Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

OS Google Chrome – teorie a praxe

David Plecháč

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

David Plecháč

Informatika

Název práce

OS Google Chrome – teorie a praxe

Název anglicky

OS Google Chrome – theory and practice

Cíle práce

Diplomová práce je tématicky zaměřena na problematiku aplikací pro operační systém Chrome od společnosti Google. Hlavním cílem práce je charakteristika vývoje aplikací, jejich distribuce a uplatnění na trhu.
Dílčí cíle diplomové práce jsou analýza obecných požadavků na aplikace pro operační systém Google Chrome, charakterizování různých pohledů na využití operačního systému Google Chrome a vyzkoušení některé aplikace v praxi.

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní řešení je realizováno formou návrhu a implementace vlastní jednoduché aplikace pro operační systém Google Chrome. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

Doporučený rozsah práce

60 -80 stran

Klíčová slova

Chromium, OS Google Chrome, Google Dart, Chrome Dev Editor, Google Cloud, Chromium aplikace

---

Doporučené zdroje informací

ANTO, Y. Chrome OS and Secret of Google. Saarbrücken: Lambert Academic Publishing. 2012. 271 pp. ISBN 978-3-659-17127-7.

MILLER, M. My Google Chromebook. Indianapolis, Ind.: Que. 2012. 271 pp. ISBN 07-897-4396-5.

ROOT, G. Cloud Computing with Google Chrome, 1 edition. Seatle:Amazon. 2013. 116 pp. ISBN 978-1483902258.

ŠIKA, M. Virtuální počítač: praktická řešení pro domácí uživatele. Vyd. 1. Brno: Computer Press. 2011. 256 str. ISBN 978-80-251-3334-7.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Čestmír Halbich, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 28. 10. 2015

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11. 11. 2015

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 27. 03. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "OS Google Chrome – teorie a praxe", jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29.3.2017 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu mé diplomové práce panu Ing. Čestmíru Halbichovi, CSc. za jeho podporu a vstřícný přístup. Dále bych rád poděkoval své rodině a přátelům za celoživotní podporu nejen při tvorbě této práce.

Google Chrome OS – teorie a praxe

Souhrn

Má diplomová práce se soustředí na charakterizování klíčových vlastností operačního systému Google Chrome OS a vyhodnocení možností využití tohoto operačního systému v praxi. Teoretická část práce popisuje smysl a filosofii komerčního operačního systému Google Chrome OS a jeho vazbu na prohlížeč Chrome. Vysvětlena je spojitost s open-source projekty Chromium a Chromium OS. Rozebrány jsou detaily vývoje aplikací a možnosti nasazení v organizační struktuře v podobě virtualizace a správy domény. V praktické části práce vyústí teoretické poznatky z vývoje aplikací pro Chrome OS ve vytvoření aplikace tenkého klienta pro vzdálené ovládání hardwarového čidla po internetu. Toto čidlo je navrženo a postaveno na platformě Arduino. Čidlo měří teplotu a vlhkost okolního vzduchu a vzdáleně spíná silnoproudá relé. Pro komunikaci po TCP/IP je použit moderní protokol MQTT, pro který je nakonfigurovaný server s MQTT Brokerem Mosquitto. Dále je demonstrován postup kompilace vlastního buildu operačního systému Chromium OS z veřejně poskytovaných zdrojových kódů společnosti Google. Na základě shrnutí všech nabytých teoretických a praktických poznatků jsem vyhodnotil možnosti nasazení operačního systému Chrome OS v soukromé i komerční oblasti.

Klíčová slova: Google Chrome, Google Chrome OS, Chromium, Chromium OS, IoT, M2M, Chrome management console, Chromebook, Chromebox, Chromecast, Chromebit

Google Chrome OS – theory and practice

Summary

The thesis is focused on the characteristics of the key features of the operating system Google Chrome OS and on the evaluation of deployment of this operating system in practise. The theoretical part describes the purpose and philosophy of the commercial operating system Google Chrome OS and its relation with Chrome web browser. The connection with open-source projects Chromium and Chromium OS is explained. Details of application development and deployment options, in the form of virtualization and domain administration, are analysed. The practical part applies theoretical knowledge from the development of applications for Chrome OS to create thin client application for remote control of hardware sensor over the internet. This sensor is designed and built on the platform Arduino. The sensor measures the temperature and humidity of the ambient air and remote controls high-voltage switching relays. Communication over TCP/IP uses modern MQTT protocol, for which the server is configured with MQTT Broker Mosquitto. The next section demonstrates the procedure for building a customized build of Chromium OS from the publicly provided Google source codes. Based on the evaluation of the acquired theoretical and practical knowledge, I assess the possibility of deploying the Chrome OS in private and commercial areas.

Keywords: Google Chrome, Google Chrome OS, Chromium, Chromium OS, IoT, M2M, Chrome management console, Chromebook, Chromebox, Chromecast, Chromebit

Obsah

1 Úvod **12**

2 Cíl práce a metodika **13**

2.1 Cíl práce 13

2.2 Metodika 13

3 Teoretická východiska **15**

3.1 Google, Inc. 15

3.2 Google Chrome 16

3.3 Chromium 18

3.4 Chrome OS 20

3.5 Hardware pro Chrome OS 21

3.5.1 Chromebook 21

3.5.2 Chromebox 22

3.5.3 Chromecast 23

3.5.4 Chromebit 24

3.6 Chromium OS 24

3.7 Chrome management console 25

3.8 Virtualizace 26

3.9 Chrome OS aplikace 28

3.9.1 Technologie pro vývoj Chrome OS aplikací 28

3.9.2 Architektura Chrome OS aplikace 29

3.9.3 Životní cyklus Chrome OS aplikace 30

3.9.4 Bezpečnostní model Chrome OS aplikace 32

3.9.5 Publikování a monetizace Chrome OS aplikace 33

4 Vlastní práce **35**

4.1 Analýza problematiky 35

4.2 Návrh řešení 36

4.3 MQTT Broker 37

4.3.1 MQTT 37

4.3.2 Instalace serveru a MQTT Brokeru 39

4.4 Hardwarové čidlo 43

4.4.1 Arduino 43

4.4.2 Hardware 44

4.4.2.1 Arduino UNO – mozek zařízení 44

4.4.2.2 Ethernet Shield – připojení k internetu 45

4.4.2.3 DHT22 - měření teploty a vlhkosti vzduchu 46

4.4.2.4 Silnoproudé rélé – spínání elektrických zařízení ... 46
4.4.2.5 LED diody – signalizace spojení s MQTT Brokerem ... 46
4.4.3 Firmware ... 47
4.5 Instalace operačního systému Chromium OS ... 49
4.5.1 Hardware pro kompilaci Chromium OS ... 49
4.5.2 Software pro kompilaci Chromium OS ... 50
4.5.3 Stažení zdrojových kódů Chromium OS ... 51
4.5.4 Kompilace Chromium OS ... 52
4.6 Vytvoření aplikace pro Chrome OS ... 55
4.6.1 Struktura aplikace ... 55
4.6.2 Grafické uživatelské rozhraní (GUI) – Bootstrap ... 56
4.6.3 MQTT komunikace - Eclipse Paho ... 59
4.6.4 Instalace aplikace do Chromium OS ... 61

5 Výsledky a diskuse ... 63
5.1 MQTT Broker ... 63
5.2 Hardwarové čidlo ... 64
5.3 Instalace operačního systému Chromium OS ... 64
5.4 Aplikace pro Chrome OS ... 65

6 Závěr ... 66

7 Seznam použitých zdrojů ... 69

8 Přílohy ... 72

Seznam obrázků

Obrázek č. 1 - Původní domovská stránka vyhledávače Google - LENSSEN, Philipp. Google.com 1997-2011 [online]. In: . Google Blogoscoped, 2006 [cit. 2017-03-20]. Dostupné z: http://blogoscoped.com/archive/2006-04-21-n63.html

Obrázek č. 2 - Původní a nové logo prohlížeče Chrome - DOČEKAL, Daniel. Google Chrome má nové logo. A Chromium také. [online]. 2011 [cit. 2017-03-20]. Dostupné z: http://www.justit.cz/wordpress/2011/03/16/google-chrome-ma-nove-logo-a-chromium-take/

Obrázek č. 3 - Původní a nové logo prohlížeče Chromium - DOČEKAL, Daniel. Google Chrome má nové logo. A Chromium také. [online]. 2011 [cit. 2017-03-20]. Dostupné z: http://www.justit.cz/wordpress/2011/03/16/google-chrome-ma-nove-logo-a-chromium-take/

Obrázek č. 4 - Chrome management console pro správu domény – vlastní zpracování
Obrázek č. 5 - Klient aplikace Teamviewer pro Chrome OS - TeamViewer pro Chrome OS [online]. 2016 [cit. 2017-03-20]. Dostupné z: https://www.teamviewer.com/cs/download/chrome-os/
Obrázek č. 6 - Kontejnerový model Chrome OS aplikací - App container model [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_architecture
Obrázek č. 7 - Ukázka použití události pro inicializaci Storage API - vlastní zpracování
Obrázek č. 8 - Schéma událostí po spuštění Chrome OS aplikace - Chrome App Lifecycle: How the lifecycle works [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_lifecycle
Obrázek č. 10 - Chrome Web Store - Internetový obchod Chrome [online]. [cit. 2017-03-20]. Dostupné z: https://chrome.google.com/webstore/category/extensions?hl=cs
Obrázek č. 11 - Schéma komunikace mezi zařízeními - vlastní zpracování
Obrázek č. 12 - TCP/IP komunikace mezi zařízeními - vlastní zpracování
Obrázek č. 13 - MQTT Broker naslouchá na určených TCP portech - vlastní zpracování
Obrázek č. 14 - Lokální ověření funkčnosti MQTT Brokeru - vlastní zpracování
Obrázek č. 15 - Schéma zapojení HW čidla - vlastní zpracování
Obrázek č. 16 - Arduino UNO rev. 3 - vlastní zpracování
Obrázek č. 17 – Callback() funkce pro spínání relé - vlastní zpracování
Obrázek č. 18 - Funkce reconnect() pro připojení k MQTT Brokeru - vlastní zpracování
Obrázek č. 19 - Cros_sdk chroot - vlastní zpracování
Obrázek č. 20 - Struktura aplikace - vlastní zpracování
Obrázek č. 21 - Událost vytvářející hlavní okno aplikace - vlastní zpracování
Obrázek č. 22 - Formulář připojení k MQTT Brokeru - vlastní zpracování
Obrázek č. 23 - Grafického znázornění naměřených hodnot - vlastní zpracování
Obrázek č. 24 - Grafický ovládací panel pro spínání silnoproudých relé - vlastní zpracování
Obrázek č. 25 - Použití notifikačního API v Chrome OS aplikaci - vlastní zpracování

Obrázek č. 26 – Funkce connect() pro připojení Chrome OS aplikace k MQTT Brokeru - vlastní zpracování
Obrázek č. 27 - Funkce onConnectionLost() je volána při ztrátě připojení k MQTT Brokeru - vlastní zpracování
Obrázek č. 1 - Funkce onMessageArrived() spravuje přijaté zprávy od MQTT Brokeru - vlastní zpracování
Obrázek č. 29 - Funkce přihlašující Chrome OS klienta k odběru vybraných témat z MQTT Brokeru - vlastní zpracování
Obrázek č. 30 - Nainstalovaná aplikace v Chromium OS - vlastní zpracování
Obrázek č. 31 - Ikona aplikace pro rychlé spuštění na hlavní liště Chromium OS - vlastní zpracování
Obrázek č. 32 - Spuštěná Chrome OS aplikace po připojení k MQTT Brokeru - vlastní zpracování
Obrázek č. 33 - Ukázka nezabezpečené komunikace pomocí sledování TCP streamu v programu Wireshark - vlastní zpracování

Seznam tabulek

Tabulka č. 1 - Srovnání Chromebooku s konkurencí - vlastní zpracování
Tabulka č. 2 - srovnání Chromeboxu s konkurencí - vlastní zpracování
Tabulka č. 3 - Srovnání Chromecastu s konkurencí - vlastní zpracování
Tabulka č. 4 - Srovnání Chromebitu s konkurencí - vlastní zpracování
Tabulka č. 5 - Javascriptová API pro Chrome OS - JavaScript APIs: Stable APIs [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/api_index
Tabulka č. 6 - Přehled událostí životního cyklu Chrome OS aplikace - Chrome App Lifecycle: How the lifecycle works [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_lifecycle
Tabulka č. 7 - Životní cyklus Chrome OS aplikace - Chrome Apps Architecture: Programming model [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_architecture
Tabulka č. 8 – Návrh MQTT komunikace - vlastní zpracování
Tabulka č. 9 - Hardwarové prostředky MQTT serveru - vlastní zpracování
Tabulka č. 10 - Rozsahy grafického znázornění měřených hodnot - vlastní zpracování

1 Úvod

Společnost Google není asi třeba příliš představovat. Pravidelně soutěží se společnostmi jako je Apple nebo Samsung o post nejhodnotnější firmy světa. Svůj věhlas firma získala především díky svému stejnojmennému internetovému vyhledávači, který každodenně obslouží přes tři miliardy dotazů a je celosvětově používaný jako hlavní nástroj pro vyhledávání na webu. Jeho obrovský dopad na každodenní život ilustruje už jen ustálení výrazu „googlovat“ ve všech světových jazycích. Nedávný výpadek routingu a DNS serverů společnosti Google v listopadu 2016 způsobil v centrální Evropě propad provozu internetu o 40%. Frustrace uživatelů, kteří si ani nemohli „vygooglovat“, co se děje, byla obrovská. Tvrzení, že Google je vlastně téměř internet jako takový, není daleko od pravdy. Společnost to dostává do monopolní pozice podobné Microsoftu v 80. letech a stejně tak musí řešit právní spory ohledně zneužití svého dominantního postavení.

