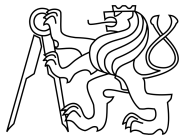
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Autonomní systém pro automatizaci a monitoring budov
Student:	Bc. Daniel Pospíšil
Vedoucí:	Ing. Pavel Benáček
Studijní program:	Informatika
Studijní obor:	Počítačové systémy a sítě
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Proveďte analýzu a návrh vlastního řešení autonomního systému pro automatizaci a monitoring bytových prostor. Při tvorbě návrhu klaďte důraz na budoucí rozšiřitelnost. Pro implementaci použijte jednodeskový počítač. Pomocí monitoringu bude možné zjistit narušení bytových prostor, základní údaje o klimatických podmínkách včetně detekce možného nebezpečí úniku plynu a přítomnosti kouře. V případě neautorizovaného vstupu systém pošle notifikaci a pořídí vizuální záznam. Autorizace bude probíhat pomocí bezkontaktní technologie. Dále vytvořte modul pro ovládání elektrických spotřebičů a světel. Modul bude demonstrován na ovládání osvětlení. Automatizace bude umožněna pomocí kombinací modulů v centrální řídící jednotce. Komunikace mezi moduly bude realizována bezdrátově. Pro automatizaci implementujte logiku na rozsvícení osvětlení při neoprávněném vstupu do monitorovaného prostoru. Monitoring bude přístupný přes protokoly HTTP a SSH.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. října 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

Autonomní systém pro automatizaci a monitoring budov

Bc. Daniel Pospíšil

Vedoucí práce: Ing. Pavel Benáček

9. května 2016

Poděkování

Rád bych poděkoval svému vedoucímu panu Ing. Pavlu Benáčkovi za cenné připomínky, rady, nápady a hlavně čas, který věnoval pomoci při vzniku této diplomové práce. Dále bych rád poděkoval panu Bc. Petru Medonosovi za veškeré připomínky k této diplomové práci. V neposlední řadě bych rád poděkoval všem lidem, kteří mě podporovali při psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. května 2016 .....

Odkaz na tuto práci

Pospíšil, Daniel. *Autonomní systém pro automatizaci a monitoring budov.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato diplomová práce se zabývá domácí automatizací. Práce diskutuje různé možnosti pro realizaci domácí automatizace, analytickou část návrhu pro popis implementace a její efektivitu s ohledem na limity pro komunikaci v počtu zařízení. Jako implementační platforma byly využity jednodeskové počítače Raspberry Pi a Arduino. Pro implementaci byly zvoleny programovací jazyky Python ve verzi 3.4, Wiring a PHP ve verzi 5.6.

Klíčová slova Domácí automatizace, IoT, Python, nRF24, Arduino, Raspberry Pi.

Abstract

This diploma thesis is focused on home automation. This thesis describes different approaches to home automation, an analytical part of design of the implementation, the implementation itself and a discussion of communication efficiency based on number of devices. Raspberry Pi and Arduino were used as the main implementation platforms, programming language Python in version 3.4, Wiring and PHP in version 5.6 were used to implement the solution.

Keywords Home automation, IoT, Python, nRF24, Arduino, Raspberry Pi.

Obsah

Seznam obrázků

Úvod

Domácí automatizace dovoluje člověku automatizovat veškeré běžné činnosti, které nějakým způsobem ovlivňují každodenní chod domácnosti. Od nastavení teploty termostatu, přes automatické rozsvícení světel až po ovládání bezpečnostních prvků v objektu. Toto vše bez jakéhokoliv zásahu uživatele probíhá automaticky podle jeho preferencí. Další neoddělitelnou části domácí automatizace je její monitoring. Uživatel může sledovat závislosti mnoha ukazatelů na svých nastaveních – například spotřebu energií. Tento ukazatel má přímý vliv na finanční náročnost objektu. Mezi další ukazatele můžeme zařadit teplotu a vlhkost v objektu, kde tento ukazatel má přímý vliv na subjektivní pocit komfortu jednotlivých osob v objektu. Těchto ukazatelů je široká škála a je na každém uživateli co si přeje sledovat. Na základě sledovaných ukazatelů může uživatel dělat rozhodnutí pro úpravu nastavení, doplnění systému o další prvky či dokonce stavební úpravy objektu.

Mezi základní výhody domácí automatizace patří zvýšení komfortu uživatelů objektu, kteří se již nemusí starat o běžné činnosti, jako je větrání objektu, nastavení teploty či opomenutí vypnutí světel. Všechny tyto činnosti se dějí automaticky a dovolují uživatelům je provádět vzdáleně pomocí internetu například i z dovolené. V případě přítomnosti zabezpečovacího modulu s kamerami je možné zjistit neoprávněné vniknutí do objektu, narušitele vizuálně zaznamenat a zavolat bezpečnostní složky ještě v době neoprávněného narušení objektu.

V posledních letech se zvyšuje zájem o domácí automatizaci nejen mezi lidmi s technickým vzděláním, ale hlavně mezi lidmi, kteří mají zájem na ekologickém využívání energií a snižování množství odpadu, který vyprodukují v jakékoliv formě. Tohoto si všímají i firmy, které se přirozeně snaží na tomto segmentu čím dál tím více vydělat. Dle Transparency Market Research [9] dosáhne hodnota celosvětového trhu s domácí automatizací v roce 2020 na celkovou částku 21,6 miliardy amerických dolarů.

Dnes lze na trhu najít veliké množství komerčních produktů pro domácí automatizaci. Jeden z faktorů těchto firem je však profit, který může náklady

na pořízení systému zvýšit. V rámci této diplomové práce se budu zabývat analýzou produktů Ubiquiti mFi, Belkin WeMo a Samsung SmartThings.

V případě komunitních systémů jsou naopak náklady mnohem nižší, ale uživatel se musí spolehnout na podporu komunity. Na druhou stranu je u většiny projektů možné využít běžně dostupné senzory mnoha výrobců a ty pak zkombinovat v rámci jednoho systému. V rámci této diplomové práce se budu zabývat analýzou open source projektů Freedomotic, OpenHab a Domoticz.

Práci jsem rozdělil do tří základních kapitol. V první kapitole s názvem Analýza a návrh se zabývám analýzou existujících řešení domácí automatizace, analýzou možných platforem pro provoz domácí automatizace a výběrem implementační platformy. Dále v rámci této platformy specifikuji požadavky, které musí výsledné řešení splňovat. V neposlední řadě se zabývám návrhem samotné implementace. V podkapitole Návrh řešení se věnuji popisu domácí automatizace s popisem centrální řídící jednotky (kontroléru), tří základních modulů, ověření identity pomocí bezkontaktních karet a monitoringem. Druhá kapitola s názvem Realizace popisuje implementaci jednotlivých částí zmíněných v rámci návrhu řešení. Třetí kapitola je věnována testování implementovaného systému a v závěru této kapitoly je provedena diskuze dosažených výsledků a diskuze budoucích úprav.

Cíl práce

Cílem této diplomové práce je vytvoření funkčního systému domácí automatizace a monitoringu. Tento systém bude umět monitorovat stav a kvalitu ovzduší v monitorovaném objektu, dokáže objekt zastřežit a odstřežit pomocí bezkontaktních karet. Dále dokáže zaznamenat neoprávněný vstup do zastřeženého prostoru a poslat uživateli upozornění pomocí emailu. V neposlední řadě bude systém umět ovládat osvětlení a další zařízení dle přání uživatele pomocí spínacího relé. Veškerá komunikace mezi moduly a centrální řídící jednotkou bude probíhat bezdrátově a nebude závislá na jiné žádné síti. V rámci centrální řídící jednotky bude možné jednotlivé moduly provázat a provádět akce na základě událostí z jiných modulů dle nastavení uživatele. V rámci této diplomové práce bude implementována logika pro rozsvícení světel v případě, že prostor je zastřežen a došlo k jeho narušení, které bylo zjištěno pomocí detektoru pohybu. Systém bude umístěn v bytové jednotce o výměře 45 m^2 s dvěma místnostmi a chodbou. Dále bude otestován maximální dosah bezdrátové komunikace mimo bytovou jednotku v kancelářském prostoru s přítomností bezdrátových prvků, které jsou využívány pro komunikaci.

Hlavní důraz je kladen na budoucí rozšiřitelnost systému. Díky existenci široké škály různých senzorů a prvků je nutné, aby existovala možnost, jak tyto prvky jednoduše připojit k tomuto systému. Mimo to bude kladen důraz na přehlednou prezentaci údajů ze systému a možnost základního ovládání systému uživatelem.

Kapitola 1

Analýza a návrh řešení

Při analýze jsem nastudoval tři existující komerční řešení a tři nejpopulárnější open source řešení v době psaní této práce. V podkapitole s názvem Existující řešení je postupně popíši s jejich výhodami a nevýhodami. Dále v této kapitole provedu analýzu bezdrátových modulů pro komunikaci mezi jednotlivými senzory, jednodeskových počítačů pro implementaci centrální řídící jednotky a senzorů, bezkontaktních technologií pro autentizaci uživatelů a softwaru pro implementaci řešení.

1.1 Existující řešení

V rámci této kapitoly porovnávám tři existující komerční řešení a tři existující open source řešení. V rámci komerčních řešení jsem porovnal Samsung SmartThings [10], který vyhrál porovnání na webu DarwisDen, který se zabývá tématikou smart home. Dále jsem k porovnání vybral Belkin WeMo [11], který získal ocenění pro nízkou cenu. Jako posledního zástupce jsem vybral systém Ubiquiti mFi z důvodu osobní zkušenosti.

Pro testování mezi open source projekty jsem vybral projekty OpenHab, Domoticz a Home Assistant, kteří se umístili mezi prvními třemi v porovnání na webu HomeToys [12], který se zabývá tématikou smart home. Dále také mají tyto tři systémy velmi pozitivní hodnocení v rámci komunity.

1.1.1 Komerční řešení

1.1.1.1 Samsung SmartThings

SmartThings Inc. je společnost, která byla založena v roce 2012 se sídlem ve městě Palo Alto v Kalifornii ve Spojených státech amerických. Společnost vznikla díky nápadu jednoho ze zakladatelů, který po vytopení sklepa tušil, že pokud by měl nějaký detektor úniku vody, tak mohl tak velké škodě zabránit. Vymyslel tedy projekt SmartThings, který na serveru Kickstarter

zaujal mnoho lidí a bylo vybráno 1,2 milionu amerických dolarů. V roce 2014 proběhla akvizice společností Samsung [13].

Samsung SmartThings [14] se skládá se základní řídící jednotky nazvané jednoduše Hub, která slouží jako základní řídící jednotka, a z řady senzorů a zařízení, které lze pomocí hubu ovládat. Do systému je možné připojit i zařízení od jiných výrobců. Veškerá zařízení jsou ovládaná pomocí nativní aplikace SmartThings App. Systém podporuje řadu komunikačních protokolů jako je ZigBee, Z-Wave a zařízení dostupná pomocí IP protokolu. Dále systém podporuje řadu protokolů jako je například systém IFTTT, který dovoluje vykonat akci na základě nějaké jiné akce v nějaké webové službě [15]. SmartThings poskytuje API pro vývojáře a tedy je možné vytvořit vlastní aplikace a systémy za využití akci dostupných pomocí SmartThings Hubu. Jinak je systém uzavřený a vývojáři jsou limitování jeho funkcionalitami a podporovanými komunikačními protokoly. Maximální dosah bezdrátové komunikace je přibližně 40 metrů v závislosti na vnitřní konstrukci budovy.

Bohužel v tuto chvíli není možné se systémem dělat složitější věci a uživatel je tedy nucen počkat, dokud výrobce nepřijde s dalšími možnostmi jak se systémem pracovat [15]. Další záporná vlastnost celého řešení je jeho cena. Hub samotný stojí 99 amerických dolarů, senzor teploty a vlhkosti stojí 45 dolarů, přepínatelná zásuvka 54,99 dolarů, senzor pohybu 39,99 dolarů [16]. V neposlední řade se mezi záporné vlastnosti řadí to, že systém není možné v době psaní této práce koupit v oficiální distribuci v České republice.

1.1.1.2 Belkin WeMo

Belkin WeMo je systém pro domácí automatizaci, kterou vyrábí společnost Belkin International se sídlem ve městě Playa Vista v Kalifornii ve Spojených státech amerických.

Zařízení Belkin WeMo [17] nevyžadují pro svou funkčnost nic jiného než funkční WiFi připojení, veškeré nastavení a automatizace se děje skrze zdarma poskytovanému řešení v cloudu. Belkin nabízí automatizované řešení, které poskytuje velké množství ohledně prostředí – například WeMo Insight switch, který poskytuje kromě automatického vypínání a zapínání spotřebičů také podrobné statistiky ohledně spotřeby elektrické energie, doby, po kterou byl spotřebič v provozu, a ceny jeho provozu. Celý systém je možné ovládat pomocí jednoduché a přehledné mobilní aplikace. Pro vývojáře je dostupné API skrze cloudové služby [18].

Na druhou stranu považuji nutnost existence WiFi připojení pro funkčnost řešení za velký nedostatek celého systému, jelikož tam, kde nebude dostupné WiFi připojení nebude možné toto řešení vůbec využít.

Mezi další omezení patří slabší podpora zabezpečení objektu. Do systému je možné připojit jen jeden detektor pohybu a celé možnosti řešení zabezpečení jsou velmi omezené. Uživatelé se nakonec uchylují k napojení WeMo zařízení k Samsung SmartThings Hubu [11], jelikož v době psaní této práce žádné

jiné zařízení WeMo produkty nepodporovalo. Velmi limitující je také absence lokálního řešení, které by nahradilo kontrolér v cloudu [18].

Cena WeMo zařízení se řadí mezi levnější produkty. Chytrá přepínatelná zásuvka (WeMo Insight switch) stojí 49,99 amerických dolarů, obyčejná přepínatelná zásuvka (WeMo switch) bez statistik stoji 39,99 dolarů [19]. Samostatný teplotní senzor bohužel není dostupný (senzor teploty je ve všech přepínatelných zásuvkách) a je nutné využít nějaký, který lze připojit pomocí IFTTT – např. Wireless Tag.

WeMo zařízení jsou tedy vhodné pro instalace, které nebudou nijak komplexní a bude postačovat jednoduchá a levná automatizace. Pro větší instalace je nutné systém propojit s nějakým komplexnějším řešením jako je Samsung SmartThings.

1.1.1.3 Ubiquiti mFi

Ubiquiti Networks je firma se sídlem ve městě San Jose v Kalifornii ve Spojených státech amerických. Firma Ubiquiti se zaměřuje na tvorbu drátových i bezdrátových síťových prvků a systému domácí automatizace. Tato platforma má název Ubiquiti mFi. Platforma se dělí na tři produktové řady – mPort, mPower a InWall Outlet and Switch [20].

Řada mPort slouží pro interakci se senzory. Bohužel velká nevýhoda této řady je nutnost připojit senzory pomocí kabelu s RJ45 konektorem nebo kabelem pro sériovou linku.

Řada mPower slouží pro interakci se zásuvkami se silovou elektřinou. Jsou k dispozici ve variantách s jednou, se třemi a s osmi zásuvkami, které lze nezávisle ovládat a monitorovat.

Řada InWall Outlet and Switch slouží pro interakci se zařízeními, které se zabudují přímo do zdi objektu. V době psaní této práce existovala dvě různé zařízení. mFi Outlet, který slouží jako chytrá zásuvka, která umožňuje monitoring spotřeby a ovládání elektrické energie pro zařízení. mFi Switch je chytrý přepínač a stmívač, který je možné také ovládat pomocí kontroléru.

Produkty ze všech tří řad je pak možné ovládat pomocí kontroléru mFi, který je napsaný v programovacím jazyku Java a je tedy kompatibilní s velkým množstvím platforem. Zajímavostí je, že všechny zařízení z řad mFi jsou postaveny nad linuxovým jádrem a je možné se pomocí protokolu SSH na ně připojit [18] a pro vývojáře je dostupné HTTP API, které je velmi dobře zdokumentované.

Jelikož nelze dohledat doporučené ceny na stránkách výrobce, tak jsem pro porovnání vyšel z ceníku jednoho z oficiálních distributorů [21]. Cena přepínatelné zásuvky je 59 amerických dolarů, senzoru teploty 24 dolarů a pohybového čidla 30 dolarů.

Mezi hlavní nevýhody tohoto systému je jeho uzavřenost a nutná instalace kabelů mezi senzory, což je pro mnoho uživatelů nepřijatelné kvůli nutnosti stavebních úprav v objektech.

1.1.2 Open source řešení

1.1.2.1 Home Assistant

Home Assistant [22] je projekt, který byl poprvé uveden v roce 2014. Jako implementační platforma byl použit programovací jazyk Python 3. Provoz je možný i na Raspberry Pi. Jsou podporovány operační systémy Windows, Linux a OS X. K dispozici je i kontejner pro Docker. Instalace probíhá jednoduše pomocí příkazu pip3 [23].

Konfigurace probíhá pomocí konfiguračních souborů, které jsou sestaveny pomocí YAML formátu. Při prvním spuštění se pokusí Home Assistant zjistit všechny potřebné informace a vygenerovat výchozí konfiguraci a pokusí se o úvodní detekci zařízení. Home Assistant podporuje množství zařízení a protokolů, jejich kompletní seznam je uveden ve velice přehledném a kategorizovaném seznamu na webu projektu [24]. Mezi další velice zajímavé funkcionality patří nativní detekce přítomnosti uživatelů, která funguje pomocí napojení na podporované směrovače, díky kterým systém zjistí, že některý z uživatelů je přítomen v objektu. Druhá možnost sledování polohy uživatelů je pomocí aplikace v jejich chytrých telefonech. Veškerá zařízení je možné vložit do skupin.

Automatizace je provedena pomocí konfiguračního souboru, který je napsán pomocí YAML formátu. Možnosti automatizace jsou rozsáhlé, lze nastavit podmínky včetně pozice slunce na obloze či pozice uživatele kdekoliv na světě [23].

Pro vývojáře je připravena a udržována rozsáhlá dokumentace včetně postupu jak do systému připravit modul pro interakci se zařízením, které ještě nemá v systému podporu. Pro vývojáře je taktéž dostupné RESTful API a Python API [25]. Tvorba vlastního uživatelského rozhraní je možná skrze využití služeb API.

Mezi hlavní nevýhodu tohoto řešení bych řadil nutnost konfigurace pomocí souborů v YAML formátu, což není přívětivé pro běžné uživatele. Jinak bych vybral toto řešení jako nejlepší z tohoto výběru open source systémů pro domácí automatizaci.

1.1.2.2 OpenHAB

OpenHAB [26] je projekt pro integraci jiných řešení a technologií pro domácí automatizaci do jednoho řešení, které dovoluje jejich propojení a řízení. OpenHAB byl navržen jako systém, který není závislý na žádném výrobci. Je napsán v Javě a je tedy multiplatformní a může běžet na libovolném zařízení, které podporuje Java Virtual Machine.

Systém tedy dovoluje kombinaci řešení jiných výrobců dohromady a dovoluje je rozšířit o vlastnosti jiných systémů či vlastních řešení. Hlavní princip fungování je v existenci takzvaného Event busu, který slouží pro komunikaci mezi jednotlivými zařízeními. Zařízení jsou evidována v OpenHAB repository, který dovoluje ukládat jejich stav, pokud je to nutné a nestačí pouze bezsta-

vová komunikace. Každé zařízení je spojeno s Event busem pomocí binding modulu, který obstarává komunikaci s jednotlivými připojenými zařízeními. Automatizace se provádí pomocí pravidel. Pravidlo spojuje akce, které se mají stát, s akcemi, které se staly, jako je například zapnutí světel ve 22:00.

Implementace již obsahuje velké množství binding pluginů, které dovolí komunikovat pomocí různých zařízení – například pomocí MQTT protokolu. Vývoj dalších modulů je možný, ale vývojářská dokumentace je velice strohá a je nutné procházet existující zdrojové kódy pro získání lepšího přehledu.

Pro zobrazování dat ze senzorů a ovládání zařízení je možné vybrat si z několik uživatelských rozhraní, které se hodí pro zařízení na kterém je rozhraní zobrazováno [27]. Možnost vlastního rozhraní je například pomocí tvorby vlastního bindingu, což hodnotím jako ne úplně čisté řešení [28].

Mezi hlavní nevýhody patří nutnost běhu na zařízení, které je schopno bez problému provozovat aplikace postavené na Javě. Je sice možné systém provozovat na platformě Raspberry Pi, ale každý restart systému trvá neúměrně dlouho. Dále díky velice komplexnímu systému konfigurace pomocí textových souborů je složité se naučit rychle se systémem pracovat [29].

1.1.2.3 Domoticz

Domoticz [28] je systém, který dovoluje ovládat a monitorovat množství zařízení a senzorů. Lze nainstalovat na mnoho operačních systémů včetně Raspberry Pi, Windows a OS X. Jako uživatelské rozhraní je připraven škálovatelný webový frontend v HTML5, který je responzivní a tedy je možné ho zobrazit na zařízeních o různém rozlišení obrazovky bez snížení uživatelského komfortu včetně mobilních zařízení. Projekt byl poprvé uveden v roce 2010.

Automatizace je možná pomocí skriptů v jazyku Lua, které mají syntaxi podobnou programovacímu jazyku Python. Nutnost vytvářet automatizaci pomocí skriptu hodnotím jako velice uživatelsky nepřívětivé a pro běžného uživatele velice složité. A jiný způsob se mi nepodařilo nalézt. V době psaní této práce nebyla sekce na webu projektu, která se věnovala automatizaci, dostupná [30].

Domoticz lze konfigurovat pomocí webové aplikace včetně připojení různých zařízení, které ale bohužel musí být podporované. Seznam podporovaných zařízení lze získat na webu projektu a v jeho manuálu [31]. V systému je možné nastavit notifikace, dobu ukládání historie, nastavení vzdáleného sdílení, nastavení metrik a mnoho dalšího.

Domoticz hodnotím jako nadějný projekt, kterému chybí lepší dokumentace pro vývojáře a nástroje pro zjednodušení automatizace pro běžné uživatele. Mezi dalšími věcmi, které hodnotím jako limitující, patří nutnost využívat jedno uživatelské rozhraní bez vývojářské dokumentace jak napsat vlastní v jiném programovacím jazyce. Přesto jsou zde projekty (například projekt Domoticz-Home-UI), které se o toto snaží [32].

