Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Princip bezpečného vzdáleného ovládání zařízení pomocí mobilní aplikace

Bakalářská práce

Ondřej Filip

Školitel: Mgr. Jiří Pech, Ph.D.

České Budějovice 2017

Bibliografické údaje

Filip, O., 2017: Princip bezpečného vzdáleného ovládání zařízení pomocí mobilní aplikace. [Principle of secure and remote control device using the mobile application. Bc.. Thesis, in Czech.] - 85 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Bakalářská práce se zabývá navržením a popsáním principu zabezpečené bezdrátové komunikace sestaveného systému heterogenních zařízení s vyvinutou uživatelskou mobilní aplikací v kontextu využití v nezávislém prostředí domácí automatizace. Systém zařízení se skládá z modulárního výkonného prvku, který je realizován pomocí zařízeních od firmy Texas Instruments. Dále se skládá ze serverového prvku, který slouží pro lokální komunikaci s výkonným prvkem a pro externí komunikaci s mobilní aplikací. Mobilní aplikace disponuje základními funkcemi pro ovládání a zjišťování stavu aktivního prvku.

Klíčová slova

Internet věcí, Domácí automatizace, Zabezpečená komunikace, Wi-Fi, MQTT, Raspberry Pi, Texas Instruments, Mobilní aplikace

Annotation

The bachelor thesis deals with designing and describing the principle of secure wireless communication assembled system of heterogeneous devices with developed user mobile application in the context of the use in independent home automation environment. This system device consists of a modular executive unit, which is implemented by using devices from Texas Instruments. It also consists of a server unit, which is used for local communication with executive unit and for external communication with mobile application. Mobile application has basic functions for controlling and monitoring status of the executive unit.

Keywords

Internet of Things, Home automation, Secure communication, Wi-Fi, MQTT, Raspberry Pi, Texas Instruments, Mobile application

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne ……. Podpis autora …….

Poděkování

Děkuji Mgr. Jiřímu Pechovi, Ph.D. za cenné rady, odborný dohled a čas věnovaný vedení této práce. Dále děkuji své rodině za podporu v průběhu studia i při vypracování bakalářské práce.

Obsah

Úvod

Stav dnešní společnosti a lidského života se neustále vyvíjí a zejména v aktuální době rychlým způsobem mění. Žijeme v informační době, ve světě nekonečných možností, kde technologický vývoj otevírá dveře novým možnostem. Pomocí moderních technologií, jako je například internet, se můžeme virtuálně propojit s celým světem a nemusíme ani vstávat ze židle. Společnost se snaží odjakživa zlepšovat kvalitu svého života a zjednodušovat si zvládání každodenních činností. V současnosti technologický pokrok dospěl do revolučního věku Internetu věcí.

Internet věcí se stává jistě jedním z nejvýznamnějších novodobých komunikačních technologií, která začíná být součástí našeho každodenního života. Prudký nárůst zájmu o výzkum této nové technologie přispívá k rychlé evoluci naší informační společnosti. V současné době jsou prvky IoT[1] úspěšně aplikovány v různých oblastech činností. Použití a aplikace se neustále rozšiřuje do nových směrů. Použití IoT aplikací se může rozdělit na tři velké pilíře: industriální, spotřební a takzvaný Internet všeho (Internet of Everything).

Internet věcí ve spotřebním sektoru logicky vyvolal největší rozruch a pozornost, a to díky medializaci a vyvolání komerčnosti věci. Vyvolaný zájem o tyto technologie však nezpůsobila velká zainteresovanost uživatelů, ale široký rozsah nových možností aplikací a zařízení, která dorazila na trh. Důvodem je také cenová dostupnost komponentů vhodných právě pro typické IoT aplikace, umožňující lidem realizovat nové nápady a inovovat existující návrhy vlastní cestou. Do kategorie spotřebního sektoru IoT spadá například použití v inteligentních domácnostech, využití wearables[2] nebo jakékoliv další zefektivnění nebo zpříjemnění každodenního života.

Tato práce je zaměřena na vývoj a popsání principu zabezpečené komunikace sestaveného systému heterogenních zařízení s vyvinutou uživatelskou mobilní aplikací v kontextu využití v nezávislém prostředí domácí automatizace. Čtenář je seznámen s chronologickým postupem vývoje takového prostředí od analýzy teoretických východisek až po finální realizaci samotného systému zařízení.

[1] IoT (Internet of Things) - anglický překlad pro Internet věcí.
[2] Wearables – anglický překlad pro nositelnou elektroniku, např. chytré hodinky.

Toto téma práce jsem si vybral z důvodu vlastního zájmu o tuto problematiku. Navíc perspektiva a potenciál těchto technologií je v dnešní době nezměrná. Komerční systémy se v plném proudu dostávají na trh. Proto je pochopení této problematiky a následné vytvoření vlastního prototypu ekosystému, jehož části nejsou závislé na žádné ze služeb třetích stran lákavé.

Cíle práce

Hlavním cílem práce je navržení a popsání principu zabezpečené bezdrátové komunikace sestaveného systému heterogenních zařízení s vyvinutou uživatelskou mobilní aplikací. Tento systém zařízení se skládá z modulárního výkonného prvku a serveru. Uskutečnění tohoto cíle je rozděleno na dílčí kroky:

- Navržení komunikačního systému, který vychází z analýzy dostupných technologií využitelných při řešení IoT aplikací.
- Konfigurace a zabezpečení serveru který poskytuje realizaci bezdrátové lokální komunikace s výkonným prvkem a vzdálené komunikace s mobilní aplikací.
- Sestavení výkonného prvku na bázi zařízení Texas Instruments, který disponuje základním systémem senzorů a možností bezdrátové komunikace.
- Vyvinutí mobilní aplikace, která dokáže výkonný prvek vzdáleně a zabezpečeně ovládat a zjišťovat jeho stav.

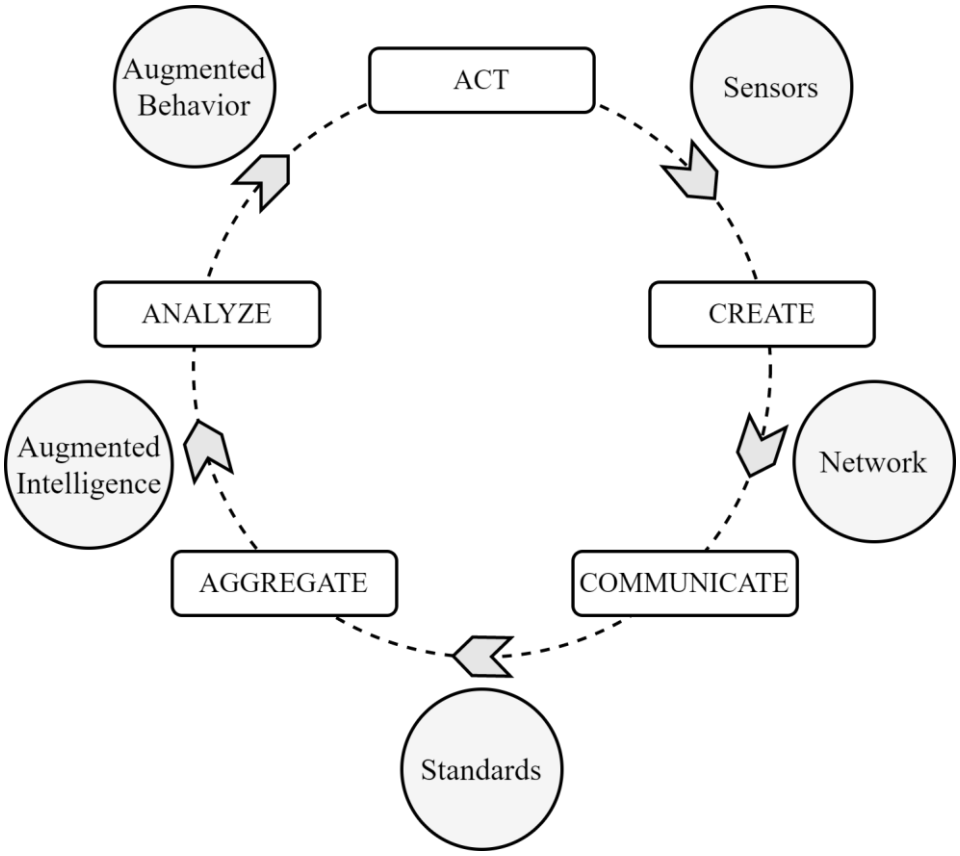
Metodika práce

Pro dosažení uvedených cílů je nejdříve využito teoretických metod. Je provedena analýza současného stavu problematiky Internetu věcí a chytré domácnosti. Výchozím bodem je rozbor dostupných a umožňujících technologií nezbytných při aplikaci ve vývoji IoT řešení. Rozbor se zaměřuje na porovnání specifických komunikačních systémů, které vystihují požadavky zadání této práce. Rešeršní šetření je provedeno kompletně v elektronické podobě. Hlavní motivací autora je volba zpracování této práce pouze za pomoci open-source projektů a zároveň tak vybudování soběstačného systému, který není závislý na využívání služeb třetích stran.

1 Internet věcí

1.1 Vymezení pojmu

V roce 1988 Mark Weiser popsal termín „ubiquitous computing“ svět, ve kterém by objekty všeho druhu uměly cítit, komunikovat, analyzovat, jednat nebo reagovat na lidi a jiné stroje autonomně, a to způsobem bez jakéhokoliv rušivého nebo pozoruhodného elementu, než jak jsme v současné době zvyklí na pouhé zapnutí světla nebo otevření kohoutku [1]. Jeden způsob, jak zachytit proces implicitně ve Weiserově modelu je pomocí tzv. smyčky hodnoty informace-viz převzatý a upravený obrázek (Obrázek 1-1). Struktura této práce je inspirována posloupností této smyčky.



Obrázek 1-1: Vizualizace smyčky hodnoty informace [2]

Samotná fráze „Internet věcí“ není vůbec žádnou novinkou, už v roce 1999 se o tomto termínu jako první zmínil Kevin Ashton ve své prezentaci o nových možnostech technologie RFID [3]. V této nedávné minulosti ovšem takové myšlenky předbíhaly možnosti

technologického vývoje. Dnešním dnem už jsou tyto nároky pokořené, a tak umožňující technologie pro vývoj IoT aplikací jsou globálně dostupné.

Pro jednoznačné vytváření definice této novodobé technologie je však stále ještě brzy. Je to stále vyvíjející se technologie, expandující do nových směrů a každý na to může hledět jinou perspektivou, není tudíž lehké pro ni určit ustálený význam. Je to v samotné podstatě evoluční etapa internetu. Termín „Internet věcí“ může být definován jako síť propojených zařízení, která jsou jednoznačně adresovatelná s tím, že tato síť je založena na standardizovaných komunikačních protokolech umožňující výměnu, sdílení dat a informací, jejichž analýzou je možné docílit vyšší přidané hodnoty [4]. Tato zařízení spojuje do stejné množiny několik klíčových podobností. Je to vhodnost pro chytré hospodaření s elektrickou energií, datové uložiště a správu systému nebo potřebu schopnosti komunikace. Dále například přítomnost výkonných mikrořadičů, snímačů a aktuátorů a v neposlední řadě i potřeba silnější záruky pro ochranu soukromí a bezpečnosti.

1.2 Inteligentní domácnost

Hlavní zaměření této práce je usměrněno na problematiku odvětví spotřebního sektoru Internetu věcí, konkrétněji na automatizaci inteligentní domácnosti. Inteligentní domácnost nebo inteligentní dům zajišťuje optimální vnitřní prostředí pro komfort osob prostřednictvím stavební konstrukce, techniky prostředí, řídících systémů, služeb a managementu. Je efektivní ekonomicky, energeticky i z hlediska působení na vnější prostředí a umožňuje víceúčelové použití a rekonfigurace. Automatizace domácnosti reaguje na potřeby obyvatel s cílem zvýšit jejich pohodlí, zpříjemnit život, zaručit co nejvyšší bezpečí a snížit náklady na provoz samotné domácnosti. [5]

Existuje spousta komerčních platforem softwaru i hardwaru, které se dají koupit na trhu pro automatizaci domova. Existují také i open source vývojářské prostředí a platformy se kterými si každý znalejší v oboru dokáže pohrát a vytvořit si tak vlastní domácí automatizaci, mezi zmíněné prostředí patří například Node-RED, IFTTT, Blynk, OpenHAB, Home Assistant nebo Domoticz. Tato práce se ovšem zabývá vyvinutím vlastní mobilní aplikace, která poslouží jako testovací platforma vyvinutá speciálně pro požadavky tématu práce. Tato aplikace by se měla teoreticky přiblížit stejné funkcionalitě, jako některé z výše zmíněných platforem.

1.3 Úskalí technologie

V této kapitole jsou probrány úskalí a výzvy Internetu věcí společně s automatizací běžné domácnosti, explicitně jsou vybrány jen problémy, které souvisí s náplní zadané práce. Těmito problémy je vývoj práce podněcován k obezřetnosti.

1.3.1 Zabezpečení a ochrana soukromí

Myšlenka Internetu věcí je připojit určité objekty, aby byl proces efektivnější, poskytnul větší komfort nebo zlepšil naší práci a osobní život v jakémkoliv slova smyslu. Ale připojovat objekty jako auta, domovy a stroje také odhaluje velké množství soukromých a citlivých dat. Existují různé druhy dat, které nejsou určeny pro veřejnost a měla by být chráněna pilíři informační bezpečnosti: utajení, integrita a dostupnost. Je zde také další pohled na věc – když jsou kompromitovány skutečné stroje nebo věci, útočník může škodit skutečným lidem. Například na dálku deaktivované brzdy u jedoucího auta nebo otevření zabezpečených dveří. Nebo - i když není definována žádná skutečná osoba, odhalení citlivých dat může poškodit dobré jméno jakékoliv společnosti. A tak se stálým vytvářením nových a nových zařízení a produkováním více a více dat, která jsou shromažďována každý den různými korporátními společnostmi, tady vznikají etické a existenční otázky. Bezpečnost a ochrana soukromí jsou témata, která jsou rozhodující a důležitější než kdy jindy.

Nemělo by být žádných pochyb o tom, zda by zabezpečení mělo být implementováno nebo ne, ale Internet věcí přináší nové výzvy. Tyto výzvy se ještě více stupňují jednoduše proto, že existuje více vstupních bodů nebo zařízení s omezeným výpočetním výkonem a kapacitou paměti a tím se vytváří více potenciálních penetračních bodů pro útočníka. Proto není lehké využití šifrovacích algoritmů, které potřebují více systémových zdrojů, než která mají tato omezená zařízení. Další markantní výzvou je samotná aktualizace zařízení. K dispozici je často nespolehlivé připojení a bezpečnostní problémy jsou nutné opravit aktualizacemi, což může být obtížné při zavádění do všech zařízení najednou. Navíc je výzvou z hlediska bezpečnosti být stále intuitivní a atraktivní pro uživatele, protože přijetí nových uživatelů závisí na snadné instalaci a údržbě. Z toho všeho vyplývá, že implementace zabezpečení by měla být přítomna od samotného počátku vývoje IoT aplikací. [6]

1.3.2 Interoperabilita otevřených standardů

Systém Internetu věcí se skládá z mnoha jednotlivých zařízení a platforem a každý s vlastními specifikacemi. Některé vrstvy IoT technologií postrádají standardy a některé zase mají nespočet konkurenčních standardů bez jasného vítěze, některé standardy definují jednu síťovou vrstvu a některé definují celé end-to-end specifikace. Dříve nebo později se situace dostane do bodu, kde všechna inteligentní zařízení budou spolu vzájemně komunikovat nebo spolupracovat s jinými systémy. Vývoj Internetu věcí lze analogicky přirovnat k samotnému vývoji internetu, několik let trvalo, než se stav technologií víceméně stabilizoval. Přesto i když je velice pravděpodobné, že vývoj IoT aplikací bude realizován za pomoci open source prostředků, univerzální standardy a protokoly zatím zaostávají za vývojem inteligentní technologie. Těch několik málo úsilí, která existují bývají specifická pro technologie a mají tendenci se soustředit na uplatňováni stávajících norem nebo protokolů inteligentních zařízení, spíše, než aby vyvíjeli činnost pro nové požadavky IoT. Řešení této výzvy je hlavním cílem pověřených organizací jako např. IEEE, které definují specifikace a zkušební postupy, jejichž cílem je zajištění interoperability. Interoperabilita představuje schopnost různých systémů a dodavatelů vzájemně komunikovat, dosáhnout vzájemné součinnosti. Bez nějakého většího stupně spolupráce, může být růst vývoje Internetu věcí pomalejší, než se zdá. [7]

Aktuálně jedním z protagonistů vize otevřenosti IoT projektů a řešení je organizace Eclipse IoT. Eclipse IoT je ekosystém společností a jednotlivců, kteří společně pracují na vytvoření Internetu věcí založeným na otevřených technologií. Eclipse IoT poskytuje velké spektrum technologií pro samotné omezené zařízení až po cloudové platformy. [8]

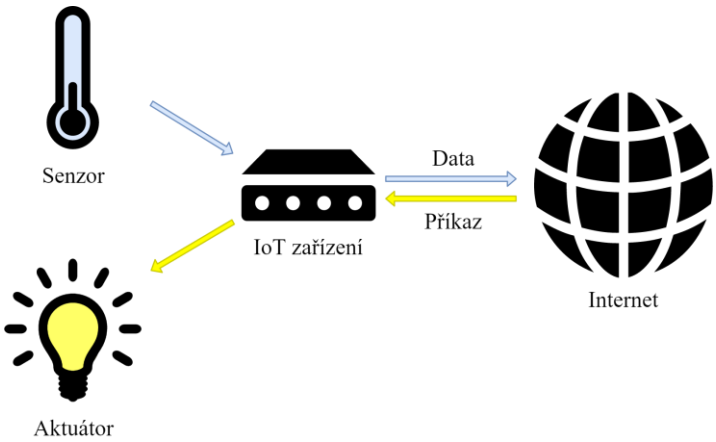
2 Teoretická východiska

Umožňujících použitelných technologií pro vyvíjení IoT aplikací je nespočet. V této kapitole je čtenář postupně seznámen s představením a porovnáním těchto technologií. Zvolené technologie jsou následně popsány podrobněji.

2.1 Systémy senzorů a aktuátorů

Senzorické systémy jsou nedílnou součástí světa Internetu věcí. Tyto senzory představují jakési smyslové ústrojí hardwarových zařízení. Senzor je zařízení, jehož účelem je detekovat událost nebo změny ve svém fyzickém prostředí a poté poskytnout odpovídající výstup. Senzory mohou poskytovat různé typy výstupů, obvyklými případy to jsou elektrické nebo optické signály [9]. Data vyprodukovaná ze senzorů jsou pak elektronicky transformována jiným zařízením, do jisté hodnotné informace. Díky abnormálně rychlému vývoji technologií se stále zdokonalují v přesnosti a variací snímaní a také redukováním fyzické velikosti. Důvodem, proč se tato zařízení tak moc rozšířila, je jejich cenová dostupnost, která se razantně zlepšila.

Hlavním úkolem senzorických prvků je snímání skutečnosti a následné získání žádané informační hodnoty. Tato data se dále mohou sbírat, analyzovat anebo spouštět další události. Aktuátory neboli jinak řečeno akčními členy je označována jakási protistrana senzorů. Princip aktuátorů je naopak převod z informační hodnoty na část fyzickou [10]. Například se může jednat o mechanické, akustické nebo optické výstupy. Obrázek (Obrázek 2-1) představuje právě popsaný princip.



Obrázek 2-1: Vizualizace principu senzoru a aktuátoru

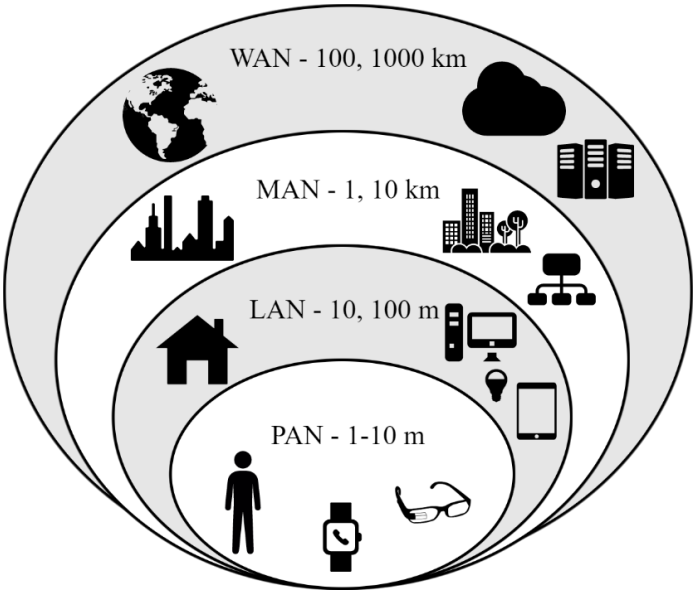
2.2 Počítačové sítě

Informace, které senzory vytvoří, málokdy dosahují své maximální hodnoty v čase a místě vzniku. Signály z čidel často musí být sdělena do jiných míst pro agregaci a analýzu. To typicky zahrnuje přenos dat po síti. Senzory a jiné zařízení jsou pak připojeny do sítí za využití síťových zařízení, jako jsou například brány, směrovače a přepínače, záleží na aplikaci.

