Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Prostředky domácí automatizace a zabezpečení, s návazností na komerční systémy

Plzeň 2019 Josef Štědronský

Prohlašuji, že jsem předloženou závěrečnou práci vypracoval(a) samostatně s použitím zdrojů informací a literárních pramenů, které uvádím v přiloženém seznamu literatury.

V Plzni dne 31. března 2011

……..……

vlastnoruční podpis

Abstrakt

Tato práce se zabývá konstrukcí a softwarovým vybavením systému domácí automatizace na platformě Raspbery Pi a jeho kooperací s některými komerčními systémy. Cílem práce bylo zkonstruovat systém inteligentního domu, který poskytne uživateli požadovanou funkcionalitu. V práci jsou řešeny základní potřebné funkce tohoto systému a nastíněny další možnosti funkcí, kterými bude postupně dovybaven.

Abstract

This work deals with construction and software equipment of home automation system on Raspbery Pi platform and its cooperation with some commercial systems. The goal of the work was to design an smart home system that would provide the user with the required functionality. Basic necessary functions of the systém are solved in the work and other possibilities of functions, which will be to the system gradually retrofitted, are outlined.

Obsah

Seznam obrázků

Úvod

„Domov - vždycky to budu opakovat - není prostor, ale proces. Tady se schovávám před deštěm a zimou, tady žiju, tady trávím svůj volný čas, tady se rodí moje rodina, tady ji tvořím léty, trpělivostí a tolerancí a vůbec řadou kladných principů. Domov je bytost.“

(Miroslav Horníček)
https://citaty.net/citaty/5807-miroslav-hornicek-domov-vzdycky-to-budu-opakovat-neni-prostor-a/

Důvodem volby tématu bakalářské práce, tedy domácí automatizace, bylo to, že mě tato problematika dlouhodobě zajímá, koresponduje celkem přesně i se zájmy o hardware a software a v neposlední řadě i to, že vytvořím něco užitečného, co pak bude moci ve své funkci mnoho let dobře sloužit.

V dnešní době, kdy se začíná rozšiřovat trh o produkty IoT, *internet of things*, tedy internet věcí. Je pravděpodobné, že bude i poptávka po produktech jak z této kategorie, tak obecně domácí automatizace větší. Tedy lze i očekávat, že bude stále častěji potřeba tyto produkty vzájemně propojit. V posledních letech značně narůstá podíl kyber-kriminality na zločinu celkově, tedy i z tohoto pohledu může být problematika vlastní konstrukce zajímavá.

Technologie nám dnes umožňuje dříve nemožné, otázkou je, zda to opravdu chceme, či zda je nám to někdy jen vnucováno marketingem a reklamou. Není jisté, zda skutečně toužíme po tom, aby nám v budoucnu lednička nakupovala. Také v případě kamerového systému existují rizika, že nám někdo může nahlížet do soukromí, pokud se „naboura“ do našich kamer. Je tedy třeba vždy pečlivě zvážit všechny klady a zápory nových technologií. Samozřejmě, že u komerčních produktů toho máme pod kontrolou daleko méně, než u toho, co si sami vytvoříme. Je tedy možné, že počet zájemců o konstrukci vlastního systému bude časem stoupat. Samozřejmě, že jen v určité části populace, která bude kreativní v tomto směru. Nicméně mnoho atributů společnosti se neustále mění a nelze vyloučit, že se opět zvýší počet jedinců, kteří si raději svépomocí zajistí řadu činností, než by platili drahé specialisty, na každou dílčí potřebnou činnost. Tato práce by mohla ukázat cestu, jak si s relativně malými prostředky vybudovat celkem robustní, dostatečně uživatelsky přístupný, mnohostranný, konfigurovatelný a rozšiřitelný systém domácí automatizace. Práce si neklade za cíl přibližovat se komerčním zařízením, ale vytvořit systém dostatečně uspokojující potřeby uživatele, který není extrémně náročný a má vztah k technickým řešením takovéhoto typu.

Systém nebude tvořen jako optimalizovaný pro co největší jednoduchost, ale tak, abychom se na něm mohli co nejvíce naučit. Proto budou použita různá rozhraní a různé způsoby komunikace, což bychom se určitě v případě komerčního produktu snažili pokud možno omezit. Účelem práce však je spíše prozkoumat různé možnosti a získat tak co největší vhled do dané problematiky, v rozsahu odpovídajícímu rozsahu práce. Také hardwarové prvky budou použity různorodé. Částečně bude využito komerčních zařízení, se kterými se náš systém naučí komunikovat, částečně to budou komerční prvky s upraveným firmware pro naše potřeby, a částečně prvky vytvořené na bázi systému Arduino, s patřičnými doplňky, jako jsou čidla, relé, síťová rozhraní apod. Systém bude navržen tak, aby umožňoval rozvoj i modifikace v budoucnosti a bude instalován, užíván a rozvíjen v reálném prostředí rodinného domu. Software použitý v práci bude povětšinou vlastní tvorby, pokud bude použit software třetích stran, bude na to upozorněno. Jelikož uvažuji i o možnosti komerčního, nebo komunitního využití, dostal projekt název „BranaDomIoT“ a byla zaregistrována doména BranaDom.cz.

1 Teoretická část

1.1 Základní filosofie systémů domácí automatizace

1.1.1 Výzvy pro konstruktéry domácí automatizace

Touha po automatizaci

Touha po automatizaci domácnosti začala již velmi velmi dávno. První pračka prádla byla sestrojena koncem 18. století. Ve století 19. a 20. se podíl různých zařízení, která nám usnadňují chod domácnosti postupně zvyšoval. Rozvoj šel také ruku v ruce s měnícími se možnostmi technologií, od jednoduchých zpravidla mechanických, později elektromechanických systémů řízení takových pomocníků, až ke dnešnímu *IoT* – „internet of things“ tedy česky řečeno „internetu věcí“. Co se týče domácí automatizace, někomu by se mohl tento pokrok jevit jako lenost, a je pravdou, že lenost byla nezřídka „hnací silou“ pokroku. Dnes již si asi málokdo dokáže představit domácnost bez automatické pračky, myčky, domácí pekárny, inteligentního kávovaru apod.

(ELSENPETER, 2003, str. 3)

Klíčové pohledy

Je vhodné, aby systém byl rozdělen – distribuován do menších funkčních celků. Typicky jedna řídící jednotka (centrála) a mnoho dalších podsystémů, tedy jednotek, které vykonávají dílčí funkcionalitu, což může být například termostat, elektrický zámek, spínaná zásuvka apod. Tyto subsystémy pak mohou komunikovat s centrálou prostřednictvím různých protokolů a nosných médií. Systém je vhodné navrhnout heterogenní, neboť v dnešní době existuje mnoho možností komunikace a mnoho přenosových protokolů. Samozřejmě každá dílčí aplikace má jiné požadavky zvláště na přenosovou rychlost, spolehlivost a zabezpečení přenosu. Další odlišností může být bateriové, či síťové napájení a s ním často spojený přenos občasný, či trvalý. Chceme-li používat systémy různých výrobců, je pravděpodobné, že se mohou v požadavcích na tyto parametry přenosu významně odlišovat. Další zásadní vlastností systému by měla být jeho budoucí rozšiřitelnost a to jak kvantitativní, tak kvalitativní. Tedy možnost přidávat větší počet stejných subsystémů, ale také možnost přidávat zcela nové, postupně s rozvojem technologií přibývající. Výše již byla zmíněna bezpečnost, i v tomto ohledu může být velká varieta požadavků. Je potřeba neopomíjet možnost adaptace, či auto-adaptace, například při změně chování uživatelů, což bude blíže rozebráno v dalších kapitolách. Další důležitou užitnou vlastností je jednoduchost, ergonomie a intuitivnost užívání celého systému. To se obvykle nejvíce týká uživatelského rozhraní, ale i celkové filosofie funkcionality.

(AL-QUTAYRI, 2010, str. 6 – 8)

Bezpečnost systému, způsoby zabezpečení

Bezpečnost jistě zajímá mnoho potenciálních uživatelů takových systému, přesto, že se zdá, že čím mladší generace, tím méně jí obecně bezpečnost čehokoli trápí. Jako bychom pod tlakem dnešní doby postupně ztráceli naše instinkty. Nicméně je jasné, že bezpečí je zásadní součástí lidských potřeb. Tedy i technologie domácí automatizace by měly brát tento faktor v potaz. Mnoho produktů, například IoT, na trhu, potažmo jejich výrobci, si s bezpečností velmi často příliš nelámou hlavu. Naopak jedná-li se například o průmyslovou automatizaci, či prvky systémů komerčního elektronického zabezpečení, je bezpečnost často jedním z nejdůležitějších požadavků. Způsoby zajištění bezpečnosti jsou nesčetné. Může se jednat o zabezpečený přenos, tak aby nešel odposlouchat, zabezpečení doručení informace nebo povelu, například potvrzováním zpráv na různých vrstvách komunikačních protokolů, či zpětné vazbě, která zajistí informaci o tom, že požadovaný povel byl skutečně vykonán, například z nějakého kontrolního snímače. Bezpečnost však můžeme vnímat i globálněji, tedy tak, že nám domácí automatizace zajistí bezpečnost našeho domu, například pohybovými, kouřovými či záplavovými senzory, spolu s komunikačním kanálem, nejlépe zálohovaným pro použití v případě chybového stavu či přímo poplachu, který může být nasměrován jak na majitele, tak na bezpečnostní agenturu. Další možností, která je ale spíše implementována do mobilních zařízení, je monitorování zdravotního stavu uživatele. Tedy jeho základních životních funkcí, případně k detekci nezvyklého chování, či pádu, například u těžce pohyblivých osob, či epileptiků.

(AL-QUTAYRI, 2010, str. 4 - 5)

Měření a regulace spotřeby

Co se týče vlastní regulace, bude zmíněna na mnoha dalších místech práce. Připomeňme si tedy jen potřebu získávat a zpracovávat určité naměřené údaje. O regulaci vytápění, klimatizace, ventilace, či zastínění budeme také hovořit v dalších částech. Zmiňme tedy jen možnost získávat naměřená data nejen například z vnitřních či vnějších teploměrů, vlhkoměrů apod., ale také pomocí elektroměrů měřit aktuální spotřebu a na základě ní vyvolávat nějaké akce. Jako jednoduchý příklad si můžeme uvést to, že díky tomuto měření a prvkům domácí automatizace, můžeme efektivně balancovat spotřebu, tak, abychom si vystačili například s menším hlavním jističem, jehož velikost určuje část pravidelných nemalých plateb dodavateli elektrické energie. Další možností je vyvažovat spotřebu v období nízkého a vysokého tarifu elektrické energie, případně v kombinaci se solárními panely či větrnou elektrárnou. Tato regulace může být do budoucna zajímavá i pro vlastníky elektromobilů, pro vhodnou regulaci doby a spotřeby jejich nabíjení.

(AL-QUTAYRI, 2010, str. 6)

Vytápění, ventilace, stínění a klimatizace

Výzvami v této oblasti může být inteligentní regulace schopná předvídat. Pokud možno poněkud lépe, než inteligentní systémy výrobců plynových kotlů apod., které často vykazují velmi podivné chování. Tedy regulace za použití domácí automatizace může využívat podstatně větší množství senzorů, a to jak lokálních, tak například z meteostanic na internetu, případně využít předpovědi počasí, i když ta také nebývá zdaleka spolehlivá. Tento systém také souvisí s výše uvedeným systémem regulace spotřeby a měl by s ním tedy kooperovat.

(AL-QUTAYRI, 2010, str. 6)

1.2 Možnosti systémů

1.2.1 Jak se mohou objevovat potřeby v průběhu dne

Ráno

Náš inteligentní dům může vstávat dříve než my. Po noci kdy jsme snížili teplotu pro lepší spánek a větší úsporu, nám po ránu přitopí v místech kde se budeme pohybovat, například v koupelně či kuchyni. Může nás také vzbudit tichou hudbou či postupným rozsvěcením světel, roztahováním žaluzií, závěsů nebo vytažením rolet. Může změnit nastavení zabezpečovacího systému, tedy například zóny v noci aktivované deaktivovat. Dnes již nám také může připravit předem kávu či domácí pekárna může poskytnout právě dopečenou bábovku. Venku už může být spuštěno automatické zalévání trávníku.

(ELSENPETER, 2003, str. 4-5)

Na základě předpovědi počasí vám také modul venkovních dveří, například na bázi Arduina, může připomenout abyste si vzali deštník, nebo že dnes jedou popeláři a je potřeba dát vaši popelnici na ulici.

(GOODWIN, 2013, str. 16)

Během dne

Ve chvíli kdy všichni členové rodiny opouští dům, sprchy a horké vody již nebude potřeba, tedy je možno snížit teplotu užitkové vody, taktéž teplotu vytápění. Je také možno, pomocí predikce počasí při znalosti tepelné dynamiky domu, kalkulovat kdy a jakým výkonem bude potřeba topit, abychom dosáhli požadované křivky teplot v obytných prostorách. Jakmile všichni odejdou, je na čase aktivovat zabezpečení celého domu, případně ověřit, zda byla zavřena okna, zamčeny dveře, vypnuty spotřebiče. Je také možno spotřebiče které nebudou používány od sítě odpojit, aby ve stand-by režimu zbytečně nekonzumovaly a taktéž nebyly v době nepřítomnosti potenciální „roznětkou“ . Dále se náš systém může postarat o naše mazlíčky. Nakrmit rybičky, ale i kočku či psa, a zajistit jim dostatek čerstvé vody. Také je možno domácí mazlíčky kontrolovaně vypouštět například na zahradu v danou dobu nebo otevírat „vrátka“ jen při přiblížení našeho mazlíčka, abychom eliminovali stav, že jimi projde jakékoli zvíře. Součástí řízení vytápění může být samozřejmě i natáčení žaluzií, či ovládání klimatizace. Z bezpečnostních aktivit snad zmiňme záplavu vodou, požár či výpadek napájení.

(ELSENPETER, 2003, str. 4-5)

Příchod domů

Systém může znovu ohřát užitkovou vodu, natopit požadované místnosti, třeba pracovnu a přivítat uživatele šálkem horké kávy. Pokud by bezpečnostní systém hlásil nějaké problémy, je možno se z mobilní aplikace pomocí kamer podívat, zda je vše v pořádku, a jedná se jen o falešný poplach, či zda je v domě například lupič, požár nebo sousedova kočka. Tím se vyhneme případnému nepříjemnému překvapení při příchodu.

(ELSENPETER, 2003, str. 5-6)

Večer

Systém se může postarat o naši zábavu. Umožní nám například distribuci audio či video signálu po celém domě, při sledování TV přenastaví způsob osvětlení místnosti,

zatáhne žaluzie a to nejlépe vše jako reakci na prosté zapnutí televize. Po uložení ku spánku systém může znovu přehodnotit aktivaci hlídaných zón v domě, nastavit noční režim osvětlení, či automatické spínání slabého osvětlení, při pohybu osob v noci, například cestou na toaletu.

(ELSENPETER, 2003, str. 6-7)

Základní funkcionality inteligentního domu

Jak již humanistický psycholog Abraham Maslow ukázal na své proslulé pyramidě potřeb (1943), existuje velký rozdíl mezi tím co chceme a co skutečně potřebujeme. A co to má společného s domácí automatizací ? Zdaleka ne všichni využijí všechny možnosti, mnoha lidem příliš složitá a košatá funkcionalita může být na obtíž. Mnoho uživatelů bude chtít mnoho funkcí a nebudou je třeba schopni efektivně využít, či případná složitost jim způsobí více nepříjemností, než radosti a užitku. Chytrý dům nám ušetří peníze, ušetří naši práci a zredukuje částečně naši starost o to, co se v době naší nepřítomnosti děje. Nicméně na straně druhé nás bude nutkat k častým kontrolám, případně nás občas vyděsí falešnými alarmy. Taktéž budeme možná mít potřebu neustále něco ladit a vylepšovat, což může být oproti „blažené nevědomosti“, dosti stresující. Tedy pokud bychom lehce parafrázovali české rčení, můžeme prohlásit: „Není na světě systém ten, který by se zavděčil uživatelům všem“. Na druhou stranu získáme větší zabezpečení díky kombinaci více systémů. Je možné vytvořit systémy velmi jednoduché, ale i zcela komplexní, limitující může být například cena komerčních systémů.

(ELSENPETER, 2003, str. 8-9)

1.2.2 Prvky systému

Základní stavební kameny

Každý inteligentní dům, potřebuje určité základní vybavení, k tomu, aby smart systém mohl fungovat. Samozřejmě, že toto hardwarové vybavení může být velmi různorodé. Od jednoduchých prvků po sofistikované, od přímo spínaných po síťové, bezdrátové či infračervené.

Základním prostředkem komunikace může být domácí síť a to jak klasický metalický Ethernet IEEE 802.3, tak připojení pomocí Wi-Fi IEEE 802.11, či speciální komunikační protokoly jak bezdrátové, tak prostřednictvím připojení kabelem. Může se jednat o sběrnice I2C, CANBUS, OneWire, RS485, X10, o bezdrátovou komunikaci na 433 MHz Z-Wave, ZigBee, apod. Na klasické počítačové síti pak můžeme použít protokolu MQTT pro komunikaci s dalšími prvky na této síti a využít tak existující rozvody, či Wi-Fi acces pointy.

Velmi často se dnes stává, že uživatelé zvláště novostaveb, jsou alergičtí na jakýkoli otvor, či drát a vše chtějí řešit bezdrátově, i za cenu všech úskalí, která to s sebou nese. Nicméně pro některé aplikace může být propojení drátem stále vhodnější.

Dalším zásadním prvkem bude často zabezpečovací systém. Zpravidla bývá instalován systém, jednoúčelový, kvalitní a odolný. Je pak vhodné aby centrum domácí automatizace dokázalo alespoň částečně s tímto systémem kooperovat. Tedy není-li možno využít ke komunikaci systémové sběrnice, čemuž se výrobci z pochopitelných důvodů

snaží zabránit , pak alespoň pomocí standardních vstupů a výstupů či tzv. programovatelných (dále jen PGM) vstupů a výstupů. Pravděpodobně je tato možnost lepší, než se pokoušet konkurovat výrobcům zabezpečovací techniky, kteří svá zařízení dělají velmi dobře, spolehlivě, a mají případně patřičné bezpečnostní certifikáty či atesty, které mohou být například vyžadovány pojišťovnami.

Hlavním středobodem automatizačního systému je zpravidla centrální jednotka. Ta obsahuje základní inteligenci celého systému a má případně schopnost komunikace s okolím. Tedy jak uvnitř objektu, tak dálkovými, třeba na zahradě, či ve skleníku a podobně. Komunikuje s jednotlivými vlastními subsystémy, čidly, ale i systémy stávajícími, například kromě výše zmíněného zabezpečovacího systému například se systémy vytápění, osvětlení, zalévání, systémy audiovizuálními, kamerovými apod. V dnešní době a do budoucna, pak může obsahovat komunikaci se všemi IoT zařízeními v domácnosti, případně na jejich aktivity dohlížet a filtrovat je.

Z výše uvedených požadavků je tedy jasné, že centrální jednotka bude muset mít relativně dost vysoký rozsah činností, tedy bude potřeba jí realizovat pomocí nějakého výpočetního systému. Nabízejí se dvě řešení, a to MCU s kompletně naprogramovanou obsluhou všeho, bez OS nebo použití nějakého minipočítače, s operačním systémem, který nám zjednoduší do značné míry návrh, neboť mnoho práce odvede za nás. První řešení je vhodné pro jednodušší aplikace, druhé spíše pro komfortnější a komplexnější systémy. Na systému s OS budeme snáze realizovat různé druhy komunikací, snadno obsloužíme standardní periférie, podstatně jednodušeji a komfortněji vytvoříme webové rozhraní. Snadno můžeme jednotlivé periférie připojovat, zdvojovat, zálohovat a podobně. Na druhé straně, náš systém bude zranitelný tak, jak bude v danou chvíli zranitelný operační systém na něm použitý. Budeme nuceni řešit aktualizace OS a případné problémy a nefunkčnosti s tím spojené.

Jednodeskových minipočítačů vhodných pro podobné účely, jsou dnes na trhu desítky. Jejich hardwarové a softwarové vybavení se velmi liší a hlavně se celkem zásadně liší jejich podpora. U některých není téměř žádná, nicméně toto bude rozebráno v kapitole zaměřené právě na tyto minipočítače.

Vnější prvky systému

Systém domácí automatizace nepotřebuje jen komunikaci se svými subsystémy, ale taktéž komunikaci navenek. Tu je potřeba zajistit zpravidla pomocí internetu. To však má jeden zásadní háček. Není velký problém internet do daného domu odpojit, rušit a podobně. Proto by mělo být využito i záložní formy komunikace, například pomocí mobilního internetu, volání a SMS v případech, kdy si to vážnost situace vyžaduje, například při poplachu bezpečnostního systému, požáru, zaplavení apod. Další nutnou podmínkou funkcionality je vlastní elektrické napájení celého systému a zvláště pak centrální jednotky a komunikačních subsystémů. To je možno zajistit buď pomocí standardních UPS nebo lépe přímo na úrovni hardware, pomocí akumulátorů s dobíjecími obvody. Taktéž je nutno zajistit monitorování výpadků napětí, poklesu napětí či kvality akumulátoru a dalších parametrů a včasné odeslání varování uživateli. A to jak standardními, tak případně urgentními kanály. Dalšími vnějšími prvky pak mohou být externí antény, kamery, zahradní systémy závlah, větrání či vytápění skleníku, vyhřívání či řízení filtrace bazénu a další.

(ELSENPETER, 2003, str. 10-11)

Garáž

Další uplatnění mohou tyto systémy najít v garáži. Tam si zpravidla uživatel vystačí s autonomním systémem pro otevírání dveří, ale komplexní systém nám může například ve správnou dobu nejen otevřít vrata, ale také rozsvítit na cestu v domě i před domem a případně nám může v zimě auto předem zevnitř například elektricky vytopit, či v extrémních mrazech nahřát motor apod. Zajistí kontrolu a zavření dveří po odjezdu i příjezdu, případně může v zimě zapnout topení a usušit vůz s nánosy sněhu.

(ELSENPETER, 2003, str. 10-11)

Domácí zábava

Do této kategorie bude patři hlavně audio a video. Tedy distribuování videa a audia po celém domě apod.

(ELSENPETER, 2003, str. 10-11)

Další aktivity

Jak již bylo výše zmíněno je to například vytápění, chlazení, ohřev TUV, větrání a rekuperace, natáčení žaluzií apod. Také ovládání zahradní techniky, zavlažování, sekání trávy a jiné.

(ELSENPETER, 2003, str. 10-11)

Interface

Jako interface pro obsluhu, jeví se jako nejjednodušší řešení webové či mobilní aplikace. Webové jsou výhodné svou univerzalitou, mobilní nám zase umožní přenášet jen minimum dat. Jako centrální ovládáni bude asi lépe používat PC a webové rozhraní, neboť nám poskytne dík daleko větší obrazovce, podstatně větší komfort při konfiguraci, či sledování například grafů stavových veličin. Taktéž klávesnice a myš nám poskytne poněkud lepší práci než dotykový displej telefonu. Samozřejmě na ovládání televize, DVD, či otevření vrat garáže bude lepší využít mobilní aplikaci s několika buttony. IR dálkové ovladače jsou jistě výborné pro jednoúčelové ovládání. Kdo však má zkušenost s univerzálními ovladači pro několik zařízení, možná není z jejich užívání nadšen tak, jak původně očekával. I zde může například webové rozhraní poskytnout větší komfort a generovat IR signál k ovládání zařízení, například klimatizace, která nemá jiný „interface“.

(ELSENPETER, 2003, str. 11-12)

1.3 Moudrý dům

V knize Gerharda Leitnera „The Future Home is Wise, Not Smart“, tedy přeloženo : „Dům budoucnosti je moudrý, nikoli chytrý“ , je na věc nahlíženo z poněkud odlišného úhlu. Autor tento pohled nazývá novým paradigmatem a používá termín humano-centrická perspektiva, tedy středobodem je právě člověk. Zmiňuje zde „computer psychology“, což je pro náš kontext asi nejblíže tzv. Inženýrské psychologii a možná části ergonomie a také a *HCI – human computer interaction* tedy v české literatuře často nazýváno komunikace člověk versus stroj. To je jistě důležitý aspekt, neboť na počátku éry personálních počítačů nebyla zrovna tato komunikace pro každého a každý pamětník si jistě za desítky let vývoje nese zkušenosti jak dobré, tak špatné. Podobné zkušenosti postupem času nabýváme s běžnými spotřebiči a ve své době byly ukázkou například mobilní telefony, kde se ergonomie obsluhy zpočátku velmi lišila značka od značky.