Modul	Cena (USD)
Xbee S2 (ZigBee)	26,95
HM-10 (Bluetooth)	10,53
XD-FST + XD-RF-5V	3,35
nRF24l01+	1,95

Tabulka 1.1: Ceny jednotlivých modulů dle portálu Amazon

1.2 Analýza bezdrátových modulů

Pro analýzu bezdrátové komunikace jsem vybíral řešení, které dovolují komunikovat mezi jednotlivými moduly autonomně – tedy aby neexistovala závislost na žádné síťové infrastruktuře. Dále je nutné, aby byly k dispozici knihovny pro programovací a skriptovací jazyky, které umožňují provozovat tyto řešení na jednodeskových počítačích bez nutnosti větších zásahů do systému dané platformy. Pro analýzu jsem tedy vybral zařízení založené na standardech ZigBee a Bluetooth. Dále jsem vybral moduly XD-FST + XD-RF-5V a nRF24l01+.

Tabulka 1.1 porovnává ceny jednotlivých modulů. Ceny jsou převzaty z internetového portálu Amazon (http://www.amazon.com/).

1.2.1 ZigBee

ZigBee [33] je technologie, která je založena na specifikaci IEEE 802.15.4. Tento standard byl ratifikován Institutem pro elektrotechnické a elektronické inženýrství teprve v roce 2003. Standard specifikuje protokol pro komunikaci založenou na paketech a je určen pro energeticky úsporná zařízení, která jsou napájená z baterií. Dalším s aspektů, na které tento standard cíli, jsou nízké pořizovací náklady.

Standard ZigBee je určen do prostředí, která obsahují velké množství různých elektromagnetických vysílačů, a jsou tedy silně zarušená. Taková prostředí jsou dnes běžná v komerčních objektech.

Mezi hlavní přednosti standardu ZigBee patří podpora několika síťových topologií včetně point-to-point a sítě typu mesh, které dovolují komunikaci modulů skrze ostatní moduly v síti. Díky této vlastnosti je možná komunikace mezi moduly, i když mezi nimi není přímé spojení a je tedy možné zvýšit komunikační dosah. Komunikace je plně zabezpečena pomocí symetrické šifry AES s velikostí klíče 128 bitů. V takto vytvořené síti může existovat až 65 000 komunikačních uzlů.

Bohužel i přes snahu standardu o minimalizaci ceny jsou komunikační moduly pro jednodeskové počítače v porovnání s jinými řešeními několikanásobně dražší (tabulka 1.1), což nepříjemně zvyšuje cenu nutnou investovat do domácí automatizace.

1.2.2 Bluetooth

Bluetooth [34] je technologie, která má počátky v roce 1994, kdy si firma Ericsson nechala vypracovat studii realizovatelnosti náhrady kabelového propojení dvou zařízení bezdrátovou technologií. Následně v roce 1998 byla pěti předními firmami (IBM, Toshiba, Intel, Ericsson a Nokia) založena skupina Bluetooth Special Interest Group. Tato skupina měla za úkol vytvořit univerzální standard pro osobní bezdrátovou počítačovou síť. V roce 1999 byla uvedena první specifikace 1.0a. Po uvedené této specifikace se začala skupina rozrůstat, kde dnes má tato skupina více než 10 000 členů.

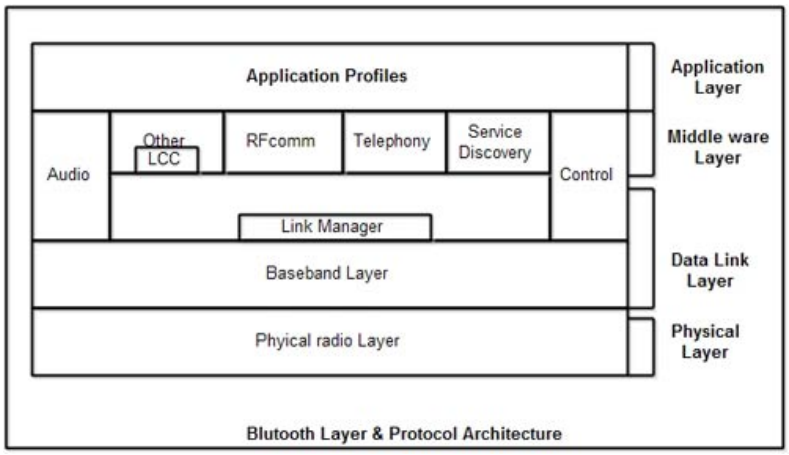
Postupem času byly vydávány nové verze této specifikace. Verze 1.2 představovala úplné přepracování původní specifikace. Verze 2.0 přinesla rozšíření Enhanced Data Rate, které dovolilo přenášet data až do rychlosti 2,2 Mbit/s. Verze 3.0 přinesla specifikaci Ultra Wide Band, která teoreticky umožňuje komunikovat rychlostí až 480 Mbit/s, reálně je ale uváděna možnost do 24 Mbit/s [35]. Specifikace Bluetooth ve verzi 4 ponechala přenosovou rychlost na 24 Mbit/s, ale rozšířila dosah komunikace až na 100 metrů při snížení spotřeby elektrické energie. Zároveň byla uvedena podpora symetrické šifry AES s velikostí klíče 128 bitů.

Specifikace Bluetooth definuje několik vrstev protokolu. Fyzická vrstva specifikuje metody komunikace na bezlicenční frekvenci 2,4 GHz pomocí metody kmitočtových skoků. Linková vrstva se zabývá popisem topologie sítě, principy adresování a přístupu do sítě. Vrstva je rozdělena na dvě podvrstvy. První podvrstva se nazývá Link Management Protocol a má na starosti správu sítě včetně navazování spojení, ověřování a nastavení úsporného režimu. Druhá podvrstva s názvem Logical Link Control and Adaptation Protocol, která propojuje protokoly vyšších vrstev s operacemi prováděných na vrstvě baseband a má na starosti hlavně datové přenosy.

Na obrázku 1.1 je možné vidět zjednodušenou architekturu technologie Bluetooth.

Síť Bluetooth je postavena na principu master – slave. Master je zařízení, které řídí tok dat, a slave je pasivní zařízení čekající až mu master umožní s ním komunikovat. Jeden master může připojit až sedm slave zařízení. Takto vytvořená síť se nazývá piconet. Dále je možné propojit několik piconet sítí čímž vznikne scatternet. Ve scatternetu je více zařízení typu master, kde zařízení je vždy master pouze pro svůj piconet. Zařízení, které je master ve svém piconetu může být slave v piconetu jiném.

V době psaní této práce nebylo dostupných mnoho modulů podporujících standard Bluetooth 4.0 a mají tedy teoretický dosah 100 metrů. Jedním z těchto modulů je HM-10. Cena tohoto modulu je ale v rámci porovnání v tabulce 1.1 stále vysoká oproti jiným dostupným modulům. Další nevýhodou tohoto řešení je nutnost párování zařízení, což je pro připojení jednoduchých senzorů nevhodné [36].



Obrázek 1.1: Architektura Bluetooth (převzato z [1])

1.2.3 XD-FST + XD-RF-5V

Jedná se vlastně o dvojici modulů, které fungují na frekvenci 433,92 MHz. Modul XD-FST slouží jako vysílač a modul XD-RF-5V slouží jako přijímač. Dosah komunikace pomocí těchto modulů je přibližně 50 metrů [37].

Bohužel tyto moduly nenabízejí žádné pokročilé metody pro komunikaci. Jemné ladění (neboli výběr kanálu) je nutné provádět pomocí malého potenciometru na modulech. Jednotlivé moduly není možné nijak adresovat. K dispozici není ani žádná možnost ověření doručení dat. Jedinou výhodou těchto modulů je možnost komunikace s běžnými ovladači domácích zařízení, které operují taktéž na frekvenci 433,92 MHz [38].

Tyto moduly mají sice velmi příznivou cenu, ale nenabízí žádné pokročilé funkcionality, které by zaručovaly spolehlivost komunikace v rámci domácí automatizace. Pro obousměrnou komunikaci je nutné k jednomu jednodeskovému počítači připojit přijímač a vysílač zároveň, což zvyšuje nároky na energii a prostor.

1.2.4 nRF24l01+

Modul nRF24l01+ [39] je vylepšená verze modulu nRF24l01 od výrobce Nordic Semiconductor. Jedná se o nízkoodběrový digitální přijímač a vysílač, který komunikuje v rámci bezlicenčního pásma 2,4 GHz s možností volby jednoho ze 126 dostupných kanálů. Pro funkčnost je potřeba k modulu přidat pouze 16 MHz krystal, obvod a anténu. Většina prodávaných zařízení s tímto modulem toto vše již obsahuje [40].

Pro komunikaci je možné vybrat jednu z dostupných rychlostí – 2 Mbps, 1 Mbps a 250 kbps. Moduly jsou navrženy pro úsporný provoz a je možné zvolit mezi několika energetickými profily, které ovlivňují dosah komunikace.

Dosah nRF24l01+ se pohybuje okolo 100 metrů na volném prostranství. V zástavbě se dosah snižuje podle použitých materiálů v budově. Existují i speciální verze obvodů, které ve svých specifikacích tvrdí, že zvyšují dosah teoreticky až na 1 km [41]. Bohužel se mi nepodařilo najít zdroj, který by potvrdil, že je takový dosah reálný. Další aspekt, který ovlivňuje dosah, je volba rychlosti komunikace, kde volba 250 kbps může zvýšit dosah až trojnásobně [42].

Modul má odběr 11.3 mA při vysílacím výkonu 1 mW a v módu standby-I je odběr pouze 26 µA.

Komunikace probíhá pomocí tzv. pipe. Pipe je virtuální komunikační kanál, který má svou adresu ve formátu pěti hexadecimálních číslic, kde každá je v rozsahu 00 až FF. Validní adresa tedy může například být F0:F0:E1:E1:F0. Každý modul může zároveň poslouchat až na šesti různých adresách. Vysílat může pouze pomocí jedné pipy. Jedna zpráva může obsahovat maximálně 32 bytů dat. Pokud je zpráva delší, je nutné implementovat logiku pro její rozdělení v samotné aplikaci, která tento modul pro komunikaci využívá. Technologie linkové vrstvy s názvem Enhanced ShockBurstTM zajišťuje potvrzení doručení zpráv pomocí ACK paketů, kde je pro kontrolu paketu využito mechanismu Cyclic Redundancy Check (CRC).

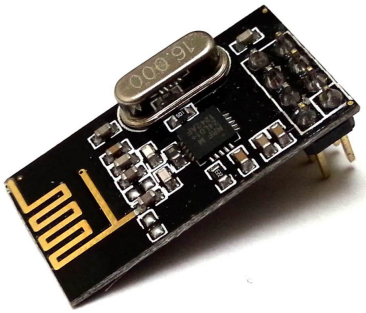
Pro modul nRF24l01+ existuje mnoho knihoven pro jeho jednoduché použití v rámci různých programovacích jazyků včetně jazyků skriptovacích. Například projekt TMRh20 [43] se soustředí na implementaci různých protokolů pro tyto moduly – RF24Ethernet implementuje protokol TCP/IP a RF24Mesh implementuje podporu sítě typu mesh.

Mezi nevýhody tohoto řešení je omezení velikosti jednoho paketu na 32 bytů a absence nativního zabezpečení komunikace. Tyto negativa se dají vyřešit vhodně navrženým komunikačním protokolem či v rámci aplikace, která bude ke komunikaci toto řešení využívat. Vzhledem k ostatním vlastnostem a ceně těchto modulů hodnotím toto řešení jako nejlepší volbu pro implementaci výsledného řešení.

Na Obrázku 1.2 je ukázán modul nRF24l01+.

1.3 Analýza jednodeskových počítačů

Jednodeskové počítače [44] usnadňují lidem prototypování projektů, které potřebují nějaký výpočetní výkon pro automatizaci. Dnešní moderní jednodeskové počítače disponují dostatečným výkonem pro běh plnohodnotného operačního systému, který dovoluje rozšířit projekty o mnoho dalších funkcionalit pomocí jednoduché instalace software. Mimo prototypování mají tyto jednodeskové počítače širokou škálu dalších využití – od domácího multimediálního centra až po poskytování síťových služeb (např. Network Area Storage a přidělování adres pomocí protokolu DHCP).



Obrázek 1.2: Modul nRF24l01+ (převzato z [2])

Jednodeskové počítače využívají speciálního integrovaného obvodu, který se nazývá system on a chip (SoC). Tento obvod v sobě obsahuje procesor, operační paměť, grafický čip a další komponenty, které jsou potřeba pro provoz počítače. Zároveň má celý tento systém menší spotřebu elektrické energie než běžné počítače.

Kromě výše zmíněného integrovaného obvodu můžeme najít v rámci jednodeskového počítače různé druhy vstupních a výstupních rozhraní jako je například USB či HDMI. Na novějších jednodeskových počítačích již bývají integrovány čipy pro podporu bezdrátové komunikace pomocí standardu WiFi či Bluetooth.

Pro tuto diplomovou práci jsem vybíral jednodeskový počítač, který by mohl vykonávat činnost kontroléru celého řešení. Počítač tedy musí umožňovat přímé připojení bezdrátového modulu nRF24l01+, musí obsahovat alespoň jeden USB port pro připojení zařízení, pomocí kterého bude prováděna bezkontaktní autentizace uživatele. Dále je nutné mít možnost lokálně ukládat veškerá naměřená data a poté je vhodně prezentovat uživateli pomocí připojitelného displeje. Mezi hlavní kritéria výběru také patří existence možnosti připojení k počítačové síti pro vzdálený monitoring pomocí protokolů HTTP a SSH jak je požadováno v rámci zadání.

Na základě těchto požadavků jsem pro porovnání vybral jednodeskové počítače Intel Edison a Raspberry Pi, které tyto požadavky splňují.

Pro provoz senzorů není nutný vysoký výkon, není nutné připojení k počítačové síti a není potřeba žádný USB port. U senzorů je priorita nízká spotřeba elektrické energie a možnost připojit modul nRF24l01+ pro bezdrátovou komunikaci s kontrolérem.

1.3.1 Intel Edison

Intel Edison [45] je malý jednodeskový počítač ve velikosti podobné paměťové kartě typu Secure Digital. Edison byl uveden v roce 2014. Počítač je

vybaven procesorem Intel Atom s úspornou architekturou Silvermont s taktem 500 MHz, 1 GB operační paměti RAM a 4 GB flash paměti sloužící jako úložiště. Připojení k počítačové síti je možné díky standardu WiFi 802.11 (a/b/g/n) a úsporné verze Bluetooth ve verzi 4.1. Připojení senzorů a různých modulů je možné pomocí čtyřicet general-purpose input/output pinů, nebo pomocí rozhraní SPI, I2C či ADC.

Samotná malá deska ale neumožňuje pohodlně systém používat, jelikož nejsou dostupné klasické konektory pro připojení vstupních a výstupních zařízení. Proto jsou k počítači Intel Edison k dispozici rozšiřující základní desky, které již veškeré požadované rozhraní obsahují. V době psaní této práce společnost Intel nabízí dva modely základní desky – Intel Edison Kit with Breakout Board a Intel Edison Kit for Arduino.

Intel Edison Kit with Breakout Board je jen o něco větší než samotný počítač a poskytuje přístup k nativním 1.8 V I/O portům modulu Edison, USB OTG s micro USB, převodník UART na USB.

Druhý dostupný modul Intel Edison Kit for Arduino je několikanásobně větší než samotný počítač Intel Edison. Tento modul dovoluje připojit rozšiřující moduly pro platformu Arduino, dále obsahuje 20 general-purpose input/output pinů včetně 4 pinů podporující PWM výstup, který se používá pro dosažení analogového chování pomocí digitálního řízení [46].

Na počítači běží speciální linuxová distribuce Yocto Linux, která slouží především pro IoT elektroniku [47]. Intel Edison ale disponuje dostatečným výkonem a je tedy možné na něj nahrát i jiné linuxové distribuce jako je například Debian. Oficiální podporované programovací jazyky v rámci distribuce Yocto Linux jsou JavaScript (pomocí prostředí Node.js), C, C++ a Python.

Intel Edison bohužel má slabší podporu vstupních a výstupních zařízení v základu. Neobsahuje ani žádné rozhraní pro video výstup, chybí klasické USB a pro pohodlné používání je nutné zakoupit rozšiřující desku, což zvyšuje náklady pro implementaci řešení.

1.3.2 Raspberry Pi

Raspberry Pi [48] je jednodeskový počítač o velikosti kreditní karty, který je vyráběn nadací Raspberry Pi Foundation. Tato nadace je zaregistrovaná jako charitativní organizace se sídlem ve Velké Británii. Raspberry Pi byl původně určen jako nízkonákladový počítač pro studenty s cílem zvýšit počítačovou gramotnost. Postupem času projekt získal velkou oblibu komunity, bylo vydáno několik dalších verzí a dnes může tento jednodeskový počítač sloužit jako jednoduchý domácí počítač pro přístup k internetu, běžnou kancelářskou práci či pro sledování filmů.

Poslední model – Raspberry Pi 3 [49] – je postaven na integrovaném obvodu Broadcom BCM2837, který obsahuje čtyřjádrový procesor ARM Cortex-A53, který má pracovní frekvenci 1.2 GHz a podporuje 64 bitové instrukce, grafický

čip Videocore 4, 1 GB operační paměti, integrovanou bezdrátovou síťovou kartu s podporou WiFi 802.11 (a/b/g/n) a Bluetooth 4.1.

Verze 3 obsahuje čtyři USB 2.0 porty, čtyřicet general-purpose input/output pinů včetně SPI a I2C sběrnice, HDMI port, ethernetový port, kombinovaný 3.5mm audio jack, CSI rozhraní pro připojení kamery, DSI rozhraní pro připojení displeje a microSD slot.

Mimo verze 3 existuje zmenšená verze s názvem Raspberry Pi Zero [50], která je primárně určená pro integraci do projektů vyžadující automatizaci. V této verzi je jednojádrový procesor operující na frekvenci 1 GHz, 512 MB operační paměti, mini HDMI, čtyřicet general-purpose input/output pinů včetně SPI a I2C sběrnice a dva micro USB porty – jeden pro napájení a druhý pro připojení externích periferií.

Pro Raspberry Pi je dostupná řada operačních systému. Oficiální operační systém od Raspberry Pi Foundation se jmenuje Raspbian a jedná se distribuci založenou na známé distribuci Debian. Pro využití počítače jako multimediální centrum je možné využít operační systém OPENELEC. Pro Raspberry Pi verze 2 a 3 je možné využít operační systém Windows 10 IoT Core.

Velkou výhodou této platformy je dostupnost široké řady oficiálního příslušenství včetně dotykových displejů pro DSI rozhraní, kamer pro CSI rozhraní a další. Při volbě operačního systému Raspbian je možné využít téměř libovolný hardware s rozhraním USB pro který existují ovladače pro OS Linux. Díky podpoře GPIO přímo v operačním systému je možné komunikovat i s velkým množstvím různých modulů pomocí sběrnic I2C a SPI včetně modulu nRF24l01+.

Jednodeskový počítač Raspberry Pi alespoň ve verzi 2 splňuje veškeré požadavky, které byly uvedeny v kapitole Analýza jednodeskových počítačů 1.3, a proto tuto platformu hodnotím jako nejlepší volbu pro realizaci výsledného řešení.

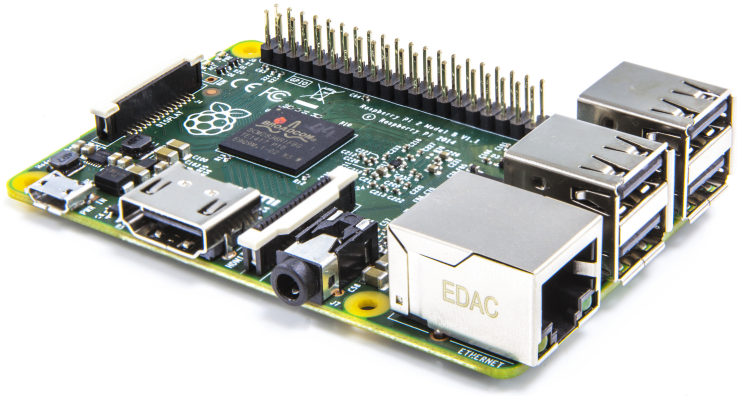
Na Obrázku 1.3 je ukázán jednodeskový počítač Raspberry Pi 2 Model B.

1.3.3 Arduino

Arduino [51] je open source prototypovací platforma založená na mikroprocesoru ATMega [52], která slouží hlavně pro interakci se senzory a aktivními prvky, které dokáže ovládat. Projekt vznikl v Itálii jako levný nástroj pro podporu výuky pro technologické studenty. Veškerý hardware a software je open source a je tedy možné si desku nechat vyrobit i mimo oficiální distribuci.

Programování pro počítač probíhá za pomocí programovacího jazyku Wiring, který je vlastně jen knihovnou do C++ [53]. Samotné programování počítače – tedy nahrání programu v jazyku Wiring – probíhá pomocí USB kabelu a programu Arduino IDE, který je dostupný pro platformy Windows, OS X a Linux.

V době psaní této práce již existovalo přes deset různých modelů desek [54], kde každý měl svoje zařazení. Od opravdu malých desek jako je Arduino



Obrázek 1.3: Jednodeskový počítač Raspberry Pi 2 Model B (převzato z [3])

Gemma pro nositelnou elektroniku až po rozšířené Arduino Mega s rozšířeným počtem general-purpose input/output pinů, 256 KB paměti a mikrokontrolérem ATmega2560 [55].

Model Uno je nejrozšířenější deskou Arduino platformy vůbec [56]. Deska je založena na mikrokontroléru ATmega328P s 16 MHz krystalem, USB konektorem, 14 digitálními vstupně – výstupními piny, kde šest z nich lze využít jako PWM výstup a šesti analogovými piny. Arduino Uno má k dispozici 32 KB flash paměti a 2 KB operační paměti SRAM. Deska samotná má odběr 46.5 mA. Režim spánku sníží tuto spotřebu na 34.4 mA [57].

Mimo samotných desek existuje velké množství rozšiřujících modulů. Tyto rozšiřující moduly jsou pojmenovány „shields“. Rozšiřující moduly se jednoduše nasadí na desku Arduino a tím okamžitě rozšíří funkcionalitu celého řešení. Mezi oficiální shiledy se řadí například Xbee shield (modul pro komunikaci pomocí standardu ZigBee), Motor Control shield pro kontrolu krokových motorků, Ethernet shield pro připojení Arduino desky k počítačové síti [58] a mnohé další.

Pro tuto diplomovou práci jsem vybral desku Arduino Uno, která splňuje veškeré požadavky specifikované v kapitole Analýza jednodeskových počítačů 1.3.

1.4 Analýza bezkontaktních technologií

Pro pohodlnou autentizaci uživatelů při vstupu do objektů se dnes v mnoha budovách využívá bezkontaktních technologií. Jedná se o variantu bezdrátové komunikace na blízkou vzdálenost do několika málo centimetrů. Díky této technologii uživatel pouze přiloží svůj čip ke čtečce na blízkou vzdálenost a čtečka z něj přečte potřebná data na základě kterých je uživatel ověřen. Bo-

hužel je nutné udržovat přístupový čip v bezpečí a v případě jeho odcizení či ztráty ho okamžitě zablokovat. Druhou možností je postupovat dle doporučení normy ISO 27002 [59], která doporučuje, aby kromě bezkontaktní technologie musel uživatel pro svou identifikaci zadat několikamístný kód PIN.

