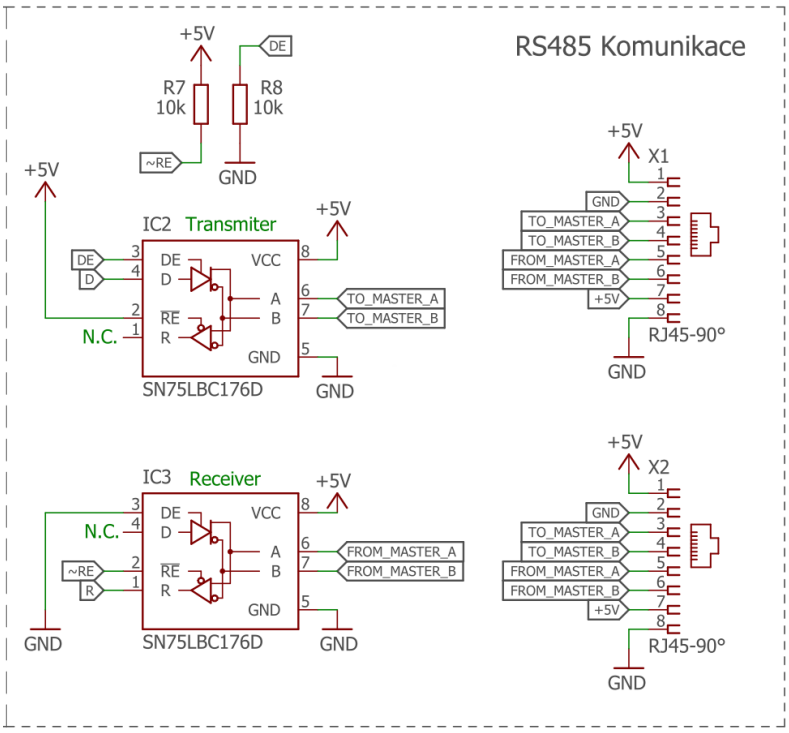
obr. 12: Zapojení mikrokontroléru

Filtrovací kondenzátor, krčící se v levém dolním rohu, slouží pouze pro vyhlazení napájecího napětí v případě kolísání napětí na sběrnici. Na první pohled chybí blokovací kondenzátory, které eliminují pronikání rušení na vyšších frekvencích vznikající při normální funkci digitálních obvodů. Ve schématu jsou vždy zapojeny paralelně na jednom místě, je tomu tak, aby se nesnižovala přehlednost schématu. Je ovšem třeba dbát na správné umístění blokovacích kondenzátorů při rozmisťování součástek na DPS (**d**eska **p**lošných **s**pojů). Tento blok blokovacích kondenzátorů je k nalezení na dalším schématu, jelikož kondenzátory slouží nejen pro napájecí piny ATmegy.

3.2.2 Komunikační rozhraní

Celá síť komunikuje přes drátové rozhraní, tudíž každý modul musí disponovat komunikačním rozhraním. Obvod se skládá pouze ze dvou převodníků diferenčního napětí na 5V logickou úroveň, jak lze vidět na obr. 13. Použití dvou převodníků je odůvodněno potřebou duplexního režimu komunikace, ovšem tento převodník nabízí polo-duplexní režim, proto byly použity dva. Převodníky, které nabízely duplexní komunikaci, byly mnohem dražší. Na problém vyšší energetické náročnosti odpovídají vstupy, které umožňují aktivovat a deaktivovat jednotlivé převodníky uvnitř integrovaného obvodu.

Vždy je využita jedna polovina převodníků IC_2 a IC_3, proto mají jeden ze vstupů pro blokování přímo připojený na logickou úroveň, která deaktivuje polovinu obvodu. Jak grafická značka znázorňuje, diferenční vstupně-výstupní port je označen jako A a B, a proto je připojen ke konektoru RJ45 (X_1, X_2). Skrz tento konektor jsou moduly připojeny do sběrnice. Pull-up rezistor R_7 slouží pro deaktivaci druhé poloviny převodníku IC_1 v případě, že µPC je například v úsporném režimu. Rezistor R_8 má stejný význam jako rezistor R_7, pouze s tím rozdílem, že je pull-down a stará se o deaktivaci zbytku převodníku IC_2. Názvy nad oběma převodníky popisují, jakým směrem komunikují.

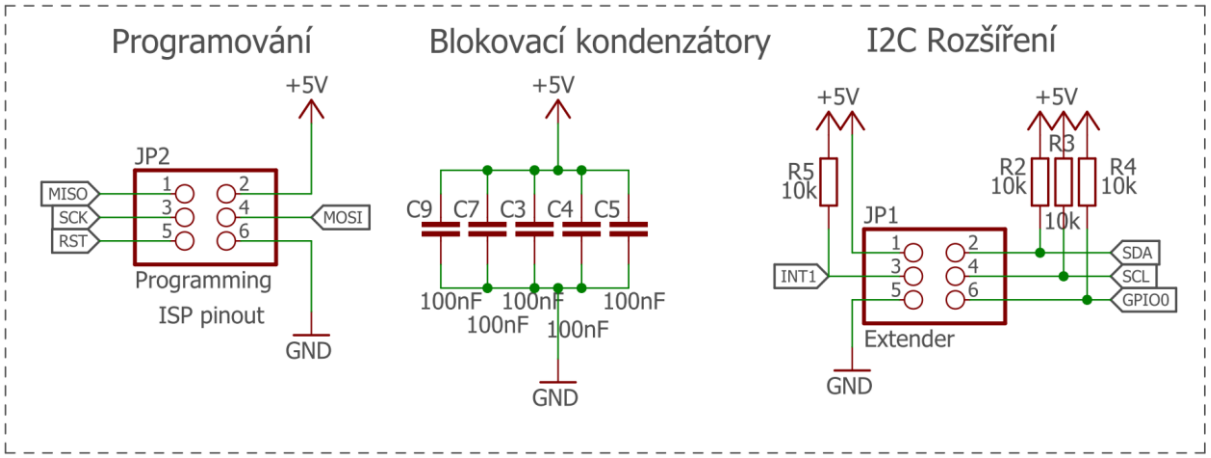


obr. 13: Obvod komunikace

3.2.3 Programování a rozšíření

Ke každému µPC musí vést cesta, jak jej naprogramovat, proto i každý modul obsahuje programovací port, který je navrhnut dle 6pin ISP (*In-System Programming*) pinů od AVR. Jedná se o programování přes SPI (*Serial Peripheral Interface Bus*) rozhraní, což lze s výhodou využít při komunikaci s rozšiřujícím modulem, který disponuje také tímto rozhraním. Pro případ, kdyby rozšiřující modul komunikoval pouze přes I^2C (*Inter-Integrated Circuit*) rozhraní, je tu připraven konektor s názvem *Extender*.

Již dříve byl zmiňován blok blokovacích kondenzátorů, který na první pohled ve schématu nemá žádný smysl. Vždy se volí takový počet kondenzátorů, kolik je napájecích pinů integrovaných obvodů, zejména ty, které pracují na vyšší frekvenci. Je tedy potřeba dávat pozor při rozmisťování součástek na DPS, takzvaně layout.



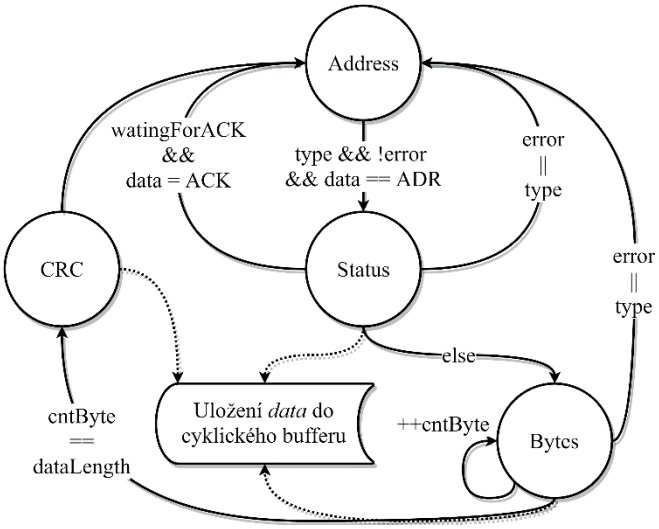
obr. 14: Programovací a rozšiřovací piny, blokovací kondenzátory

3.3 Podpůrná knihovna komunikace

Stejně jako předchozí podkapitola (3.2) představuje obvody, které se používají ve většině modulů, tato podkapitola představuje knihovny, které jsou použity v každém modulu. Veškeré moduly chovající se jako Slave modul, mají jednotnou knihovnu pro komunikaci. Modul Master musí komunikovat rozdílně oproti Slave modulům, a jelikož je jen jeden pro celou síť, bude tato knihovna pro komunikaci Master modulu představena přímo v podkapitole týkající se tohoto modulu.

3.3.1 Příjem zpráv

Prvním krokem pro přijímání zpráv je aktivování a správné nastavení UART komponenty v registrech µPC. Přijímání bytů je nastaveno tak, aby bylo vyvoláno přerušení při přijetí bytu. V registrech je možno přečíst, zda rámec zprávy přišel správně a data v zásobníku nepřetekla. Tyto chybové informace jsou uloženy v booleovské proměnné *error*. Druhou důležitou booleovskou proměnnou je *type*, ve které je uložená informace o typu rámce, který je buď datový či adresový. Příjem zpráv je v prvním kroku zajištěn stavovým automatem, který řídí a třídí příjem jednotlivých bytů. Mimo jiné tento stavový automat, graficky reprezentovaný na obr. 15, analyzuje příchozí byty a skládá z nich zprávy.



obr. 15: Statový automat přijímání zpráv

Stavový automat je tvořen čtyřmi stavy, mezi nimiž se přechází spolu s přerušením, jež signalizuje nově přijatý byte. V případě, kdy se jedná o zprávu, která přijde v pořádku, stavový automat prochází stavy v následujícím pořadí: *Address, Status, Bytes, CRC*. Postupně budou vysvětleny jednotlivé stavy a objasněny přechody mezi nimi, které jsou řízeny přerušením a zároveň uvedenou podmínkou.

1. *Address* – První a nejvíce používaný stav, v němž je povoleno přijímání jen adresových rámců, filtruje zprávy, které jsou určeny pouze pro daný modul. Jelikož každý modul má svoji adresu, která je uložena jako konstanta *ADR*, tak třídění probíhá jednoduše porovnáním adresového rámce. Tento stav je opuštěn jen případě, kdy adresový rámec obsahuje adresu shodnou s *ADR*, potom je umožněno přijímat i datové rámce.

2. *Status* – Ze stavového bytu je přečtena délka zprávy, která je uložena a následně využita pro určení konce zprávy a významu jednotlivých bytů zprávy. V předchozích kapitolách byl popsán způsob ověřování správnosti zpráv, který využívá CRC kontroly. V případě, kdy je zpráva přijata správně, posílá/přijímá se zpráva ACK. V případě, kdy stav očekává přijetí zprávy ACK, ověřuje, zda místo stavového bytu nepřišla ACK (potvrzovací zpráva). Ověřování je důležité kvůli tomu, aby přijímací stavový automat správně vyhodnotil přijímanou zprávu. Zpět do stavu *Address* automat přechází, když je přijata chyba, či adresový rámec. Nejedná-li se o ACK byte a stavový byte je správně přijat, ukládá se do kruhového zásobníku, jak je naznačeno šipkou.

3. *Bytes* – V předchozím stavu byla zjištěna délka zprávy, jinými slovy počet přijímaných bytů. Tento stav kontroluje správnost bytů a v případě, kdy je vše v pořádku, ukládá byty do zásobníku. V době, kdy přijal všechny byty, přechází do dalšího stavu.

4. *CRC* – Jelikož je potřeba, aby stavový automat byl rychlý, CRC kód pouze uloží a přijímací UART komponentu přepne na akceptování pouze adresových rámců. Zároveň je nastaveno, že byla přijata celá zpráva a program se může postarat o její zkontrolování a následně poslání na výstup z této knihovny.

Jak již bylo předesláno, po přijetí celé zprávy je nastavena vlajka (*flag* – anglické označení pro booleovskou proměnnou, která poskytuje informaci o daném stavu), která signalizuje přijetí a uložení celé zprávy do kruhového registru. Je tedy na čase, aby program ověřil, zda přijatá zpráva byla přijata správně. Kapitola 2.4.4 popisuje, jak probíhá výpočet 8bit CRC. Výsledek tohoto výpočtu se porovná s přijatou hodnotou CRC, v případě, kdy jsou hodnoty stejné, je tato zpráva označena jako správná a odeslána potvrzovací zpráva ACK.

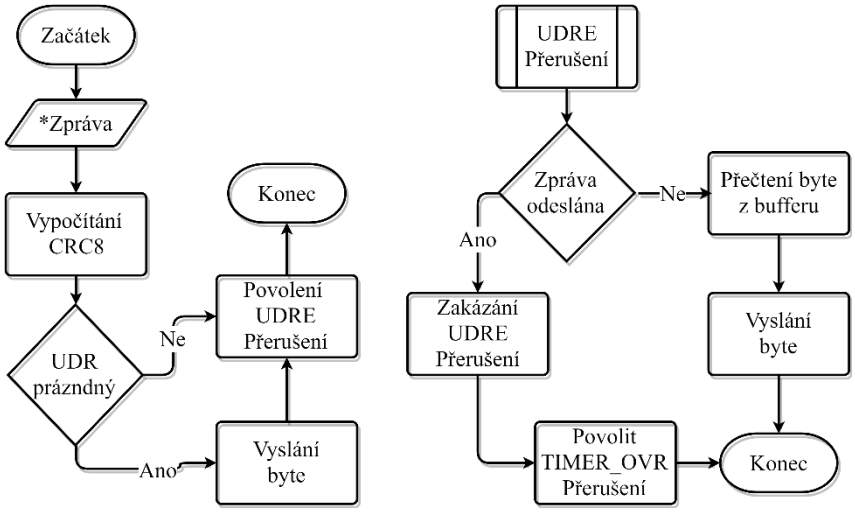
Zprávy jsou uloženy v cyklickém zásobníku, který uchovává až 64 bytů. Ukládání je naznačeno ve zmiňovaném stavovém automatu jemně tečkovanou čarou. Vždy je potřebné během ukládání ukládat i průběžnou hodnotu, kolik bytů bylo doposud uloženo, kvůli tomu, aby bylo možné dočasná data vymazat, v případě, kdy CRC kód neodpovídá, nebo nastane chyba již v době přijímání. Nebylo ošetřeno přetečení kruhového zásobníku, jelikož se tím ztratí pouze zpráva, která nebude potvrzena, tudíž zpráva přijde znovu.

3.3.2 Vysílání zpráv

Každý Slave modul vysílá zprávy pouze Masterovi, tudíž první adresový rámec/byte nemusí být při každé zprávě nastavovaný, je tedy dán jako konstanta. Jelikož je spolu se zprávou vysílán i CRC kód, je třeba jej spočíst. Když první byte je konstantní, není potřeba, aby se první průběžné CRC počítalo pořád dokola, je tedy vypočítáno při inicializaci a následně se započítávají jen zbylé byty. Za poslední byte se přidá CRC a zpráva je kompletní, připravena na vyslání. Pro lepší názornost je zde uveden obr. 16.

Levá strana vývojového diagramu na počátku přijme zprávu jako ukazatel na pole bytů. Následně se postupně spočítá CRC8, jak bylo předesláno. Odesílání zpráv pobíhá postupně po jednom bytu. V případě, kdy registr určený pro uložení a následnému vyslání (*UDR*) je prázdný, program do toho registru uloží první byte a povolí přerušení (*UDRE*), které je vyvoláno v případě uvolnění tohoto registru. Pokud je registr již plný, pouze se umožní přerušení (*UDRE*).

Pravá strana vývojového diagramu graficky popisuje vykonávání rutiny přerušení. Ta se vyvolává tak dlouho, dokud není odeslána celá zpráva. Po jejím celém vyslání se zakáže přerušení (*UDRE*) a povolí se přerušení při přetečení časovače, který se také tímto krokem spustí. Po překročení časového limitu je zpráva vyslána znovu, což vysvětluje, proč jsou data ze zásobníku nedestruktivně čtena (*peek*). Limit opakovaného vysílání je nastaven na tři pokusy. Pokud přijde potvrzení ACK od Master modulu o správném přijetí zprávy, časovač je zastaven a zpráva vymazána ze zásobníku.



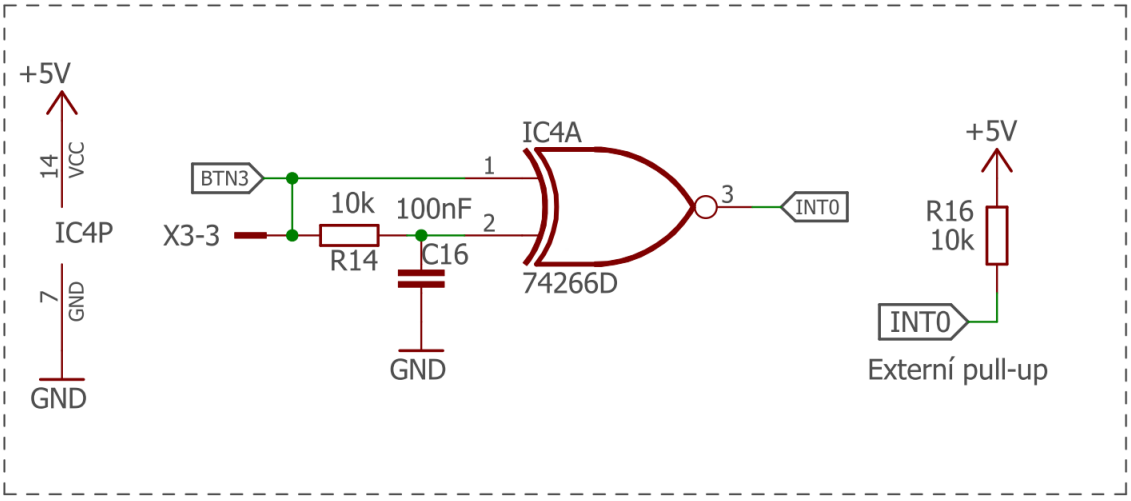
obr. 16: Vývojový diagram programu pro vyslání zpráv

3.4 Button modul

Button modul neboli modul tlačítek není zmíněn jako první z navrhovaných modulů jen tak náhodou. Jedná se o nejpoužívanější komponentu v systému inteligentní domácnosti. Slouží pro snímání tlačítek umístěných na zdi. Ač systém inteligentního domu je moderní záležitost, neustále využívá nástěnné vypínače, či tlačítka, která se používají od dob prvních elektroinstalací (samozřejmě s menšími obměnami). Požadavky na tento modul se zejména týkají jeho velikosti a energetické nenáročnosti (pasivní modul z pohledu napájení). Je potřeba, aby se tento modul vlezl do prostoru za tlačítko na zdi, tudíž musí být velmi malý. Energetická nenáročnost je požadována z důvodu nepřítomnosti zdroje, a proto tento modul musí být napájen ze sběrnice.