(Gerhard Leitner, str. 6-7)

„ . . . technologies that fit the human environment instead of forcing humans to enter theirs.“

(Gerhard Leitner, 2015, str. 8)

Tedy v Leitenerově pojetí se nestáváme otroky technologií ale ony tak nějak splynou s naším prostředím. Samozřejmě z psychologického pohledu bychom museli daleko více technologie diferencovat. Technologie typu termostat, či otevírání vrat garáže nás asi příliš neovlivní a jen stěží na nich získáme závislost, nicméně v případě mobilních telefonů, tabletů a PC to je již na povážení.

Svůj na člověka orientovaný přístup pak rozděluje do tří kategorií: *Elderly, Energy a effectuation*, tedy starší, energie a provedení či uskutečnění. Elderly – primárně souvisí se změnou demografického složení obyvatel nejen Evropy a nedostatkem personálu, který by o stárnoucí populaci pečoval. Energy – je kategorie zabývající se spotřebou energie, její kontrolou, měřením a regulací, neboť v souvislosti s ubývajícími fosilními palivy, takto můžeme významně přispět k šetření energiemi. Například i odpojováním desítek zařízení, která zcele zbytečně konzumují energii ve stand-by režimu. Efektuation – hlavně v dobách ekonomických krizí, potřebujeme co nejvíce snížit náklady. Toho můžeme dosáhnout i využitím on-line služeb, místo služeb off-line. Lidé jsou stále častěji konfrontováni s bankovními, státními či medicínskými on-line službami, což může být pro některé problém. Je tedy potřebné dělat technologie přístupné co nejširší skupině lidí. Lidé se nechtějí učit pracně obsluhovat technologie, chtějí je využívat ke zkvalitnění svého života.

(Gerhard Leitner, 2015, str. 9)

Koncový uživatel se soustřeďuje na uživatelský interface, ten je proň vlastně reprezentací celého produktu. Ale v tomto případě je uživatel doslova obklopen produktem samým. Varieta vstupních zařízení může býti velmi široká. Od jednoduchých dvoustavových vypínačů na stěně, až po integrovaná zařízení jako jsou smartphony, tablety, panely apod. A dnes samozřejmě mezi těmito dvěma extrémy, široká škála všemožných IoT zařízení – zařízení z kategorie Internet Of Things tedy internet věcí. Dalšími prvky mohou být prvky zpětné vazby, které nám umožňují ověřit, že k požadované akci opravdu došlo, jako různá čidla, či vstupy stavových signálů jiných zařízení, např. Systému EZS či stavu ovládání vysokého a nízkého tarifu spotřeby elektřiny (HDO). Pro to, aby systém mohl mít určitou „vlastní inteligenci“ potřebuje často mít nasbíráno dostatek dat, ze kterých lze vyvozovat nějaké predikce.

(Gerhard Leitner, 2015, str. 10-17)

1.3.1 Ambient inteligence

Ambient inteligence je další pojem vyskytující se v této oblasti a jedná se v podstatě o reakce systému na přítomnost lidí v daném prostoru. Na základě poznatků z čidel o pohybu osob v prostoru, systém dělá rozhodnutí a spíná požadované prvky. Ambient inteligence je však obvykle budována bez prvků umělé inteligence, tedy její funkcionalita je omezena. Tak jako umělá inteligence, stejně i lidská je v psychologii a obdobných vědách popisována mnoha teoriemi a různými modely. Definic je mnoho a neexistuje konsensus. Běžná definice říká asi zhruba to, že inteligence je schopnost porozumět komplexním ideám, tak abychom se byli schopni adaptovat v prostředí, ze zkušenosti se poučili a produkovali různé formy uvažování k překonání překážek. Tyto schopnosti se budou lišit na základě různých situací v různých kontextech a rozhodnutá budou vznikat na základě různých kritérií. V této definici můžeme samozřejmě najít i inspiraci pro to, jak by mohla fungovat inteligence umělá, ale také určité rozdíly, které mohou být zásadním problémem při interakci člověka a stroje. (Gerhard Leitner, 2015, str. 17-18)

1.3.2 Potenciál systému

Každá technologie má určitý potenciál člověka rozrušit. Ze všech stran pískající zařízení, která dokončila svou činnost a požadují od obsluhy minimálně zmáčknout nějaké tlačítko, nejlépe všechna najednou v tu nejnevhodnější dobu apod. Ale může to být i jinak. U chytrých zařízení můžeme stav a průběh sledovat pomocí telefonu nebo nám přímo může zaslat E-mail, či SMS. Moudrý systém by mohl například rozpoznat potřeby a motivace uživatelů a podle nich sám prostředí přizpůsobit. Na druhé straně musí být i přímé ovládání co nejjednodušší, aby například uživatel nemusel projít celé menu, když si chce rozsvítit na chodbě.

(Gerhard Leitner, 2015, str. 17-18)

Není jednoduché, na jedné straně komponenty co nejvíce standardizovat a na straně druhé se snažit respektovat lidské potřeby. Ty jsou v mnoha aspektech u mnoha lidí podobné, ale v mnoha jiných se mohou zcela diametrálně odlišovat, v závislosti na mnoha faktorech. Mimo jiné je to faktor věku, potažmo i generačních zvyklostí, faktor kulturní, národnostní, genderové, někdy i zvyklosti krajové a mnoho dalšího. Není proto zdaleka jednoduché vyhovět potřebám všech uživatelů. Každá větší univerzálnost přinese komplikovanější obsluhu. Naopak specifičnost pro určitou skupinu, ovládání i návrh pro danou skupinu obsluhu zjednoduší. Bohužel zpravidla na úkor toho, jak velký segment trhu bude takto uspokojen. Každá z cest má své výhody a nevýhody. Jak již bylo zmíněno výše, jeden z rostoucích segmentů trhu je stárnoucí populace. Ta pravděpodobně dá přednost jednoduchému a intuitivnímu ovládání, navrženému v souladu s tím, co v české kotlině nazýváme selským rozumem, před košatostí funkcionality, barevností a chytlavostí interface. Naopak jiný segment trhu osloví pravý opak. My se tedy budeme věnovat návrhu aplikace spíše jednoduché a srozumitelné, bez potřeby oslňovat designem a šíří funkcionality.

1.4 Hardwarové a softwarové možnosti domácí automatizace

1.4.1 Hardware

Pro budování domácí automatizace je samozřejmě možno využít existující hardware, který je v domě instalován pro různé existující aplikace. Standardně bude využívána počítačová síť, tedy její aktivní i pasivní prvky mnohých výrobců a to jak klasické metalické spojení Ethernet 802.3, tak bezdrátové, 802.11 ale i řada dalších možností bude popsána dále. Z běžných síťových aktivních prvků budeme používat například routery, DSL modemy, Wi-Fi klienty a access-pointy, switche apod. Kabeláž metalického rozvodu pro připojení zařízení, zásuvky spojky aj. Pro komunikaci lze využít i stávající rozvod elektrické sítě 230V. Pro domácí použití si zpravidla vystačíme s relativně levnými produkty různých značek a pravděpodobně s podstatně nižšími náklady, než například v průmysloví automatizaci. Co se týče vlastních „akčních prvků“ domácí automatizace, existuje opět široká paleta počínaje od průmyslových pro náročné aplikace a konče u Čínských produktů, které lze objednat za několik dolarů v mnoha e-shopech. Pro mnoho aplikací i ty nejlevnější produkty budou dostačující a i v naší práci budou z mnoha důvodů také použity. Pochopitelně, že v případě komerčního využití je potřeba důkladně zvážit patřičnou lokální legislativu, aby nedošlo k instalaci něčeho, v je v rozporu s českými normami. Pro účely experimentální si však s těmito produkty zcela vystačíme. Dlužno však asi dodat, že většina těchto produktů je mnoha českými firmami běžně distribuována i na

českém trhu, jen zhruba za dvoj až trojnásobné ceny. Měly by tedy, alespoň hypoteticky, splňovat naše normy.

1.4.2 Základní platformy

Standardní komerční systémy domácí automatizace jsou zpravidla vystavěny na bázi určité kompatibility a interoperability mezi jednotlivými prvky. To neznamená, že by musely býti kompatibilní systémy jednotlivých výrobců mezi sebou. Standardizace je sice v takových oblastech často podporována tím, že standardy určují například konsorcia velkých firem, ale ne vždy to funguje jak bychom si představovali. Zdaleka ne každý standard se povede. Pokud ano, umožňují pak tyto standardy kooperaci zařízení mnoha různých výrobců, tedy potažmo široké možnosti funkcionality a mnoho rozličných prvků. Systém moudrého domu, může být postaven na *service oriented architecture (SOA),* která poskytuje prostor různým službám a poskytovatelům a taktéž zahrnuje hardware. WISE platforma je implementována na bázi OSGi midle-ware architektury. OSGi platforma je zdarma a otevřená. Taktéž má rostoucí komunitu vývojářů kteří přispívají k jejímu rozvoji a je možno jí uživatelsky přizpůsobit či rozšířit. Může být považována za významnou právě v oblasti domácí automatizace. Také je na této architektuře postaveno mnoho výzkumných projektů. Vrstva vztahující se k hardware zajišťuje integraci připojených zařízení a jejich abstrakci v několika kategoriích A, B a C. Pak je možno k nim přistupovat pomocí teto vrstvy dle kategorií v každé z nich jednotně, tak jako by byly součástí jednoho systému. (Gerhard Leitner, 2015, str. 37-48)

Platforma umožňuje integrovat funkce bezpečnosti, zdraví, řízení spotřeby, funkce zábavy a pohodlí. Důraz je třeba klást na dostatečnou miniaturizaci komponentů, aby snadno zapadly do stávajícího prostředí, taktéž na cenu realizace a neméně na flexibilitu konfigurace či případné rekonfigurace nebo budoucího rozšiřování. Aby nám kvalitní software mohl nabídnout vše co potřebujeme, je důležité mít potřebný hardware v co nejširší variabilitě, abychom mohli dosáhnout požadované variability. Většinou dnes jako první volba vítězí bezdrátové systémy. Je to hlavně z důvodu minimální pracnosti a zásahů do stávajícího stavu. Na trhu je mnoho hardware se značkou „smart home“. Dovoluje nám to velkou flexibilitu a možnost konfigurovat systémy uživateli „na míru“. Taktéž můžeme zahrnout uživatelský hardware postavený na široké platformě Arduino, což nám umožní kompenzovat případné nedostatky konkrétního zařízení na trhu vlastním návrhem. Taktéž může pomoci pokud je potřeba propojit například dva nekompatibilní systémy. Software a hardware však není to co tvoří moudrý dům. V první řadě je to filosofie řízení. Dva základní principy můžeme považovat za nejdůležitější. Prvním je tzv. *Explicitní interakce* což je reakce systému na uživatelovu záměrnou okamžitou potřebu, projevenou interakcí uživatele se systémem. Jako druhý princip označme *implicitní interakci* což je princip, který nereaguje přímo na povely uživatele, ale je založen na analýze toho, co se uvnitř domu děje, tedy jak vlastní aktivity obyvatelů domu, tak i na analýze jejich explicitních interakcí se systémovými prvky. Dá se říci, že se systém pokouší analyzovat rituály a zvyky obyvatelů domu. Tady je možno spatřit rozdíl mezi chytrým a moudrým domem. Právě datová znalostní báze, která umožňuje predikovat potřeby uživatele je hlavní silou moudrého domu. Standardní systémy neumí analyzovat data natolik aby rozeznaly záměry, či určité vzorce chování uživatelů.

(Gerhard Leitner, 2015, str. 49-51)

1.4.3 Explicitní interakce

Prostředky explicitní interakce mohou být velmi různé. Nejjednodušším prostředkem může být klasický vypínač, či tlačítko na stěně, ale také stisk tlačítka na dálkovém ovladači nebo přímo na nějakém zařízení domácnosti. Ideální představa je taková, že by všechny prvky měly být integrovány tak, aby vytvořily dojem holistické jednoty celého domu. I obyčejný vypínač dnes může být vybaven technologií IoT a stává se tak hybridní kombinací klasického vypínače, ale například zcela bez nutnosti spínání silových vodičů, které může být realizováno až přímo v daném spotřebiči, např stropním svítidle, IoT spínačem. Taktéž je potřeba u takových zařízení zpětné vazby, abychom znali aktuální stav daného spotřebiče. To nám umožňuje spínat daný spotřebič jak standardním vypínačem, tak například pomocí dálkového ovladače či webového rozhraní, ale i na základě splnění určité podmínky. Systém může postupně vytvářet mentální modely uživatele a spínat prvky na základě nich. Existují i možnosti ovládání gesty snímanými kamerou případně ovládání hlasem. Zde již je potřeba poněkud větší výpočetní výkon centrální jednotky. Jedním z možných konceptů může být URC (Universal Remote Console). Tato univerzální vzdálená konzole se snaží o určitou standardizaci interface hardwarové vrstvy a dovolit tak jednotný přístup k nim. Přistupuje tedy k nižším vrstvám WISE systému, pomocí nějaké abstrakce a standardizace protokolů. Zaměřuje se tedy spíše na hardware a *back-end software*, než-li na *front-end* vrstvu, tedy uživatelské rozhraní. Může tedy pomoci s integrací jednotlivých zařízení. Ne vždy ale musí být dopady jen pozitivní. Neblahý dopad vývoje uživatelských interface, co se týče jejich použitelnosti, lze najít například u mnoha mobilních aplikací. Takže můžete mít jednu aplikaci pro ledničku, druhou pro světla, třetí pro termostaty atd. Rozhodně můžete očekávat, že každá aplikace bude jiná, s jinou filosofií obsluhy a dalšími vlastnostmi. Je tedy vidět, že určité konvence mají smysl, nicméně je potřeba, aby byly opravdu navrženy moudře. Velmi snadno můžeme vidět vývoj konvencí na mobilních či webových aplikacích. Kupříkladu nákupní košík v e-shopech, vyhledávání tamtéž, stačí menší změna oproti běžným zvyklostem a může způsobit velkou rozladěnost uživatelů. Dnešní podoba e-shopů se vyvíjela a konvence se stabilizovaly mnoho let. U mobilních zařízení například způsob nastavení času, například pro budíky. Ten kdo vyzkoušel různé značky MP3 přehrávačů, či starších tlačítkových mobilních telefonů, mohl zažít velké zklamání, při změně značky, tedy i způsobu obsluhy. Velmi zásadně je výsledný produkt i věkem jeho vývojářů a také jejich zkušenostmi z jiných oborů, které někdy bývají blízké nule. Co se týče interface v moudrém domu, je situace obtížnější, neboť musíme skloubit řadu různých technologií a pokusit se najít vhodný způsob ovládání. Pokud si vezmeme například ovládání teploty. Klasický radiátor má kolečko, kterým lze regulovat, nicméně v interface smart-home bývá toto nastavení zpravidla realizováno pomocí tahového slideru, který je standardní komponentou většiny software pro tvorbu uživatelského rozhraní. I taková maličkost může být uživatelem považována za neintuitivní. Celá řada situací v domě může být ošetřena standardní SMART technologií, ale ze své podstaty nikoliv WISE. Například komunikace se starými audiovizuálními přístroji může být problematická, uživatel si zvykl na nějaké ovládání a nechce se učit jiné. Starší uživatelé mohou mít značný problém i pokud jim jen nahradíme klasické dálkové ovládání nějaké značky se specifickou obsluhou za univerzální dálkový ovladač. Univerzální ovladač se snaží býti univerzální i ve filosofii užití tlačítek. Pokud je k ovládání užito webového rozhraní, je dobré uvažovat i o responsivním designu, alespoň tam, kde to je možné a dává to smysl, což nám umožní provoz na různých zařízeních a platformách. Jako další názorný příklad konvencí si můžeme uvést osobní automobil. Taktéž je vyráběn mnoha výrobci, ale ovládání se po mnoha letech nějak unifikovalo. Kdo střídá vozy různých značek, jistě ví, jaký může být velký problém si zvyknout i na malou odlišnost, například v ovládání stěračů nebo tlumení světel. Je možno uživatelům umožnit, aby si některé prvky vytvořili

tak říkajíc „na míru“ sami, například za použití vizuálního programování. Tato možnost se nazývá End user development (EUD). To umožňuje pokrýt v podstatě všechny potřeby zákazníka, které komerční zařízení přes jejich velkou variabilitu pokrýt neumí. A také to může být pro některé uživatele i motivace, podobně jako tzv „Ikea bias“, tedy tzv. kognitivní zkreslení, v tomto případě pocit, že když jsem něco dodělal doma, jsem vlastně téměř tvůrce. Uživatel také ztrácí obavy ze systému, do kterého je mu umožněno nahlédnout hlouběji. I důvody nedůvěry ke komerčním systémům, které mohou být a často jsou zneužity, může být motivací pro EUD, neboť uživatel má pocit, že má nad svými daty kontrolu. Taktéž cykly vývoje standardního software a hardware ne vždy pokryjí potřeby všech uživatelů.

(Gerhard Leitner, 2015, str. 51- 57)

1.4.4 Implicitní interakce

Jak již bylo výše zmíněno, implicitní chování systému je vyvozeno ze znalostní báze nasbíraných dat a jedná se vlastně o jakousi predikci chování uživatele. Tedy nevyžaduje explicitní spuštění vyvolané nějakou akcí uživatele, například stisk tlačítka. Jedná se tedy o určitou umělou inteligenci (UI), v angličtině obvykle artifical inteligency (AI). Výše byl zmíněn i pojem ambient inteligence, pro kterou Leitner používá také zkratku (AI), což může být poněkud matoucí. Jak již bylo výše zmíněno WISE platforma využívá open OSGi architektury díky níž je možno do jednoho systému integrovat různá zařízení různých výrobců a komunikovat s nimi pomocí abstraktní vrstvy, tedy nezávisle na vlastním použitém hardware. AI na základě sledování uživatele pak sama rozhodne o určité aktivitě, aniž by došlo k přímé interakci s uživatelem. Klasický příklad může být stav, kdy uživatel v noci vstane a jde na WC. AI pak může sama podle získaných dat a v nich objevených opakujících se vzorců nasvítit celou cestu v žádoucí intenzitě světla a posléze jí zase zhasnout. Skutečná inteligence systému je ale ještě poněkud dále. Z nasbíraných dat derivuje vzorce obvyklé (rituály), na jejichž základě vykoná nějakou funkcionalitu a vzorce neobvyklé, na jejichž základě zase může zasílat varovné signály, že se děje něco výrazně neobvyklého. Pokud bude systém dostatečně inteligentní, může počítat i kolik obyvatel se kde nachází a zda se pohybují tak jak je zvykem. Pokud by nám například začali přibývat lidé v místnosti s jedním vchodem, aniž by prošli předchozí místností, asi nebude vše v pořádku a přišli třeba oknem. Pokud systém ví, že všichni dům opustili, může zapnout poplachovou ústřednu do módu střežení. (Gerhard Leitner, 2015, str. 57- 58)

1.4.5 Komplexní funkcionalita

Spojením jak explicitního tak implicitního modelu, dosáhneme požadované funkcionality. Explicitní část nám zajistí sběr patřičných dat které pak dále zpracuje AI a poskytne pro implicitní ovládání. Je záhodno, aby implicitní model neměl vyšší prioritu než sám uživatel a nemohl si takříkajíc dělat úplně cokoli. Musí být možnost zásahu do implicitních funkcí a nastavení optimální funkcionality. Pokud vše funguje jak má, měla by technologie zajišťovat fyzickou i psychickou pohodu uživateli. Kombinace lidské inteligence a AI, by měla zajistit výhodu pro WISE technologie oproti standardním SMART. Další funkce systému mohou být spojeny s vytápěním a ohřevem užitkové vody v závislosti na přítomnosti, či době nepřítomnosti uživatelů. Také systém může odpojovat zásuvky a spotřebiče, kde je malá pravděpodobnost, že budou v dohledné době použity, třeba v závislosti na tom, že se v dané místnosti nikdo nenachází. Můžeme ztlumit vyzvánění v době, kdy je pravděpodobné, že uživatelé spí. Stále však musí zůstat zachována možnost, aby uživatel mohl reagovat v případě potřeby explicitně.

(Gerhard Leitner, 2015, str. 57- 58)

1.4.6 Některé komunikační protokoly a vhodný hardware

Aby mohla jednotlivá zařízení v domácí automatizaci fungovat, musí se mezi sebou nějak domluvit. K tomu slouží komunikační protokoly. Komunikačních protokolů jsou jen pro domácí automatizaci desítky (v roce 2010 více než 50). Za populární můžeme označit zhruba desítku z nich. Některé z nich si popíšeme níže. Každý protokol s sebou nese obvykle i určitou faktickou implementaci. Některé využívají k přenosu kabely a to jak standardní rozvody Ethernetu tak i speciálně položené kabely pro daný účel, například pro EZS (elektronický zabezpečovací systém), či mohou využívat i silové rozvody elektrické Energie, např: protokol X-10. Další velkou rodinou jsou pak různé protokoly bezdrátové. Samozřejmě se mohou vyskytnout i rozvody optické. Každé řešení má své výhody, jak již bylo výše zmíněno, a taktéž je vhodné pro určitou oblast použití. Pro vysokou bezpečnost a vysokou rychlost stále platí jako nejlepší nosné médium klasický drát, zpravidla twisted pair CAT 5 nebo CAT 6, případně optika. Samozřejmě existují i protokoly využívající menší počet vodičů, a nižší rychlosti, jako např.: I2C, OneWire, CANBUS, Spi. Tyto však spíše budou využity na komunikaci uvnitř jednotek nebo na komunikaci s čidly, než na komunikaci jednotek s centrálou. Bezdrátové spojení jsou pak dle typu různě rychlá, různě zabezpečená, či nezabezpečená, různě spolehlivá a mají různý dosah.

(AL-QUTAYRI, 2010, str. 4 - 5)

X10

Jednou z možností komunikace po drátu je použití protokolu X10 který umožňuje komunikovat po stávajícím síťovém rozvodu 230V. To je na jedné straně velmi pohodlné, neboť řada zařízení je již k němu připojena. Nabízí se i jako výhodný pro ovládání zásuvek a světel, zařízení jsou relativně levná a instalace jednoduchá. Nicméně má i jisté nevýhody. Například v bytovém domě, bude přesahovat samotný byt a může být ovlivňován, či ovlivňovat zařízení jiných vlastníků. Příkazy posílané pomocí X10 mohou být například zapni, vypni, nebo ztlum svit na 50%. Každé X10 zařízení má vlastní adresu skládající se ze dvou částí *unit code* a *house code.* House code je písmeno A-P, tím se dá mírně kompenzovat ovlivnění nejbližšími sousedy, ale vzhledem k pouhým 16 možnostem, nepříliš. Přeběhu lze zabránit filtry v rozvaděči. Druhá část X10 kódu, je kód jednotky 0 – F hexadecimálně, tedy opět jen 16 možných voleb, tedy nemnoho. Každý z X10 modulů je postaven tak aby některé povely akceptoval a jiné ignoroval. Tedy například spínač zásuvky kávovaru nebude akceptovat povel pro setmění stmívače. Tedy pak příkaz pro všechny jednotky bude proveden jen jednotkami které takový příkaz akceptují. Po odeslání zprávy se nic dalšího nekoná. Přijímače neodesílají potvrzení. Sice existují *two-way spínače*, které na dotaz vrací svůj stav, ale jsou výrazně dražší, neboť vysílací část je konstrukčně výrazně náročnější a proto má i značně vyšší cenu. Abychom tedy zajistili validitu přenosu, pošlou se data dvakrát po sobě, a oba pokusy se porovnají. Celkem to trvá asi 0.64 sekundy. Tato prodleva může být nepříjemná. Pokud má dojít například k rozsvícení světla netrpělivý uživatel mačká a mačká a světlo s prodlevou bliká a bliká. Protokol X10 má taky řadu problémů s rušením a zkreslením signálu, nemá kontrolu přenosu, ale na druhou stranu existují například spínače, které se dají zamontovat přímo do krabice zásuvky na stěně apod.

(GOODWIN, 2013, str. 19-21)

Vlastní obsluha sítě

Někdy může být jednodušší, vyrobit a poskládat si packet programově, a komunikovat na nižší síťové vrstvě. V naší práci to bude jednou použito, k obsluze periferie, tvořené Arduinem NANO, doplněným jednoduchým Ethernet *shieldem*, který neumí TCP socket. Bližší informace najdeme v realizační části této práce.