Pro porovnání jsem vybral technologie RFID a NFC, které jsou dnes nejčastěji používané pro bezkontaktní komunikaci na krátkou vzdálenost.

1.4.1 RFID

Radio Frequency Identification [60] neboli RFID je systém pro identifikaci pomocí radiofrekvenční vln. Systém vznikl původně jako systém pro náhradu čárového kódu a za jeho vývojem stála maloobchodní firma WallMart. Cílem bylo vyvinout technologii, která by umožňovala identifikaci objektu v reálném čase bez jeho přímé viditelnosti. Tato technologie se velice rychle ujala a dnes slouží v mnoha odvětvích včetně logistiky, sledování majetku, evidenci osob a další.

Informace o objektu jsou zaznamenány na datový nosič – RFID tag, který se připevní na sledované objekty. RFID tag obsahuje malý čip s anténou a pamětí. Data uložená v paměti tohoto čipu je možné přečíst pomocí elektromagnetických vln. Pomocí vyzářených vln z čtecího čipu dojde k nabití čipu RFID tagu, který následně pošle zpět informace uložené v jeho paměti. Každý tag obsahuje své unikátní číslo – electronic product code, což je sériové číslo tagu.

RFID tagy se dělí na aktivní a pasivní. Aktivní tagy obsahují vlastní baterii a vysílají informace do okolí. Dosah těchto tagů je až 100 metrů. Aktivní tagy vyžadují vyšší náklady na pořízení. Pasivní tagy mají menší komunikační vzdálenost – od 50 cm do maximálně 10 metrů a dnes patří mezi nejrozšířenější.

Tagy komunikují na několika frekvencích – nejběžnější jsou dvě frekvence 125 kHz (takzvané Low Frequnecy tagy) a 13,56 MHz (takzvané High Frequnecy tagy) [61]. RFID tagy se běžně umísťují do přívěšku na klíče či do karet o velikosti kreditní karty kde slouží jako ověření uživatelské identity a umožňuje uživateli přístup do objektů.

Bohužel dle vlastní zkušenosti lze obyčejné RFID tagy na frekvenci 125 kHz jednoduše okopírovat pomocí levného zařízení (například zařízení s produktovým označením CF-RL129), které lze bez problému zakoupit na internetu. Je tedy možné uživateli odcizit tag aniž by o tom věděl a následně ho využít k neoprávněnému přístupu do objektů či dalších prostor.

1.4.2 NFC

Near Field Communication [62] je novější varianta pro komunikaci pomocí radiofrekvenční vln na krátkou vzdálenost. Komunikace probíhá podobně jako u RFID, ale je omezena na vzdálenost kolem čtyř centimetrů, což je pova-

žováno za jednu z bezpečnostních vlastností této technologie. Komunikace probíhá na frekvenci 13,56 MHz.

Technologie NFC si postupem času získala velkou oblibu a dnes je využívána nejen pro identifikaci osob, ale i pro přenos kontaktů, URL adres a dokonce tuto technologie najdeme i v platebních kartách.

NFC tagy umožňují kromě čtení i zápis. Do tagů můžeme zapsat libovolnou informaci – jsme omezeni pouze kapacitou paměti daného tagu. Data je možné uzamknout nebo dokonce i šifrovat, takže jsou chráněna proti neoprávněnému čtení či manipulaci s nimi. Komunikace s tagy může probíhat ve třech různých režimech. Režim peer-to-peer slouží pro komunikaci dvou zařízení, které si mohou vzájemně vyměňovat informace. Režim reader/writer slouží pouze pro čtení dat z pasivního NFC čipu nebo pro jejich zápis. Poslední režim se nazývá card emulation. Tento režim umožní aktivnímu zařízení se chovat jako zařízení pasivní.

Díky módům peer-to-peer a card emulation se tato technologie dostala i do mobilních telefonů, kde s nimi uživatelé mohou například jednoduše platit, připojit se k WiFi sítím aniž by museli opisovat dlouhá hesla či si vyměňovat kontaktní informace pouhým přiložením telefonů k sobě [61].

Pro tuto diplomovou práci jsem vybral technologii NFC díky možnostem jejího dalšího rozšíření kromě jednoduché identifikace, možnosti šifrování a faktu, že mnoho karet, které se dnes běžně používají – jako například Opencard či ISIC [63] – implementují tuto technologii. Díky tomuto je tedy možné tyto stávající karty využít pro identifikaci uživatele a není nutné mu vydávat další kartu, kterou by musel nosit. Dalším důležitým faktorem je podpora NFC přímo v operačním systému Linux díky projektu NFC Tools [64], který zjednodušuje komunikaci pomocí NFC a jeho knihovny lze využít pro tvorbu vlastního řešení, které NFC využívá.

1.5 Analýza Software pro implementaci

Pro implementaci bylo nutné vybrat programovací jazyk pro kontrolér, pro jednodeskový počítač Arduino a pro řešení zobrazení monitoringu přes protokol HTTP a SSH. V následujících kapitolách provedu diskuzi a výběr nejvhodnějších řešení.

1.5.1 Skriptovací jazyk

Skriptovací jazyk je interpretovaný programovací jazyk a pro svůj běh vyžaduje interpret a nemusí být předem zkompilován. Interpret postupně vyhodnocuje jednotlivé příkazy za běhu. Hlavní výhodou skriptovacích jazyků je jejich snadná integrace s jinými systémy či programovacím a skriptovacími jazyky [65].

Interpret je počítačový program, který umožňuje vykonávat příkazy jiného programu. Díky této vlastnosti není nutné skript převádět do strojového kódu.

Kód je tak přenositelný na jiné platformy, které daný skriptovací jazyk podporují [66]. Z této vlastnosti plyne další výhoda – rychlost jednoduchých úprav bez nutnosti celé řešení opět kompilovat. Dále programátor neřeší správu paměti, což zvyšuje i bezpečnost celého řešení. Správa paměti je starostí interpretu a program samotný není pak náchylný na chyby správy paměti jako je například heap či stack overflow [67].

Nevýhoda skriptovacích jazyků oproti kompilovaným jazykům je jejich rychlost. Program, který je zkompilován, je již přeložen na základní instrukce procesoru a mohou se rovnou vykonat. Interpret musí vyhodnocovat textový zápis příkazů a teprve poté může jednotlivé bloky skriptu vykonávat, což vyžaduje vyšší systémové prostředky. Tuto nevýhodu se ale tvůrci skriptovací jazyků snaží odstranit pomocí kompilace za běhu do takzvaného byte kódu či nativního objektového kódu [65].

Pro implementaci kontroléru a monitoringu pomocí protokolu SSH je potřeba zpracovat respektive zobrazit velké množství textových dat. U implementace kontroléru je nutné propojit více systému v jeden celek. Pro implementaci jsem tedy zvolil skriptovací jazyk Python, který se jeví jako nejlepší volba. Programovací jazyk Python popíši v následující kapitole.

1.5.1.1 Python

Python [67] patří mezi nejmladší skriptovací jazyky. Byl vytvořen v roce 1990 Guidem van Rossumem v institutu Stichting Mathematisch Centrum v Nizozemsku. Python byl navržen jako platforma pro podporu rozsáhlých aplikací se zachováním bezproblémové čitelnosti výsledného kódu. Python patří mezí nejjednodušší programovací jazyky, co se týče potřebnému času k jeho naučení. V základu plně podporuje objektové programování a lze tedy program jednoduše dělit do logických celků, využívat dědičnosti a dalších vlastností objektového programování. Mimo tyto vlastnosti obsahuje Python v základu širokou škálu modulů a rozšíření, které výrazně urychlují vývoj aplikací [68].

Pro svou diplomovou práci jsem zvolil skriptovací jazyk Python ve verzi 3.4 právě kvůli jeho nativní podpoře objektového programování, osobní znalosti jazyka, přehledné syntaxi a dostupnosti modulů pro komunikaci s moduly nRF24l01+ a komunikaci s rozhraními na jednodeskovém počítači Raspberry Pi. Dva nejdůležitější moduly pro tuto část implementace popíši v následující kapitole.

Příklad zdrojového kódu:

```
def receive(self):
        data = ""
        start = time.time()
        if self.radio.available(0):
                receivedMessage = []
                self.radio.read(receivedMessage, self.radio.
                    getDynamicPayloadSize())
                for i in receivedMessage:
                        if (i >= 32 and i <= 126):
                        data += chr(i)
        return data
```

1.5.1.2 Moduly pro komunikaci a práci s daty v jazyku Python

Pro komunikaci s moduly nRF24l01+ bylo nutné využít knihovnu lib_nrf24 [69], která je určena pro jednodeskový počítač Raspberry Pi. Knihovna byla v době psaní této práce ve stavu beta, ale autor tvrdí, že základní vlastnosti jako je komunikace mezi Raspberry Pi a Arduino jsou funkční a otestované, což pro tuto práci plně postačuje.

Pro ukládání dat do databáze byl využit modul peewee [70]. Peewee je malý a efektivní modul pro objektově relační mapování a ukládání takto vytvořených objektů do databáze. Více o tomto je možné se dočíst v kapitole 1.6.3 – ORM. Peewee nativně podporuje databáze SQLite, MySQL a PostgreSQL. Díky využití tohoto modulu je kód, který je využit pro komunikaci s databází čitelný. Další vlastností, které využití peewee přináší, je transakční zpracování vymezených bloků kódu (za podpory transakčního zpracování v rámci zvoleného databázového enginu). V neposlední řade je důležitá možnost bezproblémového využití spojení s databází ve více vláknech či procesech, což je nutné pro modulární zpracování výsledného řešení.

1.5.1.3 JavaScript

JavaScript [71] je interpretovaný skriptovací programovací jazyk, který je primárně určen pro webové stránky, ale existují i mnohé interprety pro využití i mimo webový prohlížeč. JavaScript může být využíván buď jako procedurální či objektově orientovaný jazyk. Klasický JavaScript běží na straně klienta v jeho webovém prohlížeči, kde se pomocí něho dají vytvářet dynamické odezvy na různé akce. Tyto odezvy mohou následně sloužit pro změnu vzhledu webu, odeslání dat na pozadí a mnoho dalších akcí.

JavaScript se dnes používá pro tvorbu interaktivních webových stránek – od jednoduchých efektů až po složité interakce s webovými službami. JavaScript prošel dlouhým vývojem. První verze vyšla v roce 1995 pod jménem Mocha a ještě ten samý rok byl název změněn na LiveScript. Postupem času byl jazyk standardizován organizací ECMA. Standard nese název ECMAScript. JavaScript je tedy implementací tohoto standardu. Mezi další im-

plementace se řadí například ActionScript. V dnešní době je JavaScript velice populární, existuje mnoho interpretů a platforem (například Node.js), které dovolují použít JavaScript i mimo web. Existuje i HTML5 API pro ovládání různých mediálních souborů, tvorbu web socketů a dokonce i propojení se zařízeními jako je akcelerometr [72].

Jeden z největších problému JavaScriptu je to, že různé interprety je můžou interpretovat mírně odlišně a často se stává, že kód, který bez problému funguje v jednom prohlížeči, není funkční v jiném. Proto vzniklo několik JavaScriptových knihoven, které tyto problémy řeší a přináší mnoho dalších funkcí a zjednodušení pro práci s JavaScriptem. Jednou z těchto knihoven je například knihovna jQuery [73].

V rámci této diplomové práce budu využívat JavaScript a knihovnu jQuery pro přidání dynamických reakcí do monitoringu přes protokol HTTP.

Příklad zdrojového kódu:

```
function toggle_relay(device_id){
        $.nette.ajax({
                url: "?do=relayToggle&device_id=" + device_id,
                spinner: true,
                complete: function (payload) {
                        $("#flashes").show("medium");
                        setTimeout(function(){$("#flashes").hide("
                            fast");}, 5000);
                }
        });
}
```

1.5.1.4 PHP

PHP [74] (rekurzivní akronym PHP: Hypertext Preprocessor) je rozšířený open source skriptovací jazyk, který je především určen pro vývoj webových řešení a může být vložen i do HTML. První verze PHP byla uvolněna v roce 1994 Rasmusem Lerdorfem v podobě Common Gateway Interface (CGI) knihoven napsaných v programovacím jazyku C. PHP vzniklo původně jako nástroj pro sledování návštěv na webu s Rasmusovým životopisem a bylo nazváno Personal Home Page Tools. Postupným vývojem se změnil nejen název, ale celý projekt byl kompletně několikrát přepsán do dnešní podoby, kde v roce 2015 byla vydána poslední verze PHP 7.

Pro provedení PHP kódu se většinou využívá webový server, ale je možné využít i interpret v příkazové řádce. PHP je možné využít na mnoha platformách včetně Linuxu, Windows a Mac OS X. Podpora pro PHP je dnes integrována do mnoha webových serverů jako je Apache, IIS. Mimo nativní podpory lze PHP využít díky podpoře FastCGI, kde toto využívá například webový server nginx.

PHP je dynamicky typovaný jazyk s plnou podporou procedurálního programování i objektově orientovaného programování. PHP má nativní podporu

mnoha databází, lze využít objektové rozhraní PDO (PHP Data Objects), které nám dovoluje abstrahovat od reálně použité databáze pro ukládání dat.

PHP má mnoho funkcí pro zpracování textu včetně nativního zpracování formátu JSON, XML a dalších. Mimo nativních rozšíření existuje mnoho dalších, které lze použít pro rozšíření funkcionalit jazyka.

Pro svou diplomovou práci jsem vybral PHP ve verzi 5.6, která je běžně dostupná na operačním systému Raspbian pro jednodeskový počítač Raspberry Pi. PHP je využito pro přípravu dat pro monitoring pomocí protokolu HTTP. Pro rozšíření funkcí PHP jsem využil Nette framework.

Příklad zdrojového kódu:

```
public function handleChangelog()
{
        $this->template->windowTitle = "Changelog";
        $this->template->windowContent = $this->database->table('
            web_interface')->where('name', 'changelog')->fetch()->
            value;
        if ($this->isAjax()) {
                $this->redrawControl('overlayWindow');
        }
}
```

1.5.2 HTML

HyperText Markup Language [75] je značkovací jazyk, který se využívá pro tvorbu webových stránek a webových aplikací. HTML kód se zapisuje pomocí HTML elementů (text uzavřený v úhlových závorkách <>), kde syntaxe je podobná formátu XML. V době psaní této práce byla nejaktuálnější verze jazyka HTML 5, která vyšla ve své finální specifikaci na konci roku 2014. Aktuálně probíhá příprava další verze tohoto standardu s označením HTML 5.1 [76].

Jazyk vznikl v roce 1990 v organizaci CERN za účelem usnadnění sdílení dokumentů a následně byl jazyk využit pro vznik prvních webových stránek. Postupem času vyšlo několik verzí a jazyk samotný byl na určitou dobu nahrazen jiným – XHTML, který využíval syntaxi jazyka XML [66]. Po odmlce se opět začalo pracovat na rozšiřování jazyka až do dnešní podoby HTML5.

HTML využívá dva druhy značek – párovou a nepárovou [75]. Párová značka obsahuje otevírací a zavírací značku, kde zavírací se liší od té otevírací v přítomnosti lomítka před názvem značky. Pro lepší představu uvádím příklad párové značky, která označuje odstavec:

```
<p>Odstavec</p>
```

Mimo párových značek existují i značky nepárové, které nemají uzavírací značku. V HTML 5 lze tuto značku psát stejně jako v HTML 4.01 i XHTML. Tedy před uzavírací úhlovou závorkou je znak lomítka volitelný. Jako příklad uvedu značku pro odřádkování:

```
<br />
```

Vývoj a správa standardu spadá pod konsorcium W3C, které spravuje i další standardy pro World Wide Web.

V rámci této diplomové práce budu využívat HTML5 pro zobrazení monitoringu pomocí protokolu HTTP jelikož se jedná o aktuálně o nejnovější standard, díky definici mnoha nových tagů lze psát čistší a lépe udržovatelný kód. HTML5 obsahuje i mnoho dalších zajímavých možností, které zatím v této práci nevyužiji, ale je možné ji do budoucna rozšířit. Jedná se například o funkci geolocation pro lokalizaci uživatele či možnost vložit do webové stránky video bez nutnosti instalovat jakékoliv rozšíření [77].

1.5.3 CSS

CSS je zkratka Cascading Style Sheet a označuje jazyk pro popis způsobu zobrazení elementů v jazycích jako je HTML nebo XML.

Pomocí tohoto jazyku je možné určit grafické vlastnosti jednotlivých elementů jako je například velikost textu, jeho tloušťka, odsazení od okolních elementů, efekty a mnohé další vlastnosti. Definice stylu se skládá ze selektoru a bloku pravidel. Selektor uvádí, pro jaké HTML elementy se blok deklarací aplikuje (je možné uvést více selektorů pro jeden blok). Blok deklarací obsahuje jednotlivé deklarace, které jsou od sebe odděleny středníkem. Celý blok deklarací je uzavřen ve složených závorkách. Jednotlivé deklarace jsou ve formátu „identifikátor vlastnosti“: „hodnota“. Příklad deklarace, která obarví veškerý text na stránce (pokud jiná deklarace neurčí jinak) na červenou barvu, muže být následující [66]:

```
body {
        color: #F00;
}
```

V době psaní této práce byla nejaktuálnější verze tohoto jazyka (někdy označována jako CSS3) udělána modulární a tedy se nedá říci, že jazyk bude někdy dokončený, jelikož moduly, které jej tvoří, probíhají neustálým vývojem. Samotné W3C konsorcium přešlo na uvádění takzvaných snapshotů dokumentace, které ji zachycují v daném čase. Toto řešení je výhodnější než verzování [78].

1.5.4 AJAX

AJAX [79] je zkratka pro Asynchronous JavaScript and XML. AJAX není technologie ale spíše popis přístupu jak využívat několik technologií dohromady pro výsledný efekt. Tento popis zahrnuje mimo jiné HTML, CSS, JavaScript a nejvýznamnější – XMLHttpRequest objekt. Tento objekt je API, které poskytuje funkcionalitu klientovi pro získání dat z URL bez nutnosti obnovit stránku [80].

AJAX se používá pro dynamickou obnovu části stránky, aniž by bylo nutné překreslit celou stránku. Nejprve uživatel iniciuje nějakou akci – například vyhledávání mezi produkty – a JavaScript pomocí XMLHttpRequest na pozadí kontaktuje cílový server, který mu zpět předá požadovaná data. JavaScript poté překreslí pouze danou část webové stránky, kde uživatel následně uvidí nový obsah, aniž by musel jakkoliv obnovovat celou stránku.

Ve své diplomové práci budu využívat AJAX v monitoringu pomocí HTTP protokolu.

1.5.5 Nette framework

Nette framework [81] je framework pro programovací jazyk PHP. Slouží pro usnadnění práce tím, že za programátora vyřeší mnoho drobností a úkoly, které se neustále při tvorbě finálního řešení opakují. Koncepce Nette frameworku uvádí, že je stavěn pro to, aby byl co nejvstřícnější a nepoužitelnější. Mezi hlavní výhody tohoto framewroku patří existence šablonovacího systému Latte, podpora pro ladění aplikace, efektivní databázová vrstva, řešení zabezpečení před zranitelnostmi, podpora technologií HTML5 a AJAX a existence rozsáhlé dokumentace.

Framework se umístil jako třetí nejpoužívanější framework ve světě v roce 2015 [82] a jako první nejpoužívanější České republice.

V rámci této diplomové práce jsem tento framework využil pro monitoring pomocí HTTP protokolu, kde jsem hlavně využil podporu konceptu model-view-controller, který slouží pro logické oddělení kódu aplikační logiky od kódu logiky zobrazovací [83]. Mimo jiné jsem také využil podporu pro požadavky pomocí sady technologií AJAX a šablonovací systém Latte. Dále využívám propracovaný systém dependency injection, který dovoluje třídám odebrat zodpovědnost za získávání služeb, které pro svou činnost potřebují, a tuto zodpovědnost přesunují na tu část systému, která tyto služby využívá [84].

1.6 Databáze

Databáze [85] umožňuje ukládat informace a poté k nim opět přistupovat. Informace uložená v databázi je tedy organizovaná tak, že je možné k ní jednoduše přistupovat, měnit ji, případně i odstranit. Nejčastěji používaný typ databáze je relační. V rámci relační databáze jsou informace ukládány do tabulkové struktury, kde jsou jednoznačně definovány a mohou být libovolně přeorganizovány. Pro přístup k informacím je následně možné využít mnoho kritérií, které pomáhají je najít a odfiltrovat ty, které zrovna v danou chvíli nepotřebujeme. Pro přístup k databázi je využíván software, který se anglicky nazývá database management system (DBMS).

Mezi nejznámější DBMS patří MySQL, PostgreSQL, Oracle, Microsoft SQL Server a další. DBMS slouží jako mezivrstva mezi uživatelem, aplikacemi

a databází samotnou. Díky DBMS mohou uživatelé vytvářet, číst, modifikovat a mazat data z databáze. Skládá se ze tří základních komponent – engine, schéma a data. Engine dovoluje přístup k datům včetně jejich modifikace, zamykání a dalších funkcí. Schéma definuje logikou strukturu databáze – tedy určuje jaké data je možné do tabulky uložit, jaké jsou jejich povolené hodnoty a co se má stát v případě jejich změny. Tyto tři hlavní části slouží pro zajištění integrity a bezpečnosti dat a umožňují přístup pro více uživatelů najednou. Pro komunikaci s DBMS se využívá jazyk SQL, který slouží jako programovací jazyk pro zacházení s databází [86].

Ve své diplomové práci budu pro ukládání dat využívat relační databázi pro zefektivnění jejich ukládání a k následnému přístupu. Jedním z požadavků je přístup k datům z více míst najednou – démon bude data ukládat a monitoring je bude zobrazovat a je tedy nutné, aby data byla neustále v konzistentním stavu. Tento požadavek splňují moderní databáze nativně. Do budoucna umožňuje využití databáze rychlejší rozvoj řešení a lepší správu celého systému, jelikož pro většinu programovacích jazyků existuje různé způsoby napojení na databázi a tedy odpadá nutnost parsovat data z textových či jiných souborů.