Z internetového vyhledávače společnost vytvořila vlastní webový prohlížeč Chrome a z prohlížeče dokonce operační systém Chrome OS. Tento operační systém dnes funguje na milionech zařízení po celém světě. Všechny tyto produkty společnosti Google tvoří celou platformu primárně cloudových služeb. Osobně mi přijde sympatické, oproti např. společnosti Apple, že zdrojové kódy platformy Google jsou z velké části otevřené a firma je prezentuje jako open source, což umožnilo vznik tak velkého a ekonomicky úspěšného ekosystému tvořeného aplikacemi, vývojáři a uživateli.

2 Cíl práce a metodika

2.1 Cíl práce

Tato diplomová práce je tematicky zaměřená na operační systém Chrome OS. Cílem práce je charakterizovat tento operační systém a na základě syntézy teoretických poznatků a výsledků vlastního řešení prozkoumat možnosti jeho nasazení.

V teoretické části se zabývám smyslem tohoto operačního systému a jeho klíčovými vlastnostmi. Charakterizovat budu jeho distribuci a vývoj, jeho pozadí otevřené platformy v podobě Chromium OS. V práci popíši fundamenty fungování systému, jeho zabezpečení, uživatelské rozhraní a možnosti jeho správy při administrování více chrome OS zařízení v organizované struktuře. Velkou pozornost věnuji vývoji Chrome OS aplikací.

V praktické části práce předvedu postup zprovoznění open source varianty Chromium OS. Podobný postup používají výrobci hardware určeného pro OEM licencování Chrome OS. Díky tomu mohou dokonale odladit synergii mezi svým hardwarem a operačním systémem Chrome OS. Tento systém simuluje operační systém Chrome OS pro odladění aplikace, která pro něj bude v navazující části vytvořena. Tato aplikace využije teoretických poznatků vývoje aplikací pro Chrome OS a zaměří se na stále více moderní téma, Internet věcí. Tím chci demonstrovat možnosti vývoje aplikací pro chrome OS i pro nově vznikající odvětví IT.

2.2 Metodika

Metodika řešené problematiky této diplomové práce je postavena na analýze a studiu odborných informačních zdrojů. Tato platforma je relativně mladá a neustále se vyvíjí, proto není k dispozici dostatek relevantních knižních zdrojů. Jako primární zdroj informací tedy využívám hlavně bázi znalostí přímo od společnosti Google spolu s omezenou odbornou literaturou.

Vlastní práce se bude krajně zabývat i Internetem věcí, abych ukázal, k čemu všemu lze Chrome OS použít. Práce spočívá v návrhu, sestavení a naprogramování samostatného hardwarového čidla na platformě Arduino. Dále pokračuje zprovozněním serveru pro výměnu zpráv pomocí MQTT komunikačního protokolu. V hlavní části diplomové práce se zaměřím na zkompilování vlastního buildu Chromium OS a naprogramování Chrome

OS aplikace v podobě tenkého klienta pro komunikaci s hardwarovým čidlem skrz internet.

3 Teoretická východiska

3.1 Google, Inc.

Společnost Google, Inc. byla založena v roce 1998 na Stanfordově univerzitě. Její zakladatelé, Larry Page a Sergej Brin, kteří v té době na univerzitě studovali informatiku, měli již během svých dřívějších studií zkušenosti s vývojem prohlížeče BackRub. Název společnosti vznikl ze slova „googol“, jež vyjadřuje jedničku následovanou sto nulami (10^{100}). Podle zakladatelů název reflektuje symboliku v cíli uspořádat prakticky nekonečné množství informací na internetu. [1,2]

Obrázek č. 2 - Původní domovská stránka vyhledávače Google

Zpočátku měla společnost problémy s financováním. Prvním jejím investorem se stal Andy Bechtolsheim, spoluzakladatel společnosti Sun Microsystems. Výše jeho investice byla údajně 100,000 $. To bylo v té době dostatečné množství peněz pro spuštění první verze Google vyhledávače. Jeho relativní úspěch se dostavil poměrně záhy, neboť ho časopis PC Magazine zařadil na žebříček stovky nejlepších webových stránek roku 1998. Tento úspěch a rostoucí popularita webového prohlížeče zapřičinila růst tržní ceny společnosti, nicméně i přesto byla společnost poměrně finančně nestabilní. To dokonce dovedlo její zakladatele k neúspěšnému pokusu o její prodej v roce 1999. Zájemcem byla společnost Excite, která Google odmítla za cenu tři čtvrtě milionu dolarů koupit. Tohoto rozhodnutí musí dodnes litovat. Ještě tentýž rok přišla velká investice ve výši 25 milionů dolarů od kapitálových společností Kleiner Perkins Caufield & Byers a Sequoia Capital. Tato investice společnost stabilizovala a dovolila jí expandovat. Kanceláře společnosti se přesunuly do Palo Alto v Kalifornii, do takzvaného domova technologických start-upů Silicon Valley. V roce 2004 společnost vybudovala svůj vlastní komplex budov, tzv. Googleplex, ten se nachází také v Kalifornii a slouží jako hlavní sídlo společnosti. Kromě

domovských Spojených států společnost zamířila i do ostatních částí světa. Otevřeny byly nové pobočky po Evropě a posléze i v Jižní Americe. Další, pro Google důležitou událostí roku 2004, byl vstup společnosti na burzu. Její akcie se na Wall Street začaly veřejně prodávat za cenu 85 $ za kus v celkové emisi 19,5 milionu akcií. [3]

3.2 Google Chrome

Po úspěchu svého vyhledávače začal Google přidávat další služby, s takto silnou pozicí bylo logickým krokem vytvoření vlastní softwarové platformy. Některé služby byly k tomu účelu zamýšlené od začátku. Jako např. mobilní operační systém Android, s kterým Google ovládá většinu mobilního trhu. Nicméně největší platformou pro společnost Google se stal webový prohlížeč Chrome.

Historie prohlížeče Google Chrome se datuje od roku 2001. Ze začátku bylo vedení společnosti proti vytvoření vlastního prohlížeče, pouze rozhodlo o podpoře již zaběhnutého webového prohlížeče Firefox, patřícímu společnosti Mozilla Foundation. Stále větší touhu společnosti po vlastní platformě ale nemohl cizí prohlížeč uspokojit. Proto v roce 2008 vznikla, pouze pro operační systém Microsoft Windows, první verze webového prohlížeče Google Chrome. Zajímavostí je, že tato první verze byla již zpočátku uvedena ve 43 jazykových mutacích, včetně češtiny. Od této první verze až do současnosti, nezávisle na platformě, je prohlížeč Google Chrome dostupný pod striktní licencí EULA (End-User-Licence-Agreement). Toto licenční omezení dovoluje prohlížeč využívat pouze na uživatelské úrovni, jakékoli změny zdrojového kódu jsou veřejnosti zapovězeny.

První verze Google chrome využívala jako své renderovací jádro open-source řešení WebKit, které využívali i jiné konkurenční prohlížeče (např. Safari). Vykreslovací jádro WebKit bylo po několika vývojových verzích nahrazeno mnohem výkonnějším jádrem Blink. Chrome již od raných verzí podporoval nejnovější technologie jako HTML5, akceleraci efektů CSS3, WebGL, WebAudio a WebRTC. Velmi kladně hodnoceným aspektem je také skriptovací engine Java V8, díky kterému se Chrome umisťuje na předních příčkách benchmarků určených pro měření výkonu webových prohlížečů. Monitorovat údaje o využití dostupného výkonu webovými stránkami, pluginy a rozšířeními lze pomocí integrovaného Správce úloh. Ten poskytuje detailní informace o obsazení dostupné paměti, využití CPU, GPU a dalších prostředků. Každá otevřená

záložka prohlížeče vzniká jako samostatný proces, který je možno pomocí toho správce úloh administrovat. [2,3]

Obrázek č. 3 - Původní a nové logo prohlížeče Chrome

Během svého vývoje získal Chrome spoustu zajímavých funkcí. Zcela klíčovou funkcí je synchronizace, ta umožňuje prohlížeči synchronizovat nastavení, oblíbené záložky a historii na všech zařízeních a systémech, ke kterým je uživatel pomocí svého Google účtu přihlášen. Další funkcí jsou také automatické aktualizace. Prohlížeč celý proces aktualizace provádí v pozadí, aniž by nějak rušil uživatele. Velmi důležitá je funkce „rozšíření“ prohlížeče. Ta umožňuje instalovat do něj programy rozšiřující jeho možnosti. Tato funkce rozšíření je v podstatě ekvivalentem Chrome OS aplikací, s tím rozdílem, že pochopitelně na jiné platformě nemůže využívat Chrome OS API.

Jestliže prohlížeč Chrome exceluje nad konkurenčními prohlížeči ve výkonu, tak z hlediska bezpečnosti má pořád co dohánět. Chrome používá systém dvou černých listin (blacklistů). Jednu pro evidenci stránek s malware, druhou pro podvržené stránky phishingem. Pokud se uživatel pokouší dostat na stránku, která figuruje na blacklistu, je varován. V Chrome je také implementován víceúrovňový bezpečnostní model. Ten se skládá ze dvou úrovní: uživatelské a sandboxové. Každý proces je spuštěný v takzvaném sandboxu, který mu přiděluje pouze procesorový výkon, ale zabraňuje mu ve využívání kritické paměti (paměti operačního systému nebo uživatelských dat) a ostatních procesů (například vzít číslo kreditní karty z jiné záložky). Proces v sandboxu může reagovat pouze na komunikaci vyvolanou uživatelem. Využívanou bezpečnostní funkcí je také bezesporu „anonymní režim“, který prohlížeči brání v ukládání historie prohlížených webových stránek a cookies. Co se týče bezpečnosti rozšíření, tak Chrome využívá „Internetový obchod Chrome“, který je jediným oficiálním distribučním kanálem pro získání rozšíření.

Již od prvních verzí se Google Chrome odlišoval od ostatních prohlížečů svým minimalistickým uživatelským grafickým rozhraním. Zobrazovací plocha například

neobsahuje stavový řádek, obvyklý u konkurenčních prohlížečů. Celé ovládací rozhraní se omezuje na tři pod sebou umístěné lišty. První lišta obsahuje všechny aktuálně otevřené záložky. Druhá lišta slouží k ovládání celého prohlížeče. Obsahuje základní navigačními prvky (zpět, vpřed, refresh), adresní pole a rozhraní pro spouštění nainstalovaných rozšíření a pluginů. Ve druhé liště je také možné vyvolat rolovací menu, umožňující správu a konfiguraci samotného prohlížeče. Poslední, třetí lišta, je volitelná, jedná se o takzvanou „lištu záložek“, do které si uživatel ukládá svoje oblíbené záložky. [7]

Od svého uvedení obliba prohlížeče Chrome mezi uživateli neustále roste. V současnosti je s obrovským náskokem před konkurencí nejpoužívanějším webovým prohlížečem na světě. Na tomto faktu má jistě podíl svázání každého chytrého telefonu s operačním systémem Android s uživatelským Google účtem. Díky širokým možnostem synchronizace mezi všemi službami, které Google nabízí, je sdílení obsahu mezi mobilním telefonem, počítačem a například i automobilem velmi jednoduché. To zcela jistě také zvyšuje oblibu celé platformy. [1,2,3]

3.3 Chromium

Chromium je open-source projekt, ze kterého Google Chrome čerpá svůj zdrojový kód. Projekt podléhá licenci BSD (Berkeley Software Distribution), díky čemuž jsou možnosti jeho využívání téměř neomezené. V rámci této licence je dovoleno zdrojový kód Chromium libovolně upravovat a poté i distribuovat se změněnou licencí. Jedinou podmínkou je uvedení autora, informace o licenci a upozornění na zřeknutí se odpovědnosti za dílo.