C-Bus

C-Bus system byl vyvinut Australskou firmou Clipsal, původně k dálkovému ovládání světelných systémů, v komerčních centrech, arénách apod. K tomu byla potřeba aby fungoval na dlouhých kabelech a dokázal adresovat větší množství koncových zařízení, než bylo běžné v domácí automatizaci. Délka kabelu je maximálně 1 km a počet zařízení je 100 v jedné subsíti. Sub-sítí může být celkem 6 propojených síťovými mosty. Na rozdíl od X10 se u C-Bus nevysílá signál po silovém vodiči ale odděleně v kabelech CAT-5 twisted pair. To sice zvyšuje počáteční cenu, ale jakmile jsou položeny, nároky na údržbu jsou minimální. Nedávno byla představena i bezdrátová verze C-Bus, která tyto náklady redukuje, může zahrnovat 128 bitové šifrování, pro vyšší bezpečnost. Pro domácí automatizaci je vhodná verze kabelová, pokud byly kabely již položeny, s možností pozdějšího rozšíření bezdrátovou, pokud kapacita kabelů bude překročena.

(GOODWIN, 2013, str. 50-55)

RF 433 MHz

Další z možností je bezdrátový přenos na frekvenci 433 MHz. Je používán u meteo stanic, malých dálkových ovládání typu „klíčenka“, čteček RFID čipů apod. Udávaný dosah, bývá cca 25 m. Pro všechny vysílače postačí na řídící jednotce jeden přijímač. Případně je možno použít i konvertory – gateway mezi 433 MHz a X10. Vhodné prvky jsou například zařízení zahrady, kde připojení kabelem není příliš vhodné.

(GOODWIN, 2013, str. 22)

Z-Wave

Z-Wawe je další z protokolů určených pro domácí automatizaci. Vznikl v roce 1999, jako proprietární protokol. Dnes vyrábí Z-Wave zařízení sdružení cca 200 výrobců Z-Wave Alliance a jejich portfolio tvoří zhruba 600 různých zařízení. Komunikace Z-Wave zařízení probíhá na frekvenci 900 MHz. Rychlost přenosu je kolem 40kbits/sec. Ale některá zařízení zvládnou až 100kbps. Dosah signálu je v závislosti na prostředí zhruba 25 metrů. Zařízení mohou posílat signál dále jako repeater a tím dosah prodloužit. Snadno však může při těchto rychlostech dojít k zahlcení komunikace, je-li zařízení použito jako repeater. Taktéž existují huby, na které se zařízení mohou připojovat, například do USB socketu nebo připojené na Ethernet. Existuje i přímo kontrolér pro Raspberry Pi minipočítač. Z-Wave zařízení se po zapnutí spojí a spáruje s nejbližším kontrolérem na základě síly signálu a podle toho se pak vytváří routovací tabulka. Pro zabezpečení je použito párování jako například u Bluetooth zařízení. Tedy je nutno na obou zařízeních podržet tlačítko, kterým potvrdíme, že spolu mají komunikovat. Takto se výrazně omezí bezpečnostní útok typu man-in-the-middle. Z-Wawe má podobně jako X10 tzv. *house-code*, nazývaný zde též *Network ID*, a druhou část tvoří *unit – code* nebo též *Node ID*. V tomto případě však máme možnost vybírat z celkem 232 různých nódů v jedné síti, což je o poznání lepší, než 16 v případě X10 a u běžných domů bychom si s tímto počtem měli vystačit. V případě, že bychom si nevystačili, je možné vytvořit další síť a se stávající ji

spojit bridgem – síťovým mostem. Největší výhodou Z-Wave, je velká kompatibilita a interoperabilita mnoha prvků vyrobených mnoha firmami. Každý výrobce musí Z-Wawe alianci žádat o svůj kód, tedy systém není určen pro open source produkty. Leč za pomoci hackerů a reverzního inženýrství se povedlo tuto bariéru prolomit a je několik možností, jak open source využít. Jsou to například projekty *Open – Z-Wave* nebo *LinuxMCE* .

(GOODWIN, 2013, str. 48 - 50)

ZigBee

ZigBee je další z bezdrátových protokolů. Umožňuje komunikaci na 10-100 metrů a na rozdíl od dalších nepotřebuje zvláštní kontrolér. Každé ze zařízení, může nabývat jak funkce koordinátora, tak routeru nebo koncového zařízení. Zařízení fungují v mesh topologii, takže není problém aby se dosah zvětšoval. Zabezpečení komunikace je 128 bitovým klíčem. A zařízení jsou levná, čipy vyrábí mnoho výrobců. Taktéž ZigBee má svou alianci, která sdružuje kolem 400 firem dodávající na trh přes 600 různých výrobků. Bohužel členství v alianci je drahé. Software pro obsluhu není možno zaintegrovat do jádra Linuxu, dík GPL licenci.

(GOODWIN, 2013, str. 50)

Protokol MQTT

MQTT (dříve: Message Queuing Telemetry Transport, dnes MQ Telemetry Transport) je protokol pro komunikaci například IoT tedy internetu věcí. Jelikož bude v naší práci použit, rozebereme jej podrobněji. Je postaven na návrhovém vzoru *publisher – subscriber*. Je to jednoduchý protokol, sloužící k zasílání zpráv. Je vhodný pro webové aplikace k řízení IoT. Samozřejmě , že je potřeba dbát na patřičnou bezpečnost, při použití tohoto protokolu, jelikož není nijak šifrován, tedy je snadno odposlouchatelný a zneužitelný. Výhodou je, že můžete komunikovat mezi mnoha různými programovacími jazyky, což se u IoT dá předpokládat, že bude běžná situace.

(HILLAR, 2017, str. 16)

MQTT protokol je vhodný pro aplikace IoT, embeded a mobilní aplikace. Je jednoduchý a přenos dat nevyžaduje velkou režii. Pro zasílání dat není potřeba mnoho paketů a lze posílat relativně velká množství dat. Je asynchronní a obousměrný. Není problém posílat zprávy z jednoho zdroje mnoha klientům i naopak. Architektura je orientována na události, a je vhodná pro zařízení vždy připojená i připojená jen občasně. Zvládá velké množství zařízení, typicky až stovky. Taktéž je vhodný pro nespolehlivé sítě, s nekvalitním spojením. Je vhodný pro zařízení s bateriovým napájením, kde je vyžadována nízká spotřeba a také je schopen velmi rychlého skoro real-time doručení. Díky použití návrhového vzoru publisher – subscriber, klienti neví o existenci druhých, komunikují jen se serverem, také nazývaným *broker.* Vysílající klient se nazývá *publisher* a klient přijímající je *subscriber ,* ti se navzájem neznají a oba komunikují jen s brokerem. Příjemce může obdržet zprávu i později, viz dále, ale nijak tím neblokuje další vysílání klienta, který mu zprávu zaslal. Pro doručení zprávy stovkám subscriberů postačí aby klient poslal jednu zprávu na broker a ten zajistí distribuci. Subscriber může taktéž vystupovat v roli publisher, například při potvrzení obdržení zprávy.

(HILLAR, 2017, str. 32- 34)

Filtrování zpráv

Abychom doručili subscriberům jen to, co požadují, je potřeba na straně brokera zprávy dle požadavků subscriberů filtrovat. Každý subscriber pak dostává jen takový typ zpráv, o který si sám požádal. Toto filtrování se nazývá *Topic-base filtering* nebo *subject – base filtering.* Broker nezajímá obsah zprávy, tzv. *Pyload,* ale jen její *topic* na základě něj pak zprávy filtruje a distribuuje.

(HILLAR, 2017, str. 34)

Client-server

Jak již bylo výše zmíněno, MQTT používá spojení typu TCP. Existuje množství knihoven pro různé platformy a programovací jazyky, které nám umožní vytvoření MQTT serveru. Každá však poskytuje určitou omezenou funkcionalitu, což je potřeba při návrhu zvážit. Jakékoli zařízení umožňující TCP stack se může stát klientem MQTT. Je mnoho dostupných MQTT serverů pro různé platformy. MQTT server je jakýsi centrální *hub* návrhového modelu, je zodpovědný za autentizaci a autorizaci klientů. Klient a server navazují TCP spojení, které se zpravidla udržuje stále, tedy dokud nedojde k jeho přerušení nebo ukončení klientem. Pokud se nejedná o klienty, kteří se připojují jen sporadicky, například low-pover s bateriovým napájením. Klient pošle serveru *CONNECT controlpaket* se zprávou obsahující potřebné informace pro zahájení spojení. MQTT server provede autentizaci a autorizaci vrátí *CONNACK control paket*. Pokud by klient poslal špatný *CONNECT control* paket, pak server spojení ukončí. *CONNECT control packet* musí obsahovat patřičné informace. V datové části zprávy – pyload musí být obsaženo Id klienta - *ClientId* (každý klient musí mít unikátní Id), tedy pokud klient Id nevyplní, bude mu vygenerováno serverem. To závisí na obsah *CleanSession* field. *Clean session flag* obsahuje boolean hodnotu, na základě níž se určí chování serveru při odpojení a znovupřipojení klienta. Hodnota *True* říká, že spojení bude udrženo jen dokud nedojde k rozpojení, pak budou všechny informace o spojení zahozeny. Naopak *False* udává, že i při rekonektování dojde k obnovení komunikace při zachování původních parametrů a navíc, MQTT server bude stále ukládat všechny zprávy pro klienta, které mu předá, jakmile se znovu připojí (*persistent session*). Pokud klient požaduje aby se pracovalo i s jeho *user name ,* nastaví *user name flag* na hodnotu *True* a přidá své jméno, stejně tak pokud chce aby byl požadován *password* nastaví *Password flag* na *True* a specifikuje heslo. Dalšími poli a flagy v CONNECT paketu jsou: *ProtocolLevel* klientem požadovaná verze MQTT protokolu, *KeepAlive* [čas v sec] – klient se zavazuje, že bude do určité doby vždy něco posílat, pokud nemá data, pošle *pingreq* jakmile se server v dané době nedočká od klienta obdoby, spojení ukončí. *Will, WillQoS, WillRetain, WillTopic WillMessage* jsou flagy, kterými klient požaduje splnění své „poslední vůle“ tedy toho, co server provede ve chvíli, kdy spojení je ukončeno nebo přerušeno. Co se týče serveru a jeho packetu CONNACK, odpovídá klientovi a v hlavičce nastavuje tyto flagy: *SessionPresent* který nám říká zda bylo požádáno o clean session, či nikoli, *ReturnCode* pokud proběhla autentizace a autorizace správně, pak je tento flag nastaven na nulu, pokud došlo k chybě, je vráceno číslo od 1 do 5, které specifikuje druh chyby.

(HILLAR, 2017, str. 36 - 54)

Jedním z MQTT serverů, který funguje jak na Linuxu, tak na Windows ale i na iOS a bude použit i v naší práci je server *Mosquitto*. Pro pokusy a odlaďování MQTT komunikace, použijeme software klienta MQTTFX https://mqttfx.jensd.de/index.php ,

který nám dovoluje zasílat zprávy, přihlašovat si odběr a sledovat co se na síti děje. V našem případě bude použita verze pro Win 64 bit. V Linuxu můžeme například použít řádkové Mosquitto_pub, pro publish a Mosquitto_sub, pro subscribe.

Vytváření tématu zprávy

Publisher si vytvoří téma (topic – subject) tak jak je potřebuje. Struktura tématu je podobná stromové struktuře file systému na disku. Příkladem může být: „Termostaty/1_patro/loznice/temperature/ a v pyloudu pak bude např. naměřená teplota. Pokud se chce subscriber přihlásit k odběru, pak může vybrat např vše: # nebo všechny termostaty: „*Termostaty/#*“, všechny termostaty v prvním patře: „*Termostaty/1_patro/#*“ apod. Názvy témat jsou case senzitive, tedy je potřeba si na to dát pozor. Můžeme použít všechny znaky z UTF 8, vyjma dvou, což je „+“ a „#“ které se používají jinak. Je však z důvodu kompatibility s různými knihovnami a platformami, komplikovanějším znakům raději vyhnout, například je dobré nemít na začátku znak „$“. Podobně, jako když tvoříme například URL. Strukturu tématu je třeba volit s rozmyslem, aby logicky vyhovovala tomu, co budeme v realitě skutečně potřebovat od sebe oddělit.

(HILLAR, 2017, str. 65 - 68)

Quality of service

QoS aneb Quality of Service, je požadovaná úroveň kvality síťových služeb. Standardně je nastavena hodnota *level of QoS* na „0“. Připomeňme, že úrovně QoS mohou být nastaveny různě mezi komunikujícími stranami a to i tak, že jako klient můžeme s jinou úrovní QoS publikovat a s jinou naopak přijímat. Jednotlivé úrovně se liší ve významu doručení zprávy. Úroveň 0, znamená nejvýše jedno doručení, tedy zpráva je jednou odeslána a zda došla, se již neřeší. Úroveň 1 – nejméně jedno doručení, zde naopak zpráva dorazí minimálně jednou, ale může dorazit i vícekrát, prakticky to znamená, že si vysílající strana vyžádá potvrzení a dokud jej nedostane, opakuje vysílání znovu a znovu. Tedy pokud se nám budou ztrácet potvrzovací pakety, bude zpráva vysílána neustále opakovaně. Tudíž na straně přijímací by měla být logika, která si duplicitu v případě potřeby kontroluje. Level 2 QoS pak znamená – přesně jedno doručení. Komunikace a potvrzování je vytvořeno tak, aby k duplicitě nedocházelo, což samozřejmě zvyšuje režii na obou stranách přenosového kanálu. Nicméně, máme-li stabilní síť a nevadí, že se zpráva doručí opakovaně, vystačíme si s QoS level 0, a tím, že pakety posíláme opakovaně. Naopak v případě, kdy by duplicita příkazu způsobila problém a je pro nás důležité aby vše došlo, využijeme QoS 2. Pokud publisher pracuje s vyšší úrovní než subscriber, musí jí snížit na úroveň subscribera.

(HILLAR, 2017, str. 68 -70)

Zabezpečení komunikace MQTT

Bezpečnost je v řadě aplikací velmi důležitá. Samozřejmě, že někdy nás příliš zajímat nemusí, pokud nepřenášíme důležitá data a pohybujeme se jen ve vlastní vnitřní síti. Pokud ale například řídíme dron, není dobré, aby jej mohl řídit i někdo další a například jím někoho zranit. Vždy je bezpečnost zaplacena zvýšenou režií, potažmo tedy i šíří potřebného pásma k přenosu. Někdy nás to donutí i použít výkonově lepší prvky, jako procesory, IoT moduly apod. Tedy je dobré zvážit bezpečnostní rizika před tím, než budeme investovat do pořízení hardware. Je třeba zvážit další aspekty. Například použití

certifikátů může být komplikované, pro jejich potřebné množství, apod. Pro komunikaci se serverem MOSQUITTO, který bude použit i v naší aplikaci, můžeme využít zabezpečení na několika úrovních. Na úrovni sítě, můžeme například použít VPN (Virtual private network), pokud chceme rozšířit funkcionalitu kamkoli na světě. Na úrovni transportní vrstvy, jelikož MQTT využívá TCP, můžeme použít šifrování TLS/SSL. Prostřednictvím tohoto šifrování pak můžeme komunikovat mezi klientem a serverem, což bývá někdy označováno jako MQTTS. Na úrovni aplikační je to pak autentizace a autorizace, taktéž můžeme na této úrovni zašifrovat pyload, tedy přenášená data.

(HILLAR, 2017, str. 70 -72)

Java Script

Java Script je další možností pro domácí automatizaci. Může být spuštěn přímo v browseru, kde nám bude velmi užitečný při vytváření *front-endu*, například malování grafů, či práce s daty v tabulkách apod. Navíc, lze použít Node.js, což je verze Java Scriptu běžící na serveru, tedy potažmo na některých IoT zařízeních, kde se může starat jak o příjem dat, tak o vlastní funkcionalitu. Java Script se může uplatnit na straně browseru pro přímou komunikaci prostřednictvím *MQTT over WebSockets* což moderní browsery a například server MOSQUITTO umožňují. (HILLAR, 2017, str. 191 - 194)

IR Over IP

Často by se nám hodilo ovládat různá zařízení, jejichž jediné použitelné rozhraní je IR senzor. Samozřejmě, že i to je realizovatelné prostředky domácí automatizace a to v obou směrech. Můžeme tedy náš systém naučit povely z dálkového ovladače a pak je pomocí IR vysílače vyslat v požadovanou chvíli, například pro zapnutí a vypnutí klimatizační jednotky. Pak za jakéhokoli ovládacího interface, například mobilního telefonu, či webového rozhraní, můžeme ovládat vše, co má IR ovládání jak záměrně, tak automaticky systémem, třeba v danou dobu. Můžeme také využít stávající ovladač od nějakého zařízení nebo ovladač univerzální, jako klávesnici pro povely našemu chytrému domu. Stejně tak lze použít mobilní telefony, mající IR vysílač. Jako další možnost se nabízí prodloužení dosahu dálkového ovládání, tedy že na jednom místě domu, načteme co vyslal skutečný ovladač a na jiném vzdáleném místě domu, můžeme totéž opět vyslat a spustit daný spotřebič. Existují i software, které obsahují knihovny povelů stovek ovladačů, ale asi nejjednodušší cesta, například pokud potřebujeme jen několik povelů je, načíst sekvenci ze stávajícího ovladače a uložit.

(GOODWIN, 2013, str. 73 -77)

1.4.7 Jednoduché senzory a aktuátory

Světlo/tma

Na jednotlivých GPIO (general purpose IO) pinech, můžeme přímo snímat některé potřebné údaje. Na pinech digitálních jen hodnoty 0 nebo 1 (False/True) a na pinech analogových, dle druhu MCU hladiny napětí odstupňované například v rozlišení 1024 hodnot. Na digitálním vstupu můžeme patřičnou hodnotu 0/1 získat pomocí děliče napětí, stejně tak děličem můžeme analogový vstup přizpůsobit rozsah měření, jako u například běžného voltmetru. Tak můžeme vytvořit snímač světlo/tma, jako napěťový dělič, s

kombinací rezistoru a fotorezistoru. Fotorezistor má za tmy velmi vysokou rezistanci a za světla naopak velmi nízkou. Snadno tak vytvoříme logický signál světlo/tma. Hrozí však při vstupním signálu na hranici logické 0 a 1 že dojde ke kmitání, což je dobré nějak ošetřit buď na úrovni hardware, nějakým hysterezním obvodem nebo na straně software.

(GOODWIN, 2013, str. 99)

Spínání relé z GPIO

Podobně lze GPIO využít přímo ke spínání malé zátěže a pomocí tranzistoru, bipolárního, či unipolárního možnou zátěž zvýšit. Pro velké napěťové, či proudové zátěže můžeme zařadit ještě relé a zvyšovat spínanou zátěž a to i z jiného zdroje, například síťovou, dle potřeby. Taro relé pak mohou být buď přímo součástí zařízení samého nebo součástí „chytré zásuvky“. Samozřejmě toto lze jak přímo z centrální jednotky, tak i ze subsystémových, periferních jednotek.

(GOODWIN, 2013, str. 100)

Tlačítka

Jedno tlačítko je možné připojit na digitální vstup, s použitím pull-up, či pull-down rezistoru a to buď vestavěného (v některých MCU) nebo vnějšího. Zákmity lze opět ošetřit zevně hardwarově, například integračním kondenzátorem nebo v obslužném softwaru. Pokud potřebujeme dvě a více tlačítek je možné je snímat pomocí analogového vstupu, se zvolenou sadou rezistorů v napěťovém děliči, a rozlišit tlačítka dle úrovně napětí. Takto lze po jednom vodiči snímat i větší množství tlačítek, ale jsou zde rizika zákmitů a špatného kontaktu, což může způsobit špatné vyhodnocení. Opět, lze realizovat jak na centrální, tak na periferní – vzdálené jednotce.

(GOODWIN, 2013, str. 101)

Chytré žárovky

Dnes již je na trhu řada žárovek, tedy spíše LED žárovek, které mají ve své patici vestavěno ovládání vypínání, zapínání, stmívání, některé i míchání barev, včetně mnoha různých efektů. Ovládání těchto žárovek je realizováno mnoha různými způsoby. IR ovladačem, BT např z mobilního telefonu, X-10 nebo Wi-Fi. Samozřejmě, že s použitím patřičného software, je lze ovládat i hlasovým asistentem jako je např. *Alexa*.

(VANDOME, 2018, str. 7)

Chytré zámky

Dálkově ovládané zámky, obvykle například na vchodových dveřích, garážových vrat, vjezdové brány, či vchodových vrátek. Také by mohly otevírat například trezory apod.

(VANDOME, 2018, str. 7)

1.4.8 Autonomní systémy

CCTV kamery

CCTV kamery jsou další nedílnou součástí výbavy domácí automatizace a zabezpečení. V dnešní době se již výhradně používají IP kamery, tady kamery připojené do sítě, nevyžadující připojení přímo do PC. Ať již jsou kamery bezdrátové nebo připojené k Ethernetu kabelem, vždy vyžadují napájení, které může být realizováno jako PoE. Vždy však musíme počítat s tím, že do kamery povede minimálně jeden kabel, tedy i do bezdrátové musíme připojit napájení. Další významné rozdíly jsou mezi kamerami venkovními a vnitřními. Venkovní jsou schopny odolávat povětrnostním vlivům, ale jsou také patřičně dražší. Kamery mohou mít přímo vestavěn web server, kde je lze jak konfigurovat, tak přímo sledovat videostream. Mohou umožňovat detekci pohybu, odesílání mailů a také ukládání na FTP. Další vlastnosti kamer mohou být: noční vidění, s různými způsoby přisvětlení, zpravidla IR LED. Kamery mohou být uživatelsky natáčené, či samy aktivně natáčející se za pohybem, s možností otočení i 360°. Kamery IP bezdrátové využívají Wi-Fi, ale existují i kamery které mají speciální průmyslové bezdrátové standardy a speciální záznamová a kontrolní zařízení, ale to je již mimo zájem naší práce. Existuje mnoho software pro správu kamer, pro většinu OS. Některé univerzální, jiné proprietární. Samozřejmě nároky na CCTV v domácnosti versus například v hypermarketu budou asi výrazně odlišné.

(GOODWIN, 2013, str. 68 -70)

Robotické vysavače a sekačky na trávu

Další část domácí automatizace, zpravidla tato zařízení pracují autonomně, je otázka, nakolik by mohla být centrálně řízena, ale například sekačka trávy by mohla spolupracovat s meteo stanicí, či předpovědí počasí, či vysavač by mohl jezdit po místnosti jen pokud se v ní nikdo nepohybuje apod.

(VANDOME, 2018, str. 8)

Voice asistenty

Zcela samostatně uveďme hlasové (voice) asistenty. Tyto produkty se pokouší s člověkem komunikovat přirozeným jazykem a na základě například jeho povelů vykonávat určité činnosti. Vyhledávání na internetu, zodpovídání dotazů, objednávky různých služeb, např. Taxi, pizza, lékař. Ale stejně tak s patřičným dalším software a hardware by měly býti schopni ovládat i domácí automatizaci. V podstatě jen nahrazují například webové rozhraní, čímsi, co bychom mohli nazvat rozhraním hlasovým. Samozřejmě by asi bylo problematické provádět hlasovou konfiguraci a nastavení některých parametrů domácí automatizace, ale na povely typu rozsviť/zhasni apod. Muže být *voice asistant* relativně praktický. Jelikož je však toto téma samo o sobě velmi rozsáhlé a nehodláme jej v naší práci nikterak využít, zůstaňme jen u takto povrchního náhledu. Snad pro zajímavost uveďme, že voice asistant služby zpravidla nesou ženská jména a ve výchozím stavu mluví ženským hlasem. Jejich jména jsou například: Siri, Cortana a Alexa.