1.6.1 MySQL

MySQL je druhý nejoblíbenější a nejpoužívanější relační databázový systém dnešní doby [87]. MySQL [88] vznikl v roce 1995 ve Švédsku, kde byla první verze uvedena firmou MySQL AB. Projekt byl založen třemi hlavními iniciátory – Michaelem Wideniusem, Davidem Axmarkem a Allanem Larssonem. Do roku 2000 se jednalo o projekt s uzavřeným zdrojovým kódem, ale v tomto roce byl zdrojový kód uvolněn jako open source, což znamenalo pro firmu pokles zisku o 80%. V roce 2008 byla provedena akvizice společnosti MySQL AB společností Sun Microsystems. Nato dva z původních zakladatelů MySQL, Michael Widenius a David Axmark, z projektu odešli, jelikož nebyli spokojeni, jak Sun s projektem nakládá. V roce 2009 byla společnost Sun Microsystems koupena společností Oracle, což je v podstatě přímý konkurent MySQL. Společnost Oracle se zavázala, že bude projekt nadále rozvíjet. Bez tohoto závazku by neschválila akvizici Evropská komise.

MySQL [89] podporuje v základu mnoho databázových enginů, které jsou v podstatě moduly, a je tedy možné využít moduly, které vznikly v jiných společnostech. Mezi nejpoužívanější patří MyISAM a InnoDB. MyISAM je od verze MySQL 5.0 výchozí engine. Tento engine podporuje mnoho funkcionalit včetně komprese, opravy tabulek, zamykání a další. Bohužel nepodporuje transakční zpracování a cizí klíče. InnoDB umí navíc i transakční zpracování i cizí klíče ale výměnou za určitý pokles výkonu [89]. InnoDB byl původně produktem finské společnosti Innobase. V roce 2006 byla společnost Innobase koupena společností Oracle [90].

Bohužel se ukazuje, že Oracle není tolik motivován k rozvoji řešení, které je open source a je jeho přímým konkurentem [91]. V roce 2012 vyšel článek od jednoho z původních autorů MySQL – Michaela Wideniuse – který uvádí, že v MySQL existuje řada bezpečnostních chyb, o kterých Oracle sice věděl, ale dlouhou dobu nebyly opraveny. V roce 2014 také vypršel závazek společnosti Oracle vůči Evropské komisi a není tedy jisté, co s projektem bude dále. Díky těmto důvodům vzniklo několik klonů MySQL, kde mezi nejznámější a nejrozšířenější patří MariaDB.

1.6.2 MariaDB

MariaDB [92] je projekt, který založili Michael Widenius, David Axmark a Allan Larsson – tedy všichni tvůrci původní MySQL. V roce 2012 vznikla nadace MariaDB Foundation, která celý projekt zastřešuje a zajišťuje, že celý projekt zůstane open source.

Vývoj MariaDB je mnohem více transparentní, jsou detailně popsány veškeré bezpečnostní aktualizace a v posledních letech přináší více nových funkcionalit oproti MySQL. Například bylo uvedeno několik dalších enginů pro ukládání dat. Mimo jiné MariaDB se ukazuje jako výkonnější než MySQL. V posledních letech získala MariaDB velkou popularitu – distribuce Red Hat Enterprise Linux 7 a Suse Enterprise Linux 12 uvádí MariaDB místo MySQL [93]. Mezi webovými službami lze uvést mezi uživatele MariaDB Facebook, Wikipedii nebo Google [92].

Jeden z hlavních aspektů MariaDB je fakt, že je binárně kompatibilní s MySQL. Je tedy bez problému možné data mezi jednotlivými databázemi migrovat a mít je tedy přístupná v obou řešeních [91].

Ve své diplomové práci budu díky důvodům popsaných v této kapitole používat jako relační databázový systém právě MariaDB, který si zachovává otevřený přístup ke komunitě, má aktivní vývoj a již pár široce používaných distribucí operačních systémů zvolilo toto řešení jako výchozí.

1.6.3 ORM

ORM [94] je zkratka pro objektově relační mapování. Objektově relační mapování je programovací technika, která dovoluje přistupovat k objektům bez starosti o to, jak je objekt uložen a jakým způsobem spolu data s objektem souvisí. Tato technika dovoluje programátorům zachovat objektový model bez starosti kdy a jak je nutné data získat či uložit. Další výhodou ORM je možnost změny datového zdroje, takže je tedy možné data migrovat z jednoho typu databáze do jiné aniž by bylo nutné jakkoliv změnit kód aplikace a uživatel se nemusí o změně ani dozvědět.

V rámci své diplomové práce budu využívat peewee ORM pro interakci s databází pro ukládání a získávání dat v rámci programovacího jazyka Python.

1.7 Analýza protokolů pro monitoring

V rámci zadání této diplomové práce byla specifikována nutnost implementace monitoringu pomocí protokolů SSH a HTTP. Oba protokoly popíši v následujících kapitolách.

1.7.1 SSH

SSH [95] je zkratka Secure Shell, což označuje zabezpečený protokol pro komunikaci po nezabezpečené počítačové síti pomocí protokolu TCP/IP. SSH se využívá pro bezpečnou komunikaci mezi dvěma zařízeními. Primárně se tento protokol využívá k přístupu na vzdálený interpret textových příkazů (shell). Mimo tuto funkci může SSH sloužit i pro zabezpečené prohlížení webu skrze zabezpečený proxy server i pro připojení k VPN (virtuální privátní síť) [66].

Monitoring pomocí protokolu SSH se hodí hlavně pro rychlé získání přehledných textových informací o stavu objektu. Z těchto informací je pak dále možné parsovat potřebné části, které lze pak dále využívat v dalších aplikacích. Jelikož se jedná o textové informace tak je možné je zobrazit na téměř jakémkoliv zařízení a je tedy možné mít monitoring dostupný bez vysokých nároků na systém a na přenášená data.

1.7.2 HTTP

Hyper Text Transfer Protocol (HTTP) [96] je název pro protokol aplikační vrstvy, jehož historie sahá až do roku 1990. Protokol byl původně navržen pro přenos surových dat skrze síť internet pomocí protokolu TCP/IP. První verze nesla označení HTTP/0.9. Verze HTTP/1.0 přinesla formát, který připomínal standard MIME. Tento standard dovoluje určit typ přenášených dat pomocí metainformací, které data popisují. Tato verze ale neuvažovala existenci různých systémů pro ukládání informací mezi webovým serverem a klientem (takzvaná cache). Dále chyběla podpora pro vytváření persistentních spojení mezi webovým server a klientem. Pro vyřešení těchto problémů byla uvedená nová verze protokolu HTTP/1.1.

Protokol funguje na principu požadavek – odpověď. V rámci požadavku klient specifikuje URL, která serveru uvede o jaká data klient žádá. Server tento požadavek následně zpracuje a klientovi vrátí odpověď, která rovněž využívá HTTP protokol a obsahuje výsledek zpracovaného požadavku. Klient může k požadavku připojit vlastní data, která server použije při jeho zpracování.

Klient je většinou webový prohlížeč, který slouží pro zobrazování dat formátovaných pomocí jazyka HTML. Pro zabezpečenou komunikaci lze využít protokol HTTPS, který rozšiřuje protokol HTTP o šifrování s využitím protokolů SSL nebo TLS.

Monitoring pomocí tohoto protokolu dovoluje připravit řešení, které je interaktivní a dovoluje uživateli získat data v přehledné formě formátované

pomocí CSS. Mimo zobrazení dat je možné přidat funkcionality s využitím programovacího jazyka JavaScript, díky kterým bude možné jednoduše celé řešení ovládat, upravovat jeho vlastnosti a data periodicky obnovovat bez zásahu uživatele.

1.8 Návrh řešení

Autonomní monitoring a automatizace pro budovy musí implementovat komunikaci, která není závislá na žádné existující počítačové síti či jiné síti, která vyžaduje pro komunikaci nějaký jiný prvek mimo komunikující strany.

Data mohou být jakákoliv, a jelikož moduly nRF24l01+ podporují maximální délku zprávy o velikosti 32 bytů, je tedy potřebné navrhnout vhodný komunikační protokol. Mimo jiné je nutné, aby byla možná komunikace s více než šesti moduly v rámci systému, což je jedno z dalších uvedených omezení tohoto modulu uvedených v kapitole 1.2.4 – nRF24l01+.

Je nutné implementovat prvek – kontrolér – který bude celý systém řídit a bude sloužit jako i přístupový bod pro uživatele, který si díky tomuto prvku bude moci zobrazit data z jednotlivých modulů. Mimo jiné bude tento prvek i bodem, kde bude probíhat automatizace, kterou si uživatel bude moci jednoduše definovat a nastavit. V neposlední řadě bude nutné, aby sám prvek data skladoval a umožnil jejich zobrazení pomocí protokolů HTTP či SSH. Jako vhodnou další funkcionalitu bych uvedl existenci displeje přímo na tomto prvku, kde uživatel bude moci sledovat aktuální data ze systému a bude moci systém i ovládat. V rámci kontroléru bude implementována logika na řízení přístupu do objektu.

Mimo kontrolér pak bude nutné implementovat samotné moduly, které budou provádět monitoring či nějaké akce – například spínat osvětlení. Tyto moduly popíši v dalších kapitolách.

Mnou implementované řešení by mělo být snadno rozšiřitelné, veškeré výstupy by měly být pro uživatele přehledné a jednoduše přístupné. Mezi jednotlivé cíle bych proto zařadil:

1. Modulární řešení kontroléru i monitoringu, které dovolí jejich jednoduché rozšiřování,
2. komunikace mezi kontrolérem a moduly musí být realizovatelná bez existence jakéhokoliv dalšího prvku,
3. přehledné výstupy a možnosti ovládání celého systému,
4. konfigurovatelnost celého řešení bez nutnosti úpravy zdrojových kódů.

1.8.1 Hardware

Jedním ze základních požadavků výsledného řešení je vytvoření platformy, na které celé řešení bude provozováno. Pro kontrolér bylo potřebné spojit jednodeskový počítač Raspberry Pi s dotykovým displejem, modulem nRF24l01+, čtečkou bezkontaktní technologie NFC. Pro zprostředkování monitoringu vzdáleně pomocí protokolů HTTP a SSH je vyžadováno připojení k počítačové síti. V rámci této diplomové práce vznikly dva prototypy kontrolérů. První kontrolér byl bohužel nedopatřením poškozen, a proto se musel vytvořit druhý prototyp, kde jsem zaimplementoval poznatky a vylepšení oproti verzi první.

V následujících kapitolách popíši obě verze kontrolérů a jejich vlastnosti.

1.8.2 Kontrolér – verze 1

První verze byla navržena s maximálním ohledem na cenu celého řešení. Tato verze kontroléru byla založena na jednodeskovém počítači Raspberry Pi 2 Model B. Jako zobrazovací jednotka byl vybrán dotykový displej od alternativního výrobce Waveshare. Tento displej sloužil i jako kostra celého řešení a umožňoval zařízení jednoduše umístit do prostoru. Displej byl s jednodeskovým počítačem Raspberry Pi propojen pomocí HDMI kabelu a USB kabelu pro podporu dotykového rozhraní. Pomocí USB rozhraní byl připojen i modul pro bezkontaktní technologii NFC.

Pro připojení k počítačové síti byl využit USB adaptér s podporou WiFi. Modul nRF24l01+ pro bezdrátovou komunikaci v rámci celého systému byl připojen pomocí GPIO pinů (konkrétně SPI sběrnice).

Jako operační systém byl využit Raspbian. Zařízení je uvedeno na obrázku 1.4.

1.8.3 Kontrolér – verze 2

Mezi nedostatky kontroléru verze 1 patřil hlavně nekvalitní dotykový panel, který i po kalibraci nebyl dostatečně přesný a i kvalita obrazu vykazovala vady jako například rozostření obrazu.

Druhá verze kontroléru byla postavena na jednodeskovém počítači Raspberry Pi 3 Model B, který již obsahuje v základu možnost připojení k bezdrátové síti WiFi a tím tedy odpadá nutnost využívat USB adaptér. Jako zobrazovací jednotka byl využit oficiální sedmipalcový LCD displej od Raspberry Pi Foundation. Pro získávání vizuálního záznamu byla využita oficiální kamera od Raspberry Pi Foundation „Raspberry Pi Camera“ [97]. Jako kostra kontroléru bylo využito taktéž řešení od Raspberry Pi Foundation, které pojme počítač Raspberry Pi, LCD displej, oficiální kameru a lze do něj umístit i modul nRF24l01+. Modul nRF24l01+ byl připojen pomocí SPI sběrnice na GPIO pinech stejně jako u první verze.

Pomocí USB rozhraní byl v druhé verzi kontroléru připojen pouze modul pro bezkontaktní technologii NFC.

Obrázek 1.4: První verze kontroléru

Jako operační systém byl využit Raspbian. Zařízení je uvedeno na obrázku 1.5.

1.8.4 Senzory a aktivní zařízení

Veškeré senzory v rámci této diplomové práce budou postaveny na jednodeskovém počítači Arduino. Je ale možné je postavit i na jiné platformě, která umožňuje propojení s modulem nRF24l01+.

Senzory budou zjišťovat aktuální stav v objektu podle svého určení. Bude se jednat o zařízení, které budou pouze sbírat informace a nebudou vykonávat žádné akce. Druhým typem jsou aktivní zařízení, které mají vykonávat nějakou akci. V rámci této diplomové práce bude implementováno zařízení pro ovládání jiných elektrických zařízení.

V následujících kapitolách popíši návrh jednotlivých senzorů a zřízení, které budou implementovány v rámci této diplomové práce.

1.8.4.1 Senzor kvality ovzduší

Senzor kvality ovzduší bude sloužit pro monitorování ovzduší v budově. Bude pomocí něj možné zjistit aktuální teplotu, vlhkost vzduchu a přítomnost kouře nebo plynu v ovzduší. Pro měření teploty a vlhkosti ovzduší bude využit modul DHT11 [6] a pro zjištění kouře nebo plynu bude využit modul MQ-2 [98].

Obrázek 1.5: Druhá verze kontroléru

Senzor bude periodicky číst data z modulů a bude je ukládat do vnitřních proměnných. V momentě, kdy bude senzor vyzván kontrolérem k dodání dat, je zašle kontroléru pomocí modulu nRF24l01+.

1.8.4.2 Senzor pohybu

Senzor pohybu bude sloužit k detekci narušení prostoru v okamžiku, kdy bude prostor zastřežen. Pro monitoring pohybu bude sloužit infračervený modul HC-SR501 [8]. Senzor bude periodicky monitorovat pohyb, a pokud zjistí narušení pohybu, tak si uloží do vnitřních proměnných tento stav. V momentě, kdy bude kontrolérem vyzván k dodání dat, tak je zašle pomocí modulu nRF24l01+ a kontrolér sám vyhodnotí, jestli byl prostor zastřežen či nikoliv. Po odeslání dat si opět vnitřní proměnné nastaví na stav, který říká, že nebylo detekováno žádné narušení monitorovaného prostoru.

1.8.4.3 Ovládání zařízení

Tento modul bude jediný zástupce aktivních zařízení. Hlavní částí modulu bude relé SRD-05VDC-SL-C [99], kterým se bude ovládat připojené zařízení. V rámci této diplomové práce se bude ovládat světelný zdroj. Modul bude moci sloužit jako vypínač či přepínač – bude záležet na koncovém využití a zapojení relé.

Modul ve výchozím stavu nastaví relé do pozice „normally closed“. Tedy napětí bude přivedeno do jednoho z výstupních terminálů na relé. V případě, že bude do tohoto terminálu připojené nějaké zařízení, bude tímto napájeno.

Modul si do interní proměnné nastaví polohu „OFF“. Jakmile relé dostane od kontroléru příkaz „ON“, tak si nejprve tento stav zaznamená do interní proměnné a pak přepne relé do pozice „ON“ a tím dojde k přepnutí napětí z terminálu „normally closed“ na terminál „normally open“. V tuto chvíli bude zařízení, které je na tento terminál připojeno, napájeno.

Mimo přepínání bude modul po vyzvání kontrolérem na něj odesílat aktuální stav vnitřní proměnné, která označuje stav relé. Celá komunikace bude probíhat pomocí modulu nRF24l01+.

1.8.4.4 Bezkontaktní ověření

Bezkontaktní ověření uživatele bude probíhat pomocí technologie NFC. Jako čtečka NFC bude použito zařízení ACR122U [100], které se připojí přímo ke kontroléru pomocí USB portu. Pro interakci s tímto zařízením bude využita knihovna libnfc, pro kterou existuje i možnost využití v rámci skriptovacího jazyku Python.

Ověření bude provádět kontrolér, který bude pro tuto celou akci mít oddělený proces. Tento proces po úspěšném načtení NFC tagu či karty ověří, zdali daný tag má možnost přístupu do objektu. Pokud uživatel má přístup, tak celý systém zastřeží či odstřeží – bude záležet na předchozím stavu systému. Pokud se bude zastřežovat, tak se na displeji kontroléru zobrazí odpočet času, do kterého musí uživatel opustit prostor. Po uplynutí tohoto času dojde k samotnému zastřežení. Pokud dojde k narušení prostoru, tak bude systém vyčkávat určitý čas, který bude možné nastavit v rámci databáze. Pokud během tohoto času dojde k odstřežení, tak nebude poplach spuštěn.

1.8.5 Software

Software pro výsledné řešení bude tvořen v několika programovacích jazycích. Bude nutné vytvořit ovládací software pro kontrolér, pro zobrazení stavu systému pomoc protokolů HTTP a SSH. Dále pak bude potřeba vytvořit ovládací software pro jednotlivé senzory a zařízení. Řešení jednotlivých částí popíši v následujících kapitolách.

1.8.5.1 Kontrolér

Kontrolér bude muset být v provozu neustále a bude ovládat celý systém. Tato funkce bude realizována pomocí implementace démona, který bude na pozadí dělat veškeré akce pro získání dat pro monitoring, ovládání aktivních zařízení, vykonání příkazů uživatele a provedení logiky automatizace. Démon bude implementován v skriptovacím jazyku Python a bude modulárně rozšiřitelný, což popíši v následující kapitole.

Démon bude periodicky provádět následující akce:

1. Získá z databáze seznam senzorů a zařízení,

2. pokusí se připojit nepřipojená zařízení do systému, pokud toto připojení vyžadují,

3. projde veškeré zařízení, a pokud je zařízení určeno pro periodické získávání dat, tak je získá pomocí modulu pro daný typ zařízení,

4. zpracuje opověď od senzoru či zařízení a předá tato data obslužným modulům, které nad nimi mohou provést další akce,

5. zpracuje příkazy, které jsou ve frontě příkazů.

Démon bude pro každou iteraci periodického bloku příkazů znovu získávat seznam zařízení z databáze, což umožní přidávat či odebírat zařízení bez nutnosti restartovat démona. Pro každou akci, která se má provést pro nějaké zařízení, bude vytvořena nová instance obslužného modulu pro toto zařízení, což umožní dynamické načítání zařízení a odpadne nutnost řešit přednačítání těchto modulů předem. Na druhou stranu tento přístup může být výpočetně a paměťově náročnější. Jelikož je ale platforma Raspberry Pi výkonná a má dostupný dostatek operační paměti, tak by to neměl být problém. V rámci kapitoly testování bude proveden test paměťové náročnosti.

1.8.5.2 Modulární přístup

Jedním z hlavních cílů této diplomové práce je vytvoření řešení, které bude modulární a jednoduše rozšiřitelné. Každý typ zařízení bude tedy mít k dispozici vlastní modul. Moduly budou zastřešovat veškerou komunikaci se zařízeními a démon toto nebude vůbec řešit. Bude tedy dokonce možné jednoduše vyměnit moduly nRF24l01+ za nějaké jiné beze změny kódu démona.

Moduly se budou volat a vytvářet „on-demand“. Tedy budou se volat pouze v případě, že bude potřeba interagovat s nějakým zařízením, pro které existuje v systému ovládací modul.

Automatizace bude řešena takzvanými „hooky“. Hook bude definovaná akce cílového zařízení, která se provede při provádění jiné akce. Například pokud detektor pohybu spustí bezpečnostní poplach, tak bude možné se pomocí hooku navázat na modul ovládající světla a ta zapnout. Hooky budou definovány v databázi a uživatel je bude moct v rámci ní jednoduše měnit. Spouštět je bude démon ve chvíli, kdy bude provádět příkaz na který je hook navázán.

Moduly se budou skládat z jednoho souboru, který bude umístěn ve složce s moduly. V této složce bude umístěn speciální soubor, který bude definovat, které moduly jsou dostupné a jak jsou označené jednotlivé moduly textově. Toto umožní volat obslužné metody správného modulu na základě jeho textového popisu uloženého v databázi.

Démon si při spuštění načte veškeré dostupné moduly, které bude možné podporovat. V periodických rutinách pak bude volat jednotlivé moduly až v případě, že s nimi bude potřeba nějakým způsobem interagovat. Interakce

bude probíhat pomocí volání předem specifikovaných metod, které budou využívat dědičnost od základního modulu. Každý modul bude tedy moci tuto metodu přetížit a danou funkci implementovat svým způsobem.

Modul bude moci ukládat svá data do databáze. Data budou kategorizována. Pro účely historizace bude existovat kategorie označující periodická data. Intervaly historizace bude moci uživatel nastavit pomocí databáze. Kategorie nebudou nijak omezeny a bude záležet pouze na implementaci modulu, jakou kategorii zvolí. Jediná podmínka bude přiřazení dat pomocí cizího klíče k zařízení, ke kterému se daná data vážou.

Modul bude moci definovat své akce při následujících událostech:

1. při startu démona
2. při každém vytvoření instance modulu
3. při spuštění periodických akcí
4. při provedení příkazu
5. při vyhodnocení výsledku příkazu
6. při uložení dat
7. při spuštění hooku
8. při výpadku zařízení

1.8.5.3 Monitoring pomocí HTTP protokolu

Monitoring pomocí protokolu HTTP bude sloužit jako hlavní přehled pro uživatele o stavu objektu. Bude tedy potřeba vytvořit rozhraní, které bude přehledné a dostatečně dimenzované pro ovládání a monitoring celého objektu. Monitoring se bude zobrazovat v rámci webového prohlížeče. Pro tvorbu monitoringu bude využit programovací jazyk PHP s rozšířeními, které nabízí Nette framework. Monitoring bude zabezpečen pomocí přihlášení uživatelským jménem a heslem, které budou navázány na uživatelský profil uživatele uložený v databázi.

Po přihlášení se uživateli zobrazí pracovní plocha – takzvaný dashboard. V rámci dashboardu bude mít uživatel okamžitý přehled o datech z jednotlivých senzorů, zdali je zastřeženo a zdali je senzor aktivní a připojen. U aktivních zařízení bude mít uživatel možnost je ovládat. Data v rámci dashboardu se budou periodicky obnovovat pomocí AJAXu. Pokud uživatel zastřeží prostor pomocí bezkontaktní technologie, je nutné, aby se v rámci dashboardu zobrazilo upozornění s odpočtem času, který zbývá do zastřežení objektu.