Rozdílné oproti Google Chrome je na první pohled modře vybarvené logo Chromium oproti barevnému logu Chrome. První verze Chrome byla v roce 2008 uvedena pouze na platformě Windows, oproti tomu Chromium nabízelo první buildy i pro operační systémy macOS a Linux. Navíc díky dostupnosti zdrojových kódů bylo umožněno uživatelům zkompilovat Chromium s libovolnými modifikacemi.

Obrázek č. 4 - Původní a nové logo prohlížeče Chromium

Dalším rozdílem, stojícím za zmínění, je podpora pluginu Adobe Flash (PPAPI) přímo v Chrome. Tento plugin Google automaticky aktualizuje spolu s Chrome a zároveň je jedinou možností, jak dostat nejnovější verzi Adobe Flash na Linux. Adobe nabízí ke stažení pouze starší verzi NPAPI. Nicméně je možné na Linuxu nainstalovat PPAPI Flash plugin spolu s Chrome a pak jej používat v Chromium.

Poměrně velké rozdíly jsou ale v multimediální podpoře. Chrome obsahuje licencované kodeky pro proprietární formáty mediálního obsahu jako H.264, AAC a MP3, což může být nepříjemnou komplikací při sledování streamovaných videí pomocí technologie HTML5, která tyto kodeky využívá. Oba prohlížeče pak obsahují základní, volně šiřitelné kodeky jako Opus, Vorbis, VP8, VP9, WAV a Theora.

Celkem velkou nevýhodou je funkce automatických aktualizací. Prohlížeč Chrome nabízí automatické aktualizace pro operační systémy Windows a macOS. Linuxoví uživatelé využívají svůj standardní nástroj pro správu software, který pak stahuje aktualizace pro Chrome přímo z Google softwarového repozitáře. Oproti tomu prohlížeč Chromium automatické aktualizace postrádá. Ty tvoří uzavřenou součást Chrome. Uživateli tedy nezbývá, než využívat aktuální buildy Chromium od důvěryhodné třetí strany. Další možností je, aby si uživatel sám kompilovat každou novou verzi ze zdrojových kódů, což je poměrně nepraktické.

Z hlediska bezpečnosti mohou některé Linuxové distribuce defaultně nepovolovat bezpečnostní funkcionalitu Sandbox u prohlížeče Chromium. Chromium pro většinu svých funkcionalit používá stejné Google servery jako Chrome. Jedná se o funkce jako například automatické opravy, doplňování textu, vyhledávaných adres, funkce předvídání, anti-phishing a další. Stejná pro oba prohlížeče je také funkce synchronizace. Díky tomu je možné na Chrome i Chromium používat totožný Google účet a synchronizovat obsah mezi oběma prohlížeči. [1,2,15]

3.4 Chrome OS

Brzy po uvedení prohlížeče Google Chrome začala raketově růst jeho obliba mezi uživateli. Díky širokému portfoliu cloudových služeb, které jsou navíc v základu zdarma, platforma doslova změnila uživatelskou práci s počítačem jako takovou. Většina běžných uživatelů totiž svoji práci s počítačem začala omezovat pouze na zavedení operačního systému a spuštění prohlížeče Chrome. Samotný webový prohlížeč byl schopný poskytnout všechny služby, které uživatelé požadovali. Ať už šlo o emailového klienta, textový či tabulkový editor, nebo konzumaci zábavního obsahu. Všechny tyto poznatky vedly k myšlence jednoduchého operačního systému, který by sloužil pouze jako nutný základ pro jádro Google platformy – webový prohlížeč Chrome. Z této myšlenky vzešel v roce 2010 operační systém Chrome OS. Ten si Google představoval jako ideální operační systém pro běžné i firemní uživatele, tj. systém, který startuje v rámci sekund a využívá naplno cloudových služeb, a to v maximální možné míře. Tento koncept se vrací zpět k myšlence mainframe, kdy veškeré náročné operace probíhají v cloudu. V tomto operačním systému uživatelé nemusí řešit instalaci ovladačů zařízení, firewally, antiviry, ani aktualizace systému. Dokonce ani to, který stroj zrovna používají, protože díky synchronizaci mají „vše všude“. [2,3]

Licenční podmínky Chrome OS jsou obdobné jako u prohlížeče Chrome. Jedná se o komerční operační systém dodávaný výhradně přes OEM kanál spolu se značkovým hardware, který je pro něj určený. Systém je podporován na procesorech typu x86 a ARM.

Hlavní komponenty Chrome OS tvoří linuxové jádro a prohlížeč Chrome. Díky linuxovému jádru (založenému na Gentoo linuxu), je možné spouštět paralelně s Chrome OS také další linuxové distribuce zároveň pod jedním jádrem. Ani jinými parametry se Chrome OS příliš neliší od linuxových distribucí. Obsahuje GNU systémové utility, je kompilován pomocí GCC, grafika systému je postavena na X Serverem.

Grafické uživatelské rozhraní operačního systému Chrome OS je podobné tradičním operačním systémům jako MS Windows nebo OS X. Nabízí tradiční plochu s nastavitelnou tapetou a hlavní lištu s připnutými zástupci aplikací a notifikační oblastí. Prostředí podporuje multi-tasking, a tedy i otevírání více oken, s kterými je možno provádět všechny obvyklé operace včetně maximalizace, přesouvání a podobně.

Důležitou součástí systému je také poměrně jednoduchý správce souborů, který umožňuje přistupovat k paměťovým zařízením jako jsou pevné disky, flash paměti, paměťové karty nebo externí pevné disky. Krom fyzických médií je systémem upřednostňováno cloudové úložiště Google Drive. Podporované jsou všechny hlavní souborové systémy jako FAT, NTFS, EXT2, EXT3, EXT4 a HFS+ pro čtení i zápis. Díky tomu je zajištěna kompatibilita při přenosu souborů ze všech hlavních konkurenčních platforem. Další součást tvoří multimediální přehrávač pro práci s offline médii. Tento přehrávač je velice jednoduchý a nezvládá, krom jiného, ani zobrazovat externí titulky. [5,9]

3.5 Hardware pro Chrome OS

3.5.1 Chromebook

Prvními zařízeními určenými pro Chrome OS se staly laptopy známé jako chromebooky. Úplně prvním referenčním strojem se v roce 2010 stal chromebook CR-48, označený podle izotopu chromu. Ten obsahoval ranou verzi Chrome OS a Google ho poskytl hlavně testerům a recenzentům. O rok později začala retailová výroba. Nejprve vlastní model Samsung chromebook Series 5 uvedla společnost Samsung. Jen o pár měsíců později přišla s vlastním modelem AC 700 i společnost Acer. Trvalo další dva roky, než se v roce 2013 přidaly další velké společnosti - Lenovo s Thinkpadem X131e a Hewlett-Packard s Pavilonem 14. Do tohoto okamžiku se všechny chromebooky vyznačovaly velmi levnou cenou a tomu odpovídajícímu zpracování i použitým materiálům, což dávalo perfektní smysl. Jejich účelem bylo být jednoduchý a levný netbook. Ve stejném roce 2013 ale Google vydal vlastní model chromebooku a nazval ho Google Pixel. Tento model startoval na ceně 1300$, čímž mířil na úplně jinou cílovou skupinu než levné chromebooky. Pixel byl top designově propracovaný, vyrobený z drahých materiálů a nabízel velmi kvalitní display a vysoký výkon. Následovala další vlna nových modelů od zainteresovaných společností a popularita chromebooků strmě rostla. V roce 2013 se vyšplhal podíl chromebooků na PC trhu téměř na úctyhodných 10 %. Tato skutečnost nemohla uniknout pozornosti dalších velkých hráčů na PC trhu. Proto vlastní výrobu zahájily také velké společnosti jako Dell, Toshiba a Asus. Díky narůstajícímu počtu výrobců vzrůstala rozmanitost jednotlivých modelů chromebooků. V roce 2014 již dosáhly takové popularity, že prodejnost v tomto roce překročila 4 miliony kusů. V roce 2016 se

dokonce prodalo více chromebooků než Apple Maců, a to navzdory faktu, že objem PC trhu konstantně klesá. [9,10]

	Acer Chromebook 14 For Work	**Dell Latitude E7470**
CPU	Intel Core i3 6100U Skylake	Intel Core i5 6300U Skylake
GPU	Intel® HD Graphics 520	Intel® HD Graphics 520
RAM	4GB DDR3	8GB DDR4
Display	14“ Antireflexní FullHD IPS	14“ Antireflexní FullHD IPS
Pevný disk	eMMC 32GB	M.2 SSD 128GB
OS	Google Chrome OS	Windows 10 Pro 64bit OEM
Cena (s DPH)	12 990,-Kč	44 590,-Kč

Tabulka č. 1 - Srovnání Chromebooku s konkurencí

3.5.2 Chromebox

Chromebox jednoduše představuje stolní variantu chromebooku. Úplně první Chromebox uvedla společnost Samsung v roce 2012. S vlastním výkonným modelem, podobně jako v případě chromebooku Pixel, přišel Google o rok později. Během dalších let vydaly svoje modely další velké firmy jako Asus, Hewlett-Packard, Acer, Dell a AOPEN. Někteří z výrobců nabízeli i bundlované prodeje. Například společnost Hewlett-Packard přidávala k chromeboxům klávesnici a myš. Společnost Asus dokonce v roce 2014 vydala chromebox rozšířený o takzvaný videokonferenční balíček. Ten obsahoval kamerový modul s vysokým rozlišením 1080p, externí mikrofon s reproduktory a dálkové ovládání. Pořízení tohoto videokonferenčního systému vyšlo na 999 $ plus 250 $ ročně, což je o tisíce dolarů méně než stály videokonferenční systémy od velkých společnostní jako Cisco nebo Polycom.

Hardwarovou specifikací chromeboxy odpovídají chromebookům. Většinou se jedná o takzvané mini počítače formátu mini-ITX, což jen podtrhuje jejich minimalistický design. Tyto stroje většinou nedisponují moc vysokým výkonem. Jsou tedy velice energeticky efektivní a díky ohromně nízké spotřebě elektrické energie nemusí být nijak extra dimenzováno jejich chlazení. Disponují poměrně slušnou konektivitou, například USB3 porty, čtečkou paměťových karet, video porty HDMI a display port, bezdrátovou síťovou kartou standardu 802.11 a/b/g/n/ac. [11]

	Asus Chromebox	Lenovo IdeaCentre 200-01IBW
CPU	Intel Celeron 3215U Broadwell	Intel Celeron 3215U Broadwell
GPU	Intel® HD Graphics	Intel® HD Graphics
RAM	2GB DDR3	2GB DDR3
Pevný disk	16GB SSD	M.2 SSD 32GB
OS	Google Chrome OS	Windows 10 Pro 64bit OEM
Cena (s DPH)	4 990,-Kč	5 219,-Kč

Tabulka č. 2 - srovnání Chromeboxu s konkurencí

3.5.3 Chromecast

Chromecast vznikl v roce 2013 jako takzvané streamovací zařízení. Velikostí téměř odpovídá běžnému flash disku, ale místo USB konektoru je vybaven portem HDMI pro připojení k televizi nebo monitoru. Po připojení k zobrazovacímu zařízení je možno chromecast spárovat s mobilním telefonem nebo počítačem a přenášet na něj veškerý obsah z prohlížeče Chrome, nebo z webových služeb podporující protokol Cast. Podmínkou je, aby obě zařízení byla připojena do stejného síťového subnetu.

Postupně vzniká velké množství aplikací podporující Chromecast. V současnosti jich existuje přes 20,000. Vlastní aplikace nabízí například americké kabelové televize. Chromecast je možno ovládat přes aplikaci pro mobilní telefony s operačními systémy Android a iOS. Druhá možnost je ovládání pomocí prohlížeče Chrome, tato varianta dokáže pracovat s profilem uživatele a jeho preferencemi při používání prohlížeče.

HDMI port Chromecastu podporuje technologii CEC (Consumer Electronics Control). Pomocí CEC může ovládat zařízení, do kterého je chromecast připojen, pokud to zařízení podporuje. O napájení chromecastu se stará microUSB. Síťovou konektivitu zajišťuje Wi-Fi 802.11 b/g/n/ac 2,4/5 Ghz nebo metalický Ethernet přes USB adaptér. V roce 2016 byla vydána nejnovější verze Chromecast Ultra. Tato verze umožňuje žádané přehrávání obsahu v rozlišení 4K Ultra HD a zvuk High dynamic range (HDR10, Dolby Vision). Cena této nejvyšší edice je stanovena na 69 $. [12]

	Google Chromecast 2	Apple TV 2015
CPU	ARM 1,2Ghz	A8 64bit
Pevný disk	eMMC 256MB	eMMC 32GB
OS	Google Chrome OS	tvOS
Cena (s DPH)	1 299,-Kč	4 890,-Kč

Tabulka č. 3 - Srovnání Chromecastu s konkurencí

3.5.4 Chromebit

Chromebit je další zařízení od společnosti Google určené do HDMI portu. Oproti Chromecastu je Chromebit, navzdory velikosti USB klíčenky, plnohodnotný počítač vybavený operačním systémem Chrome OS. Nabízí tedy podobnou funkcionalitu jako Chromebook nebo Chromebox. Chromebit se snaží přímo konkurovat podobnému výrobku společnosti Intel, který nabízí operační systémy Windows a Ubuntu.