(VANDOME, 2018, str. 8)

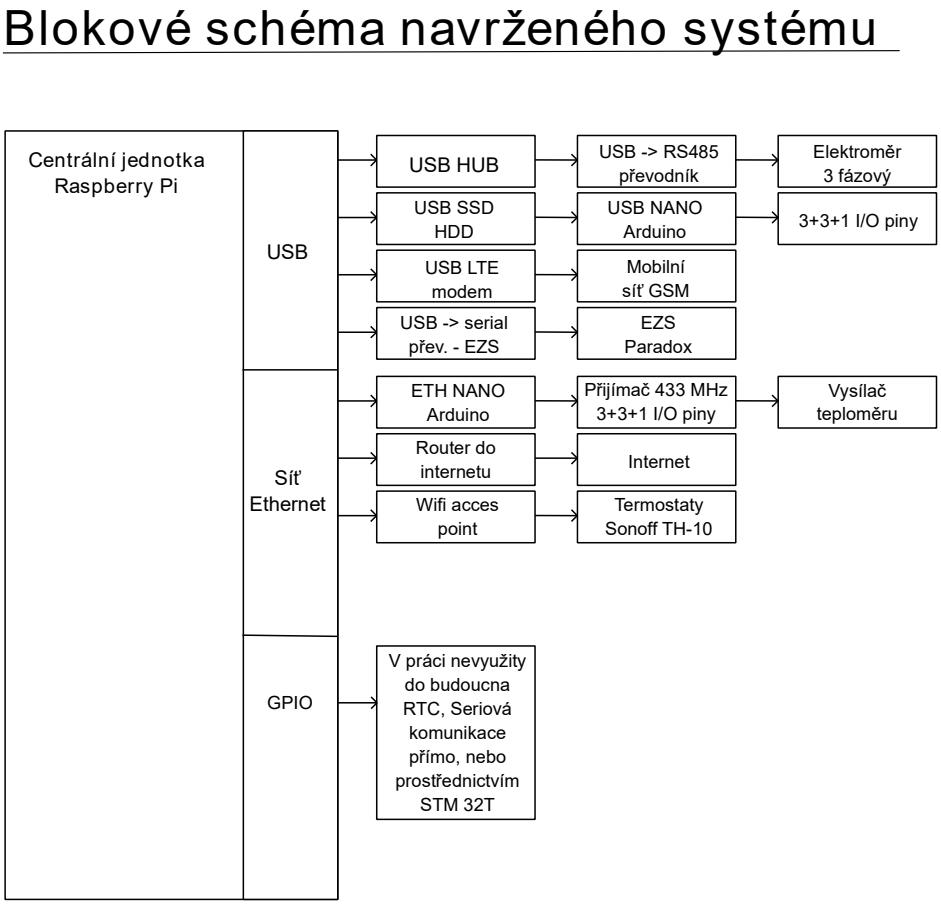
2 Praktická část

2.1 Návrh koncepce domácí automatizace

Jak již bylo výše zmíněno, realizace této práce bude následně fungovat v rodinném domu snad po dlouhá léta. Systém tedy bude koncipován tak aby tomuto účelu vyhověl. Vzhledem k rozsahu bakalářské práce, nelze řešit všechny možné funkcionality. V práci tedy zcela záměrně nebude nijak řešena komunikace s domácí zábavou, jako audio a video systémy, rozhodně zde nebude řešena komunikace s voice asistenty a taktéž zde nebude například řešen CCTV systém. CCTV má celkem dobrá a levná profesionální tovární řešeni, tedy se nabízí spíš do budoucna komerční systém s naší centrálou nějak propojit. Co se týče elektrického zabezpečovacího systému, pak se samozřejmě spolehneme opět na tovární řešení, tentokráte od kanadské firmy Paradox a jen se naučíme s ústřednou systému, Paradox Spectra, komunikovat. Dále budou řešeny analogové a digitální vstupy, digitální výstupy, komunikace s teploměry, síťovými moduly drátem připojenými i systémy připojenými bezdrátově pomocí Wi-fi. Systém bude také komunikovat s elektroměrem, který bude sledovat průběžně spotřebu případně v budoucnu ukládat a analyzovat naměřené údaje.

Do základní konfigurace pro tuto práci byla tedy zvolena určitá množina funkcionalit, které považujeme za esenciální. Bude se tedy jednat o logické vstupy a výstupy, tedy entity, nabývající hodnot pravda versus nepravda nebo jinak řečeno zapnuto/ vypnuto či 1 / 0. Tyto digitální hodnoty jsou naprosto běžné i v jakékoli nespojité regulaci a obklopují nás skoro všude. Vstupy a výstupy, budou pro účely této práce jen indexované entity, ale v reálném prostředí budou některé reprezentovat určitou stavovou informaci, v případě vstupů a určitý, úkon v případě výstupů. Tedy uveďme si některé příklady, pro získání lepší představy o tom, jak bude systém fungovat v reálném prostředí.

2.1.1 Základní konfigurace



Obrázek 1: Blokové schéma kompletní sestavy navrženého systému

2.1.2 Komunikace celého systému s vnějším světem

Logické signály

Vstupní

První zásadní logická informace bude od poplachové ústředny (EZS) a to například logický signál „poplach“, který může donutit naší centrálu například zatelefonovat na daná čísla, poslat SMS na další čísla, případně třeba rozsvítit všechna světla, vypustit z boudy dobrmana a mnoho dalšího. Druhým signálem z EZS bude signál „armed“ (zapnuto do stavu střežení). Tento signál nám zcela jednoznačně říká, že dům je prázdný a neměl by se v něm nikdo pohybovat, a tudíž mít jakékoli potřeby. Na základě tohoto signálu, můžeme například vypnout přívod vody do domu a tím zcela eliminovat možné vytopení při poruše. Další co by mohlo být na tomto signálu závislé, je například vytápění, které můžeme po dobu nepřítomnosti ztlumit. Můžeme zapnout systém vytváření iluze o tom, že v domě stále někdo je, tedy například rozsvěcení světel, či štěkot psa. Fantazii se jistě meze nekladou. Tyto signály bychom sice nemuseli mít v takovéto diskrétní podobě, neboť tak jako tak budeme řešit komunikaci s ústřednou, ale vzhledem k jejich důležitosti, se může vyplatit, využívat i přímé výstupy z ústředny, například poplach, než riskovat to, že při komunikaci cokoli selže. Ty mohou být použity takto zcela přímo a bez záludností. Jako ještě lepší varianta se nabízí, tyto přímé signály napojit na zcela autonomní systém GSM pageru, který nebude nikterak napojen do PC sítě, tedy jej nebude možno zvenku narušit a odstavit. Tato kombinace bude použita i v realizaci této práce. Dalším digitálním signálem může být signál světlo/tma, tedy signál, který eviduje, jaká světelná intenzita je v daný moment venku. Na základě tohoto signálu pak můžeme podmiňovat například rozsvěcení některých světel nebo například zavření dvířek kurníku. Dlužno dodat, že ke čtení digitálních vstupů, lze použit jak GPIO, přímo na desce centrály, tak jakýkoli podporovaný digitální I/O pin na jednotlivých periferních jednotkách. Jako další vstupní signály, které by bylo dobré sledovat a snímat co nejspolehlivěji se jeví například signály z požárního a záplavového čidla, které sice mohou být součástí stávající EZS, ale také nemusí. Další užitečný logický vstupní signál by mohl být signál HDO, zapínající vysoký, či nízký tarif spotřeby elektrické energie. V závislosti na něm pak můžeme zapínat, či vypínat některé spotřebiče.

Výstupní

Výstupní logické signály asi v drtivé většině případů ponesou význam: něco zapnout/ vypnout. Nejčastěji tedy lze očekávat, že budou ovládat například relé, stykače, spínaní zásuvky nebo u vzdáleného termostatu přímo zapínat, či vypínat topení na daném místě. Mimo dům takový signál může zapínat třeba zavlažování zahrady, nebo periodickou filtraci bazénu. Vzdálený Wi-Fi termostat pak může kontrolovat teplotu ve skleníku na druhém konci zahrady tedy zapínat topení, či naopak ventilaci.

Analogové signály

Analogový, tedy spojitý signál, bude v našem případě reprezentovat nějakou naměřenou hodnotu jakékoli veličiny a nebude skutečně spojitý, ale bude schopen rozlišit na určitém rozsahu třeba 1024 různých hodnot. Můžeme měřit vše co lze nějakým způsobem převést na elektrické napětí. Tedy hypoteticky by to mohla být intenzita světla, teplota a mnoho dalších veličin. V praktické realizaci našeho systému, bude prozatím analogový vstup použit asi jen pro měření stavu vybíjení záložní baterie, při výpadku

síťového napětí.

Složitější signály

Jako datové signály můžeme označit komplexnější informaci, ale v našem případě se bude vyskytovat většinou až na další vrstvě systému. Výjimkou asi bude přenos dat z ústředny EZS, nesoucí v sobě řadu komplexních informací. Dále z čidel komerčních meteo-stanic, který kromě naměřené teploty a někdy i vlhkosti, přenáší i identifikaci sítě a čidla, která slouží na straně přijímače k rozlišení mezi měřenými místy a kterých mohou být až desítky. Další výjimku bude tvořit elektroměr který komunikuje po RS485. Ze signálů, které nebudou v práci realizovány by to pak mohl být například signál dálkového IR ovladače.

2.1.3 Komunikace systému uvnitř, centrála – periferie

Sériová komunikace

USB převodníky

Část námi navržených periferních modulů, bude hardwarově připojena na sériové porty centrály, které budou reprezentovány USB převodníky. Poskytuje nám to určitý komfort, jelikož pomocí USB hubů, můžeme připojit mnoho takových zařízení, a pokud dojde k neopatrné manipulaci na vstupech a výstupech periferních modulů, je pravděpodobné, že zničíme jen levný modul a centrála bude zachována. Na standardní sériový port centrály bychom zpravidla stejně museli dávat převodníky napěťových úrovní, neboť Raspberry Pi má logickou úroveň 3.3 V a řada jiných zařízení, může používat logické signály s úrovní 5V. Nelze však vyloučit určitou nestabilitu USB převodníků, které mohou občas přestat pracovat. Tento problém by bylo možno řešit sledováním činnosti a případným odpojením napájecího napětí. V podstatě podobně, jako u systémů *watch-dog.*

2.2 Volba hardware a software

2.2.1 Volba hardware pro řídící jednotku

MCU

Jak již bylo dříve naznačeno, řídící jednotka, či jinými slovy centrála, je zařízení, od kterého požadujeme mnoho úkonů a bude tedy potřeba dostatečné výkonové rezervy. Hned na počátku byla vyřazena možnost realizace centrální jednotky na nějaké platformě založené na MCU. Tato varianta by byla pravděpodobně také realizovatelná, ale přineslo by to s sebou problémy, které řešit nechceme. Samozřejmě pro úkoly jednodušší by MCU mohla být dobrou volbou, ale v našem případě jsme tuto variantu zavrhli.

ARM jednodeskový minipočítač

Dalším možnou platformou pro realizaci centrální jednotky je jednodeskový minipočítač obvykle založený na procesorech ARM a běžící pod nějakou verzí operačního systému Linux. Existují i verze minipočítačů na které lze nainstalovat Windows. Těchto jednodeskových minipočítačů jsou dnes na trhu desítky, ne-li stovky. Nejznámější z nich je nepochybně platforma Raspberry Pi. Díky skvělému marketingu jí zná snad každý, komu je tento obor blízký. Ovšem na trhu je velké množství dalších podobných minipočítačů. V

Číně se vyrábí mnoho desek velmi podobných, zpravidla lépe vybavených a o dost levnějších. Podobnost je nápadná mnohdy i ve jméně, například Orange Pi, Banana Pi apod.

https://www.tvfreak.cz/raspberry-pi-orange-pi-banana-pi-ktery-jako-htpc/5533-4

Na počátku práce, byla tendence zvolit desku s co nejvíce vestavěnými periferiemi. Tyto požadavky obvykle splňují různé desky čínské provenience a jsou, vzhledem ke své hardwarové výbavě, relativně levné. Po prostudování mnoha materiálů a diskuzních fór, bylo však od tohoto plánu upuštěno. Dalo by se říci, že kromě platformy Raspberry, která je skutečně kultovní a stojí za ní velký tým vývojářů, je u dalších desek velký problém s podporou, která je malá, nebo skoro žádná. Na základě těchto zjištění bylo následně od volby různých zajímavých výrobků upuštěno a přednost v této práci byla dána stabilitě a dobré podpoře a jako platforma pro centrálu bylo zvoleno Raspberry Pi 3. Tato deska minipočítače je na rozdíl od některých dalších, výše zmíněných, vybavena relativně málo. Někteří konkurenční výrobci osazují na své desky periferie jako například: Wi-Fi, LTE modem, řadič SATA disků, a mnoho dalších. Vzhledem k relativně chudé výbavě Raspberry Pi3, bylo nutno volit některé periferie jako přídavné. V realizaci této práce to bude hlavně LTE modem. Ten bude užit pro odesílání varovných a stavových SMS, případně jako nouzové připojení k internetu, pokud vypadne standardní konektivita domu. Do budoucna je plánováno připojení externího USB SSD disku, na kterém bude systém, místo stávajícího umístění na SD-kartě. Případně jen části, tedy složky, do kterých se často zapisuje (například logy). Pro opakované zápisy není SD karta, vzhledem k jejich omezenému počtu, vhodným médiem. Další plánovanou periferií byl původně voice modem, který by umožňoval jakýsi interkom, tedy v případě poplachu dovolil dům odposlouchávat a také zpětně prostřednictvím reproduktorů a k nezvaným návštěvníkům promlouvat. Připojení tohoto modemu nebude v této práci realizováno. Desky minipočítačů také umožňují přímé připojení speciální kamery přímo k desce, které také nebude použito, neboť celý systém bude umístěn v rozvaděči na chodbě kde by kamera nebyla v nejpraktičtější pozici. GPIO piny přímo na desce Raspberry nebudou prozatím využity. To by se sice nabízelo, ale nese to s sebou rizika snadného poškození desky a bylo by nutno desku doplnit ve většině případů o převodník napěťové úrovně, neboť Raspberry má napájení jen 3,3V. Zdá se tedy jednodušší použít moduly, které byly pro tuto práci navrženy a umožňují připojení několika vstupů či výstupů, budou popsány později. Do budoucna bude pravděpodobně sériová komunikace vyřešena speciálním modulem na bázi kontroléru STM32.

Operační systém

Jak bylo výše zmíněno, existují i desky, na kterých lze spustit Windows, ale v našem případě, vzhledem k tomu, jaké aplikace budeme používat, byl Linux jasná volba. A opět pro co největší stabilitu a bezpečí, byla zvolena asi nejužívanější verze, tedy systém Raspbian. Jak již bylo výše zmíněno, operační systém bude nainstalován na SD kartě. Mechanika na SD kartu je standardní součástí desky Raspberry Pi. Přes jisté nevýhody, je to řešení nejjednodušší a pro účely této práce a testování, dostatečné. V reálném provozu pak bude SD karta doplněna, nebo nahrazena, USB SSD diskem. Pro sériová rozhraní budou využity USB porty, pro převodník na RS485 také. Modem LTE bude obsazovat další USB. Bude využit USB hub, abychom měli dostatek portů pro všechna zařízení.

Napájení

Řídící jednotka bude napájena ze stávajícího systému zálohovaného napájení

ostatních aktivních prvků sítě. Napájení je řešeno standardním ATX zdrojem, ze kterého se využívají napětí +5V a +12V, která jsou rozvedena k routerům, access pointům a switchům, tak aby při výpadku napájení fungovalo připojení k síti. Záloha je prozatím zajištěna standardní UPS Victron 750VA, která stávající zatížení udrží v provozu zhruba 7 hodin, v závislosti na stavu baterií. Do budoucna je plánováno zálohované napájení pro EZS a námi vytvořený systém nebo alespoň jeho centrální jednotku. Toto napájení by bylo řešeno záložní baterií s dostatečnou kapacitou a automatickým nabíjecím obvodem. Doposud má EZS svou autonomní baterii a kontrolu nabíjení a vybíjení. K monitorování stavu nabití baterie pak bude sloužit jeden z analogových vstupů a centrální jednotka bude schopna odesílat varovné maily či SMS při poklesu napětí baterie. Taktéž bude monitorováno napětí síťové, a jeho výpadky budou uživateli nějakým kanálem odesílány případně budou logovány. K monitorování síťového napětí bude pravděpodobně použito buď relé nebo optočlen s nějakým zdrojem malého napětí, aby bylo vše zcela odděleno od napájecí sítě. Pro účely této práce budou funkce měření napětí, či výpadku jen simulované, neboť praktická realizace bude vyžadovat zásah a připojení do rozvodné sítě.

2.2.2 Periferní jednotky

Volba periferních jednotek byla také velkou výzvou. Existuje skutečně obrovské množství možností. Můžeme si skutečně vybrat ze stovek komerčních výrobků produkovaných stovkami výrobců. Bylo nutno zvážit i tuto možnost. V komerčních produktech můžeme najít různé kategorie, počínaje levnými produkty pro domácnosti a konče průmyslovou automatizací. Přenosových protokolů je také nespočet a to jak pro přenos bezdrátový, tak po drátě. Jedna z prvních myšlenek byla, použít sběrnici RS485 a komerční I/O moduly. Cena by byla sice o dost vyšší, ale vzhledem k předpokládanému počtu požadovaných periferií by to nebylo nic dramatického. Nicméně znamenalo by to do různých částí domu instalovat nové kabelové rozvody a je obvykle poněkud problematické. Tedy další úvaha byla využití stávající sítě Ethernet, která již je v domě zavedena a sítě Wi-Fi. I v této oblasti existuje mnoho komerčních produktů, jak profesionálních tak různých levných hobby produktů, například čínské provenience. Další možností je samozřejmě vyrobit si veškeré periferie na míru. I v tomto případě je mnoho možností, jak takové periferie stvořit. Jako možnost první volby, se nabízí využití nějakých MCU, které jsou vybaveny software pro námi požadovanou funkcionalitu. Na trhu je velké množství produktů, které jsou schopny tuto roli zastat. Vzhledem k tomu, že funkce požadované od periferních subsystémů pro tuto práci, nebudou příliš sofistikované, dá se říci, že je zvládne téměř jakákoli MCU. Nejoblíbenějšími MCU mezi „kutily“ jsou určitě PIC výrobce Microchip, AVR produkt firmy Atmel, nebo STM32 vyroben STMicroelectronics. Firma Atmel byla firmou Microchip koupena, takže dnes jsou tedy jak PIC tak AVR produkty firmy Microchip. Jako další varianta se nabízí použít čím dál tím oblíbenější čínský modul MCU ESP8266, s integrovanou Wi-Fi, který za cenu dolaru nabízí celkem zajímavé parametry.

Vlastní jednotky

Po zvážení všech kladů a záporů, byla zvolena kombinace jak komerčních tak vlastních periferních prvků. Jako platforma pro vlastní prvky bylo zvolena deska Arduino NANO s procesorem ATmega328P. Volba tohoto modulu MCU byla dána hlavně dostupností, cenou a velikou komunitní podporou projektu Arduino. Výhodou je také to, že tyto moduly jsou vybaveny headery pro vývoj aplikací a také pro zapojení stohovatelných přídavných modulů - *shieldů*, v práci bude například využit Ethernet *shield* – tedy přídavný modul Ethernet.

Obrázek 2: Arduino Nano

Technické parametry:

Mikrokontrolér Atmel ATmega328

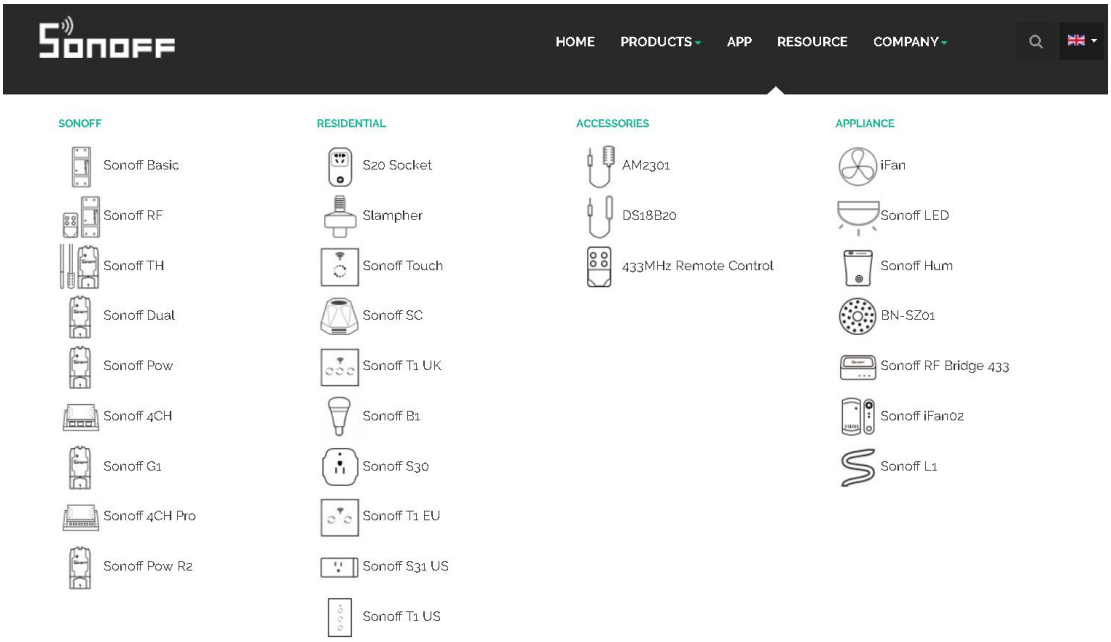
•Pracovní napětí 5V
•Maximální napájecí napětí 6 až 20V
•Digitální I/O piny 14 (ze kterých 6 poskytuje PWM výstup)
•DC proud na I/O pinu 40 mA
•DC proud pro 3,3V pin 50 mA
•Flash paměť 16KB ze kterých jsou 2KB použity pro Boot loader
•SRAM 1 KB, EEPROM 512bytů
•Taktovací frekvence 16MHz
•USB Konektor typ Mini B samec

https://cz.farnell.com/arduino-org/a000005/arduino-nano-evaluation-board/dp/1848691

Upravená tovární zařízení - jednotky Sonoff

Jako kompromis mezi továrním výrobkem a vlastní tvorbou bude využit modul firmy Sonoff, který vlastně obsahuje jen MCU ESP 8266, paměť, napájecí zdroj a spínací relé.

Výrobky asijského výrobce Sonoff se prodávají běžně v prodejní síti po celém světě, Českou republiku nevyjímaje. V práci bude použita jednotka *Sonoff TH-10* – tedy termostat 10A, ale stejně tak je možno použít jakékoli jiné jednotky v podstatě i se stejným software. Tedy například pro spínání spotřebičů *Sonoff Basic.* I tento modul lze dálkově ovládat jen jako prostý spínač. Sonoff vyrábí i samostatné spínače s jedním i několika vstupy a jedním i několika výstupy. Volba tohoto modulu byla motivována tím, že je celkem robustního provedení, existuje k němu vodotěsná krabice a je běžně prodáván na českém trhu, tedy by měl být dostatečně bezpečný na to, abychom jej mohli připojit k elektrické síti a spínat jím elektrické vytápění. Jelikož výrobce má v těchto zařízeních software pro naše účely nevhodný, bude nahrazen softwarem jiným, který umožní námi požadovanou funkcionalitu. Od původního záměru vytvořit vlastní software, bylo upuštěno, neboť byl nalezen kvalitní, svobodný software třetí strany, vhodný pro použití v našem systému.



Obrázek 3: Nabízené produkty Sonoff

httpsd.cc/en/products/sonoff/sonoff-th://sonoff.itea

Obrázek 4: Sonoff TH Wi-Fi termostat

https://sonoff.itead.cc/en/products/sonoff/sonoff-th

Specifikace jednotky Sonoff TH:
Specification
Power Supply: 90V~250V AC
Max. Current: 10A /16A
Wireless Standard: WiFi 2.4GHz b/g/n
Security Mechanism: WEP/WPA-PSK/WPA2-PSK
Operating Temp.: 0℃~40℃
Operating Humidity: 5%-95%
Material: FR-ABS
Connector: universal

https://sonoff.itead.cc/en/products/sonoff/sonoff-th

Tovární zařízení do kterých nebude nijak zasahováno

Ústředna EZS

Hlavním továrním zařízením spojeným s naším systémem, je ústředna elektronického zabezpečovacího systému Paradox. Konkrétní ústředna použitá v naší práci je Spectra Magelan 50. Pokusíme se navázat s ní komunikaci prostřednictvím její sběrnice pro přídavné moduly, abychom z ní mohli získávat požadované údaje, případně jí posílat příkazy. Bude tedy nutno dešifrovat provoz na této sběrnici a následně zpracovat. Data na sběrnici, která ústředna poskytuje nám umožní monitorovat veškerý provoz, případně ukládat pro další použití. Pro naši práci jsou nejdůležitější data uživatelská, tedy ve kterých zónách dochází k pohybu, kde byl spuštěn poplach, narušen tamper, kdy vypadlo napájení a podobně. Po sběrnici budou ústřednou posílána i data charakteru servisního, jako je nastavení jednotlivých zón, a jiných parametrů. Tato data jsou však již určena pro programování ústředny a výrobce k tomuto účelu poskytuje proprietární software, tedy určitě nebude naší snahou tato data nějak využívat, natož modifikovat. Snad jediné, co se pokusíme zprovoznit, bude vzdálené zapnutí „zaarmování“ ústředny. Pokud by byl problém vnutit ústředně příkaz, pak tuto funkci vyřešíme použitím PGM vstupu ústředny, který propojíme s logickým výstupem některé z periferních jednotek.