3.4.1 Hardware

Na obr. 17 je jeden ze vstupních obvodů pro tento modul. Jednoduše se jedná o detektor hrany, což je výhodné kvůli umožnění vzbuzení µPC při změně stavu na tlačítku. Zde je nutné říct, že je počítáno s tlačítky, tedy po puštění tlačítka se vrací do své původní polohy. Kvůli potřebě nízké spotřeby je µPC v úsporném režimu po celou dobu. Když je µPC v úsporném režimu, tak jsou oba převodníky deaktivovány, aby zbytečně nepřeváděly zprávy ze sběrnice, jež by µPC nevyhodnotil. Spotřeba je tedy minimální, ovšem je potřeba vzbudit µPC v pravou chvíli. Ke vzbuzení slouží osm detektorů hrany, které vzbudí µPC při každé změně na tlačítku, jinými slovy µPC je aktivní pouze po dobu vyhodnocení stavu na tlačítkách, odeslání zprávy a přijetí potvrzovací zprávy od Mastera. Díky popsanému přístupu se spotřeba modulu sníží na minimum.



obr. 17: Vstupní obvody pro Button modul

Každý z logických hradel XNOR (Komplement exklusivní disjunkce) má výstup typu otevřený kolektor (OC – open collector), díky kterému je možné jednoduše spojit všechny výstupy jednotlivých hradel do jednoho uzlu. V tomto případě se realizuje logická disjunkce všech výstupů XNOR hradel, což znamená, že při detekci změny stavu na jakémkoliv tlačítku bude vzbuzen µPC. Rezistor R_{16} zapojený jako pull-up nastavuje výchozí hodnotu na vstupním pinu µPC pro přerušení na logickou jedničku. Při nepoužití rezistoru R_{16} by nebyla definovaná napěťová úroveň při všech otevřených výstupních kolektorech, což by zapříčilo citlivost na rušení až nefunkčnost obvodu.

3.5 PWM modul

Druhým v pořadí nejpoužívanějších modulů je PWM modul, který, jak již z názvu vyplývá, generuje výkonový PWM signál pro řízení svitu 12V LED pásků, obecněji LED světel. Naopak od dříve popisovaného Button modulu se nejedná o senzor, ale aktuátor neboli akční člen. Druhou signifikantní odlišností je aktivní typ modulu z pohledu napájení, tedy dodává napětí do sběrnice.

3.5.1 Problém přinášející PWM regulace – blikání

Nechtěné blikání světla je způsobeno PWM regulací svitu. Viditelnost blikání ovlivňuje dominantní frekvence modulace. Když je frekvence relativně nízká (pod 90 Hz), tak je blikání pro člověka viditelné ve většině případů. Na vyšších frekvencích blikání již nemusí být přímo viděno, ale je možné jej zaznamenat rychlým pohybem hlavou, očí či předmětu. Tento efekt je známý jako stroboskopický efekt. [7]

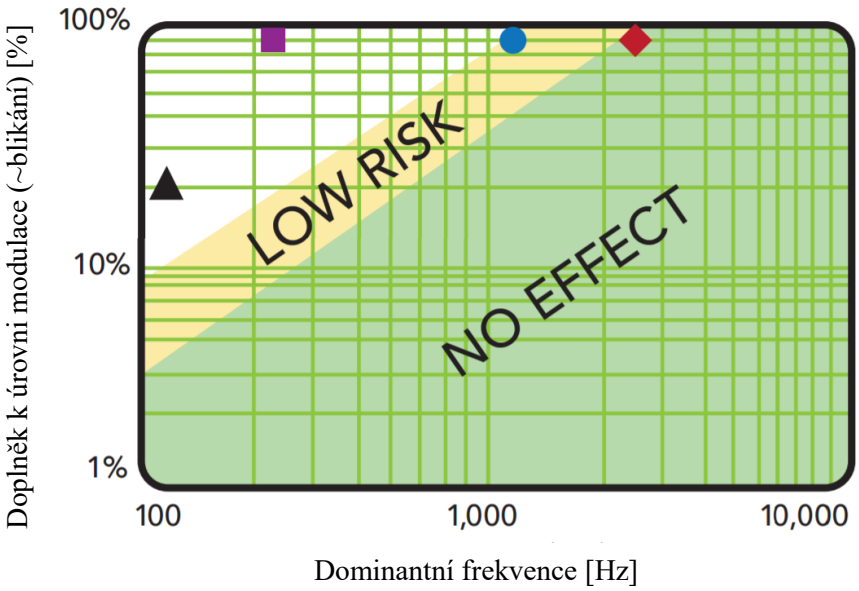
Sluneční světlo se vyznačuje stálým (neblikajícím) efektem, stejně tak i žárovky se žhnoucím vláknem a halogenové žárovky, nebo je jejich blikání minimální, projevující se spíše mírným pohasínáním nežli blikáním. Tento jev je způsobený tepelnou kapacitou wolframového vlákna. Výbojky jako jsou fluorescentní světla a polovodičové zdroje světla jako jsou LED a lasery se již nevyznačují, či jen minimálně, setrvačností svitu. Kvůli této neschopnosti je potřeba těmto světlům dodávat stabilní napájení či napětí s vysokou frekvencí. Podívejme se tedy na to, jakou frekvenci potřebujeme, abychom eliminovali vnímání blikání světla. [7]

Prozatím jsme se bavili pouze o dominantní frekvenci modulace, která ovlivní viditelnost blikání, ač přímou či nepřímou. Zamyslíme-li se nad PWM regulací, musíme uznat, že viditelnost blikání

musí záviset i na střídě signálu. Bude-li střída 5%, světlo bude sepnuto pouze na jednu dvacetinu periody, tedy povětšinu doby bude zhasnuto, a tudíž bude blikání světa snadněji postřehnutelné. Bude-li naopak 95% střída, světlo zhasne pouze na jednu dvacetinu periodu, tudíž nebude lehké zaznamenat zhasnutí světla (což ani nechceme). Jednoduše popsaná problematika ovlivnění viditelnosti blikání úrovní modulací je znázorněna na obr. 18. Obrázek reprezentuje závislost viditelnosti blikání jak na dominantní frekvenci modulace, tak i na úrovni modulace, neboli v případě PWM modulace se jedná o střídu. Osa *Y* je doplňkem ke střídě (*1 – Střída)*, odpovídá úrovni blikání.

Popis ke grafu (obr. 18):

- Doporučená minimální frekvence pro nízké riziko (1250Hz)
- Xicato XIM, na 1% úrovni intenzity (2500Hz) {Prezentace výrobku}
- „Kvalitní“ Stmívač, na 5% úrovni intenzity
- Tradiční stmívače, na 20% úrovni intenzity



obr. 18: Závislost vnímání blikání světla na frekvenci a úrovni modulace [7]

Co je vlastně tím popisovaným rizikem, kterého je možno se zbavit vyšší dominantní frekvencí modulace? Každý člověk vnímá blikání na vyšší frekvenci jinak. Někteří nejsou ani při nižších frekvencích schopni zpozorovat blikání světla, někteří naopak jsou mnohem citlivější a vidí blikání světla, o kterém si většina myslí, že již kontinuálně svítí. Stejně tak, jak je různá rozlišovací schopnost jedince, jsou i různé následky vnímání blikání světla na danou osobu. Důsledek vystavení jedince blikání na frekvenci, která je nižší než 2 kHz, může namáhat oči a mít za následek jejich únavu. Nejhorším případem je až bolest hlavy, která je pravděpodobně způsobena tím, že se mozek snaží nějak vypořádat s blikáním, které jedinec vnímá pouze podvědomě. Je zcela přirozené, že mozek filtruje mnoho informací, které mu přichází z centra vidění (tedy očí), což nám dokazují optické klamy. Například, když oko zaostří na jiný předmět s rozdílnou vzdáleností, než byl prvně sledovaný objekt, jedinec si povětšinou neuvědomuje přechodný jev přeostření (záleží na rozdílu vzdáleností), proto nevidíme ani blikání světla na vysoké frekvenci, ale mozek jej vnímat může. Mimo jiné blikání nebude vítáno v případě pořizování fotek, což je ale již minoritním problémem.

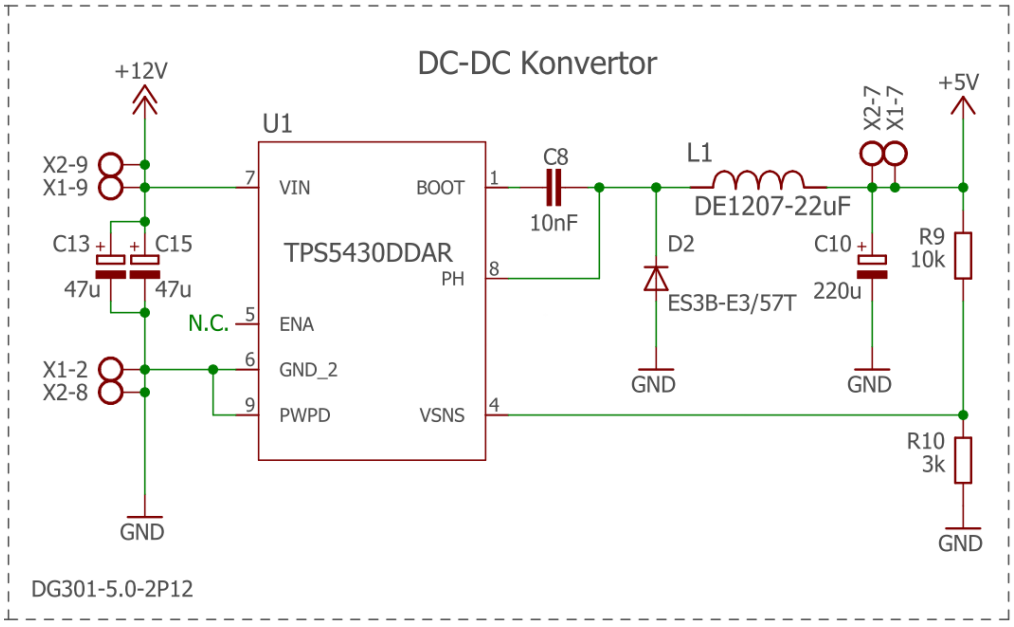
3.5.2 Střída PWM vs. Intenzita světla

Nelineární závislost střídy PWM na intenzitě světla není příjemná z pohledu regulace. Otázkou je, jaká je mezi těmito veličinami závislost? Pro zodpovězení této otázky je potřeba znát význam PWM regulace a teorii LED. Bude tedy použita jednoduchá úvaha pro objasnění této závislosti.

Dodávaný výkon PWM regulací je lineárně závislý na střídě, což umožňuje jednoduché řízení. Ovšem závislost intenzity světla na dodávaném výkonu již lineární není. Výrobci LED pásků jsou velmi skoupí na informace ohledně jejich výrobků. Většinou uvedou jen základní informace, což pro účel zjištění závislosti intenzity světla na výkonu není moc vhodné. Některé datasheety o LED (SMD), které jsou mimo jiné použity v LED páscích, uvádějí závislost intenzity světla na procházejícím proudu. Jednotlivé datasheety se většinou shodují na logaritmickém průběhu, ovšem každý datasheet má trochu odlišné parametry logaritmického průběhu. Nejjednodušším zjištěním přibližné závislosti bylo realizovat PWM regulaci, připojit daný LED pásek a pozorovat, jak se mění svit při lineárním zvyšování střídy. Pozorování potvrdilo přibližnou logaritmickou závislost. Aby bylo možné tuto nelinearitu linearizovat, je potřeba zvolit řídící exponenciální funkci, čímž se eliminuje nelinearita. Jak již bylo naznačeno, není jednoduché zjistit parametry logaritmické funkce, tudíž hledání exponenciální funkce pro linearizaci je nejjednodušeji proveditelné pomocí metody pokus omyl. Aby toho nebylo málo, přichází další problém u RGB pásků, a to, že každá barva má jiné parametry logaritmické funkce, tudíž by bylo třeba mít více exponenciálních funkcí pro kompenzaci různých funkcí logaritmických.

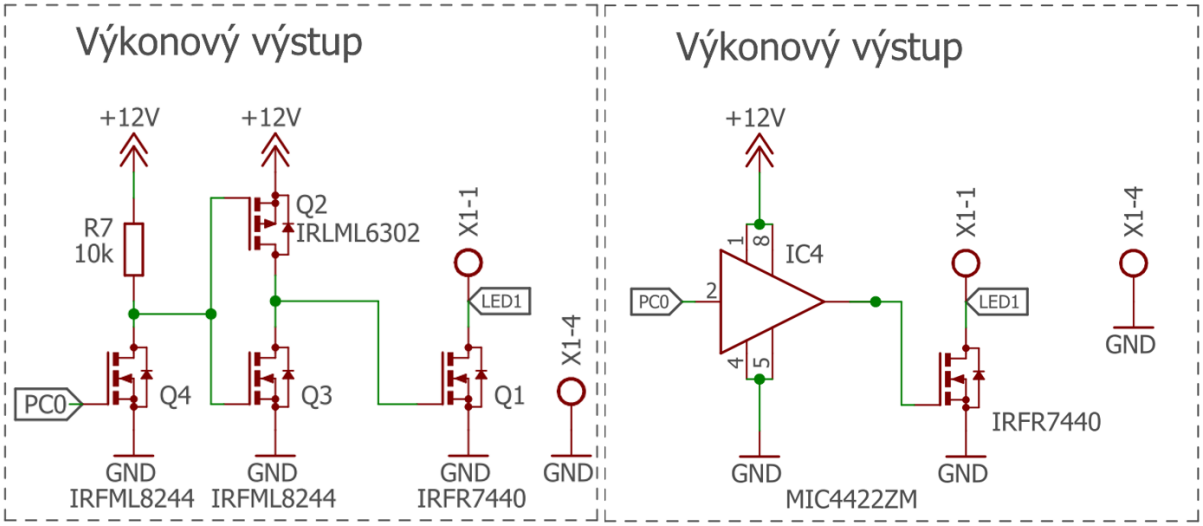
3.5.3 Hardware

Obvod dodávající energii do sběrnice je na obr. 19. Integrovaný obvod *TPS5430DDAR* je srdcem napájecího obvodu, jež vychází z datasheetu tohoto integrovaného obvodu. S velkou účinností (až 95 %) převádí vyšší 12V napětí na nižší napětí 5 V. Zároveň je tento obvod výkonný, jelikož je schopný dodat do obvodu 3 A, špičkově až 4 A. Spínací frekvence 505 kHz je výhodná pro nižší náročnost výstupního filtru. Jediným rozdílem oproti doporučenému zapojení je nahrazení tunelové (B320A) diody za usměrňovací diodu (ES3B). Tato dioda splňuje požadavek maximálního procházejícího proudu 3 A.



obr. 19: DC-DC konvertor

Výstupní výkonový obvod je tvořen třemi stupni, jak lze vidět na obr. 20. Nejprve bude popsáno zapojení v levém čárkovaném obdélníku. Vícestupňový návrh je zvolen pro vyšší účinnost i při vyšší spínací frekvenci. První jednoduchý stupeň je pouze pro převedení 5V řídícího signálu z µPC na 12V, který je přiváděn na stupeň druhý. Zapojení tranzistoru Q_4, tvořící 1. stupeň, představuje spínací režim s odporovou zátěží R_7. Druhý stupeň tvořen tranzistory Q_2 a Q_3 je invertorem se strmou převodní charakteristikou. Právě strmá převodní charakteristika je výhodná pro tvarování spínacího signálu pro třetí, poslední, stupeň. Tvarování spínacího obdélníkového signálu je ve smyslu zvyšování strmosti jeho hran. Právě při malé strmosti hran obdélníkového signálu dochází ke ztrátám energie při spínání, proto je zde použit tento invertor. Je potřeba si uvědomit, že tento výkonový invertor je nutné sestavit z tranzistorů, které mají podobné parametry, aby invertor pracoval v co nejlepším souběhu. Jelikož dva tranzistory použité v obvodu nemají zcela stejné parametry, tak ve vyšších frekvencích dochází ke špatnému souběhu, a tudíž obvod je nepoužitelný. Proto je zde uveden i druhý obvod, v pravém čárkovaném obdélníku, o kterém bude zmínka v následujícím odstavci. Třetí stupeň je tvořen pouze tranzistorem, jehož zátěž se připojuje externě a může jí být zmiňované LED světlo. Na tento tranzistor jsou kladeny nejvyšší nároky z pohledu výkonu. Použitý tranzistor *IRFR7440* se vyznačuje skvělými parametry pro spínání vysokých proudů. Datasheet tohoto tranzistoru uvádí hned několik proudů, které jsou limitovány pouzdrem, křemíkem a propojovacími drátky čipu. Právě propojovací drátky nastolují nejvyšší omezení proudu a to na 90 A. Proud, který tranzistor dokáže špičkově sepnout je až 760 A. Jen pro informaci datasheet upozorňuje na to, že tyto údaje jsou vypočítány z maximální povolené teploty. Důležitým parametrem je také odpor tranzistoru v sepnutém stavu, který je typicky 1,9 mΩ. Důležitým parametrem při výběru tohoto tranzistoru byla doba náběžné resp. sestupné hrany tranzistoru (39 ns resp. 34 ns).



obr. 20: Výkonový výstup PWM modulu

Obvod v pravém čárkovaném obdélníku je mnohem jednodušší díky použitému integrovanému obvodu *MIC4422ZM*. Tento integrovaný obvod slouží k ovládání MOSFET tranzistorů, jako je použitý výkonový tranzistor *IRFR7440*, jinými slovy se jedná o budič výkonových MOSFET tranzistorů. Výhodou je, že vstup budiče je kompatibilní s TTL a CMOS logikou, tudíž není potřeba k tomuto obvodu přidávat žádný další hardware. Budič je schopný dodat špičkově až 9 A, což je výhodné pro přebíjení velké vstupní kapacity hradla výkonového tranzistoru, která je pro použitý tranzistor 4610 pF. Datasheet tohoto budiče garantuje náběžnou a sestupnou kolem 25 ns při zatížení 10nF kapacitou. S budičem je možné se dostat na mnohem vyšší frekvence spínání, při nižším odběru oproti obvodu s tranzistory, zejména když je použit rezistor 1 kΩ namísto 10 kΩ (pro rychlejší náběžnou hranu).

3.5.4 Software

Použitý µPC ATmega8 je vybavený jedním PWM výstupem, což je zcela nedostačující, proto bylo zvoleno generování PWM signálu čistě softwarově. Každý výstup má svoji 8bit proměnnou, která určuje střídu pro daný výstup. V hlavní smyčce se neustále inkrementují tyto proměnné a porovnávají se s nastavenou střídou, která jde nastavit v rozsahu 0 – 255. Při simulaci programu byla dominantní frekvence modulace spočtena na 1740 Hz, což je podle předchozí studie velmi dobrým výsledkem. Tento modul s frekvencí PWM, která se pohybuje za hranicí (nízkého) rizika, je tedy možno považovat za modul bez rizika vyvolání problému při používání funkce stmívání.

Dříve zmíněná problematika nelineární závislosti dodávaného výkonu PWM regulace na intenzitě světla, kterou je potřeba linearizovat exponenciální funkcí je řešena následovně. Jelikož µPC nenabízí vysoký výpočetní výkon, není akceptovatelné, aby tento převod byl pocítěn při přenastavení intenzity svitu. Proto byla využita tabulka, která nabízí exponenciální převod. Tabulka (angl. *Lookup table*) na vstupu přijímá hodnoty v procentech (0 – 100 %), na jejímž výstupu je hodnota střídy PWM regulace (0 – 255).