Srdcem Chromebitu je procesor Rockchip RK3288-C postavený na architektuře ARM, ten má k dispozici 2 GB operační paměti DDR3. V procesoru je integrovaná grafická karta Mali T764. Pro ukládání dat je možno využít 16GB eMMC. Pro připojení k zobrazovacímu zařízení podporuje rozlišení Full HD. Z hlediska konektivity je Chromebit vybaven jedním USB 2.0 portem, Bluetoothem 4.0, Wi-Fi 802.11ac a napájí se pomocí microUSB. Cena je stanovena na 110$. [13]

	ASUS Chromebit CS10	Intel Compute Stick STK1AW32SC
CPU	RockChip 3288-C	Quad Atom x5-Z8300
GPU	ARM Mali-T764	Intel® HD Graphics
RAM	2GB	2GB
Pevný disk	eMMC 16GB	eMMC 32GB
OS	Google Chrome OS	Windows 10 Home OEM
Cena (s DPH)	2 759,-Kč	3 599,-Kč

Tabulka č. 4 - Srovnání Chromebitu s konkurencí

3.6 Chromium OS

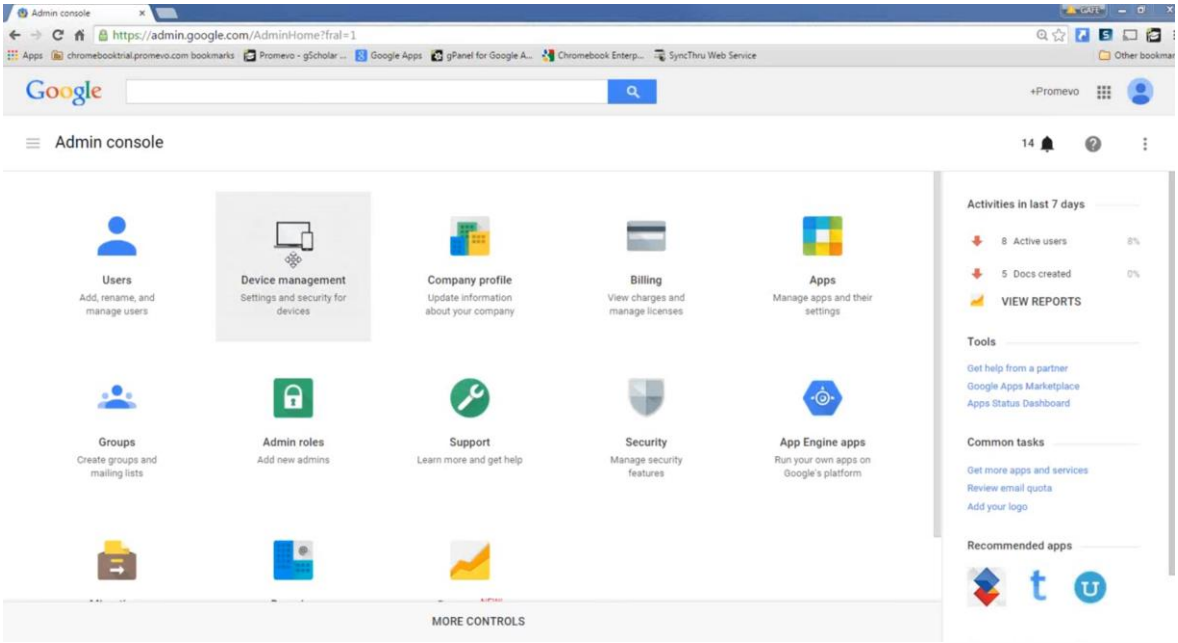
Chromium OS je pro Chrome OS obdobný open-source projekt jako Chromium pro Chrome. Vzhledem k tomu, že příslušný webový prohlížeč je základním komponentem

těchto operačních systémů, tak většina rozdílů mezi nimi je shodná jako mezi webovými prohlížeči. Krom toho, Chromium OS postrádá oproti Chrome OS například funkce verified boot a easy recovery, které vyžadující specifický hardware a jemu na míru upravený firmware.

Chrome OS je distribuován pouze oficiálním OEM kanálem spolu se značkovým hardware. Tudíž jediná legální možnost, jak ho získat, je zakoupit specializovaný Chromebook. Oproti tomu Chromium OS je, stejně jako Chromium, veřejně dostupný a volně šiřitelný. Tudíž je možno ho zkompilovat téměř pro libovolný hardware za cenu možné hardwarové nekompatibility, nemožnosti automatických aktualizací atd. Této skutečnosti bude využito v praktické části práce. [14]

3.7 Chrome management console

Chrome management console je adresářovou službou pro správu vlastní domény založené na zařízeních s operačním systémem Chrome OS. Tato funkcionalita je absolutně nezbytná pro jakoukoli centrálně řízenou organizační strukturu. Ať už se jedná o školu se 100 chromebooky, nebo nadnárodní firmu s 10,000 stroji. Administrace domény probíhá, tak jak by se dalo u Chrome čekat, výhradně v Chrome webovém prohlížeči.



Obrázek č. 5 - Chrome management console pro správu domény

Stručné možnosti správy domény umožňují:

- Vytvářet doménové uživatele a uživatelské skupiny a dále je třídit podle nadefinovaných kritérií. Na tyto uživatele a skupiny je možno uplatňovat doménové politiky, ať už jde o různá nastavení aplikací nebo operačního systému.
- Předinstalovat aplikace nebo jejich instalaci naopak zakázat. Stejně tak lze blokovat instalaci rozšíření do prohlížeče Google Chrome nebo blokovat nepovolená URL.
- Přidělit zařízení specifickým uživatelům a sledovat konfiguraci a informace o používání zařízení.
- Řídit uživatelské přístupy. Určit, kdo a kdy může používat doménové zařízení. Zabránit nedoménovým uživatelům se přihlásit pod google účtem. Zakázat režim hosta.
- Konfigurovat síťová nastavení. Provést za uživatele nastavení pro něj určené, datové sítě, eventuálně nastavení proxy pro zajištění maximální funkcionality a uživatelského pohodlí. Navíc díky tomuto kroku má administrátor jistotu, že je uživatel chráněn správnými firewally a web filtry.
- Modifikovat uživatelská nastavení, jako například oblíbené záložky a volitelná grafická témata (například logo školy)

Správa domény je podobná jiným adresářovým službám, s kterými ale není kompatibilní. Toto platí i obráceně. Tím pádem není možné spravovat pomocí Chrome management console, například počítač s operačním systémem Windows nebo Linux. Stejně tak nelze přidat chromebook do adresářové služby Active directory od společnosti Microsoft. [15,16]

3.8 Virtualizace

Z hlediska korporátního využití je velmi důležitou kapitolou vizualizace, a to z toho důvodu, jelikož není reálné každou aplikaci, kterou potřebujeme, přeprogramovávat pro Chrome OS. Je tedy téměř jisté, že vždy najdeme scénář, kdy uživatelé budou potřebovat přistupovat k nějaké službě běžící na Windows nebo Linux serveru. Typickým příkladem

může být přístup k terminálovému serveru nebo například vzdálený přístup do zabezpečeného serveru, umožňující ovládání průmyslových zařízení.

Moderním trendem je upouštění od klasických appliance řešení pro každou jednotlivou službu, kterou je potřeba nasadit. Místo toho se jednotlivé služby virtualizují na menším počtu fyzických strojů. Díky tomu můžou být na jednom fyzickém stroji nasazeny řádově desítky strojů virtuálních. Tyto „virtualizační nody“ mohou být, pro dosažení lepší dostupnosti soustředěny do takzvaných fail-over clusterů. V případě výpadku některého z fyzických strojů dokážou virtuální stroje přemigrovat na jiný virtualizační node v reálném čase a dál fungovat bez výpadku.

Vzrůstajícímu vlivu Chrome OS nemohou vzdorovat různé virtualizační platformy. Velké společnosti jako Citrix A VMware začaly podporovat tento operační systém už v roce 2015. Společnost Citrix nabízí pro Chrome OS produkt Citrix’s HTML 5 Receiver a společnost Vmware produkt VMware Horizon View 6. Oba produkty umožňují přistupovat k backendu hostované virtuální infrastruktury přes webový prohlížeč Chrome. Z hlediska funkcionality umožňují plně přistupovat k systémovým zdrojům virtualizovaných strojů. To umožňuje vzdálené využívání tiskových úloh, připojených periferií, externích portů, audia a videa. Tato řešení obsahují nevýhodu v náročnosti na hardware. Jelikož digitální kódování a dekódování videa a audia vytváří značnou zátěž na procesor využívaného serveru, je vhodné server osadit grafickou kartou s podporou VDI nebo procesorem s integrovanou grafickou kartou s touto podporou. Díky tomu se zátěž procesoru sníží a server zvládne naráz obsloužit mnohem více klientů. I přes podporu ostatních společností, ten největší hráč na trhu, Microsoft, Chrome OS vytrvale ignoruje. Jeho virtualizační platforma Hyper-V operační systém od Google nepodporuje a pravděpodobně ani podporovat nebude. Microsoft dokonce ani neuvolnil vlastního RDP klienta pro vzdálený přístup pomocí tohoto protokolu. Sice existují neoficiální varianty, ty ale obsahují omezené funkcionality a nefungují zcela spolehlivě. Nicméně i nepodpora Chrome OS na virtualizační platformě Hyper-V se dá řešit, a to pomocí nástrojů pro vzdálenou správu třetí strany. Jako například stejnojmenným produktem společnosti TeamViewer. Ta poskytuje jak serverovou část pro libovolný serverový operační systém, tak i klientskou aplikaci pro Chrome OS. [4,17]

Obrázek č. 6 - Klient aplikace Teamviewer pro Chrome OS

3.9 Chrome OS aplikace

3.9.1 Technologie pro vývoj Chrome OS aplikací

Hlavní technologie pro vývoj Chrome OS aplikací jsou HTML5, CSS a Javascript. Tyto technologie jsou dnes standardem pro vývoj webových aplikací a již dosahují takové úrovně, kdy je, s jejich pomocí, možno vyvinout téměř libovolnou aplikaci. Díky tomu je možné poměrně lehce konvertovat webové aplikace na Chrome OS aplikaci. Nicméně, musí se počítat s tím, že každá Chrome OS aplikace by měla, oproti webovým aplikacím, poskytovat alespoň minimální offline funkcionalitu. Velkou výhodou je také možnost použití frameworků rozšiřujících tyto webové technologie přímo pro Chrome OS aplikaci. Samozřejmě v případě použití frameworků je nutné, aby splňovaly kritéria pro Chrome OS aplikaci, hlavně vyžadovaný bezpečnostní model.

Kromě standardních webových technologií nabízí Chrome OS řadu speciálně účelových Javascriptových API. Sady těchto API se neustále rozšiřují a dělí se převážně na stabilní, vývojářské a experimentální. [3,4,18]

Název	Popis	Od verze
alarms	Umožnuje spustit kód periodicky nebo určitý čas	22
app.window	Umožnuje vytvářet aplikační okna	23
bluetooth	Umožnuje připojení k bluetooth zařízení	37
browser	Umožňuje interakci s webovým prohlížečem Chrome	42
fileSystem	Zajišťuje přístup k lokálnímu filesystemu uživatele	23

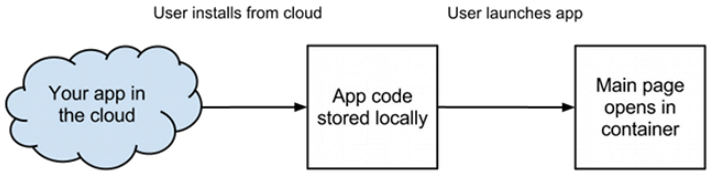
notifications	Umožňuje zobrazovat notifikace	28
serial	Obsluhuje sériový port	23
permissions	Slouží k udělování přístupových oprávnění za běhu aplikace	16
usb	Slouží k interakci se zařízeními připojenými pomocí USB	26
vpnProvider	Implementuje VPN klienta	43
wallpaper	Dovoluje pozměnit Chrome OS pozadí plochy	43
tts	Přehrává sysntetizovaný text-to-speech	15
Sockets.tcp	Přijímá a odesílá data po síti pomocí protokolu TCP	33
Sockets.udp	Přijímá a odesílá data po síti pomocí protokolu TCP	33
power	Umožňuje obluhovat řízení spotřeby systému	27

Tabulka č. 5 - Javascriptová API pro Chrome OS

3.9.2 Architektura Chrome OS aplikace

Oproti obyčejné webové aplikaci v prohlížeči pracuje Chrome OS aplikace blíže k operačnímu systému, díky tomu může být její funkcionalita mnohem větší a robustnější než v případě webu. Aplikace je v Chrome OS uzavřena do takzvaného aplikačního kontejneru. Tento kontejner má na nejnižší úrovni podobu prosté prázdné čtvercové oblasti. Postrádá jakékoli uživatelské ovládací prvky typické pro webový prohlížeč proto, aby uživatel nemohl zasahovat do logiky aplikace, například změnou URL adresy.