Mimo dashboard bude možné si v rámci HTTP monitoringu zobrazit přehled připojených zařízení s údaji o nich, seznam příkazů, které čekají na zpracování s možností jejich zrušení a statistiky o systému, kde bude uvedeno počet

připojených zařízení, počet celkem čekajících příkazů a počet záznamů s údaji uložených v databázi.

O vykreslení dat se bude starat technologie HTML a CSS. Modulárnost celého řešení bude realizována využitím šablonovacího systému Latte a přístupem Model-View-Controller (MVC). Tento přístup oddělí logiku získávání dat, která bude univerzální pro všechny moduly. Latte bude následně zobrazovat tato data již v závislosti na typu modulu a přidání nového modulu bude znamenat pouze úpravu této šablony.

1.8.5.4 Monitoring pomocí SSH protokolu

Monitoring pomocí protokolu SSH bude sloužit pouze pro základní zobrazení aktuálních dat uživateli. Monitoring nabídne uživateli jednoduché textové menu, kde si uživatel zvolí, jaká data chce zobrazit a jakým způsobem. Bude možné zobrazit seznam připojených zařízení včetně detailů o nich. Dále bude možné zobrazit jednorázově data ze všech senzorů popřípadě je zobrazit s periodickým obnovováním.

Zobrazování dat bude opět řešeno modulárně stejným přístupem jako u kontroléru. Tedy moduly se budou skládat z jednoho souboru, který bude umístěn ve složce s moduly. V této složce bude umístěn speciální soubor, který bude definovat, které moduly jsou dostupné a jak jsou označené jednotlivé moduly textově. Toto umožní volat obslužné metody správného modulu na základě jeho textového popisu uloženého v databázi. Podporované moduly budou zaregistrovány při spuštění samotného monitoringu.

Monitoring se bude zobrazovat buď speciálnímu uživateli přímo po přihlášení místo terminálu, nebo uživateli, který si ho explicitně spustí.

1.8.5.5 Software pro senzory a aktivní zařízení

Software pro senzory a aktivní zařízení bude specifický tím, že každé zařízení bude mít svůj vlastní program připravený pro dané zařízení. Zařízení budou pouze sdílet rutiny pro komunikaci s kontrolérem. Pro každé zařízení bude nutné připravit vlastní obslužný kód, který bude komunikovat s moduly, které budou k němu připojené pomocí GPIO pinů.

Senzory budou periodicky číst stav z připojených modulů nezávisle na kontroléru a následně je uloží do své vnitřní paměti. Pokud dostanou z kontroléru příkaz na dodání dat, tak dodají poslední data z paměti a nebudou je v tu chvíli získávat z modulů.

1.8.5.6 Komunikace pomocí modulů nRF24l01+

Z důvodu omezení maximální velikosti posílaných dat modulem nRF24l01+ na 32 bajtů v jedné zprávě je nutné udělat komunikaci co nejúspornější. Proto byl navržen jednoduchý textový protokol. Textová data je následně jednoduché parsovat jak pomocí jazyka Python, tak i pomocí jazyka Wiring. Data budou

ID	Typ	Popis třetího segmentu	Příklad
1	connect	Třetí segment obsahuje pět znaků reprezentující adresu kontroléru. Adresa je ASCII kód daného znaku.	1;3;rfgEr
2	connection_ack	Třetí segment je prázdný.	2;3;
3	get_periodic_data	Třetí segment je prázdný.	3;3;
4	periodic_data_response	Třetí segment obsahuje data, které posílá modul a formát je libovolný dle specifikací modulu a zařízení.	4;8;13.6
5	device_command	Třetí segment obsahuje příkaz, který bude vykonán cílovým zařízením. Podporované příkazy závisí na implementaci daného modulu a zařízení.	5;7;on
6	device_command_ack	Třetí segment je prázdný.	6;7;

Tabulka 1.2: Typy dat a popisy třetího segmentu v rámci komunikace

oddělena na celkem tři segmenty pomocí středníku, který se nesmí vyskytovat v prvních dvou segmentech. První segment bude označovat typ dat.

Druhý segment bude označovat ID zařízení, pro který jsou data určena v případě, že vysílací stranou je kontrolér. Pokud bude vysílací stranou dané zařízení, tak do tohoto segmentu uvede své ID. Číslo ID je nutné z důvodu možných zpoždění při vysílání a kontrolér tedy musí vždy vědět, z jakého zařízení data dorazila.
Data budou tedy posílána v následujícím formátu:

TYP_DAT;ID_ZARIZENI;DATA

Přítomnost třetího segmentu bude volitelná a bude se odvíjet od typu dat, který je posílaný. Tabulka 1.2 ukazuje podporované typy dat, popis třetího segmentu a příklady celé zprávy.

Navázání komunikace začne nejprve na kontroléru, který se pokusí zařízení připojit. S nepřipojeným zařízením není možná komunikace. Kontrolér tedy na adresu uvedenou v databázi odešle zprávu typu `connect`, kde bude v druhém segmentu uvedeno ID připojovaného zařízení a v třetím segmentu adresa kontroléru. Zařízení po přijetí této zprávy musí vždy upravit konfiguraci dle přijatých parametrů. Do vnitřní paměti si tedy uloží adresu kontroléru, se kterým bude komunikovat, a své přiřazené ID, které bude uvádět ve všech vysílaných zprávách. Pro úspěšné připojení je nutné poslat zpět na kontrolér zprávu typu `connection_ack` a ten poté označí zařízení jako připojené.

Pokud se již připojené zařízení nepovede déle jak jednu minutu kontaktovat s běžnou komunikací, tak se nastaví jako nepřipojené a kontrolér musí před další komunikací znovu navázat připojení. Tento postup řeší možné výpadky jak kontroléru, tak i samotných senzorů a odstraňuje potřebu uvádět přímo do jejich kódu adresu kontroléru a ID zařízení. Díky tomuto se může adresa kontroléru změnit a zařízení jsou za běhu schopné se přeregistrovat. Do kódu zařízení je tedy nutné uvést pouze adresu jeho samotného. Tato adresa musí být v rámci jednoho objektu vždy unikátní. Na této adrese pak bude zařízení poslouchat a přijímat příkazy od kontroléru.

Běžná komunikace bude probíhat tak, že nejprve kontrolér pošle zařízení dotaz, zařízení tento dotaz zpracuje a odešle kontroléru zpět odpověď. Zařízení nebudou moci komunikovat bez požadavku od kontroléru. V rámci periodických získávání dat bude každé zařízení mít časové okénko, kdy bude vyzváno kontrolérem k dodání dat. K tomuto dodání bude mít 300 ms. Velikost tohoto časového okna vznikla z empirického měření uvedeného v kapitole 3.2.1 – Testování dosahu bezdrátového modulu NRF24l01+. Pokud data odešle později, tak již data kontrolér nezpracuje z důvodu zpracovávání dat od dalšího zařízení. Zařízení tedy bude muset data vyslat znovu v další periodě. Takto postavený protokol umožní využít většinu celkové kapacity jedné zprávy pro třetí segment, kde se mohou posílat užitečná data.

Kapitola 2

Realizace

V této kapitole blíže popíši jednotlivé aspekty návrhu řešení autonomního systému pro automatizaci a monitoring budov. Budu se věnovat popisu jednotlivých komponent, ze kterých se celý systém skládá. Dále popíši vlastní realizaci kontroléru. V rámci tohoto popisu uvedu realizaci modulárnosti celého řešení a popíši jednotlivé moduly. V neposlední řadě uvedu strukturu databáze, která slouží pro ukládání dat a bude sdílená pro všechny části systému.

Veškerá označení portů v tabulkách znázorňující propojení pinů odkazují na piny uvedené pro daný modul v příloze B – Schémata zařízení a modulů.

2.1 Zapojení hardwaru kontroléru

Ke kontroléru je potřeba zapojit celkem čtyři další zařízení. Jedná se o LCD dotykový displej, oficiální kameru, modul nRF24l01+ a čtečku NFC karet a tagů ACR122U. Čtečka ACR122U se zapojí do libovolného volného USB portu na jednodeskovém počítači Raspberry Pi 3. Zapojení modulu nRF24l01+ ukazuje tabulka 2.1.

Zapojení dotykového displeje se provede pomocí rozhraní DSI, které se propojí s využitím plochého kabelu. Dále je nutné propojit piny 5V a GND

Raspberry Pi 3	**nRF24l01+**
06 / GND	1 / GND
01 / 3.3V DC	2 / VCC
22 / GPIO25	3 / CE
24 / GPIO08	4 / CS
23 / GPIO11	5 / SCK
19 / GPIO10	6 / MOSI
21 / GPIO09	7 / MISO

Tabulka 2.1: Zapojení modulu nRF24l01+ k Raspberry Pi 3

na displeji s korespondujícími piny na Raspberry Pi 3 (piny 02 pro 5V a 14 pro GND).

Zapojení kamery se provede pomocí rozhraní CSI, které se propojí s využitím plochého kabelu.

2.2 Návrh databáze

Databáze bude využívána jako úložiště veškerých dat ze senzorů, konfigurace zařízení, uživatelské konfigurace, uživatelů, ovládacích příkazů celého systému, pravidel automatizace a dat pro prostředí monitoringu pomocí protokolu HTTP.

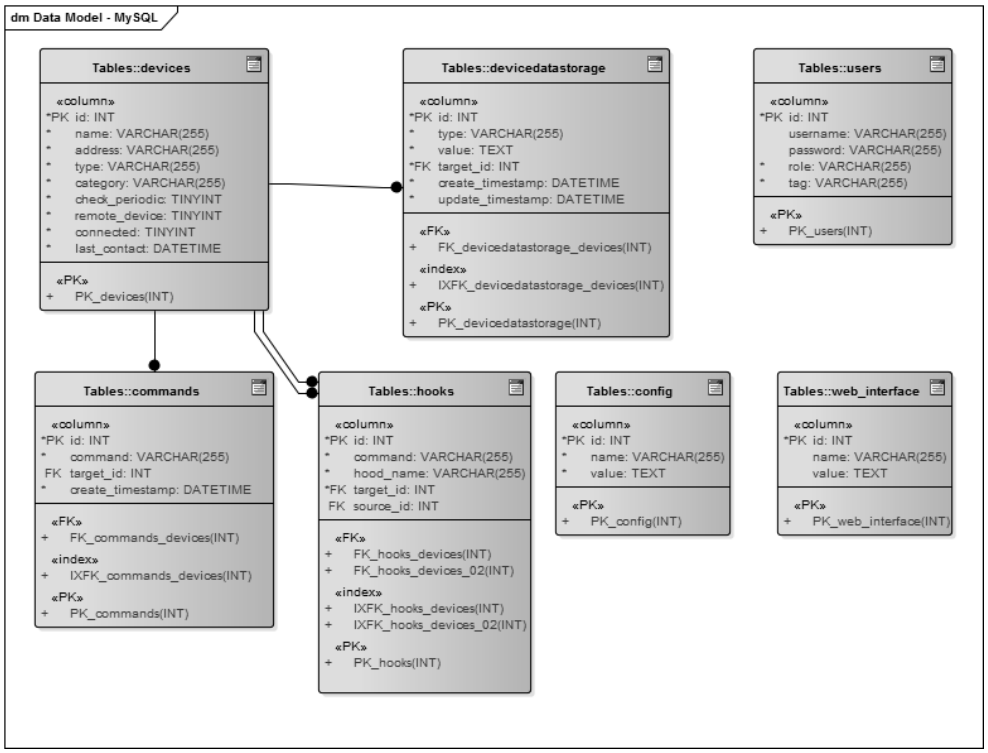
Jednotlivé tabulky budou mít strukturu vytvořenou pomocí ORM modulu peewee. Tato struktura je využitelná i pro zpracování bez tohoto modulu, jelikož vytvořené schéma neobsahuje žádné další položky mimo těch definovaných při vytváření tabulky. Dále modul peewee využívá cizí klíče a je tedy možné s takto navrženou databází pracovat i bez tohoto modulu. Nette framework umí tabulky referencované cizími klíči spojovat a vytvářet nad nimi složitější dotazy. Dále díky definovaným integritním omezením zůstanou data v tabulkách vytvořených modulem peewee v konzistentním stavu i když se s nimi manipuluje bez tohoto modulu (za podmínky, že RDBMS tyto omezení kontroluje a vynucuje). Každá tabulka bude mít atribut `id`, který bude sloužit jako primární klíč.

Na obrázku 2.1 je možné vidět schéma databázových tabulek používaných v rámci tohoto řešení.

Tabulka `devices` slouží pro ukládání dat ohledně jednotlivých zařízení. Atribut `name` slouží pro pojmenování zařízení uživatelem, `address` slouží pro uložení adresy zařízení, `type` je určen pro uložení typu zařízení. Atribut `category` slouží pro uložení kategorie zařízení, `check_periodic` určuje, zdali má kontrolér periodicky žádat zařízení o data, `remote_device` určuje zdali je nutné se zařízením komunikovat pomocí modulu nRF24l01+, atribut `connected` určuje, zdali je zařízení připraveno pro komunikaci. Atribut `last_contact` ukládá datum a čas poslední úspěšné komunikace se zařízením.

Tabulka `devicedatastorage` slouží pro ukládání dat modulů, které jsou napojeny na démona. Atribut `type` slouží pro určení typu dat v atributu `value`. Data v atributu `value` by vždy měly být ve formátu JSON. Atribut `create_timestamp` a `update_timestamp` slouží k zaznamenání času vytvoření daného záznamu respektive poslední aktualizace. Atribut `target_id` slouží jako cizí klíč a označuje zařízení, kterému data patří.

Tabulka `commands` slouží pro realizaci fronty příkazů ke zpracování démonem. Atribut `command` slouží pro uložení příkazu a atribut `create_timestamp` je určen pro uložení času vložení příkazu. Atribut `target_id` je nepovinný cizí klíč tabulky `devices` a umožňuje nějaký příkaz přiřadit pouze pro jedno zařízení.

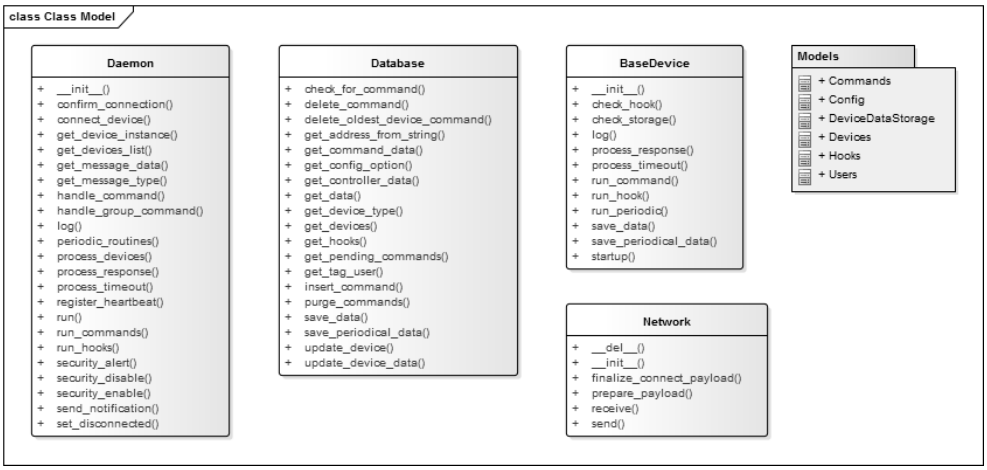


Obrázek 2.1: Schéma databáze

Tabulka `hooks` slouží pro definici akcí, které se mají stát při vykonání nějakého příkazu démonem. Atribut `command` určuje, pro jaký příkaz je hook určen. Atribut `hook_name` určuje jméno samotného hooku podle kterého obslužný modul provede definovanou akci. Atribut `target_id` je cizím klíčem tabulky `devices` a určuje které zařízení má daný hook provést. Atribut `source_id` je cizím klíčem tabulky `devices` a dovoluje omezit spuštění hooku pouze v případě, že příkaz je určen pro dané zařízení.

Tabulky `config` a `web_interface` mají stejné schéma, jelikož obě slouží pro ukládání konfiguračních voleb a dalších proměnlivých voleb pro výsledné řešení. V tabulce `config` se ukládá konfigurace démona a globálních parametrů, pomocí kterých se bude konfigurovat celý systém. Například zde bude určení doby zpoždění od vložení příkazu na zastřežení až k samotnému zastřežení. Tabulka `web_interface` slouží pro uložení dat relevantních pouze pro řešení monitoringu pomocí HTTP protokolu. Atribut `name` určuje jméno či typ daného záznamu a atribut `value` slouží pro uložení dat relevantních k danému typu záznamu.

Tabulka `users` slouží k uložení uživatelů, kteří mají přístup do systému. Struktura tabulky byla převzata z ukázkové aplikace Nette frameworku. Atri-



Obrázek 2.2: Třídy pro realizaci démona

but `username` slouží k uložení uživatelského jména uživatele, `password` je určen pro uložení hashe hesla, `role` je volitelný atribut k uložení role daného uživatele a `tag` je atribut pro uložení přístupového ID tagu pro bezkontaktní technologie.

2.3 Démon

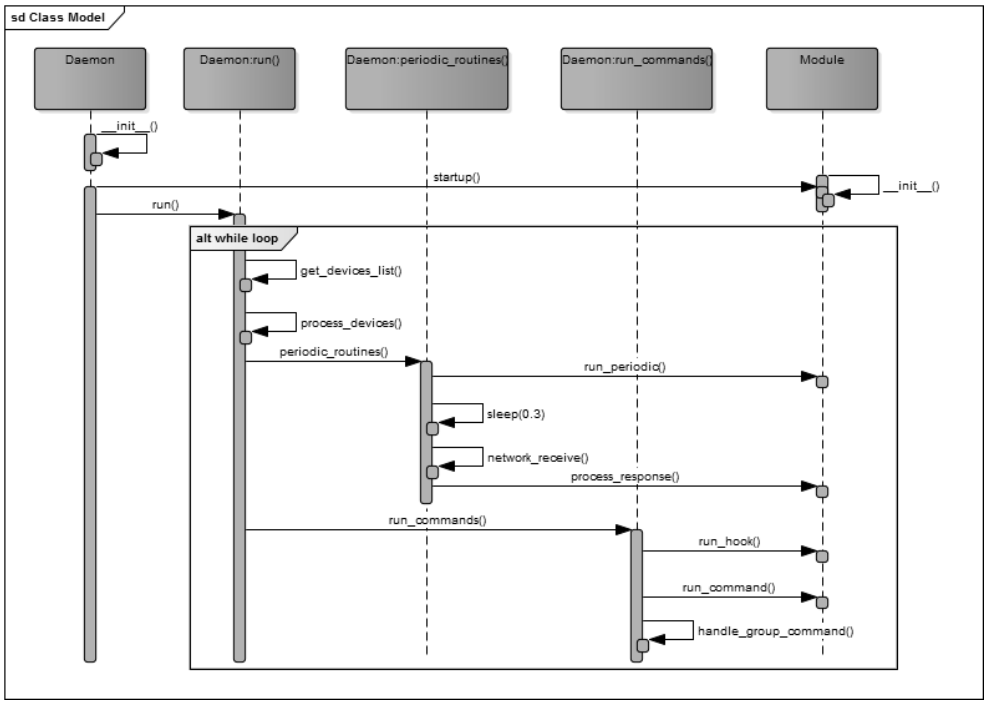
Démon je program, který bude běžet na pozadí a bude zastřešovat celé řešení. Bude odpovědný za komunikaci se senzory a zařízeními, ukládání dat do databáze, vyřizování uživatelských požadavků, zabezpečení prostoru a automatizaci. V následujících kapitolách popíši jeho strukturu, způsob realizace modulárního řešení a jeho životní cyklus.

2.3.1 Návrh tříd

Démon bude pro svou práci potřebovat součinnost více systémů a bude využívat více knihoven. Rozdělil jsem proto realizaci do několika tříd, které slouží jako pomocné třídy hlavní třídě démona, která je nazvaná `Daemon`. Na obrázku 2.2 je možné vidět základní rozdělení řešení démona na jednotlivé třídy včetně metod, které třídy obsahují.

2.3.2 Démon a jeho životní cyklus

Třída `Daemon` je zodpovědná za řízení celého systému. Na obrázku 2.3 je možné vidět zjednodušený životní cyklus démona bez větvení na základě stavu připojení modulu a bez smyček mimo hlavní smyčku. V rámci této kapitoly tento cyklus blíže popíši.



Obrázek 2.3: Životní cyklus démona

Životní cyklus začíná spuštěním démona. Pro spuštění démona stačí vytvořit novou instanci třídy `Daemon`. Při jejím vytvoření se spustí metoda `__init__`, která zajistí vytvoření spojení s databází vytvořením instance třídy `Database`. Dále je vytvořena instance třídy `Network`, která má na starosti komunikaci pomocí modulů nRF24l01+. Následně je z databáze získán seznam zařízení a pro každé zařízení je zavolána metoda `startup`, v rámci které si může každý načtený modul provést akce, které jsou nutné pro jeho běh. Metoda `startup` je volána pouze jednou při startu démona. Jako poslední se provede zavolání metody `run`, která zajišťuje běh démona do doby, dokud není ukončen.

Metoda `run` se periodicky opakuje a vždy volá stejnou sekvenci metod. Nejprve je zavolána metoda `get_devices_list`, která vrátí díky interakci s databází seznam všech v ní definovaných zařízení. Jelikož se tato metoda volá v každém cyklu znovu, tak je možné přidat zařízení za běhu démona bez nutnosti jeho restartu. Následně je zavolána metoda `process_devices`, která zjistí, jestli byla v posledních šedesáti sekundách provedena úspěšně jakákoliv komunikace se zařízením. Pokud ne, tak je zařízení označeno jako odpojené a je nutné jej znovu připojit. Poté je řízení vráceno zpět do metody `run`. Následně tato metoda zavolá metodu `periodic_routines`, která má na starosti veškeré činnosti, které je nutné opakovat.

Metoda `periodic_routines` provede stejný cyklus pro všechna zařízení. Nejprve ověří, zdali je nutné zařízení periodicky kontaktovat, a pokud nikoliv, tak je zařízení přeskočeno. Pokud není zařízení přeskočeno, tak se ověří, zdali je připojeno (provede se kontrola atributu `connected`). Pokud není a je ho potřeba připojit (kontrola parametru `remote_device`), tak se zavolá metoda `connect_device`, která odešle zařízení zprávu pomocí modulu nRF24l01+ s jeho ID a adresou kontroléru. Pokud je zařízení připojené, tak se získá správný modul pro tento typ zařízení a zavolá se pomocí něj metoda `run_periodic`. Následně se počká 300 milisekund, po které vzdálené zařízení musí odeslat odpověď. Poté je zavolána metoda `network_receive`, která pomocí třídy `Network` zkusí získat data od vzdáleného zařízení. Následně je zavolána metoda `process_response`, která tuto odpověď zpracuje.