První krok v procesu přenosu dat z jednoho zařízení do druhého přes síť je unikátní identifikování každého zařízení. K tomu slouží tzv. Internet Protocol (IP). IP je otevřený protokol, který poskytuje unikátní adresy do různých zařízení připojených do internetu. V současné době existují dvě verze IP a tím je IP verze 4 (IPv4) a IP verze 6 (IPv6). Dnešním datem jsou oficiálně IPv4 o kapacitě přibližně 2^{32} unikátních adres všechny vyčerpány. Dnes se díky různým metodám poskytovatelů a obchodováním s adresami drží IPv4 při životě. Bohužel příchod IPv6 o kapacitě 2^{128} unikátních adres není tak idylický, jak by se zprvu zdálo. Zpomalené přijetí IPv6 prostředí je otázkou ekonomické až politické záležitosti. Rovněž v kontextu IoT, některá zařízení s omezenými výkonnými prostředky mohou mít problém s kompatibilitou do takového prostředí. Vzhledem k situaci, že počet zařízení připojených do internetu se odhaduje na 26 miliard k poslednímu dni roku 2016 a předpokládaný nárůst na 50 miliard do roku 2020 se tak tudíž přijetí IPv6 adresace stane nutností pro následující roky. [11]

2.2.1 Rozsah sítě

Volba síťových technologií také do značné míry závisí na zeměpisném rozpětí, které má být pokryto. Když data mají být převedena v krátkých vzdálenostech jako například jedna místnost, zařízení mohou používat technologie personální sítě (PAN), jako je Bluetooth a ZigBee, nebo pomocí drátových technologií jako USB. Bezdrátové zařízení personální sítě mají obvykle nízký rádiový vysílací výkon a postačí jim jen malé baterie. Když je ovšem vyžadován datový přenos přes relativně větší prostor jako jsou například byty a rodinné domy, zařízení můžou používat technologie lokální sítě (LAN). Technologie lokální sítě jsou buď drátové nebo bezdrátové (nebo kombinace těchto dvou). Typický dosah těchto sítí je do 100 metrů. Jako příklady lokálních drátových technologií je Ethernet. Bezdrátové LAN (WLAN) zahrnuje technologie, jako je Wi-Fi která nejčastěji poskytuje připojení osobním počítačům a chytrým mobilním telefonům. Pokud data mají být přenesena přes širší oblast v rámci budov a měst, je použit systém WAN, který se zřizuje připojováním lokálních sítí pomocí směrovačů. Infografika viz (Obrázek 2-2) dotváří představu možností co se týče typických rozsahů sítí.[7]



Obrázek 2-2: Rozdělení počítačových sítí podle rozlehlosti

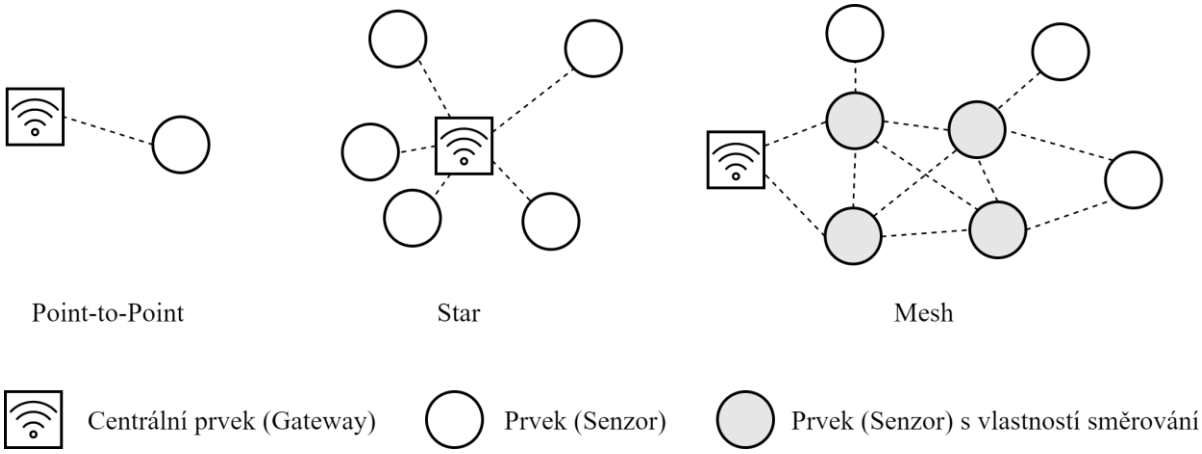
2.2.2 Topologie a velikost sítě

Architektury počítačové sítě mohou být také kategorizovány pomocí jejich topologie. Zapojením různých prvků do počítačových sítí a zachycením jejich reálné a logické podoby. Výběr správné topologie je zásadní faktor, který může ovlivnit složitost a robustnost celého systému. V kontextu práce jsou vybrány tři typické síťové topologie – Point-to-Point, Star a Mesh viz (Obrázek 2-3). Point-to-Point je velmi jednoduchá topologie, která navazuje přímé spojení pouze mezi dvěma prvky. Tento typ se nejvíce hodí pro dočasné sdílení dat. Například pomocí Bluetooth spojení mobilního telefonu a sluchátek. V Star topologii, všechny prvky jsou připojeny do centrálního bodu, který je typicky používán jako brána do internetu. Oblíbeným příkladem Star topologie je Wi-Fi síť, kde centrální prvek je přístupový bod (AP[3]) a ostatní prvky jsou nazývány stanice. Tento typ topologie vyžaduje od centrálního bodu určitá výkonnostní minima, aby mohl zpracovávat dotazy od dalších stanic v reálném čase, naopak stanice nepotřebují pracovat, když nejsou požadovány. Nevýhoda je v centralizaci této topologie, když přestane fungovat centrální bod, tak stanice nic nezmůžou. V Mesh topologii, se prvek může připojit na více jiných prvků. Každý prvek se může chovat jako internetová brána. Populárním představitelem této topologie je technologie ZigBee. Mesh topologie jsou však více komplexní na navržení a prokazují zpoždění ve směrování paketů ze vzdáleného prvku. Výhodou je možné rozšíření rozsahu sítě přes více skoků, při zachování nízkého

[3] AP (Access Point) – anglický překlad pro Přístupový Bod. V kontextu technologie Wi-Fi je to zařízení nejčastěji směrovač, ke kterému se připojují klienti.

rádiového přenosu energie. Mohou také dosáhnout lepší spolehlivosti tím, že umožňují více cest pro přenos zpráv sítí.

Velikost sítě, nebo maximální počet současně připojených zařízení, je také důležitým aspektem v návrhu systému. Některé technologie, jako Bluetooth podporují až 20 připojení, ZigBee podporuje až tisíce možných připojení. [7]



Obrázek 2-3: Rozdělení počítačových sítí podle topologie

2.3 Síťové komunikační technologie a standardy

Tato etapa v procesu implementace zahrnuje součet všech činností s manipulací, zpracováním a ukládáním dat získaných senzorů. Tato agregace zvyšuje hodnotu dat zvýšením míry, rozsahu a frekvenci dostupných dat pro analýzu. Takové agregace lze dosáhnout pouze použitím různých standardů v závislosti na použité IoT aplikaci.

2.3.1 Bezdrátové technologie připojení

Síťové technologie jsou elementárně klasifikovány jako drátové nebo bezdrátové. Zadání této práce je zaměřeno na bezdrátovou komunikaci mezi zařízeními. S neustálým pohybem uživatelů a zařízení bezdrátové sítě poskytují pohodlí díky téměř nepřetržité konektivitě, zatímco kabelové spoje jsou stále užitečné pro relativně spolehlivé, zabezpečené a velkoobjemové sítě. V rychle se rozvíjejícím Internetu věcí, aplikace od osobní elektroniky po průmyslové stroje a senzory se dostávají většinou k bezdrátovému připojení do internetu. Pokrytí široké škály užití, v různém prostředí a servírování různorodých požadavků, nemůže žádný jediný standard pro bezdrátové připojení odpovídat všem naráz. S četnými standardy

používanými na trhu, rozptýlenými po celém spektru využitelnosti a při používání různých komunikačních protokolů, vybrání správné technologie není snadné. [11]

Bezdrátových komunikačních technologií použitelných pro IoT aplikace je značný počet. Z toho důvodu je selekce zúžena jen na bezdrátové technologie do maximálního rozsahu LAN, tedy kategorie krátkého dosahu, která je využitelná právě pro potřeby práce. Pro srovnání jsou vybrány nejvíce typické technologie v oblasti Internetu věcí a domácí automatizace.

BLE

BLE (Bluetooth Low Energy, také známé jako Bluetooth Smart) je nedávný dodatek k samotné specifikaci Bluetooth. Oproti klasickému Bluetooth je navržen pro nižší datové propustnosti a také signifikantně snižuje spotřebu energie s udržením dosahu typického Bluetooth. Hlavní využití této technologie v paradigmatu Internetu věcí je aplikace v osobním prostoru člověka jako jsou například hodinky pro fitness a kontrolu stavu zdravotní péče. Většina chytrých telefonů nativně podporuje tuto technologii, pro které to bylo také záměrem. V prosinci 2016 byla oznámena specifikace Bluetooth 5, která oproti své předešlé verzi zněkolikanásobila dosah, rychlost a kapacitu nespojovaných datových přenosů. Takovým posunem vývoje dosáhne zajisté Bluetooth své důležitosti v budoucím vývoji IoT aplikací. [12]

Wi-Fi

Wi-Fi je dnes bezpochyby nejvíce rozšířená bezdrátová technologie, zejména s ohledem na všudypřítomnost Wi-Fi infrastruktury ve většině domácích prostředí. To je také důvodem snadné integrace IoT aplikací. Skoro všechna běžná zařízení jako například chytré telefony, tabletové počítače nebo notebooky mají nativní podporu Wi-Fi. Proto použití Wi-Fi je přirozený krok v době Internetu věcí. Tato technologie nabízí velkou propustnost dat v rozsahu stovek Mb/s, což je v pořádku například pro přenos souborů, ale může to být příliš náročné pro mnoho IoT aplikací. Na tento problém odpovídá nový standard Wi-Fi HaLow, který byl představen v lednu 2016. Tento nový standard byl vytvořen pro podporu myšlenky Internetu věcí. Jeho cílem je proto připojit co nejvíce zařízení, na co nejdelší vzdálenost a při co nejmenších energetických nárocích za cenu menších datových přenosů. Z těchto důvodů Wi-Fi HaLow rozšiřuje svoji působnost pásmem s nižší frekvencí 900 Mhz. [13]

ZigBee

ZigBee je standardem platným od roku 2003 a vychází ze standardu IEEE 802.15.4[4]. Patří jako Bluetooth do sítí o rozlehlosti PAN. Je to technologie s nízkým výkonem využívaným nejčastěji v ad-hoc[5] bezdrátových mesh sítích. Tato technologie byla záměrně nadefinována pro jednodušší a levnější implementaci než u Bluetooth nebo Wi-Fi. ZigBee je široce využíván v aplikacích domácí automatizace, průmyslového řízení a lékařského sběru dat. [14]

Z-Wave

Z-Wave oproti právě zmíněním technologiím operuje v tzv. sub-1GHz pásmu, toto pásmo je tak odolné vůči rušení od technologií, který využívají klasické 2,4 GHz pásmo. Z-Wave podporuje klíčování frekvenčním posunem (FSK). Podobně jako ZigBee podporuje využití sítě typu mesh. Na rozdíl od ZigBee ovšem využívá pouze 8 bitové adresování, z čehož vyplývá větší limit v možném počtu použitelných zařízení. [15]

6LowPAN

6LowPAN (IPv6 over Low-Power Wireless Personal Area Networks) je síťová technologie nebo adaptační vrstva, která umožňuje efektivní přenos IPv6 paketů linkovými rámci, jako jsou ty definované standardem IEEE 802.15.4. [16]

Thread

Thread je bezdrátový protokol, který využívá potenciálu adresace technologií 6LoWPAN zároveň s technologiemi standardu IEEE 802.15.4 v mesh sítích. Thread je oproti ZigBee méně komplexní, Thread například nijak nedefinuje aplikační vrstvu. Technologie Thread byla vytvořena skupinou Nest (Google), existuje také open-source implementace nazývaná „OpenThread“. [17]

Srovnání typických parametrických specifikací výše uvedených systémů je uvedeno v následující tabulce viz (Tabulka 2-1).

[4] IEEE 802.15.4 – standard, který definuje operace „low-rate wireless personal area networks“. Definuje fyzickou a linkovou vrstvu OSI modelu.
[5] Ad-hoc – decentralizovaný typ bezdrátové sítě, která je nezávislá na předem existující infrastruktuře.

	BLE	Wi-Fi	ZigBee	Z-wave	Thread
Rozsah	PAN	LAN	PAN	LAN	PAN
Dosah	75 m	100 m	10 - 100 m	100 m	100 m
Topologie	Star, Mesh	Star, Mesh	Mesh	Mesh	Mesh
Přenosová rychlost	1 Mb/s	10-100 Mb/s+	20–250 Kb/s	100 Kb/s	250 Kb/s
Frekvence	2,4 GHz	2,4; 5 GHz	868, 928 MHz 2,4 GHz	865,2 – 926 MHz	2,4 GHz
Počet zařízení	200	250	1000+	232	250+
Šifrování	128 b AES	WPA, WPA2	128 b AES	128 b AES	128 b AES

Tabulka 2-1: Srovnání specifikací bezdrátových technologií do rozsahu o velikosti LAN

Shrnutí

Existuje mnoho bezdrátových technologií, kterých se dá využít pro vývoj IoT aplikací, každá má své výhody i nevýhody. Výběr takové technologie záleží hlavně na specifikaci a účelu vyvíjené aplikace. Pro aplikaci této práce byla zvolená technologie Wi-Fi, z důvodů enormního využití v praxi a explicitně reálného využití v domácím prostředí, kde je tato technologie většinou přítomna. Také požadavky zadání práce vystihují patřičné potřeby použití této technologie. V následující kapitole je tato technologie v kontextu IoT podrobněji přiblížena.

2.3.1.1 Wi-Fi

Technologie Wi-Fi založená na standardu IEEE 802.11, byla vyvinuta jako bezdrátová náhrada populárního kabelového standardu Ethernet. Jako standard byl od prvého dne vytvořen pro připojení k internetu. Tato technologie umožňuje připojení elektronických zařízení do tzv. bezdrátové lokální sítě (WLAN).

Na vlně úspěchu chytrých mobilů a tabletů, se Wi-Fi stala všudypřítomnou a dostupnou. Wi-Fi přístupové body jsou dnes přítomné ve většině domácností, kancelářích, školách, letištích, kavárnách a maloobchodních prodejnách. Obrovský úspěch Wi-Fi je do značné míry přičítán pozoruhodným programům interoperability provozovaným Wi-Fi Aliancí a je odpovědí na zvyšující se poptávku na trhu z důvodu snadného a nákladově efektivního přístupu do internetu. Wi-Fi je součástí již skoro všech zařízení, po notebooky až televize. Využití stávající obrovské nasazené infrastruktury v domácnostech a firmách, je tak Wi-Fi přirozeným dalším krokem ve věku připojených zařízení do internetu.

Wi-Fi a TCP/IP software je poměrně obsáhlý a komplexní. Pro laptopy a chytré mobily s výkonným mikroprocesorem a obstojnou kapacitou paměti to nepředstavuje žádný problém. Až do nedávné doby, přidávání Wi-Fi připojení do omezených zařízení s malým výpočetním výkonem, jako jsou termostaty a jiné domácí spotřebiče nebylo možné nebo zkrátka bylo neefektivní. V současné době omezené zařízení a moduly vycházejí na trh, již se zabudovaným Wi-Fi a TCP/IP softwarem. Tato nová zařízení eliminují větší přetížení z mikroprocesorů a umožňují tak internetové připojení s nejmenšími mikrokontrolery.

Z důvodu vysoké přenosové rychlosti (100MBps +) a dobrému vnitřnímu pokrytí, mají však Wi-Fi přijímače poměrně vysokou spotřebu energie. U některých IoT zařízení, která se napájí z baterií a nemůže se tak často nabíjet, může být Wi-Fi spotřeba příliš nákladná. Nová zařízení proto aplikují pokročilé spánkové protokoly a rychlé on/off spuštění k dramatické redukci spotřeby energie. Protože většina IoT produktů nespotřebuje toho tolik co technologie Wi-Fi nabízí, chytré návrhy spotřebního managementu mohou efektivně čerpat z produktů, a vydržet tak mnohem déle času bez potřeby opětovného nabíjení.

Závěrem, Wi-Fi je dnes všudypřítomná bezdrátová technologie k připojení do internetu. Její vysoká spotřeba energie a komplexnost byla její největší překážkou pro IoT vývoj. Dnes ale nová zařízení tyto bariéry překonala a umožnila tak Wi-Fi integraci do rozvíjejících IoT aplikací. [7]

2.3.2 Komunikační protokoly

Bezdrátová technologie pro přenos dat je specifikována a nyní je zapotřebí uplatnění technologie pro komunikaci mezi zařízeními. K tomu slouží takzvané přenosové neboli komunikační protokoly. Standardizovaných komunikačních protokolů, kterých by se dalo aplikovat je vícero, avšak v kontextu problematiky vývoje IoT aplikací, jsou následující dva komunikační protokoly v tomto odvětví nejvíce signifikantní a často zmiňované.

CoAP

Komunikační protokol CoAP (Constrained Application Protocol) vychází z protokolu HTTP[6]. Na rozdíl od HTTP je však více triviálnější, lehčí a je vytvořen pro potřeby omezených zařízení. Tento protokol poskytuje model interakce request/response. CoAP podporuje technologii transportní vrstvy UDP, to přináší své klady a zápory. Klienti a servery komunikují skrze nespojovanou komunikaci. Tento protokol je primárně zaměřen na interakci one-to-one pro přenos informací o stavu mezi klientem a serverem. Ovšem u IPv6 podporuje i multi-cast požadavky. CoAP je navržen tak, aby spolupracoval s HTTP a RESTful prostředím pomocí jednoduchých proxy serverů, z důvodu nativní komptability. [18]

MQTT

Komunikační protokol MQTT (MQ Telemetry Transport) je také protokol navržený pro lehkou a nenáročnou M2M[7] komunikaci. Byl vyvinut firmou IBM a dnes je to otevřený standard. Je to protokol pro předávání zpráv mezi více klienty prostřednictvím centrálního bodu – tzv. brokeru. To odděluje výrobce a spotřebitele tím, že umožňuje klientům publikovat a brokeru majícímu moc rozhodnout, kam směrovat nebo kopírovat zprávy. U protokolu MQTT probíhá přenos pomocí TCP a používá návrhový vzor publish/subscribe. Důležitým aspektem je také lehká implementace na straně klienta. To skvěle pasuje pro zařízení s omezenými možnostmi. MQTT také disponuje kvalitou servisu (QoS) o třech úrovních. [19]

Následující tabulka viz (Tabulka 2-2) realizuje porovnání specifikací těchto dvou protokolů.

[6] HTTP (Hypertext Transfer Protocol) – internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML.
[7] M2M (Machine to Machine) – představuje přímou komunikaci mezi zařízeními za použití jakéhokoliv komunikačního kanálu.

	CoAP	MQTT
Komunikační vzor	Request/Response	Publish/Subscribe
Protokol transportní vrstvy	UDP	TCP
Implementace	Komplexní	Jednoduchá
Podpora	Dobrá	Vynikající
Škálovatelnost	Ano (Komplexní)	Ano (Jednoduchá)
QoS úrovně	2	3
Zabezpečení	DTLS	SSL/TLS

Tabulka 2-2: Srovnání specifikací dvojice komunikačních protokolů

Shrnutí

MQTT a CoAP jsou vlajkovými datovými protokoly ve spojitosti s IoT aplikacemi, každý má své výhody a nevýhody. MQTT je vyspělejší protokol proto má lepší zázemí, podporu a rozšířenější funkce. Je také poměrně lehčí implementace MQTT sítě než CoAP sítě. Velký rozdíl také spočívá v protokolu transportní vrstvy, MQTT s TCP dodává spolehlivost za cenu pomalejšího zpracování a většího overheadu[8]. UDP je na druhou stranu jednoduší a rychlejší, ale nezaručuje garanci přijetí zpráv a neskládá segmenty podle pořadí. Použití MQTT je vhodné na Star topologii, kde message broker obsadí roli centrálního zařízení, což přináší nevýhodu v možnosti zahlcení centrálního bodu. Oba dva protokoly mají své klady a zápory, ale nelze říct, který je lepší. Každý protokol se hodí na specifické úlohy. Pro splnění zadaných cílů této práce splňuje MQTT většinu požadavků. V případě této práce je proto tedy zvoleno použití protokolu MQTT. V následující podkapitole je představen jeho přehled a vysvětlení principu užití.

[8] Overhead – jakákoliv kombinace přebytku nebo nepřímého výpočetního času, paměti, šířka pásma, nebo jiného prostředku, který jsou potřebný k provedení určitého úkolu.