Chrome aplikace se načítají jinak než webové aplikace. Obojí sice načítá stejný typ obsahu (HTML dokumenty s CSS a Javascriptem). Nicméně Chrome aplikace je nahrána do aplikačního kontejneru, nikoli do záložky webového prohlížeče. K tomu navíc musí být Chrome aplikace nahrána do aplikačního kontejneru z lokálního zdroje. Toto klade nárok na alespoň minimální offline funkcionalitu aplikace a dovoluje vynucování striktnějších bezpečnostních nároků na aplikaci. [2,3]



Obrázek č. 7 - Kontejnerový model Chrome OS aplikací

Každá Chrome aplikace obsahuje tři základní komponenty. Prvním z nich je soubor manifest.json. Tento soubor je strukturovaný ve formátu JSON a poskytuje operačnímu systému důležité informace o aplikaci. Mezi tyto informace patří například název aplikace, její verze, jazykové lokalizace a minimální požadovaná verze operačního systému. Poskytuje také velmi důležité informace o oprávněních aplikace, které jí uživatel musí schválit při instalaci. Tato oprávnění aplikaci dovolují například přístup k lokálnímu úložišti, a to jak v módu čtení, tak zapisování, oprávnění pro přístup aplikace k hardware jako je bluetooth, USB a dalším periferiím. Soubor manifest také umožňuje ukládání kryptografických informací, jako jsou šifrovací klíče nebo například navazování síťových socketů protokolů TCP a UDP. Velice důležitou a povinnou součástí manifestu je takzvaná „event page“. Tato sekce musí nutně obsahovat výčet veškerých skriptů, které aplikace využívá.

Další komponentou aplikace je povinný skript background.js. Jak jsem již zmínil, název tohoto skriptu musí event page také obsahovat. V těle skriptu musí být definována událost onLaunched(). Ta určuje, co přesně aplikace provede po jejím spuštění.

Poslední komponentou Chrome aplikace jsou assety využívané aplikací, jako ikony, obrázky a skripty. [4,19]

3.9.3 Životní cyklus Chrome OS aplikace

Každá z fází životního cyklu Chrome aplikace je svázána event listenerem, který zpracovává nastalou událost a volá naprogramované metody tak, aby na každou událost bylo možno pružně reagovat. Tento systém je velmi podobný programovacímu jazyku Java, potažmo Javascript.

Akce	**událost**
Instalace a aktualizace aplikace	onInstalled()
Restart aplikace	onRestarted()
Spuštění aplikace	onLaunched()
Ukončení aplikace	onSuspend()
Přerušení ukončení aplikace	onSuspendCanceled()

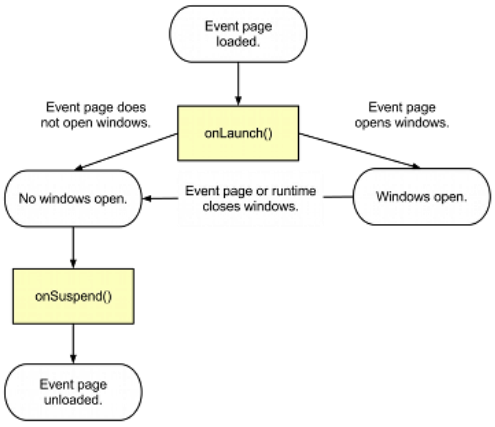
Tabulka č. 6 - Přehled událostí životního cyklu Chrome OS aplikace

Při instalaci a aktualizaci Chrome aplikace je volána událost onInstalled(), například pro nastavení lokálního úložiště pomocí Storage API.

```
chrome.runtime.onInstalled.addListener(function() {
  chrome.storage.local.set(object items, function callback);
});
```

Obrázek č. 8 - Ukázka použití události pro inicializaci Storage API

Po spuštění Chrome OS aplikace se, ze všeho nejdříve, načítá lokálně uložená event page této aplikace. Zároveň s event page se spouští i událost onLaunch(). Tato událost předává event page informace o tom, jaká okna bude aplikace otevírat a současně i jejich rozměry. Stejně tak vykonává další akce, které jsou nadefinovány provést po startu aplikace.



Obrázek č. 9 - Schéma událostí po spuštění Chrome OS aplikace

Pokud event page již neprovádí žádný javascriptový kód, nemá čekající callbacky ani žádná otevřená okna, tak se aplikace ukončí. Ještě před ukončením aplikace se zavolá událost onSuspend(). Skrz tu je možné řešit „úklidové“ úkony před finálním zavřením aplikace. Pokud je toto ukončování aplikace z nějakého důvodu přerušeno, vyvolá se událost onSuspendCanceled(). [4,20]

Stav aplikace	Popis stavu
Instalace aplikace	Uživatel zvolí instalaci aplikace a explicitně jí povolí oprávnění
Spuštění aplikace	Načte se event page, spustí se naskriptované události a okna aplikací využívaná

Ukončení aplikace	Aplikace může být kdykoli ukončena a také rychle obnovena do předchozího stavu. Chrome API obsahuje metody pro uložení a obnovení dat
Aktualizace aplikace	Aplikace může být kdykoli aktualizována, nicméně nelze měnit kód za běhu aplikace
Odinstalace aplikace	Uživatel může aktivně odinstalovat aplikaci. Po odinstalaci nezůstane v počítači žádný kód nebo data

Tabulka č. 7 - Životní cyklus Chrome OS aplikace

3.9.4 Bezpečnostní model Chrome OS aplikace

Bezpečnostní model, na němž stojí Chrome aplikace, poskytuje uživatelům poměrně velkou jistotu v bezpečném a zabezpečeném nakládání s jejich informacemi a daty. Načítání aplikace z lokálního úložiště vynucuje striktnější zabezpečení než načítání webové aplikace ze vzdáleného serveru. Při instalaci vyžaduje Chrome aplikace explicitní povolení, oprávnění pro přístup a použití uživatelských dat. Každé API, které Chrome OS nabízí, ať už pro přístup k lokálnímu úložišti, periferiím atd., vyžaduje po uživateli udělení oprávnění zvlášť.

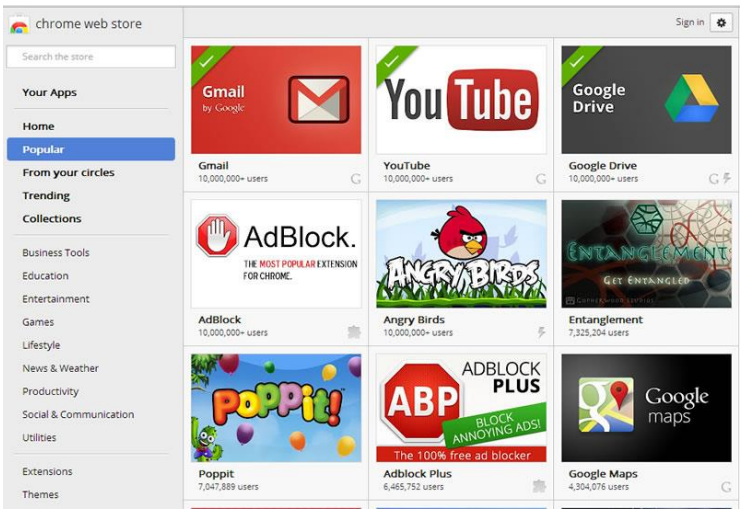
Chrome OS používá pro aplikace systém isolace procesů, kdy jeden proces nemůže zasahovat do fungování procesu jiného. K tomu je navíc izolováno i úložiště a externí obsah pro jednotlivé aplikace. Díky tomu má každá aplikace svůj vlastní privátní prostor pro ukládání dat a nemůže přistupovat k prostoru jiné aplikace. Dokonce i všechny externí procesy jsou izolovány mimo aplikaci. Například pro zakomponování externího embedovaného obsahu v rámci aplikace je třeba použít speciální tag, který tento externí obsah spustí v procesu separovaném od aplikace.

Dalším zásadním prvkem bezpečnosti Chrome aplikace je nutnost dodržování Content Security Policy (CSP) při programování aplikace. Aplikaci nedodržující tuto politiku není možno do Chrome OS nainstalovat. Hlavním smyslem této poměrně striktní politiky je implicitní ochrana před potencionálními cross-site scripting (XSS) útoky na aplikaci. CSP vpodstatě představuje zapovězení používání konkrétních mechanismů a postupů při programování Chrome aplikace. Mezi tyto nebezpečné mechanismy patří hlavně načítání a spouštění různých zdrojů. Při programování je tedy nutné přistupovat k některým fundamentálním postupům jinak než v případě programování standardní

webové aplikace. Mezi zapovězené postupy patří hlavně používání inline scriptingu, například klasické použití event handlerů (<button onlick="#">). Dále odkazování se, v kódu aplikace, na externí zdroje (kromě audio a video zdrojů), takže například zakomponování externího obsahu do iframe. Také je zakázáno používání nebezpečných metod typu string-to-JavaScript, jako například eval() a new Function(). Tyto funkce umožňují běžný řetězec přeměnit na spustitelný kód. [21]

3.9.5 Publikování a monetizace Chrome OS aplikace

Pro publikaci aplikací vyvinula společnost Google vlastní internetový obchod a nazvala jej Chrome Web Store. Tento internetový obchod je podobný obchodu Google Play, který společnost provozuje pro operační systém Android.



Obrázek č. 10 - Chrome Web Store

Prvním krokem pro publikování vytvořené Chrome aplikace je vytvoření ZIP souboru s celou strukturou aplikace popsanou v předchozích kapitolách. Před nahráním aplikace je třeba zadat svůj vývojářský účet. Tento účet bude spravovat publikované aplikace a také řešit platby.

Pokud publikovaná aplikace využívá externích zdrojů, je potřeba dokázat vlastnictví odkazovaných URL. Vývojářský účet nabízí nástroj k dokazování vlastnictví těchto URL. Stejně tak obsahuje i nástroj pro převod vlastnictví aplikace na jiného vlastníka. Pokud již existuje verze publikované aplikace pro operační systém Android a je publikována na Google Play, je možno ji provázat s Chrome Web Store. Poté se u profilu aplikace objeví

odkaz „Available for Android“. Podmínkami pro tuto provázanost mezi obchody je identický název aplikací a vlastnictví obou aplikací stejným vývojářským účtem.

Po přihlášení k uživatelskému účtu je možno v jednoduchém rozhraní nahrát zabalenou Chrome aplikaci v ZIP souboru. Pokud aplikace splňuje všechny požadované náležitosti, tak bude úspěšně nahrána. Nově nahrané aplikaci bude přiděleno app ID, což je její jednoznačný identifikátor. [1,3]

Po úspěšném nahrání aplikace do Chrome Web Store si můžeme zvolit systém plateb. Pokud chceme aplikaci distribuovat zcela zdarma, je možno tento krok přeskočit. Pokud budeme chtít aplikaci zpoplatnit, musíme registrovat vývojářský účet vlastnící aplikaci jako obchodní účet zvaný Google Checkout merchant. Google si nárokuje 5 % z ceny každé uskutečněné transakce. Ceny mohou být různé pro každý geografický region. Pomocí obchodního účtu je možné zvolit následující varianty monetizace:

- **Mikrotransakce** – například počítačová hra distribuovaná zdarma. Zpoplatněny mohou být virtuální herní předměty.
- **Jednorázová platba** – například aplikace, jejíž stažení je podmíněno provedením jednorázové platby.
- **Předplatné** – Chrome Web Store umožňuje využít funkci jak měsíčního, tak ročního předplatného.
- **Trial verze** – dočasná trial verze, která pro plnou funkcionalitu vyžaduje provedení jednorázové platby.