Program také umožňuje plynulé nastavení intenzity světla pomocí přerušení. Parametry časovače je možné ovlivnit, díky čemuž je možno nastavit rychlost, s jakou je zvyšována, respektive snižována, intenzita světla. Stejně tak je možno volit, jak velké kroky budou použity. Tato funkce se již využívá pro procentuální nastavení intenzity světla, což má za následek pocit lineárního krokování. Nastavení intenzity svitu je možno jak pomocí procent, tak pomocí střídy. Je potřeba umět přepočítat hodnotu střídy na procenta. Dříve použitá tabulka neumožňuje tento (inverzní) směr převodu, proto byla pro účely převodu vytvořena vyhledávací funkce pomocí bisekce (půlení intervalu), která najde nejbližší hodnotu v tabulce a převede ji na procenta. Následuje ukázka kódu, která zahrnuje i možnost hledání co nejbližší hodnoty pro více než jednu proměnnou.

```
uint8_t left, middle, right;
for (int n = 0; n < 6; n++) {
      if (gradualMask & (1 << n)) {
         left = 0;
         right = 100;
         for (int i = 0; i < 10; i++) {
             middle = left + (right - left)/2;
             if (LED[n] == gammaCorrection [middle])
                 break;
             if (LED[n] < gammaCorrection[middle])
                 right = middle;
             else
                 left = middle;
         }
         desiredLED[n] = middle;
      }
}
```

V prvním řádku jsou deklarovány proměnné, které zpočátku určí interval, ve kterém se vyskytuje hledaná proměnná. První *for* cyklus slouží pouze pro umožnění hledání správných hodnot hned pro všech šest výstupů. Dále jsou inicializovány hranice intervalu. Druhý cyklus již slouží pro hledání nejbližší hodnoty v tabulce, která je v kódu pojmenována jako *gammaCorrection* a jedná se o pole bytů. Proměnná *middle* uchovává vždy střed intervalu daného proměnnou *left* a *right*, proto se této metodě říká půlení intervalu. Polovina intervalu se vždy vypočte tak, že se nejdříve vypočte polovina mezi proměnnou *left* a *right*, tedy polovina velikosti intervalu. Ovšem tato polovina velikosti je pouze relativní vzdáleností, a proto se musí připočíst posun od nuly, který odpovídá proměnné *left*. Rovnice je popsaná tak, aby bylo zřejmé, jak byl vztah odvozený. Jednodušší tvar této rovnice je: *middle = (right + left)/2*. Následně se vyzkouší, zda náhodou nebyla trefena správná hodnota, pokud ano, tak je cyklus ukončen a hodnota uložena. Netrefil-li se algoritmus do správné hodnoty rovnou, je zjištěno, zda hodnota *middle* je větší či menší, podle čehož se posunou hranice intervalu. Takto, nanejvýše po deseti krocích, je nalezena velmi rychle nejbližší hodnota. Bylo by možné iterace ukončit pomocí vyhodnocení akceptovatelné minimální chyby, či jakmile řešení začne oscilovat kolem výsledku, ovšem algoritmicky je jednodušší pokusy ukončit po 10 krocích a nezatěžovat µPC dalším vyhodnocováním.

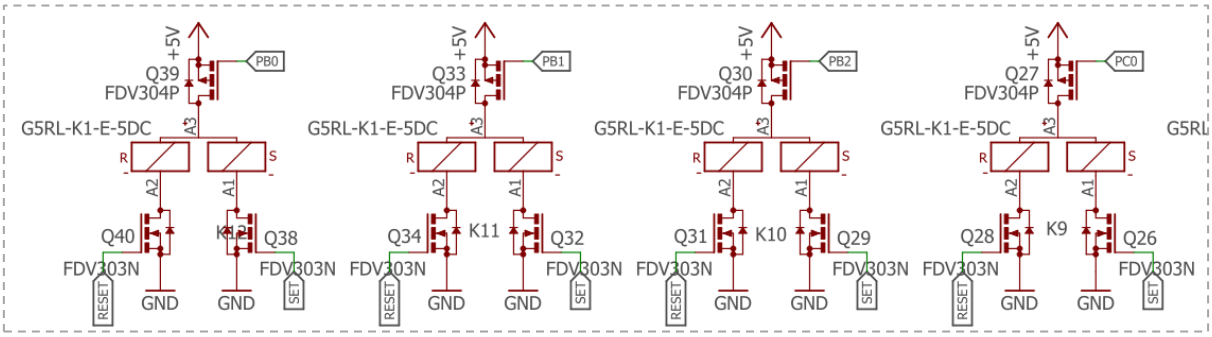
3.6 Relay modul

Modul obsahující relé nemusí sloužit pouze ke spínání světla, jak by se na první pohled mohlo zdát, ale i například přivádění elektřiny do různých okruhů v bytě. Pořád se nejedná o výčet všech možností. Dle použitých aktuátorů je možné tímto modulem ovládat pohon závěsů, spínání odvětrávání nebo dvoustavové ovládání elektrických termohlavic. Využití je tedy široké a závisí pouze již na výběru aktuátorů.

3.6.1 Hardware

Hardware toho zařízení je relativně jednoduchý, jelikož využívá relé a výkonově posílené výstupy přímo z µPC, zapojené do matice. Hned si řekneme, proč se jedná o maticové zapojení. Nejdříve však ale zvažme energetickou zátěž tohoto modulu. Již bylo předesláno, že celý design má být energeticky nenáročný. Při použití obyčejného monostabilního relé, které je v aktivním stavu pouze v době, kdy cívkou relé protéká proud, tak by tento modul byl relativně náročný na spotřebu energie. Trh nabízí skvělé řešení, kterému se říká bistabilní relé. Bistabilní relé oproti obyčejnému relé disponuje i permanentní magnety, které drží relé buď v sepnutém stavu či rozepnutém do doby jeho vynuceného přepnutí. Spotřeba energie je tedy pouze v době přepnutí z jednoho stavu do stavu druhého. Tím je tedy energetická náročnost snížena na minimum. Jelikož je potřebné určit, zda bude relé sepnuto či rozepnuto, tak je potřebné využít k jednomu relé dva vodiče a třetí k uzavření obvodu. K uzavření obvodu je využit zemnící vodič, tudíž by byl potřebný dvojnásobný počet vodičů k počtu relé. K dvanácti relé by tedy bylo třeba 24 vodičů, což požitý µPC neumožňuje. Právě maticové řešení zapojení je řešením pro snížení počtu potřebných vodičů na počet relé, plus dva vodiče na určení sepnutí či rozepnutí relé. Na obr. 21 je vidět použité maticové zapojení cívek relé. Posílení výstupů je provedeno jednoduše pomocí unipolárních tranzistorů, které jsou přizpůsobeny na spínání větších induktivních zátěží, jelikož mají v sobě implementované diody, chránící proti zpětnému proudu. Jednotlivé cívky relé jsou spojeny ve stejném smyslu. Podívejme se na první relé zleva na obr. 21. Horní tranzistor Q_{39} slouží k adresování relé, zatímco tranzistory Q_{38} a Q_{40} určují, zda relé bude sepnuto, či rozepnuto.

Relé (*G5RL-K1-E-5DC*) disponuje dvěma protilehlými kontakty, z nichž jeden je schopný vést 16 A a druhý 5 A. Maximální spínané napětí je 250 V střídavých nebo 24 V stejnosměrných. Modul s výhodou používá ze dvou kontaktů pouze ten více zatížitelný, aby bylo možné spínat i obvody elektrických zásuvek a světel.



obr. 21: Maticové zapojení cívek relé

3.6.2 Software

Ovládací software má za úkol dekódovat příchozí zprávy a podle nich spínat či rozepínat relé. Druhým úkolem je časové vypnutí relé, které se hodí zejména v případě, kdy relé ovládá odvětrávací ventilátory. Modul je také schopný odeslat zprávu o stavu jednotlivých relé, aby bylo možné centrálnímu systému oznámit, v jaké stavu se relé nacházejí. V případě časovaného relé je nutné, aby tento modul odeslal informaci centrálnímu systému o jeho rozepnutí, aby se nemusel systém zabývat časováním.

Využitý µPC nedisponuje dvanácti časovači, aby pro každé relé mohlo být časované, proto je využitý jeden časovač, který do pole o dvanácti proměnných inkrementuje hodnotu reprezentující sekundy. Proměnné jsou typu *uint16_t*, neboli kladné 16bit proměnné, která nabývá hodnoty až 65 535. Relé tedy je schopno být vypnuto až za přibližně 18 hodin a to s rozlišením na sekundy. Tento parametr je předimenzovaný, ale využitý µPC již není dalšími funkcemi zaneprázdňován.

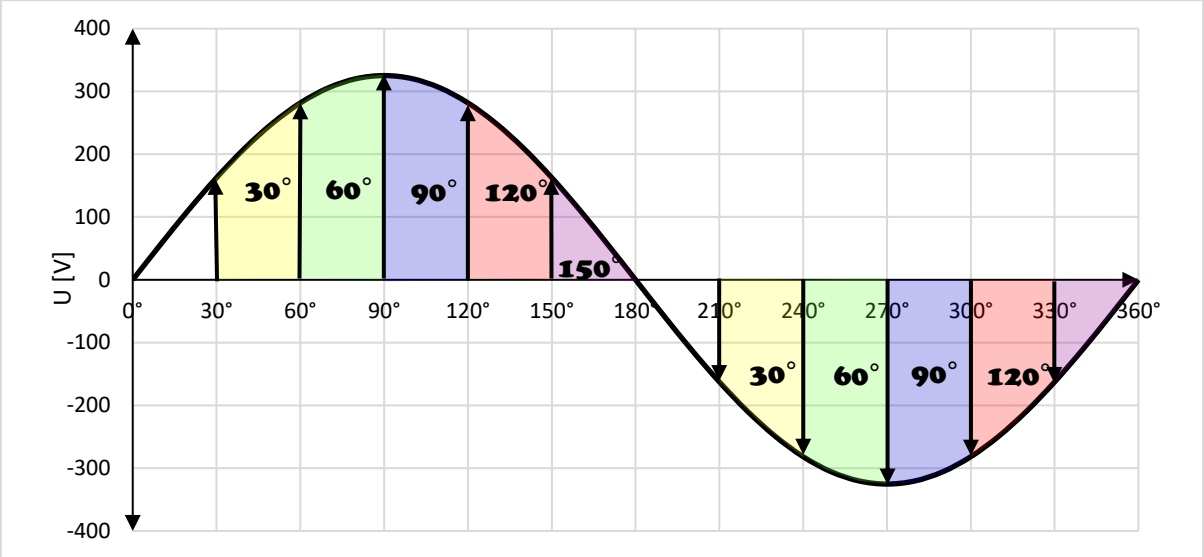
Přepínání relé je prováděno krátkými impulzy. Datasheet použitého relé (*G5RL-K1-E-5DC*) udává dobu pulzu jak pro sepnutí, tak i pro rozepnutí 30 ms. Právě doporučených 30 ms je použito v programu. Po tuto dobu je vodivý již popisovaný adresovací tranzistor. Vždy se nejprve sepne jeden ze dvou tranzistorů určující sepnutí či rozepnutí relé, poté se teprve aktivuje adresovací tranzistor. Překlopení relé trvá podle výrobce 15 ms, takže 30ms pulz je dostatečně dlouhý na to, aby překlopení relé bylo stabilní. Při použití kratšího pulzu jak 30 ms, může dojít k tomu, že se relé nastaví do původní polohy. Jedná se o jakousi mechanickou metastabilitu, kdy není zcela jasně definován stav, ve kterém se bude relé nacházet po jeho nastavení.

3.7 Triac modul

Modul byl pojmenován podle polovodičové součástky triak, jež se obvykle používá v obvodech střídavého proudu pro fázové řízení, jakož tomu je u tohoto modulu. Modul tedy umožňuje řídit úhel otevření triaku α, a tím ovlivnit dodávaný výkon až do šesti spotřebičů. Zde je spotřebičem myšlena žárovka a jí podobné spotřebiče, které se vyznačují setrvačností svitu. V kapitole 3.5.1 již byl zmíněn problém týkající se stroboskopického efektu a jeho vnímání člověkem. Problém byl signifikantní zejména při použití nižší frekvence regulace, což je přítěží tohoto modulu. Triak modul pracuje čistě s frekvencí sítě

50 Hz. Řízení probíhá každou půlvlnu, tudíž frekvence regulace je 100 Hz, což je pořád nízká frekvence na to, aby nebyla vnímána člověkem, spíše nepřímo vnímána. Kvůli popsanému problému je nutné použít tento modul pouze pro světla se schopností setrvačného dosvitu.

Fázové řízení triaku je znázorněno na obr. 22, spolu s vyznačenými příklady úhlů otevření triaku. Doba otevření triaku určuje, jak moc velká část výkonu půlvlny se přenese ze zdroje do zátěže. Tato doba je počítána od okamžiku průchodu signálu nulou, jež se dá převést pomocí periody či frekvence na úhel otevření, jak ukazuje obr. 22 a popisuje vztah (1). Úhel otevření je vždy vyznačen šipkou pro vybrané úhly otevření α, mezitímco vypnutí nastává, z podstaty triaku, průchodu proudu nulou.



obr. 22: Úhel otevření triaku α

Přepočet doby, uvažované od průchodu signálu nulou do okamžiku sepnutí triaku, je uveden ve vztahu (1), kde *t* je popisovaná doba, *f* je frekvence signálu a *T* jeho perioda.

$$\alpha = 360°ft = 360°\frac{t}{T}, \qquad [°] \tag{1}$$

Následně bude odvozen vztah mezi dodaným výkonem do obvodu v závislosti na úhlu otevření. Vzorec (2) uvádí okamžitou hodnotu výkonu na čistě ohmické zátěži v závislosti na čase. Vzorec (3) uvádí závislost okamžitého výkonu na úhlu otevření, jež bude využit pro odvození závislosti mezi dodaným výkonem na úhlu otevření triaku α.

$$p(t) = \frac{1}{R}U_{max}^2 sin^2(2\pi ft), \qquad [W] \tag{2}$$

$$p(\alpha) = \frac{1}{R}U_{max}^2 sin^2\left(\frac{\alpha\pi}{180°}\right), \qquad [W] \tag{3}$$

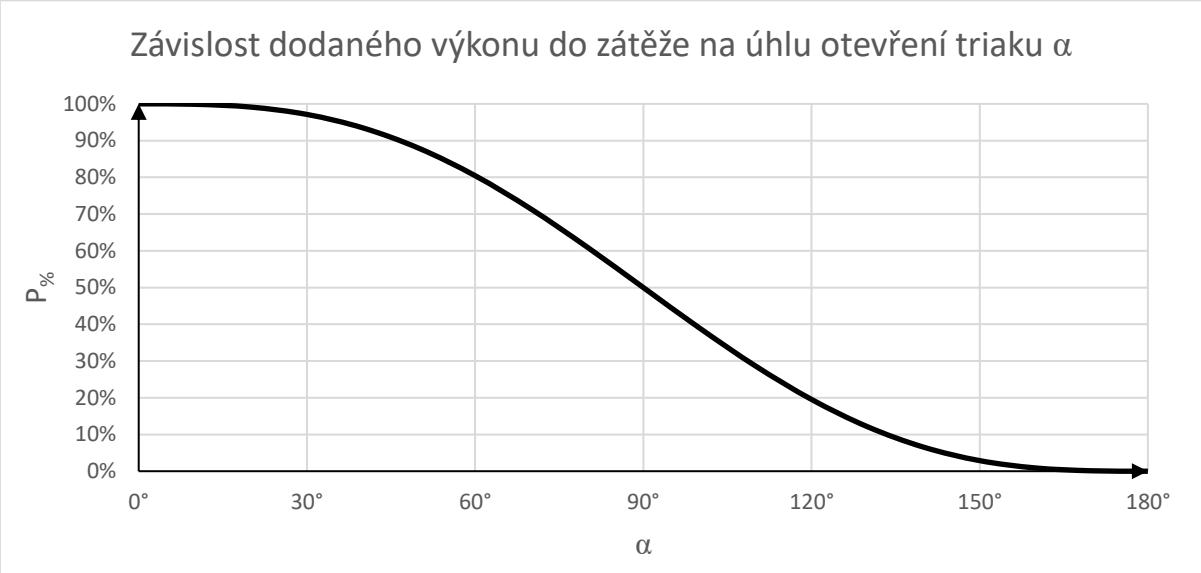
Poměrná hodnota výkonu dodané do zátěže bude vypočítána jako poměr plochy pod půlvlnou vůči ploše pod určitou částí půlvlny, která je určena úhlem otevření. Je vycházeno z definice střední hodnoty signálu, kde se navíc dělí integračním intervalem, v tomto případě by se jednalo o půlperiodu. Odvození tohoto vztahu, bez uvedených mezikroků integrace, je uvedeno ve vztazích (4) až (6).

$$P_{\%}(\alpha) = \frac{\left(\frac{1}{R}U_{max}^2\right)\int_{\alpha}^{180°} sin^2\left(\frac{\varphi\pi}{180°}\right)d\varphi}{\left(\frac{1}{R}U_{max}^2\right)\int_{0°}^{180°} sin^2\left(\frac{\varphi\pi}{180°}\right)d\varphi} \times 100, \qquad [\%] \tag{4}$$

$$P_{\%}(\alpha) = \frac{1}{90}\int_{\alpha}^{180°} sin^2\left(\frac{\varphi\pi}{180°}\right)d\varphi \times 100, \qquad [\%] \tag{5}$$

$$P_{\%}(\alpha) = \left[1 - \frac{\alpha}{180} + \frac{1}{2\pi}sin\left(\frac{\alpha\pi}{90}\right)\right] \times 100, \qquad [\%] \tag{6}$$

Graficky vyjádřená závislost (6) dodaného výkonu do zátěže je na obr. 23. Jelikož uživatel nebude nastavovat úhel otevření, ale procenta svitu zdroje světla, bude potřeba tento vztah zabudovat do programu, který bude převádět nastavený svit v procentech na úhel otevření triaku α. Bude se tedy jednat o inverzní funkci k funkci (6).

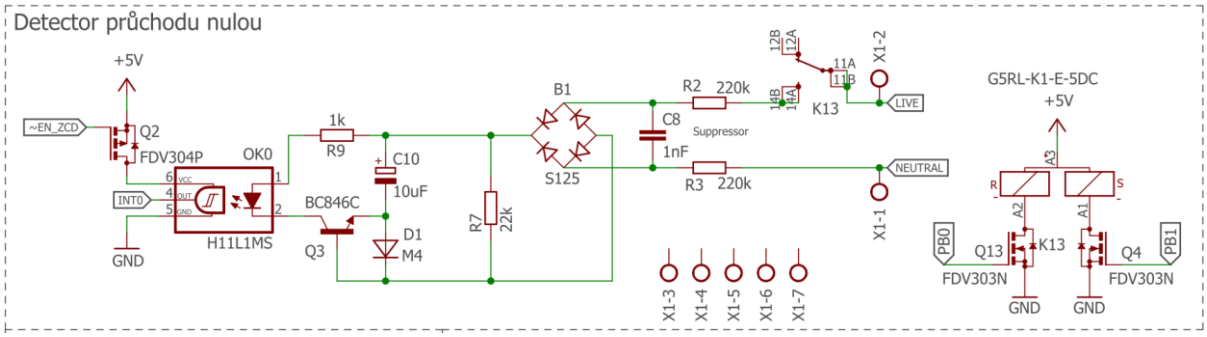


obr. 23: Závislost dodaného výkonu na úhlu otevření triaku α

3.7.1 Detektor průchodu nulou

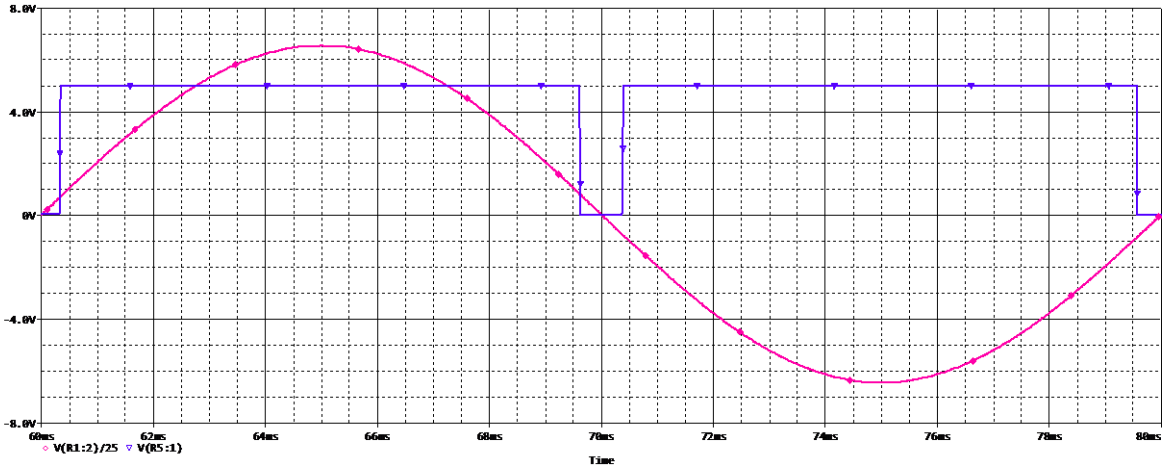
Častokrát bylo zmíněno, že úhel otevření je počítán z doby dle vztahu (1), v němž je potřeba znát čas sepnutí od průchodu signálu nulou. Proto je potřeba vědět, kdy signál prochází nulou. Tuto informaci poskytuje detektor průchodu nulou. Obvod detektoru byl převzat z webové stránky [10] . Úprava tohoto obvodu byla provedena přidáním relé, které umožní deaktivaci detektoru odpojením od síťového napájení. Relé spolu s tranzistorem Q_2 pomáhají snížit spotřebu zařízení v době, kdy není modul používaný.