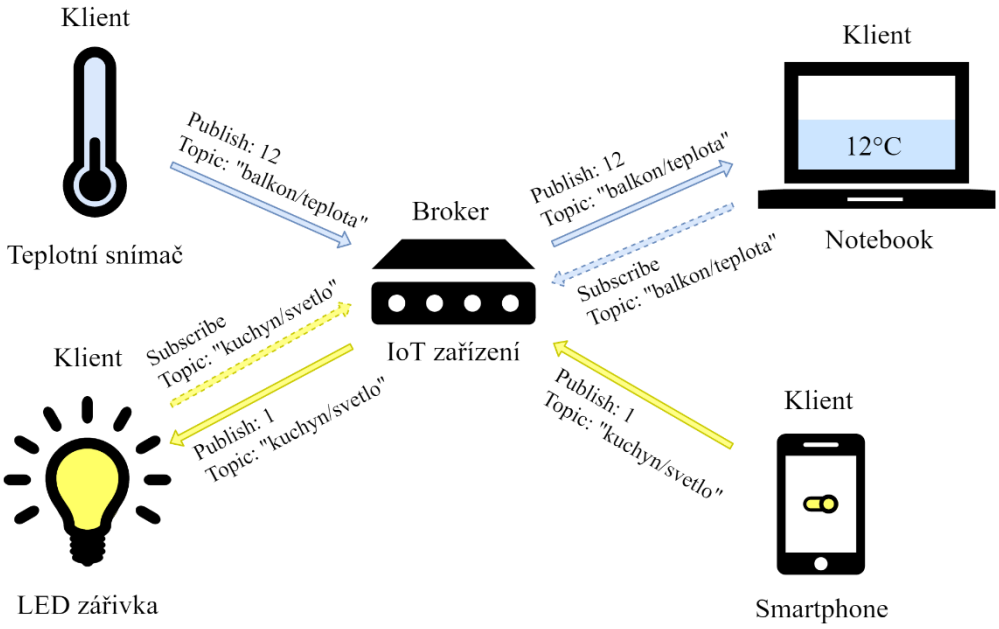
2.3.2.1 MQTT Protokol

MQTT (MQ Telemetry Transport) je klient-server protokol pro přenos zpráv. Jedná se o mimořádně jednoduchý, otevřený, spolehlivý a nenáročný protokol, který je navržen pro jednoduchou implementaci. Je určen pro zařízení, která jsou omezená malou šířkou pásma, vysokou latencí nebo nespolehlivou sítí. Principem návrhu je uskutečnění minimalizace zatížení sítě a požadavků na nároky zařízení a zároveň zajištění spolehlivosti a určitý stupeň zajištění dodání zpráv. Díky těmto vlastnostem je ideální pro použití v mnoha situacích, včetně omezených prostředích, jako pro komunikaci M2M a IoT světě připojených zařízení. To samé platí při vývoji mobilní aplikace, kde šířka slova a kapacita baterie je důležitějším požadavkem.

Protokol MQTT byl vynalezen Andy Stanford-Clarkem (IBM) a Arlenem Nipperem (Cirrus Link) v roce 1999, kde pro jejich případ užití bylo nutno vytvořit protokol pro minimální ztrátu energie baterie a minimální šířku pásma spojující ropovody přes satelitní připojení. V roce 2010 firma IBM vydala první volnou verzi protokolu 3.1. V roce 2014 MQTT byl oficiálně schválen jako standard OASIS. Dnešní aktuální verze protokolu je 3.1.1 ve které přibylo pár minoritních změn v porovnání s předchozí verzí 3.1.. MQTT má také zarezervované TCP/IP porty organizací IANA. Jsou to porty 1883 a 8883 (Secure MQTT).

MQTT využívá takzvaný publish/subscribe vzor viz (Obrázek 2-4). Tento vzor je alternativou tradičního modelu klient-server, kde klient komunikuje přímo s koncovým bodem. Publish/subscribe odděluje klienta, který posílá konkrétní zprávu (nazývaný publisher) dalšímu klientovi (nebo klientům), kteří zprávu přijímají (nazývaný subscriber). Dále je tu třetí komponent nazývaný broker, který filtruje všechny příchozí zprávy a podle toho je distribuuje dál. MQTT využívá filtrování zpráv založené na principu tématu (nazývané topic). Takže každá zpráva obsahuje topic, který poté broker používá pro zjištění, jestli subscriber obdrží zprávu nebo ne. Z toho vyplývá, že publisher a subscriber neví o vzájemné existenci. Podstatnou součástí tohoto protokolu jsou také 3 úrovně QoS (Quaility of Service) zaručující kvalitu přenosu zpráv. Důležité je zmínit, že tyto úrovně se vyskytují na aplikační úrovni, na transportní úrovni je kvalita servisu zaručena protokolem TCP. Nultá úroveň nazvaná „Nanejvýš jednou“ je úroveň, kde poslaná zpráva není potvrzená příjemcem nebo uložená ani přeposlaná odesílatelem. První úroveň „Alespoň jednou“ je úroveň kde je zaručeno, že zpráva bude doručena alespoň jednou příjemci, ale zpráva může být duplikována.

Druhá úroveň „Přesně jednou“ je finální úroveň. Je to nejbezpečnější a zároveň nejnáročnější úroveň. Zaručuje, že každá zpráva je přijata příjemcem pouze jednou.



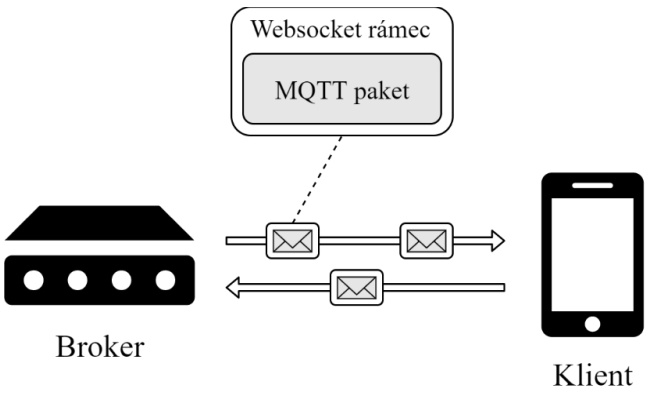
Obrázek 2-4: Znázornění principu vzoru publish/subscribe na příkladu

MQTT spojení je vždy mezi jedním klientem a brokerem, žádný klient není připojen k jinému klientovi přímo. Spojení je zahájeno prostřednictvím klienta odesláním CONNECT zprávy brokeru. Broker odpoví odesláním CONNACK zprávy, který obsahuje statusový kód. Jakmile je navázáno spojení, broker udržuje spojení otevřené tak dlouho, dokud klient neodešle příkaz pro odpojení nebo sám neztratí spojení. Mezi další specifické vlastnosti patří například LWT (Last Will and Testament) zprávy. Tyto zprávy jsou uloženy v brokeru v případě neočekávaného výpadku klienta a slouží k jeho navrácení do běžného stavu. Další vlastností je také například použití tzv. Wildcard témat, která slouží pro publikování a odebírání z více témat najednou a zabraňují tak přepisování stejného kódu. [19]

2.3.2.2 MQTT s využitím technologie WebSockets

WebSockets je webová technologie, která umožňuje obousměrnou komunikaci mezi klientským webovým prohlížečem a webovým serverem. Vytváří trvalé TCP spojení mezi klientem a serverem, při kterém je možné přenášet data v reálném čase. Websockets mají velmi malý overhead, pokud jde o šířku slova a latenci v porovnání s klasickým HTTP. Tato filozofie dobře zapadá do MQTT systému s potenciálním využitím webových technologií. Tento protokol byl standardizován v roce 2011 a dnes už drtivá většina prohlížečů má vestavěnou podporu WebSockets. Stejně jako u MQTT, WebSockets jsou založeny na TCP. S představením technologie HTML5, a dostupností WebSockets vznikají nové možnosti pro vývoj hybridních webových aplikací.

Princip této technologie s použitím MQTT je vcelku jednoduchý, MQTT zpráva je zapouzdřena jedním nebo více WebSockets rámci o velikosti 2 B, při přenosu přes síť viz převzatý a upravený obrázek (Obrázek 2-5). WebSockets jsou vhodné pro součinnost s MQTT, protože komunikace je obousměrná a bezztrátová. Aby bylo možné komunikovat s MQTT brokerem přes WebSockets, broker musí mít nativní podporu protokolu WebSockets. Pro klientskou část je tu podpora ze strany Eclipse IoT dostupnou knihovnou Paho Javascript client viz (3.1.2). Ohledně bezpečnosti je možné použití zabezpečené komunikace WebSockets s šifrováním celého spojení za využití TLS. [20]



Obrázek 2-5: Znázornění principu využití technologie Websockets s MQTT [20]

2.3.2.3 Zabezpečení MQTT

Cílem MQTT je poskytnout doopravdy odlehčený a lehce použitelný komunikační protokol pro IoT. To je ovšem také důvodem proč protokol postrádá na hojnosti bezpečnostních mechanismů. Zabezpečení MQTT je rozděleno do více úrovní. Tyto úrovně jsou rozdělené podle TCP/IP modelu a každá úroveň zabezpečuje vlastní vrstvu před různými typy útoků. Tyto úrovně se nazývají: síťová, transportní a aplikační. [6]

Síťová úroveň

Na síťové úrovni lze aplikovat ochranné prvky použitím fyzicky zabezpečené sítě nebo VPN a jako základ pro komunikaci mezi klienty a brokerem je to jeden ze způsobů, jak zajistit bezpečné a spolehlivé připojení.

Transportní úroveň

Na transportní úrovni je zajištěno, že komunikace je šifrovaná a proklamovaná identita je ověřena. Cílem je zajistit důvěrnost a ve většině případů je SSL/TLS používán pro šifrování komunikace. To poskytuje bezpečný a osvědčený způsob, jak se ujistit, že nikdo nepřečte komunikaci a také ověří obě strany při použití ověřování uživatele pomocí certifikace. SSL/TLS poskytují zabezpečený komunikační kanál mezi klientem a serverem. Ve svém jádru TLS a jeho předchůdce SSL jsou kryptografické protokoly, které používají „handshake“[9] protokol ke sjednání zabezpečeného spojení. Po handshake začíná šifrovaná komunikace mezi klientem a serverem a je tak ochráněná před útočníkem. Servery poskytují X509 certifikát, typicky vydaný důvěryhodnou autoritou, kterou klienti používají k ověření totožnosti serveru. Šifrování je nutností v případě použití TCP/IP. TCP paket projde celou škálou komponentů předtím, než dojde do cíle. Útočník by tak mohl paket odposlechnout nebo upravit. V případě použití autentizace, heslo by si mohl na síti kdokoliv odposlechnout. SSL/TLS je skvělá metoda, jak zabezpečit datovou komunikaci, ale z důvodu výskytu handshake protokolů, zvýšené velikosti paketů a šifrování komunikace můžou nastat zátěžové a časové problémy u omezených zařízení. Na tento problém je zpracovaný elaborát [21], ze kterého vyplývá následující tvrzení – při počátečním handshake protokolu je výrazná zátěž převážně na procesorové prostředky, jakmile se však klient připojí a relace dál pokračuje, overhead

[9] Handshake – inicializační protokol, který slouží pro zahájení spojení mezi klientem a serverem.

komunikace je nadále zanedbatelný. Výhodou MQTT klienta je tak potřeba jediného navázání spojení za relaci.

Aplikační úroveň

Poslední je aplikační úroveň. Protokol MQTT poskytuje identifikátor klienta a dvojici jméno/heslo, které lze použít také k ověřování zařízení na aplikační úrovni. Tyto vlastnosti jsou poskytovány samotným protokolem. Když přijde na povolení nebo na to, co každé zařízení má povoleno provést, s tím už zachází vlastní implementace a konfigurace brokeru. Další možností je použití šifrování samotného obsahu zprávy na aplikační úrovni.

3 Použité technologie

3.0 Rozšířená inteligence

Další etapa v procesu implementace zahrnuje pohled na extrahování dat určená pro analýzu a také schopnost analyzovat nestrukturovaná data. Tato analýza může být prováděna vyššími kognitivními technologiemi nebo různými doprovodnými modely.

3.0.1 MQTT broker

MQTT broker slouží jako centrální bod neboli server MQTT systému. Je to softwarová aplikace. Mezi jeho hlavní funkce patří distribuování dat na základě přihlášení k odběru jednotlivých témat, řízení administrace klientů a kryptografické mechanismy. Mezi jeho další funkce patří tzv. klastrování brokeru, které umožňuje komunikaci mezi ostatními brokery a vytváří tak naddimenzovanou síť propojených brokerů.

Výběr brokeru je omezen jen na open-source projekty. Mezi hlavní požadavky od brokeru je zvolena podpora QoS o všech úrovních, podpora Websockets, bezpečnostních mechanismů SSL/TLS a autentizace uživatelů.

V následující tabulce viz (Tabulka 3-1) je představeno porovnání mezi následujícím výběrem těchto brokerů.

	Mosquitto	MQTT.js	Mosca	HMBQTT	RabbitMQ
QoS 0,1,2	Ano	Ano	Ne	Ano	Ne
Autentizace	Ano	Ne	Ano	Ano	Ano
WebSockets	Ano	Ano	Ano	Ano	N/A
SSL/TLS	Ano	Ano	N/A	Ano	Ano
Dyn. témata	Ano	Ano	N/A	Ano	Ano
Bridge	Ne	Ne	N/A	Ne	Ne
Cluster	Ano	Ne	Ne	Ne	N/A
Plugin systém	Ano	Ne	Ne	Ano	N/A

Tabulka 3-1: Porovnání výběru open source MQTT brokerů [22]

Shrnutí

Pro účely této práce je vybrán broker Mosquitto. Splňuje všechny zadané požadavky viz (tab. 2-3). Je to veřejný open source projekt, který spadá pod organizaci Eclipse IoT a je velice populární mezi uživateli, z toho vyplývá vynikající podpora a mnoho uživatelských rad. Mosquitto má také vlastní implementaci testovacího brokeru k libovolnému použití na doméně „test.mosquitto.org“. Motivací této práce je však implementace vlastního identického brokeru. Jako role zhoštění Mosquitto brokeru je vybrán minipočítač Raspberry Pi, který se skvěle hodí pro tento typ úkolu.

3.0.1.1 Mosquitto

Eclipse Mosquitto je jeden z několika open source message brokerů, který implementuje MQTT protokol verze 3.1 a 3.1.1. Mosquitto je projektem organizace Eclipse IoT, která má na starosti vícero open source implementací pro IoT projekty. První live verze byla vydána dne 14. srpna 2012. V roce 2015 byla vydána verze 1.4 kde byla přidána zmiňovaná nativní podpora WebSockets. Aktuální verzí je nyní 1.4.11, která byla vydána 21. února 2017. [23]

3.0.1.2 Raspberry Pi

MQTT broker Mosquitto je hostován na jednodeskovém minipočítači Raspberry Pi 2 Model B (Obrázek 3-1). Toto zařízení se hodí pro tento typ úlohy z důvodu nízké náročnosti na hardwarové podmínky a zároveň pro možnost testování zvolených technologií. Raspberry Pi druhé generace model B vyšel v únoru 2015. Mezi hlavní specifikace patří 900MHz čtyřjádrový ARM Cortex-A7 procesor, který je až 6x rychlejší než u předchozí verze a operační paměť s kapacitou 1 GB, která je sdílená s grafickým procesorem Broadcom VideoCore IV 400 Mhz/300 Mhz s OpenGL ES 2.0. Deska je osazená veškerými všestrannými vstupy a výstupy (např. HDMI, 3,5 mm jack, USB, Ethernet a rozhraní GPIO). Dnes se aktuální cena za tento minipočítač pohybuje kolem jednoho tisíce korun českých. [24]

Mezi zvolené příslušenství Raspberry Pi patří micro SD karta o kapacitě 8 GB jako interní paměť. Tato kapacita by měla stačit, ale je to také minimální doporučená velikost. Interní paměť slouží především k uložení operačního systému. Další součástí je USB Wireless 802.11N Nano Dongle, který slouží jako Wi-Fi adaptér, třetí generace Raspberry Pi má Wi-Fi podporu integrovanou přímo na desce. Jako poslední je nedílná součást pro provoz samotného Raspberry Pi a tou je microUSB napájecí zdroj. Pokud nechce uživatel pracovat jen s terminálem a SSH spojením, může použít dodatečně USB nebo Bluetooth myš s klávesnicí a přes periferii HDMI použít monitor.

Obrázek 3-1: Raspberry Pi 2 model B + Wi-Fi Dongle + Micro SD karta

3.0.2 Výkonný prvek

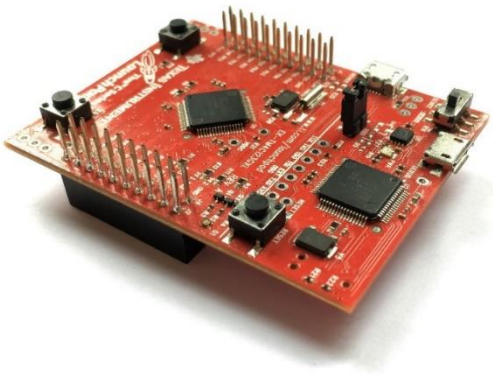
Tato kapitola by měla patřit k sekci systému senzorů a aktuátorů, vzhledem k přehlednosti práce je probrána v této části. Výkonný prvek je množina spolu komunikujících prvků plnící roli lokálního klienta v MQTT systému. Slouží pro účel navázání obousměrné komunikace s brokerem. Mezi testovací funkce tohoto systému patří měření, zpracování, publikování senzorických dat a přijímání instrukčních dat. Pro účely této práce jsou zadané hardwarové prvky od firmy Texas Instruments. Texas Instruments je americká firma s dlouholetou historií zabývající se výrobou technologických zařízení pro různá odvětví. Jakožto technologický gigant je také angažován ve vývoji a výrobě různých hardwarových zařízení pro IoT řešení. Tyto zařízení ovšem nejsou komerčního typu, jsou spíše zaměřené na specifickou menšinu vývojářů. Nejsou tak lehce sehnatelná ani cenově srovnatelná jako klasičtější představitelé, mezi které patří například zařízení Arduino.

Pro zpracování senzorických dat je vybrán LaunchPad[10] Tiva C. Pro připojení do bezdrátového světa je vybrán Wi-Fi Boosterpack[11] CC3100. Tyto zařízení jsou produktem společnosti Texas Instruments.

3.0.2.1 Vývojářská platforma

Pro zpracování dat je vybrán nízkorozpočtový LaunchPad Tiva C Series EK-TM4C123GXL viz (Obrázek 3-2). Je to vývojová platforma založená na ARM Cortex - M4F mikroprocesorech od firmy Texas Instruments. Konstrukce TM4C123G LaunchPadu zdůrazňuje TM4C123GH6PM mikroprocesor s rozhraním USB 2.0 a hibernačním modulem. EK – TM4C123GXL také disponuje programovatelnými tlačítky a RGB LED pro jakékoli použití. Tato platforma je výborným řešením pro rozšíření funkcionality za pomocí dalších periferií nazývanými BoosterPacky. Cena tohoto Launchpadu je dnes okolo tři sta korun českých. [25]

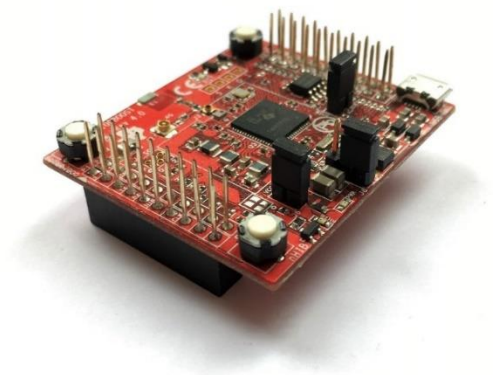
[10] Launchpad – vývojářské kity mikrokontrolerů od firmy Texas Instruments.
[11] Boosterpack – plug-in modul se specifickou funkcionalitou, který je kompatibilní s Launchapady.



Obrázek 3-2: Tiva C Series EK-TM4C123GXL Launchpad

3.0.2.2 Wi-Fi modul

Pro připojení výkonného prvku do bezdrátové sítě je vybrán Boosterpack CC3100 SimpleLink Wi-Fi viz (Obrázek 3-3). Obsahuje TCP/IP stack. Tento malý Boosterpack poskytuje flexibilitu přidání Wi-Fi připojení k jakémukoli Launchpadu. Je to první verze řešení Wi-Fi na chipu z rodiny SimpleLink, proto také podporuje pouze IPv4 adresaci. Tento chip obsahuje vše, co je potřeba pro snadné vytváření IoT řešení. Aktuální cena tohoto Boosterpacku je okolo pět seti korun českých. [26]



Obrázek 3-3: SimpleLink Wi-Fi CC3100 BoosterPack

3.0.2.3 Senzory a aktuátory

Tato kapitola pojednává o využitém systému senzorických a akčních prvků. Tento vybraný systém slouží pro demonstraci funkční komunikace. Jako testovací senzory a aktivátory jsou zvoleny následující: LED, teplotní čidlo, PIR senzor, Servomotor.

LED

Pro demonstraci funkčního přijímání příkazů z mobilní aplikace, byla zvolena polovodičová elektronická součástka, jejíž vlastností je vyzařování světla – světelná dioda. Přidáním 5V relé modulu je možné také zkonstruovat elektrický obvod pro klasickou žárovku.

Teplotní čidlo

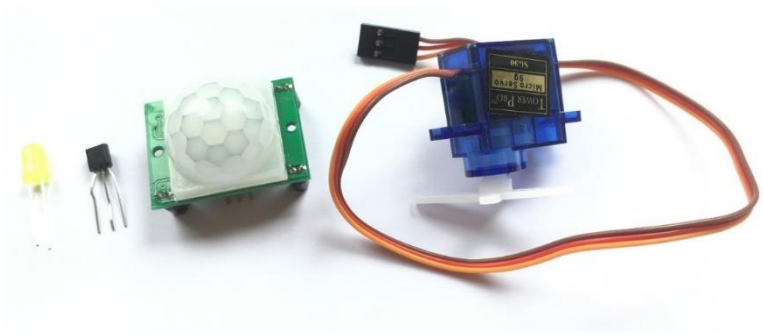
Jako testovací senzor je vybrán teplotní snímač LM35DZ, pro jednoduché měření okolní teploty. Řada LM35 jsou přesné teplotní senzory, jehož výstupní napětí je přímo úměrná teplotě ve stupních Celsia. LM35 tak má výhodu oproti lineárním teplotním čidlům, která jsou kalibrována v Kelvinech, protože uživatel není povinen odečíst velkou konstantu napětí, ze kterého by získal údaje ve stupních Celsia. Cena se pohybuje v jednotkách českých korun za jeden kus. [27]

PIR senzor

Druhým testovacím senzorem je zvoleno pohybové čidlo HC-SR501. Tento infračervený senzor pohybu dokáže registrovat pohyb do vzdálenosti 7 m v úhlu 110 stupňů. Umožňuje nastavení citlivosti detekce a dobu po kterou je výstup aktivní. Cena tohoto senzoru se pohybuje v desítkách českých korun. [28]

Servomotor

Jako testovací aktuátor je vybrán servomotor SG90 9g Micro Servo, pro příkladné ovládání okenních žaluzií. Servomotor je motor pro pohony, u kterých lze na rozdíl od běžného motoru nastavit přesnou polohu natočení osy. Tento malý servomotor je schopen maximální rotace o 180 stupňů. Cena tohoto aktuátoru se pohybuje kolem sta korun. [29]



Obrázek 3-4: Zleva žlutá LED, Teplotní čidlo, PIR senzor, Servomotor

3.1 Rozšířené chování

Rozšířené chování může být vyjádřeno jako vývoj rozhraní pro komunikaci člověka se strojem pomocí UI a UX technologií. Tyto rozhraní slouží ve skutečnosti pro vykonání požadované akce systému. Způsobů, jakým lze zprostředkovat rozhraní uživateli existuje mnoho, avšak zadané požadavky této práce jsou orientované na problematiku vývoje mobilních aplikací.