Po vybrání systému monetizace pro Chrome aplikace je třeba doplnit její profil tak, jak bude zobrazována v internetovém obchodě. Důležité položky profilu každé aplikace jsou: [23]

- **Detailní popis aplikace** – popsat aplikaci tak, aby byla po uživatele zajímavá
- **Ikona pro Chrome Web Store** – ikona zobrazovaná v obchodě v rozlišení 128x128 pixelů, může být stejná jako použitá ikona aplikace
- **Screenshoty aplikace** – alespoň jeden obrázek v rozlišení 1280x800 nebo 640x400 pixelů znázorňující, co publikovaná aplikace dělá (může být použito i YouTube video)
- **Dlažlicová ikona** – malá ikona v rozlišení 440x280 - bude použita na zdi nabídky Chrome Web Store

- **Primární kategorie** – aplikaci je třeba zatřídit do příslušné kategorie podle jejího primárního využití
- **Jazyk** – dostupné jazykové mutace aplikace

Před finálním publikováním aplikace je třeba zaplatit takzvaný vývojářský poplatek ve výši 5 $. Tento poplatek je jednorázový a platí se pouze při publikování první aplikace na daném vývojářském účtu.

Nejrozumnější forma prvního publikování aplikace je publikovat ji testovacím účtům. To znamená, že aplikace nebude dostupná ve vyhledávači v Chrome Web Store. Dostupná bude pomocí přímého odkazu na aplikaci, a to pouze specifikovaným uživatelským účtům. Tato funkcionalita dává možnost využití testerů pro důkladné otestování před ostrým nasazením aplikace.

Po důkladném otestování je možné aplikaci publikovat světu. To znamená, že bude veřejně viditelná ve vyhledávači aplikací v Chrome Web Store.

Další možností pro publikování aplikace je takzvané skupinové publikování. To umožňuje sdílet položky v Chrome Web Store s jinými vývojáři. Tato skupina vývojářů pak tvoří takzvanou Google Group a může přistupovat a měnit položky, které vlastník aplikace povolí.

Po veřejném publikování Google nabízí nástroje analýzy využívání aplikace. Tím nejdůležitějším z hlediska vývoje aplikací je nástroj pro distribuci updatů. Pomocí tohoto nástroje můžeme nastavit procento uživatelů aplikace, které bude aktualizováno na její nejnovější verzi. Můžeme tím předejít šíření nové verze aplikace obsahující nějakou závažnější chybu. Pokud tedy aplikace bude mít 20 000 uživatelů verze 1.0, můžeme novou verzi 2.0 rozšířit pouze mezi například 10 % z nich. Pokud bude výsledných 2 000 uživatelů s novou verzí spokojeno, rozšíříme ji i mezi zbytek. [22]

4 Vlastní práce

4.1 Analýza problematiky

Praktická část této diplomové práce je zaměřená na reálné využití operačního systému Chrome OS. Jak bylo nastíněno v teoretické části, jediný legální způsob, jak získat Chrome OS, je jeho zakoupení v podobě OEM licence spolu s Chromebookem. To mi

vzhledem k studijnímu charakteru této práce nepřišlo nijak zajímavé, proto jsem se rozhodl pro zprovoznění open-source varianty Chromium OS. Tento systém bude zcela vyhovující pro vytvoření a otestování aplikace tak, jak by běžela i na Chrome OS.

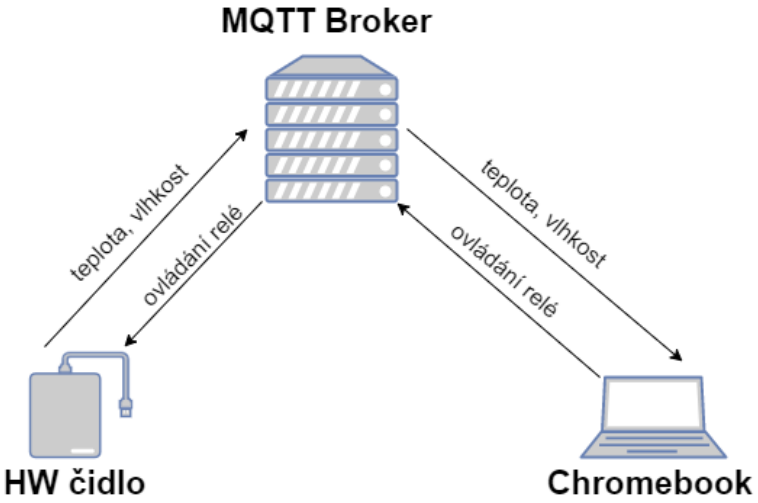
Jelikož je tento systém silně orientovaný na internetové služby, tak samotnou aplikaci pro Chrome OS chci zaměřit na velmi moderní trend Internetu věcí, kde se všemožná zařízení propojují s internetem. Mým cílem je vytvořit hardwarové zařízení neboli čidlo, které bude možné ovládat přes internet pomocí aplikace pro Chrome OS. Aplikace bude z čidla přijímat data v podobě teploty a vlhkosti naměřené v jeho okolí. Zároveň bude aplikace data odesílat, neboť bude ovládat relé pro spínání silnoproudých elektrických zařízení.

4.2 Návrh řešení

Pro realizaci praktické části jsem navrhl následující postup. Celý návrh je vzhledem ke své komplexnosti velmi zjednodušený. Případná úskalí nebo vylepšení se budu snažit naznačit ve výsledcích.

Pro emulaci Chromebooku, tedy počítače vybaveného Chrome OS, uvedu postup instalace Chromium OS pro otestování vyvíjené aplikace na standardním notebooku. Pro tento systém vytvořím aplikaci pro přijímání a odesílání dat z čidla. Aplikace bude mít grafické rozhraní vykreslující hodnoty naměřené čidlem a také bude disponovat ovládacími prvky pro spínání elektrických zařízení připojených k čidlu. Aplikaci vytvořím pomocí technologií HTML, CSS a Javascript.

Jako komunikační protokol využiji standard MQTT. Je to jednoduchý a nenáročný protokol pro předávání zpráv mezi klienty. Oproti například protokolu HTTP nabízí řadu výhod. Ta nejpatrnější je, že nemusíme vytvářet extra serverovou aplikaci pro komunikaci mezi klienty. To obstará centrální bod, takzvaný MQTT Broker, který se stará o výměnu zpráv. Tento server s MQTT Brokerem bude mít přidělenu veřejnou IP adresu tak, aby byl dostupný kdekoli na světě. Díky tomu budeme moci z našeho Chromebooku přijímat a odesílat data do čidla, a to i když obě zařízení budou připojena do internetu každé na jiném konci světa.



Obrázek č. 11 - Schéma komunikace mezi zařízeními

Hardwarové čidlo bude realizováno pomocí platformy Arduino využívající 8bitový mikrokontroler AVR. Bude použita síťová karta pro komunikaci po TCP/IP, čidlo k měření teploty a vlhkosti, relé pro spínání elektrických zařízení a LED diody pro signalizaci připojení k MQTT Brokeru. Firmware pro čidlo naprogramuji v jazyce C a bude využívat veřejně dostupných knihoven. Hlavní funkce čidla spočívá v odesílání naměřených dat teplot a vlhkosti a v přijímání signálů pro sepnutí příslušného relé.

4.3 MQTT Broker

4.3.1 MQTT

MQTT (Message queuing telemetry transport) byl navržen v roce 1999 v IBM, dnes za ním stojí Eclipse foundation a před nedávnem prošel standardizací OASIS a stal se tak standardem ISO / IEC 20922. Protokol je to nenáročný, s nízkým zatížením sítě, orientovaný především na rychlost nebo na nespolehlivé připojení. Zároveň se snaží o zachování spolehlivosti a určité úrovně jistoty doručení zpráv. Díky těmto vlastnostem je protokol vhodný pro myšlenku „machine-to-machine“ (M2M) nebo „Internet of Things“ (IoT), tedy propojení velkého množství „chytrých“ zařízení mezi sebou, potažmo do internetu. Stejně tak je vhodný pro mobilní aplikace, kde je vhodné co nejmenší využití datových přenosů a nízké zatěžování baterie.

Protokol MQTT používá návrhový vzor publisher – subscriber a spojení probíhá pomocí TCP. V IoT systémech by snímače neměly určovat, kdo má co dělat. Jejich úkolem je pouze posílat data do centrálního místa, v našem případě takzvaného MQTT Brokeru, který se stará o výměnu zpráv. Zprávy se třídí do takzvaných témat (topiců). O víc by se snímače starat neměly, tomuto odesílání dat Brokeru se říká publikování. Broker poté, co tato data obdrží, je ukládá a publikuje dalším zařízením, která jsou přihlášena k jejich odběru (subscribe). Jedno zařízení samozřejmě může být najednou v některých tématech publisher a v jiných subscriber podle potřeby.

Samotný obsah zpráv není nijak daný ani pevně vyžadovaný. Jsou to prostě binární data, která protokol přepraví. Nejčastěji se používají formáty JSON, BSON, ale může to být opravdu cokoliv. Velikost zprávy je aktuálně omezena na 256 MB. Nicméně naprostá většina zpráv je mnohem menší, v našem případě v řádu kilobytů. Jak jsem již uvedl, MQTT je protokol nenáročný, takže přidává jen minimum servisních dat. Umožňuje tři úrovně QoS (Quality of Servise), tedy potvrzování doručení zpráv. V našem případě použijeme nejnižší úroveň 0, ta znamená, že zpráva je odeslána bez potvrzení a není zaručeno její doručení. Prostřední úroveň 1 znamená doručení alespoň jednou, a nejvyšší úroveň QoS 2 znamená, že každá zpráva je doručena právě jednou.

Z hlediska bezpečnosti protokol umožňuje jednoduchou autentizaci klienta oproti Brokeru pomocí jména a hesla, i tak ale přenáší data pomocí TCP nativně v plaintextu. Dají se tudíž jednoduše odposlechnout nebo pozměnit. Pro úplně bezpečnou komunikaci je však potřeba využít šifrování TSL. To si samozřejmě vybere svoji daň ve větších nárocích na výpočetní výkon. U Brokeru to zase takový problém není, nicméně u koncových zařízení, která zpravidla nejsou navržena na složité výpočetní úlohy, to problém může být. Pro jednoduchost a názornost šifrování nebude v této práci použito.

Jak již bylo řečeno, zprávy v MQTT patří do specifických témat (topiců). Každá zpráva spadá právě do jednoho tématu. Tato témata jsou řetězce kódované v UTF-8, takže mohou obsahovat i diakritiku. Témata jsou hierarchicky strukturovaná, jako oddělovací znak slouží lomítko. Tato hierarchie není nijak pevně dána, záleží na návrhu aplikace. Témata se nemusí nijak zakládat ani kontrolovat. Jakmile Broker přijme zprávu od publishera pro téma, které ještě nemá, tak jej založí. Subscriber si obdobně přihlásí odběr požadovaného tématu. Z logiky věci je tedy důležité, aby si Publisher a Subscriber dohodly požadovaná témata. V tabulce číslo X jsem navrhl jednoduchý datový model pro výměnu

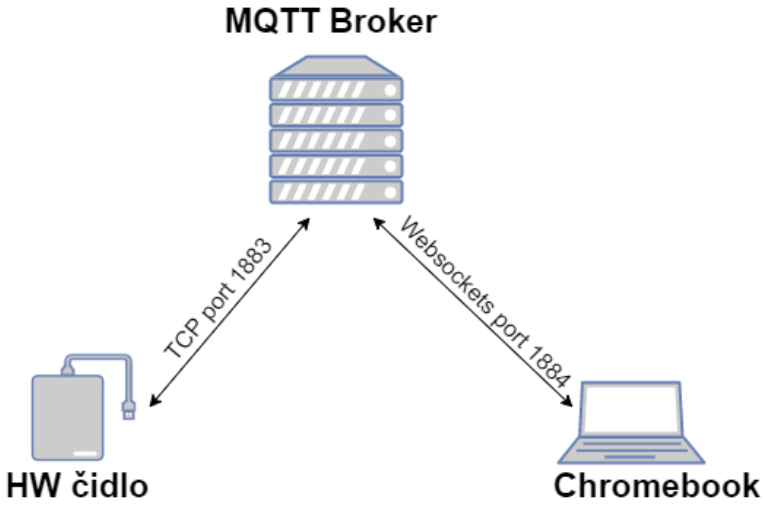
zpráv mezi čidlem a klientem. „Node-1“ reprezentuje čidlo číslo 1. V případě více čidel, by se číslo iterovalo. V podúrovni jsou potom obsaženy hodnoty, s kterými čidlo operuje. „Relay-1“ a „relay-2“ představují kontrolní signály pro ovládání relé. Jak si můžeme všimnout, tak je publikuje klient v ChromeOS, zatímco čidlo je přihlášeno k jejich odběru. Přesně opačně je to s měřenými hodnotami v tématech „temperatureC-1“, teplotě ve stupních celsia a „humidity-1“, tedy procentuální vlhkosti vzduchu. Čidlo také publikuje stavy relé v tématech „relay-1state“ a „relay-2state“ pro případ, kdy by mělo na sobě například hardwarová tlačítka pro manuální přepnutí relé. [24]