V případě, že odpověď je prázdná, tak metoda `process_response` zavolá metodu `process_timeout`, která ověří, že nedošlo k překročení limitu šedesáti sekund od poslední komunikace. Pokud k tomuto došlo, tak se provede odpojení zařízení. Poté metoda `process_response` ukončí svou činnost.

V případě, že odpověď dorazila, tak je z ní nejprve získáno ID zařízení, které je ověřeno vůči aktuálně zpracovávanému zařízení. Pokud se ID liší, je tato odpověď zahozena a metoda `process_response` se spustí od začátku. Pokud se ID shodují, tak je označena komunikace jako úspěšná a zaznamená se nový čas úspěšné komunikace do databáze pomocí metody `register_heartbeat`. Následně se přejde ke zpracování samotné odpovědi.

Pokud byla metoda `process_response` zavolána s platným parametrem `handler`, tak metoda nejprve rozparsuje typ zprávy a data a s těmi předá řízení správnému modulu pro daný typ zařízení pomocí zavolání metody `process_response` daného modulu. Pokud nebyl parametr `handler` specifikován, ověří se typ zprávy a pokud je mezi známými typy, tak se provede daná akce pomocí zavolání obslužné metody démona. V rámci této implementace je přípustná pouze metoda pro potvrzení připojení zařízení, která je obsloužena pomocí metody `confirm_connection`, která nastaví zařízení do stavu připojeno. Pokud není nalezen podporovaný typ, tak se zaloguje chybová hláška. Poté se řízení vrací zpět metodě `run`.

Metoda `run` zavolá poslední metodu v rámci periody – `run_commands`. Tato metoda nejprve získá z databáze seznam čekajících příkazů na provedení. Následně se spustí cyklus se všemi čekajícími příkazy. Pokud má příkaz uvedeno i cílové zařízení je nejprve provedeno spuštění hooků zavoláním metody `run_hook` na příslušném obslužném modulu. Poté je získán správný obslužný modul a je na něm zavolána metoda `run_command`. Pokud není cílové zařízení uvedeno, jedná se o příkaz pro démona a je pro tento příkaz zavolána metoda `handle_group_command`.

Tato metoda nejprve provede veškeré zaregistrované hooky zavoláním metody `run_hook` na příslušném obslužném modulu. Následně projde vnitřní seznam existujících příkazů, a pokud je nalezena shoda, je spuštěna příslušná metoda. V rámci této implementace je možné zavolat příkazy kontroléru pro

zastřežení, odstřežení prostoru a bezpečnostní poplach. V rámci vyhlášení bezpečnostního poplachu je zavolána metoda `send_notification`, která odešle email s upozorněním. Adresu příjemce je možno nastavit volbou `notification_email` v tabulce `config`.

2.3.3 Realizace komunikace s databází

Komunikace s databází je realizována třídou `Database`, která využívá modulu peewee. V rámci souboru `Models.py`, který obsahuje definující třídy pro peewee. Jsou definovány třídy `Devices`, `Config`, `DeviceDataStorage`, `Users`, `Commands` a `Hooks` korespondující s jednotlivými tabulkami uvedenými v kapitole 2.2 – Návrh databáze.

V rámci třídy `Database` je definováno několik metod. Většina metod slouží pro získání, smazání či aktualizaci dat. Kompletní seznam metod je možné vidět na obrázku 2.2 a jejich popis je možné najít v dokumentaci na přiloženém CD. V tabulce 2.2 uvedu několik nejzajímavějších metod z této třídy.

Velmi důležitým pravidlem je explicitní uzavírání spojení s databází pomocí volání metody `db.close` po provedení každého dotazu. Bez tohoto uzavírání není možné k databázi přistupovat z více vláken či procesů najednou a dochází ke kolizím, které vedou k pádu celého démona.

2.3.4 Realizace modulárního řešení

Démon je řešen hlavní třídou `Daemon`, která se kromě řízení celého řešení stará i o kontrolu modulů, jejich načítání a volání metod jednotlivých modulů. Pro modulární řešení jsem vybral řešení pomocí dědičnosti. Rodičem je třída `BaseDevice`, která definuje metody, které démon volá za určitých podmínek. Třída, která rozšiřuje třídu `BaseDevice`, může využít existující implementaci metod existujících v rámci rodičovské třídy, nebo je může přetížit vlastními. Každý modul musí být implementován v jednom souboru a může si naimportovat libovolné závislosti. Pro aktivaci modulu je nutné modul přidat do souboru `__init__.py` do sekce importu modulů a přidat do slovníku `AVAILABLE_HANDLERS` záznam pro překlad textového označení typu zařízení na název třídy implementující obsluhu daného zařízení. Po tomto kroku je možné do databáze přidat zařízení uvedením záznamu do tabulky `devices`. Poté bude nově implementované zařízení zařazeno do systému.

Hooky jsou řešeny pomocí překladu textového příkazu na název metody, která je poté v rámci modulu zavolána. Seznam podporovaných textových názvů je uveden jako klíč u slovníku `hooks`, který je definován v rámci kódu modulu.

V rámci tabulky 2.3 je možné vidět přehled metod s popisem jejich určení.

Mimo metody, které volá démon, jsou definovány metody `__init__`, která slouží pro předání dat od démona, `log` pro záznam zpráv pomocí

Metoda	Popis
`get_address_from_string`	Metoda převede adresu zapsanou v textovém formátu tvaru na pole pěti celých čísel. Adresa uložená v databázi je ve tvaru pěti hexadecimálních číslic oddělených dvojtečkou, kde každá je v rozsahu 00 až FF. Validní adresa tedy může například být F0:F0:E1:E1:F0.
`update_device_data`	Metoda nejprve načte aktuální data zařízení, následně převede JSON na původní formát, provede aktualizaci dat, data zpět zabalí do formátu JSON a uloží do databáze.
`delete_oldest_device_command`	Metoda odstraní nejstarší příkaz pro dané zařízení. Metoda existuje z důvodu možnosti existence více příkazů pro dané zařízení.
`purge_commands`	Metoda smaže veškeré příkazy, které obsahují určitý text. Tato metoda se hodí například pro smazání veškerých bezpečnostních akcí po odstřežení objektu.
`check_for_command`	Metoda slouží pro ověření existence příkazu ve frontě příkazů. Ověření probíhá jak pro příkazy určené pro určité zařízení, tak pro příkazy určené pro kontrolér.
`get_tag_user`	Metoda slouží pro nalezení uživatele podle načteného tagu z čtečky NFC tagů.
`get_data`	Metoda se pokusí najít data dle parametru textttkey. Pokud jsou data nalezena, tak je proveden převod z formátu JSON a původní formát a data jsou předána metodě, která o data žádala.
`save_periodical_data`	Metoda slouží pro ukládání periodických dat. Nejprve je ověřena existence staršího záznamu. Pokud starší záznam existuje, tak je spočítáno jeho stáří podle atributu textttcreate_timestamp. Pokud je záznam starší než uživatelem definovaný čas, tak je provedena jeho archivace a je vytvořen záznam nový. Tvorba nového záznamu se provede taktéž, pokud záznam neexistuje.

Tabulka 2.2: Popis vybraných metod z třídy Database

Metoda	Popis
`process_response`	Metoda slouží pro zpracování odpovědi od zařízení. Je volána pouze v případě, že dorazí data.
`process_timeout`	Metoda slouží pro akce, které se mají stát, pokud zařízení neodpovídá.
`run_command`	Metoda, která je volána, pokud ve frontě příkazů je příkaz pro dané zařízení.
`run_hook`	Metoda, která je zavolána, pokud je v databázi definován hook pro dané zařízení. Metoda `check_hook` následně ověří validitu požadovaného příkazu.
`run_periodic`	Metoda slouží pro vykonání periodických akcí jako je získání dat ze senzorů. Je volána démonem pravidelně, pokud je atribut zařízení `check_periodic` nastaven na hodnotu 1.
`startup`	Metoda je volána při startu démona, pokud existuje v databázi zařízení daného typu.

Tabulka 2.3: Přehled metod modulů volaných démonem

funkce `log` třídy `Daemon`. Dále jsou definovány metody `save_data` a `save_periodical_data`, které slouží pro ukládání dat získaných ze zařízení do databáze.

2.3.5 Komunikace pomocí modulu nRF24l01+

Komunikaci pomocí modulu nRF24l01+ zastřešuje třída `Network`. Tato třída využívá pro samotnou komunikaci knihovnu `lib_nrf24` [69]. Třída implementuje celkem čtyři metody, konstruktor a destruktor. Konstruktor (metoda `__init__`) nastaví veškeré parametry pro komunikaci pomocí tohoto modulu – jedná se o nastavení kanálu, data rate, vysílaný výkon, adresu přijímací pipy a zahájí naslouchání pro příchozí data. Volba těchto parametrů vychází z testu dosahu, který lze nalézt v kapitole 3.2.1 – Testování dosahu bezdrátového modulu NRF24l01. Destruktor při ukončení činnosti zařídí zastavení poslouchání modulu pro příchozí data.

Metoda `send` převede textová data na datový typ list. Pokud bude počet prvků v tomto listu menší než 32, bude list doplněn nulami, aby tento list obsahoval přesně 32 prvků. Poté jsou data odeslána cílovému zařízení. Metoda `receive` ověří, zdali jsou dostupná data k přijetí. Pokud ano, tak jsou data přijata a převedena do textové podoby pomocí nativní funkce `chr`. Tato funkce převede celočíselný kód znaku do jeho textové podoby dle ASCII tabulky.

Metoda `prepare_payload` slouží pro zkonstruování finální textové po-

Raspberry Pi	nRF24l01+
GND	1 / GND
3V3	2 / VCC
9	3 / CE
10	4 / CS
13	5 / SCK
11	6 / MOSI
12	7 / MISO

Tabulka 2.4: Zapojení modulu nRF24l01+ k Arduinu

doby zprávy dle specifikací textového protokolu uvedené v kapitole 1.8.5.6 – Komunikace pomocí modulů nRF24l01+. Metoda přijímá parametr `data`, který umožňuje naplnit třetí segment podle uvedeného typu zprávy. V rámci tohoto řešení je implementována pomocná metoda s názvem `finalize_connect_payload`, která vytvoří třetí segment pro zprávu typu `connect`.

2.4 Senzory a aktivní zařízení

Senzory a aktivní zařízení, které budou se systémem komunikovat, jsou všechny založeny na jednodeskovém počítači Arduino Uno. Modul pro bezkontaktní ověření uživatele a modul pro získání vizuálního záznamu jsou připojeny přímo ke kontroléru. V následujících kapitolách popíšu jejich realizaci včetně zapojení hardwaru.

2.4.1 Komunikace

Všechny vzdálené senzory a aktivní zařízení, které jsou realizovány v rámci této diplomové práce, jsou založeny na jednodeskovém počítači Arduino Uno. Propojení tohoto počítače s modulem nRF24l01+ popisuje tabulka 2.4. Po zapojení dle této tabulky a nahrání příslušného programu je možné okamžitě komunikovat bez jakéhokoliv dalšího nastavení. Arduino využívá pro spojení s modulem knihovnu `RF24` od TMRh20 [43].

V rámci funkce setup Arduino nejprve nastaví veškeré parametry stejně, jako jsou nastaveny na straně kontroléru. Poté čeká ve `while` cyklu, dokud ho nekontaktuje kontrolér, který mu pošle svou adresu a přiřadí danému zařízení číslo ID. Poté je běh funkce ukončen a je předáno řízení funkci `loop`.

Funkce `loop` proběhne jednou za 100 milisekund. Nejprve se pokusí pomocí funkce `receive` přijmout data z modulu nRF24l01+, poté se z dat pokusí získat jejich typ. Poté následuje konstrukce `switch`, která dle typu dat spustí příslušnou funkci. Pokud přijde požadavek o připojení, tak je vyřízen pomocí funkce `process_connect_command`. Pokud se jedná o poža-

davek získání periodických dat, tak je zavolána funkce `process_periodic_data_request_command`, jejíž implementace se liší dle modulu. Pokud je typ zprávy `device_command`, tak je zavolána funkce `process_command`, která se opět liší dle modulu.

Funkce `receive` slouží pro přijímání dat pomocí modulu nRF24l01+. Funkce si nejprve ověří, že jsou dostupná nějaká data. Pokud nejsou, tak funkce končí. Pokud jsou data dostupná, tak si funkce vytvoří pole o velikosti 32 znaků (datový typ char). Do tohoto pole jsou následně nakopírována přijatá data.

Funkce `radio_send` nejprve otevře zapisovací pipu na adresu kontroléru, pozastaví čtecí pipu z důvodu nemožnosti současného čtení a zapisování. Následně převede textová data do pole znaků. Toto pole je následně odesláno pomocí funkce `write`. Po odeslání dat je opět povoleno přijímání dat.

Funkce `get_message_type`, `get_device_id` a `get_message_data` rozparsují příchozí data na jednotlivé segmenty a vrací je volající funkci.

Funkce `process_connect_command` slouží pro aktualizaci adresy kontroléru a ID zařízení. Funkce nejprve prochází všech pět přijatých bajtů adresy, z kterých pomocí operací bitového posunu a bitové operace `or` postupně konstruuje adresu kontroléru, která musí být uložena v datovém typu uint64_t. Poté pomocí funkce `get_device_id` získá nové ID zařízení a to uloží do proměnné `device_id`.

2.4.2 Senzor pro monitoring kvality ovzduší

Senzor pro monitoring kvality ovzduší v rámci funkce `process_periodic_data_request_command` získává data o teplotě a vlhkosti z modulu DHT11. Data o přítomnosti škodlivých látek v ovzduší získává pomocí modulu MQ-2. Pro získání těchto dat jsou v rámci periodických průběhů funkce `loop` volány postupně funkce `get_temperature`, `get_humidity` a `gas_sensor_check`, které se senzorů vyčtou příslušné údaje a uloží je do vnitřních proměnných. V případě dotazu kontroléru na tyto data je funkce `process_periodic_data_request_command` pouze přečte z vnitřních proměnných a pošle kontroléru.

Korespondující modul v rámci démona nese název `AirQuality` a pouze přetěžuje metodu `process_response`, kde definuje periodická data, která se ukládají do databáze.

Tabulka 2.5 ukazuje zapojení modulu DHT11 a tabulka 2.6 ukazuje zapojení modulu MQ-2 k jednodeskovému počítači Arduino.

2.4.3 Senzor pro detekci pohybu

Senzor pro detekci pohybu v rámci funkce `process_periodic_data_request_command` získává data z modulu HC-SR501. Pro získání těchto dat je v rámci periodických průběhů funkce `loop` volána funkce `get_motion_status`, která

Arduino	DHT11
5V	1Pin
Digital Pin 2	2Pin
GND	4Pin

Tabulka 2.5: Zapojení modulu DHT11 k Arduinu

Arduino	MQ-2
5V	Vcc
Analog Pin 0	Output
GND	Gnd

Tabulka 2.6: Zapojení modulu MQ-2 k Arduinu

Arduino	HC-SR501
5V	+Power
Digital Pin 8	High / Low Output
GND	GND

Tabulka 2.7: Zapojení modulu HC-SR501 k Arduinu

z modulu vyčte data a uloží je do vnitřní proměnné. Pokud je zjištěn pohyb, tak dokud nejsou data odeslána kontroléru, není možné vnitřní proměnnou změnit. V případě dotazu kontroléru na data je funkce `process_periodic_data_request_command` pouze přečte z vnitřních proměnných a pošle kontroléru.

Korespondující modul v rámci démona nese název `MotionDetector` a přetěžuje metodu `process_response`, kde definuje periodická data, která se ukládají. Pokud byl zjištěn pohyb a prostor je zastřežen (kontrola atributu `space_armed`) a ještě nedošlo k vyhlášení poplachu (kontrola atributu `security_alert`), tak modul zařadí příkaz k vyhlášení poplachu do fronty příkazů.

Tabulka 2.7 ukazuje zapojení modulu HC-SR501 k jednodeskovému počítači Arduino.

2.4.4 Zařízení pro ovládání jiných zařízení

Senzor pro ovládání zařízení v rámci funkce `process_periodic_data_request_command` pouze vrátí stav vnitřní proměnné `device_state`, která označuje, jestli je zařízení ve stavu `on` či `off`.

Dále zařízení implementuje funkce `device_on` a `device_off`, které jsou volány z funkce `process_command` podle přijatého příkazu. Funkce zařídí přepnutí relé do pozice `normally open` respektive `normally closed`.

Arduino	SRD-05VDC-SL-C
5V	VCC
Digital Pin 8	IN
GND	GND

Tabulka 2.8: Zapojení relé SRD-05VDC-SL-C k Arduinu

Korespondující modul v rámci démona nese název `Relay` a přetěžuje metody `process_response`, kde podle typu odpovědi modulu buď uloží periodická data (typ odpovědi `periodic_data_response`) nebo provede potvrzení provedení příkazu (typ odpovědi `device_command_ack`). Mezi další přetížené metody patří `run_hook`. Tato metoda dovoluje zapnout zařízení v případě provádění příkazu `security_alert`. Přetížená metoda `run_command` dovoluje zařízení zapnout, vypnout či přepnout na základě aktuálního stavu. K samotné změně jsou využity metody `switch_on` a `switch_off`, kde tyto metody pošlou nový cílový stav zařízení pomocí zprávy typu `device_command`.

Tabulka 2.8 ukazuje zapojení relé SRD-05VDC-SL-C k jednodeskovému počítači Arduino.

2.4.5 Zařízení pro bezkontaktní ověření

Zařízení pro bezkontaktní ověření je čtečka karet NFC ACR122U, která je připojena do USB portu kontroléru. U tohoto zařízení existuje pouze modul v rámci démonu s názvem `NFCReader`. Tento modul přetěžuje metodu `startup`. Tato metoda vytvoří nový proces pomocí rozhraní `multiprocessing`, které spustí metodu `run` v rámci jiného procesu. Tato metoda není tedy závislá na halvním procesu démona a běží odděleně. Metoda `run` nejprve připojí čtečku ACR122U pomocí funkce `register_reader`, která nastaví veškeré potřebné parametry pro čtení běžné dostupných NFC karet skrze knihovnu `libnfc`. Po tomto nastavení probíhá cyklus `while`, který se neustále snaží pomocí metody `read_tag` přečíst NFC tag a následně pomocí metody `process_tag` tento tag ověřit vůči databázi. Pokud není načten žádný tag do přibližně pěti sekund, tak se cyklus čtení opakuje znovu.

Pokud je čtení tagu úspěšné a bylo možné jej dohledat v databázi, tak bude vložen do fronty příkazů příkaz pro zastřežení nebo odstřežení objektu pomocí metody `toggle_security` podle aktuálního stavu prostoru.

Kontrolér následně příkaz zpracuje a prostor odstřeží nebo zastřeží. Zastřežení má od vydání příkazu zpoždění, které je možno nastavit volbou `lockdown_time` v tabulce `config`.

2.4.6 Zařízení pro získání vizuálního záznamu

Zařízení pro získání vizuálního záznamu je realizováno kamerou připojenou k rozhraní CSI. Jelikož se jedná o lokálně připojené zařízení, tak existuje pouze modul v rámci démona. Tento modul nese název `Camera`. Přetíženy jsou metody `run_periodic`, která dovoluje zaznamenávat fotografii prostředí periodicky a metoda `run_hook`, která dovoluje navázat vytvoření fotografie na jinou událost.

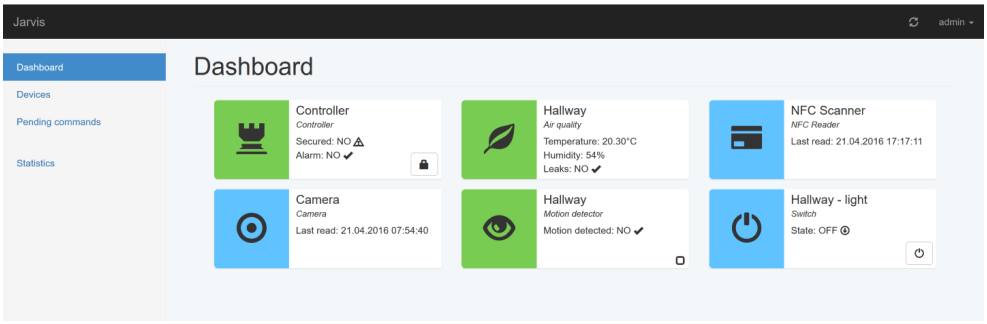
Vytvoření fotografie probíhá pomocí metody `capture_photo`, která pro získání snímku využívá rozhraní `picamera`. Snímek je uložen do úložiště, které je definované volbou `photo_storage` v tabulce `config`. Název fotografie je složen z aktuálního data a času ve formátu `YYYY-MM-DD-HH-MM-SS.jpg`.

Volba rozlišení fotografie probíhá uložením datového typu tuple do proměnné `resolution` ve tvaru (X, Y). Výchozí rozlišení fotografie je 1024x768 obrazových bodů. V případě pořízení fotografie v rámci hooku je nastaveno rozlišení na 1920x1080 obrazových bodů.

2.5 Monitoring pomocí HTTP protokolu

Pro implementaci monitoringu pomocí HTTP protokolu byl využit programovací jazyk PHP a framework Nette. Z Nette frameworku byla využita logika pro přihlašování uživatelů a je tedy možné tento monitoring využít i na LCD displeji kontroléru samotného a v případě, že uživatel nechce nechávat přístupná data po opuštění domácnosti, tak je možné se jednoduše odhlásit a data nebudou přístupná. Řešení využívá princip MVC. V rámci tohoto principu byl implementován prezentér `HomepagePresenter`. Tento prezentér má na starosti získávání veškerých dat a předávání je šablonám, které z nich vygenerují HTML kód. Dále byl implementován model s názvem `JarvisModel`, který má na starosti komunikaci s databází kontroléru a získávání dat. V rámci `HomepagePresenter` jsou definovány celkem čtyři views – `Default`, `Devices`, `Commands` a `Statistics`. Každý z těchto view zastřešuje stránku s jiným druhem informací. Pro podporu AJAXového řízení monitoringu jsou definovány celekm čtyři metody (nazvané signály dle terminologie Nette frameworku) – `handleTilesUpdate`, `handleRelayToggle`, `handleSecurityToggle` a `handleSecurityStatus`.

Prezenter `HomepagePresenter` slučuje veškeré views a signály v jeden celek. View `Default` slouží pro zobrazení základního dashboardu, kde může uživatel vidět aktuální stav systému. Tento stav se pomocí AJAXového volání signálu `handleTilesUpdate` periodicky obnovuje jednou za třicet sekund. Mimo obnovy dat se provede ještě dotaz na signál `handleSecurityStatus`, který určuje, zdali došlo k zastřežení objektu. Pokud ano, tak zobrazí překryvné okno s odpočtem do vlastního zastřežení. Toto ověření se spustí jednou za dvě sekundy.