3.1.1 Uživatelská mobilní aplikace a její vývoj

Jako výstupní proměnná tohoto systému je vyvinuta mobilní aplikace, která disponuje prvky základní názorné funkcionality jako ovládání výkonného prvku a vizualizace požadovaných senzorických dat. Tato aplikace zastupuje funkci externího MQTT klienta pro vzdálenou komunikaci.

Mobilní aplikace je softwarová aplikace poskytující určité služby a je určená pro provoz na mobilních a bezdrátových zařízení, jako jsou chytré telefony a tabletové počítače. Mobilní aplikace jsou navrženy s ohledem na požadavky a omezení zařízení. V dnešní době dominují světu mobilních zařízení dva operační systémy, čímž je Android a iOS. Vývoj mobilních aplikací je kategorizováno do třech základních skupin: nativní, webové a hybridní aplikace, viz převzatý a upravený obrázek (Obrázek 3-5).

Nativní aplikace

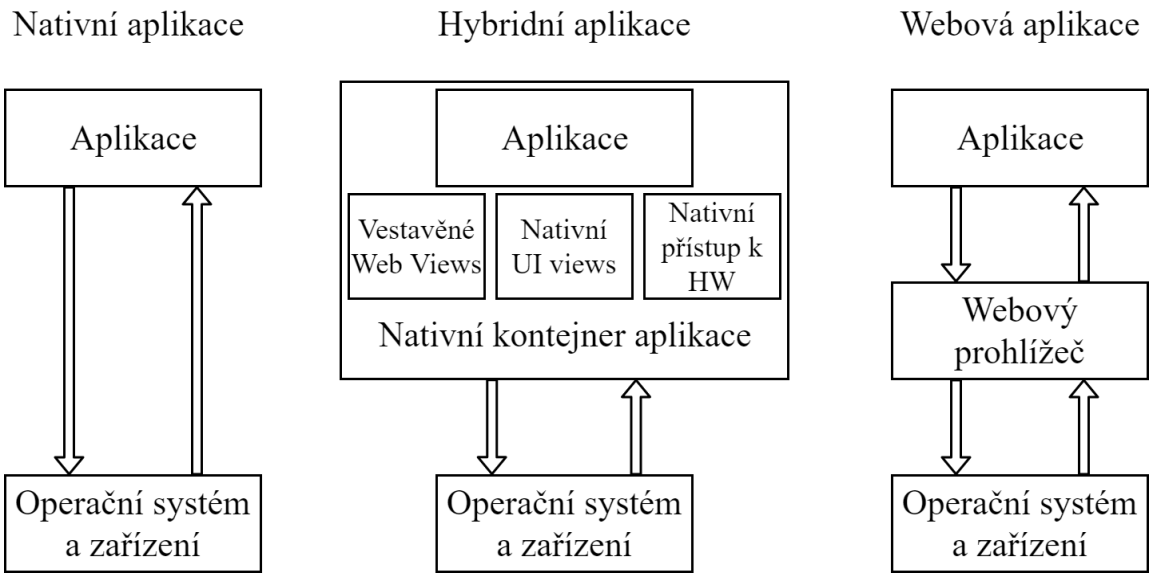
Nativní aplikace jsou vytvářené pro konkrétní platformu a hardware specifického zařízení. Takovéto aplikace jsou napsané ve vyšších programovacích jazycích, pro android je to Java a pro iOS je to Objective-C nebo Swift. Tím pádem skvěle využívají svůj hardwarový potenciál. Jsou rychlé a svižné a podporují nativní UI prvky. Naopak nevýhodou je absence multiplatformního využití. Z toho se odvíjí problémy jako těžší a časově náročnější vývoj.

Webová aplikace

Webová aplikace je univerzální forma, která funguje napříč všemi platformami. Webová aplikace typ klient-server, kde klient je vlastně jen webová stránka přizpůsobená mobilním zařízením. Největší výhodou tohoto řešení je v jednoduchosti a rychlosti vývoje. Tyto aplikace nepotřebují distribuci přes obchody třetích stran. Zároveň se aplikace se nemusí vůbec instalovat, stačí jen prezence webového prohlížeče. Nevýhodou je však omezená funkcionalita, nativní prvky UI a nutnost být připojený k internetu.

Hybridní aplikace

Hybridní aplikace je kompromis mezi předchozími zmíněnými možnostmi. Zjednodušeně řečeno je to aplikace postavená webovými technologiemi, která je poté zabalena do tzv. nativního kontejneru. Aplikace je instalovaná na zařízení, dokáže využít hardwarových schopností telefonu stejně jako u nativní aplikace. Dobře vyvinutá hybridní aplikace je pro obyčejného uživatele nerozeznatelná od nativní aplikace. Použití webových technologií zaručuje multiplatformní podporu.



Obrázek 3-5: Stručný koncept principu rozdělení mobilních aplikací [30]

Shrnutí

Pro účely práce je vybrán vývoj hybridní mobilní aplikace. Hlavním důvodem je existence MQTT klientské knihovny Paho Javascript Client (3.1.2), která využívá technologii Websockets, která se dá prakticky využít právě u vývoje hybridní a webové aplikace. Volba hybridní aplikace před aplikací webovou je důvodem zájmu o poměrně nový druh trendu vývoje aplikací. Také vývoj hybridní aplikace spočívá v jednoduchosti a menší časové náročnosti než u vývoje nativní aplikace. V následující kapitole je přiblížen vývoj hybridních aplikací.

3.1.1.1 Vývoj hybridní aplikace

Vývoj hybridních aplikací je v aktuální době velice atraktivním tématem. Hybridní aplikace je v podstatě webová aplikace postavená na HTML, CSS a Javascriptu, zabalená do nativního

kontejneru. To znamená, že použití takových aplikací je multiplatformní a je využito potenciálu webového responsivního designu. Klíčovým prvkem u vývoje hybridních aplikací je využití tzv. WebView (Webové rozhraní), což je schopnost aplikace zobrazovat webový obsah přímo uvnitř samotné aplikace. Tímto se také zpřístupní možnost využití nativních schopností jako je například kamera nebo akcelerometr. Výhodou takových aplikací je jednoduší a časově méně náročnější vývoj než u nativních aplikací. Nejrozšířenějším řešením pro vývoj hybridních aplikací je platforma Apache Cordova. [31]

Apache Cordova

Pro tvorbu hybridní aplikace je použita technologie Apache Cordova. Jedná se o framework v nativním kódu platformy, který obsahuje WebView zvětšené na celou obrazovku, kde je spuštěna webová aplikace. Apache Cordova je sada API s kódem, který umožňuje vývojářům mobilních aplikací přistupovat k nativním prvkům zařízení, jako jsou třeba kamera nebo různá čidla, přímo z JavaScriptu. Pomocí Apache Cordova může být vytvořena plnohodnotná mobilní aplikace jen s použitím JS, HTML a CSS. [32]

Ionic

Jako konkrétní framework je zvolen Ionic 1. Je vybrán z důvodu rozsahu působnosti v prostředí vývoje hybridních aplikací a jeho proslulostí. Ionic je velice populární front-end JavaScript Framework pro vyvíjení multiplatformních mobilních aplikací za využití platformy Apache Cordovy. Je to kompletně open-source SDK[12], postaven na AngularJS[13] a Apache Cordova. Ionic poskytuje nástroje a služby pro vývoj hybridních mobilních aplikací pomocí webových technologií jako CSS, HTML5 a Javascript. Aplikace mohou být postaveny s těmito webovými technologiemi a poté distribuovány prostřednictvím nativních obchodů jako Appstore a Google play. [33]

3.1.2 MQTT Klientské knihovny

Klienti v MQTT systému, kteří chtějí inicializovat komunikaci s MQTT brokerem se musí realizovat za pomoci takzvaných klientských knihoven. Drtivá většina MQTT klientských knihoven je vyvíjena zmíněnou skupinou Eclipse IoT pod projektem nazvaným The Eclipse Paho. Tento projekt poskytuje open-source klientské implementace zaměřených na nových,

[12] SDK (System Development Kit) – sada vývojových nástrojů umožňující vytváření aplikací.
[13] AngularJS – javascriptový open-source webový framework založený firmou Google.

stávajících a nově vznikajících aplikací pro Internet věcí. Takovýchto otevřených knihoven existuje několik, většinově jsou rozdělené podle jazykových preferencí a možnostech použití. Pro realizaci MQTT připojení výkonného prvku s brokerem je použita knihovna PubSubClient. Pro realizaci MQTT připojení pomocí protokolu WebSockets mobilní aplikace s brokerem je vybrána javascriptová knihovna Paho Javascript Client. Dále jsou tyto knihovny přiblíženy.

PubSubClient

PubSubClient knihovna byla vytvořena pro platformu Arduino v roce 2009. V tu dobu byl pro Arduino vydán první Ethernet Shield, což bylo základem a podmínkou pro přirozený vývoj a další cestu ke spuštění MQTT. Následně byla knihovna modifikována a spuštěna pro zařízení Texas Instruments a nástroj Energia[14]. V omezeném prostředí malých zařízení je důležité, aby knihovna zůstala co nejmenší. Toho by mohlo být dosaženo pouze implementováním funkcí, které mají smysl. Zejména podporuje Clean Sessions a nepodporuje úroveň QoS 2 z důvodu omezené paměti a nepřítomnosti persistentního mechanismu. Také nepodporuje zmíněný mechanismus SSL/TLS. [34]

Paho Javascript Client

Paho Javascript Client je MQTT knihovna založená pro potřeby webových technologií napsaná v jazyku JavaScript, která využívá výhody WebSockets připojení k MQTT brokeru. Knihovna byla původně napsána Andrewem Banksem v IBM a byla darována Eclipsu od IBM v roce 2013. Knihovna je považována za velmi stabilní a je používána v mnoha MQTT webových aplikacích. Použitím WebSockets Javascriptová knihovna umožňuje využívat vývojářům MQTT bez použití portu 1883. Podporuje bezpečnostní prvky autentizace a SSL/TLS. [35]

[14] Energia – vývojové prostředí pro zařízení od firmy Texas Instruments.

4 Návrh řešení

V této kapitole je využito teoretických východisek, analýz a použitých technologií z předchozí části práce. Jako první je vytvořen pohled na síťovou komunikaci pomocí TCP/IP modelu. Následně je navržena architektura počítačové sítě. Dále jsou nadefinované požadavky pro demonstrování funkcionality systému. Závěrem je navržené blokové schéma návrhu řešení komunikace celého ekosystému zařízení.

4.0 TCP/IP model

Pro celkovou představu je vytvořen pohled na realizaci síťové komunikace pomocí TCP/IP modelu[15] viz (Tabulka 4-1).

Aplikační vrstva	MQTT (v3.1.1), Websockets
Transportní vrstva	TCP, SSL/TLS (TLS v1.2)
Internetová vrstva	IPv4
Síťová vrstva	Wi-Fi (802.11 bgn mixed, 2,4 GHz)

Tabulka 4-1: Vizualizace návrhu pomocí TCP/IP modelu

Síťová vrstva

Síťová vrstva zajišťuje konverzi bitů na rádiové signály (a naopak), stará se o rámování dat pro spolehlivou bezdrátovou komunikaci a řídí přístup do stanice. Zvolená technologie je Wi-Fi.

Internetová vrstva

Internetová vrstva adresuje a směruje data prostřednictvím sítě. Internetový protokol poskytuje IP adresu zařízení a řízení IP paketů z jednoho zařízení na další. Použitý protokol je IPv4.

[15] TCP/IP model – hierarchický model síťové komunikace, který je rozdělený do čtyř vrstev, kde každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší.

Transportní vrstva

Transportní vrstva vytváří komunikační relace mezi aplikacemi běžícími na dvou koncích sítě. To umožňuje více aplikacím běžet na jednom zařízení, z něhož každá používá vlastní komunikační kanál. MQTT protokol je podporován přes TCP komunikaci. TCP je převládající přenosový protokol na internetu. TCP je spolehlivý protokol díky svému potvrzovacímu mechanismu komunikace, zároveň pomocí skládání paketů podle pořadí. Také obsahuje zabezpečující prvek v podobě zmíněného SSL/TLS.

Aplikační vrstva

Aplikační vrstva je zodpovědná za formátování dat a řídí tok dat v optimálním schématu pro specifické aplikace. Zvolen je MQTT protokol ve spolupráci s technologií Websockets.

4.1 Architektura počítačové sítě

Rozsah sítě

Z myšlenky realizování domácí automatizace je rozsah počítačové sítě primárně určen pro bytový prostor o rozloze pokrytí jednoho pokoje. Tato specifikace tak spadá do kategorie WLAN. V tomto pokoji se společně nachází výkonný prvek, MQTT broker i Wi-Fi směrovač. Proto otázka řešení dosahu Wi-Fi signálu odpadá.

Topologie sítě

Z důvodu užití principu MQTT systému, který pojednává o komunikaci mezi MQTT klienty a MQTT brokerem, což je alternativou modelu klient-server je proto nejjednodušším přístupem zvolení Star topologie. Nevýhodou této topologie v případě zvoleného systému je centralizace do jediného bodu v síti, z čehož vyplývá hlavní nedostatek. Pokud selže funkčnost nebo bude centrální prvek zahlcen požadavky, bude tak ochromen celý systém.

4.2 Požadované vlastnosti

V této kapitole jsou nadefinované funkční požadavky od celého ekosystému zařízení, podle kterých je strukturována kapitola o realizaci řešení viz kapitola (5).

MQTT broker

- Konfigurace dostupného připojení pro klienty.
- Zabezpečení celého systému pomocí autentizace, autorizace a realizace šifrované komunikace pomocí SSL/TLS.
- Přítomnost funkce logování systémových údajů pro potřeby analýz.
- Dostupnost pro potřebu externí komunikace s mobilní aplikací.
- Zaručení dostupnosti služeb pro klienty.

Výkonný prvek

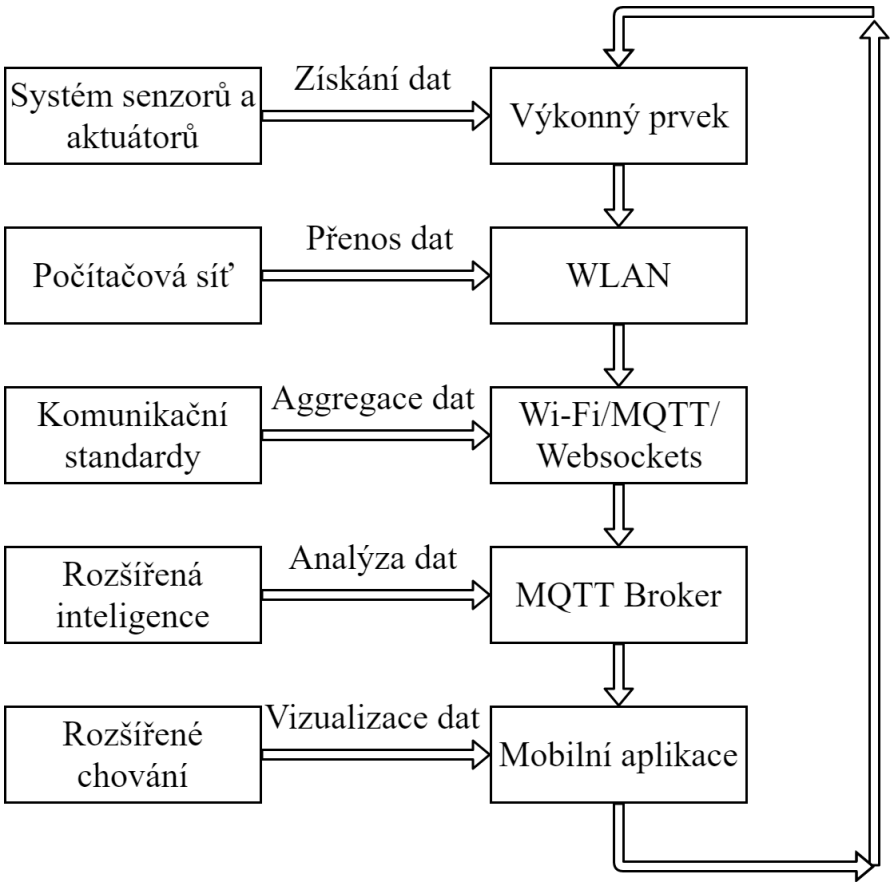
- Sestavení výkonného prvku spolu s nadefinovanou množinou senzorů a aktuátorů.
- Navázání obousměrné komunikace s MQTT brokerem pomocí protokolu MQTT.
- Generování reálných vytvořených senzorických dat a vytvoření podmínek pro aktuátory v případě přijetí zpráv ve formě příkazů z mobilní aplikace.
- Realizace optimalizovaného publikování dat a přijímání příkazů do/z mobilní aplikace.

Mobilní aplikace

- Navázání zabezpečené obousměrné komunikace s MQTT brokerem pomocí webové technologie WebSockets a SSL/TLS.
- Demonstrování základní funkcionality jako je ovládání výkonného prvku a vizualizace vygenerovaných senzorických dat.
- Reálné publikování mobilní aplikace.

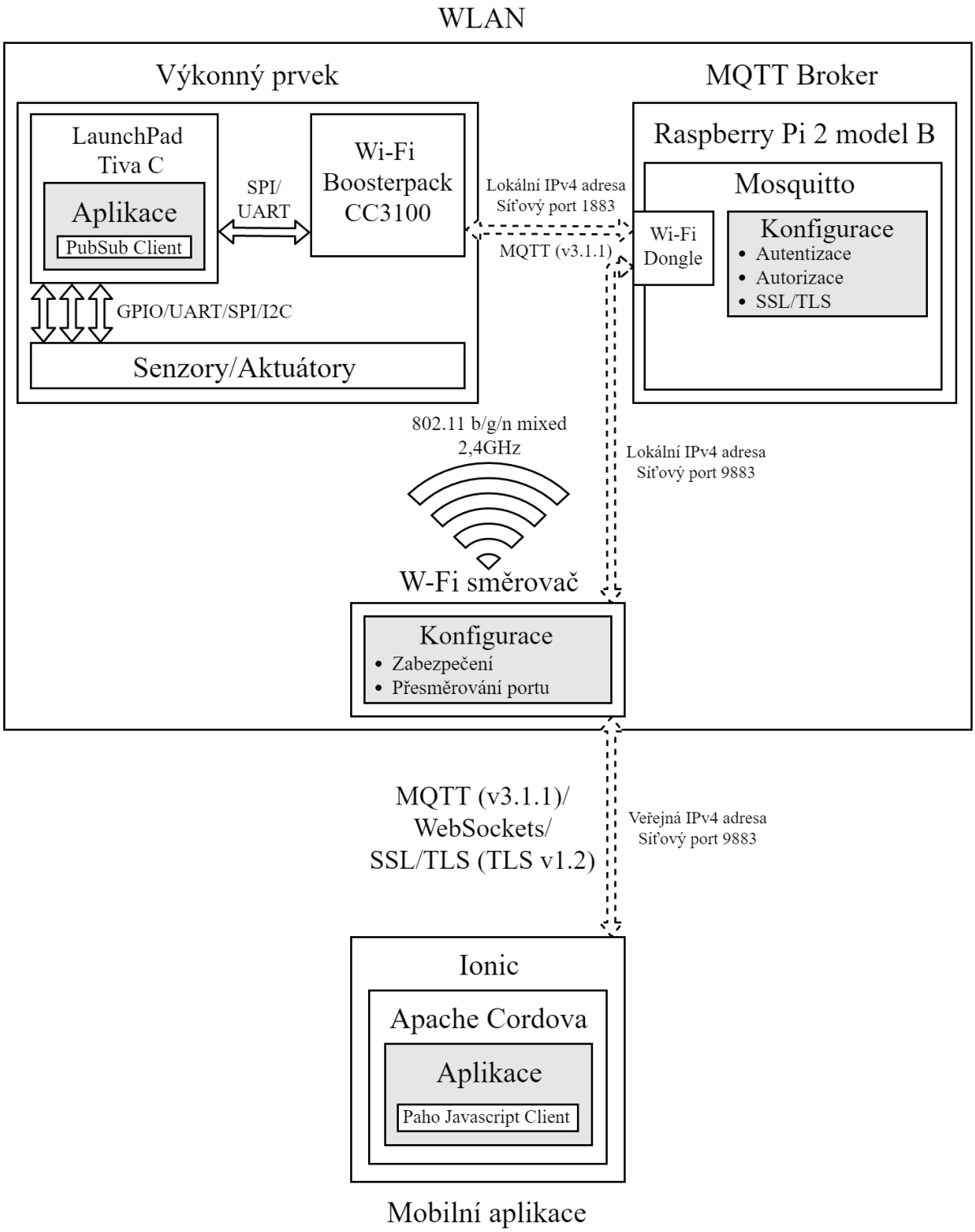
4.3 Blokové schémata

První blokové schéma viz (Obrázek 4-1) je v samé podstatě naplnění myšlenky a principu počáteční definice smyčky hodnoty informace viz (Obrázek 1-1).