Topic	**Publisher**	**Subscriber**
node-1/relay-1	ChromeOS	HW čidlo
node-1/relay-2	ChromeOS	HW čidlo
node-1/temperatureC-1	HW čidlo	ChromeOS
node-1/humidity-1	HW čidlo	ChromeOS
node-1/relay-1state	HW čidlo	ChromeOS
node-1/relay-2state	HW čidlo	ChromeOS

Tabulka č. 8 – Návrh MQTT komunikace

4.3.2 Instalace serveru a MQTT Brokeru

Na obrázku číslo 2 jsem naznačil schéma spojení. HW čidlo se spojí s Brokerem pomocí TCP. Použijeme port 1883. (Pro šifrované spojení s TLS se používá port 8883). V současné době bohužel není možné komunikovat čistým MQTT s webovými prohlížeči, protože jim chybí možnost navázat raw TCP spojení (existují pouze experimentální buildy několika prohlížečů). Tato skutečnost by znemožňovala komunikaci s ChromeOS, proto nakonfiguruji MQTT Brokera tak, aby komunikoval pomocí Websockets na portu 1884. Websockets je technologie, která nám umožní navázat obousměrné spojení s Brokerem, se kterým si pak můžeme vyměňovat informace v reálném čase. Díky tomu nám odpadne složité dotazování serveru (takzvaný heartbeat u HTTP) a zachováme si i požadovanou rychlost, jelikož servisní data v odesílaných rámcích obsahují navíc jen 2B. [26]



Obrázek č. 12 - TCP/IP komunikace mezi zařízeními

MQTT Brokerů existuje celá řada. Dostupné jsou jako binární řešení pro vlastní instalaci, tak i jako cloudové služby. Záleží tedy hlavně na tom, jak moc chcete mít centrální prvek pod kontrolou a s jakou ochotou poskytnete jiné straně vaše data a kontrolu nad nimi. V tomto případě jsem zvolil vlastní instalaci.

Jako vhodný MQTT Broker jsem vybral program Mosquitto. Je kompatibilní se všemi myslitelnými operačními systémy jako Windows, Linux, MAC a FreeBSD. Mosquitto je open source program pod licencí EPL/EDL. Implementuje MQTT protokol ve verzi 3.1 a poskytuje řadu pokročilých funkcí jako je šifrování a QoS.

Pro instalaci MQTT Brokeru jsem vytvořil virtuální server o následujících parametrech. Musím podotknout, že výkon serveru je velmi nadsazený. Požadavky na výkon jsou totiž velmi malé. Díky tomu je velmi populární například implementace mosquitto pro mikropočítač Raspberry Pi. [25]

Operační systém	Debian 8.6.0 „Jessie“
CPU	Intel Core i7 4770K – 2 jádra
RAM	1 GB
HDD	8 GB

Tabulka č. 9 - Hardwarové prostředky MQTT serveru

Samotnou instalaci a konfiguraci MQTT Brokeru uvedu jako posloupnost příkazů tak, jak jsem je zadával do terminálu pod standardním uživatelem.

Ze všeho nejdříve updatuji repozitáře a nainstalovaný software. Poté nainstaluji nástroje potřebné ke kompilaci.

```
sudo apt-get update
sudo apt-get upgrade
apt-get install install wget cmake zlib1g-dev libssl-dev libc-res-dev uuid-dev git gcc g++
```

Websockets je možné nainstalovat jako součást webového serveru (např. Apache2), nicméně v tomto případě to není potřeba. Stačí nám MQTT Broker, tedy program mosquitto s podporou websockets. Zkompiluji si ho tedy s podporou libwebsockets, což je nenáročná knihovna napsaná v jazyce C. Začnu kompilací libwebsockets, jediný parametr je –DLIB_SUFFIX=64, ten říká, že kompilace je určena pro 64bitový procesor.

```
git clone https://github.com/warmcat/libwebsockets.git
cd libwebsockets
mkdir build
cd build
cmake .. -DLIB_SUFFIX=64
sudo make
sudo make install
```

Nyní se pustím do instalace samotného MQTT Brokeru mosquitto. Začnu přidáním GPG veřejného klíče k repozitáři s balíčky mosquitto do systému pro správu software.

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
```

Dále vložím seznam odkazů pro použitou verzi operačního systému Debian Jessie na tento repozitář a poté z něj mosquitto nainstaluji.

```
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
```

```
sudo apt-get install mosquitto
```

Následujícím příkazem vytvořím konfigurační soubor.

```
cat > /etc/mosquitto/mosquitto.conf <<EOF
pid_file /var/run/mosquitto.pid                    #soubor s proces ID
persistence true                                   #povolíme ukládání komunikace
persistence_location /var/lib/mosquitto            #umístění databáze
log_dest file /var/log/mosquitto/mosquitto.log     #ukládání logů programu
listener 1883                                      #naslouchame na portu 1883
protocol mqtt                                      #protokol mqtt
listener 1884                                      #naslouchame na portu 1884
protocol websockets                                #protokol websockets
EOF
```

Můžu mosquitto nastartovat jako službu a nastavit automatické spuštění serveru při startu operačního systému.

```
sudo /etc/init.d/mosquitto start
sudo update-rc.d mosquitto defaults
```

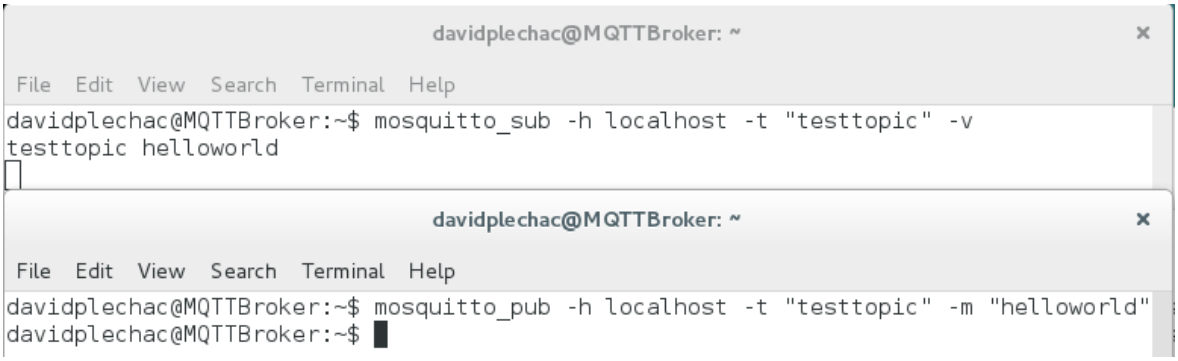
Ověřím, zdali služba opravdu poslouchá na žádaných portech.