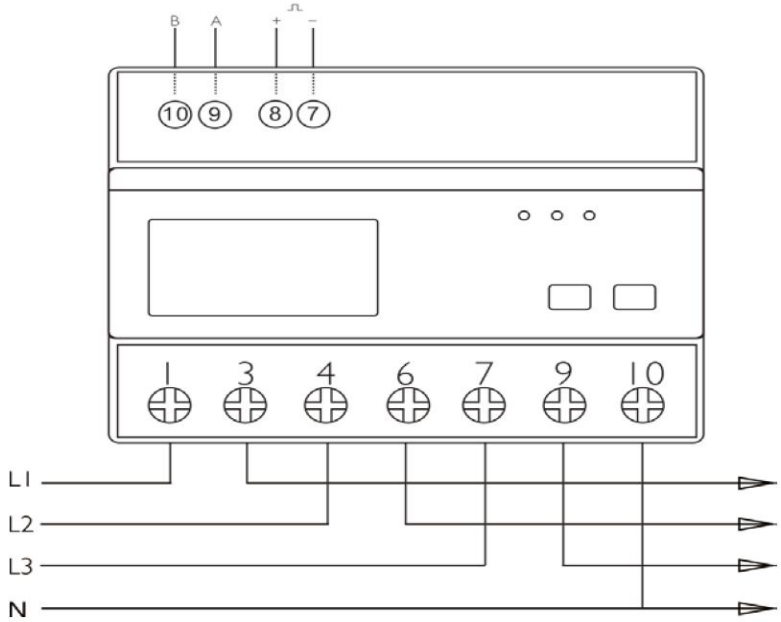
Technické údaje ústředny Paradox Spectra Magelan 5050:

Max. proudový odběr z AUX výstupu	1 A	**Sběrnicové detektory**	ne	**Interval dohledu**	80 minut/24 hodin
Dělení na podsystémy	2	**Definice závislé / intelli zóny**	ano	**Max. počet bezdrátových detektorů**	32
Přímé spojení s PC přes interface I307	ano	**Počet podsystémů**	2	**Typy bezdrátových detektorů**	všechny typy bezdrátových detektorů řady K
Max. počet zón v systému	32	**Vytvoření závislého podsystému**	ne	**Opakovač pro prodloužení dosahu**	ano, RPT1
Maximální počet klávesnic v systému	15	**Společné zóny pro oba podsystémy**	ano	**Max. počet opakovačů signálu**	2
Historie událostí	256	**Master kód**	1 + 2	**Bezdrátový PGM výstup**	ano, typ 2WPGM
Napájení	16 V~, 20/40 VA	**Kód údržby**	1	**Max. počet bezdrátových PGM**	16
Typ zdroje	spínaný	**Instalační kód**	1	**Aktivace PGM výstupu klíčenkou**	ano, libovolný PGM v systému
Typ AUX výstupu	elektronická vratná pojistka 1,1 A	**Délka uživatelského kódu**	4 nebo 6 místný	**Max. počet bezdrátových klávesnic**	8
Max. proudový odběr z výstupu BELL	2 A	**Automatické zapnutí**	ano, podle času, klidu v systému	**Počet zón na expandérech (APR-ZX8)**	24
Typ BELL výstupu	elektronická vratná pojistka 3A	**Typy zapnutí**	úplné, FORCE, STAY, SLEEP	**Počet vstupů na desce ústředny**	5
Proudový odběr ústředny	100 mA	**Zapnutí StayD**	ano	**Změna firmware**	ano, pomocí WinLoad
Maximální délka sběrnice /součet/	230m	**Počet telefonních čísel na PCO**	2+1 záložní	**Zobrazení historie událostí**	software WinLoad, na klávesnici K32LCD CZ
Maximální délka sběrnice k modulu	76m	**Počet tel. čísel na občanský telefon**	5	**Optická signalizace**	LED dioda CHARGE, STATUS
Firmware	uložen v EEPROM paměti	**Pracovní teplota**	-10 až 50°C	**Doporučený typ transformátoru**	trafo kryté 40 VA
Detekce telefonní linky	ano	**Národní bezpečnostní úřad**	stupeň utajení „D“ (2)	**Doporučený typ boxu**	BOX M-40, BOX S-40, BOX VZ-40
Dobíjecí proud záložního akumulátoru	350/700 mA	**TestAlarm**	2. nízké až střední	**Max. počet zón na desce ústředny**	10
Doporučený záložní akumulátor	12 V, 7 Ah/18 Ah	**Bezdrátové ovládání klíčenkou**	ano	**Typy naprogramovaných zón**	22
Hardwarový reset	ano, tlačítko reset	**Počet klíčenek v systému**	32	**Max. počet keyswitch vstupů**	5, pouze vstupy na ústředně
Max. počet zón na klávesnicích	15 (jedna klávesnice - jedna zóna)	**Typ klíčenek**	REM1, REM 15, REM2, REM3	**Max. počet PGM výstupů v systému**	16
Zónový expandér	ano, 8 zón jeden expandér	**Bezdrátové detektory**	ano	**Uživatelské kódy**	30
Max. počet zónových expandérů	3	**Dohled nad bezdrátovými detektory**	ano	**Software**	WinLoad
Bezdrátové klávesnice	ano, K32RF a K37	**Bezdrátové zóny**	ano, 32	**Klávesnice**	K32LCD, K32, K35, K37, K636
Počet vstupů (zón) na desce ústředny	5 (ATZ 10)	**Bezdrátové PGM výstupy**	ano, 16	**PGM výstupy na ústředně**	4x opto-relé 50 mA polarita +/-

https://www.stasanet.cz/Paradox-a-ostatni-EZS/Systemy-SPECTRA-MAGELLAN/Ustredny-Magellan/MG-5050-ustredna-EZS-2x5-10-zony-ATZ-max-32-zon-4xPGM.html

Elektroměr digitální

Dalším továrním zařízením do kterého nebude nikterak zasahováno, bude třífázový elektroměr. Z něj se pokusíme získat jím poskytované údaje, které prozatím nebudou využity nijak, ale v budoucnu by posloužily k analýze průběhu spotřeby, vyvážení zatížení jednotlivých fází a případnému lepšímu „vyladění“ spotřeby efektivnějším využitím a následně například snížení hodnoty hlavního jističe. Elektroměr komunikuje po sběrnici RS485. Komunikačním protokolem je protokol *Modbus*, který se běžně využívá v průmyslové automatizaci. Jednotlivé hodnoty elektroměru jsou uloženy v adresovatelných registrech. Jejich hodnoty získáme tak, že zařízení zašleme dotaz s adresou požadovaného registru a je nám vrácena hodnota, kterou registr obsahuje.



Obrázek 5: Zapojení elektroměru:

http://www.eastrongroup.com/data/uploads/SDM530C_Three_Phase_Multi-function_Remote_Control_Direct_Connect_Energy_Meter_with_Built-in_Relay.pdf

The format for each byte in RTU mode is:	
Coding System:	8-bit per byte
Data Format:	4 bytes (2 registers) per parameter.
	Floating point format (to IEEE 754)
	Most significant register first (Default). The default may be chang
	required -See Holding Register "Register Order" parameter.
Error Check Field:	2 byte Cyclical Redundancy Check (CRC)
Framing:	1 start bit
	8 data bits, least significant bit sent first
	1 bit for even/odd parity (or no parity)
	1 stop bit if parity is used; 1 or 2 bits if no parity

Obrázek 6: Komunikační protokol elektroměru

Technické údaje v data-sheetu na odkazu výše.

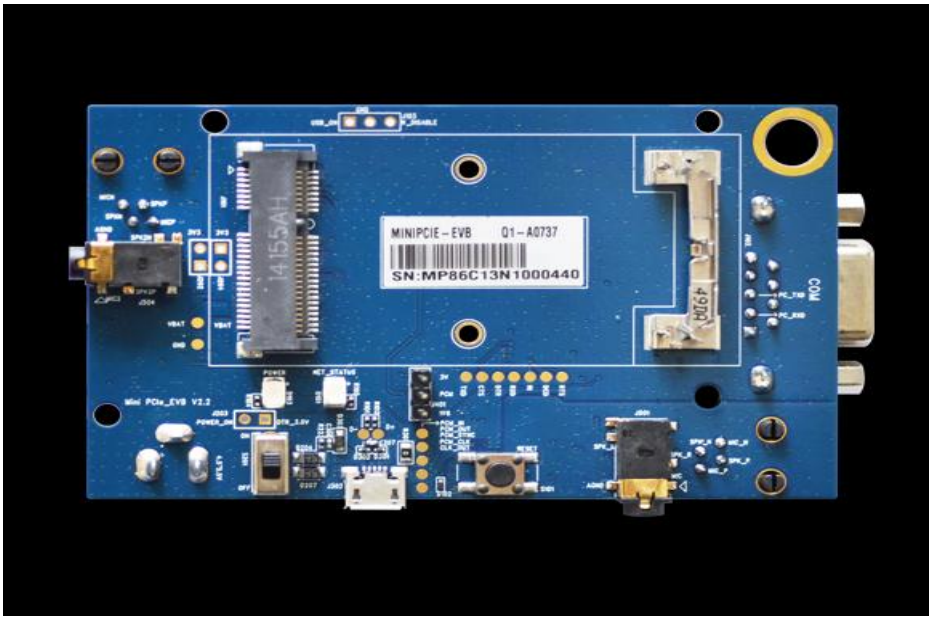
http://www.eastrongroup.com/data/uploads/Eastron_SDM630MV_CT_protocol_V1_0_.pdf

Kompletní popis registrů a komunikace je ve výše uvedeném dokumentu.

LTE modem

Také LTE modem je tovární výrobek a nebude nijak upravován. Bude použit pro posílání SMS v případě významnějších varování a bude sloužit jako záloha internetového připojení při výpadku standardního připojení domu. Jako modem je možno použít téměř jakýkoli produkt, který bude v operačním systému fungovat. Pro komunikaci bude využito systémových služeb.

Obrázek 7: Použitý modem quectel
https://www.quectel.com/product/ec21minipcie.htm



Obrázek 8: USB/Serial redukce k modemu

https://www.quectel.com/product/minievb.htm

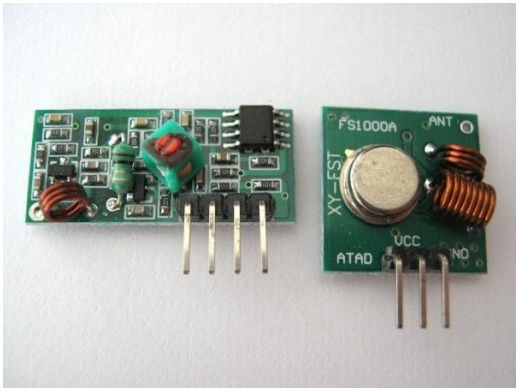
(U)SIM Interface	USIM/SIM card (6 pins) connector with push loading USIM/SIM card: 3V and 1.8V
Audio Interface	Used for earphone and handset
UART Interface	COM-serial interface for data communication (default 115200bps) Max. baud rate: 460800bps
USB Interface	USB 2.0
Signal Indication	2 LEDs are available for signal indication
Physical Characteristics	Size: 94mm × 58mm

Obrázek 9: Technické údaje modemu

https://www.quectel.com/product/minievb.htm

Čidla pro měření teploty

Abychom mohli měřit teplotu na různých, případně i odlehlých místech, bylo původním záměrem vytvořit k tomuto účelu vlastní hardware. Bylo to však již mimo časové možnosti, tedy alternativní řešení bylo objeveno ve využití komerčních bezdrátových teploměrů, v cenách kolem 200 Kč, ze kterých použijeme jen měřící a vysílací část. Tyto vysílače pak můžeme rozmístit kdekoli v dosahu signálu, jsou robustního venkovního provedení a vydrží na bateriové napájení v provozu několik měsíců i venku a v mrazech. Pro nás pak zbývá jen jejich signály zachytit, patřičně dekódovat a dle potřeby využít. Jako příklad využití se nabízí hlídání mrazu ve skleníku, pokud nemá elektrickou přípojku a nelze tedy realizovat ochranu termostatem a vytápěním. Pokud se teplota začne blížit bodu mrazu, systém to vyhodnotí a zašle např SMS „Mráz ve skleníku“. Uživatel pak může uprostřed noci vyběhnout do skleníku vybaven např. petrolejovou lampou, aby zachránil úrodu rajčat.



Obrázek 10: Přijímač vysílač 433MHz

https://www.santy.cz/moduly-c22/modul-433mhz-arduino-sada-i63/

Specifikace:

Přijímač:
- Model: MX-05V
- Volty: DC 5V
- Ampery: 4mA
- Frekvence: 433.92MHZ
- Citlivost: -105DB
- Velikost: 30*14*7mm

Vysílač:
- Model:MX-FS-03V
- Dosah: 20-200 Metrů (záleží na napětí)
- Volty: 3.5-12V
- Velikost : 19x19mm
- Pracovní mód: AM
- Rychlost : 4KB/S
- Vysílací výkon: 10mW
- Frekvence: 433.92MHZ

https://www.santy.cz/moduly-c22/modul-433mhz-arduino-sada-i63/

V práci je použit jen přijímač, jako vysílače slouží vnější jednotky komerčně vyráběných teploměrů. Ty jsou dostupné za velmi nízkou cenu a lze koupit jen samotné vnější jednotky. Pro účely práce, byla pořízena venkovní jednotka WT450.

Obrázek 11: Venkovní jednotka WT450

https://www.hodinarstvi.cz/produkty/meteostanice/nahradni-senzory/bezdratove-cidlo-k-modelu-garni-2100-wt450.html

Technická data přijímače 433 Mhz:

Modul vysílače + přijímače na frekvenci 433 MHz

Typ přijímače XY-MK-5V

Typ vysílače XY-FST

Dosah 20 - 200 m

Režim modulace ASK /OOK

Šířka pásma 2 MHz

Vysílací výkon 25 mW (315 MHz, 12 V)

Přenosová rychlost: 10 Kbps (teoretická hodnota)

Provozní proud ≤ 5.5 mA (5.0 V)

Vysílací rychlost < 9.6 Kb / s (315 MHz -95 dBm)

Možnost přelazení 315 - 433.92 MHz

https://arduino-shop.cz/arduino/1003-arduino-433mhz-vysilac-prijimac.html

Převodníky

Nedílnou součástí celého projektu budou i různé převodníky. Pokud bychom propojovali zařízení s různými logickými úrovněmi, typicky se to stává mezi logikou 3,3 a 5V, je zpravidla nutno konvertovat napěťové úrovně logických signálů. K tomu slouží různé monolitické převodníky úrovní. Tedy na tento problém bychom například narazili pokud bychom chtěli propojovat například GPIO na Raspberry Pi (3,3V) a GPIO na modulech Arduino NANO (5V). Toto ale není plánováno v naší práci použít, tedy pravděpodobně nebudeme tyto převodníky potřebovat. Další velmi často potřebné převodníky v podobných zapojeních, jsou různé převodníky pro sériovou komunikaci, typicky USB → RS232, RS422, RS485, ale také například Serial → Ethernet a naopak, paralelní, CAN BUS, I2C a další. V této práci budeme potřebovat převodníky z USB na RS232, tedy sériový port, s 5V logikou, abychom mohli komunikovat s ústřednou Paradox, a převodník na průmyslový standard RS485, pro komunikaci s digitálním elektroměrem. Samozřejmě při rozšiřování systémů se mohou objevit potřeby mnohých dalších. Pokud bychom například chtěli využívat dostupných modulů pro RS485 v odlehlé části domu, lze tento protokol poslat po běžné síti Ethernet a na požadovaném místě pak zkonvertovat.

Obrázek 12: Převodník USB <-> RS485

https://www.arduiner.com/en/usb-to-ttl-rs232-rs485/8461-cp2102-usbttlrs485rs232-interconversion-6-in-1-serial-converter-uart-module-3809200642792.html

Obrázek 13: Převodník USB <-> RS232

https://www.ptshop.cz/Prevodnik-USB-na-TTL-RS232-PL2303-d143.htm

2.2.3 Další nutná zařízení k provozu, která nejsou součástí práce

K tomu aby vše fungovalo, je nutno použít i stávající infrastrukturu instalovanou v domácnosti či domu. Jedná se připojení do internetu, tedy router, v našem případě Mikrotik RB133. Ten bude zprostředkovávat jak připojení systému k internetu, například pro zasílání mailů, tak musí pomocí port forwardingu, zajistit i komunikaci „dovnitř“ tedy přístup na web-server na Raspberry Pi z vnějšího světa. Je tedy potřeba mít pevnou IP adresu. Dalším prvkem nutným pro fungování Wi-Fi modulů je stávající access point, v našem případě opět Mikrotik RB133, který musí umožnit připojení těmto modulům. Tedy například je potřeba povolit v MAC filtrování MAC adresy všech modulů. V neposlední řadě bude systém využívat metalickou síť Ethernet, tedy přepínače, kabely a zásuvky.

2.3 Vlastní realizace

2.3.1 Návrh finálního řešení

Rozdělení druhu signálů

Logické – digitální signály

Aby bylo možno postupovat v návrhu systematicky, bylo potřeba stanovit určité obecné schéma vlastností systému. Postupováno bylo od požadavků na vnější funkcionalitu až k centrální jednotce. Obecně bychom si mohli rozdělit signály na digitální či logické, tedy požadavek například zapnuto/vypnuto, reprezentováno logicky jako 0/1 a na periferních prvcích pak přepnutím GPIO do stavu LOW/HIGH. Je potřeba si uvědomit, že signály budeme někdy potřebovat v aserci, jindy v negaci, v závislosti na tom, co budou fyzicky spínat, nicméně to je jen drobná úprava software, případně lze zahrnout možnost negování i do uživatelského interface. Obě varianty mají své výhody i nevýhody a záleží na zdatnosti uživatele, jaká varianta pro něj bude přijatelnější.

Analogové signály

U analogových signálů bude vždy přenášena nějaká hodnota. Bez ohledu na to, jak bude přenášena, bude se vždy jednat o nějaké číslo. U standardních analogových vstupů, realizovaných na vstupech našich periferií postavených na Arduinu NANO, to bude hodnota 0-1024. Samozřejmě bychom tuto hodnotu mohli již přímo na periferní jednotce přepočítat například na volty, v případě měření napětí. Totéž lze udělat na centrále, rozsah měření by mohl být i součástí nastavení v uživatelském interface. Nicméně jako ukázka funkcionality je víceméně jedno, jaký formát přenášené hodnoty zvolíme, proto bude ponechán v surové podobě.

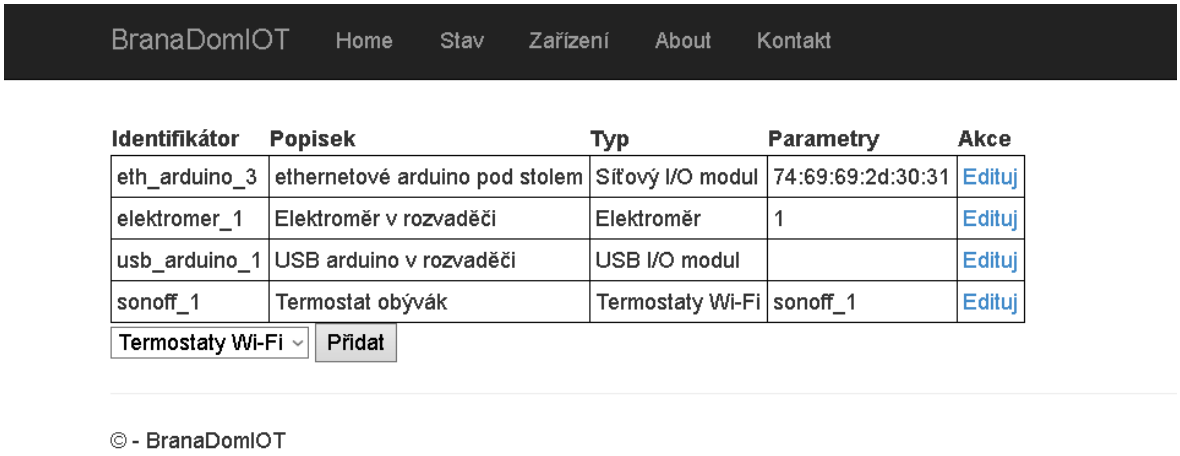
Komplexní data

Ta se budou často vyskytovat jako meziprodukt, tedy například při přenosu teploty z čidel na 433MHz nebo z termostatů ale ve finále to bude jen analogová veličina, tedy číslo. Stejně tak data z elektroměru se stanou jen čísly. Taktéž komunikace s ústřednou EZS bude na konci zpracování jen směsice logických a analogových hodnot. Komplexní data jako informaci budeme přenášet například modemem, ať již formou SMS, nebo jako záložní linky. Vzhledem k tomu, že modem je standardní zařízení, postaráme se jen o jeho určitou obsluhu a zbytek zajistí služby operačního systému. Další komplexní informaci by mohl přenášet systém přenosu IR ovládání, ale ten nebyl do naší práce zahrnut.

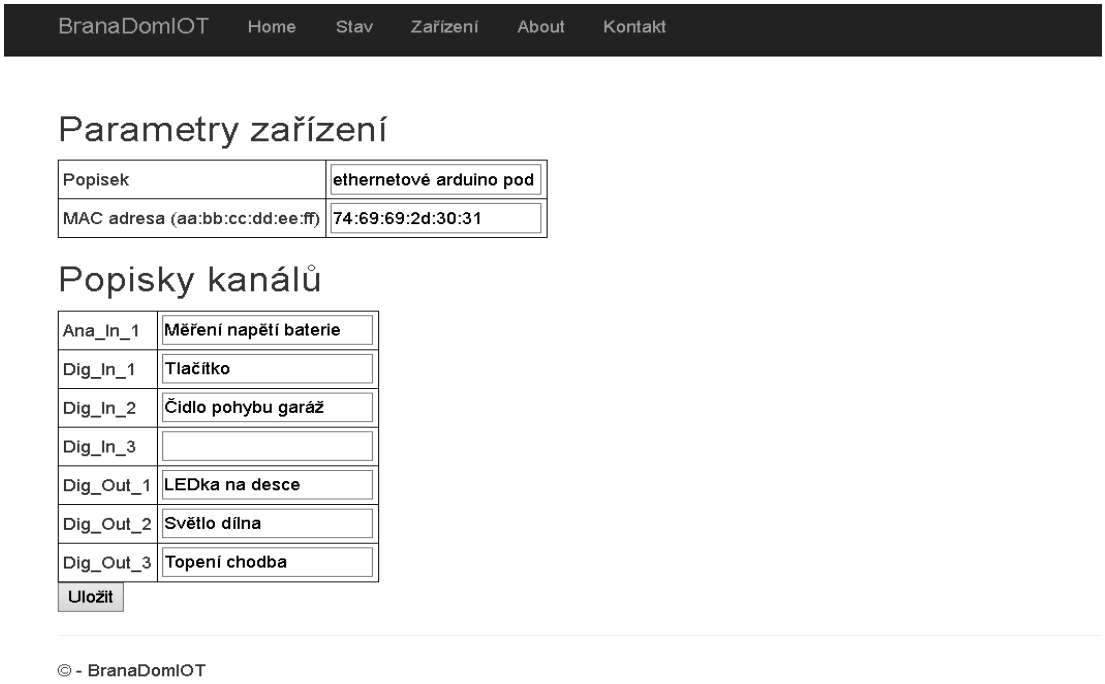
Architektura systému

Centrální jednotka

Centrální jednotka je postavená na platformě Raspberry Pi a systému Raspbian. Systém automatizace je pak navržen v několika základních modulech. Příchozí data z periferních jednotek, která přicházejí po Ethernetu, tedy i Wi-Fi přes stávající access point, a skrze různé porty, prostřednictvím USB převodníků, jsou zachytávána patřičnými drivery běžícími na Raspbianu na pozadí. Tyto drivery se starají o obousměrnou komunikaci, tedy nejen naslouchání příchozím datům, ale i odesílání povelů do periferních jednotek. Na druhou stranu pak zajišťují obousměrnou konverzi z a do protokolu MQTT, který byl zvolen pro vnitřní komunikaci mezi drivery a řídícím software a také ke komunikaci s termostaty Sonoff připojenými přes Wi-Fi. Z pohledu možné modifikace a rozšiřování, jeví se toto rozvrstvení jako optimální. MQTT zprávy obsluhuje MQTT server Mosquitto, běžící na centrální jednotce. Všechny MQTT zprávy pak zachytává MQTT logger, který je loguje do JSON souboru, reprezentujícího aktuální stav celého systému. Zde může nastat určitý problém, neboť zprávy přicházejí někdy velmi často a logování je celkem náročné, obzvláště na SD kartu, na které je Raspbian uložen,. Proto budou soubory s častým zápisem, nesoucí informaci o aktuálním stavu, uloženy v adresáři na RAM disku. Toto logování zajišťuje *mqtt logger* a informace o stavu je ukládána do souboru stav_systemu.json. Tento soubor je pak čten na straně webového rozhraní a reprezentován v uživatelském interface. Na centrální jednotce také poběží webový server, který bude prozatím zajišťovat jediný uživatelský interface. Mohl by být do budoucna například doplněn o aplikaci v mobilním telefonu. Webový server je postaven na Pythonu, frameworku *Flask* a šablonovacím nástroji pythonu *Jinja.* Krom úvodní stránky a několika stránek informativních, které jsou zde spíše pro ilustraci, server obsahuje stránky pro nastavení, tedy přidávání, editaci a mazání jednotlivých podporovaných periférií. Takže pokud například přidáme termostat Sonoff do další místnosti, jednoduše jej do stávajícího systému začleníme pomocí přidání nového zařízení. V sekci editace se v podstatě dostaneme ke stejným atributům již založeného zařízení a můžeme je měnit. Zde jsou to převážně popisky vstupů a výstupů. Tedy každému kanálu přiřadíme lidsky srozumitelný popis viz obrázky níže.