Obrázek 4-1: Blokové schéma naplnění principu smyčky hodnoty informace

Následující blokové schéma návrhu řešení vychází z teoretických východisek a z použitých technologií. Interpretuje požadovaný princip realizace sestavení úplného komunikačního systému zařízení viz (Obrázek 4-2). Šedé diagramy v obrázku indikují, čím se následující kapitola o realizaci řešení zaobírá.



Obrázek 4-2: Návrh požadovaného řešení celého komunikačního systému

5 Realizace řešení

Realizace řešení vychází z návrhu řešení za použití definovaných technologií z teoretické části. Jako první je přiblížena implementace a konfigurace Mosquitto na platformě Raspberry Pi v roli MQTT brokeru. Následuje sestavení výkonného prvku v roli lokálního MQTT klienta a poté je popsán vývoj mobilní hybridní aplikace v roli druhého externího MQTT klienta.

5.0 MQTT broker

Tato kapitola pojednává o přípravě Raspberry Pi k instalaci Mosquitto brokeru a jeho následné požadované konfiguraci a realizaci zabezpečujících prvků.

5.0.1 Instalace operačního sytému

Jako operační systém je zvolen Raspbian Jessie s novým rozhraním Pixel verze – září 2016. Je to operační systém na bázi Debianu s oficiální podporou pro Raspberry Pi. Právě Raspbian přichází s předinstalovanými moduly a prvky.

První krok samotné instalace počíná ve stažení systémového image souboru, který obsahuje aktuální verzi operačního systému Raspbian Jessie z oficiálních webových stránek Raspberry Pi. Následně na zformátovanou micro SD kartu se pomocí nástroje Win32 Disk Imager zapíše tento image souboru. Po dokončení kopírování je micro SD karta vložena do slotu v Raspberry Pi. Nyní se může Raspberry Pi zapojit do elektrické sítě a vyčkat než se operační systém nabootuje do základní obrazovky.

Dalším krokem je zapojení USB Wi-Fi adaptéru, který disponuje „Plug and play“, takže se stačí pouze připojit na požadovanou síť bez jakékoliv instalace ovládacích prvků. Výchozí nastavení přednastaveného uživatele „rpi“ v systému neobsahuje autentizaci pomocí hesla. Z bezpečnostních důvodů je uživatel nucen toto přednastavení sám změnit vytvořením uživatele s heslem. Dále je doporučeno po čerstvé instalaci operačního systému a dále v častých časových intervalech zadat v příkazové řádce tyto příkazy.

```
#Načtení balíků ze zdrojů a porovnání verzí balíků.
$ sudo apt-get update

#Inteligentní aktualizace nainstalovaných balíků na nejnovější verzi.
$ sudo apt-get dist-upgrade
```

Nyní je operační systém nainstalován a připraven k použití.

5.0.2 Instalace Mosquitto

Na začátku je důležité zmínit, že pro využití plné podpory Mosquitto je nutná prezence nejnovější verze Raspbianu tedy Jessie. Starší verze Raspbianu nemusí podporovat některé knihovny a je potřeba je dokompilovat ručně ze zdrojových kódů. Nyní k samotné instalaci, příkazy v této kapitole jsou zadávány do příkazové řádky. Jako první krok se musí systému pro správu software předat a potvrdit GPG veřejný klíč k repozitáři s balíčky Mosquitto.

```
#Stáhnutí GPG klíče.
$ wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

#Přidání do správy balíčků.
$ sudo apt-key add mosquitto-repo.gpg.key
```

Dále se vloží seznam odkazů pro použitou verzi operačního systému Raspbian (Jessie) do repozitáře */etc/apt/sources.list.d*.

```
#Změna aktivního repozitáře.
$ cd /etc/apt/sources.list.d/

#Obstarání seznamu odkazů pro verzi Jessie.
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

Nyní se obnoví databáze instalovatelných balíčků a může se nainstalovat samotný Mosquitto.

```
#Načtení balíků ze zdrojů a porovnání verzí balíků.
$ sudo apt-get update

#Instalace Mosquitto.
$ sudo apt-get install mosquitto
```

Jako další prvek je nainstalován balíček pro spustitelné klienty z příkazové řádky a Python vazbu.

```
#Instalace balíku pro práci s klientmi v terminálu.
$ sudo apt-get install mosquitto-clients

#Instalace Python vazby.
$ sudo apt-get install python-mosquitto
```

5.0.3 Konfigurace Mosquitto

Konfigurace Mosquitto brokeru se odehrává převážně ve výchozím adresáři */etc/mosquitto*, kde se také nachází konfigurační soubor *mosquitto.conf*. V této kapitole je čtenář seznámen s požadovanou konfigurací Mosquitto brokeru. Následující příkazy jsou součástí zmíněného konfiguračního souboru.

Základním prvkem je definování takzvaných posluchačů. Posluchači jsou definováni otevřenými síťovými porty a případně požadovaným protokolem.

```
#Definování základního posluchače s rezervovaným portem 1883 s výchozím
protokolem MQTT.
listener 1883

#Definování posluchuče pro komunikaci pomocí WebSockets s portem 9001.
listener 9001
protocol websockets
```

Nyní pro komunikaci s Mosquitto je otevřen síťový port 1883 a 9001. Další vlastností je perzistence zpráv. Funkce perzistence zpráv je v podstatě ukládání informací o odeslaných zprávách do souboru na disk, pro potřeby odpojených klientů. Toto řešení neslouží k použití databáze, ale jen pro případy výskytu použití vyšších QoS úrovní a persistentních klientů.

```
#Povolení vlastnosti persistence zpráv.
persistence true

#Vytvoření databázového souboru.
persistence_file mosquitto.db

#Nastavení lokace databázového souboru.
persistence_location /var/lib/mosquitto/
```

Následujícím krokem je nastavení logování požadovaných systémových zpráv do logovacího souboru pro informaci a případnou analýzu komunikace.

```
#Vytvoření logovacího souboru v daném adresáři.
log_dest file /var/log/mosquitto/mosquitto.log

#Výběr požadovaných typů zpráv, které je doporučené logovat.
log_type error
log_type warning
log_type notice
log_type information

#Nastavení logování připojení a odpojení klienta.
connection_messages true

#Nastavení logování časové známky.
log_timestamp true
```

Nyní je Mosquitto broker nakonfigurován do požadovaného stavu. Další etapa pojednává o inicializaci zabezpečení Mosquitto brokeru pro potřeby komunikačního systému.

5.0.4 Zabezpečení Mosquitto

Podle kapitoly o zabezpečení MQTT komunikace v teoretické části viz (2.3.2.3) se provedou následující kroky. Zabezpečení brokeru je prvořadým krokem k zabezpečení celého systému zařízení.

Síťová úroveň

Základním krokem je fyzické zabezpečení lokální Wi-Fi sítě. K zabezpečení Wi-Fi sítě je zapotřebí konfigurace přístupového bodu. V případě této práce je jako přístupový bod zvolen Wi-Fi směrovač TP-LINK TL-WR841N. Konfigurace probíhá na úrovni webového rozhraní samotného směrovače. Následující body reprezentují požadované zabezpečení lokální Wi-Fi sítě:

- Změna výchozích přihlašovacích údajů pro administraci AP.
- Šifrování sítě (doporučeně WPA2-PSK s AES šifrováním a bezpečným heslem).
- Změna SSID[16] (výchozí SSID může napomoci útočníkům k identifikaci značky AP).
- Zákaz vysílání SSID (diskutabilní).
- Filtrování síťových zařízení podle jejich fyzické identifikace (MAC adres).
- Omezení fyzického přístupu nepověřeným osobám k AP.

Transportní úroveň

Zabezpečení na transportní úrovni pojednává o zabezpečení komunikace technologií SSL/TLS na transportní úrovni. Požadavkem pro další kroky implementace je přítomnost nástroje openssl, většina Unixových distribucí by ho však měla obsahovat ve výchozím stavu. Pro vytvoření a nastavení certifikátů je použit script z projektu OwnTracks [36]. Vlastností tohoto scriptu je vygenerování vlastních certifikačních souborů, nadefinování hashovacích algoritmů a potřebných údajů k vytvoření reálného certifikátu (časová expirace certifikátu). Taktéž součástí scriptu je vytvoření vlastní certifikační autority spolu s klíči, která slouží k podepsání serverového certifikátu. Tento postup slouží pouze pro potřeby testování a ověření výsledků této práce. V případě publikování výsledků práce pro komerční účely se vyžadují oficiální postupy ověřené reálnou certifikační autoritou. Kroky ke splnění cílů jsou dále popsány.

[16] SSID (Service Set Identifier) – Identifikátor bezdrátové Wi-Fi sítě, který je periodicky vysílán.

```
#Stažení požadovaného scriptu.
$ wget https://github.com/owntracks/tools/raw/master/TLS/generate-CA.sh

#Spuštění scriptu.
$ bash ./generate-CA.sh
```

Nyní se vytvořilo 6 souborů. K následnému nastavení brokeru jsou potřeba implicitně následující tři soubory:

- ca.crt – veřejný certifikát certifikační autority
- raspberry.crt – vydaný veřejný certifikát pro účel brokeru
- raspberry.key – privátní klíč brokeru

Po vytvoření certifikačních souborů jsou tyto soubory následně zkopírovány do výchozích složek Mosquitto. Přístup k těmto složkám musí být nastaven výhradně pro správce systému.

```
#Zkopírování certifikátů do konfigurační složky Mosquitto.
$ sudo cp ca.crt /etc/mosquitto/ca_certificates
$ sudo cp raspberry.crt /etc/mosquitto/certs/
$ sudo cp raspberry.key /etc/mosquitto/certs/
```

Poté dojde ke konfiguraci samotného Mosquitto. Následující postup se odehrává v konfiguračním souboru *mosquitto.conf* .

```
#Nastavení portu 9883 s použitím protokolu WebSockets a přidáním cesty k
certifikačním souborům.
listener 9883
protocol websockets
cafile /etc/mosquitto/ca_certificates/ca.crt
certfile /etc/mosquitto/certs/raspberry.crt
keyfile /etc/mosquitto/certs/raspberry.key
```

Nyní po restartu Mosquitto je nastaven otevřený síťový port 9883 s protokolem WebSockets pro účely externí zabezpečené komunikace pomocí SSL/TLS s mobilní aplikací. Pro úplnost je doporučeno použití aktuální verse TLS v1.2.

```
#Nastavení verze TLS na verzi 1.2.
tls_version tlsv1.2
```

Aplikační úroveň

Prvním bezpečnostním prvkem na aplikační úrovni je možnost autentizace klienta. Autentizace je proces ověření proklamované identity subjektu. Když přijde na ověřování, MQTT protokol sám o sobě poskytuje klientovi možnost vlastnit autentizační údaje v podobě uživatelského jména a hesla. Tyto údaje se posílají v MQTT paketu, přesněji v CONNECT zprávě. Z tohoto důvodu má klient možnost poslat uživatelské jméno a heslo při připojování k MQTT brokeru.

V praxi to znamená, že v nastavení konfiguračního souboru musí být následovně zakázán anonymní přístup k brokeru.

```
#Zakázání anonymního přístupu.
allow_anonymous false
```

Nyní se se k brokeru můžou přihlásit jen registrovaní uživatelé. Pro registrované uživatele se následně vytvoří soubor pro správu hesel. Hesla jsou zahashovaná a přístup k němu musí být výhradně nastaven pouze pro správce.

```
#Přiřazení souboru pro správu hesel uživatelů v daném adresáři.
password_file /etc/mosquitto/pwfile
```

Vytvoření uživatele se provede v příkazové řádce následovně – *mosquitto_passwd -c /etc/mosquitto/pwfile „uživ. jméno“*. Pro testovací účely jsou vytvořeni klienti s uživatelskými jmény „klient_senzory“ pro účely autentizace výkonného prvku a „klient_aplikace“ pro účely autentizace mobilní aplikace.

Pokračujícím bezpečnostním krokem po autentizaci uživatelů je jejich autorizace. Autorizace je proces získávání souhlasu s provedením nějaké operace, povolení přístupu někam, k někomu nebo něčemu. MQTT klienti mají po úspěšném připojení k brokeru pouze dvě elementární kompetence, publikovat nebo odebírat zprávy. Bez jakékoliv autorizace může jakýkoliv klient publikovat anebo odebírat jakékoliv téma.

Pro správu autorizace uživatelů je nastaven soubor právě pro tyto účely. K tomuto souboru musí mít přístup pouze správce.

```
#Přiřazení souboru pro správu autorizací uživatelů v daném adresáři.
acl_file /etc/mosquitto/aclfile
```

Pro testovací účely mají oba klienti přiřazená následující přístupová práva. Tyto práva se nastavují ve zmíněném souboru *aclfile*.

```
#Klienti „klient_senzory“ a „klient_aplikace“ mohou publikovat a odebírat
pouze Wild témata „senzory/#” a „aktuatory/#”.
user klient_senzory
topic readwrite senzory/#
topic readwrite aktuatory/#

user klient_aplikace
topic readwrite senzory/#
topic readwrite aktuatory/#
```

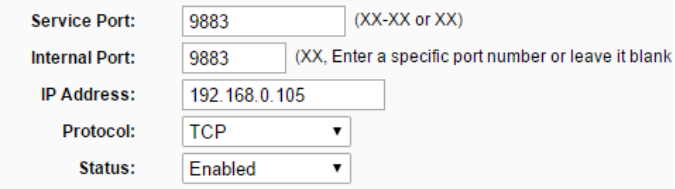
Nyní jsou v systému pouze dva klienti, kteří mohou odebírat a publikovat zprávy pod tématy *senzory/#* a *aktuatory/#*. Vše ostatní nemá přístup k ničemu. Současně lze Mosquitto broker považovat za zabezpečený a připravený k dalšímu kroku.

5.0.5 Přístup z internetu

Aby bylo možné pomocí mobilní aplikace komunikovat vzdáleně s domácí lokální sítí, musí být centrální bod v podobě MQTT brokeru dostupný z internetu. K dosažení tohoto cíle slouží metoda přesměrování portu tzv. „port forwarding“. Tato metoda slouží ke směrování portů z jednoho síťového uzlu na druhý. Typickým použitím je umožnění vnějšímu uživateli připojit se na síťový port na privátní adrese v lokální sítí prostřednictvím směrovače. Možnosti působnosti takového směrování portů, ale také ovšem závisí na poskytovateli internetového připojení a infrastruktuře počítačové sítě v podobě dalších směrovačů, firewallů, překladačů síťových adres (NAT) nebo třeba vlastnictví dynamické IP adresy. Tyto možnosti můžou být potenciálním omezením. Přístup z internetu také samozřejmě otevírá možnosti útočníkům a dává tak systému nové rozměry zranitelnosti a ohrožení zvenčí, proto je nutností mít zveřejňovaný systém předem dostatečné zabezpečený.

Konfigurace směrování portů se nastavuje ve webovém rozhraní samotného směrovače. Metodika nastavení je však různá od výrobce, od kterého je směrovač vyroben. Prvním krokem je doporučeno si permanentně zarezervovat fyzickou MAC adresu k příslušné privátní IP adrese. MQTT broker tak dostane přidělenou vždy stejnou privátní IP adresu od DHCP

serveru. Dalším krokem je zmíněné přesměrovaní portu viz (Obrázek 5-1), kde se nastaví privátní IP adresa MQTT brokeru, požadovaný síťový port (9883 – Secure Websockets) a protokol transportní vrstvy (TCP).



Obrázek 5-1: Ukázka přesměrování portu na směrovači.

5.0.6 Ověření výsledků

Ověření konfigurace připojení je zobrazena pomocí tabulky aktivních síťových spojení viz (Obrázek 5-2).

- Port 1883 – MQTT
- Port 8883 – Secure MQTT
- Port 9001 – Websockets
- Port 9883 – Secure Websockets

```
pi@raspberrypi:~ $ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:9883            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:1883            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8123            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9001            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8883            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
```

Obrázek 5-2: Tabulka aktivních síťových spojení

Ověření realizace publikování a odebírání požadovaných témat, požadovanými klienty je zobrazena pomocí příkazů *mosquitto_pub* a *mosquitto_sub* viz (Obrázek 5-3). Hesla klientů jsou zakrytá.

```
pi@raspberrypi:~ $ mosquitto_pub -t senzory/svetlo -m 0 -d -u klient_senzory -P [redacted]
Client mosqpub/14543-raspberry sending CONNECT
Client mosqpub/14543-raspberry received CONNACK
Client mosqpub/14543-raspberry sending PUBLISH (d0, q0, r0, m1, 'senzory/svetlo', ... (1 bytes))
Client mosqpub/14543-raspberry sending DISCONNECT

pi@raspberrypi:~ $ mosquitto_sub -t senzory/svetlo -d -u klient_aplikace -P [redacted]
Client mosqsub/1252-raspberryp sending CONNECT
Client mosqsub/1252-raspberryp received CONNACK
Client mosqsub/1252-raspberryp sending SUBSCRIBE (Mid: 1, Topic: senzory/svetlo, QoS: 0)
Client mosqsub/1252-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/1252-raspberryp received PUBLISH (d0, q0, r1, m0, 'senzory/svetlo', ... (1 bytes))
1
Client mosqsub/1252-raspberryp received PUBLISH (d0, q0, r0, m0, 'senzory/svetlo', ... (1 bytes))
0
```

Obrázek 5-3: Ověření požadovaného publikování a odebírání dat

Ověření autentizace je provedena pomocí zkoušky publikování dat bez přihlašovacích údajů viz (Obrázek 5-4).

```
pi@raspberrypi:~ $ mosquitto_pub -t testauntetizace -m zprava -d
Client mosqpub/14677-raspberry sending CONNECT
Client mosqpub/14677-raspberry received CONNACK
Connection Refused: not authorised.
Error: The connection was refused.
```

Obrázek 5-4: Ověření funkční autentizace

Ověření autorizace není součástí specifikace aktuální verze protokolu MQTT 3.1.1. Broker není nijak definován pro kontaktování klienta, že není autorizován. Ověření funkční autorizace tedy probíhalo na úrovni poslané zprávy a určení, zda byla přijata či nikoliv.

Ověření funkčního zabezpečujícího prvku SSL/TLS je uskutečněno pomocí publikování zprávy za použití veřejného certifikátu viz (Obrázek 5-5).

```
pi@raspberrypi:~ $ mosquitto_pub --cafile /etc/mosquitto/ca_certificates/ca.crt -t senzory -m t
est -d -u klient_aplikace -P [redacted] -p 8883
Client mosqpub/14343-raspberry sending CONNECT
Client mosqpub/14343-raspberry received CONNACK
Client mosqpub/14343-raspberry sending PUBLISH (d0, q0, r0, m1, 'senzory', ... (4 bytes))
Client mosqpub/14343-raspberry sending DISCONNECT
```

Obrázek 5-5: Ověření funkčního zabezpečujícího prvku SSL/TLS

Rovněž je ověřeno připojení klienta a uskutečnění handshake protokolu viz (Obrázek 5-6).