Rezistory R_2, R_7 a R_3 tvoří dělič napětí, který mimo jiné zajistí, že napětí na kondenzátoru nepřesáhne 10 V. Skrze rezistory R_2 a R_3 se nabíjí kondenzátor C_{10} v době, kdy tranzistor Q_3 není otevřený. Dioda D_1 zajišťuje správný směr průchodu proudu kondenzátorem C_{10}. Po většinu doby je tranzistor zavřený, ale otevírá se v době, kdy napětí na diodě D_1 klesne na nulu. Jakmile se sepne tranzistor Q_3, uzavře se obvod tvořený rezistorem R_9, kondenzátorem C_{10}, přechodem tranzistoru Q_3 kolektor-emitor a LED optočlenu OK_0. Kondenzátor C_8 spolu s rezistory R_2 a R_3 tvoří filtr prvního řádu pro odstranění rušení od okolních zařízení.



obr. 24: Schéma detektoru průchodu nulou

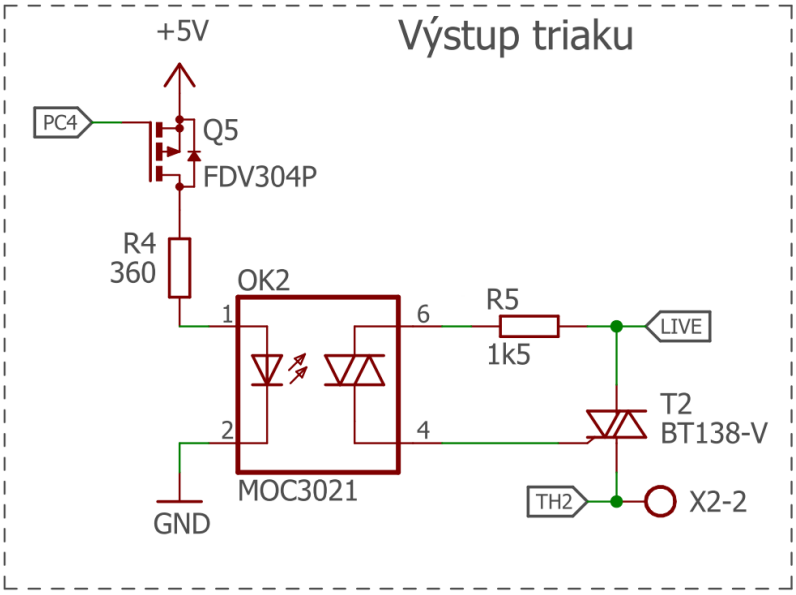
Obvod generuje pulzy o šířce přibližně 1 ms a to symetricky kolem průchodu signálu nulou, jak ukazuje obr. 25 ze simulace. Modrá křivka ukazuje průběh síťového napětí dělený 25. Fialová křivka znázorňuje výstupní napětí z optočlenu. Jelikož pulz zpracovává mikrokontrolér, tak šířka pulzu, která začíná dřív, než průchod signálu nulou, nepředstavuje problém, pouze je zpracování složitější.



obr. 25: Výstupní simulovaný signál detektoru průchodu nulou

3.7.2 Výkonová část

Triak ke své funkci potřebuje řídící obvod, který bude určovat, kdy se má tato polovodičová součástka sepnout. O rozepnutí součástky se již řídící obvod nemusí starat, jelikož střídavý proud jdoucí skrze triak prochází nulou, a právě v této době dochází k rozepnutí triaku. Řídící obvod tedy musí vědět, kdy je potřeba triak sepnout a do zátěže dodat výkon ze zbytku půlvlny. O řízení triaku se stará mikrokontrolér, který využívá pro řízení informace získané z detektoru průchodu nulou. Výstupní triak T_2 je řízen přes oddělovací optotriak OK_2, jak je vidět na obr. 26. Proud do LED optotriaku OK_2 je regulován MOS tranzistorem Q_5 a omezen rezistorem R_4. Jakmile se sepne triak v optotriaku OK_2, tak se vnutí proud přes rezistor R_5 do řídící elektrody triaku T_2, který se sepne a umožní průchod proudu, přesněji jedné půlperiody, do zátěže. Nutnou podmínkou je použít triak a optotriak, který se spíná pouze v nule.



obr. 26: Obvod pro galvanicky oddělené spínání triaku

3.7.3 Software

Ve své podstatě program zpracovává pouze informaci z detektoru průchodu nulou a dle nastavení, které přijímá ze sběrnice, spíná triak. Prvně se program musí vypořádat s určitou šířkou pulzu, který je symetrický kolem průchodu signálu nulou. Zároveň program musí počítat s tím, že od doby aktivace detektoru průchodu nulou musí počkat, než začne tento obvod správně fungovat. Latence funkce od spuštění je způsobena dobou nabíjení kondenzátoru C_{10}, která je podle simulace přibližně 200 ms. Po odeznění tohoto přechodného jevu se provede inicializace a uložení poloviny šířky impulzu do EEPROM.

Potřeba inicializace je pouze při úplně prvním spuštění modulu. Provede se 16 měření, která se zprůměrují. Číslo 16 je vybráno záměrně pro jeho jednoduché dělení, tedy pouze posun o 4 bity doprava. Jelikož pulz je symetrický kolem nuly, tak se číslo dále dělí dvěma. Podělené číslo reprezentuje dobu průchodu signálu nulou od sestupné hrany, která je generována obvodem detektoru průchodu nulou. Číslo je uloženo do paměti EEPROM a při dalším spuštění není třeba jej znovu zjišťovat.

V předchozí kapitole byla zmíněna funkce obr. 23, (6), která popisuje vztah mezi úhlem otevření triaku a poměrným dodaným výkonem. Tato funkce je příliš složitá pro přepočítávání, a proto byla vytvořena převodní tabulka o 100 hodnotách, kde každá zastupuje 1 %. Vstupním parametrem je hodnota v procentech a výstupním parametrem je doba sepnutí triaku od detekování průchodu signálu nulou. Výstupní parametr funkce (6) je úhel otevření triaku, musí se tedy přepočítat na čas v sekundách za pomocí vzorce (1). Sekundy musí být nadále přepočítány na počet tiků časovače v mikrokontroléru.

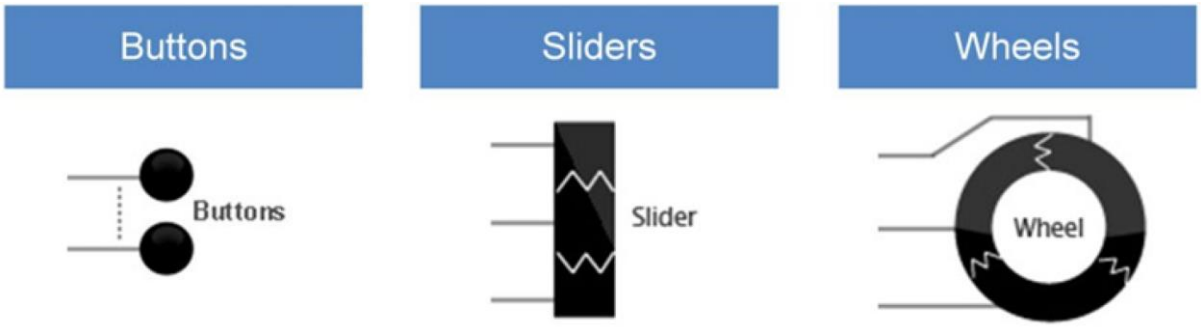
Úspora energie je zajištěna odpínáním detektoru průchodu nulou od síťového napětí v době, kdy není využíván tento modul. Nevýhodou odpínání může být latence funkce obvodu od jeho spuštění. Ovšem je upřednostněna úspora energie před latencí, která nabývá hodnot kolem 200 ms. Do úvahy také přichází myšlenka využívání detektoru průchodu nulou pouze jednou na počátku spuštění modulu a dále pouze při jeho rekonfiguraci. Frekvence síťového napětí je velmi přesná, a proto v případě stejně přesného časovače v µPC může být začátek periody počítán, nikoli měřen.

3.8 Touch panel

Dotykový panel nahrazuje konvenční mechanické rozhraní a ke své funkci nepotřebuje žádné mechanické prvky. Technologie kapacitních dotykových senzorů poskytují velkou flexibilitu jak v designovém směru, tak i ve směru funkčním. Panel může obsahovat tři typy senzorů, viz obr. 27. Prvním typem jsou jednoduchá tlačítka, která umožňují nejen rozeznat dotyk, ale i přiblížení prstem k tlačítku. Vhodnou kombinací dotykových senzorů vznikají další dva druhy senzorů. Posuvník umožňuje jednoduše nastavit například úroveň svitu světla. Senzor ve tvaru kolečka je vhodný pro nastavení barvy RGB LED pásků.

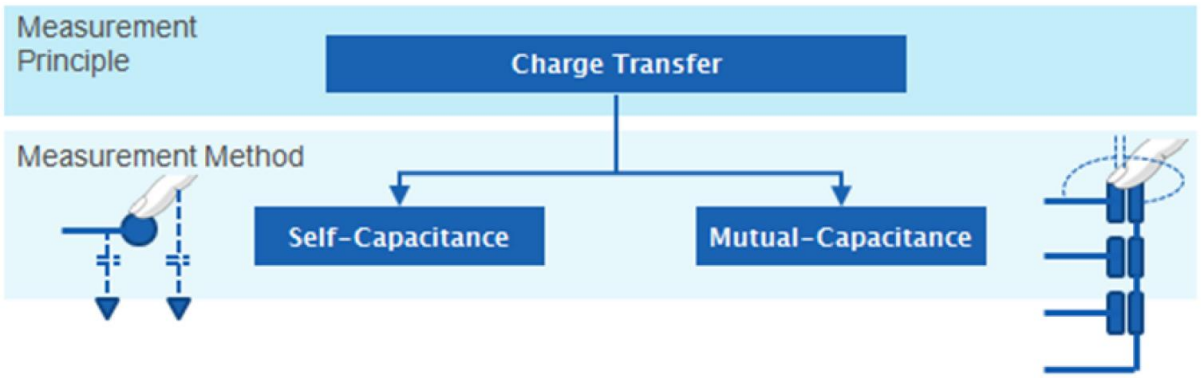
3.8.1 Atmel® Qtouch® Peripheral Touch Controller (PTC)

ATmega328PB má zabudovaný hardware pro měření změn kapacity kapacitních dotykových senzorů pojmenovaný jako Atmel® Qtouch® Peripheral Touch Controller (PTC). [12] Zároveň Atmel nabízí kompletní podporu pro tento hardware. Mezi podporu se řadí kvalitně sepsaná literatura o knihovnách a návrhu senzorů, programový průvodce pro správné nastavení knihoven, studio pro měření aktuálních hodnot na senzoru a další nastavování parametrů, pro co nejlepší vlastnosti senzorů. Při správné konfiguraci knihovny stačí programátorovi vyhodnocovat pouze dvoustavovou proměnnou reprezentující dotyk senzoru.



obr. 27:Tři typy kapacitních dotykových senzorů [12]

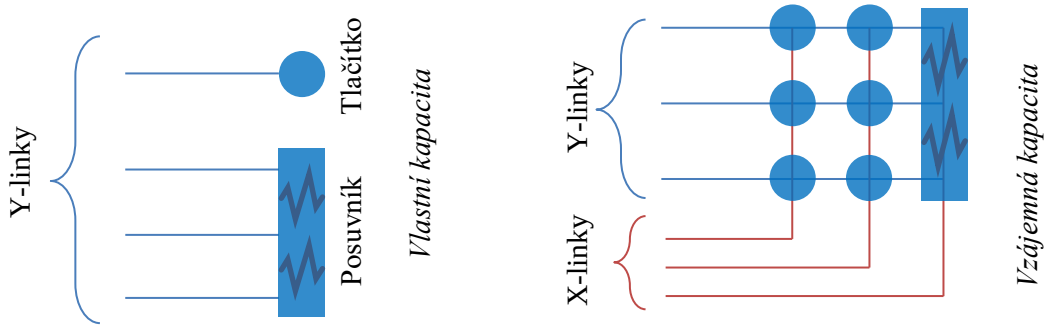
Výhodou implementace tohoto hardwaru do mikrokontroléru je možnost přímého připojení senzoru, tedy kapacitní plošky o určitém tvaru, ke vstupu µPC bez potřeby přidání jakékoliv další součástky. Zmíněná knihovna umožňuje dvě metody měření kapacity: vlastní kapacita a vzájemná kapacita, jak naznačuje obr. 28.



obr. 28: Principy měření kapacity [12]

Metoda měření *vlastní kapacity* zahrnuje nabíjení snímající elektrody neznámé kapacity vůči známému potenciálu, jak je viděno v levé části obr. 29. Výsledný náboj je přesunut do měřícího obvodu. Kapacitance senzorové plošky je určena měřením náboje v jednom či více měřících cyklech. Výhodou této metody měření je robustnost a jednoduchost vytvoření tohoto senzoru, ideální pro malý počet senzorů. [12]

Metoda měření *vzájemné kapacity* používá pár senzorových plošek, viz pravá část obr. 29. Jedna elektroda se chová jako vysílač, do kterého je posílán náboj sestávající se z logických pulzů v nárazovém režimu. Druhá elektroda se chová jako přijímač. Ta je kapacitně vázána na první elektrody pomocí dielektrika, které je tvořeno překrývajícím panelem. Jakmile se prst dotkne panelu, kapacitní vazba je utlumena, a tím zaznamenán dotek. Metoda je vhodná pro vysoký počet senzorů a lépe snáší vlhkost oproti metodě vlastní kapacity. [12]



obr. 29: Příklad zapojení kapacitních senzorů pro metodu vlastní a vzájemné kapacity

3.8.2 Návrh kapacitních senzorů

Návrh kapacitních senzorů je komplexní problematikou a spolu s ním přichází hodně vstupních proměnných ovlivňujících pak celou funkčnost senzoru. Firma Atmel nabízí průvodce [13], jak vytvořit senzory pro specifické aplikace a snaží se vysvětlit obecnou problematiku návrhu. Nejdříve se průvodce zabývá obecnou teorií kapacitních senzorů a postupně přechází k praktickým záležitostem, jako je měření správného tvaru pulzů, volba vhodného panel překrývajícího materiálu a končí praktickými návrhy kapacitních senzorů. Jelikož se většinou leptají motivy dotykových panelů na desku plošných spojů, nabízí se otázka, zda již neexistují motivy, které by byly uloženy v knihovně návrhového programu. Firma Atmel na tuhle otázku také myslela. Program Altium implementoval knihovnu s kapacitními senzory a tím byla snížena náročnost návrhu těchto senzorů na minimum. Stačí pouze správně nastavit parametry motivu. Knihovna a její prvky jsou kvalitně popsány v technické dokumentaci programu Altium, viz [14].

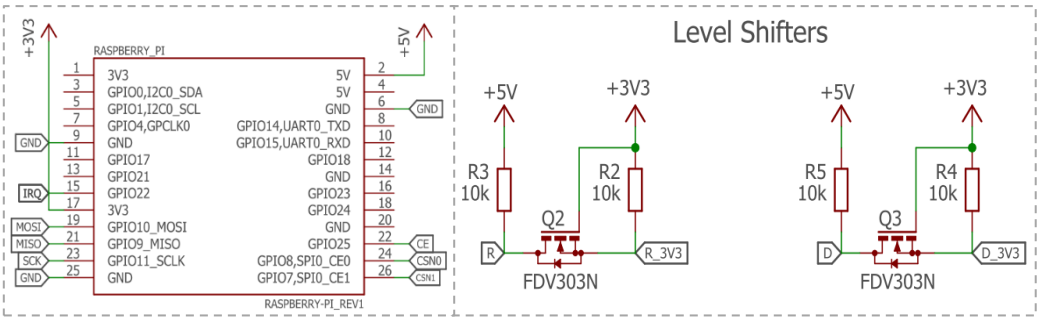
3.9 Raspberry Pi shield

Shield slouží k tomu, aby bylo možno propojit síť SID s Raspberry Pi. Nejedená se o přímé propojení sběrnice mezi kontakty RPi, jelikož není možné nastavit komunikaci COM portu RPi na parametry komunikace sítě. Parametry komunikace na sběrnici SID nejsou standardní, jelikož je využita rychlost 143 kbps a devíti bitová komunikace. Jsou způsoby, jak obejít devátý bit v komunikaci, ale s přizpůsobením rychlosti jsou již větší problémy. Navíc se nesmí stát, aby tento bod, kdy se propojuje celý systém s mozkem sítě, byl nestabilní a nespolehlivý.

Proto byl využit µPC, který nejen že převádí zprávy z komunikace na sběrnici SID na zprávy posílané přes SPI, ale také rovnou kontroluje shodu CRC8 kódů. Mikrokontrolér se chová jako malý zásobník zpráv, který by byl schopný vyrovnat rozdíly mezi rychlostí zpracování, což ovšem v případě využití Raspberry ani nepřipadá v úvahu právě tehdy, když bude napřímo připojený pouze jeden modul s SPI komunikací. Důležitou vlastností je automatické odpovídání na přijetí zprávy takzvaným ACK, a to proto, aby se snížila doba latence odezvy na minimum. Raspberry Pi tedy již neřeší posílání ACK zpráv, pouze ukládá do záznamového souboru (angl. *log file*) chybně přijaté zprávy, které již nejsou nijak vyhodnocovány. Ukládání záznamu slouží zejména při nutnosti hledání chyby v komunikaci.