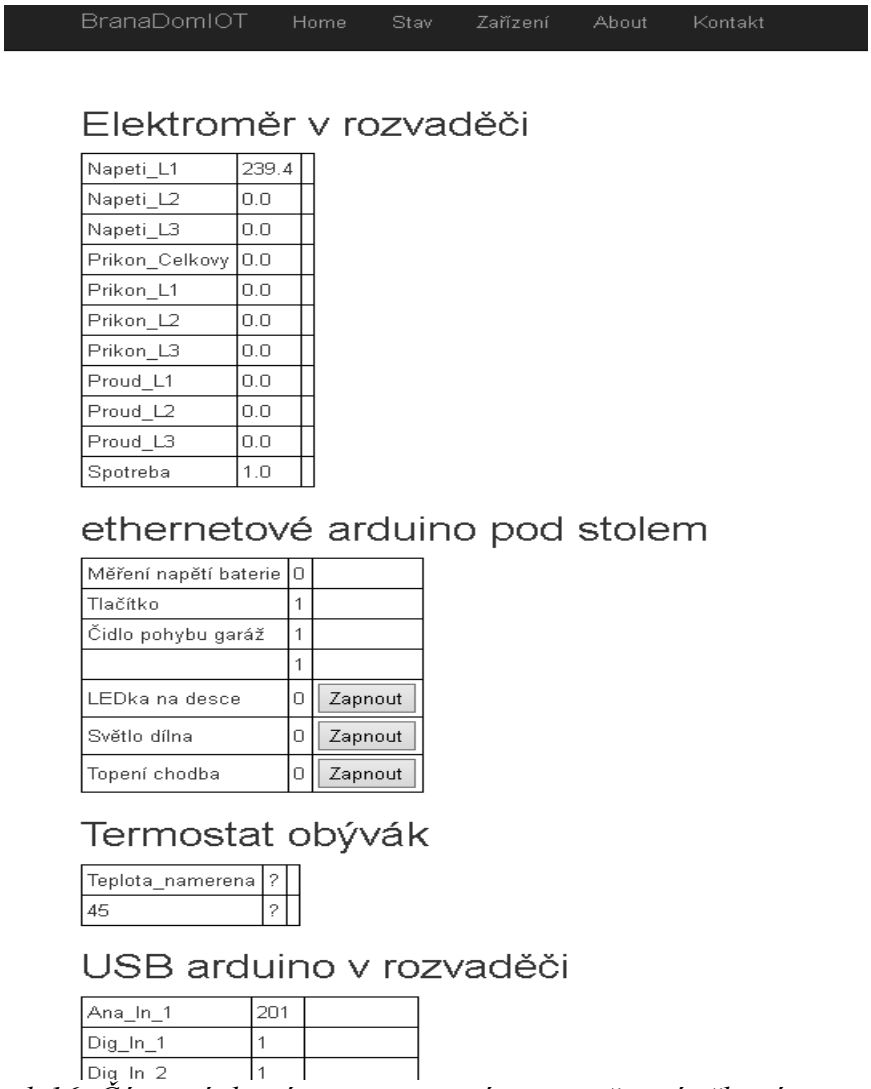


Obrázek 14: Stránka zařízení, s možností přidání a editace existujících



Obrázek 15: Stránka pro nastavení jednoho zařízení

V sekci stav pak můžeme uživatel monitorovat stav jednotlivých modulů a jejich I/O kanálů a některé lze přímo „natvrdo“ zapnout a vypnout. Zapnutí a vypnutí bude také podmíněno složitější rozhodovací logikou, která bude popsána dále.



Obrázek 16: Část stránky výpisu stavu systému, s možností některé logické výstupy přímo ovládat

Periferní jednotky

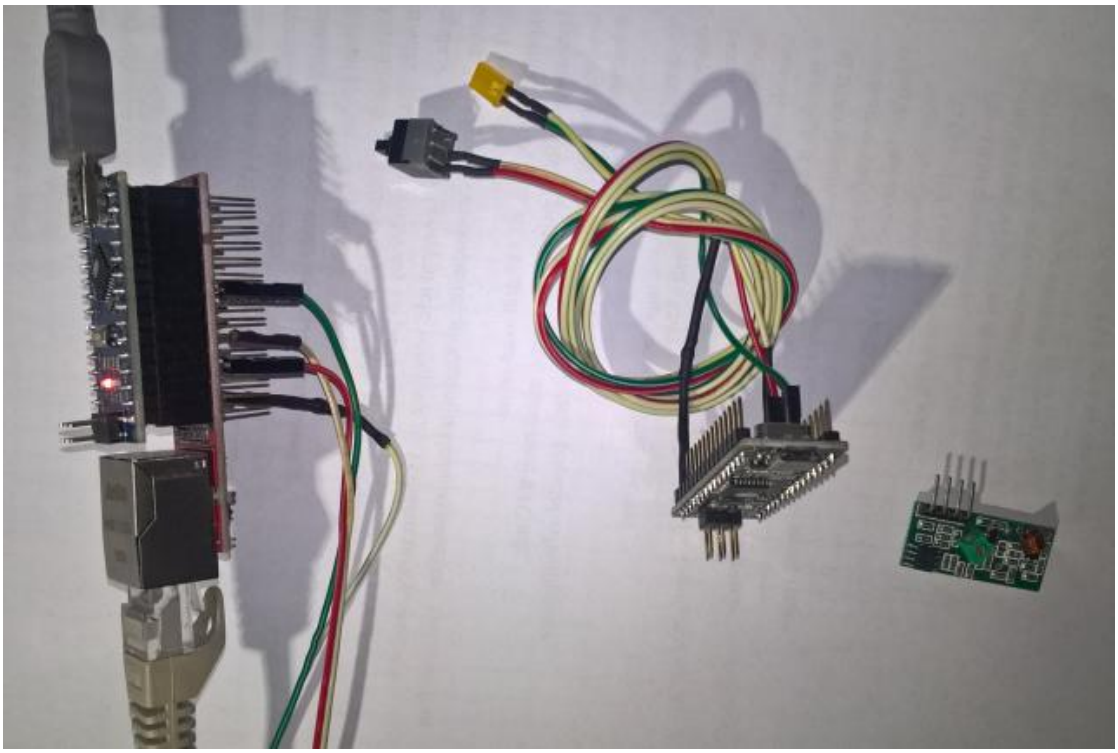
Vlastní jednotky na platformě Arduino NANO

Byly navrženy a realizovány dvě jednotky. Jedna pro sériové rozhraní, a jedna připojitelná kabelem RJ45 na síť Ethernet.

Jednotka Arduino sériová

Tato jednotka je připojena k centrále prostřednictvím USB portu. Deska Arduino NANO má na sobě obousměrný převodník USB → serial. Tato jednotka nám supluje přímé použití GPIO na Raspberry Pi, kterému jsme se z několika důvodů chtěli vyhnout a bude tedy obsluhovat zařízení poblíž centrály, která je umístěna vedle domovního rozvaděče a ústředny EZS, tedy v srdci ovládání celého domu. Na tuto jednotku je také připojen přijímač 433MHz pro teploměry, který by hypoteticky měl obsloužit všechny teploměry v dosahu. Další možností je pomocí výstupů této jednotky spínat například relé v rozvaděči a naopak pomocí jejích vstupů číst stavové signály z rozvaděče, například

HDO vysoký/nízký tarif nebo číst stav ústředny EZS - „armed“ a „poplach“, přímo na PGM výstupech, či pomocí PGM vstupu ústředny vzdáleně zapnout střežení. Software pro tuto jednotku byl napsán v jazyce C a k programování byl použit komunitní software Arduino.



Obrázek 17: Praktická realizace síťové a sériové jednotky s přijímačem 433 MHz(vpravo)

Jednotka Arduino síťová

Tento subsystém využívá stávajícího metalického rozvodu sítě Ethernet IEEE 802.3, tedy kabelové rozvody přepínače a zásuvky. Díky stávajícímu rozvodu tedy může být umístěn kdekoli v nemovitosti, nemaje požadavky na další kabeláž. Napájení může být externí 5V nebo může být použit injektor a splitter PoE. Modul má mikro USB konektor, k jeho napájení tedy postačí jakákoli usb nabíječka, například od mobilního telefonu, kterými je svět zaplaven. Tento modul je postaven opět na platformě Arduino NANO. K síťové komunikaci je využit násuvný modul, shield, se síťovým rozhraním Ethernet. Zvolený síťový modul je nejlevnější na trhu, a mnoho toho neumí, ale pro naše potřeby by měl být dostatečný. Komunikace bude probíhat na linkové vrstvě protokolového zásobníku. Moduly odesílají zprávy o svém stavu jako broadcast. Zpětná komunikace z centrály je pak odesílána na MAC adresy jednotlivých modulů. MAC adresa je každému modulu dána v době ukládání firmware, a je tedy potřeba ji u každého modulu změnit před nahráním firmware ve zdrojovém kódu. Bylo by možno tuto adresu a náhodně generovat, ale vzhledem k malému rozsahu modulů, by to bylo spíše kontraproduktivní. Bohužel procesor v sobě nemá žádné unikátní číslo, ze kterého by se dala MAC adresa generovat, ale šlo by například využít po prvním spuštění modulu naměření šumu na analogových vstupech s trochou matematiky.

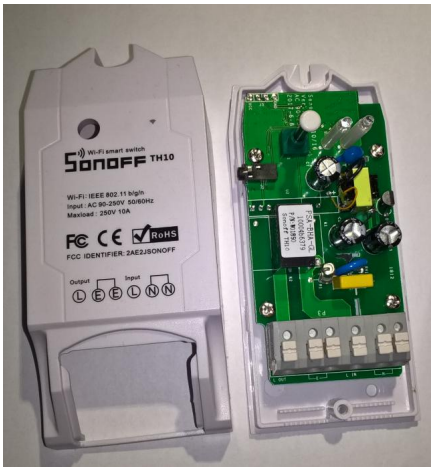
Wi-Fi jednotky Sonoff

V naší práci je použit modul TH-10 vybavený novým firmware. Instalovaný firmware, měl být původně nahrazen naším software. Při hledání informací však bylo zjištěno, že velice propracovaný svobodný software již byl vytvořen a bylo by ztrátou času, pokoušet se vytvořit alespoň stejně dobrý. Byl tedy použit software Tasmota, kterým byl přehrán původní firmware výrobce Sonoff. Tento software je možno stáhnout v mnoha verzích z těchto stránek:

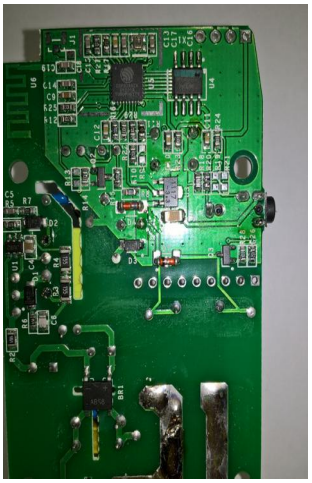
https://github.com/arendst/Sonoff-Tasmota

Návod jak hardwarově upravit modul k nahrávání najdeme např. Zde:

https://blog.vyoralek.cz/iot/sonoff-produkty-nahrani-firmware-tasmota/



Obrázek 19: Takto vypadá otevřený Sonoff TH



Obrázek 18: Druhá strana

Obrázek 20: Sonoff s doplněnými headery

Obrázek 21: Pohled z druhé strany

Sonoff vybavený software Tasmota nám umožňuje komunikaci přímo s modulem využitím web-serveru na tomto modulu, který použijeme jen ke konfiguraci správného čidla, IP adresa MQTT serveru a vhodného MQTT prefixu, abychom odlišili jednotlivé kusy navzájem. Vše ostatní již naše aplikace nevyužije, neboť jakmile toto nastavíme, budeme se s modulem spojovat již jen pomocí MQTT serveru. Z něj již jsou zprávy zpracovány tak, jako z ostatních modulů po průchodu jejich drivery.

Elektroměr

Elektroměr je připojen přímo k centrále, pomocí převodníku USB → RS485 a obsluhován driverem na straně centrály, který přijímá a loguje jeho stavy. Komunikace probíhá protokolem Modbus. Tyto údaje pak mohou být dále využity, jak bylo zmíněno výše. V práci šlo o to naučit se s elektroměrem komunikovat. Ke komunikaci byl použit levný čínský modul, který bychom asi v reálném provozu nahradili nějakým kvalitnějším.

EZS ústředna Paradox Spectra

Komunikace s touto ústřednou je zajištěna pomocí sériové linky. Osciloskopem bylo změřeno jaké napěťové úrovně se na sběrnici vyskytují. Zjištěná úroveň napětí byla 5V. Průběh připomínal sériovou komunikaci tedy byla odhadnuta komunikační rychlost a k velkému překvapení bylo po připojení převodníku k PC zjištěno, že sériová data ústředny jsou vysílána skutečně jako naprosto obyčejný sériový protokol, bez jakéhokoli šifrování. Když byl učiněn pokus nalézt, jakými příkazy by mohla být ústředna ovládána, zcela náhodně byl objeven software, který dokáže data ústředny přijímat i vysílat a dále je obousměrně překódovat do a z MQTT příkazů. Tedy alespoň by tak fungovat měl. Pokusíme se jej tedy částečně využít a zprovoznit obousměrnou komunikaci.

2.3.2 Obslužný software systému

Centrální jednotka

Software je na centrální jednotce vytvořen v jazyku Python. Je rozdělen do několika modulů. Pro komunikaci s jednotlivými subsystémy – zařízeními slouží obslužné programy, tzv. drivery. Těchto driverů je v momentální konfiguraci sedm. K obsluze elektroměru slouží `elektromer_driver.py`, k obsluze komunikace s EZS ústřednou `paradox_driver.py`, a k obsluze periferií postavených na arduinech jsou to pak dva drivery `eth_arduino_driver.py` a `usb_arduino_driver.py`. S termostaty tedy obecně moduly Sonoff, komunikuje driver `sonoff_driver.py`. Pro obsluhu modemu pak slouží `gsm_modem_driver.py` a odesílíní mailových notifikací, zajišťuje `mailer.py`. Další programy zajišťující funkcionalitu jsou `even_handler.py`, `event_loger.py` a `mqtt_loger.py`. Event handler naslouchá mqtt a zachytává eventy, a když dostane nějaký, pro který je nastavená obsluha, vygeneruje mqtt zprávy pro nastavené akce. Mailer a GSM driver pak zajišťují komunikaci s uživatelem dvěma nezávislými kanály. Mailer posílá e-maily a GSM driver zajišťuje odesílání SMS prostřednictvím mobilní GSM sítě.

Tyto drivery běží na pozadí v systému Raspbian a shromažďují příchozí data z jednotlivých modulů. Tato data pak zpracují a vytvoří z nich MQTT zprávu, která je pak

přijata serverem Mosquitto a distribuována dále, například do uživatelského interface, dalším driverům, event handleru aopd.

Drivery jsou spouštěny jako služby. Ke spouštění slouží linuxový *systemd* démon pro správu služeb systému. Je to první, kořenový proces všech ostatních procesů, který je spuštěn v user space při zavádění operačního systému. Stará se o spouštění a správu všech ostatních démonů.

https://www.linux-mint-czech.cz/2015/07/co-je-dobre-vedet-o-systemd/

Struktura zpráv

MQTT zprávy, které se distribuují napříč systémem jsou čtyř typů. Dva typy, *události* a *akce* nesou jako obsah – payload zprávu v JSON formátu, obsahujícím předávanou informaci. Další dva typy *stav* a *nastavení*, obsahují jako payload hodnotu. Typ stav je například logická hodnota digitálních vstupů nebo hodnota naměřená analogovým vstupem. Naopak hodnota akce nastavení, odesílá subsystémům výstupní informaci. V případě logických výstupů informaci digitální 1/0, nebo v případě analogových číslo, například požadovanou teplotu do modulu termostat. Uveďme si několik příkladů, jak takové zprávy vypadají.

Příklad zprávy stavové ze zařízení Sonoff:

```
sonoff/stat/STATUS{"Status":
{"Module":4,"FriendlyName":"Sonoff","Topic":"sonoff","ButtonT
opic":"0","Power":1,"PowerOnState":3,"LedState":1,"SaveData":
1,"SaveSt
```

Příklad zprávy z driveru EZS Paradox:

```
bnd/paradox/paradox_1/event/Pohyb {"Cas": "2019-04-26
20:09:02", "Zona": "Obyvak-klima", "_ts": 1556302142.7119343,
"_msg": "Pohyb v z\u00f3n\u011b Obyvak-klima", "Cislo_Zony":
1}
```

Reakce *event_handleru* na předchozí zprávu pro *event_logger:*

```
bnd/event_logger/event_logger_1/action/log_write
{"_event_perif_ident": "paradox_1", "_event": "Pohyb",
"_event_fields": {"_msg": "Pohyb v z\u00f3n\u011b Obyvak-
klima", "_ts": 1556302142.7119343, "Zona": "Obyvak-klima",
"Cislo_Zony": 1, "Cas": "2019-04-26 20:09:02"}, "body":
"Pohyb v z\u00f3n\u011b Obyvak-klima"}
```

Informace o hodnotě z elektroměru

```
bnd/elektromer/elektromer_1/stat/Prikon_L1 0.0
```

Konfigurační soubory

Konfigurační soubory systému slouží k ukládání uživatelského nastavení. Jsou to soubory JSON a jsou v systému ve složce `\var\lib\bnd` tři. Jedná se o soubory `event_konfig.json`, `perif_konfig.json` a `termostaty_konfig.json`. Do všech těchto souborů se ukládají nastavení, která uživatel provedl prostřednictvím uživatelského rozhraní, které bude popsáno v dalších kapitolách. V souboru `event_konfig.json`, jsou uloženy události, které uživatel nakonfiguroval ve

stejnojmené sekci uživatelského rozhraní. V práci je jako vstupní zařízení implementována jen obsluha událostí z ústředny Paradox, ale do budoucna je plánována i obsluha digitálních a analogových vstupů. V souboru `perif_konfig.json` jsou uložena uživatelská nastavení periferních modulů. Jsou zde uloženy názvy – popisy jednotlivých zařízení, jejich vstupů, výstupů, registrů, MAC adresy síťového Arduina, MQTT prefixu u termostatů Sonoff apod. V konfiguračním souboru `termostaty_konfig.json` jsou uloženy časové plány teplot jednotlivých termostatů. Součástí zdrojovího projektu je pak soubor obsahující definice jednotlivých zařízeni. Tento soubor je vytvářen ručně, jako součást zdrojového kódu.

Driver pro elektroměr

Driver elektroměru je poměrně jednoduchý. Ke komunikaci se používá protokol *modbus* a pro python existuje knihovna *pymodbus* která je i v tomto driveru použita. Pomocí funkcionality této knihovny získáme data z registrů elektroměru. Dále je použita knihovna *paho.mqtt* kterí slouží ke komunikaci se serverem Mosquitto a odeslání námi získaných hodnot k dalšímu zpracování případným odběratelům. v našem případě jen webovému rozhraní.

Driver EZS Paradox

Tato obsluha byla nejobtížnější z celé práce. Přesto, že komunikační sběrnice EZS nepoužívá šifrovaná data a používá standardní sériový protokol, bylo obtížné se v datech správně zorientovat. Data jsou vysílána v rámcích 37 byte. Na počátku každého rámce je v jednom byte uložen typ zprávy, takže v podstatě jakákoli hodnota a v posledním byte je kontrolní součet. Protokol tedy neobsahuje žádné synchronizační oblasti, aby bylo možno jednoduše a jednoznačně rozeznat začátek zprávy. Ústředna sama vysílá některé stavové zprávy a jiné vrací až na vnější dotaz. Aby bylo možno nějak rozeznat začátek zprávy, musí driver zkoumat, kdy se mezi daty objeví delší „mezera“, pak načíst 37 byte a zkontrolovat kontrolní součet, pokud se to povede, přijali jsme zprávu, pokud ale například vlivem rušení ztratíme část informace, zpráva se zahodí a proces se opakuje. K popisu formátu zpráv byly použity struktury z výše zmíněného projektu :

https://github.com/jpbarraca/pai/blob/master/paradox/hardware/spectra_magellan/parsers.py

Tento projekt se zabývá komunikací s ústřednami tohoto typu, nicméně pro naše potřeby byl zbytečně rozsáhlý a nevyzpytatelný Tedy z něj bylo použito část definic struktur zpráv, což ušetřilo čas a zbytečnou práci. Je využito knihovny Pythonu construct, která slouží k obousměrnému parsování dat do a z binární podoby.

https://pypi.org/project/construct/

Pro komunikaci na sériovém portu slouží knihovna *pyserial* ta se stará na nejnižší úrovni o přenos jednotlivých byte. Nad ní pracuje třída *framer,* což třída zabezpečující dělení na jednotlivé zprávy – *frame* tedy zprávy zasílané mezi EZS ústřednou a centrální jednotkou po sériovém portu, skládající se ze 37 byte, jejich struktura je viditelná z výše zmíněných definic struktur, které jsou součástí zdrojového kódu. Metoda *loop* z třídy framer běží v dalším vlákně a stará se o zachytávaní jednotlivých frame a ukládání *time-stampů*. Třída *ParadoxProtocol* zajišťuje kódování a dekódování vlastních zpráv zasílaných mezi centrální jednotkou a EZS ústřednou. Driver pak dále zajišťuje příjem a odesílání MQTT zpráv.

Driver ethernetového Arduina

Tento driver zajišťuje komunikaci se subsystémem, který byl vytvořen z modulu Arduino NANO, zkombinovaného s nejlevnějším modelem přídavného modulu síťového rozhraní Ethernet. Tento modul nemá TCP stack. Protože softwarové implementace jsou problematické, bylo rozhodnuto, že bude komunikace postavena na přímém odesílání Ethernet rámců, bez využití protokolů vyšších vrstev. Tedy nad vrstvou linkovou je již vrstva aplikační. Použité rámce mají v hlavičce číslo protokolu 9999 podle nějž je identifikujeme. Tento modul pak lze připojit kdekoli na metalický rozvod sítě Ethernet. Modul má tři digitální vstupy, tři výstupy a jeden vstup analogový s rozlišením 1024 úrovní. Driver má tedy na starost číst všechny vstupy, jak analogové, tak digitální a také odesílat příkazy pro změnu nastavení digitálních výstupů. V driveru jsou použity funkce *on_message* – zpracování MQTT zprávy ze serveru Mosquitto a předání funkci *handle_command* v té se příkaz dále zpracuje a předá funkci *send_socket* která zajistí vytvoření Ethernet rámce a jeho odeslání na danou MAC adresu. Naslouchání MQTT zajišťuje client běžící na pozadí v separátním vlákně. Naslouchání síti je zajištěno pomocí knihovny *socket.* Operace je blokující. Pokud přijde zpráva z některého Ethernet subsystému, je zpracována a přeposlána jako MQTT zpráva dalším odběratelům.