Obrázek 2.4: Ukázka dashboardu

Veškeré AJAXové požadavky jsou řízeny pomocí rozšíření `nette.ajax.js`. Logika využívající toto rozšíření je naimplementována v souboru `main.js`. V rámci tohoto souboru je definována funkce `security_status`, která slouží pro řízení detekce zastřežení a zobrazení odpočtu pomocí funkce `security_countdown`. Dále jsou definovány funkce `refresh_tiles`, která slouží pro obnovu dat z monitoringu, `toggle_security`, která slouží k zastřežení respektive odstřežení objektu a `toggle_relay`, která slouží pro přepnutí aktivního zařízení.

Zobrazení dat se provádí pomocí objektů nazvaných dlaždice. Každá dlaždice obsahuje data o jednom zařízení. Jak se data zobrazí, definuje příslušná šablona ve formátu Latte, která vygeneruje HTML kód. Grafické zobrazení je definováno pomocí CSS v souboru `custom.css`.

Grafickou ukázku dashboardu je možné vidět na obrázku 2.4.

View `Devices` slouží pro zobrazení veškerých dat o zařízení v přehledné tabulce. View `Commands` zobrazí v tabulce seznam čekajících příkazů na provedení s možností jejich odstranění pomocí akce `actionDeleteCommand`. Poslední implementovaný view `Statistics` zobrazuje tabulku se statistickými údaji o celém systému.

Veškerá data pro views se získávají pomocí modelu `JarvisModel`, který má implementovány metody pro získávání dat z databáze pomocí třídy `Database`, která je implementována v Nette frameworku. Mezi tyto metody patří `get_tile`, `get_devices`, `get_pending_commands`, `get_controller_status` a další. Tyto metody fungují shodně – nejprve získají z databáze příslušná data, která pak převedou do asociativního pole a vrátí volající metodě. Jediné dvě metody, které mohou do databáze nějak zasahovat, jsou metody `enable_security` a `disable_security`. Tyto metody slouží pro zastřežení respektive odstřežení objektu po stisku příslušného tlačítka uživatelem v rámci webového rozhraní.

2.6 Monitoring pomocí SSH protokolu

Monitoring pomocí SSH protokolu slouží pro zobrazení dat ze všech dostupných senzorů a je zde možnost i zobrazit seznam zařízení definovaných v databázi. Řešení je uděláno modulárně podobně jako řešení démonu a je využito stejné třídy `Database` a obslužných tříd pro ORM peewee jako u relizace démona. Ve složce `Handlers` je základní třída `BaseHandler` od které ostatní obslužné třídy dědí. Tato třída definuje dvě metody – `print_data` a `get_friendly_type_name`. První metoda slouží pro vypsání dat ze zařízení na obrazovku a druhá pro vrácení jména modulu pro zobrazení uživateli. Všechny moduly tyto metody přetěžují. Mapování je uloženo v souboru `__init__.py` v proměnné `AVAILABLE_HANDLERS`.

Monitoring je ovládán pomocí třídy `MenuControl`, která slouží pro zobrazení menu uživateli, kde si uživatel pomocí volby čísla vybere akci, která se má stát. Tyto čísla jsou klíčem do slovníku `active_menu`, který překládá tyto čísla na názvy metod, které se při jeho zvolení provedou.

Po zvolení je zavolána metoda `process_menu`, která danou volbu přeloží a zavolá příslušnou funkci. Dostupné metody jsou `display_devices`, `show_devices_data` a `monitor_devices_data`.

Metoda `display_devices` získá seznam zařízení z databáze a postupně je vypíše v cyklu na obrazovku. Metoda `show_devices_data` získá zařízení s daty pomocí metody `get_devices_with_data` a ty následně vypíše na obrazovku s pomocí volání metody `print_data` jednotlivých obslužných modulů. Metoda `monitor_devices_data` provádí v cyklu `while` tu samou funkcionalitu, kterou provádí metoda `show_devices_data`. Tato metoda slouží pro kontinuální monitoring objektu a vypisování dat na konzoli pro uživatele.

Kapitola 3

Výsledky a testování

3.1 Testování

Testování výsledné implementace probíhalo převážně v ostrém provozu celého řešení. Nástroj byl nasazen na monitoring bytové jednotky a bylo sledováno jeho chování. K řešení byly připojeny dva senzory – senzor kvality ovzduší a detektor pohybu – a jedno aktivní zařízení pro rozsvěcení světel. Řešení bylo připojeno k WiFi síti a bylo monitorováno hlavně pomocí webového nástroje.

Nástroj byl dále testován pomocí jednotkových testů, kterými se ověřovala správná funkčnost metod důležitých pro komunikaci a správu databáze. V neposlední řadě proběhl test démona na náročnost na systémové prostředky. Mezi dva poslední testy se řadí test na výpadek senzoru a reakce na neoprávněné narušení prostoru.

3.2 Dosažené výsledky

V následujících kapitolách proberu jednotlivé testy, které byly provedeny. U každého testu bude popsána metoda testu a jeho výsledky.

3.2.1 Testování dosahu bezdrátového modulu NRF24l01+

Testování dosahu proběhlo v rámci kancelářské budovy, která je koncipována jako open space se zdmi ze sádrokartonu. Test probíhal mezi dvěma moduly, mezi kterými byly dvě stěny. V prostoru bylo přítomno několik WiFi sítí na frekvenci 2,4 GHz. Test probíhal tak, že byly nejprve nastaveny parametry modulu nRF24l01+ stejně na obou komunikujících stranách. Následně byly moduly postupně vzdalovány od sebe. Komunikace byla testována vysláním deset zpráv. Komunikace byla označena jako úspěšná, pokud dorazilo alespoň osm zpráv. Jakmile dorazilo méně zpráv, bylo poslední místo úspěšné komunikace označeno jako maximální dosah pro tyto parametry.

Bitrate	Vysílací výkon	Dosah
2 Mbps	MIN	18 m
2 Mbps	MAX	27 m
1 Mbps	MIN	22 m
1 Mbps	MAX	31 m
250 kbps	MIN	27 m
250 kbps	MAX	35 m

Tabulka 3.1: Testování dosahu bezdrátového modulu NRF24l01+

Během testování se ukázalo, že modul potřebuje nějaký čas mezi přepnutím mezi módy pro posílání dat a poslouchání pro příchozí data. Oboustrannou komunikaci se podařilo spolehlivě navázat až při vložení zpoždění o velikosti 100 ms mezi odesláním a přijetím dat. Vzhledem k nutnému času pro výpočet a získání dat doporučuji toto zpoždění nastavit na 300 ms. Tato doba by měla poskytnout dostatečnou dobu pro komunikaci a zpracování příkazu na adresátovi komunikace.

Mezi parametry, které byly testovány, patří bitrate a vysílací výkon. Pro minimalizaci kolizí byl zvolen poslední možný komunikační kanál, který je mimo standardní pásmo, které je využíváno standardem WiFi v rámci pásma 2,4 GHz. Vysílací výkon se nastavoval pomocí konstant definovaných v rámci knihoven pro komunikaci s modulem NRF24l01+.

Tabulka 3.1 shrnuje výsledky testování. Z těchto výsledků vyplývá, že pro největší dosah je výhodné nastavit bitrate na 250 kbps a nejvyšší možný vysílací výkon. Při tomto nastavení se dá dosáhnout komunikace na přibližně 35 metrů v běžné kancelářské budově se zdmi ze sádrokartonu.

3.2.2 Jednotkové testování

Jednotkové testy slouží k testování dílčích částí – jednotek – zdrojového kódu. V rámci své diplomové práce jsem testoval třídy `Database` a `Network`. Ostatní třídy jsou nějaký způsobem závislé na ostatních zařízeních. Tyto třídy byly testovány v rámci testů celého řešení. Jednotkové testy jsou uloženy v souboru `test.py` v kořenovém adresáři démona. Pro testování jsem byl využit modul `unittest`, který je standardní součástí interpretu jazyka Python.

Pro třídu `Database` se testují metody pro získávání dat z databáze. Chyby v těchto metodách by nemusely být vůbec viditelné, ale vedly by k nekorektnímu chování celého řešení. Testovány byly metody `get_devices`, `get_address_from_string`, `get_config_option`, `get_controller_data` a `get_tag_user`. Výstup z těchto metod je porovnán s očekávaným výstupem. Pokud nějaký výstup není shodný, tak je nahlášena chyba.

Pro třídu `Network` je testována metoda `prepare_payload` a `finalize_`

Test	První běh	Druhý běh	Třetí běh	Průměr
Test kontroléru – paměť	28,4 MB	28,4 MB	28,4 MB	28,4 MB
Test kontroléru a dvou senzorů – paměť	28,4 MB	28,4 MB	28,4 MB	28,4 MB
Test kontroléru a čtyř senzorů – paměť	28,4 MB	28,4 MB	28,4 MB	28,4 MB
Test kontroléru – CPU	5,3%	5,5%	5,5%	5,4%
Test kontroléru a dvou senzorů – CPU	10,6%	10%	10,4%	10,3%
Test kontroléru a čtyř senzorů – CPU	12,4%	12,7%	12,3%	12,5%

Tabulka 3.2: Test náročnosti na systémové prostředky

`connect_payload`, které slouží pro přípravu dat pro komunikaci pomocí modulu nRF24l01+. Výstupy jsou opět porovnány se vzorovým výstupem. V případě neshody je nahlášena chyba.

3.2.3 Testování náročnosti na systémové prostředky

Testování náročnosti na systémové prostředky proběhlo pomocí zaznamenávání aktuálního stavu každou sekundu běhu po dobu jedné minuty. Z těchto údajů byl poté spočítán aritmetický průměr. Byly provedeny tři druhy testů. První byl test kontroléru bez přítomnosti ostatních zařízení, druhý test byl za přítomnosti dvou senzorů a třetí test byl za přítomnosti čtyř senzorů. Každé testování proběhlo celkem třikrát. Testování začalo jednu minutu od nastartování démona, aby se do výsledku nezapočítávaly systémové prostředky nutné k nastartování celého řešení.

Tabulka 3.2 shrnuje výsledky tohoto testu.

Během testování samotného kontroléru bylo vidět, že démon nejprve vytíží na několik sekund CPU až skoro na 100%, ale poté vytížení rapidně klesá a ustálí se kolem 5%.

Z testu vyplývá, že paměťové nároky jsou stabilní a nijak se nemění s počtem připojených senzorů. Naopak vytížení CPU se mění s počtem senzorů znatelně. Největší rozdíl je patrný mezi samotným kontrolérem a dvěma senzory – rozdíl je v průměru 5%. Mezi dvěma a čtyřmi připojenými senzory je pak rozdíl již jen 2,2%. Naměřená čísla využití CPU jsou ale

stále v přijatelných mezích. Při připojení několika desítek senzorů by ale již problém mohl nastat a byla by nutná optimalizace kódu.

3.2.4 Reakce na výpadek senzoru

V rámci tohoto testu byl za běhu odpojen senzor kvality ovzduší od napájení. Kontrolér po šedesáti sekundách zareagoval vypsáním varovné hlášky a záznamem události do logu. Zařízení bylo nastaveno jako odpojené a kontrolér nadále nepředával řízení obslužnému modulu. Místo toho se snažil zařízení opět připojit. Jakmile bylo napájení opět připojeno, tak se povedlo zařízení kontaktovat a provoz se vrátil zpět do standardního režimu.

3.2.5 Reakce na neoprávněné narušení prostoru

V rámci tohoto testu byla otestována reakce kontroléru na neoprávněné narušení prostoru, které bylo zjištěno senzorem pro detekci pohybu.

Objekt byl nejprve zastřežen pomocí NFC karty. Po deseti sekundách, které má uživatel k opuštění objektu, byl prostor reálně zastřežen. Následně byl proveden pohyb před senzorem s pohybovým čidlem, který pohyb zaznamenal a při periodické kontrole tuto skutečnost odeslal kontroléru. Modul v kontroléru následně vydal příkaz k vyhlášení bezpečnostního poplachu. Na to zareagoval hook ovladače osvětlení, který odeslal zařízení příkaz o zapnutí osvětlení, které bylo v řádu jednotek vteřin zapnuto. Po deseti sekundách byl vyhlášen bezpečnostní poplach a byl odeslán email notifikující uživatele o této skutečnosti.

3.2.6 Celkové hodnocení testu

Výsledky testů indikují, že výsledné řešení je funkční, jelikož úspěšně prošel veškerými testy. Naměřené hodnoty náročnosti na systémové prostředky ukazují, že řešení je možné provozovat na jednodeskové platformě Raspberry Pi 3. Jediný problém může nastat s dosahem komunikace, která je maximálně 35 metrů v běžné moderní kancelářské budově. Nad tento limit klesá pravděpodobnost doručení zprávy na první pokus. Zde vidím možnost pro budoucí rozvoj tohoto řešení.

Řešení se ukázalo jako dostatečně odolné proti výpadku senzorů. Možnosti automatizace se prokázaly jako dostatečné pro běžnou automatizaci dle pravidel uživatele.

Závěr

Úkolem této diplomové práce bylo implementovat autonomní systém pro automatizaci a monitoring budov. Během implementace jsem se seznámil s komunikačním modulem nRF24l01+, jednodeskovými počítači Arduino a Raspberry Pi.

V první části práce byla provedena teoretická analýza problému a porovnání existujících řešení. V této části byla taktéž provedena analýza možností pro řešení daného problému a byl uveden návrh řešení.

Při návrhu řešení jsem kladl důraz na modulárnost celého řešení. Díky tomuto návrhu je možné toto řešení nadále v budoucnu jednoduše rozšiřovat – například přidání dalších senzorů či aktivních zařízení. V rámci implementace probíhalo ihned testování nově implementovaných funkcionalit. Při tomto testování se odhalilo nejvíce chyb v implementaci.

Testování se skládalo z několika nezávislých částí. Nejprve se celé řešení testovalo v ostrém provozu, kde se odhalilo největší množství chyb. Důležité částí kódu démona, které lze provozovat nezávisle, byly otestovány pomocí jednotkových testů. Tyto testy zajistí správné chování metod i po úpravách zdrojových kódů. Při těchto úpravách lze zanést do systému chybu a díky jednotkovým testům lze tuto chybu včas odhalit. Po úspěšném dokončení implementace byl proveden test náročnosti řešení na systémové prostředky. Toto testování prokázalo, že řešení je možné provozovat na jednodeskovém počítači Raspberry Pi 3. V poslední fázi testování byla ověřena reakce řešení na výpadek senzoru a byla ověřena reakce na neoprávněné narušení prostoru. Při testování nebyly objeveny žádné problémy.

Výsledkem této diplomové práce je funkční systém pro monitoring budov a automatizaci. Údaje monitoringu je možné zobrazit pomocí protokolů SSH a HTTP. Tento systém je nasazen do bytové jednotky, kde slouží pro monitoring stavu této jednotky včetně upozorňování na její neoprávněné narušení.

Budoucí práce

Implementované řešení se hodí pro menší objekty vzhledem k dosahu bezdrátových modulů nRF2401+. V budoucnu bych rád toto řešení rozšířil pomocí knihoven RF24Ethernet a RF24Mesh [43], které přináší podporu TCP/IP protokolu a podporu pro tvorbu mesh sítě. Takto postavená síť dovolí dynamické adresování a směrování dat skrze jednotlivé moduly a tedy je možné rozšířit dosah sítě pomocí ostatních modulů, které budou předávat data k centrální řídící jednotce. Dále by při využití této knihovny bylo vhodné implementovat komunikaci pomocí protokolu MQTT, jelikož tato knihovna efektivně řeší limit komunikace s maximálně šesti zařízeními najednou díky sestavení stromové topologie.

Dále bych rád rozšířil možnosti HTTP monitoringu, aby uživatel mohl dynamicky přidávat zařízení a upravovat nastavení chování systému za běhu a nemusel tyto parametry upravovat přímo v databázi.

V neposlední řadě bych rád implementoval zabezpečení komunikace mezi jednotlivými prvky, aby nemohlo dojít k jednoduchému napadení systému a neoprávněnému pozměňování dat a příkazů mezi řídící jednotkou a senzory.

Literatura

[1] Thakur, D.: *Bluetooth - What is Bluetooth?* [online]. [cit. 9. května 2016]. Dostupné z: http://ecomputernotes.com/computernetworkingnotes/communication-networks/bluetooth

[2] tilz0R: *Library 17- nRF24L01+ for STM32F4* [online]. Červen 2014 [cit. 9. května 2016]. Dostupné z: http://stm32f4-discovery.com/2014/06/library-17-nrf24l01-stm32f4xx/

[3] Raspberry Pi Foundation: *Raspberry Pi 2 on sale now at $35 - Raspberry Pi* [online]. Leden 2015 [cit. 9. května 2016]. Dostupné z: https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/

[4] Stuurman, N.: *Arduino* [online]. Prosinec 2012 [cit. 9. května 2016]. Dostupné z: https://micro-manager.org/wiki/Arduino

[5] pchan: *Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout* [online]. Leden 2015 [cit. 9. května 2016]. Dostupné z: https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout

[6] D-Robotics: DHT11 Humidity & Temperature Sensor. Červenec 2010. Dostupné z: http://www.micropik.com/PDF/dht11.pdf

[7] Learn about Electronics: *MQ-2 Smoke Sensor Circuit Built with an Arduino* [online]. [cit. 9. května 2016]. Dostupné z: http://www.learningaboutelectronics.com/Articles/MQ-2-smoke-sensor-circuit-with-arduino.php

[8] *HC-SR501 PIR MOTION DETECTOR* [online]. [cit. 9. května 2016]. Dostupné z: https://www.mpja.com/download/31227sc.pdf

[9] Sudip.S, M.: *Global Home Automation Market will be worth US$21.6 billion by 2020* [online]. Září 2015 [cit. 9. května

2016]. Dostupné z: http://www.transparencymarketresearch.com/pressrelease/home-automation-market.htm

[10] Darwin: *The Best of the Home Automation Controllers* [online]. Únor 2016 [cit. 9. května 2016]. Dostupné z: http://darwinsden.com/best-home-automation-controller/

[11] Comet, A. J.: *TESTED: Smart Home Automation – Comparing Apple HomeKit vs Samsung SmartThings Hub vs Wink Connected Home Hub vs INSTEON Hub Pro vs Amazon Echo vs Belkin WeMo* [online]. Listopad 2015 [cit. 9. května 2016]. Dostupné z: https://arijaycomet.com/2015/11/16/tested-smart-home-automation-comparing-apple-homekit-vs-samsung-smartthings-hub-vs-wink-connected-home-hub-vs-insteon-hub-pro-vs-amazon-echo-vs-belkin-wemo/

[12] home toys: *Nine Open Source Home Automation Projects* [online]. Říjen 2015 [cit. 9. května 2016]. Dostupné z: http://www.hometoys.com/article/2015/10/nine-open-source-home-automation-projects/32466

[13] Alois, J.: *SmartThings Raised $1.2 Million on Kickstarter, 2 Years Later The Company is Acquired by Samsung for $200 Million* [online]. Srpen 2014 [cit. 9. května 2016]. Dostupné z: http://www.crowdfundinsider.com/2014/08/47009-smartthings-raised-1-2-million-crow-on-kickstarter-2-years-later-company-has-been-acquired-by-samsung-for-200-million/

[14] Samsung: *What is the range of the SmartThings Hub?* [online]. [cit. 9. května 2016]. Dostupné z: https://support.smartthings.com/hc/en-us/articles/203061570-What-is-the-range-of-the-SmartThings-Hub-

[15] Dunn, W.: *Samsung SmartThings review* [online]. Prosinec 2015 [cit. 9. května 2016]. Dostupné z: http://www.stuff.tv/samsung/smartthings/review

[16] Samsung: *SmartThings Shop* [online]. [cit. 9. května 2016]. Dostupné z: https://shop.smartthings.com/#!/

[17] Belkin International: *WeMo® Insight Switch* [online]. 2016 [cit. 9. května 2016]. Dostupné z: http://www.belkin.com/us/F7C029-Belkin/p/P-F7C029/

[18] S, G. T.: *Belkin WeMo and Ubiquiti mFi Home Automation Platforms Review* [online]. Duben 2015 [cit. 9. května 2016]. Dostupné z: http://www.anandtech.com/show/9162/belkin-wemo-and-ubiquiti-mfi-home-automation-platforms-review

[19] Belkin International: *WeMo Home Automation* [online]. 2016 [cit. 9. května 2016]. Dostupné z: http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/

[20] Ubiquiti Networks: Machine-to-Machine Management System. 2014. Dostupné z: https://dl.ubnt.com/datasheets/mfi/mFi_DS.pdf

[21] DoubleRadius: *Ubiquiti mFi* [online]. [cit. 9. května 2016]. Dostupné z: http://www.doubleradius.com/Manufacturers/mFi-1/

[22] Home Assistant: *Home Assistant* [online]. [cit. 9. května 2016]. Dostupné z: https://home-assistant.io/

[23] Home Assistant: *Getting Started* [online]. [cit. 9. května 2016]. Dostupné z: https://home-assistant.io/getting-started/

[24] Home Assistant: *Components* [online]. [cit. 9. května 2016]. Dostupné z: https://home-assistant.io/components/

[25] Home Assistant: *Developers* [online]. [cit. 9. května 2016]. Dostupné z: https://home-assistant.io/developers/

[26] Bruce, J.: *Getting Started with OpenHAB Home Automation on Raspberry Pi* [online]. Září 2015 [cit. 9. května 2016]. Dostupné z: http://www.makeuseof.com/tag/getting-started-openhab-home-automation-raspberry-pi/

[27] openHAB UG: *openHAB - Features - User Interface* [online]. 2016 [cit. 9. května 2016]. Dostupné z: http://www.openhab.org/features/ui.html

[28] TheKorn2: *ImperiHome Binding* [online]. Duben 2016 [cit. 9. května 2016]. Dostupné z: https://github.com/openhab/openhab/wiki/ImperiHome-Binding

[29] Mens, J.-P.: *A story of home automation with openHAB, Z-Wave, and MQTT* [online]. Leden 2014 [cit. 9. května 2016]. Dostupné z: http://jpmens.net/2014/01/14/a-story-of-home-automation/

[30] Domoticz: *Automation* [online]. [cit. 9. května 2016]. Dostupné z: http://www.domoticz.com/wiki/Automation

[31] Peter, R. E.: Domoticz - Open Source Home Automation System. Únor 2015. Dostupné z: http://www.domoticz.com/DomoticzManual.pdf

[32] Noren, D.: *dcnoren/Domoticz-Home-UI: UI for Domoticz, geared toward tablet and mobile web support. Also enables tieing together events and event processing, as well as event logging.* [online]. Prosinec 2015