3.9.1 Hardware

Hardwarová náročnost toho shieldu je minimální, mnoho nového nepřináší. Raspberry Pi disponuje konektorem, který nabízí mimo jiné GPIO piny. Níže na obr. 30 je v levé části zobrazen rozšiřující konektor s napájecími piny spolu s GPIO piny. Některé piny umožňují využít jejich specifické funkce, jako jsou: SPI, I^2C a port sériové komunikace. Na obrázku není µPC, který je připojen na napájení 3,3 V právě proto, aby byla kompatibilní s komunikační úrovní RPi. Ovšem napěťová úroveň již neodpovídá 5V úrovni čipu, který převádí napěťovou úroveň vztaženou vůči zemi na diferenční vyjádření logické úrovně. Právě proto je v pravé části na obr. 30 obvod pro obousměrné posunování napěťových úrovní. Jeden z GPIO pinů je využitý pro signalizaci přítomnosti nové zprávy v Shieldu RPi.



obr. 30: GPIO Raspberry Pi a posun napěťových úrovní

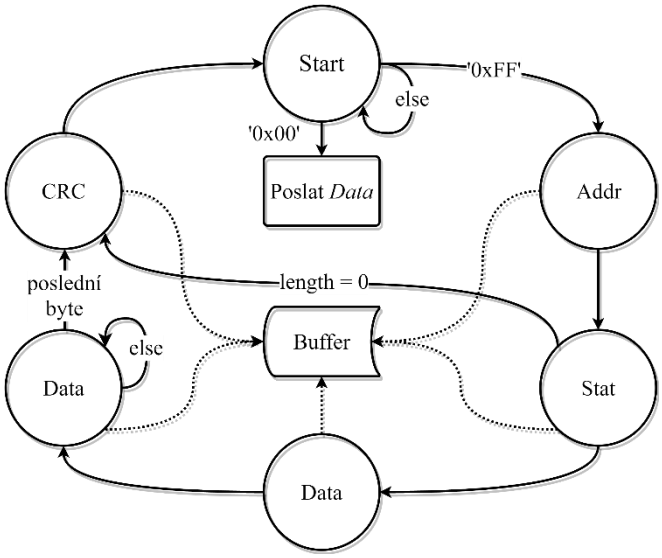
3.9.2 Software

Podstata programu je být prostředníkem mezi komunikacemi UART (strana sběrnice) a SPI (strana RPi). Obě tyto komunikace využívají ke své funkčnosti přerušení, které je vyvoláno, jakmile proběhne komunikace. UART komunikace již byla dříve popsána, proto bude rozebrána komunikace skrze SPI. Komunikace SPI je využita v komunikaci mezi RPi shield a mezi samotným Raspberry Pi.

***S**erial **P**eripheral **I**nterface*

SPI je sběrnice se synchronním sériovým rozhraním, které se používá pro komunikaci mezi jednotlivými obvody na společné desce. Není tedy využitelné pro velké vzdálenosti, což se kompenzuje vysokou rychlostí komunikace. Ve své podstatě se jedná o posuvný registr, v němž se vyměňují zprávy mezi Masterem a Slave obvodem. Výběr Slave obvodu pro komunikaci je zajištěno výběrovým pinem.

Princip přijímání zpráv je podobný s principem UART komunikace, který byl popsán v kapitole 3.3.1. Při každém zaznamenání přerušení je přijat jeden byte, který je po vyhodnocení uložen do zásobníku, kterým disponuje tento shield. Na následujícím obrázku (obr. 31) je stavový automat, jehož přechod nastává spolu s přerušením a zároveň splněním podmínky pro přechod. Není-li podmínka pro přejití do dalšího stavu udaná, automat přechází do dalšího stavu okamžitě s přerušením.



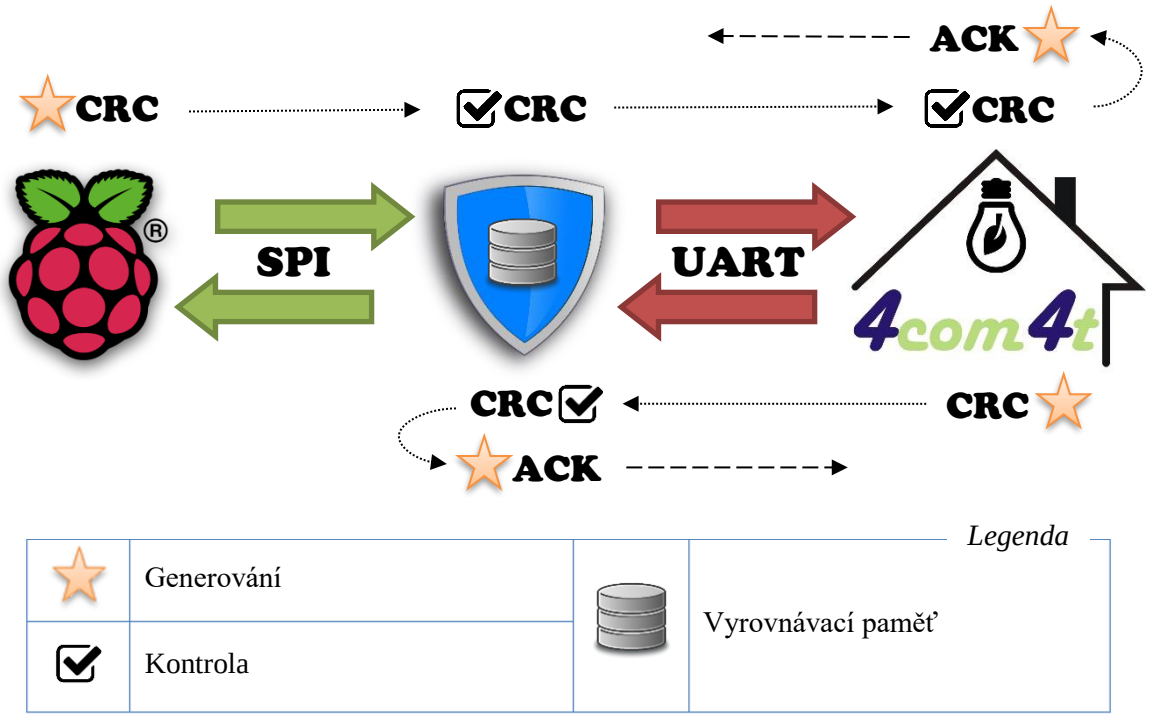
obr. 31: Statový automat pro SPI komunikaci

Stavový automat zobrazuje princip komunikace skrze SPI sběrnici. Při navrhování způsobu komunikace byl zvolen přístup, kdy se vždy jedná o jednosměrnou komunikaci, jejíž směr je přepínatelný, neboli polo-duplexní komunikace. I když SPI sběrnice umožňuje duplexní komunikaci, polo-duplexní zajišťuje vyšší spolehlivost a hlavně jednodušší ovládání komunikace. Je zde možné namítnou snižování rychlosti komunikace. Díky vysoké komunikační rychlosti SPI sběrnice a nízké rychlosti komunikace na sběrnici systému inteligentního domu, není na škodu rozdělit přijímání a odesílání dat a tím snížit prostupnost dat. V případě, kdy se automat nachází ve stavu *Start*, tak se rozhoduje, zda budou data vysílána či přijímána. Směr komunikace a zároveň oznámení jeho začátku je dáno zprávou buď *0x00* pro vysílání dat, či *0xFF* pro přijímání Slave obvodem. O vysílání dat se již stará funkce, která vyčítá data ze zásobníku. V případě, kdy není již co odesílat, pošle se nulový byte.

Rovnou při přijímání bytů jsou byty analyzovány a ukládány dle platného formátu zprávy. Analýza probíhá v kruhu stavového automatu, jehož stavy jsou:

1. *Start* – Počáteční stav, který vyčkává do doby, než je přijat první platný byte:
    a. *0x00* – Zahájení přijetí zprávy.
    b. *0xFF* – Zahájení odeslání zprávy.

2. *Addr* – Stav pouze ukládá adresa, které bude poslána tato zpráva přes sběrnici SID.
3. *Stat* – Čtení stavového rámce, ve kterém je uložena délka zprávy. V případě, kdy je délka zprávy nulová, což se může stát zejména, jedná-li se o potvrzovací zprávu, přechází automat přímo do stavu *CRC*.
4. *Data* – Podle délky zprávy tento stav přijímá data. Jakmile jsou přijaty všechny byty, automat přechází do stavu CRC.
5. *CRC* – Stav přijímá hodnotu CRC kódu, která je využita později jak při kontrole, zda nenastala chyba v průběhu SPI komunikace, tak i při ověření bezproblémové komunikace po sběrnici SID.



obr. 32: Princip kontroly zpráv srkze RPi shield

Při vyslání zprávy je pokaždé vynucena odezva ve formě ACK zprávy, která musí přijít do určité doby, nestane-li se tomu tak, je považována zpráva za neodeslanou a pokus o vyslání zprávy je opakován a to až třikrát. Jelikož interval na čekání přijetí ACK je relativně krátký, není čekáno na vyhodnocení správnosti CRC kódu modulem RPi, ale stará se o to rovnou RPi shield, jak je viděno na spodní části obr. 32. Horní část tohoto obrázku ukazuje generování CRC kódu a odezvu na něj ACK zprávou. RPi disponuje vysokým výpočetním výkonem, a proto se již v něm generuje CRC kód, který je následně kontrolován jak v RPi shieldu, tak zejména v konečném modulu v SID (poprvé uvedeno logo „4com4t“). Obrázek (obr. 32) také znázorňuje typy komunikací (SPI, UART) a mezi kterými moduly jsou používány.

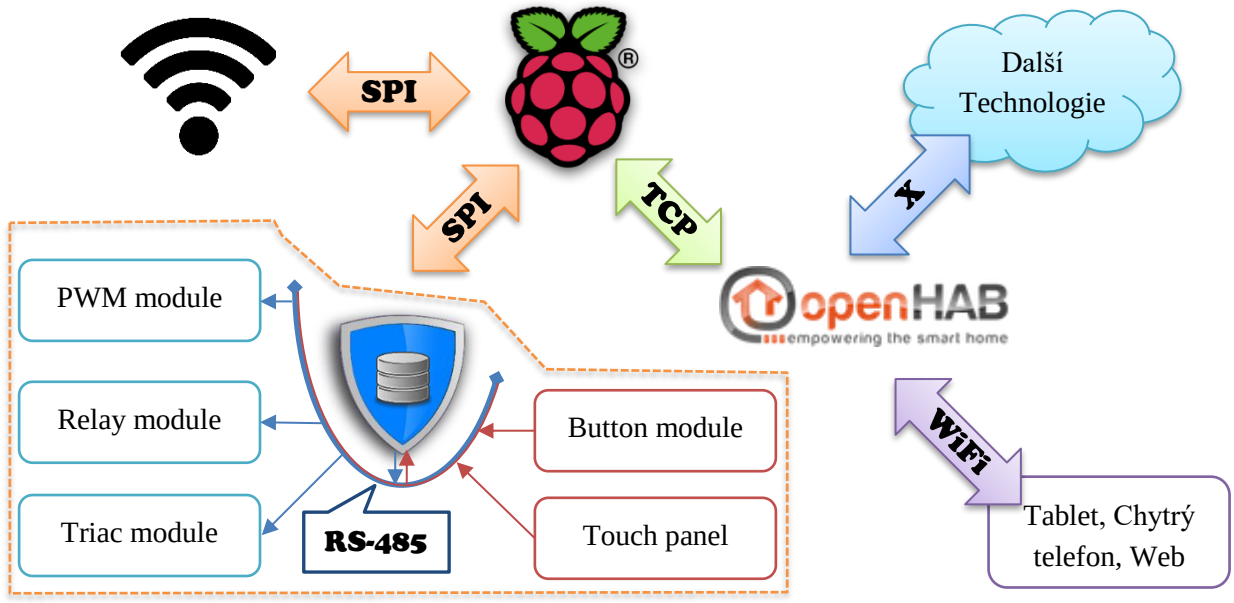
Velikost vyrovnávací paměti není nijak velká, pouze 32 bytů, proto je využit jeden GPIO pin jako přerušení pro RPi. Toto přerušení je vyvoláno RPi shieldem, který dává na vědomí RPi, že má data v zásobníku. RPi na tento podnět zareaguje a pošle zprávy na vyčtení, neboli byte *0x00*. Tento pin je nastavený do jedničky po dobu přítomnosti dat v zásobníku, což dává najevo RPi, že má neustále co číst.

4 Mozek inteligentní domácnosti

Mozkem celého SID je víceúčelový program, pojmenujme jej Master, spuštěný na jednodeskovém počítači Raspberry Pi. Tento program obstarává jak komunikaci po síti inteligentního domu, tak přijímá ovládací příkazy z intranetu.

Původně na vyhodnocování zpráv přijatých ze sítě byl využit µPC (ATmega2560), který ovšem nepostačoval svými schopnostmi a svoji kapacitou. Zejména se těžko měnila konfigurace sítě, nebylo možné jednoduše kontrolovat stav zařízení, nebylo možné ukládat zprávy. Navíc pro možnost ovládání bytu přes intranet bylo nutné využít RPi a komunikovat s využitým µPC. Pro zmíněné důvody byl µPC pro vyhodnocování událostí na sběrnici zrušen, nahrazen relativně jednoduchým Raspberry Pi shieldem a veškeré řízení domu bylo zahrnuto do jednoho programu v RPi.

Raspberry Pi, populární jednodeskový počítač, vznikl za účelem zpřístupnění výuky informačních technologií ve školách. Momentálně umožňuje levnou implementaci počítače do větších projektů, jako je například tento. Předem je také vhodné zmínit, že se nemusí přímo jednat o Raspberry Pi, ale také o Banana Pi. Co se týče modelu, tak je možné použít jakýkoliv, který nabízí alespoň jeden GPIO pin a SPI komunikaci. Raspberry Pi nabízí dostačující výkon potřebný k tomu, aby byl schopný spustit serverovou aplikaci pro umožnění ovládání SID. Druhou výhodou je malá spotřeba energie, tudíž je vhodný pro tento koncept nízké úspory energie.



obr. 33: Přehled systému inteligentního domu

Celý systém inteligentního domu není omezený pouze na Master program spolu s vytvořenými moduly na komunikační sběrnici, ale komplexní systém, jak ukazuje obr. 33. Hlavní jednotkou ucelující celý SID je právě Raspberry Pi, které nabízí spoustu rozhraní. Popíšeme-li si obrázek zleva doprava, nejprve se zmíníme o horním rohu, kde je zobrazeno bezdrátové rozhraní, které není součástí této diplomové práce. Levý dolní roh reprezentuje některé moduly z navrhovaného SID, jež komunikují přes RPi Shield s Raspberry Pi. Modré moduly reprezentují aktuátory, mezitímco červené moduly senzory. Poslední přímá komunikace s RPi je skrze TCP s programem openHAB. Tento program umožňuje další mnohá rozšíření, jak ukazuje modrý mrak skrze různé protokoly, což znázorňuje písmeno *X* v šipce. Posledním zmíněným prvkem je možnost ovládání SID zmíněnými zařízeními prostřednictvím WiFi.

Řídící program Master je napsaný v programovacím jazyce Java, pro jeho výhody přenositelnosti mezi platformami, rozšířenosti a také dobré znalosti autora této práce. Nespornou výhodou je projekt Pi4J, jež umožňuje ovládat GPIO piny v Javě, což poskytuje programátorovi přístup ke všem schopnostem RPi platformy. [8] Nejen rozhraní poskytující GPIO piny, které mimo jiné nabízí rozšířené funkce, umožňují programu komunikovat s okolním prostředím, a je tím i TCP/IP.

4.1 Databáze a objekty reprezentující prvky SID

Každý fyzický modul použitý v SID je zastoupený objektem (Java je objektově orientovaným programovacím jazykem). Objekty jsou děleny do tří základních skupin, které jsou reprezentovány jako abstraktní třídy:

- Modul – Přímo zastupuje fyzický modul, který zároveň disponuje několika *Elementy*, které jsou samostatně ovládány. *Modul* jako takový je schopný analyzovat příchozí zprávy, které jsou následně rozesílány odpovídajícím *Elementům*. Obsahuje veškeré informace o stavu jednotlivých *Elementů,* ale i nastavení pro daný *Modul*. Umožňuje namapovat právě tolik *Elementů,* kolik jich ve skutečnosti modul obsahuje. Při vytváření *Modulu* je povinné zadat jeho id neboli adresu, která je pro každý *Modul* unikátní.

- Element – Jedná se pouze o virtuální prvek, který je zahrnut do odpovídajícího *Modulu*. *Element* ve většině případu pouze čte informace z pole dat *Modulu*, ovšem ne vždy tomu tak je. Tyto *Elementy* existují zejména proto, že uživatel se dívá na koncový prvek, který ovládá danou věc, nezajímá jej, v jakém *Modulu* je element (například vypínač) obsažen. *Elementy* tedy patří do skupiny prvků, které přímo zastupují jeden reálný ovládaný či ovládací prvek.

- Jednotka – Reprezentuje reálné prvky, které existují samostatně a nemají žádné další dělení. Může se tedy jednat o teploměr, který je samostatně připojený do sítě a neobsahuje žádné další senzory či aktuátory. *Jednotka* a *Element* jsou si velmi podobné, ovšem signifikantním rozdílem je právě neexistující nadřazený *Modul,* jako tomu je u *Elementu*.

Existence abstraktních tříd je oprávněna podobnými základními vlastnostmi, ze kterých dědí třídy zastupující jednotlivé typy *Modulů*, *Elementů* a *Prvků*. Díky abstraktním třídám je možné veškeré prvky řadit do stejných polí, podle jejich typu, jinak řečeno podle toho, od jaké abstraktní třídy dědí. Abstraktní metody umožňují nad jakýmkoliv objektem, dědícím od abstraktní třídy, vyvolat metodu, která může být užitečná zejména v době, kdy je potřeba vypnout veškeré prvky atd..

Abstraktní třídy také umožňují ukládání jednotlivých položek do *Listu*, což je pokročilá třída, která se stará o ukládání objektů do pole a umožňuje nad nimi vyvolávat metody. Právě ukládání do *Listu* je využito při vytváření skupin, podle kterých jsou pak zobrazeny položky v přehledu ovládacího panelu. Kdyby nebyly *elementy* zobrazovány po skupinách, nebylo by možné se v nich jednoduše a rychle orientovat.

4.1.1 Třídy děděné z Modulu

Tyto třídy popisují reálné objekty, tedy moduly, uchovávající jejich nastavení a aktuální stavy, umožňující vygenerovat zprávy, které ovlivňují chování odpovídajícího reálného modulu a také jsou tyto třídy *Modul* schopny vyhodnocovat zprávy přicházející ze sběrnice a propagovat změny k příslušným navázaným elementům.

```
public static enum ModuleTypes {
          ButtonModule, PWMmodule, RelayModule, TriacModule
     };
```

- Enumerátor neboli výčtový typ uvádí veškeré implementované typy modulů. Slouží pro jednoduché zjištění, o jaký typ třídy modulu se jedná.

Základní třída *Modul* poskytuje všem děděným třídám následující proměnné:

```
protected byte id;
```

- Důležitá hodnota obsahující číslo, které odpovídá adrese modulu v reálné síti. Podle adresy program ví, o který modul se při příchozí zprávě jedná.

```
protected String name;
```

- Jméno modulu umožňuje přehlednější ovládání v grafickém prostředí Master programu.

```
protected TreeMap<Byte, ActionListener> actionListeners;
```

- Každý modul je schopný na sebe navázat odpovídající elementy, které implementují třídu *ActionListener*, což má za následek umožnění uložení jejich rozhraní a zavolat definovanou funkci *actionPerformed(ActionEvent e)*, která provede odpovídající akci. Všechna tato rozhraní jsou uložena v mapě, kde je klíčem bytová hodnota, která určuje, na které pozici v *modulu* se *element* vyskytuje.

Abstraktní třída *Modul* nutí implementovat následující metody:

```
abstract public void shutdown();
```

- Každý reálný modul obsahuje jednoduchou funkci pro vypnutí celého modulu. Zde vypnutí není myšleno odpojení od sběrnice, ale deaktivování výstupů. Proto byla vytvořena metoda, která je zabudována v každé děděné třídě, ovšem jedná se o metodu abstraktní, jelikož každý modul má specifickou zprávu pro deaktivování výstupů, tudíž je nutné vygenerování zprávy přenechat na každém dědici samostatně.

```
abstract public void createXML(Element e);
```

- Instance třídy vytvořené uživatelem programu jsou ukládány do XML formátu. K XML popisu třídy je připojen na vstupní parametr *e* typu *Element*, ovšem zde se jedná o „XML element“. Podrobnější rozbor ukládání do souboru bude vysvětlen později.