Driver seriového/usb Arduina

Pro komunikaci se sériovým rozhraním nebo rozhraním sériovým prostřednictvém USB připojení, slouží *usb_arduino_driver.py.* Tento driver využívá knihovnu *serial*. Stejně tak, jako ostatní drivery, na jedné straně komunikuje se subsystémem, v tomto případě tedy se sériovým portem prostřednictvím USB převodníku. Z modulu Arduina jsou odesílány textově informace po řádcích, nazpět jsou předávány informace v jednom byte neboť je to nejsnazší způsob, jak předat informaci a je výrazně jednodušší než zpracování textu, které by vyžadovalo bufferování a parsování přijatých informací. Na straně druhé pak se zbytkem systému prostřednictvím MQTT serveru. Obsahuje funkci *on_message* která zajišťuje zpracování příchozí MQTT zprávy a opět je využit client z knihovny paho.mqtt běžící v dalším vlákně a komunikující s MQTT serverem. V hlavní smyčce je naslouchán na seriovém portu.

Driver pro termostaty

S jednotkami Sonoff – TH tedy termostaty, komunikuje *sonoff_driver.py*. Podobně jako ostatní drivery. Rozdíl je v tom, že jednotky Sonoff již samy o sobě generují a přijímají MQTT zprávy. Driver *sonof_driver.py* tedy překládá obousměrně MQTT zprávy z naší aplikace a ze subsystému sonoff. Navíc driver pravidelně načítá konfigurační soubor, ve kterém jsou uloženy uživatelské „mapy“ teplot v průběhu dne a dnů v týdnu. Na základě načtených údajů pak cyklicky odesílá subsystému Wi-Fi termostatu, požadavky na nastavení teploty v konkrétním čase. Měnit nastavenou teplotu lze v každou celou hodinu, což považujeme za dostatečně jemné dělení. Konfigurace časových plánů průběhů teplot je zajištěna webovým rozhraním, v sekci termostaty. Nastavení bude popsáno v kapitolách o uživatelském rozhraní. Z důvodů bezpečnostních, budou jako „pojistka“ využity i stávající termostaty prostorové, případně na přímotopech, které budou nastaveny na nějakou vyšší teplotu a zajistí, že v případě selhání automatizace, nebudou místnosti zbytečně přetopeny.

Driver GSM modemu

Driver modemu stejně jako ostatní drivery naslouchá MQTT zprávám a v případě obdržení zprávy typu – akce, druh – poslat SMS, odešle SMS na nastavená čísla. Pro modem je využito systémových služeb operačního systému – služba ModemManager.

Díky tomu lze použít víceméně jakýkoli modem. Není potřeba jej obsluhovat pomocí *AT příkazů.* Pomocí D-BUS – komunikace mezi procesy je ModemManageru předán požadavek z driveru prostřednictvím knihovny Pydbus.

Mailer

Mailer je obdoba GSM driveru, s tím rozdílem, že posílá prostřednictvím lokálního mail serveru e-mail. Nastane-li událost - event, u kterého si uživatel nastavil ve webovém rozhraní požadavek e-mailové zprávy, je tato zpráva odeslána všem nastavením adresátům. V Linuxu je spuštěn mailserver, a driver se jen připojí na smtp.localhost a odešle zprávy.

MQTT logger

Program MQTT logger jen kontinuálně ukládá aktuální stav systému do json souboru, tedy například stav jednotlivých vstupů a výstupů, registrů elektroměru, informací z ústředny nebo teplot z teploměrů. Z tohoto souboru je možno stav systému kdykoli získat. A je načítán například webovým rozhraním.

Event logger

Program event_logger vytváří soubor event.log. Jeho obsah můžeme získat například prostřednictvím webového rozhraní v sekci *Log.* Do tohoto souboru se ukládají události, které si uživatel předem pro ukládání zvolil v sekci *Události* uživatelského rozhraní. Aktuálně se do souboru ukládají jako ukázka jen vybrané události s EZS ústředny Paradox.

Event handler

Program Event handler zajišťuje vyhodnocování stavových informací – eventů a jejich redistribuci jednotlivým, dalším modulům, na základě rozhodovací logiky nastavené uživatelem. Tedy například při zaarmování, či odarmování ústředny EZS odešle maileru požadavek na odeslání emailu, kdy, kdo a jakou akci provedl a mailer zajistí odeslání informace na přednastavené e-maily. V případě poplachu pak odešle modulu GSM driveru požadavek na odeslání SMS. Do budoucna je plánováno i prozvonění, jako levnější varianta. V uživatelském rozhraní si uživatel nastaví patřičná telefonní čísla. V tuto chvíli nejsou ještě implementovány další plánované funkcionality, uveďme tedy jen pro představu co dalšího by mělo být v budoucnu zprovozněno. Například je možno na některý digitální vstup připojit vhodným způsobem detekci přítomnosti síťového napětí a hlásit vybranými komunikačními kanály výpadky nebo na analogový vstup připojit měření stavu záložní baterie a sledovat případný pokles jejího napětí. Dle závažnosti pak odeslat zprávu zvoleným kanálem. Další zajímavá funkcionalita může být v reakci na pohyb ve střežené zóně například rozsvěcet osvětlení či noční osvětlení o snížené intenzitě a umožnit tak uživateli v noci bezpečný pohyb po domě, bez hledání vypínačů. Další vstupní parametr může být informace tma/světlo, zaarmováno/odarmováno, venkovní teplota, vnitřní teplota, výpadek a napájení ze záložního zdroje, úroveň světelné intenzity, čas a mnoho dalšího. V závislosti na těchto informacích pak můžeme řídit vytápění, větrání, ventilaci, chlazení, žaluzie, zalévání zahrady, obsluhu skleníku, vypínání zbytečných spotřebičů a další v podstatě jakékoli činnosti.

Softvare v periferních subsystémech

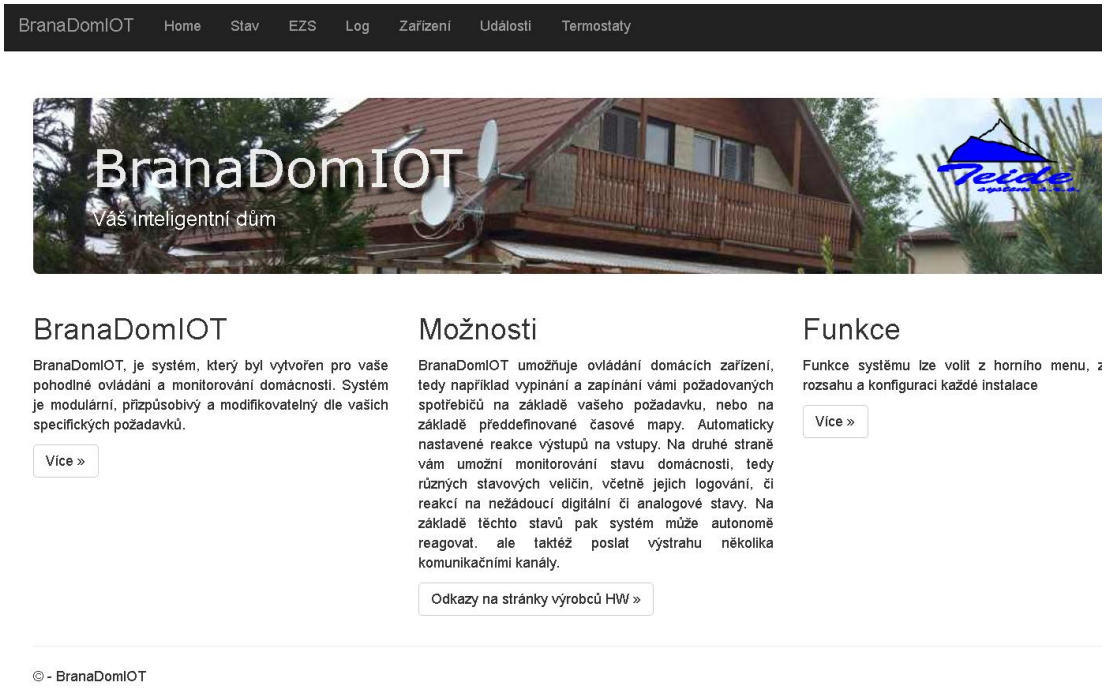
U některých periferních jednotek byl software vytvořen, u jiných upraven a do některých nebylo nijak zasahováno, jen s nimi centrální jednotka komunikuje. Tedy zcela nedotčena zůstala ústředna Paradox a elektroměr, kde je jen využito jejich komunikační sběrnice a vlastností daných výrobcem. Upravený software třetí strany byl nainstalován do termostatu firmy Sonoff, viz popis v dřívějších kapitolách. Vlastní software byl tudíž vytvořen jen pro periferní jednotky založené na Arduinu, tedy v textu je označujeme jako ethernetové a USB Arduino. Pro tyto jednotky byl napsán program s použitím Arduino software a dostupných knihoven pro dané moduly. Software byl vytvořen co možná nejjednodušší aby naplnil požadovanou funkcionalitu. Tak aby byl co nejsrozumitelnější, ale mohl se v případě potřeby dále rozvíjet. Pro přehlednost byly použity na obou modulech jen tři vstupy a tři výstupy digitální a jeden vstup analogový. Bylo by možno jich samozřejmě využít i větší množství. Například analogových vstupních portů lze využít až 8. Pro přehlednost kódu i webového rozhraní však pro tuto práci bylo zvoleno nastavení viz výše.

Webové uživatelské rozhraní

Součástí systému je uživatelské rozhraní. Pro účely této práce bylo zvoleno rozhraní webové, jelikož je zcela platformě nezávislé a vystačíme si tedy s jedním jak pro PC tak pro mobilní telefon s jakýmkoli systémem. Jak již bylo výše zmíněno, je webový server postaven na Pythonu, frameworku *Flask* a šablonovacím nástroji *Jinja.* V části obsluhy je využito různých JavaScriptů a JavaScript knihoven, které zajišťují funkcionalitu na straně prohlížeče. Obsluha webového rozhraní v pythonu je v programu `app.py`. Zde je zajištěno routování všech požadavků ze strany prohlížeče a jejich základní obsluha.

HTML šablony stránek jsou pak uloženy v projektové složce `\bnd\centrala\web\templates\`. Webové rozhraní využívá taky mnoho java script knihoven třetích stran, tyto knihovny jsou uloženy v podsložkách složky `\bnd\centrala\web\static\ .`

Ovládání je rozděleno do několika sekcí, které budou dále popsány sekcí. Ty jsou spustitelné z horní lišty webové stránky. Jsou to sekce *Stav, EZS, Log, Zařízení, Události, Termostaty*.



Obrázek 22: Úvodní stránka systému

Sekce stav

V této sekci je zobrazován stav systému. Obsahuje informace o logických úrovních digitálních vstupů a výstupů, analogových vstupů a další hodnoty získané například z teploměrů a elektroměru. Také jsou zde tlačítka, kterými lze přímo zapínat digitální výstupy.

Sekce EZS

V této sekci jsou zobrazována data získaná z EZS. V tabulce jsou přehledně zobrazeny všechny existující zóny. U každé zóny jsou dva čtverečky. V jednom je modrým probliknutím zobrazován pohyb v dané zóně a v druhém je šedou barvou zobrazován poplach uložený v paměti EZS. Pokud najedeme myší nad dané políčko, zobrazí se time-stamp tedy údaj s časem a datem posledního pohybu nebo poplachu. Nad tabulkou je informace o tom, zda je EZS ve stavu střežení *zaarmováno/odarmováno*. Pod touto informací je tlačítko umožňující zapnout systém do stavu střežení vzdáleně. V případě poplachu se objeví nad tabulkou velký červený obdélník s nápisem poplach.

Obrázek 23: Sekce EZS ústředny

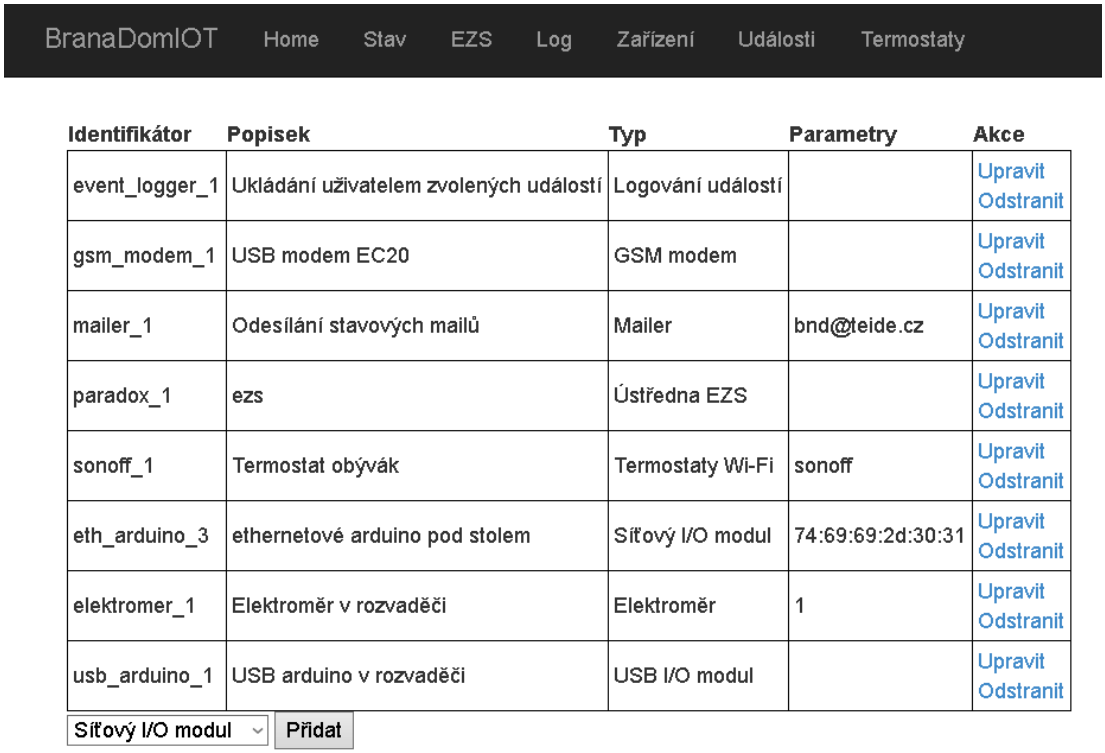
Sekce Log

V této sekci je zobrazen soubor log, do kterého jsou ukládány události, které si uživatel nastavil v sekci události. Prozatím jsou implementovány jen události z EZS Paradox, do budoucna bude možno ukládat většinu stavů, či změn stavů systému, tedy například stisknutí tlačítka, výpadek napájení nebo například pokles nějaké analogové veličiny pod určitou hodnotu, např. napětí nebo teplota.

Obrázek 24: zobrazení souboru logu

Sekce zařízení

Tato sekce umožňuje přidat nebo editovat zařízení, která jsou předem administrátorem systému nadefinována v souboru definic zařízení `periferie.json`. U přidávaného i editovaného zařízení pak umožní editovat potřebné parametry. Tedy například popisky jednotlivých signálů, MAC adresa síťového modulu nebo MQTT prefix u termostatu.

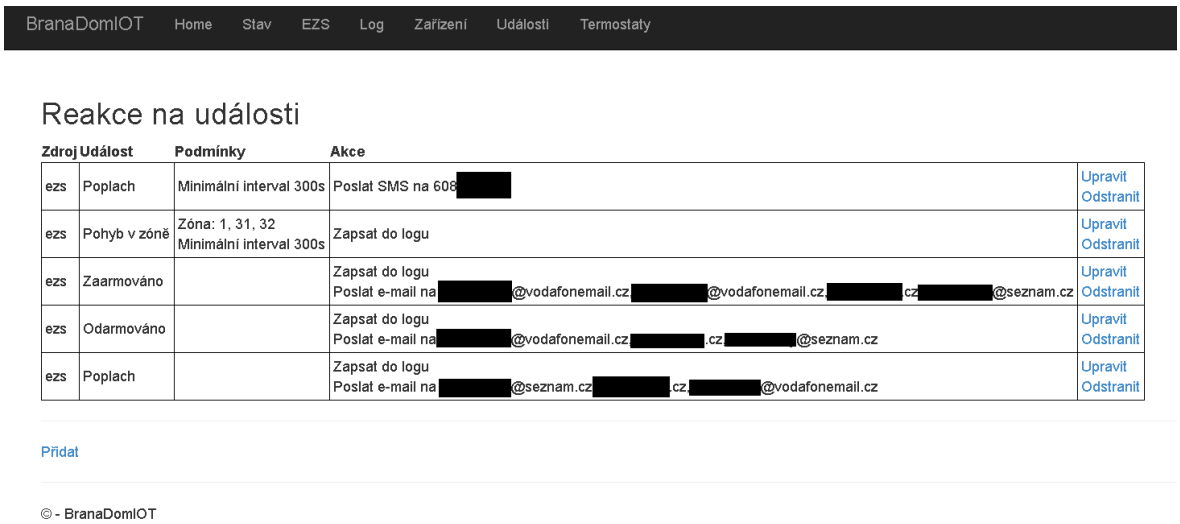


Obrázek 25: Sekce zařízení

Obrázek 26: Ukázka editace zařízení

Sekce události

V této sekci si uživatel může nastavit na jaké události bude napojena jaká akce. V této práci je jako vstupní událost implementováno několik událostí EZS ústředny. Jsou to události zaarmováno, odarmováno, pohyb v zóně a poplach. Jako odezvu lze zvolit akce zápis do logu, poslání mailu, poslání SMS případně zaarmovat. Uživatel si tedy například může nastavit zaslání SMS v době poplachu, e-mailu jako reakci na nastavení nebo odstavení módu střežení – zaarmováno/odarmováno, může ale také sledovat pohyb v zónách i v době kdy je střežení vypnuto a zaznamenat například nežádoucí pohyb osob v částech objektu, kde není pohyb očekáván, třeba v garáži a upozornit na to uživatele nějakou formou zprávy. V budoucnu je tato sekce plánována jako propojovací místo všech událostí a akcí, kde možno v podstatě jakoukoli akci podmínit kombinací téměř jakýchkoli povolených podmínek.



Obrázek 27: Sekce události

Jednotlivé události lze pak přidat a modifikovat. Pro tento účel byla použita knihovna `vuetify-jsonschema-form :`

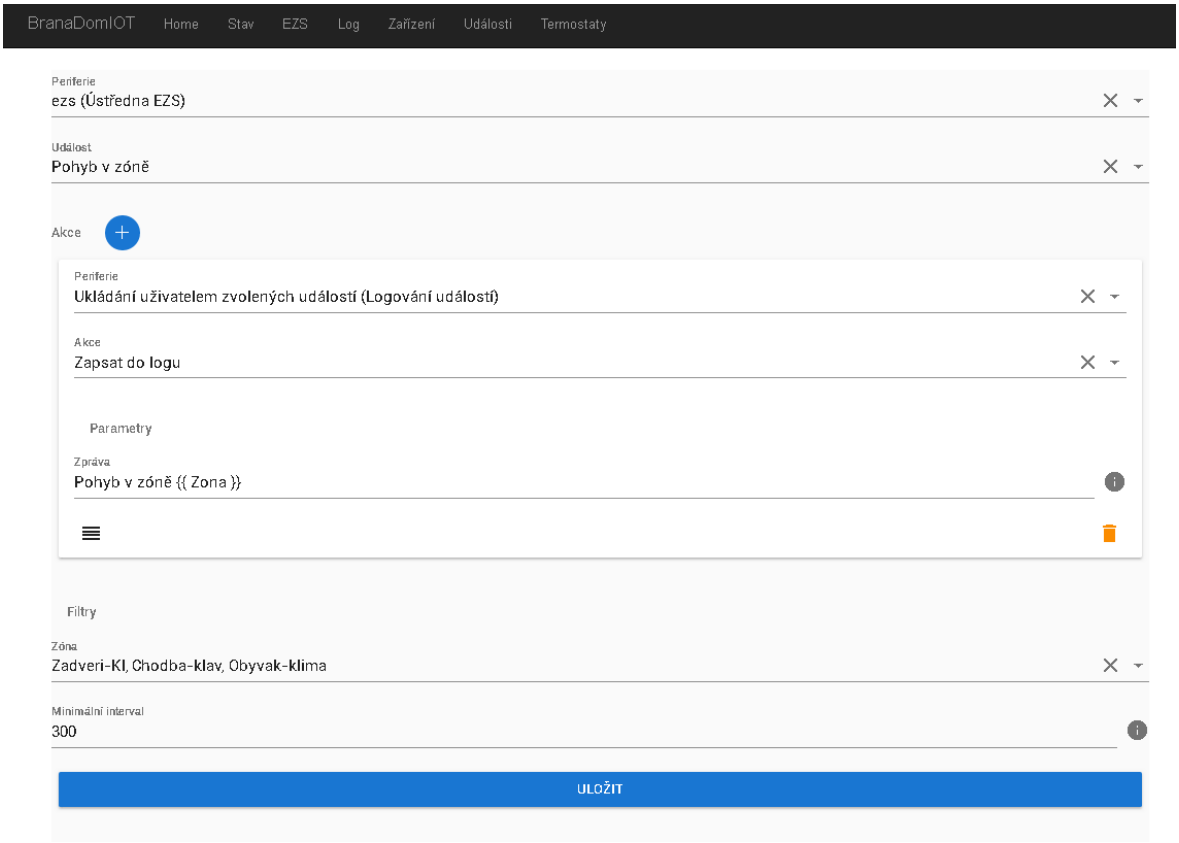
https://github.com/koumoul-dev/vuetify-jsonschema-form

Tato knihovna umožňuje vytvářet komplexní formuláře na základě definice struktury dat, ve formátu JSON schéma.

https://json-schema.org

Funkcionalitu si lze vyzkoušet přímo na webu v demoverzi, kde si můžeme sestavit a vyzkoušet jakýkoli formulář a jakékoli jeho prvky okamžitě:

https://koumoul-dev.github.io/vuetify-jsonschema-form/latest/

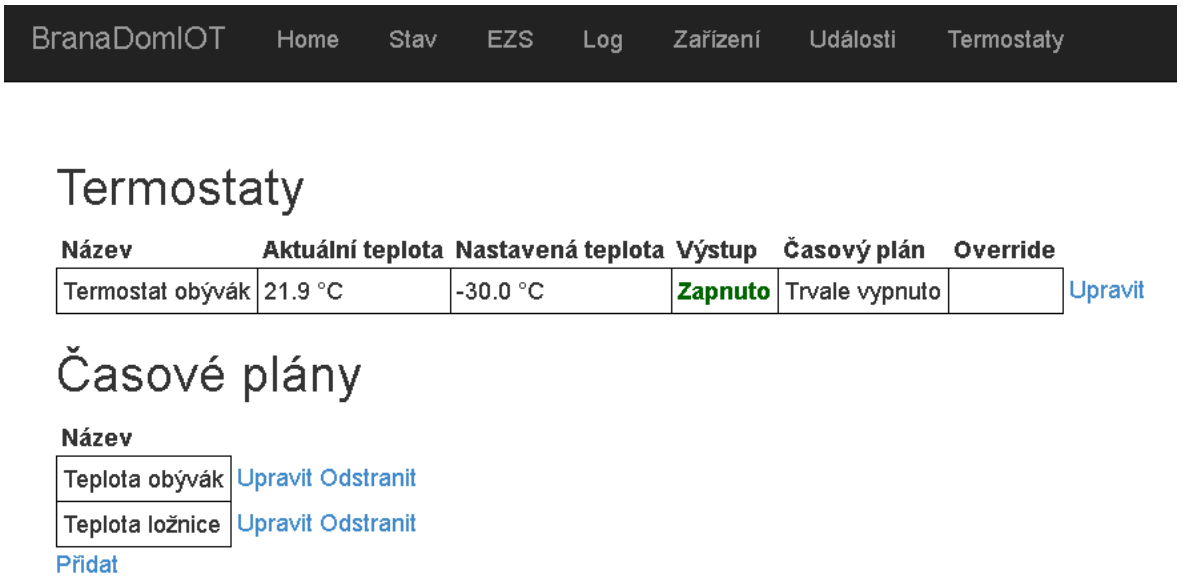


Obrázek 28: Přidání nebo editace reakce na událost

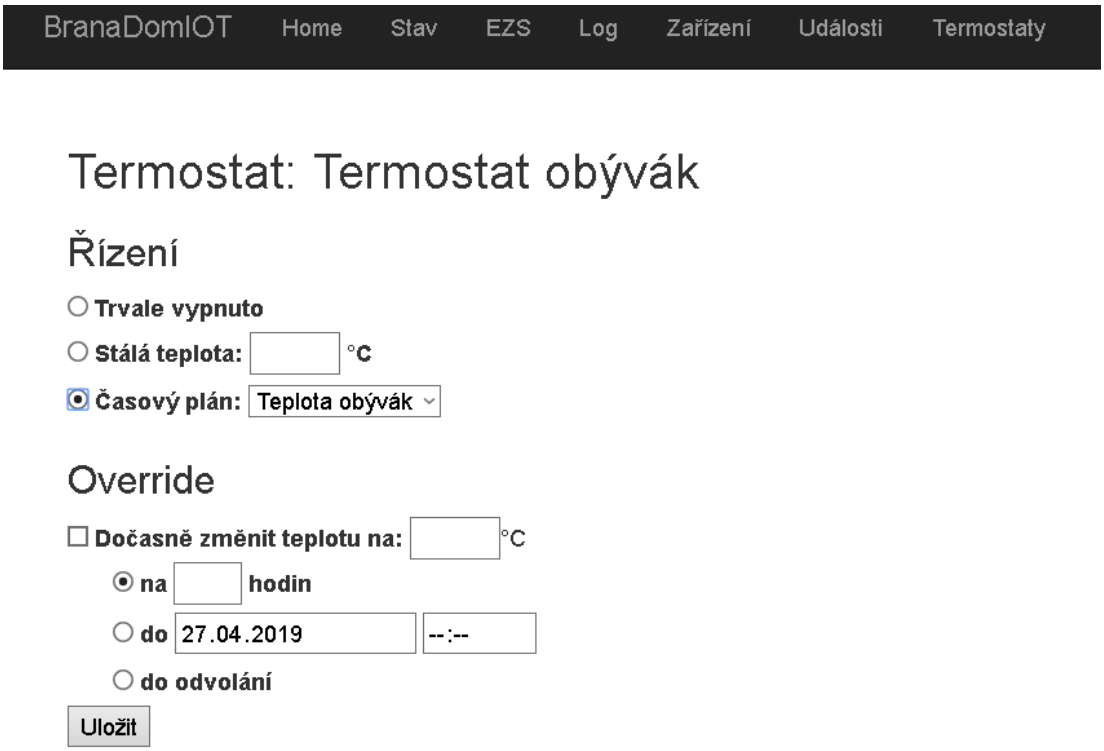
Sekce termostaty

Tato sekce umožňuje nastavení jednotlivých termostatů a časových plánů které lze s termostaty asociovat. V sekci můžeme editovat jak termostaty, tak časové plány a přidávat časové plány. Termostat jako zařízení pak přidáme v sekci zařízení. Časové plány pak lze vytvářet jakékoli a jeden časový plán může být přiřazen více termostatům. V editaci termostatu pak lze vybrat požadovaný časový plán. Další volby pak umožňují nastavení pevné teploty nebo vypnutí. Funkce override pak umožňuje přepnout na pevnou teplotu na určitou dobu a to buď trvání nastavení v hodinách nebo do daného data, či do odvolání.