```
 8 18.665699  192.168.0.103  192.168.0.105  TCP         66 53197→8883 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10 18.794707  192.168.0.103  192.168.0.105  TCP         54 53197→8883 [ACK] Seq=1 Ack=1 Win=65536 Len=0
11 18.796807  192.168.0.103  192.168.0.105  TLSv1.2    316 Client Hello
15 18.905346  192.168.0.103  192.168.0.105  TCP         54 53197→8883 [ACK] Seq=263 Ack=2729 Win=65536 Len=0
16 18.920893  192.168.0.103  192.168.0.105  TLSv1.2    180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
19 18.945208  192.168.0.103  192.168.0.105  TLSv1.2    172 Application Data
21 19.011897  192.168.0.103  192.168.0.105  TCP         54 53197→8883 [ACK] Seq=507 Ack=3004 Win=65280 Len=0
53 28.984865  192.168.0.103  192.168.0.105  TLSv1.2     85 Application Data
```

Obrázek 5-6: Ověření prezence handshake protokolu

Přiřazení zvoleného síťového portu k lokální adrese MQTT brokeru je zobrazena pomocí výstupu z webového rozhraní Wi-Fi směrovače viz (Obrázek 5-7).

ID	Service Port	Internal Port	IP Address	Protocol	Status
1	9001	9001	192.168.0.105	TCP	Enabled
2	9883	9883	192.168.0.105	TCP	Enabled

Obrázek 5-7: Požadované síťové porty přiřazené k lokální IP adrese

5.1 Výkonný prvek

Pro realizaci lokální klientské strany slouží soustava zařízeních zmíněná v kapitole viz (3.0.2). Pro vyvinutí zdrojového kódu vývojové desky je zvolen vývojářský software Energia. Energia je open-source vývojové prostředí založeno Robertem Wesselsem v lednu 2012 s cílem zpřístupnit platformy Wiring a Arduino Framework k potřebě vývojových desek Texas Instruments. Energia je cross platformový nástroj, podporuje operační systémy Mac OS, Windows a Linux. [37]

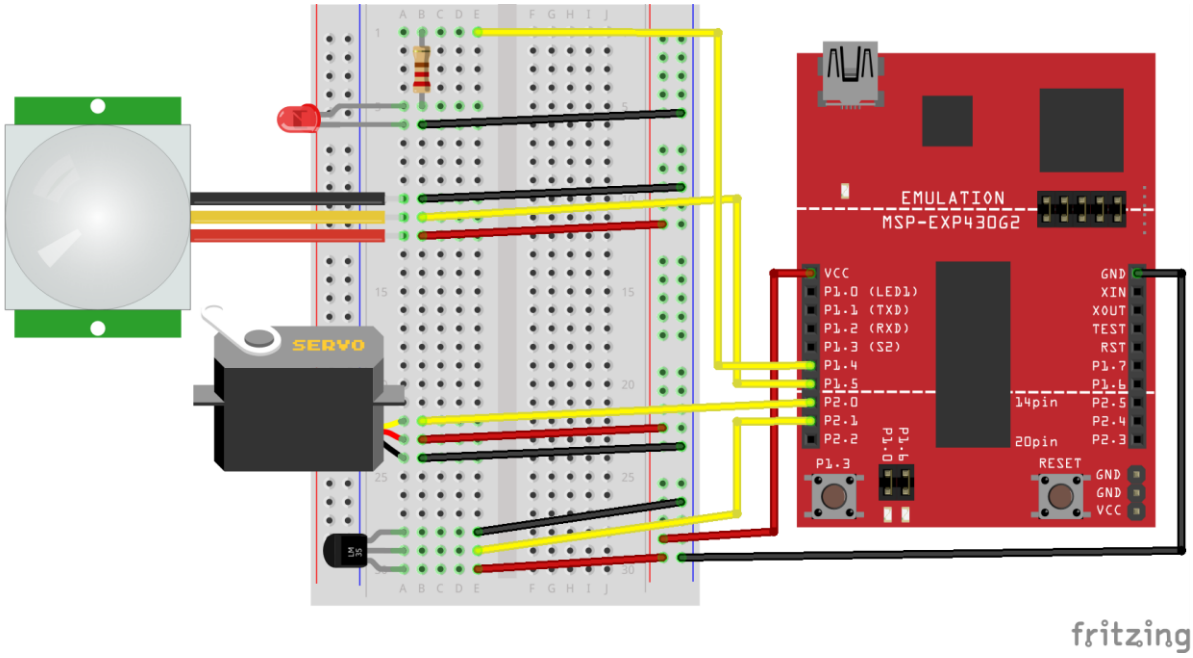
5.1.1 Instalace vývojových platforem

Nejprve se Launchpad připojí přes micro USB do USB sběrnice počítače. Na oficiálních webových stránkách Texas Instruments je u každého z jejich produktů velmi bohatý výběr dokumentů a ovladačů. Z této webové stránky se následně po jejich stažení vybrané ovladače nainstalují.

5.1.2 Schéma zapojení

Pro demonstraci funkční komunikace s mobilní aplikací je zvolen příkladný senzorový systém viz (3.0.2.3). Senzory a aktuátory jsou s vývojovou deskou propojeny pomocí nepájivého pole. V případě zapojení LED je použit u anodové nožičky rezistor o odporu 220 Ω. Ostatní prvky jsou připojeny rovnou k výstupním pinům a tvoří elektrický obvod pomocí ground a power pinů. Pro vykonání požadovaných akcí je zapotřebí vyvinout a nahrát do paměti zdrojový kód.

Následuje zobrazení schéma zapojení výkonného prvku viz (Obrázek 5-8). Nástroj Fritzing ve kterém je schéma vytvořeno, bohužel neobsahuje identický Launchpad, proto je nahrazen jiným modelem.

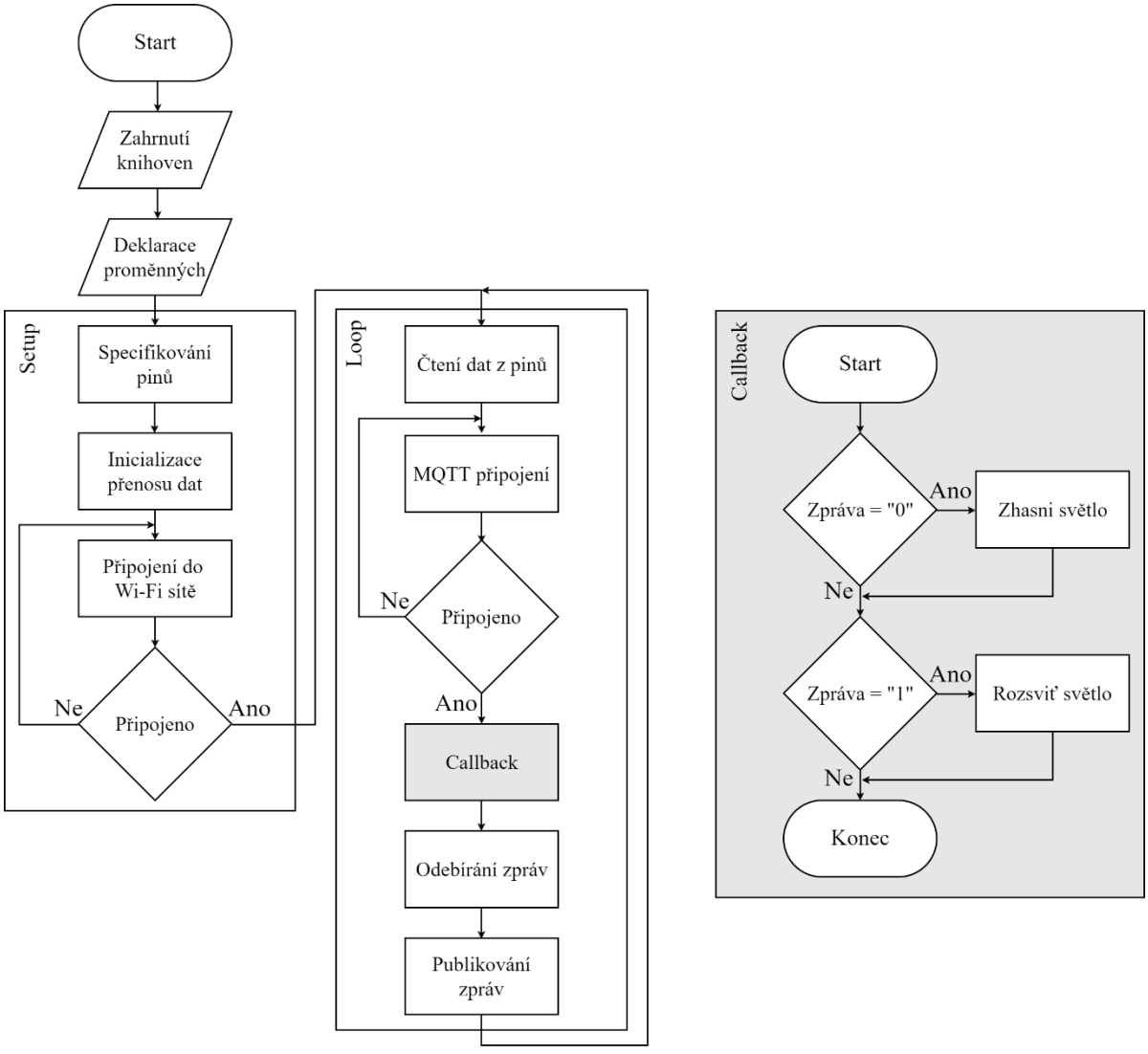


Obrázek 5-8: Schéma zapojení výkonného prvku

5.1.3 Vývoj firmware

Vývojový diagram

Logika jednotlivých kroků zdrojového kódu je znázorněna pomocí následujícího vývojového diagramu viz (Obrázek 5-9).



Obrázek 5-9: Vývojový diagram aplikace výkonného prvku

Knihovny

Pro vývoj firmwaru jsou použity následující knihovny.

```
//Knihovna pro práci s Wi-Fi připojením.
#include <WiFi.h>

//PubSubClient knihovna pro práci s MQTT připojením viz (3.1.2).
#include <PubSubClient.h>

//Knihovna pro práci s SPI komunikací. (V případě této práce, je tato
knihovna použita z důvodů prezence připojení CC3100 k Launchpadu).
#include <SPI.h>

//Knihovna pro práci se Servem.
#include <Servo.h>
```

Zdrojový kód

Zdrojový kód po definování knihoven a deklaraci proměnných dále obsahuje 4 hlavní funkce. Všechny tyto funkce jsou typu *void* bez návratového typu. V dalším textu jsou přiblíženy.

Funkce setup

Funkce setup pojednává o specifikování zvolených pinů typu INPUT nebo OUTPUT. Dále realizuje sériový přenosu dat o rychlosti 115200 bitů/s. Důležitou vlastností této funkce je inicializování pokusu o připojení se k Wi-Fi síti. Pokud se nepřipojí k Wi-Fi síti čeká, dokud se nepřipojí.

Funkce callback

Funkce callback obstarává stav situace, při které dojde k přijetí odeslané zprávy. Pro představu, když přijde zpráva s obsahem „1“ dojde k zapnutí LED světla. Obsah zprávy je datového typu char.

```
if ((char)payload[0] == '1'){
  digitalWrite(LED, HIGH);
}
```

Funkce loop

Jak už název napovídá, jedná se o typ funkce, která je vykonávaná v nekonečné smyčce. Tato funkce začíná čtením dat z teplotního čidla a následného převodu z voltů na stupně Celsia, zároveň se čtou data vygenerovaná z PIR senzoru. Tyto data jsou dále převedena na datový typ char array. Hlavní částí této funkce je však optimalizované publikování a odebírání dat podle tématu a zároveň podle klienta na zadaný MQTT broker a síťový port. Pro interpretaci je vybrána část kódu s odebíráním tématu „aktuatory/svetlo“ klientem „klient_senzory“

```
if(!client.connect("Aktivni_prvek", "klient_senzory", "Heslo")) {
      Serial.println("Pripojeni se nezdarilo");
    } else {
      Serial.println("Pripojeni se zdarilo");

      if(client.subscribe("aktuatory/svetlo")) {
        Serial.println("Odebirani uspesne");
      }
    }
```

Pro další demonstraci je zvolena část kódu s publikováním dat z teplotního čidla. Z důvodů optimalizace je tato podmínka zaštítěna další časovou podmínkou, která tak zvětší interval mezi odeslanými zprávami na 20 sekund.

```
long now = millis();
if(now - lastMsg > 20000) {
    lastMsg = now;
  if(client.publish("senzory/teplota", char_array)) {
    Serial.println();
    Serial.print("Publikovani uspesne. – Tema: senzory/teplota");
    Serial.print(" Zprava: ");
    Serial.print(teplota);
  } else {
    Serial.println("Publikovani neuspesne");
  }
}
```

Funkce WifiStatus

Tato funkce slouží čistě pro informativní účel. Činností této funkce je vypsání přidělené IP adresy od DHCP serveru, vypsání názvu připojené sítě a Wi-Fi signálu v jednotkách dBm.

5.1.4 Ověření výsledků

Ověření funkčního navázání obousměrné komunikace, publikování a odebírání požadovaných zpráv je zobrazeno pomocí výstupního seriálového okna aplikace viz (Obrázek 5-10).

```
okus o pripojeni k Wi-Fi siti: alanturing
.
Pripojeno.
Cekani na prirazeni IP adresy.
.
IP adresa obdrzena.

SSID: alanturing
IP adresa: 192.168.0.102
Sila signalu (RSSI):-63 dBm

Nepripojeno. Pripojovani....
Pripojeni se zdarilo.
Odebirani uspesne.
Odebirani uspesne.

Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/svetlo Delka: 1 Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/zaluzie Delka: 1 Zprava: 3
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/svetlo Delka: 1 Zprava: 1
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/svetlo Delka: 1 Zprava: 0
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/zaluzie Delka: 1 Zprava: 2
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Obdrzena zprava.     - Tema: aktuatory/zaluzie Delka: 1 Zprava: 3
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
Publikovani uspesne. - Tema: senzory/pohyb Zprava: 0
```

Obrázek 5-10: Ověření navázání obousměrné komunikace výkonného prvku s brokerem

5.2 Mobilní aplikace

Pro realizaci externího klientského prvku slouží hybridní mobilní aplikace zmíněná v předchozích kapitolách viz (3.1.1.1). Jako vývojové prostředí pro vyvíjení samotné mobilní aplikace je použit Visual Studio Code. Visual Studio Code je nová odlehčená multiplatformní open-source vývojářská platforma od Microsoftu pro realizaci a vývoj aplikací všeho druhu Jako front-end framework je zvolený Ionic, který ve spolupráci s Cordovou a AngularJS vytváří skvělé podmínky pro tvorbu hybridních aplikací. Pro uskutečnění tohoto vývoje je využit operační systém Windows 10.

5.2.1 Instalace Ionic

Mezi cílené platformy Ionic patří mobilní operační systémy Android verze 4.1 a vyšší a zároveň iOS 7 a vyšší. Pro vývoj jedné z těchto platforem s využitím Apache Cordova je doporučeno řídit se oficiálními návody, které jsou k nalezení na webových stránkách Apache Cordova. V případě vyvíjení Android aplikace je nutností prezence Java JDK nástrojů a Android Studia. Nyní k samotné instalaci.

```
//Instalace Apache Cordova (Nutné mít nainstalovaný Node.js).
$ npm install -g cordova

//Instalace Ionic.
$ npm install -g ionic

//Vytvoření nového projektu.
$ ionic start todo blank

//Rozhodnutí o výběru cílené platformy (obojí je možné).
$ ionic platform add ios
$ ionic platform add android
```

Nyní je vše připraveno pro vývoj aplikace.

5.2.2 Vývoj aplikace

Knihovny

Pro vývoj aplikace jsou použity následující knihovny.

```
#Knihovna pro práci s Ionic/Angularjs.
<script src="lib/ionic/js/ionic.bundle.js"></script>

#Knihovna Paho Javascript Client viz (3.1.2) pro práci s MQTT připojením.
<script src="lib/Paho/mqttws31.js"></script>

#Knihovna pro práci s Cordovou.
<script src="cordova.js"></script>
```

Zdrojový kód

Zdrojový kód se skládá ze tří hlavních částí. Javascriptová část má na starost chování aplikace. HTML část představuje strukturu aplikace. CSS část uskutečňuje prezentaci aplikace. Vzhledem k účelům a patřičnosti této práce je probrána pouze Javascriptová funkční část. Celý Ionic projekt je odkazován v přílohách práce. Po deklaraci globálních a konfiguračních proměnných následují funkce, které jsou dále vysvětleny. Vybrány jsou nejvíce podstatné pro přiblížení konceptu aplikace.

Funkce connect

Tato funkce vytváří klienta s konfiguračními parametry a zprostředkovává zahájení zabezpečeného spojení s MQTT brokerem. Funkce je také volána v html části, kde se spustí po načtení webové stránky. V ukázce kódu je proměnná *options*, ve které je nastaveno použití SSL/TLS na hodnotu *true*.

```
    var options = {
        onSuccess: onConnect,
        onFailure: onFail,
        userName: "klient_aplikace",
        password: "Heslo",
        useSSL: true,
    };
```

Funkce onConnect

Funkce onConnect zajišťuje situaci po připojení klienta. Nastavuje například možnosti úrovně kvality servisu (QoS) a nastavuje klientovi odebírané téma.

```
    options = {
        qos: 0,
        onSuccess: function (context) {},
        invocationContext: {foo: true},
    };

    client.subscribe(teplota_topic, options);
    client.subscribe(pohyb_topic, options);
```

Funkce onMessageArrived

Jak už napovídá název, tato funkce obstarává situaci po přijetí zprávy. Zde jsou nadefinované rozhodovací podmínky pro přijaté zprávy. V ukázkovém kódu je demonstrování situace přijetí dat z teplotního čidla.

```
if (message.destinationName == teplota_topic) {
        var teplota_napis = document.getElementById("teplota_status");
        teplota_napis.innerHTML = "Teplota v pokoji je " + message.payloadString
+ " &deg;C";
    }
```

Funkce svetlo_tlacitko

Funkce svetlo_tlacitko definuje co se má stát, když uživatel aplikace stiskne tlačítko zapnutí světla na displeji mobilu. Tato funkce je zde interpretovaná celá. V této části se také objevuje jedna z užitečných funkcí MQTT protokolu tzv. Retained message (zachovaná zpráva). Tato zpráva je typická zpráva, ale zároveň je označená za zachovanou a pod určitým tématem je uložena v brokeru. V případě klienta, který začne odebírat toto téma dostane tuto zprávu ihned a nemusí čekat na publikaci této zprávy od klienta.

Zvolená úroveň QoS je 0. Z důvodu maximální podpory QoS úrovní u PubSubClient knihovny, která je právě pouze 0. Jelikož QoS úrovně fungují na principu degradace a snižují se podle úrovně příjemce.

```
function svetlo_tlacitko() {
    if (svetlo_on) {
        var payload = "0";
        svetlo_on = false;
    } else {
        var payload = "1";
        svetlo_on = true;
    }

    message = new Paho.MQTT.Message(payload);
    message.destinationName = svetlo_topic;
    message.retained = true;
    message.qos = 0;
    client.send(message);
```

5.2.3 Testování aplikace

Testování aplikace lze uskutečnit v několika úrovních. První úroveň pojednává o testování aplikace ve webovém prohlížeči. Tento způsob je nejpohodlnější a nejrychlejší, po uložení projektu se změny v aplikaci automaticky obnoví.

```
#Testování aplikace ve webovém prohlížeči.
$ ionic serve
```

Druhá úroveň využívá simulací reálných operačních systémů Android a iOS.

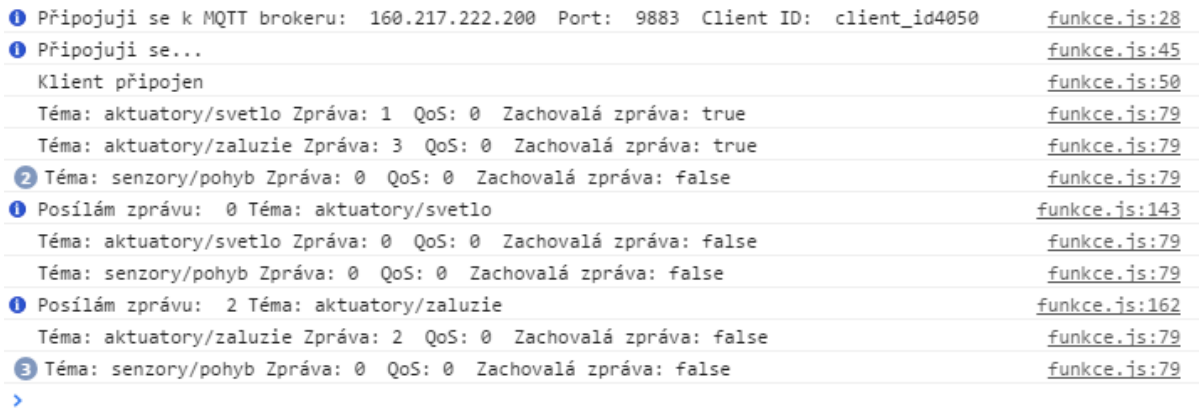
```
#Testování aplikace v simulačních prostředích reálných operačních systémů.
$ ionic build ios
$ ionic emulate ios

$ ionic build android
$ ionic emulate android
```

Jelikož hybridní aplikace je v principu nativní aplikace, může být také testována jako nativní aplikace na reálném zařízení. Tento typ testování je samozřejmě nejefektivnější, z důvodů reálného chování mobilního zařízení. Nicméně pro tento typ testování musí uživatelé iOS vlastnit vývojářský účet, který stojí 99 dolarů ročně. Uživatelé operačního systému Android můžou potenciál tohoto typu testování využít zdarma. Podmínkou pro uživatele operačního systému Android je povolení vývojářského režimu v mobilním zařízení.