4.1.2 ButtonModule

Tato třída nese popis a vlastnosti o reálném modulu tlačítek, která umožňuje připojení až osmi tlačítek, což je uloženo v konstantě, jež se využívá při ukládání *Elementu* na danou pozici. Příchozí zprávy jsou pouze dvojího typu a to maska stisknutých tlačítek a maska tlačítek, která jsou delší dobu stisknuta, tedy držena. Metoda pracující s bitovými operacemi následně vyhodnocuje, která tlačítka byla stisknuta, držena a puštěna, jak ukazuje enumerátor na následujícím řádku.

```
public static enum EnumButton {Pressed, Held, Released};
```

- Díky rozdělení do výčtového typu je možné volat pořád stejnou metodu, kterou nabízí implementovaný interface, se vstupním parametrem enumerátoru, který modifikuje chování funkce.

Dále již tato třída pouze uchovává a umožňuje změnit konstantu, která říká modulu, od kolika milisekund je držení tlačítka vyhodnoceno již jako drženo. Pro změnu konstanty je potřeba u daného modulu držet tlačítko. Jakmile modul vyhodnotí tlačítko jako dlouze držené, je možné změnit jeho konstantu držení. Tato komplikace je způsobena tím, že µPC v době klidu, tedy v době, kdy nejsou mačkána tlačítka, spí a žádné zprávy ze sítě nepřijímá. Tento modul, jako jediný, pouze loguje informaci v případě vyvolání metody *shutdown()*, jelikož se jedná o senzor, který po většinu času spí a neodebírá téměř žádnou energii ze sítě.

4.1.3 PWMmodule

PWM modul je schopen pojmout až šest připojených elementů, které jdou nezávisle na sobě ovládat. Nezávislost je myšlena pouze co se týče střídy PWM signálu. Ostatní parametry PWM signálu jsou již společné, jako je jeho frekvence. Již v dřívějších kapitolách byla představena myšlenka potřeby vysoké frekvence PWM signálu. *TriacModule* je programově téměř totožný s *PWMmodule*.

```
private final static byte gammaCorrection[] = {0, 1, 1, 1, 2,… }
```

- Stejně tak byla uvedena problematika nelineární závislost dodávaného výkonu na svítivosti LED pásků, proto tato třída disponuje polem *gammaCorrection*, které obsahuje sto prvků a k nim odpovídající hodnotu v rozsahu od 0 do 255. Tedy jedná se o převod mezi procenty a mezi osmibitovou hodnotou PWM střídy.

PWM modul umožňuje plynule rozsvěcovat světla, tedy postupně zvyšovat hodnotu PWM, a to v různě velkých krocích, pro postupný přechod při držení tlačítka a při přechodu svítivosti na určitou úroveň. Tyto dva parametry si také ukládá softwarový model, *PWMmodule*, u kterého je možné tyto parametry číst, ale i měnit. Rychlost, jak se mění svítivost při obou módech, je již společná a nastavena v další proměnné.

```
private byte[] brightness = new byte[6];
```

- Svítivost jednotlivých elementů je uložena v poli, k němuž může přistupovat navázaný element. Zajisté element má přístup pouze k hodnotě v poli s odpovídající pozicí. Svítivost lze nastavit jak v procentech, tak v přímé PWM hodnotě (0 – 255). Pole ukládá vždy přímou PWM hodnotu. Je-li potřeba přepočítat PWM hodnotu na procenta, již není možné použít přímý pří-

stup přes pole, ale je potřeba v něm hledat nejbližší hodnotu. Jelikož je pole seřazené od nejmenších hodnot po největší hodnoty, tak je možno v něm použít vyhledávací algoritmus bisekce, který byl představen v kapitole 3.5.4.

4.1.4 RelayModule

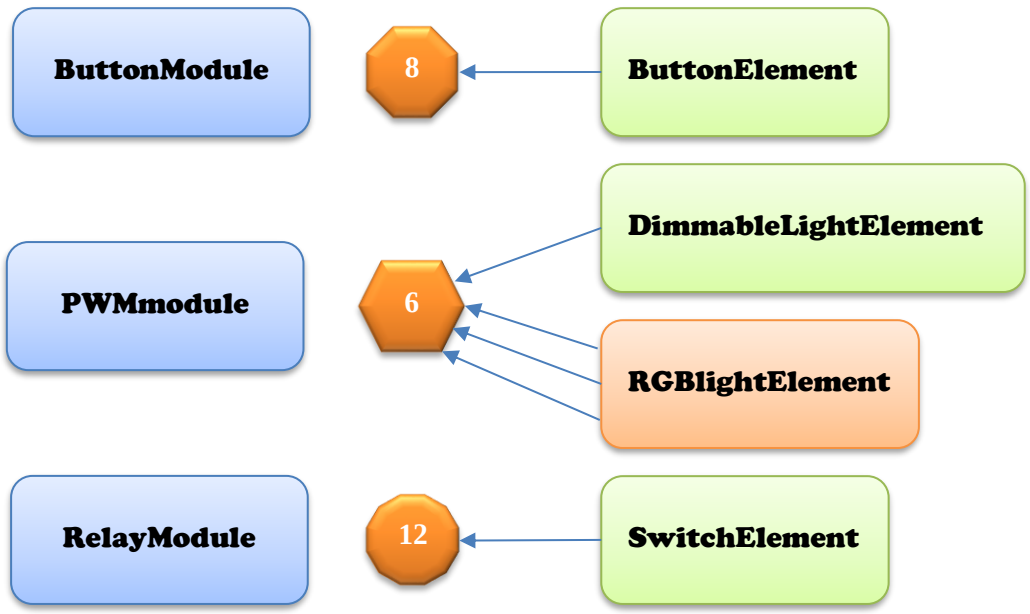
RelayModule obsahuje dvanáct volných pozic pro obsazení *elementy*, proto třída obsahuje pole o dvanácti prvcích, jež udávají, zda jsou relé sepnutá a jejich časovou konstantu. Zde je časovou konstantou myšlena doba, za jak dlouho se relé samo rozepne, pokud je zapnuto s příznakem časování. Dále již třída, jako všechny ostatní, umožňuje posílání zpráv a jejich analýzu. Jelikož 12 pozic je již vyšší číslo, třída pracuje s bitovými maskami, jež udávají, který z elementů je sepnutý. Maska je již dvoubytová.

4.1.5 Třídy děděné z Elementu

Třídy popisující *elementy*, které je možné připojovat k *modulům* na jejich odpovídající volné pozice. Každý *element* koresponduje s *modulem* a to podle obr. 34.

```
public static enum ElementType {
          ButtonElement, DimmableLightElement, RGBlightElement,
          SwitchElement
     };
```

- Stejně jako třída abstraktního *modulu* třída abstraktního *elementu* obsahuje výčtový typ všech možných typů elementů.



obr. 34: Přehled Modulů a možnosti připojení Elementů

Základní třída *Element* poskytuje všem děděným třídám následující proměnné:

```
protected String name;
```

- Jméno každého *elementu* slouží k lepší orientaci v grafickém prostředí programu Master.

```
protected ModuleType module;
```

- Ač by se na první pohled mohlo jednat, že se jedná o třídu *ModuleType,* tak taková třída neexistuje. Jedná se o generickou proměnnou, která umožňuje vložení do proměnné *module* jakoukoliv třídu, která dědí abstraktní třídy *Module*. Jinými slovy se musí jednat o modul, který byl v předchozí podkapitole 4.1.1 popsán.

```
protected ListenerType listener;
```

- Tento případ je podobný, jak bylo popsáno výše, znovu se jedná o generickou proměnnou, která tentokrát musí dědit od třídy *EventListener*. V jednoduchosti lze říct, že každý modul má svůj interface *EventListener,* který umožňuje zpětně propagovat události do modulu, na který je daný element vázaný.

```
protected boolean isOn;
```

- Každý *element* je nucený implementovat binární proměnnou, která říká, zda *element* je aktivní či nikoliv. Slouží pro rychlou kontrolu, zda jsou všechna zařízení vypnuta a je možné zjistit tento stav nad jakýmkoliv *elementem*.

```
protected byte position;
```

- Důležitá proměnná, která spolu s generickou proměnnou jednoznačně identifikuje *element* odpovídající jednomu reálnému elementu na reálném modulu. Proměnná *position* tedy říká, na jaké pozici *modulu* (proměnná *module*) se *element* nachází.

- Konstruktor této abstraktní třídy je napsaný tak, že jakmile se zadá *modul* a pozice, tak je daný *element* navázaný na *modul*. Odpadá tedy nutnost myslet na navázání komunikace mezi *modulem* a *elementem* skrze interface *EventListener*. Navíc při vytváření *elementu* je pokaždé kontrolováno u vázajícího *modulu*, zda je zvolená pozice volná. V případě, kdy není pozice volná, je vyvolána výjimka o neplatném argumentu.

Abstraktní třída *Element* nutí implementovat a implementuje následující metody:

```
public abstract JSONObject getJson();
```

- Nutná funkce pro komunikaci mezi programem Master a serverovou aplikací, přesněji se jedná pouze o směr od programu Master k serveru. Funkce vygeneruje popis daného elementu, zahrnující stav elementu spolu s konfigurací, má-li nějakou.

```
public abstract void turnOff();
```

- Jedná se o podobnou funkci, jaká byla viděna u modulu, tedy jakýkoliv element je možné vypnout, nebo alespoň je možné vyvolat nad ním tuto funkci.

```
abstract public void createXML(Element e);
```

- Druhá podobná funkce, kterou implementuje i jakýkoliv *modul*, tedy vygenerování XML popisu pro uložení všech parametrů instanciovaného *elementu*.

```
public Integer getCombinedID() {
     return (getModule().getiD() & 0xFF) << 8 | (position & 0xFF);
}
```

- Nad všemi třídami *element* je možné vyvolat tuto metodu pro získání identifikačního čísla, které jednoznačně určí *element* vázaný k *modulu*. Jak je vidět, kombinované identifikační číslo se skládá z ID *modulu* a z pozice *elementu,* které je spojeno bitovým posunem do proměnné typu *Integer*, tedy celého čísla (32 bitů).

4.1.6 ButtonElement

Nemusí se to zprvu jevit jako samozřejmé, ale právě tlačítko patří mezi nejsložitější *element* ze všech. Je tomu tak, jelikož je k němu možné navázat hned několik jiných *elementů*. Nejprve budou uvedena jednoduchá vázání, tedy na *element PWMmodul*u a *element RelayModul*u.

```
private DimmableLightControl dimmableLightControl;
private SwitchControl switchControl;
```

- Pro vyšší efektivitu byla vytvořena dvě základní vázání na tlačítko, které vychází z praktického pozorování, tedy z nejpoužívanějšího vázání na tlačítko. Je možné navázat *DimmableLightElement*, který má výchozí vázání k tlačítku, stejně tak jako má *SwitchElement*.

Pro připomenutí, existují tři různé stavy, které mohou nastat u tlačítka, je to *stisknutí*, *držení* a *puštění* tlačítka. Pro přehled je uvedena tab. 6, která ukazuje výchozí vázání *elementů* na tlačítko při dané stavu. Zde je nutné vysvětlit, že *stisknuto* je myšleno stisknutí tlačítka a jeho puštění do doby, než je považováno jako dlouze držené tlačítko. Dá se tedy říct, že program nikdy nereaguje okamžitě na prvotní stisknutí tlačítka, ale až na jeho puštění v závislosti délky držení tlačítka. Událost puštění tlačítka v tab. 6 je vyhodnocena jen v případě, když bylo tlačítko drženo.

tab. 6: Přehled výchozího vázání elementů na tlačítka

Element	**Stav tlačítka**		
	Stisknuto	*Drženo*	*Puštěno*
DimmableLightElement	Přepnutí stavu světla.	Začátek postupné změny svítivosti. Směr udává dosažení jednoho z limitů svítivosti.	Ukončení změny svítivosti a ponechání nové hodnoty svitu.
SwitchElement	Časové sepnutí relé. Je-li relé již sepnuté, vypne se.	Sepnutí relé bez časovače. Je nutné, aby bylo vypnuto uživatelem.	

Základní vázání není dostačující, proto je možné vytvořit mnoho dalších. Další vázání jsou udělána přes *Listener*, který obsahuje právě tři metody odpovídající třem stavům tlačítek, jež umožňují vyvolání jakékoliv metody. Ovšem je potřebné, aby bylo možné tyto metody přiřazovat v průběhu programu. Nabízí se tedy možnost využití ukazatelů na funkce, ale to Java neumožňuje. Druhým řešením, které již Java nabízí, jsou reflexe, ale jejich volání je neefektivní. Proto byly vytvořeny třídy, které dědí ze třídy *ButtonBinding*, která implementuje *ButtonListener*. Díky složitější hierarchii je možné pak přiřazovat do proměnné typu *ButtonListener* jakýkoliv počet tříd, z nichž každá zastupuje užitečné funkce. Prozatím byly vytvořeny následující třídy zastupující funkce:

- *ElementOff*
  - Vypnutí jakéhokoliv *elementu*.
- *ModuleOff*
  - Vypnutí jakéhokoliv *modulu*.
- *StartDimmingLights*
  - Spuštění plynulé změny svitu libovolného počtu stmívatelných světel. V případě, že skupina vybraných světel má rozdílný jas, vždy se nastaví počáteční hodnota dle prvního svitu světla.
- *StopDimmingLights*
  - Zastavení plynulého stmívání libovolného počtu stmívatelných světel.
- *SwitchOn*
  - Sepnutí libovolného počtu relé bez časování.
- *ToggleDimmableLights*
  - Přepne stav libovolného počtu stmívatelných světel. Svítí-li světlo na jakékoliv úrovni, zhasne. Je-li světlo zhasnuté, rozsvítí se naplno.
- *ToggleSwitches*
  - Sepnutí libovolného počtu relé. Relé sepnutá touto funkcí jsou časovaná.

Jakmile budou zjištěny během používání tohoto systému další potřebné funkce, jednoduše se dopíší a rozšíření funkčnosti tohoto systému tedy bude kdykoliv možné. Jednotlivé funkce se přidávají pod určité stavy tlačítek s tím, že nemusí být využity všechny stavy tlačítek. Nejčastěji se přiřazují tyto jednoúčelové funkce pod stav stisknuto.

4.1.7 DimmableLightElement

Element přiřazující se k třídě *PWMmodule* a zabírá jednu pozici na tomto *modulu*. *Element* uchovává hodnotu, zda je rozsvěcování světla na danou hodnotu okamžité, či postupné. Tato hodnota je uložena v bitové proměnné. Druhá bitová proměnná ukládá směr postupného rozsvěcování světel. Směrem je myšleno, zda má být jas zvyšován při jeho změně, či snižován. Směr se vždy přepíná po dosažení krajních hodnot svitu. Informace o svitu jsou již přímo navázané na třídu *PWMmodule*.

4.1.8 RGBlightElement

Element se taktéž navazuje na třídu *PWMmodule*, ovšem zde již zabírá tři pozice, pro každou barvu jednu. Pozice červené barvy je ukládána do proměnné *position* v abstraktní třídě *Element*.

```
private byte[] rGBposition = new byte[3];
```

- Pozice jsou uloženy v poli o třech bytech. Při jejich přiřazování jsou vždy kontrolovány všechny pozice, je-li nějaká již obsazená, program vyvolá výjimku indexu mimo hranice.

```
private boolean positionsInOrder;
```

- Jelikož program generuje zprávy zasílané do reálného modulu, je pro něj výhodné vědět, zda jednotlivé pozice barev jsou uspořádány bezprostředně za sebou, tedy červená, zelená a modrá. Je-li tato podmínka splněna, zprávy vysílané po sběrnici jsou kratší, jelikož je poslána pozice pouze červené barvy a zbytek pozic je již samozřejmý.

4.1.9 SwitchElement

Element navazující se na třídu *RelayModule* umožňuje adresovat a jednoduše přistupovat k jednotlivým pozicím tohoto *modulu*. Tento *element* vyčítá pouze informace z příslušného pole *RelayModule*, jehož pozice odpovídá pozici v poli.

```
private boolean timed;
```

- Proměnná pouze udává, zda má tento *element* povolené časované vypnutí. Některé *elementy*, například v případě připojení zásuvkového okruhu, není vhodné časovat, spínají se tedy na dobu omezenou čistě uživatelem.

```
private SwitchModificator modificator;
```

- Proměnná zejména slouží pro grafický výstup programu. Udává, jaké zařízení je připojené a tedy jaká ikona má být přidružena k danému *elementu*.

4.1.10 Skupiny elementů

Třída skupin, *ElementGroup*, umožňuje složité třídění skupin do stromové struktury. Jelikož Java ve svém základu nenabízí kolekci, která by umožňovala ukládání do stromové struktury a volné pohybování se v ní, tak bylo nutné tuto kolekci napsat pro účely tohoto *elementu*. Pohybováním je myšleno čtení potomků, rodičů, kořene a dalších rolí v této datové struktuře. Proto byla datová struktura přímo napsána a vložena do třídy *ElementGroup*. Nejedná se o binární strom, kdy umožňuje strom pouze dvě větve, ale tento strom je teoreticky neomezený ve větvení, což mírně zhoršuje manipulaci a čtení dat.

```
private ElementGroup parent;
```

- Ukládání pozice nadřazeného záznamu, tedy rodiče. Díky této proměnné je možné jít stromem až ke kořeni, tedy k úplnému základu, odkud je strom větvený.

```
private final List<ElementGroup> children = new ArrayList<>();
```

- Neomezený počet větví, označovaných jako dětí, je ukládán do kolekce typu List, což je základní kolekce bez jakýchkoliv speciálních metod.

```
private List<DefaultElement<? extends DefaultModule, ?>> elements;
```

- První dvě zmíněné proměnné reprezentovaly čistě stromovou strukturu, zatímco tato proměnná reprezentuje konečně skupinu elementů. Tedy jedná se znovu o jednoduché pole, které ukládá teoreticky neomezený počet elementů. Zde je možné vidět výhodu dědění všech tříd elementů z jedné abstraktní třídy *DefaultElement*. Díky této skutečnosti je možné uložit do kolekce různé třídy.

4.1.11 Třída Database

Třída uchovává veškeré *moduly* a prvky v kolekcích typu TreeMap. Tento typ kolekce implementuje rozhraní Map za využití stromové struktury ukládání položek v kolekci. Nejdůležitější vlastností a vlastně i důvodem použití kolekce TreeMap je efektivní ukládání páru klíč & hodnota. Tyto páry jsou vzestupně seřazené dle klíče, který je unikátní. Tyto parametry umožňují rychlý přístup k prvkům prostřednictvím klíče.

Všechny třídy jsou ukládané i vícenásobně, respektive jejich adresy, nikdy se nejedná o kopii záznamu. Veškeré elementy jsou uložené v kolekci *elements* a jejich klíče jsou generovány jako kombinované ID. Následně jsou elementy roztříděny dle typů do kolekcí, aby k nim byl rychlejší přístup v případě, kdy program ví, jaký typ elementu má při volání metody vrátit.

Třídy děděné z abstraktní třídy *module* jsou ukládány do kolekce *modules*, kde klíčem je adresa, tedy v programu ID proměnná daného modulu. Stejně jak jsou elementy roztříděny do kolekcí podle typu elementu, je tomu tak i u modulů.