Časové plány pak nastavujeme na další obrazovce. Přesto, že se může tabulka zdát nemoderní, našim představám zcela vyhovuje. Tabulka obsahuje „rozvrh hodin“ na jeden týden. Teplotu lze měnit v každou celou hodinu, což považujeme za dostatečné. Je to kompromis mezi přehledností a konfigurovatelností. Do tabulky se zapisuje číslem teplota jen pro moment změny, tedy například požadujeme-li jinou teplotu v noci a jinou ve dne, vystačíme si s dvěma čísly ve dvou buňkách pro jeden den, tedy 14 buněk na celý týden. Hodnota na konci tabulky, tedy poslední nedělní je následována první buňkou pondělní v dalším týdnu. Systém se pak sám postará o dovyplnění tabulky mezi buňkami patřičnou barvou, která by měla svou barevnou teplotou lépe navodit představu teploty v daném úseku. Dělení po týdnech považujeme za nejpřirozenější možnost. Samozřejmě vyhovující obyvatelům s pravidelnou pracovní dobou. Pokud by například uživatel pracoval ve směnném provozu, dalo by se to řešit přepínáním více časových plánů. Samozřejmě nelze v tomto případě vyhovět příliš bizarním požadavkům, aniž by se systém nestal v oblasti tohoto nastavení nepřehledný.



Obrázek 29: Úvodní stránka sekce termostaty



Obrázek 30: Editace termostatu

Obrázek 31: Ukázka vytvořeného časového plánu

Použitý software třetích stran

V práci bylo použito mnoho různých svobodný knihoven k zajištění dílčích funkcionalit. Jedná se jak o knihovny Pythonu, které byly zpravidla zmíněny u jednotlivých programů, tak o knihovny Java Script, pracující na straně prohlížeče a knihovny z projektu Arduino u periferních jednotek.

Upravené a začleněné do projektu

Název: Gateway for Wireless temperature and humidity sensors
Autor: Jaakko Ala-Paavola
URL: http://ala-paavola.fi/jaakko/doku.php?id=wireless
Licence: neznámá
Použití: Dekódování signálu 433 MHz z meteostanice WT450
Umístění v projektu: periferie/usb_433_I_O_arduino/usb_433_I_O_arduino.ino, funkce receive()

Název: EtherCard
Autor: Guido Socher
Licence: GPLv2
URL: https://github.com/njh/EtherCard
Použití: Obsluha ethernetového čipu ENC28J60 z desky Arduino Nano
Umístění v projektu: periferie/ethercard_raw/enc28j60.*

Název: PAI
Licence: EPL 2.0
Autor: João Paulo Barraca
URL: https://github.com/jpbarraca/pai
Použití: Převzaty pouze definice některých binárních struktur pro komunikaci s EZS Paradox

Umístění v projektu: centrala/obsluha_perif/paradox_driver.py, definice struktur na začátku souboru (instance tříd z knivony Construct, např. Struct či Enum)

Distribuované s projektem

Název: Vue.js
Licence: MIT
Autor: Evan You a další přispěvatelé
URL: https://github.com/vuejs/vue
Použití: Dynamická aktualizace stavu na stránce EZS ve webovém rozhraní
Umístění v projektu: centrala/web/static/vue/vue.js, použito v centrala/web/templates/paradox.html

Název: MQTT.js
Licence: MIT
Autor: Matteo Collina, Adam Rudd, další přispěvatelé
URL: https://github.com/mqttjs/MQTT.js/
Použití: Dynamická aktualizace stavu na stránce EZS ve webovém rozhraní
Umístění v projektu: centrala/web/static/scripts/mqtt.min.js, použito v centrala/web/templates/paradox.html

Název: vuetify-jsonschema-form
Licence: MIT
Autor: Alban Mouton
URL: https://github.com/koumoul-dev/vuetify-jsonschema-form
Použití: automatické generování komplexních formulářů na stránce Události ve webovém rozhraní
Umístění v projektu: centrala/web/static/vue/vuetify-jsonschema-form.*, použito v centrala/web/templates/udalosti_edit.html

Název: Vuetify
Licence: MIT
Autor: John Leider a kolektiv
URL: https://github.com/vuetifyjs/vuetify
Použití: závislost pro vuetify-json-schema-form
Umístění v projektu: centrala/web/static/vue/vuetify.*

Název: Material Icons
Licence: Apache 2.0
Autor: Google
URL: https://material.io/tools/icons/?style=baseline
Použití: závislost pro Vuetify

Umístění v projektu: centrala/web/static/vue/MaterialIcons-Regular.*

Název: jQuery
Licence: MIT
Autor: John Resig, Timmy Willison a další přispěvatelé
URL: https://github.com/jquery/jquery
Použití: zjednodušení základních JavaScriptových operací (např. manipulace s DOM)
Umístění v projektu: centrala/web/static/scripts/jquery-*, centrala/web/static/vue/jquery-3.3.1.js

Název: Bootstrap
Licence: MIT
Autor: Mark Otto a kolektiv
URL: https://github.com/twbs/bootstrap
Použití: vizuální prvky ve webovém rozhraní (rozvržení stránky, hlavní menu, část stylů)
Umístění v projektu: centrala/web/static/scripts/bootstrap.js, centrala/web/static/content/bootstrap.*

Název: Modernizr
Licence: MIT
URL: https://github.com/modernizr/modernizr
Použití: podpora různých prohlížečů
Umístění v projektu: centrala/web/static/scripts/modernizr-2.6.2.js

Název: Respond.js
Licence: MIT
Autor: Scott Jehl a kolektiv
URL: https://github.com/scottjehl/Respond
Použití: podpora pro responzivní webové rozhraní

Externí závislosti

Název: Arduino knihovny
Licence: LGPL
Autor: Massimo Banzi, David Cuartielles, Tom Igoe, David A. Mellis a další přispěvatelé
URL: https://github.com/arduino/Arduino
Použití: základní knihovny pro periferie na deskách Arduino Nano (USB I/O modul, ethernetový I/O modul)

Název: Flask
Autor: Armin Ronacher a kolektiv
Licence: BSD

URL: https://github.com/pallets/flask
Použití: framework, na kterém je postavené webové rozhraní

Název: pySerial
Licence: BSD
Autor: Chris Liechti
URL: https://github.com/pyserial/pyserial/
Použití: sériová komunikace s EZS Paradox, USB I/O modulem a elektroměrem

Název: PyModbus
Licence: BSD
URL: https://github.com/riptideio/pymodbus
Použití: komunikace s elektroměrem

Název: Construct
Autor: Arkadiusz Bulski, Tomer Filiba, Corbin Simpson, další přispěvatelé
URL: https://github.com/construct/construct
Použití: konstrukce a dekódování binárních zpráv a struktur při komunikaci s EZS Paradox

Název: Eclipse Paho MQTT Python Client (paho.mqtt)
Licence: EPL 1.0, EDL 1.0
Použití: MQTT komunikace z Pythonu
URL: https://github.com/eclipse/paho.mqtt.python/

Název: pydbus
Licence: LGPL 2.1
Autor: Linus Lewandowski
URL: https://github.com/LEW21/pydbus
Použití: komunikace s ModemManagerem pro obsluhu GSM modemu

Název: ModemManager
Licence: GPL2
Autor: Aleksander Morgado a kolektiv
URL: https://www.freedesktop.org/wiki/Software/ModemManager/
Použití: komunikace s GSM modemy

2.3.1 Realizace testovací sestavy

Abychom uvedli vše výše zmíněné do provozu, byly prvky provizorně umístěné do krabice s DIN lištami, kde bude i v budoucnu jejich místo. Nicméně zatím je vše umístěni jen jaksi přechodně a „na divoko“, protože v průběhu zkoušení a testování se bude do takovéhoto prototypu pravděpodobně muset mnohokrát zasáhnout. Finální upevnění prozatím odsuneme až do doby, kdy bude toto provizorium odzkoušeno v dlouhodobém provozu. Rozvodná krabice je umístěna poblíž domovního rozvaděče 96 modulů. A vedle rozvaděče nízkonapěťového, odkud jsou napájeny aktivní prvky sítě. Vedle tohoto je pak skříň ústředny EZS, napájecí ATX zdroj, zálohovaný UPS 750VA. Nad ústřednou EZS je umístěn 10“ RACK, v němž se nachází hlavní 24 portový switch Ethernet, VOIP brána, GSM brána a telefonní ústředna. Další prvky stávajícího zabezpečení jsou umístěny na bezpečnějších vzdálených místech.

Obrázek 32: Provizorní testovací sestava

V levé skříni je osazen USB hub, vedle něj USB LTE modem , v dolní řadě digitální elektroměr a vpravo pak srdce systému Raspberry Pi. Zcela dole je pak venkovní teploměr 433MHz s čidlem teploty. Byl umístěn pro účely testování hned vedle přijímače. Díky rušení přijímače, byly občasné problémy se vzdáleností a docházelo k výpadkům přenosu. Přijímač tedy bude nutno umístit mimo rušení a experimentálně mu stvořit nějakou anténu pro větší dosah, alespoň po vnějším obvodu domu, kam je plánováno čidla umístit. Pravá skříň pak bude využita pro budoucí technologie, prozatím je v ní osazen komerční GSM komunikátor, jako záloha, pro poplach, pokud by byl náš systém napaden a odstaven nebo došlo k jeho poruše. Kabely jsou natahány také jen provizorně. I elektroměr je jen testovací. Máme zapůjčen třífázový ale jen jedno sazbový. V konečné budoucí realizaci bude elektroměr umístěn přímo do silového rozvaděče kde nahradí stávající, který komunikaci neumožňuje. Budoucí elektroměr je však potřeba osadit třífázový a dvou sazbový. Nicméně jeho cena je stále několik tisíc korun, takže využijeme v testovacím provozu jednodušší přístroj, k tomuto účelu zapůjčený.

Závěr

Účelem této práce bylo vytvořit funkční systém domácí automatizace, který by byl schopen komunikovat s dalšími prvky, ať již továrními, upravenými nebo vlastní konstrukce. Největší výzvou bylo zvládnout komunikaci s EZS ústřednou Paradox Spectra. Domnívám se, že práce požadavky naplňuje. Jistě by mohla zahrnovat větší množství různých periférií, a funkcionalit, nicméně ukázalo se, že problematika je podstatně širší a pokud má být systém navržen komplexně, rozšiřitelně a univerzálně, je to mnohem pracnější, než jsem na počátku očekával. Původní představa byla významně softwarově jednodušší, předpokládala víceméně „zadrátovanou logiku“, tedy její rozšiřování by bylo vždy zásahem do programového vybavení. To by pro případně širší využití systému pro jiné uživatele dost zkomplikovalo možné nasazení. Systém tedy byl navržen obecněji, což podstatně zvýšilo nároky na obslužný software. Původní představa byla taková, že bude spouštěno méně služeb a budou obsluhovat více funkcí najednou. Nakonec se však ukázalo, že kvůli rozšiřitelnosti je potřeba postupovat poněkud obtížnější cestou s vyšší modularitou systému. Nakolik bude systém stabilní ukáže až dlouhodobé testování. Zatím se ukazuje, že občas přestanou pracovat čínské USB převodníky, jednoho typu. Pravděpodobně by tento problém měla vyřešit výměna za jiný typ. Vzhledem k tomu, že k výpadku dojde jednou za několik dnů, bude potřeba delší doby testovacího provozu, aby bylo možno odhalit příčinu vzniku takové chyby. Další možné řešení by bylo hlídat komunikaci a při výpadku převodníky programově přes reléo dpojit a znovu připojit na napájecí napětí. Podařilo se úspěšně zvládnout komunikaci s ústřednou Paradox. Jediný problém je v odchytávání paketů, kdy ojediněle může dojít ke ztrátě jednoho nebo několika paketů, například díky rušení, nebo při zaneprázdnění centrály, nějakou náročnější činností. Zdá se, že to ale nikterak nebrání požadované funkcionalitě, neboť události se odesílají opakovaně a stav ústředny je načítán na základě dotazu centrály také opakovaně. Poplach je zálohován komerčním GSM psgerem, připojeným přímo na výstup sirény EZS. Pro bezpečnější odchytávání dat z ústředny by bylo potřeba real-timového prostředí, tedy například zajistit komunikaci s EZS na platformě Arduino přímo v ústředně a dále pak posílat informaci bezztrátově. Nicméně vzhledem k velmi řídkým výpadkům, postačuje pro požadovaný účel funkcionalita stávající, zálohovaná pagerem. Další částí, jejíž funkcionalitu považuji za zcela dokončenou pro reálné nasazení je ovládání WI-fi termostatů. Vše funguje jak má a bylo již zkoušeno samostatně v několikaměsíčním provozu. Časové plány se ukázaly jako pro mé účely zcela dostatečné a obsluha velmi efektivní. Do budoucna bude ještě doplněno grafické zobrazování historie teplot a jejich ukládání pro případné analýzy spotřeby a podobné účely. Prozatím jsou data ukládána a lze je zobrazit softwarem třetí strany v sekci Grafy, doplněné na poslední chvíli. Některé funkcionality jsou v práci spíše demonstrační. U Arduino modulů jsou výstupech jen signalizační LED diody a na vstupech tlačítka. To je pro demonstraci funkčnosti systému zcela dostatečné, ale pro reálný provoz bude ještě potřeba vyřešit řadu záležitostí. Jedná se hlavně o vstupně výstupní prvky. Před dokončeném práce SD karta se systémem padla, zda opětovnými zápisy je obtížné říci. Byla tedy nahrazena SSD diskem M2 Western Digital 250GB, čímž by měly být problémy navždy vyřešeny. Raspberry umí z disku připojeného přes USB redukci i nabootovat, tedy systém běží zcela bez SD karty. Tyto SSD HDD si ve vlastní režii kontrolují, aby nebylo zapisováno do stejného místa paměti, tedy se o zápisy netřeba příliš starat. Vstupně výstupní moduly budou muset být vybaveny výkonovými výstupními prvky, nejčastěji tedy bezpečná relé pro spínání síťové zátěže. Na vstupech se mohou vyskytnout relé, optočleny, napěťové děliče, děliče s foto-rezistory, akustické snímače, snímače přiblížení, tlačítka, vypínače apod. Na vstupech analogových pak převodníky dalších fyzikálních veličin na napětí. Minimálně u silových prvků bude

optimální použít nehořlavých instalačních krabic. Taktéž termostaty Sonoff budou při realizaci umístěny do nehořlavých krabic. Také elektroměr je momentálně v systému spíše jako demonstrace možnosti sběru dat. Data se tedy jen zobrazují, ale neukládají se pro další analýzu, ani pro ovládání dalších zařízení. Mohla by být využita například ke spínání a odpínání určité zátěže při stoupající spotřebě, nebo k vybalancování zátěže jednotlivých fází, aby bylo maximálně využito proudové hodnoty hlavního jističe. Data z elektroměru mohou být také využita spolu s informací z HDO o přepnutí na nízký nebo vysoký tarif, případně je do budoucna plánováno zapojení solárních panelů a záložní baterie tzv. *Energy Cloudu* a pro řízení spotřeby se pak přímo nabízí systém dovybavit, případně doplnit o komunikaci se subsystémy cloudu. Systém je momentálně nasazen v testovacím provozu a zatím se zdá, že až na výše zmíněné drobné problémy, pracuje tak, jak se od něj očekávalo.

Význam některých použitých zkratek a pojmů

MQTT – dříve: Message Queuing Telemetry Transport

UTF – Unicode Transformation Format

IoT – internet of things - internet věcí

PGM – programovatelné výstupy

OS – operační systém

EZS – elektronický zabezpečovací systém

QoS – Quality of service - požadovaná úroveň kvality síťových služeb

CCTV – Closed-circuit television, uzavřený televizní okruh

FTP – File Transfer Protocol

MCU – jednočipový počítač, mikrokontrolér, MCU

GPIO – General purpose I/O - vstupně výstupní univerzální vývody MCU

SSD HDD – Solid State Hard Disk - pevný disk založený na pamětech FLASH

Ambient – prostředí, okolí

Ethernet – název souhrnu technologií pro počítačové sítě (LAN, MAN) z větší části standardizovaných jako IEEE 802.3

Access-point – Přístupový bod k bezdrátové Wi-Fi síti je zařízení, ke kterému se klienti připojují

Midle-ware – specializovaný software, který poskytuje aplikacím služby nad rámec služeb poskytovaných operačním systémem

Platforma – počítačová platforma je v informatice pracovní prostředí, jak po stránce hardware, tak i software

front end a back end – oddělení odpovědnosti mezi prezentační vrstvou (front endem), a vrstvou operující se samotnými daty (back end) softwaru, fyzické infrastruktury nebo hardwaru

SMART HOME – technologie chytrého domu

Shield – přídavná zpravidla periferní deska k platformám mini a mikropočítačů, jako je např Raspberry, nebo arduino

Parsování – proces, při kterém parser zpracovává nejaky řetězec, prochází ho a vybírá z něj hledané údaje

Time-stamp – údaj obsahující informaci o datu a času

Literatura:

AL-QUTAYRI, by Mahmoud A., ed. *Smart Home System*. Vukovar, Croatia: In-Teh, 2010. ISBN 978-953-307-050-6.

BROWN, John N.A., Anton Josef FERCHER a Gerhard LEITNER. *Building an intuitive multimodal interface for a smart home: Hunting the SNARK*. New York, NY: Springer Berlin Heidelberg, 2017. ISBN 978-3-319-56531-6.

ELSENPETER, Robert C. *Build Your Own Smart Home*. Emeryville, California 94608: Brandon A. Nordin, 2003. ISBN 0-07-223013-4.

GOODWIN, Steven. *Smart home automation with Linux and Raspberry Pi*. Second edition. Berkeley, CA?: Apress, [2013]. Technology in Action Press book. ISBN 978-1-4302-5887-2.

KYAS, Othmar. *How To Smart Home.: A Key Concept Book*. 3rd Edition. Wyk, Germany: Key Concept Press e.K., 2015. ISBN 978-3-944980-06-5.

LEITNER, Gerhard. *The Future Home is Wise, Not Smart: A Human-Centric Perspective on Next Generation Domestic Technologies*. New York Dordrecht London: Springer Cham Heidelberg, 2015. ISBN ISBN 978-3-319-23092-4.

HILLAR, Gastón C. *MQTT Essentials: A Lightweight IoT Protocol*. Birmingham: Packt Publishing Ltd., 2017. ISBN 978-1-78728-781-5.

VALEŠ, Miroslav. *Inteligentní dům*. Brno: ERA, 2006. 21. století. ISBN 80-736-6062-8.

VANDOME, Nick. *Smart Homes in Easy Steps*. Hamilton Terrace: In Easy Steps Limited, 2018. ISBN 978-1840788259.

WELLS, Quentin. *SMART GRID HOME: smart DIY designs for a stylish home*. Clifton Park, NY: CENGAGE Learning, [2013]. ISBN 11-113-1851-4.

Elektronické zdroje:

https://citaty.net/citaty/5807-miroslav-hornicek-domov-vzdycky-to-budu-opakovat-neni-prostor-a/

https://mqttfx.jensd.de/index.php

https://www.tvfreak.cz/raspberry-pi-orange-pi-banana-pi-ktery-jako-htpc/5533-4

https://cz.farnell.com/arduino-org/a000005/arduino-nano-evaluation-board/dp/1848691

https://sonoff.itead.cc/en/products/sonoff/sonoff-th

https://www.stasanet.cz/Paradox-a-ostatni-EZS/Systemy-SPECTRA-MAGELLAN/Ustredny-Magellan/MG-5050-ustredna-EZS-2x5-10-zony-ATZ-max-32-zon-4xPGM.html

http://www.eastrongroup.com/data/uploads/SDM530C_Three_Phase_Multi-function_Remote_Control_Direct_Connect_Energy_Meter_with_Built-in_Relay.pdf

http://www.eastrongroup.com/data/uploads/Eastron_SDM630MV_CT_protocol_V1_0_.pdf

https://www.quectel.com/product/ec21minipcie.htm

https://www.quectel.com/product/minievb.htm

https://www.quectel.com/product/minievb.htm

https://www.santy.cz/moduly-c22/modul-433mhz-arduino-sada-i63/

https://www.santy.cz/moduly-c22/modul-433mhz-arduino-sada-i63/

https://www.hodinarstvi.cz/produkty/meteostanice/nahradni-senzory/bezdratove-cidlo-k-modelu-garni-2100-wt450.html

https://arduino-shop.cz/arduino/1003-arduino-433mhz-vysilac-prijimac.html

https://www.arduiner.com/en/usb-to-ttl-rs232-rs485/8461-cp2102-usbttlrs485rs232-interconversion-6-in-1-serial-converter-uart-module-3809200642792.html

https://www.ptshop.cz/Prevodnik-USB-na-TTL-RS232-PL2303-d143.htm

https://github.com/arendst/Sonoff-Tasmota

https://blog.vyoralek.cz/iot/sonoff-produkty-nahrani-firmware-tasmota/

https://www.linux-mint-czech.cz/2015/07/co-je-dobre-vedet-o-systemd/

https://github.com/jpbarraca/pai/blob/master/paradox/hardware/spectra_magellan/parsers.py

https://pypi.org/project/construct/

https://github.com/koumoul-dev/vuetify-jsonschema-form

https://json-schema.org

https://koumoul-dev.github.io/vuetify-jsonschema-form/latest/

http://ala-paavola.fi/jaakko/doku.php?id=wireless

https://github.com/njh/EtherCard

https://github.com/jpbarraca/pai

https://github.com/vuejs/vue

https://github.com/mqttjs/MQTT.js/

https://github.com/koumoul-dev/vuetify-jsonschema-form

https://github.com/vuetifyjs/vuetify

https://material.io/tools/icons/?style=baseline

https://github.com/jquery/jquery

https://github.com/twbs/bootstrap

https://github.com/modernizr/modernizr

https://github.com/scottjehl/Respond

https://github.com/arduino/Arduino

https://github.com/pallets/flask

https://github.com/pyserial/pyserial/

https://github.com/riptideio/pymodbus

https://github.com/construct/construct

https://github.com/eclipse/paho.mqtt.python/

https://github.com/LEW21/pydbus

https://www.freedesktop.org/wiki/Software/ModemManager/

Přílohy:

Elektronická příloha:

Zdrojové kódy na přiloženém DVD, spolu s textem práce v elektronické formě.