```
#Testování aplikace na reálném zařízení. (Po připojení mobilního zařízení
pomocí USB sběrnice, je zadán následující příkaz.)
$ ionic run android
```

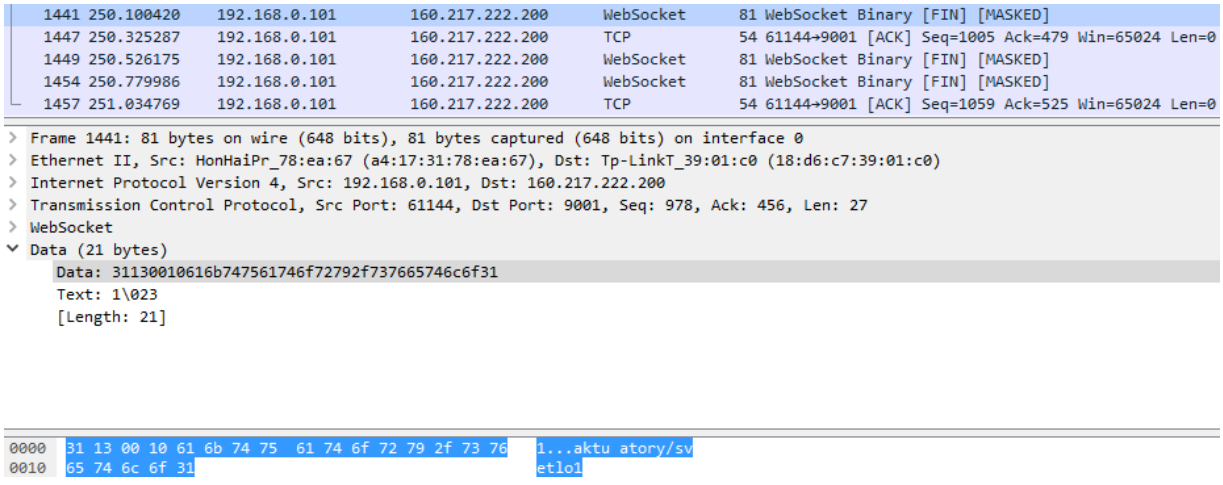
5.2.4 Ověření výsledků

Ověření navázání obousměrné komunikace technologií WebSockets s MQTT brokerem je zobrazeno pomocí vývojářské konzole ve webovém prohlížeči Chrome viz (Obrázek 5-11).

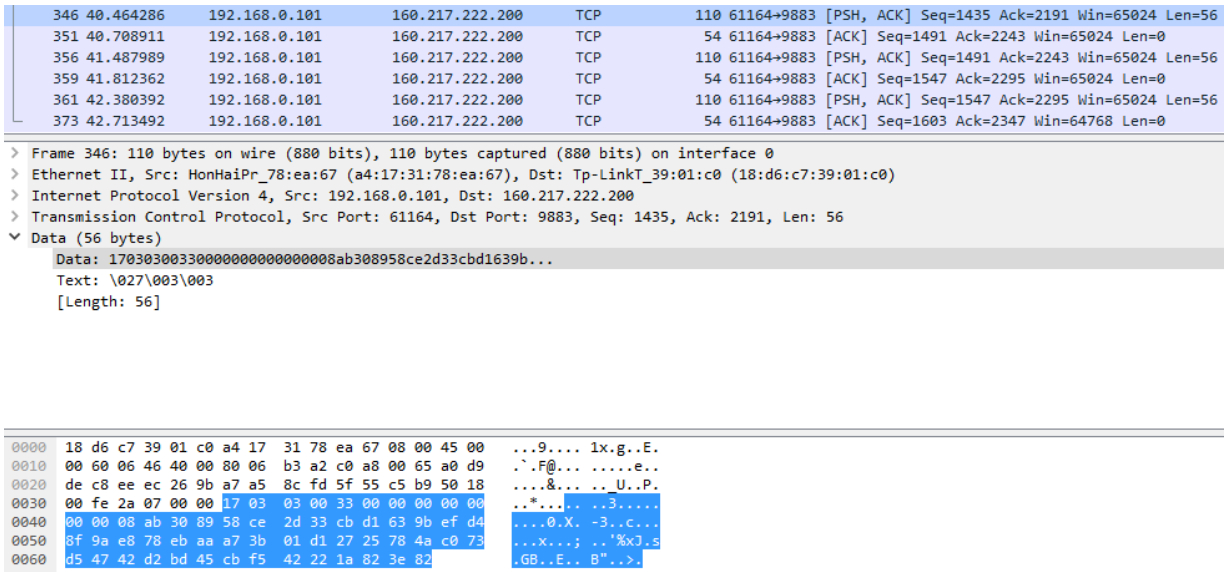


Obrázek 5-11: Ověření navázání obousměrné komunikace mobilní aplikace s brokerem

Ověření realizace šifrované komunikace s MQTT brokerem je provedeno pomocí nástroje Wireshark, který slouží pro detekování a analýzu síťového provozu. Pro test je na následujících obrázcích ukázka zachycení poslané zprávy „1“ pod tématem *aktuatory/svetlo*. Na prvním obrázku je odposlechnut WebSocketový paket nezabezpečené komunikace na síťovém portu 9001 viz (Obrázek 5-12). Na druhém obrázku je zachycena ta samá zpráva pomocí zabezpečené komunikace na síťovém portu 9883 viz (Obrázek 5-13).



Obrázek 5-12: Monitorování nezabezpečeného síťového portu



Obrázek 5-13: Monitorování zabezpečeného síťového portu

5.2.5 Publikace aplikace

Vzhledem k patřičnosti cílům a funkcím mobilní aplikace, není aplikace publikována do jednoho z oficiálních obchodů jako je Google Play nebo App Store. Výsledná aplikace je určena pro demonstrování funkcionality komunikačního systému a možnému osobnímu použití nebo inspiraci k dalšímu vývoji. Pro vygenerování takzvané nepodepsané aplikace slouží následující příkaz.

```
#Vygenerování nepodepsané výsledné aplikace pro Android.
$ cordova build --release android

#Vygenerování nepodepsané výsledné aplikace pro iOS.
$ ionic build ios --release
```

6 Testování

Tato kapitola pojednává o zvoleném testování systému a řešených problémech, které doprovázely vývoj samotného systému.

6.0 Zátěžové testování

Z důvodu myšlenky možného budoucího rozšíření o další prvky v podobě klientů a dalších připojených zařízení k MQTT brokeru, je užitečné vědět jaké se můžou od systému očekávat kapacitní a stabilitní schopnosti. Zátěžové testování je proces uvedení požadavků na softwarový systém nebo výpočetní zařízení a měření jeho odezvy. Zátěžové testy se provádí za účelem určení chování systému, v rámci normálních a předpokládaných zatěžujících podmínek. Toto testování pomáhá určit maximální provozní kapacitu aplikace, případné překážky nebo například určení prvku který způsobuje degradaci. Zátěžovému testování je podroben MQTT broker Mosquitto verze 1.14.11 viz (3.0.1.1) hostovaném na minipočítači Raspberry Pi 2 model B viz (3.0.1.2). K tomuto úkolu je zvolen testovací nástroj „Malaria“ [38]. Principem nástroje Malaria je otestování odezvy MQTT brokeru pod specifikovaným náporem ve formě virtuálních klientů, kteří paralelně publikují předem definovaný počet zpráv za daný čas. Jakým způsobem je publikování zpráv provedeno, záleží na samotném uživateli testovacího nástroje.

Účelem testování je analýza statistiky úspěšně poslaných a přijetých zpráv. Rovněž další částí je analýza odezvy MQTT brokeru při zátěži ve formě využitých procesorových prostředků. Následuje popis připravení nástrojů pro testování.

```
#Nainstalování pythonovské MQTT knihovny, která je zapotřebí pro nástroj
Malaria.
$ pip install paho-mqtt
$ git clone
git://fit.eclipse.org/gitroot/paho/org.eclipse.paho.mqtt.python.git
$ cd paho.mqtt.python
$ sudo python setup.py install

#Nainstalování samotného nástroje Malaria.
$ git clone https://github.com/remakeelectric/mqtt-malaria
$ cd mqtt-malaria
$ sudo python setup.py install
```

Podstatným krokem pro zprovoznění testovacích nástrojů je povolení přístupu neregistrovaných uživatelů, jelikož tento nástroj nepodporuje autentizační mechanismus. Testování probíhá pouze publikací zpráv na síťovém portu 1883, tedy při použití klasického protokolu MQTT. Mezi výstupní statistické hodnoty patří celkový čas přijetí publikovaných zpráv, celkový počet zpráv, počet zpráv za sekundu a úspěšnost přijetí publikovaných zpráv.

Monitorování odezvy procesorové jednotky Raspberry Pi při testování je realizováno pomocí nástroje „top“, který slouží primárně pro monitorování Unixových procesů. Tento nástroj by měl být součástí většiny Unixových systémů. Proces Mosquitto je následně monitorován v době spuštěných testů.

```
#Selekce procesu Mosquitto pro monitorování.
$ top -p „PID procesu Mosquitto“
```

Cílem pozorování tohoto procesu je zaznamenání procesorového času potřebného pro běh procesu Mosquitto při spuštěném testování. Procesorový čas je množství času, pro který je CPU použit pro zpracování instrukcí počítačového programu. Výstupní hodnota procesorového času je v jednotkách procent. Monitorování je realizováno pomocí tzv. „Solaris módu“. Tento mód vypisuje výstupní hodnoty využití CPU času dělené celkovým počtem procesorů, který daný systém obsahuje. Tím pádem celkové procento je zmenšeno na základě počtu procesorových jader a celková hodnota tak nemůže přesahovat 100 %. Specifikace parametrů Raspberry Pi 2 model B se nachází v kapitole viz (3.0.1.2). Nyní je vše připraveno k testování. Dále jsou představeny testovací scénáře a jejich výsledky. Tyto scénáře jsou seřazeny podle zátěžové náročnosti (od nejméně náročného).

6.0.1 Aktuální stav

Scénář „Aktuální stav“ se snaží simulovat přibližný stav právě vytvořeného prostředí.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 4 (jeden proces lze chápat jako jedno paralelní téma nebo klient pod kterým se publikují zprávy)
- Počet zpráv/1 proces – 600
- Velikost jedné zprávy – 2 B (obsah MQTT paketu – PUBLISH zprávy)
- Poslaných zpráv za sekundu/1 proces – 1
- Úroveň QoS – 0
- Požadovaná doba vykonání – 10 minut (600 s)

Příkaz k zadání:

```
$ malaria publish -P 4 -n 600 -s 2 -T 1 -q 0 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Aktuální stav	4	600	2 B	1	0

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

	Celkový čas	Přijatých zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	610,97 s	2 400	0,1 %	3,99	100 %
Výsledky II	610,02 s	2 400	0,1 %	3,99	100 %
Výsledky III	612,97 s	2 400	0,1 %	3,99	100 %

Tabulka 6-1: Zadání a výsledky testovacího scénáře "Aktuální stav"

6.0.2 Předpokládaný stav

Scénář „Předpokládaný stav“ simuluje stav potenciálního plánovaného prostředí po budoucím rozšíření o další zařízení.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 16
- Počet zpráv/1 proces – 600
- Velikost jedné zprávy – 4 B
- Poslaných zpráv za sekundu/1 proces – 1
- Úroveň QoS – 1
- Požadovaná doba vykonání – 10 minut

Příkaz k zadání:

```
$ malaria publish -P 16 -n 600 -s 4 -T 1 -q 1 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Předpoklád. stav	16	600	4 B	1	1

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

	Celkový čas	Celkový počet zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	613,05 s	9 600	0,2 %	15,98	100 %
Výsledky II	613,03 s	9 600	0,2 %	15,98	100 %
Výsledky III	615,05 s	9 600	0,2 %	15,95	100 %

Tabulka 6-2: Zadání a výsledky testovacího scénáře "Předpokládaný stav"

6.0.3 Možný stav

Scénář „Možný stav“ simuluje stav nadsazeného prostředí domácí automatizace ve které by se potenciálně mohl MQTT broker vyskytovat.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 40
- Počet zpráv/1 proces – 1 200
- Velikost jedné zprávy – 10 B
- Poslaných zpráv za sekundu/1 proces – 2
- Úroveň QoS – 1
- Požadovaná doba vykonání – 10 minut

Příkaz k zadání:

```
$ malaria publish -P 40 -n 1200 -s 100 -T 2 -q 1 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Možný stav	40	1 200	10	2	1

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

	Celkový čas	Celkový počet zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	616,11 s	48 000	0,9 %	79,80	100 %
Výsledky II	615,04 s	48 000	0,8 %	79,76	100 %
Výsledky III	615,13 s	48 000	0,8 %	73,73	100 %

Tabulka 6-3: Zadání a výsledky testovacího scénáře "Možný stav"

6.0.4 Zátěžový stav I

Scénář „Zátěžový stav I“ simuluje stav nereálného prostředí při velké zátěži, během kterého je otestována odezva MQTT brokeru.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 70
- Počet zpráv/1 proces – 3 000
- Velikost jedné zprávy – 20 B
- Poslaných zpráv za sekundu/1 proces – 5
- Úroveň QoS – 1
- Požadovaná doba vykonání – 10 minut

Příkaz k zadání:

```
$ malaria publish -P 70 -n 3000 -s 20 -T 5 -q 1 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Zátěžový stav I	70	3 000	20	5	1

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

	Celkový čas	Celkový počet zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	616,21 s	210 000	3,4 %	347,66	100 %
Výsledky II	617,24 s	210 000	3,2 %	347,53	100 %
Výsledky III	618,13 s	210 000	3,4 %	347,54	100 %

Tabulka 6-4: Zadání a výsledky testovacího scénáře "Zátěžový stav I"

6.0.5 Zátěžový stav II

Scénář „Zátěžový stav II“ simuluje stav nereálného prostředí při větší zátěži, během kterého je otestována odezva MQTT brokeru.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 100
- Počet zpráv/1 proces – 6 000
- Velikost jedné zprávy – 100 B
- Poslaných zpráv za sekundu/1 proces – 10
- Úroveň QoS – 1
- Požadovaná doba vykonání – 10 minut

Příkaz k zadání:

```
$ malaria publish -P 100 -n 6000 -s 100 -T 10 -q 1 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Zátěžový stav II	100	6 000	100	10	1

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

	Celkový čas	Celkový počet zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	627,09 s	600 000	7 %	978,15	100 %
Výsledky II	627,47 s	600 000	6,8 %	977,96	100 %
Výsledky III	626,63 s	600 000	6,8 %	978,64	100 %

Tabulka 6-5: Zadání a výsledky testovacího scénáře "Zátěžový stav II"

6.0.6 Zátěžový stav III

Scénář „Zátěžový stav III“ simuluje stav nereálného prostředí při největší zátěži, během kterého je otestována odezva MQTT brokeru.

Tento scénář zahrnuje následující specifikace:

- Počet procesů – 200
- Počet zpráv/1 proces – 9 000
- Velikost jedné zprávy – 1 kB
- Poslaných zpráv za sekundu/1 proces – 15
- Úroveň QoS – 1
- Požadovaná doba vykonání – 10 minut

Příkaz k zadání:

```
$ malaria publish -P 200 -n 9000 -s 1000 -T 15 -q 1 -H 160.217.222.200
```

Tabulka se vstupními a výstupními hodnoty:

	Počet procesů	Počet zpráv	Velikost zprávy (B)	Zpráv/s	QoS
Zátěžový stav III	200	9 000	1000	15	1

Se stejnými vstupními hodnotami se test provádí třikrát. Následující agregované výstupní hodnoty jsou aritmetickým průměrem všech procesů.

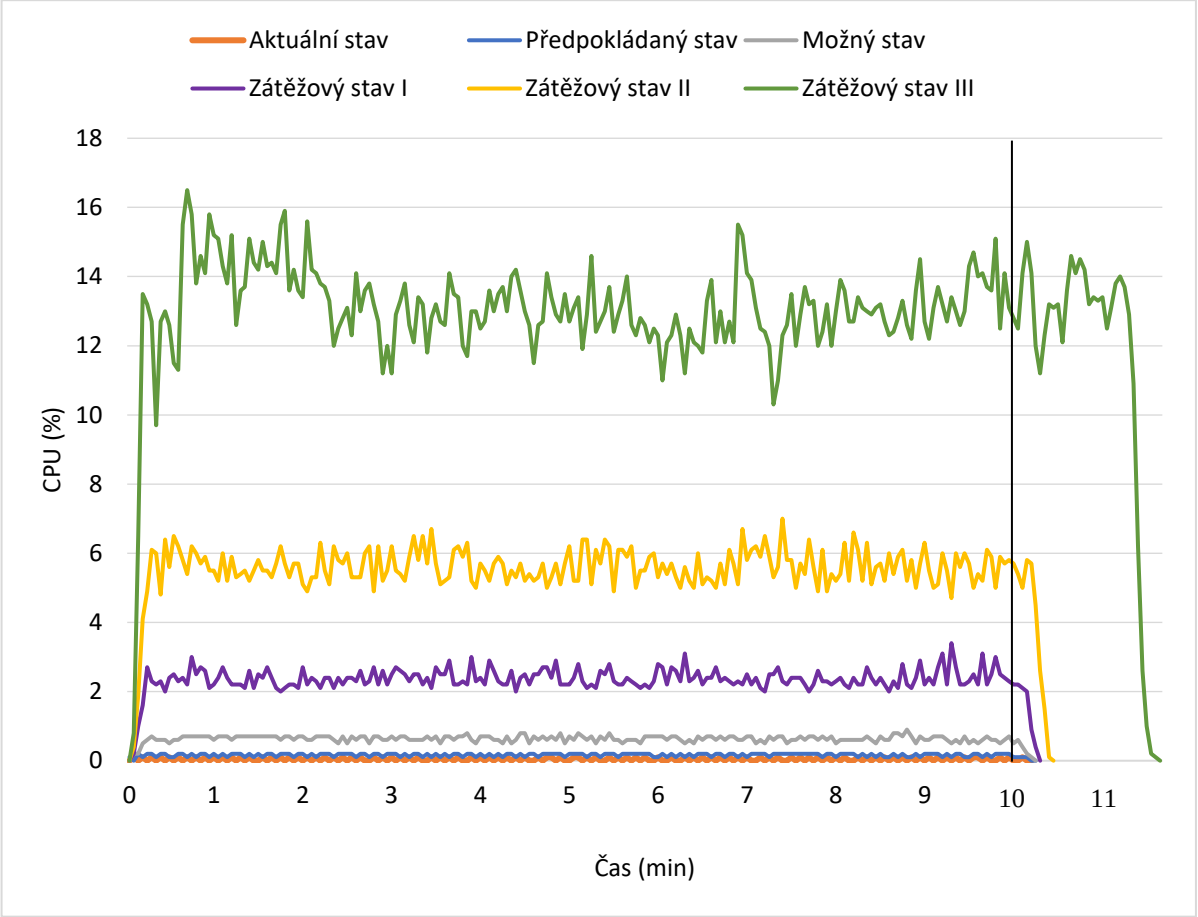
	Celkový čas	Celkový počet zpráv	Max. využití CPU	Zpráv/s	Úspěch zpráv
Výsledky I	710,85 s	1 800 000	16,5 %	2 613,56	100 %
Výsledky II	754,94 s	1 800 000	16 %	2 460,83	100 %
Výsledky III	726,43 s	1 800 000	15,8 %	2 550,30	100 %

Tabulka 6-6: Zadání a výsledky testovacího scénáře "Zátěžový stav III"

6.0.7 Shrnutí

Hlavním důvodem otestování MQTT brokeru pomocí zátěžových testů je posouzení jeho způsobilosti v případě rozšíření systému o další komunikační prvky. Broker je podroben celkovým počtem osmnácti testů, při kterých přijímal definovanou množinu publikovaných zpráv za určitý čas. Toto testování však nesimuluje úplně reálný stav, a to z důvodu absence přeposlání těchto zpráv možným odběratelům. Proto učiněný závěr z tohoto testování nelze považovat za jednoznačný.

Výsledky testování ukazují u scénáře „Aktuální stav“ menší časové zpoždění u celkového času poslaných zpráv, v případě procesorového využití je výstup bezproblémový. Také je potvrzeno, že i při kvalitě servisu (QoS) 0 mají všechny zprávy díky záruce transportní vrstvě (TCP) 100 % úspěšnost přijetí. Kvalita servisu u protokolu MQTT operuje na aplikační vrstvě, kde zaručuje přijetí zpráv od publikujícího klienta k odebírajícímu klientovi. Scénáře „Předpokládaný stav“ společně s „Možný stav“ potvrzují postup lineární posloupnosti u výstupů z testování a vykazují poměrně spolehlivé výsledky. Z těchto výstupů lze tedy usuzovat úspěšné ověření způsobilosti Raspberry Pi v hostování Mosquitto brokeru pro možné rozšíření systému o komunikační prvky například v podobě dalších MQTT klientů. U série zátěžových testů už lze pozorovat výskyt degradace požadovaného výstupu a zároveň navýšení procesorového využití. U scénáře „Zátěžový stav III“ si lze těchto výkyvů všimnout nejvíce. Počet zpráv za sekundu se snížil, tím pádem se hodnota celkového času přijatých zpráv razantně zvýšila. Hodnoty procesorového využití se také prudce zvýšily až na 16 %. Tyto zátěžové scénáře jsou však určeny pouze pro potřeby testování, do těchto stavů by se reálně daný MQTT broker v rámci jedné chytré domácnosti neměl dostat. Pro představu je vizualizován potřebný procesorový čas u procesu Mosquitto který je zaznamenán v době spuštěné první sérii testů viz (Graf 6-1).



Graf 6-1: Výsledný graf zátěže procesu Mosquitto na CPU při spuštěném testování

6.1 Řešené problémy

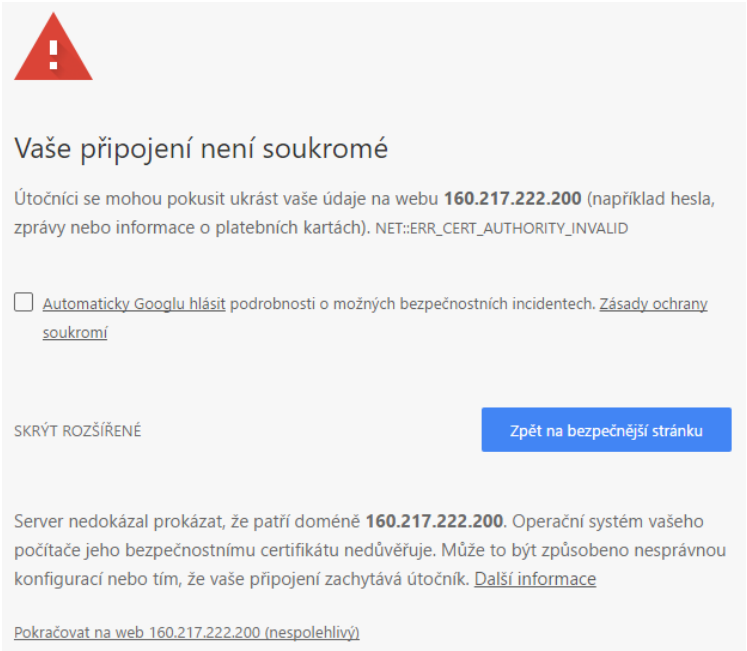
Tato kapitola pojednává o řešených problémech, které nastaly v průběhu vývoje.

6.1.1 Neaktualizovaná dokumentace Mosquitto

Pro zabezpečení Mosquitto brokeru je použita technologie SSL/TLS. Dokumentace Mosquitto je však o zrealizování tohoto zabezpečujícího prvku zastaralá a neobnovená. Proto je použit pro ulehčení postupu dostupný ověřený script, zmíněný v kapitole o transportní úrovni zabezpečení MQTT brokeru viz (5.0.4).

6.1.2 Nedůvěryhodný certifikát

Pro zabezpečení testované komunikace je využito zmiňované technologie SSL/TLS za pomoci vlastního vygenerovaného certifikátu od vlastní vygenerované nedůvěryhodné certifikační autority. Proto je nutno povolit certifikátu výjimku ve webovém prohlížeči viz (Obrázek 6-1). V této situaci musí být ovšem uživatel obezřetný a být poučen o riziku potenciálního nebezpečí.