Také skupiny mají své identifikační číslo, proto jsou ukládané také do kolekce typu TreeMap. Jelikož tyto skupiny nejsou nijak omezeny na typ *elementu* (ano, mohou ukládat pouze *elementy*, nikoliv *moduly)*, tak nepodléhají žádnému dalšímu třídění. Databáze obsahuje i další kolekce potřebné ke správnému chodu programu, ale tyto kolekce již jsou nad rámec tohoto textu.

Databáze umožňuje také jednoduše vytvořit veškeré *moduly* a *elementy*, které jsou rovnou uloženy do odpovídajících kolekcí v databázi, není tedy mít na mysli při vytváření nových objektů ukládání prvků do kolekcí v databázi. Kdyby se stalo, že daná třída není vytvořena přes třídu databáze, objekt by sice vznikl, ale program by jej již nikdy nenašel, protože vždy se přistupuje přes třídu databáze, která obsahuje veškeré vytvořené objekty SID. Veškeré metody pro vytvoření *elementu* či *modulu* vždy při vytvoření objektu daný objekt vrací, je tedy možné si uložit jeho adresu do proměnné a volat nad ní další funkce, či například přiřazovat do skupin *elementů*.

4.2 Příjem a vysílání zpráv na sběrnici

Zpracovávání zpráv se děje ihned po přijetí celé zprávy a opětovnému zkontrolování CRC kódu. Není-li CRC kód správný, zpráva se pouze eviduje jako chybná pro pozdější analýzu chyb přenosu. O odeslání potvrzovací zprávy ACK o korektním přijetí zprávy se již RPi nestará, jak již bylo řečeno. Ovšem odesílá-li RPi zprávu, ACK již očekává. Veškerá komunikace probíhá skrz Raspberry Pi Shiled, ale pro větší přehlednost bude v textu vždy uváděna přímá komunikace.

4.2.1 Příjem a analýza zpráv

První krok k přijetí zprávy je přečtení stavu na GPIO pinu, který je vstupem pro RPi shield. Tento stav udává, zda jsou nějaké zprávy připraveny na přijetí ze zásobníku v RPi shieldu. GPIO pin je konfigurován jako přerušení. Jakmile se vyvolá přerušení na GPIO pinu, zahájí se vyčítání hodnot z RPi shieldu, což znamená posílání nulových bytů do doby, než je celá zpráva přijata. Každý byte prochází přes vyhodnocení stavovým automatem, který je podobný stavovému automatu pro SPI komunikaci na obr. 31. Procházením bytů skrze stavy ve stavovém automatu se průběžně počítá CRC kód, který je rovnou porovnaný s posledním přijatým bytem. Jsou-li tyto kódy stejné, je celá zpráva poslána na zpracování do třídy *SwitchBoard* a do grafické komponenty, je-li inicializovaná, zobrazuje komunikaci po sběrnici. Grafický výstup slouží spíše k ověřování funkčnosti sítě při jejím nastavování.

Třída *SwitchBoard* se jednoduše postará o roztřídění zpráv podle stavového bytu ve zprávě, která nese informaci, od jakého typu zařízení zpráva přišla. Následně třída podle typu modulu adresuje v databázi modul se správnou adresou a přepošle mu data, která přijala ze sítě. V případě, kdy by v síti byl prvek, který nebyl zanesen do programu, nebo by přišla adresa s chybou a zároveň by nebyla detekována, tak program pouze eviduje chybnou adresu, a tím končí její zpracování. V případě správné adresy existujícího zařízení se propaguje do modulu.

Propagace zprávy k danému modulu je přímá a předáním vhodných bytů z původní zprávy modulu, který již zahrnuje metodu, změní svůj stav podle přijatých bytů. Každá třída modulu má další kontrolu správnosti formátu přijaté zprávy. Není-li formát zprávy shodný s nastaveným formátem zprávy, zpráva se pouze eviduje, čímž končí zpracování přijaté zprávy.

4.2.2 Odesílání zpráv

Zpráva může být vyslána pouze z třídy *module*, to zda je vyslání iniciováno *elementem* či *modulem*, na tom již nesejde. Již dříve bylo předesláno, že každý element je navázaný na modul a element představuje jakýsi virtuální prvek, který zjednodušuje ovládání celé třídy *module*. Jakýkoliv úkon nad elementem se propaguje do modulu, který již odesílá zprávu na sběrnici. Třída *module* má předdefinované metody, které odpovídají funkcím reálného modulu. Jedná se tedy pouze o překladač mezi událostmi v programu na události popsané bytovou zprávou. Mezi dalšími odesílanými informacemi jsou konfigurační zprávy, které popisují změnu parametru uživatelem.

Zpráva vygenerovaná modulem není přímo posílaná do sítě, ale nejdříve je spočítán CRC kód a přidán ke zprávě. Následně se zpráva odešle přímo na sběrnici, není-li třída pro odesílání zaneprázdněna dříve odesílanou zprávou. V opačném případě je zpráva uložena do zásobníku, který odesílá zprávy podle toho, do jakého modulu má být zpráva doručena a zda je s tímto modulem ukončena předchozí komunikace. Ukončená komunikace je v případě, že bylo přijato ACK. V případě neukončené

komunikace je zpráva zařazena do čekací fronty a odeslána ihned po ukončení předchozí komunikace. Zvolený způsob odesílání pouze jedné zprávy jednomu modulu je kvůli povaze potvrzovací ACK zprávy. Tato zpráva pouze potvrzuje, že daný modul přijal zprávu, ale nezpravuje Master program o tom, jaká zpráva byla přijata. Odeslaly-li by se dvě zprávy najednou, modul by potvrdil přijetí zprávy ACK zprávou, ale Master by nevěděl, které z odeslaných zpráv přiřadit odpověď ACK, což se nemůže z důvodu spolehlivosti stát.

4.3 Rozhraní programu

Program disponuje několika vstupy a výstupy, jež je možné nazvat rozhraními:

- SPI rozhraní je nejdůležitějším prvkem, jelikož skrze něj program ovládá SID.
- Grafický výstup aplikace na zařízení, na němž je spuštěna. Parametrem při spuštění aplikace je možné si vybrat, zda bude grafické rozhraní k dispozici.
- Podpora TCP/IP, skrze nějž je možné ovládat aplikaci programem openHAB.

4.3.1 SPI

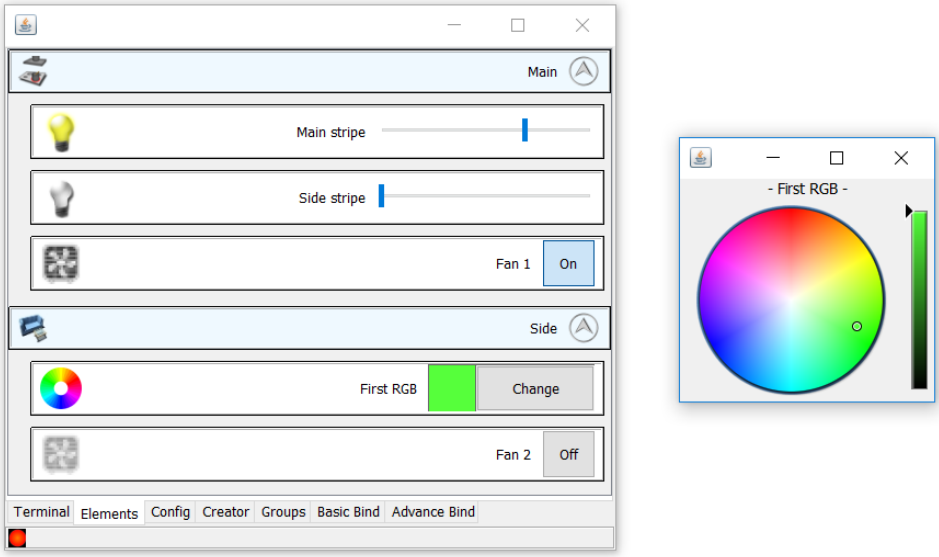
Zprostředkování přečtení informací z SPI komunikace a stavu na GPIO zajišťuje knihovna z projektu Pi4J. Přímá SPI komunikace s programem v Javě umožňuje jednoduchou implementaci této knihovny. Z principu SPI komunikace je jasné, že stačí pouze jednoduchá metoda, která je schopna vyslat data a zároveň nová přijmout. Sama o sobě tato komunikace nepodporuje přerušení v případě, kdy podřízená komponenta, Raspberry Pi Shield, má připravená data na odesílání. Bylo by tedy nutné v pravidelných intervalech ověřovat tento fakt.

Jelikož RPi nabízí GPIO piny, na než se dá navázat přerušení. Toto přerušení bylo využito k tomu, aby nebylo třeba neustále periodicky zjišťovat přítomnost nové zprávy. Program tedy reaguje na vyvolání přerušení, čímž ví, že jsou přítomny zprávy v zásobníku Raspberry Pi Shield. Program Master se postará o co nejrychlejší odbavení této potřeby. Jelikož se může stát, že při odesílání zprávy může být vyvoláno přerušení, je na každém začátku přerušení zakázáno a po odeslání celé zprávy přerušení zpětně povoleno. I když SPI nabízí příjem a odesílání zpráv současně, nebylo toho využito, aby byla komunikace spolehlivější. Zvýšila se tím spolehlivost a rychlost komunikace nijak neutrpěla, jelikož rychlost komunikace SPI je mnohonásobně vyšší oproti rychlosti komunikace na síti.

4.3.2 Grafické rozhraní

Grafický výstup programu bude používán zejména konfigurátorem sítě inteligentní domácnosti. Program nabízí různá podokna, která disponují mnoha možnostmi. Jsou to tato okna:

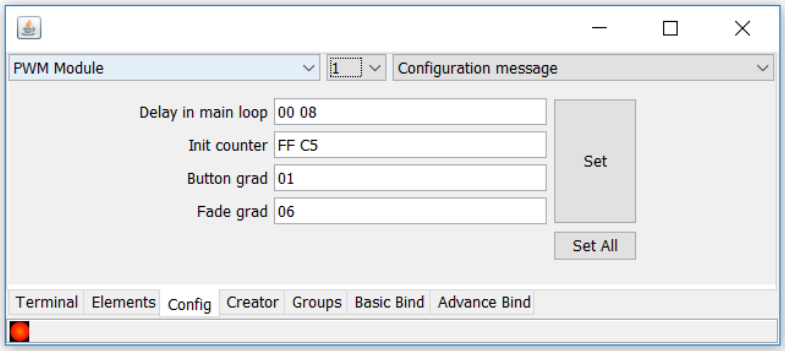
- *Terminal* – Okno slouží k přehledu příchozích zpráv, ale také možnosti přímého odesílání uživatelských zpráv v hexadecimálním formátu. Uplatní se zejména při prvním kroku oživení SID.



obr. 35: Seskupené interaktivní elementy; vpravo výběr barvy pro RGB elementy

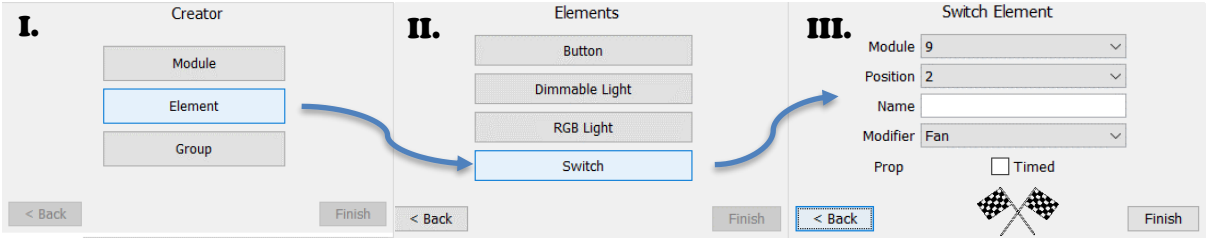
- *Elements* – Grafické zastoupení všech elementů, které byly seskupeny do skupin dle jejich umístěni v bytě. Na obr. 35 je příklad, jak může vypadat dělení do dvou skupin *Main* a *Side*. Jednotlivé skupiny lze sbalit a rozbalit kliknutím na kulaté tlačítko vpravo se šipečkou, která je při změně náhledu animovaná, stejně jako je animované sbalení celé skupiny. Popíšeme si jednotlivé řádky od shora.
  - První řádek reprezentuje skupinu *Main*, která má ikonku kuchyně (digestoř a varná deska).
  - *LightDimmableElement* je na druhém řádku, u kterého je možné měnit svítivost světla pomocí posuvníku v pravé části řádku, také vidíme, že tento *element* se jmenuje *Main stripe*.
  - Třetí řádek reprezentuje stejný typ *elementu*, jen dodejme, že podle nastavení úrovně svitu se mění ikonka v 10% krocích, může se tedy zdát, že je ikonka také animovaná. Posledním slovem o tomto typu elementu řekněme, že se zpráva o nastaveném svitu odešle až po uvolnění posuvníku.
  - Ve čtvrtém řádku vidíme *SwitchElement*, který má zde ikonku větráku, může mít ale také ikonku světla, to již záleží na tom, jaký modifikátor je nastavený pro tento *element*. Zde je využito pouze tlačítko pro ovládání *elementu*. Zdůraznění sepnutého stavu je vytmavením ikonky a modrého podkladu tlačítka.
  - Dostáváme se do druhé skupiny s názvem *Side*, která má zastupovat například obyvák pro její ikonku sofa. První položkou, tento typ ještě nebyl zmíněný, je *RGBlightElement*, který je vlevo znázorněn kruhem barev. V pravé části tohoto řádku je tlačítko

pro změnu barvy *Change* a vlevo od něj je okénko s právě zvolenou barvou. Samozřejmostí je název elementu *First RGB*. V případě, kdy je potřeba přímo vypnout RGB světlo, bylo by zbytečně pracné a nepohodlné hledat černou barvu, proto při kliknutí na náhled barvy se RGB světlo vypne. Je-li světlo vypnuté, naopak se rozsvítí do bílé barvy, při kliknutí na náhled nastavené barvy (černá = off).



obr. 36: Konfigurační okno programu Master

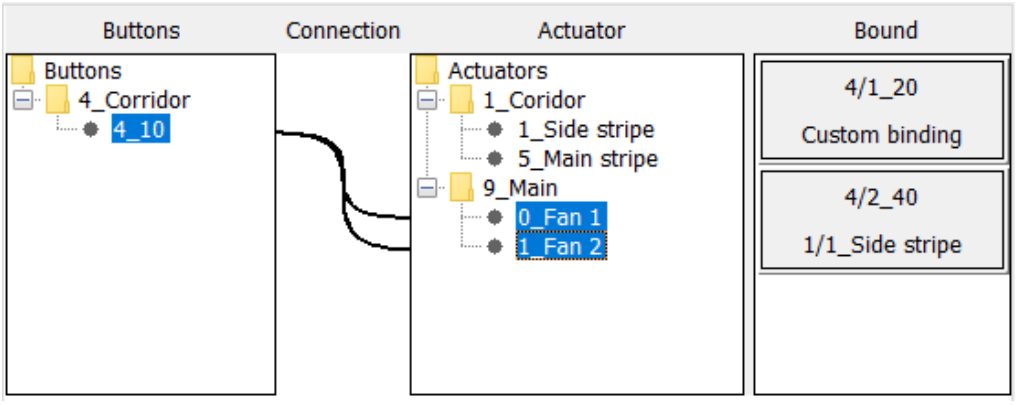
- *Config* – umožňuje konfigurovat veškeré *moduly*, které byly přidány do programu. První řádek v tomto okně je věnován volbě *modulu*. První položkou je typ *modulu* - na obr. 36 vidíme *PWM module*. Druhou položkou je ID, nebo chcete-li, jeho adresa. A konečně třetí a poslední položkou v tomto řádku je typ nastavení. Typ nastavení je odvozený od konfiguračních zpráv a je závislý na zvoleném *modulu*. Po zvolení těchto náležitostí se objeví kolonky na vyplnění vstupních parametrů, které mají již předvyplněnou hodnotu dle současného nastavení. Tlačítkem *Set* je vyslána konfigurační zpráva vybranému ID *modulu*. Stiskne-li se tlačítko *Set All*, konfigurační zpráva se pošle všem modulům na síti.



obr. 37: Příklad vytvoření nové komponenty (elementu)

- *Creator* – slouží k vytváření nových modulů, elementů a skupin. Tato záložka se chová jako průvodce, přes kterého je možné se doklikat až ke vstupním parametrům nově vytvářené instance. Na obr. 37 je vidět příklad, jak ve třech krocích lze vytvořit *SwitchElement*. V prvním kroku je zvolen prvek *Element*. Druhé okénko nabízí všechny typy *elementů* nabízející se k vytvoření. Byl vybrán *Switch*, čímž se otevřela nabídka pro výběr parametrů pro tuto komponentu. Většina komponent má podobný grafický základ. První je výběr *modulu*, na který chceme vytvářený *element* navázat. Druhé výběrové okénko nabízí všechna volná ID instancí *SwitchModule*. Druhé okénko se mění v závislosti na volbě prvního, tedy jaká instance byla vybrána, a dá uživateli na výběr ze všech volných pozic vybraného *modulu*. V grafickém vytváření *elementů* se tedy nemůže stát, že by byl *element* přidán na již zabranou pozici *modulu*, stejně tak grafické prostředí pohlídá, aby *element* nebyl navázán na neexistující *modul*. Další parametry jsou již pro každý *element* jedinečné.