[cit. 9. května 2016]. Dostupné z: https://github.com/dcnoren/Domoticz-Home-UI

[33] Digi International: *ZigBee® Wireless Standard* [online]. [cit. 9. května 2016]. Dostupné z: http://www.digi.com/resources/standards-and-technologies/rfmodems/zigbee-wireless-standard

[34] Čepička, D.: *Základy technologie Bluetooth: původ a rozsah funkcí* [online]. Únor 2009 [cit. 9. května 2016]. Dostupné z: http://pcworld.cz/hardware/Zaklady-technologie-Bluetooth-puvod-a-rozsah-funkci-6635

[35] Mitrega, M.: *Bluetooth 4.0 přichází s dosahem 100 metrů* [online]. Červenec 2010 [cit. 9. května 2016]. Dostupné z: http://pctuning.tyden.cz/component/content/article/1-aktualni-zpravy/18112-bluetooth-4-0-prichazi-s-dosahem-100-metru

[36] Jimb0: *Bluetooth Basics* [online]. Srpen 2013 [cit. 9. května 2016]. Dostupné z: https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works

[37] Arduino8.cz: *Lekce 20 - Arduino a využití 433,92 MHz* [online]. Červenec 2015 [cit. 9. května 2016]. Dostupné z: http://www.arduino8.cz/lekce-20-arduino-a-vyuziti-43392-mhz/

[38] C, S.: *433 MHz RF module with Arduino Tutorial 1* [online]. Červen 2014 [cit. 9. května 2016]. Dostupné z: http://arduinobasics.blogspot.cz/2014/06/433-mhz-rf-module-with-arduino-tutorial.html

[39] Nordic Semiconductor: *nRF24L01+ / 2.4GHz RF / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR* [online]. [cit. 9. května 2016]. Dostupné z: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P

[40] Havránek, M.: *Transceiver NRF24L01 - Základy použití* [online]. Březen 2013 [cit. 9. května 2016]. Dostupné z: http://hawwwran.launchpad.cz/clanky/transceiver-nrf24l01-zaklady-pouziti-46.html

[41] King, T.: *arduino-info - Nrf24L01-2.4GHz-HowTo* [online]. Březen 2016 [cit. 9. května 2016]. Dostupné z: https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo

[42] Chantrell, N.: *Experimenting with the nRF24L01+ 2.4GHz radios* [online]. Srpen 2013 [cit. 9. května 2016]. Dostupné z: https://nathan.chantrell.net/20130810/experimenting-with-the-nrf24l01-2-4ghz-radios/

[43] TMRh20: *Tmrh20.github.io* [online]. Prosinec 2015 [cit. 9. května 2016]. Dostupné z: http://tmrh20.github.io/

[44] Calin, D. G.: *Compare 10 Linux single-board computers to find the right one for you* [online]. 2015 [cit. 9. května 2016]. Dostupné z: http://www.intorobotics.com/compare-10-linux-single-board-computers-to-find-the-right-one-for-you/

[45] Intel Corporation: *Intel® Edison—One Tiny Module, Endless Possibility* [online]. [cit. 9. května 2016]. Dostupné z: http://www.intel.com/content/www/us/en/do-it-yourself/edison.html

[46] Arduino: *Arduino - PWM* [online]. [cit. 9. května 2016]. Dostupné z: https://www.arduino.cc/en/Tutorial/PWM

[47] Čížek, J.: *Jak jsem rozblikal diodu na mikropočítači Intel Edison* [online]. Červenec 2015 [cit. 9. května 2016]. Dostupné z: http://www.zive.cz/clanky/jak-jsem-rozblikal-diodu-na-mikropocitaci-intel-edison/sc-3-a-178927/default.aspx

[48] Raspberry Pi Foundation: *Raspberry Pi FAQs - Frequently Asked Questions* [online]. [cit. 9. května 2016]. Dostupné z: https://www.raspberrypi.org/help/faqs/

[49] Raspberry Pi Foundation: *Raspberry Pi 3 Model B - Raspberry Pi* [online]. Únor 2016 [cit. 9. května 2016]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[50] Raspberry Pi Foundation: *Raspberry Pi Zero - Raspberry Pi* [online]. [cit. 9. května 2016]. Dostupné z: https://www.raspberrypi.org/products/pi-zero/

[51] Arduino: *Arduino - Introduction* [online]. [cit. 9. května 2016]. Dostupné z: http://www.arduino.cc/en/Guide/Introduction

[52] Ježek, A.: *1. díl - Seznámení s Arduinem* [online]. 2015 [cit. 9. května 2016]. Dostupné z: http://www.itnetwork.cz/hardware-pc/arduino/arduino-seznameni

[53] Tůma, O.: *Arduino - programování v čistém C(++)* [online]. Listopad 2011 [cit. 9. května 2016]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1877

[54] Arduino: *Arduino - Products* [online]. [cit. 9. května 2016]. Dostupné z: http://www.arduino.cc/en/Main/Products

[55] Arduino: *Arduino - ArduinoBoardMega2560* [online]. [cit. 9. května 2016]. Dostupné z: http://www.arduino.cc/en/Main/ArduinoBoardMega2560

[56] Arduino: *Arduino - ArduinoBoardUno* [online]. [cit. 9. května 2016]. Dostupné z: `https://www.arduino.cc/en/main/arduinoBoardUno`

[57] Igor: *Arduino Power Consumption Normal & Sleep* [online]. Září 2013 [cit. 9. května 2016]. Dostupné z: `http://gadgetmakersblog.com/arduino-power-consumption/`

[58] Arduino: *Arduino - ArduinoShields* [online]. [cit. 9. května 2016]. Dostupné z: `https://www.arduino.cc/en/Main/arduinoShields`

[59] International Organization for Standardization: ISO27002. 2005. Dostupné z: `http://demo.protocolpolicy.com/ISO27002index.html`

[60] RFID portál: *Co je RFID* [online]. [cit. 9. května 2016]. Dostupné z: `http://www.rfidportal.cz/index.php?page=rfid_obecne`

[61] Chandler, N.: *What's the difference between RFID and NFC?* [online]. Březen 2012 [cit. 9. května 2016]. Dostupné z: `http://electronics.howstuffworks.com/difference-between-rfid-and-nfc.htm`

[62] Drhlík, J.: *Bezdrátové technologie: Co je NFC a jak ho využít?* [online]. Červenec 2015 [cit. 9. května 2016]. Dostupné z: `http://www.svetandroida.cz/bezdratove-technologie-nfc-201507`

[63] Ondra, P.: *NFC – aneb jednoduchý štítek, který umí...* [online]. Listopad 2012 [cit. 9. května 2016]. Dostupné z: `http://www.geekblind.cz/?p=26`

[64] Conty, R.: *NFC Tools* [online]. Červen 2013 [cit. 9. května 2016]. Dostupné z: `http://nfc-tools.org/index.php?title=Main_Page`

[65] Cunningham & Cunningham: *Scripting Language* [online]. Srpen 2014 [cit. 9. května 2016]. Dostupné z: `http://c2.com/cgi/wiki?ScriptingLanguage`

[66] Pospíšil, D.: *Centralizovaný monitoring RouterOS sítových prvků*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

[67] Python Software Foundation: *Python 3.4.4 documentation* [online]. Březen 2016 [cit. 9. května 2016]. Dostupné z: `https://docs.python.org/3.4/`

[68] Summerfield, M.: *Python 3 Výukový kurz*. Computer Press, 2011, iSBN 9788025127377.

[69] Lavery, B.: *BLavery/lib_nrf24: Python library for NRF24L01+ Transceivers* [online]. Listopad 2014 [cit. 9. května 2016]. Dostupné z: https://github.com/BLavery/lib_nrf24

[70] Leifer, C.: *peewee* [online]. Březen 2016 [cit. 9. května 2016]. Dostupné z: http://docs.peewee-orm.com/en/latest/

[71] Mozilla Developer Network and individual contributors: *About JavaScript* [online]. Říjen 2015 [cit. 9. května 2016]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[72] World Wide Web Consortium: *A Short History of JavaScript* [online]. Červen 2012 [cit. 9. května 2016]. Dostupné z: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

[73] Refsnes Data: *jQuery Introduction* [online]. [cit. 9. května 2016]. Dostupné z: http://www.w3schools.com/jquery/jquery_intro.asp

[74] PHP Documentation Group: *PHP Manual* [online]. Květen 2016 [cit. 9. května 2016]. Dostupné z: http://php.net/manual/en/

[75] World Wide Web Consortium: HTML5. Technická zpráva, World Wide Web Consortium, Říjen 2014. Dostupné z: https://www.w3.org/TR/html5/

[76] World Wide Web Consortium: HTML 5.1. Technická zpráva, World Wide Web Consortium, Květen 2016. Dostupné z: https://www.w3.org/TR/2016/WD-html51-20160503/

[77] Rohde, M.: *HTML 5 Features: What Web Developers Can Use NOW* [online]. [cit. 9. května 2016]. Dostupné z: http://www.htmlgoodies.com/html5/tutorials/HTML-5-Features-What-Web-Developers-Can-Use-NOW-3880701.htm

[78] World Wide Web Consortium: CSS Snapshot 2015. Technická zpráva, World Wide Web Consortium, Říjen 2015. Dostupné z: https://www.w3.org/TR/css-2015/

[79] Data, R.: *AJAX Introduction* [online]. [cit. 9. května 2016]. Dostupné z: http://www.w3schools.com/php/php_ajax_intro.asp

[80] Mozilla Developer Network and individual contributors: *XMLHttpRequest* [online]. Květen 2016 [cit. 9. května 2016]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

[81] Nette Foundation: *Seznámení s Nette Frameworkem* [online]. Leden 2016 [cit. 9. května 2016]. Dostupné z: https://doc.nette.org/cs/2.3/getting-started

[82] Skvorc, B.: *The Best PHP Framework for 2015: SitePoint Survey Results* [online]. Březen 2015 [cit. 9. května 2016]. Dostupné z: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

[83] Nette Foundation: *MVC aplikace & presentery* [online]. Únor 2016 [cit. 9. května 2016]. Dostupné z: https://doc.nette.org/cs/2.3/presenters

[84] Nette Foundation: *Dependency Injection* [online]. Leden 2016 [cit. 9. května 2016]. Dostupné z: https://doc.nette.org/cs/2.3/dependency-injection

[85] Rouse, R.: *What is database? - Definition from WhatIs.com* [online]. Duben 2006 [cit. 9. května 2016]. Dostupné z: http://searchsqlserver.techtarget.com/definition/database

[86] Rouse, M.: *database management system (DBMS)* [online]. Leden 2015 [cit. 9. května 2016]. Dostupné z: http://searchsqlserver.techtarget.com/definition/database-management-system

[87] solid IT gmbh: *DB-Engines Ranking* [online]. Květen 2016 [cit. 9. května 2016]. Dostupné z: http://db-engines.com/en/ranking

[88] Buytaert, D.: *The history of MySQL AB* [online]. Březen 2010 [cit. 9. května 2016]. Dostupné z: http://buytaert.net/the-history-of-mysql-ab

[89] Rackspace Support: *MySQL Engines - MyISAM vs Innodb* [online]. Leden 2016 [cit. 9. května 2016]. Dostupné z: https://support.rackspace.com/how-to/mysql-engines-myisam-vs-innodb/

[90] Barwick, I.: *Oracle's Acquisition of InnoDB: What does it mean?* [online]. Duben 2006 [cit. 9. května 2016]. Dostupné z: http://sql-info.de/mysql/notes/oracle-acquires-innodb.html

[91] Štrauch, A.: *Databáze MariaDB válcuje MySQL* [online]. Prosinec 2012 [cit. 9. května 2016]. Dostupné z: http://www.root.cz/clanky/databaze-mariadb-valcuje-mysql/

[92] Foundation, M.: *About MariaDB* [online]. [cit. 9. května 2016]. Dostupné z: https://mariadb.org/about/

[93] Oy, S.: Leden *2015 [cit. 9. května 2016]*, title = 10 reasons to migrate to MariaDB (if still using MySQL) [online]. Dostupné z: https://seravo.fi/2015/10-reasons-to-migrate-to-mariadb-if-still-using-mysql

[94] Rouse, M.: *object-relational mapping (ORM)* [online]. Únor 2008 [cit. 9. května 2016]. Dostupné z: http://searchwindevelopment.techtarget.com/definition/object-relational-mapping

[95] Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen and Markku-Juhani O. Saarinen: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, Internet Engineering Task Force, Leden 2006. Dostupné z: https://tools.ietf.org/html/rfc4253

[96] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, W3C/MIT: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, Červen 1999. Dostupné z: https://tools.ietf.org/html/rfc2616

[97] Raspberry Pi Foundation: *Camera Module* [online]. [cit. 9. května 2016]. Dostupné z: https://www.raspberrypi.org/products/camera-module/

[98] HANWEI ELETRONICS CO.,LTD: TECHNICAL DATA MQ-2 GAS SENSOR. Září 2011. Dostupné z: http://www.seeedstudio.com/wiki/images/3/3f/MQ-2.pdf

[99] SONGLE RELAY: RELAY ISO9002. [cit. 9. května 2016]. Dostupné z: https://www.ghielectronics.com/downloads/man/20084141716341001RelayX1.pdf

[100] Advanced Card Systems Ltd.: ACR122U USB NFC Reader. Dostupné z: http://www.acs.com.hk/download-manual/418/07-TSP-ACR122U-3.04.pdf

Příloha **A**

Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CSI Camera Serial Interface

CSS Cascading Style Sheets

DHCP Dynamic Host Configuration Protocol

DSI Display Serial Interface

GPIO General-purpose input/output

HTML HyperText Markup Language

IDE Integrated Development Environment

IFTTT If This Then That

IoT Internet of Things

JSON JavaScript Object Notation

MVC Model-View-Controller

NFC Near Field Communication

PHP Hypertext Preprocessor

RFID Radio Frequency Identification

SSH Secure Shell

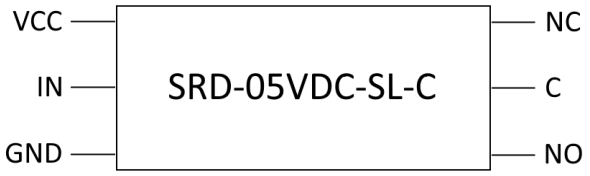
USB Universal Serial Bus

XHTML EXtensible HyperText Markup Language

Příloha B

Schémata zařízení a modulů

B.1 SRD-05VDC-SL-C

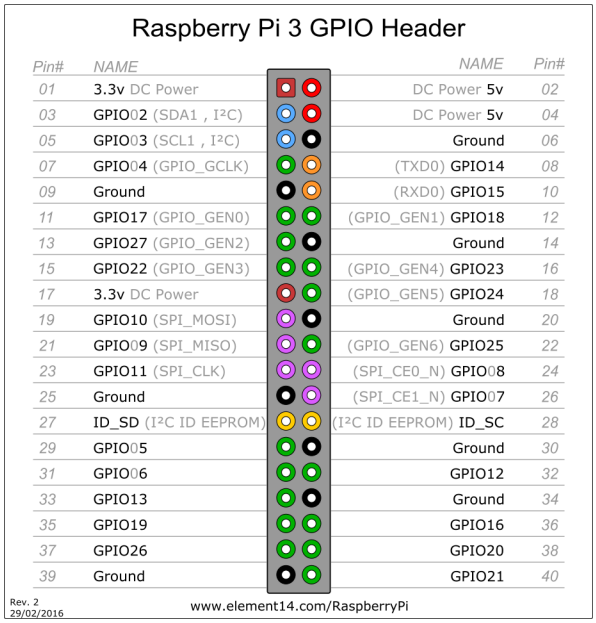


Obrázek B.1: Schéma modulu SRD-05VDC-SL-C

B.2 nRF24l01+

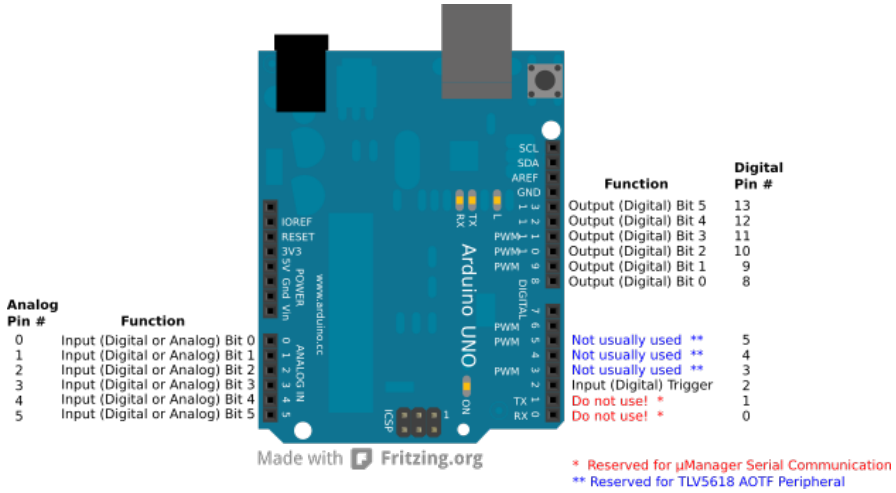
Obrázek B.2: Schéma modulu nRF24l01+ (převzato z [4])

B.3 Raspberry Pi 3 Model B



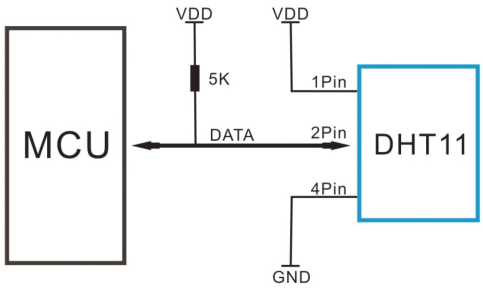
Obrázek B.3: Schéma GPIO platformy Raspberry Pi 3 Model B (převzato z [5])

B.4 Arduino Uno



Obrázek B.4: Schéma GPIO platformy Arduino Uno (převzato z [4])

B.5 DHT11

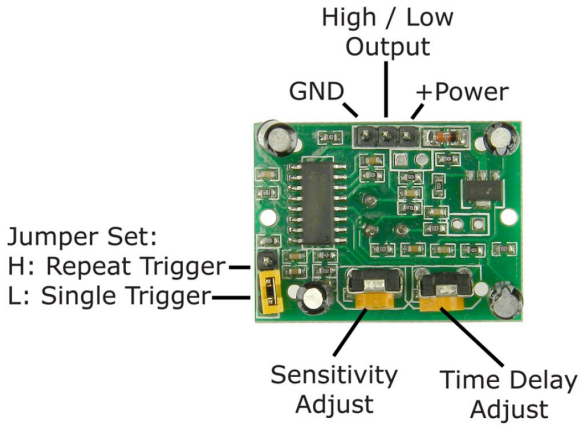


Obrázek B.5: Schéma modulu DHT11 (převzato z [6])

B.6 MQ-2

Obrázek B.6: Schéma modulu MQ-2 (převzato z [7])

B.7 HC-SR501



Obrázek B.7: Schéma modulu HC-SR501 (převzato z [8])

Příloha C

Instalace nástroje a použití

Instalace nástroje

C.0.1 Instalace démona a SSH monitoringu

Instalace démona a monitoringu pomocí protokolu je díky použitému programovacímu jazyku Python velice jednoduchá. Pro zprovoznění kontroléru je nutné zapojit veškerý hardware dle popisu v kapitole 2. Dále je nutné na počítač Raspberry Pi nainstalovat následující programy či knihovny:

- MariaDB
- Apache2
- PHP alespoň verze 5.6
- Python (doporučená verze 3.4)
- PHPMyAdmin
- libnfc alespoň verze 1.7.1 (je nutné zkompilovat, v době psaní této práce verze umístěná v oficiálním repozitáři obsahovala chybu, díky které řešení nefungovalo).

Nutné moduly pro jazyk Python 3:

- RPi.GPIO
- spidev
- peewee
- PyMySQL
- picamera

Moduly je možné nainstalovat pomocí nástroje pip3, který by měl být v systému po instalaci interpretu jazyka Python 3 dostupný. Tento nástroj stačí zavolat s parametrem `install` a názvem balíku pro jeho instalaci. V případě, že nějaký modul nebude dostupný, tak je možné veškerý potřebný software a uvedené moduly nalézt na přiloženém CD ve složce `Packages` v podobě zdrojových kódů.

Po instalaci potřebných modulů je nutné zkopírovat podložku `RPI` a `Monitor` ze složky `impl` do cílové složky odkud chcete řešení provozovat. Dále stačí pomocí nástroje PHPMyAdmin vytvořit databázi a do ní importovat obsah ze souboru `create.sql`, který je umístěn v podsložce Database ve složce `impl`. Poté je nutné v souboru `Models.py` nastavit přístupové údaje k databázi podle vašeho prostředí.

Jako poslední krok je nutné zajistit běh démona na pozadí. Je možné ho buď ručně zapnout a poslat na pozadí (je nutné využít příkaz `nohup` a přidat symbol `&` na konci příkazu). Další varianta je tvorba vlastního spouštěcího skriptu pro systemd či využití rc skriptů. Démon se spouští příkazem `python3 daemon.py`

C.0.2 Instalace webového monitoringu

Pro instalaci tohoto monitoringu stačí zkopírovat obsah podložky `Web` ze složky `impl` do cílové složky, kde chcete řešení provozovat. Je nutné nastavit parametr `DocumentRoot` webového serveru Apache, aby kořenem webové prezentace byl podadresář `www`. Poslední nutná úprava se týká souboru `config.neon`, kde je nutné nastavit přístupové údaje k databázi podle vašeho prostředí.

C.0.3 Instalace senzorů a zařízení

Po zapojení zařízení dle popisu v kapitole 2 – Realizace je nutné pomocí aplikace Arduino IDE nahrát příslušný kód senzoru či zařízení do jednodeskového počítače Arduino Uno. Implementace lze nalézt na přiloženém CD v podsložce Arduino složky `impl`. Pro nahrání je nejprve nutné do Arduino IDE importovat knihovnu `RF24`. Po nahrání je senzor respektive zařízení připraveno k provozu.

C.0.4 Použití nástroje

Po úspěšné instalaci je systém připraven pro použití a je možné si zobrazit veškerá dostupná data pomocí webového prohlížeče a systém využívat. Nastavení parametrů a vložení dalších zařízení a hooků je možné provést pomocí nástroje PHPMyAdmin. Syntaxi pro přidání zařízení je možné využít ze vzorového souboru `create.sql`. Pro zobrazení dat pomocí protokolu SSH stačí po přihlášení k systému spustit příkaz `python3 «cesta ke složce Monitor»/monitor.py`.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `Docs` ..... adresář s dokumentací
- `Packages` ..... adresář s potřebným software
- `Source`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `DS` ..... technické specifikace modulů ve formátu PDF
- `Text` ..... text práce
  - `DP_Pospisil_Daniel_2016.pdf` ..... text práce ve formátu PDF