Obrázek 6-1: Povolení nedůvěryhodného certifikátu

6.1.3 Přístup z internetu

Jak je zmíněno v kapitole viz (5.0.5) přístup z internetu je ovlivněn danou síťovou strukturou, ve které se uživatel nachází. V případě situace autora práce se nepodařilo MQTT broker úplně přesměrovat do internetu. Tento problém lze vyřešit například zakoupení veřejné NAT 1:1 IP adresy od internetového poskytovatele nebo využití VPN služby. Dalším možným řešení této situace je možnost využití tzv. bridge MQTT brokerů. Jde vlastně o přidání dalšího veřejného brokeru do komunikační sítě, který by pak komunikoval s lokálním brokerem bez přístupu do internetu a vytvoří tak jakýsi komunikační most.

6.1.4 Klientské knihovny

Klientské knihovny jsou pro vývoj MQTT systémů nedílnou součástí. Objektivně samotné knihovny splňují svojí funkci, pro které jsou vytvořeny. V případě klientských knihoven pro omezená zařízení se tu vyskytují ale určitá omezení. Nejpoužívanější knihovnou pro tyto zařízení je knihovna Arduino PubSubClient viz (3.1.2). Tato knihovna skvěle vystihuje své zaměření pro omezená zařízení svojí jednoduchostí a lehkou implementací. Ovšem na druhou stranu, díky své jednoduchosti postrádá některé možné žádané funkční prvky jako je například podpora SSL/TLS nebo například podpora QoS úrovně 1 a 2. Proto realizace zašifrované MQTT komunikace mezi omezenými zařízeními je stále velkou výzvou. Aktuálně je tento problém řešený například u Wi-Fi chipu ESP32 nebo ESP8266 pomocí použití spolupráce s novější knihovnou WiFiClientSecure [39].

7 Výsledky

Výsledkem je ekosystém spolu komunikujících zařízení, nad kterým je vyvinuta mobilní aplikace, která umožňuje zabezpečeně ovládat a kontrolovat stav cíleného výkonného prvku. Tento vytvořený ekosystém je reálně využitelný pro jakoukoliv nezávislou domácí automatizaci a je možné ho rozšířit o další chytrá zařízení, která podporují MQTT komunikaci.

MQTT broker v roli centrálního prvku systému poskytuje nepřetržitou dostupnost systému a služeb pro potřeby klientských stran. Realizuje mechanismy, které zabezpečují systém zařízení. Současně je přesměrován z lokální sítě pro potřeby vzdálené komunikace s mobilní aplikací, ovšem není úplně dostupný z internetu, tento problém je rozebrán v kapitole viz (6.1.3). Rovněž je otestován sérií zátěžových testů, ze kterých vyplývá dostatečná rezerva pro potřeby dalšího budoucího rozšíření o další připojené klienty nebo jakákoliv jiná zařízení. Je zapojen do elektrické sítě, a může být potenciálně využit k dalším projektům.

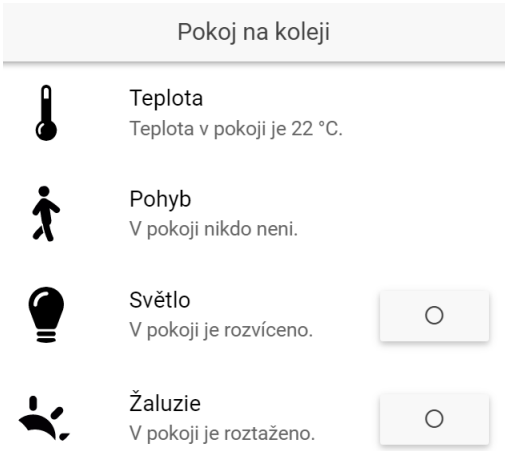
```
pi@raspberrypi:~ $ service mosquitto status
● mosquitto.service - Mosquitto MQTT Broker daemon
   Loaded: loaded (/etc/systemd/system/mosquitto.service; enabled)
   Active: active (running) since Wed 2017-04-05 16:24:53 CEST; 4 days ago
  Process: 559 ExecStart=/usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.c
onf -d (code=exited, status=0/SUCCESS)
 Main PID: 585 (mosquitto)
   CGroup: /system.slice/mosquitto.service
           └─585 /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf -...
```

Obrázek 7-1: Služba Mosquitto jako daemon proces běžící na pozadí

Výkonný prvek realizuje navázání obousměrné lokální komunikaci s MQTT brokerem pomocí MQTT protokolu a Wi-Fi připojení. Slouží k optimalizovanému publikování dat z vybraných senzorů a zpracování příkazů poslané uživatelem mobilní aplikace a převedení těchto dat na data s potenciální reálnou hodnotu – rozsvícení světla, roztažení žaluzií, měření teploty, detekce pohybu.

Nad tímto systémem je vyvinuta uživatelská mobilní aplikace, která dokáže navázat komunikaci s MQTT brokerem. Výsledkem vývoje je hybridní aplikace viz (Obrázek 7-2), která disponuje vlastnostmi responsivního designu a multiplatformního využití, což znamená, že by měla být funkční na většině mobilních zařízení a tabletových počítačích bez dalších větších úprav zdrojového kódu. Mobilní aplikace realizuje obousměrnou a zabezpečenou komunikaci se zabezpečeným MQTT brokerem. Funkce této aplikace interpretují základní

ovládání výkonného prvku a aktuátorů, zároveň vizualizováni senzorických dat vygenerovaná pomocí senzorů.



Obrázek 7-2: Výsledný vzhled mobilní aplikace

7.0 Možné rozšíření

7.0.1 Databáze

Jedním z možných nedostatků MQTT systémů je postrádání vestavěného mechanismu pro možnost využití implementace databáze. Proto pro účely ukládání, analýzy a reportování užitečných dat je realizace databáze jedním z užitečných rozšíření MQTT systému. Tento problém je řešitelný například vytvořením klienta, který by měl vlastnost odebírání Wildcard témat a následně by ukládal požadovaná data do některé z možných forem databáze, tento postup není však optimální. Ideální řešení je navržení externího plugin systému k brokeru, který by obstarával asynchronní připojení s databází.

7.0.2 Rozšíření funkcí

Tato práce se zabývá demonstrováním a vývojem ukázkového principu zabezpečené komunikace mezi zařízeními. Proto připadá v úvahu jakékoli jiné možné rozšíření funkcí systému. Hlavní výhodou MQTT ekosystému je jeho škálovatelnost. Z tohoto důvodu možnost přidání více senzorů a aktuátorů, ale také přidání dalších výkonných prvků v podobě klientů, se nyní stalo velice snadnou záležitostí. Zátěžové testy potvrdily, že rozšíření systému do předpokládaného stavu nečiní žádný omezující problém. Celý tento systém může být ovládán jedinou mobilní aplikací, tudíž i samotný vývoj mobilní aplikace může být nekončícím procesem, jakékoliv další rozšíření funkcí může této aplikaci jen prospět.

Závěr

Tato práce se zabývá problematikou vývoje soběstačného prostředí pro využití v domácí automatizaci. Primárním cílem bylo navržení a popsání principu bezdrátové komunikace sestaveného systému heterogenních zařízení s vyvinutou uživatelskou mobilní aplikací, která by dokázala zabezpečeně kontrolovat a ovládat cílené prvky. Tento systém zařízení byl složen z modulárního výkonného prvku a serveru. Výkonný prvek byl úmyslně realizován s pomocí zařízeních od firmy Texas Instruments. Tato zařízení nejsou běžně dostupná pro širší veřejnost, proto byl koncept otevřeného standardu ověřen právě na těchto zařízeních.

Prvním krokem k úspěšnému naplnění cíle bylo vymezení pojmu Internetu věcí a vyjasnění jeho využití v chytré domácnosti. Kromě toho byla také přiblížena vybraná nastávající úskalí této technologie. Následně byly formou rešerše představeny a rozebrány dostupné IoT bezdrátové technologie, včetně popsání architektur síťových topologií. Rovněž byly porovnány dva zásadní datové protokoly CoAP a MQTT, ze kterého byl vybrán a dále použit protokol MQTT. Z rozboru dostupných a použitelných technologií pro vývoj MQTT systémů byl pro hostování zvoleného Mosquitto brokeru vybrán minipočítač Raspberry Pi. Z lehkého představení praktického vývoje mobilních aplikací byla vybrána realizace hybridní aplikace ve spojení Apache Cordova a frameworku Ionic.

Poznatky z analýzy teoretických východisek a použitelných technologií byly následně využity k návrhu vlastního systému a následnou aplikaci MQTT prostředí. Toto prostředí bylo zrealizováno pomocí konfigurace a zabezpečení Mosquitto brokeru s cílem vytvoření centrálního prvku pro umožnění zabezpečené komunikace výkonného prvku s mobilní aplikací. Spolupráce platformy Raspberry Pi se softwarovým Mosquitto brokerem tvoří ideální řídící jednotku pro nezávislé MQTT prostředí. Rovněž otestování zátěže MQTT brokeru potvrdilo svoji vhodnost pro využití tohoto prvku v běžné inteligentní domácnosti. Realizace klientské strany zahrnovala vyvinutí řídícího programu a sestavení výkonného prvku na bázi zařízeních Texas Instruments spolu se základní množinou senzorů a aktuátorů. Tato zařízení prokázala splnění svého záměru, nicméně pro využití v typické inteligentní domácnosti mají díky své specifičnosti problém obstát konkurenci a jsou spíše vhodná pro účely testování a aplikování ve speciálních případech. Jako druhá klientská strana byla vyvinuta multiplatformní hybridní aplikace interpretující funkční komunikační systém pomocí

ovládání a kontrolování stavu výkonného prvku. Volba vývoje hybridní aplikace se osvědčila jako výhodná cesta pro rychlý a nenáročný vývoj a vystihující účel aplikace.

Mezi hlavní přínos práce patří navržení a popsání postupu možného prostředí chytré domácnosti, ve kterém lze pomocí jedné jednoduché mobilní aplikace bezpečně ovládat jakýkoliv zvolený cílený prvek který podporuje MQTT komunikaci. Uživatel takového prostředí má kontrolu nad celým komunikačním systémem, tím pádem i nad integritou svých citlivých dat. MQTT protokol se osvědčil jako vhodný protokol pro omezená zařízení, avšak realizace SSL/TLS mezi těmito zařízeními zůstává stále výzvou. Osvědčil se také i při realizaci zabezpečené komunikace mobilní aplikace s brokerem spolu s využitím webové technologie WebSockets. Záměrem takového ekosystému je jeho snadná rozšiřitelnost o další výkonné prvky, v podobě MQTT klientů.

Vzhledem k úspěšnému ověření výsledků a otestování centrálního prvku pomocí zátěžových testů s pozitivními výsledky ohledně rozšiřitelnosti prostředí, lze tento vytvořený koncept považovat za funkční a připravený k reálnému využití. Rovněž cíle této práce je tak možné pokládat za splněné. Autor práce byl obohacen o nové vědomosti a porozumění této problematiky, tudíž při potenciální realizaci vlastní chytré domácnosti může být inspirován konceptem vlastní práce.

Seznam použité literatury

[1] Ubiquitous Computing. WEISER, Mark. *Ubiquitous Computing* [online]. 1996 [cit. 2017-03-06]. Dostupné z: http://www.ubiq.com/hypertext/weiser/UbiHome.html

[2] The Information Value Loop. In: *Deloitte University Press: Inside the Internet of Things* [online]. 2015 [cit. 2017-03-06]. Dostupné z: https://dupress.deloitte.com/dup-us-en/focus/internet-of-things/iot-primer-iot-technologies-applications.html

[3] ASHTON, Kevin. That "Internet of Things" Thing: In the real world, things matter more than ideas. *RFID Journal* [online]. 2009, , 1 [cit. 2017-03-06]. Dostupné z: http://www.rfidjournal.com/articles/view?4986

[4] Internet of Things in 2020: A ROAD MAP FOR THE FUTURE. *European Technology Platform on Smart Systems Integration* [online]. 2008, , 32 [cit. 2017-03-06]. Dostupné z: http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf

[5] Inteligentní dům. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Inteligentn%C3%AD_d%C5%AFm

[6] Introducing the MQTT Security Fundamentals. *HIVEMQ* [online]. Germany [cit. 2017-03-06]. Dostupné z: http://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals

[7] REITER, Gil. *Wireless connectivity for the Internet of Things: One size does not fit all* [online]. Dallas (Texas), 2014, , 13 [cit. 2017-03-06]. Dostupné z: http://www.ti.com/lit/wp/swry010/swry010.pdf?DCMP=ep-con-wcs-cmtech&HQS=ep-con-wcs-cmtech-bn-whip-en

[8] *Eclipse IoT: Open Source for IoT* [online]. 2016 [cit. 2017-03-06]. Dostupné z: https://iot.eclipse.org/

[9] Sensor. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-03-06]. Dostupné z: https://en.wikipedia.org/wiki/Sensor

[10] Akční člen. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2017-03-06]. Dostupné z: https://cs.wikipedia.org/wiki/Ak%C4%8Dn%C3%AD_%C4%8Dlen

[11] Jonathan Holdowsky, Monica Mahto, Michael E. Raynor, Mark Cotteleer. Inside the Internet of Things: A primer on the technologies building the IoT. *Deilotte University Press* [online]. 2015 [cit. 2017-03-06]. Dostupné z: https://dupress.deloitte.com/dup-us-en/focus/internet-of-things/iot-primer-iot-technologies-applications.html

[12] Bluetooth® 5 Quadruples Range, Doubles Speed, Increases Data Broadcasting Capacity by 800%. *Bluetooth* [online]. Kirkland (Washington), 2016, , 1 [cit. 2017-03-06]. Dostupné z: https://www.bluetooth.com/news/pressreleases/2016/06/16/-bluetooth5-quadruples-rangedoubles-speedincreases-data-broadcasting-capacity-by-800

[13] Wi-Fi HaLow: Discover Wi-Fi. *Wi-Fi Alliance* [online]. [cit. 2017-03-06]. Dostupné z: http://www.wi-fi.org/discover-wi-fi/wi-fi-halow

[14] MUSHTAQ, Noor Ul. Zigbee in Home Automation: SMART HOMES. *CCTV Institute* [online]. 2016, , 1 [cit. 2017-03-10]. Dostupné z: http://cctvinstitute.co.uk/zigbee/

[15] About Z-Wave Technology. *Z-Wave Alliance: The Internet of Things is powered by Z-Wave.* [online]. Fremont (California) [cit. 2017-03-10]. Dostupné z: http://z-wavealliance.org/about_z-wave_technology/

[16] OLSSON, Jonas. 6LoWPAN demystified. *Texas Instruments* [online]. Dallas, Texas, 2014, , 13 [cit. 2017-03-10]. Dostupné z: http://www.ti.com/lit/wp/swry013/swry013.pdf

[17] THREAD Technology. THREAD [online]. [cit. 2017-03-10]. Dostupné z: https://threadgroup.org/technology/ourtechnology

[18] Z. Shelby, K. Hartke, C. Bormann. The Constrained Application Protocol (CoAP) [online]. 2014 [cit. 2017-03-10]. Dostupné z: https://www.rfc-editor.org/rfc/rfc7252.txt. Universitaet Bremen TZI.

[19] *MQTT Version 3.1.1.* Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[20] MQTT Essentials Special: MQTT over WebSockets. *HIVEMQ* [online]. Germany [cit. 2017-03-06]. Dostupné z: http://www.hivemq.com/blog/mqtt-essentials-special-mqtt-over-websockets

[21] How does TLS affect MQTT performance? *HIVEMQ* [online]. Germany [cit. 2017-04-14]. Dostupné z: http://www.hivemq.com/blog/how-does-tls-affect-mqtt-performance/

[22] Server support: MQTT community wiki. *GitHub* [online]. [cit. 2017-03-06]. Dostupné z: https://github.com/mqtt/mqtt.github.io/wiki

[23] Mosquitto: An Open Source MQTT v3.1/v3.1.1 Broker. *Mosquitto* [online]. [cit. 2017-03-06]. Dostupné z: https://mosquitto.org/

[24] RASPBERRY PI 2 MODEL B. *Raspberry Pi* [online]. [cit. 2017-03-06]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[25] EK-TM4C123GXL Firmware Development Package: USER'S GUIDE. : Texas Instruments [online]. 2016, , 20 [cit. 2017-03-16]. Dostupné z: http://dev.ti.com/tirex/content/TivaWare_C_Series-2.1.3.156/docs/SW-EK-TM4C123GXL-UG-2.1.3.156.pdf

[26] CC3100 SimpleLink™ Wi-Fi® and IoT Solution BoosterPack Hardware: User's Guide. : Texas Instruments [online]. 2014, , 26 [cit. 2017-03-16]. Dostupné z: http://www.ti.com/lit/ug/swru371b/swru371b.pdf

[27] LM35 Precision Centigrade Temperature Sensors. *National Semiconductor* [online]. 1999, , 13 [cit. 2017-04-04]. Dostupné z: http://www.futurlec.com/Linear/LM35DZ.shtml

[28] *HC-SR501 PIR MOTION DETECTOR* [online]. , 3 [cit. 2017-04-04]. Dostupné z: https://www.mpja.com/download/31227sc.pdf

[29] *SG90 9 g Micro Servo* [online]. , 2 [cit. 2017-04-04]. Dostupné z: http://www.micropik.com/PDF/SG90Servo.pdf

[30] *Hybrid Applications And Android Native Browser* [online]. 2014 April, , 1 [cit. 2017-04-04]. Dostupné z: https://myshadesofgray.wordpress.com/2014/04/15/hybrid-applications-and-android-native-browser/

[31] BRISTOWE, John. What is a Hybrid Mobile App? *Telerik Developer Network* [online]. 2015, , 1 [cit. 2017-04-05]. Dostupné z: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/

[32] Apache Cordova: Overview. *Apache Cordova* [online]. [cit. 2017-04-04]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/

[33] Ionic: Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. *Ionic* [online]. [cit. 2017-04-04]. Dostupné z: http://ionicframework.com/

[34] *Arduino Client for MQTT* [online]. [cit. 2017-03-10]. Dostupné z: http://pubsubclient.knolleary.net/

[35] SUTTON, James. MQTT Client Library Encyclopedia – Paho Javascript. HIVEMQ [online]. , 1 [cit. 2017-03-10]. Dostupné z: http://www.hivemq.com/blog/mqtt-client-library-encyclopedia-paho-js

[36] MENS, Jan-Piet. *OwnTracks Tools: generate-CA.sh - Create CA key-pair and server key-pair signed by CA* [online]. In: . 2013 [cit. 2017-04-04]. Dostupné z: https://github.com/owntracks/tools

[37] Energia. *Energia* [online]. [cit. 2017-04-04]. Dostupné z: http://energia.nu/

[38] PALSSON, Karl. *MQTT Malaria: A set of tools to help with testing the scalability and load behaviour of MQTT environments*. [online]. , 1 [cit. 2017-04-08]. Dostupné z: https://github.com/remakeelectric/mqtt-malaria

[39] *Arduino-esp32: WiFiClientSecure* [online]. [cit. 2017-04-15]. Dostupné z: https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFiClientSecure/src/WiFiClientSecure.cpp

Seznam použitých zkratek

6LowPAN	IPv6 over Low-Power Wireless Personal Area Networks
AES	Advanced Encryption Standard
AP	Access Point
API	Application Programming Interface
ARM	Advanced RISC Machine
B	Byte
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
CPU	Control Processing Unit
CSS	Cascading Style Sheets
DHCP	Dynamic Host Configuration Protocol
DTSL	Datagram Transport Layer Security
FSK	Frequency-shift keying
GB	Giga Byte
GPIO	General-purpose input/output
HDMI	High-Definition Multimedia Interface
HTML5	HyperText Markup Language version 5
HTTP	HyperText Transfer Protocol
IANA	Internet Assignes Numbers Authority
IBM	International Business Machines Corporation
IEEE	Institute of Electrical and Electronics Engineers
IFTTT	If This Then That
IoT	Internet of Things
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JS	JavaScript
kB/s	kilo Bytes per second
LAN	Local Area Network
LED	Light-Emitting Diode
LWT	Last Will and Testament
M2M	Machine to Machine

MAC	Media Access Control
MAN	Metropolitan Area Network
MB/s	Mega Bytes per second
MHz	Mega Hertz
MIT	Massachusetts Institute of Technology
MQTT	Message Queuing Telemetry Transport
NAT	Network address translation
OASIS	Organization for the Advancement of Structured Information Standards
OpenGL	Open Graphics Library
PAN	Personal Area Network
QoS	Quality of Service
RAM	Random Access Memory
RFID	Radio Frequency Identification
RGB	Red Green Blue
SD	Secure Digital
SDK	Systém Development Kit
SSH	Secure Shell
SSID	Service Set Identifier
SSL/TLS	Secure Sockets Layer/Transport Layer Security
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
UI	User Interface
USB	Universal Seriál Bus
UX	User Experience
VPN	Virutal Private Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WPA2-PSK	Wi-Fi Protected Access version 2 – Pre-Shared Key

Seznam obrázků

Seznam tabulek a grafů

Tabulky

Grafy

Přílohy

Přiložené CD obsahuje elektronickou verzi práce ve formátu PDF. Dále obsahuje řídící program výkonného prvku. Konfigurace MQTT brokeru je přiložena ve formě konfigurační složky *mosquitto*, ve které se nachází všechny potřebné konfigurační soubory. Zdrojové soubory mobilní aplikace jsou přiloženy ve formě Ionic projektu. Výsledná spustitelná aplikace je přiložena ve formě nepodepsaného aplikačního balíčku pro Android.

V adresáři *Dokument* je umístěna elektronická verze bakalářské práce.
V adresáři *Vykonny_prvek* je umístěn řídící program výkonného prvku.
V adresáři *MQTT_broker* jsou umístěny konfigurační soubory Mosquitto brokeru.
V adresáři *Mobilni_aplikace/Projekt* je umístěn projekt se zdrojovým kódem aplikace.
V adresáři *Mobilni_aplikace/Balicek* je umístěn spustitelný soubor mobilní aplikace.
V adresáři *Zatezove_testy* je umístěn soubor s výsledky zátěžových testů.