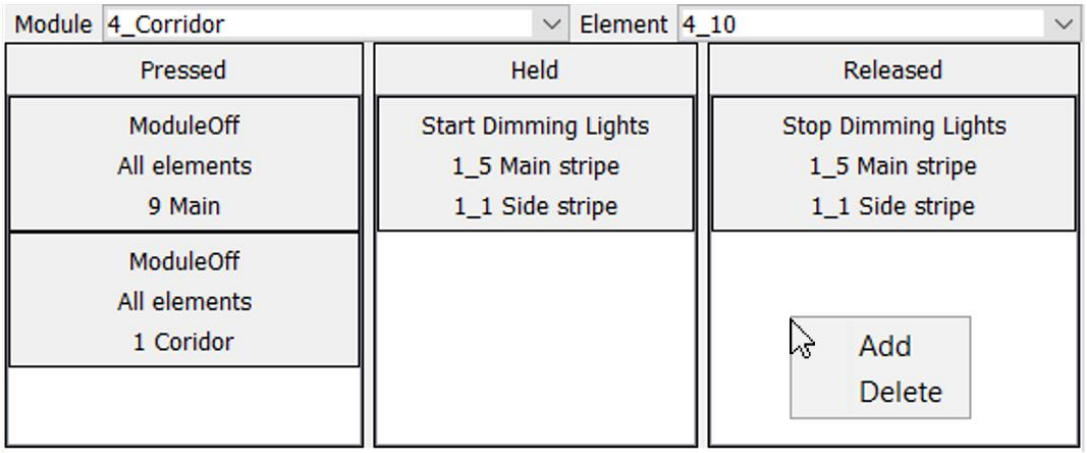
❖ *Groups* – aby bylo grafické prostředí co nejvíce jednoduché a intuitivní, bylo pro tuto záložku vybráno zobrazení *skupin*, a *elementů* v nich, pomocí stromového adresáře (podobná grafika, jako je zobrazena na obr. 38), které se používá při výběru souborů v průzkumníku. Byla také implementována funkce Drag&Drop, tedy chytni, přetáhni a pust. Díky této funkci je možné velmi jednoduše umisťovat elementy do skupin, stejně tak je z nich odstraňovat, jednoduchým přetáhnutím *elementu* na kýženou složku reprezentující skupinu.



obr. 38: Základní navazování elementů na tlačítka

❖ *Basic Bind* – Základní navazování elementů na tlačítka je poskytováno tímto oknem. V kapitole 4.1.6 o tlačítku (*ButtonElement*) byly uvedeny výchozí možnosti vázání tlačítka na element stmívatelného světla či na element spínače. Tomu odpovídá i grafická reprezentace možnosti navázání tlačítek na výchozí moduly. Okno obsahuje čtyři sloupce:

- První nabízí: seznam tlačítek (*ButtonElement*), které jsou řazeny ve „složkách“ dle modulu tlačítek (*ButtonModule*). Vždy je možné označit pouze jednu položku.
- Třetí nabízí: seznam všech možných elementů typu *DimmableLightElement* a *SwitchElement*. Stejně jak je tomu u prvního sloupce, *elementy* jsou řazeny ve „složkách“ dle příslušného *modulu*, ve kterém se nacházejí. Vždy je možné označit několik položek, ale pouze v rámci jednoho *modulu*.
- Druhý nabízí: grafickou interpretaci propojení, které se objeví poté, co se označí *elementy* v první a druhém sloupci. Na příkladu jsou zobrazena propojení pro dva *elementy*, ovšem může být jak pro víc *elementů*, tak i pro jeden. Když je výběr *elementů* v obou sloupcích korektní, při stisku klávesy *Enter* se vytvoří propojení a ubyde z prvního seznamu tlačítko, které bylo použito pro právě vytvořenou vazbu.
- Čtvrtý nabízí: seznam již vytvořených propojení. Je-li propojení složitější, vypíše se *Custom binding* jakožto popis, na který element byl *ButtonElement* navázán. V případě, kdy tlačítko bude navázáno pouze na jeden *element*, vypíše se název právě toho *elementu*. Řetězce jsou pak ve formátu IDmodulu/poziceElementu/názevElementu.



obr. 39: Pokročilé navazování elementů na tlačítka

- *Advance Bind* nabízí – přehled a vytvoření složitějších navázání funkcí při vyvolání události na tlačítku. První řádek umožňuje vybrat dané tlačítko, u kterého je žádané zobrazení již vytvořených vazeb, nebo editace vazeb. Tři sloupce v tomto oknu nabízí seznam všech úkonů, které se stanou při stisku/držení/puštění tlačítka. Každý rámec zastupuje jednu vyvolanou funkci. Počet těchto funkcí není nijak omezený. Úprava a vytváření vazeb je možná vždy přes pravý klik myši do daného sloupce, ve kterém se má provádět úprava/přidání funkcí. Kontextové menu je zobrazeno ve třetím sloupci tohoto okna. Jakmile je kliknuto na možnost *Add*, tedy přidat, otevře se průvodce podobný při vytváření nového elementu a postupně průvodce dovede uživatele ke zdárnému konci.

4.3.3 TCP Server

Master program nabízí vytvoření jednoduchého TCP serveru, který umožňuje TCP klientům se na něj připojovat a vyměňovat si zprávy ve formátu JSON (*JavaScript Object Notation*). Jedná se o na platformě nezávislý způsob zápisu informací. Výhodou oproti XML je menší datová náročnost při reprezentování stejné informace. Server jako takový umožňuje posílání pouze řetězců, proto jsou JSON objekty převáděny na řetězce. Důvodem, proč negenerovat přímo řetězce, je lepší čitelnost objektu, když jsou data ukládána do objektů. Stejně tak při příjmu řetězce je celý řetězec převeden do JSON objektů a je jednodušeji zpracován oproti možnému přímému zpracování řetězce. Server je provozován na doméně „localhost“, jeho port je možné konfigurovat v XML souboru.

4.4 Krátká exkurze do projektu openHAB

Více a více nových vychytaných zařízení a technologií přichází do našich domovů každý den. Ale i když jsou všechna zaměřena na obohacení našeho životního stylu, postrádají jednu důležitou vlastnost a tou je společný jazyk. Je potřebné, aby všechna zařízení byla schopna spolu komunikovat a vytvořit tak opravdu automatizované a chytré prostředí domácnosti. Hlavním cílem projektu openHAB je poskytnout integrační platformu pro napravení výše popsaného problému. Struktura openHABu je vysoce modulární a založena na frameworku Eclipse SmartHome. Výhodou modulárnosti je možnost rozšíření této aplikace pomocí různých „doplňků“, díky nimž je možné komunikovat s novým systémem inteligentní domácnosti nebo také nabídnout uživatelské rozhraní. Projekt openHAB kompletně napsaný v Javě a používá Apache Karaf spolu s Eclipse Equinox jako OSGi runtime a to celé je propojeno pomocí Jetty jakožto HTTP server. [9]

Přesně kvůli těmto rysům byl vytvořen program pro provázání právě nově vzniklého systému do tohoto prostředí. Nejedná se o úplné využití všech jeho vlastností, ale zejména využití jeho serverového grafického výstupu, díky kterému uživatel může mýt stylový přehled nad svým bytem a také jej ovládat. V dřívější kapitole bylo popsané vázání funkcí na tlačítka. Tato a mnohé jiné funkce, zůstávají v režii programu Master. Platforma openHAB není tedy určena k nahrazení vybudovaného programu, i když by bylo možné jej celý implementovat do této platformy. Jelikož je potřebné, aby SID byl naprosto spolehlivý, bude pracovat program Master nezávisle na openHABu.

Projekt openHAB je postavený na myšlence dopisování jeho kódu veřejností, což snižuje jeho spolehlivost a bohužel je to i znát. Ovšem na druhou stranu je tato myšlenka právě tou, která dělá z toho projektu jedinečný projekt. Relativně jednoduše, pokud je uživatel zkušeným Java programátorem, může dopsat „*Binding program*“ neboli navazovací program. Projekt openHAB jde naproti těmto developerům, a proto existuje návod, jak si vytvořit takovýto program. Ovšem jedná se o opravdu jednoduchý návod, který moc poučný není, ale ukáže uživateli první kroky, což se opravdu cení. Doporučením, jak se naučit programovat propojovací programy, je nastudovat si jiný propojovací program a postupně vybudovat svůj, k přesnému pochopení volání metod pomůže debugování.

Veškeré prvky, které je potřeba přidat do navazovacího programu pro openHAB (správně by mělo být řečeno pro Eclipse SmartHome) se nejdříve konfigurují v definovaných XML souborech, které se následně propojí s programem psaným v Javě. Dalo by se říct, že v XML souboru se definují vstupní parametry, vzhled a různé typy pro komponenty pro openHAB a následně v Java programu se napíší *Handlers*, tedy jakési programové manažery pro popsané komponenty v XML.

4.4.1 TCP klient

Propojovací program mezi openHABem a nově vzniklým SID je vlastně TCP klientem, který převádí události na obou stranách na JSON zprávy, které se v protilehlé straně vyhodnocují a propagují se k jednotlivým komponentám. To jest hlavní myšlenka propojovacího programu.

Navázání komunikace se serverem není problematické, ovšem problematické může být udržení navázaného spojení. Byla tedy vytvořena třída, která je schopna se připojit k TCP serveru a jakmile se vyskytne jakýkoliv problém v připojení, okamžitě je to oznámeno druhé třídě, která se stará o udržení komunikace. Tato třída se snaží neustále dokola, dle nastavené periody, periodicky ověřovat, zda je server již dostupný, aby se prvně zmiňovaná třída mohla připojit a nadále komunikovat mezi programem Master a openHABem. Tato část propojovacího programu je nejdůležitější, protože zajišťuje spolehlivost navázaného spojení.

4.4.2 Komponenta Bridge

V oblasti IoT systému jsou často hierarchické struktury zařízení a služeb (službou je zde myšlen neustále spuštěný program). Například, jedno zařízení se chová jako brána, která umožňuje komunikovat s ostatními zařízeními, která používají stejný protokol. V Eclipse SmartHome tento druh zařízení nebo služby se nazývá *Bridge* (neboli most, ovšem zůstaneme u originálního pojmenování tříd v projektu Eclipse SmartHome). [10]

Přesně tato brána je vhodná pro implementaci propojovacího programu. Systém inteligentní domácnosti popsaný v této práci komunikuje pomocí UART, ale zvenčí je možno ovlivňovat a číst stavy na síti pomocí TCP komunikace, můžeme říct brány, anebo také *Bridge*, jak užívá terminologie

zmíněného projektu Eclipse SmartHome, na kterém je vybudován projekt openHAB. Důležitá vlastnost mostu je schopnost uchování *Things* (věcí), o nichž se dozvíme v další podkapitole.

Z podstaty věci a výše psaného textu je jasné, že právě tento *Bridge* při vytvoření zároveň otevírá TCP komunikaci s programem Master a umožňuje sdílení zpráv. *Bridge* představuje tedy jakýsi virtuální prvek, který slouží pouze pro komunikaci s Master programem a slouží pro uchování všech *Things*. Při vytváření *Bridge* uživatel volí pouze TCP port a interval opakování v případě odpojení od TPC serveru.

4.4.3 Komponenta Things

Thing (věci) reprezentují zařízení či služby, které mohou být samostatně přidány, konfigurovány či odstraněny ze systému. Také obsahují soubor *Channels* (kanálů) nebo skupiny *Channels*. *Bridge* je specifickým typem *Things*, který je mimo jiné schopný poskytnout přístup k ostatním *Things*. Jaké *Things* mohou být sdruženy skrze určitý typ *Bridge* je definováno v XML popisu. [10]

Nyní je potřeba najít paralelu mezi programem Master a mezi *Things*. Podle popisu čerpaného z projektu Eclipse SmartHome je možno usoudit, že nejlepší shodou *Things* jsou *Elementy*. Je tedy nutno popsat všechny vytvořené typy *Elementů* vytvořených v programu Master jako *Things*, aby si tyto dva programy rozuměly a správně vyměňovaly informace.

4.4.4 Komponenta Channels

Channels (kanály) popisují specifickou funkcionalitu *Things* a mohou být nalinkovány k *Item* (položce). Základní informace tedy je, jaký typ příkazu může kanál zpracovat a jaký stav je poslán linkované *Item*, což specifikuje akceptovaný typ *Item*. Seznam navázaných kanálů je uvnitř XML popisu *Thing*, přesněji v popisu jeho typu. Definice typu Channel je specifikován na stejné úrovni jako je definice typu *Thing*. Díky tomu mohou být *Channels* využity v různých *Things*, stačí jen přiřadit jejich id. [10]

Dá se říct, že tyto kanály uživateli říkají, co vše je možné skrze *Thing* ovládat či číst. Chtěli bychom například udávat u každého světla to, zda je připojeno k systému inteligentního domu. Můžeme nastavit dva kanály (*Channels*). První kanál by sloužil pro jeho zapínání a vypínání, tedy typ *Switch*. Druhý typ by byl *String* a v případě odpojení tohoto světla od SID by se místo online statusu vypsal čas poslední komunikace s tímto světlem.

4.4.5 Příklad vytvoření Thing – SwitchLight

Pro příklad bude uveden vytvořený *SwitchLight* v openHAB systému. SwitchLight je typu *Thing*, který je paralelou *SwitchElement* s modifikátorem *Light* v programu Master. Nejprve je nutné v XML souboru definovat na jaký Bridge se připojuje, následně jeho popis, typy *Channels*, vstupní parametry a jejich popis a jako poslední položkou je definice použitých *Channels*. Poslední položka může být ve zvláštním XML souboru, a poté by bylo možné typy *Channels* mezi *Things* sdílet.

Je-li vytvořený XML popis Thing, je potřeba jej provázat s Java programem openHABu, jinak řečeno popsat jeho chování v Javě. První krokem je vytvoření *BindingConstants* (navazovací konstanty), které zastupují různá id v XML souborech. Tyto konstanty jsou následně používány v navazovacím programu. Další důležitou částí je oznámení třídě *ThingHandler* (manažer věcí) co má dělat

v případě jeho prvního připojení, tedy jak má vypadat inicializace tohoto typu *Thing*. *ThingHandler* se mimo jiné stará o vykonání popsané funkce při změně statusu *SwitchLight*. Proto je nutné, stejně tak jako u inicializace, rozpoznat typ *Thing* při vyvolání události v programu. Jakmile *ThingHandler* ví, o jaký typ *Thing* se jedná, přeposílá požadavek na zpracování třídě *SwitchLight*. Třída *SwitchLight* již musí vědět, co má dělat s přijatým příkazem. Tímto byl ve zkratce popsán princip vytvoření jedné *Thing* a naznačeno zpracování příkazů přijatých od program openHAB.

5 Závěr

Teoretický úvod otevírá problematiku systémů inteligentních domů, v níž je prostudována teorie týkající se systémů inteligentních domácností. Díky zmíněnému průzkumu, který byl vytvořen firmou Microsoft ve spolupráci s Washingtonskou univerzitou, byl utvořen i náhled na potřebné funkce pro inteligentní domácnost. Tyto funkce byly dále teoreticky rozšířeny do jednotlivých podkapitol managementu, od kterých se odvíjí návrh modulů pro inteligentní domácnost. Funkce, které byly vybrány pro realizaci, nejsou jen pro zvýšení komfortu, což je jedním z cílů inteligentních domácností, ale i snížení ekonomické zátěže.

Obsáhlou částí diplomové práce je navržení spolehlivé komunikace. Nejprve byl rozebrán standard komunikace RS-485, poté popsána multiprocesorová komunikace UART pro mikrokontroléry ATmega a nakonec navržení formátu zpráv. Kapitoly postupně přechází z teoretického úvodu do praktičtějších pasáží, jako jsou výběr mezi komunikací duplexního či polo-duplexního typu, způsob iniciace komunikace, zajištění bezchybného doručení zprávy, výběr typu kabelu, přiřazení jednotlivým párům funkci, výběr konektoru, a na konec odzkoušení komunikace bez a se zakončovacími rezistory spolu s ověřením rychlosti komunikace na delší vzdálenost. Stěžejním bodem bylo ověření komunikace, které zdárně prošlo a mohlo se tak pokračovat v práci.

Třetí kapitola již poskytuje náhled na koncepci modulů inteligentní domácnosti, který se v dalších podkapitolách zabývá jednotlivými moduly. Každá podkapitola o daném modulu uvádí čtenáře do problematiky, se kterou se autor diplomové práce potýkal při realizaci modulů. Dále podkapitoly jednotlivých modulů představují návrh a realizaci, která byla rozdělena na část softwarovou a hardwarovou. Softwarová podpora v každém modulu je vytvořena čistě pro mikrokontroléry. Jakmile byl tento základ hotov, započala práce na komunikačním rozhraní s touto sítí, aby bylo možné ovládat celý dům dálkově. Jelikož Master by byl zbytečně složitý pro implementaci programu do mikrokontroléru, byla pro něj vytvořena aplikace přímo v Raspberry Pi. Výhodou Raspberry Pi je podpora aplikace openHAB, která umožňuje ovládat SID přes TCP internetový protokol. Další výhodou je nabídka webového rozhraní a mobilní aplikace pro openHAB, která ulehčuje jeho ovládání.

Aplikace openHAB nenabízí pouze grafické rozhraní pro ovládání inteligentního domu přes internet, ale zejména umožňuje implementaci dalších prvků inteligentní domácnosti třetích stran. Stále opakující se problém nutnosti vlastnění několika ovládacích programů pro každý prvek inteligentní domácností je s touto aplikací minulostí. Tyto nesporné výhody byly natolik přesvědčivé, že byl napsán propojovací program pro nově vzniklý systém inteligentní domácnosti, díky čemuž je SID plně kompatibilní a ovladatelný aplikací openHAB. Mimo jiné, tato aplikace řeší i bezpečnostní prvky připojení a je tedy možné bezpečně ovládat svůj dům z internetu.

Závěrečným krokem diplomové práce je realizace tohoto systému v reálném prostředí, a to přímo v domě. Nejedná se tedy pouze o model inteligentní domácnosti, který ne vždy pokryje všechna úskalí projektu, jako může být komunikace na dlouhé vzdálenosti a rušení z okolních zdrojů a rozvodů v domácnosti. Diplomová práce klade důraz na realizaci zcela funkčního systému, s řešením mechanických záležitostí, jako je umisťování modulů inteligentní domácnosti do ochranných krytů, ale i jejich rozmístění v domácnosti tak, aby byly přístupné, a přitom nekazily celkový designový dojem.

Seznam obrázků

Seznam tabulek

Použitá literatura

[1] BRUSH, A.J. Bernheim, Bongshin LEE, Ratul MAHAJAN, Sharad AGARWAL, Stefan SAROIU a Colin DIXON. *Home Automation in the Wild: Challenges and Opportunities* [online]. 2011 [cit. 2016-10-19]. Dostupné z: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HomeOSCHI_cameraready_Final.pdf

[2] *RS-485 Reference Guide* [online]. Texas Instruments, 2014, , 2-5 [cit. 2016-11-20]. Dostupné z: http://www.ti.com/lit/sg/slyt484a/slyt484a.pdf

[3] KUGELSTADT, Thomas. *The RS-485 Design Guide* [online]. Dallas, Texas 75265: Texas Instruments Incorporated, 2016, , 5 [cit. 2016-11-27]. Dostupné z: http://www.ti.com/lit/an/slla272c/slla272c.pdf

[4] *Data networks and open system communication: Open systems interconnection - model and notation* [online]. 1994, , 32-49 [cit. 2016-11-28]. Dostupné z: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items

[5] *ATmega8 Datasheet* [online]. 2486AA–AVR–02/2013. Atmel, 2013 [cit. 2016-11-28]. Dostupné z: http://www.atmel.com/Images/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf

[6] N. WILLIAMS, Ross. *A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS* [online]. 3. 16 Lerwick Avenue, Hazelwood Park 5066, Australia., 1993 [cit. 2016-12-04]. Dostupné z: http://www.ross.net/crc/download/crc_v3.txt

[7] *Understanding Stroboscopic Effects (Flicker)* [online]. XICATO®, , 2 [cit. 2017-01-07]. Dostupné z: http://www.xicato.com/sites/default/files/documents/Flicker%20White%20Paper_0.pdf

[8] *The Pi4J Project: Java I/O library for the Raspberry Pi* [online]. 2016 [cit. 2017-02-22]. Dostupné z: http://pi4j.com/

[9] *OpenHAB* [online]. Německo: openHAB Community and the openHAB Foundation e.V., 2017 [cit. 2017-03-13]. Dostupné z: http://www.openhab.org

[10] *Eclipse SmartHome™: A Flexible Framework for the Smart Home* [online]. Canada: Eclipse IoT, c2017 [cit. 2017-03-14]. Dostupné z: https://eclipse.org/smarthome/index.html

[11] DIY - Isolated High Quality Mains Voltage Zero Crossing Detector. *DEXTREL - DeGo's Extreme Electronics Lab* [online]. Helsinky [cit. 2017-03-31]. Dostupné z: http://www.dextrel.net/diyzerocrosser.htm

[12] *QTouch Library Peripheral Touch Controller: USER GUIDE* [online]. Rev.M. 1600 Technology Drive, San Jose, CA 95110 USA: Atmel Corporation, 2016 [cit. 2017-04-01]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/atmel-42195-qtouch-library-peripheral-touch-controller_user-guide.pdf

[13] *Buttons, Sliders and Wheels: Sensor Design Guide* [online]. Revision A. Atmel Corporation, 2011 [cit. 2017-04-01]. Dostupné z: http://ww1.microchip.com/downloads/en/AppNotes/doc10752.pdf

[14] Support for Atmel Touch Controls. *Altium TechDocs: Online Documentation for Altium Products* [online]. United States: Altium Limited, c2017 [cit. 2017-04-01]. Dostupné z: http://techdocs.altium.com/display/ADOH/Support+for+Atmel+Touch+Controls