```
sudo ss -a -p | grep mosq
```

```
davidplechac@MQTTBroker:~$ sudo ss -a -p | grep mosq
tcp    LISTEN     0      100                *:1883             *:*       users:(("mosquitto",pid=1869,fd=4))
tcp    LISTEN     0      128                *:1884             *:*       users:(("mosquitto",pid=1869,fd=7))
```

Obrázek č. 13 - MQTT Broker naslouchá na určených TCP portech

Nakonec mohu ověřit funkčnost samotného publish – subscribe mechanismu. Program mosquitto má na to implementovány metody. V rámci serveru se v jednom terminálu přihlásím k odběru tématu „testtopic“ a následně v jiném terminálu do stejného tématu publikuji zprávu „helloworld“.



Obrázek č. 14 - Lokální ověření funkčnosti MQTT Brokeru

Z výstupu je zřejmé, že MQTT Broker mosquitto je připraven k použití a je plně funkční.

4.4 Hardwarové čidlo

Jako hardwarový základ pro čidlo vhodné k účelům praktické části této práce jsem zvolil platformu Arduino.

4.4.1 Arduino

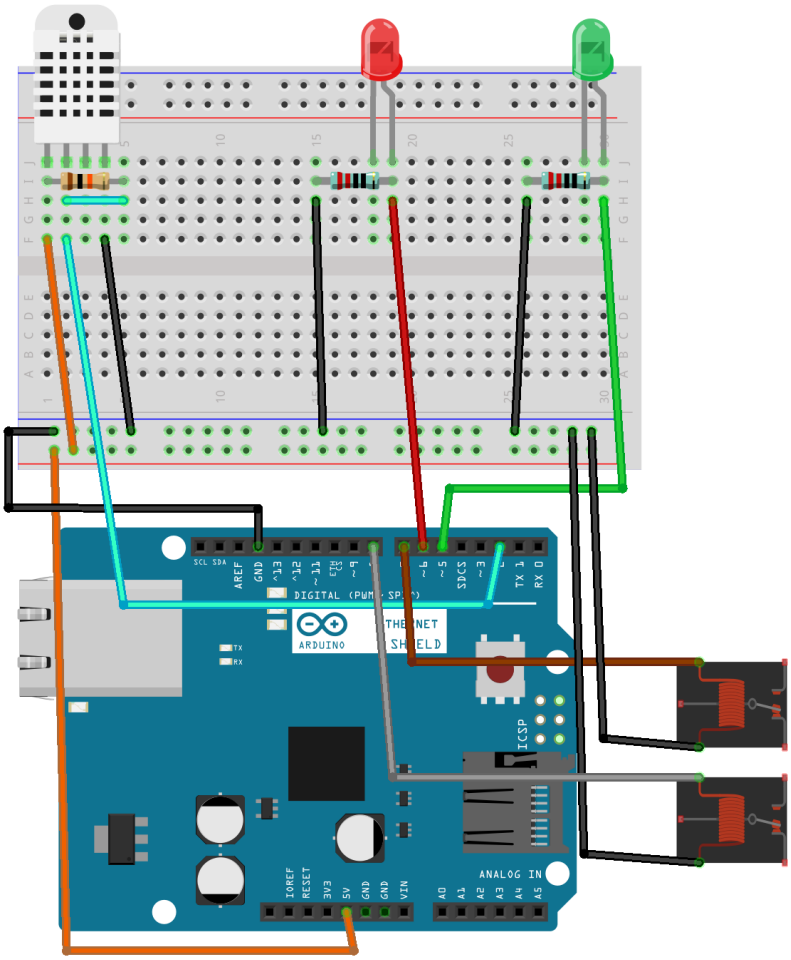
Projekt Arduino vznikl v roce 2005 v Itálii. Je to otevřená prototypovací platforma, jejímiž hlavními cíli jsou rychlý vývoj a jednoduché používání. Projekt Arduino je již od svého počátku volně dostupný (open-source) všem uživatelům, kteří jej chtějí používat nebo vylepšovat.

Arduino není zamýšleno jako plnohodnotný počítač, jeho srdcem je 8bitový mikrokontrolér z rodiny AVR od firmy Atmel, z tohoto důvodu je mikrokontrolér postavený podle harvardské architektury. Tudíž má fyzicky oddělenou programovou a datovou paměť. Řídící program je tedy vyvíjen na jiném počítači zvlášť a poté je do Arduina nahrán a spuštěn. K tomuto účelu Arduino používá USB-to-RS-232 převodníkový čip, který zajišťuje virtuální sériovou linku pro propojení s většinou platforem.

Pro Arduino existuje grafické vývojové prostředí (IDE) napsané v jazyce Java. Toto IDE přímo podporuje programovací jazyky C a C++, nicméně program pro Arduino jako takový může být napsán v jakémkoli jazyce, který je možno zkompilovat do binárního strojového kódu. [27]

4.4.2 Hardware

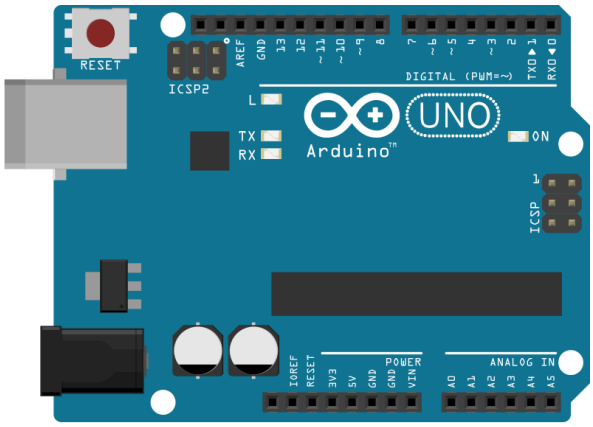
Pro zapojení jsem použil kontaktní nepájivé pole (takzvaný breadboard). To umožňuje díky pružinovým kontaktům opakovaně zapojovat jednotlivé součástky nebo propojovací dráty bez nutnosti pájení.



Obrázek č. 15 - Schéma zapojení HW čidla

4.4.2.1 Arduino UNO – mozek zařízení

Pro účely této práce jsem využil základní model Arduino UNO, který je osazen mikrokontrolerem ATmega328P. Tento model je vybaven několika diodami, napájecím konektorem, oscilátorem, obvodem zprostředkujícím komunikaci po USB virtuálním převodníku, resetovacím tlačítkem, konektory pro ICSP programování a hlavně 14 digitálními I/O piny a 6 piny analogovými. [27]



Obrázek č. 16 - Arduino UNO rev. 3

4.4.2.2 Ethernet Shield – připojení k internetu

I/O piny, kterými je deska Arduina osazena, jsou uspořádány do standardizované patice tak, aby se na ně jednoduše daly připojit další obvody, těm se v terminologii Arduina říká shieldy. Pro vytvoření této práce budeme potřebovat komunikovat po síti pomocí protokolu TCP/IP. Nabízí se tedy možnost Arduino osadit buď ethernet shieldem, který obsahuje síťovou kartu s konektorem RJ45 pro standardní UTP síťový kabel a nebo wifi shield pro bezdrátové připojení pomocí standardu IEEE802.1b/g. Pro účely práce jsem použil ethernet shield, jelikož umožňuje použití PoE (Power over ethernet), tedy napájení po UTP kabelu podle standardu IEEE802.3af a také nabízí slot pro micro-SD kartu, jenž se dá využít například pro lokální logovaní naměřených dat. Tento ethernet shield pohání kontroler Wiznet W5500, který umožňuje použití protokolů TCP a UDP a zvládne udržovat až 8 aktivních spojení v jeden okamžik.

Ethernet shield se propojí s Arduinem UNO prostým zasunutím do sebe. Komunikace mezi oběma zařízeními probíhá po sběrnici SPI a digitálních pinů 10, 11, 12 a 13. Tato komunikace je sdílená jak pro síťovou kartu, tak pro SD kartu. Síťová karta se vybere pomocí pinu 10 a SD karta pomocí pinu 4. Z toho vyplývá, že tyto dva piny nemůžou být použity pro nic jiného a zároveň, že může být použita buď síťová karta, nebo SD karta. [28]

4.4.2.3 DHT22 - měření teploty a vlhkosti vzduchu

Pro měření teploty a vlhkosti jsem použil digitální čidlo DHT22. DHT22 je základní, relativně levné čidlo. Pro měření vlhkosti je vybaveno kapacitním senzorem vlhkosti, sledujícím změnu kapacity kondenzátoru. Měřitelný rozsah vlhkosti vzduchu je 0 až 100% s 2-5% přesností. Pro měření teploty čidlo používá termistor. Měřitelný rozsah teploty vzduchu je -40 až 80 °C s přesností +- 0,5 °C. Čidlo komunikuje po 1-wire sběrnici a je schopné provádět měření obou veličin každé dvě sekundy.

Čidlo DHT22 je na prvním pinu napájeno 5V. Pomocí druhého pinu jsou z něj vyčítána data, a to pomocí digitálního pinu 2 v Arduinu. Tento datový pin je spolu s pinem napájecím propojen pull up rezistorem o odporu 10 kΩ. Čidlo funguje i bez tohoto resitoru, nicméně je výrobcem doporučeno jej použít kvůli zamezení problémů s vysokou impedancí a k zachování takzvané měkké jedničky. Třetí pin čidla je nevyužitý a posledním, čtvrtým pinem, je čidlo uzemněno. [29]

4.4.2.4 Silnoproudé rélé – spínání elektrických zařízení

Pro spínání elektrických zařízení jsem použil silnoproudé rélé. Jedná se o elektricky ovládáný spínač, zvládající AC250V při 10A. Běžně se prodávají 5V Relay moduly, které při vstupu logického napětí přepnou relé, aby proud procházel nebo neprocházel v závislosti na zapojení. Relé se obvykle skládá z cívky, jedné společné svorky, jedné rozpínací svorky a jednoho spínacího terminálu. Když je cívka pod napětím, společná svorka a normálně otevřený terminál budou propojeny. V této práci jsem se rozhodl použít dvě relé. První je zapojeno do Arduina na digitálním pinu 7, druhé relé na digitálním pinu 8.

4.4.2.5 LED diody – signalizace spojení s MQTT Brokerem

Pro názornost jsem se rozhodl hardwarové čidlo vybavit signalizací připojení k MQTT Brokeru pomocí LED diod. Rozsvícená zelená dioda značí, že je čidlo připojeno. Červená oproti tomu, že se připojení nedaří. Zelenou LED diodu jsem připojil k digitálnímu pinu 5 na Arduinu, červenou LED diodu k digitálnímu pinu 6. Každou z LED diod je nezbytné připojit k pinům před resistor, jelikož je nutné omezit proud přes ně protékající, abychom zabránili spálení či poškození LED diody, nebo dokonce

samotného mikrokontroléru. V datasheetu od výrobce LED diod je uvedená hodnota úbytku („Forward Voltage“) napětí 2,2 V při nominální proudu („Forward Current“) 20 mA. K určení vhodného resistoru se musíme řídit pomocí Druhého Kirchhoffova zákona, který říká, že součet všech napětí v obvodu je roven nule. Jelikož napájíme z 5V zdroje a LED má stanovený úbytek napětí na 2,2 V, prostým odečtením zjistíme úbytek 2,8V, s kterým se musíme resistorem vypořádat. Nominální proud, na kterém je doporučeno tyto LED provozovat, je 20 mA. Abychom se na tuto hodnotu dostali, dosadíme do rovnice předepsané Ohmovým zákonem – R = 2,8/0,02. Vychází, že potřebný resistor by měl mít výsledný odpor minimálně 140 Ω. Tak malý resistor jsem neměl, místo něj jsem použil resistor s odporem 220 Ω, to se projevilo o něco menší světelnou intenzitou, jelikož se více energie přeměnilo v teplo namísto světla.

4.4.3 Firmware

Řídící program pro hardwarové čidlo jsem vytvořil pomocí Arduino IDE a jazyka C. Použil jsem veřejně dostupné knihovny pod licencí LGPL. Zdrojový kód je připojený v příloze v souboru David_Plechac_DP2017.ino. Pro diagnostiku a ladění programu jsem používal virtuální sériovou linku.

Použitá knihovna SPI.h poskytuje obslužné funkce pro ovládání SPI sběrnice. Tuto sběrnici využívá další použitá knihovna Ethernet.h. Díky tomu může Arduino UNO komunikovat s kontrolerem Wiznet 5500 a využívat TCP/IP stack nebo SD kartu. Další využitou knihovnou je DHT.h, tato knihovna poskytuje obslužné funkce pro DHT čidla teploty a vlhkosti. Poslední použitou knihovnou je PubSubClient.h, tato knihovna umožňuje publish/subscribe komunikaci s MQTT Brokerem. V její poslední verzi je implementována podpora MQTT verze 3.1.1. Knihovna je veřejně dostupná pod MIT licencí.

Celý firmware se skládá ze dvou hlavních funkcí a několika pomocných funkcí. Hlavní dvě funkce jsou nezbytné k běhu programu.

První funkcí je setup(). Tato funkce se provede po spuštění Arduina. Nastaví se sériová linka a jednotlivé digitální piny. Inicializuje se 1-one sběrnice pro čidlo DHT22. Poté probíhá nastavení TCP/IP. To se skládá z pokusu o získání vlastní IP adresy z DHCP serveru, nastavení komunikace pro MQTT Broker a určení takzvané callback funkce.

Funkce callback() slouží k vyřizování příchozí komunikace od MQTT Brokera. V rámci této práce je to situace, kdy budeme na Chromebooku spínat relé. Při přijmutí řídícího příkazu funkce zjistí, jaký topic má obsloužit a podle toho sepne příslušné relé.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  char buffer[10];
  String str(topic);
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  // Control command for relay1
  if(str ==  "node-1/relay-1"){
       Serial.print("RELE1: ");
       rele1 = atoi(payload);
      digitalWrite(relay1, rele1);
    }
  // Control command for relay2
 else if(str == "node-1/relay-2"){
    rele2 = atoi(payload);
      digitalWrite(relay2, rele2);
  }
  }
```

Obrázek č. 17 – Callback() funkce pro spínání relé

Druhou hlavní funkcí je funkce loop(). Tato funkce realizuje smyčku, kterou mikrokontrolér provádí donekonečna. Kvůli rychlosti čidla DHT22 se smyčka provádí každé dvě vteřiny. Na začátku cyklu program ověří TCP/IP připojení k síti. Pokud bylo připojení ztraceno, pokusí se ho obnovit. Poté program ověřuje dostupnost MQTT Brokera. Pokud připojení k MQTT Brokeru není dostupné, tak se ho program snaží obnovit pomocí funkce reconnect(). Tato funkce se každých 5 vteřin pokusí připojit k MQTT Brokeru a úspěšnost tohoto pokusu světelně signalizuje pomocí připojených LED diod podle legendy zelená – připojeno, červená – odpojeno. V poslední části funkce loop() program čte digitální vstup z one-wire sběrnice od čidla DHT22 a jak naměřenou vlhkost, tak i teplotu publikuje do příslušných témat MQTT Brokeru.

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("arduinoClient")) {
      Serial.println("connected");
    // LED state - connected
      digitalWrite(greenled, 1);
      digitalWrite(redled, 0);
    // Relays subscription
      client.subscribe("node-1/relay-1");
      client.subscribe("node-1/relay-2");
    } else {
      Serial.print("failed, rc=");
    // LED state - disconnected
      digitalWrite(greenled, 0);
      digitalWrite(redled, 1);
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

Obrázek č. 18 - Funkce reconnect() pro připojení k MQTT Brokeru

4.5 Instalace operačního systému Chromium OS

Oficiální cestou jsou veřejně poskytovány pouze zdrojové kódy, nikoli již zkompilovaný systém. U neoficiálních zdrojů nelze pravost systému ověřit a nezbývá tedy než slepě věřit zdroji, od kterého si systém stáhneme. Proto jsem se rozhodl provést následující postup, kterým jsem si systém postavil ze zdrojových kódů tak, abych ho mohl provozovat na manažerském ultrabooku Dell Latitude e7740. Tento stroj bude v praktické části práce suplovat Chromebook.

K samotnému zkompilování Chromium OS potřebujeme linuxový stroj. Podotknu, že stroj výkonný, protože tento proces je poměrně náročný. Stejně tak je důležité dobré internetové připojení, jelikož se během různých kroků stahují velké objemy dat. Vytvořil jsem si tedy virtuální stroj o následujících parametrech.

4.5.1 Hardware pro kompilaci Chromium OS

- Operační systém – linux Ubuntu 14.04.5 LTS (Trusty Tahr)
- CPU Intel i7 2600 (virtuálnímu stroji jsem přidělil 4 procesorová jádra)
- 8 GB RAM (dostatečné množství pro zabránění swapování na HDD)

- 50GB HDD (dostatečně velký pro vše, co je ke kompilaci potřebné)
- Připojení k internetu 100/100 Mb/s (download/upload)

4.5.2 Software pro kompilaci Chromium OS

Pro úspěšnou kompilaci systému potřebujeme stáhnout, nainstalovat a posléze nakonfigurovat následující software. Uvedená posloupnost příkazů je zadávána do terminálu pod standardním uživatelem.

Ze všeho nejdříve updatujeme repozitáře a nainstalovaný software.

```
sudo apt-get update
sudo apt-get upgrade
```

Git – distribuovaný systém verzí, používá se pro vývoj software

```
sudo apt-get install git-core gitk git-gui curl
```

Je nutné v Gitu nastavit jméno uživatele a email, bez toho jsem později při kompilaci narazil na chyby.

```
git config --global user.email "xpled001@czu.cz"
git config --global user.name "David Plechac"
```

Depot_tools – balík skriptů sloužící ke správě zdrojových kódů

```
git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
```

Přidáme depot_tools do proměnné PATH pro jednoduché spouštění

```
Export PATH=`pwd`/depot_tools:“$PATH“
```

Nyní je potřeba upravit soubor sudoerst k vypnutí volby tty_tickets při použití příkazu sudo, jelikož není kompatibilní s později použitým cros_sdk.

```
cd /tmp
cat > ./sudo_editor <<EOF
#!/bin/sh
echo Defaults \!tty_tickets > \$1
echo Defaults timestamp_timeout=180 >> \$1
EOF
chmod +x ./sudo_editor
sudo EDITOR=./sudo_editor visudo -f /etc/sudoers.d/relax_requirements
```

Jako poslední potřebujeme nastavit implicitní masku pro nově vytvořené soubory, v našem případě stažené zdrojové kódy.

```
umask 022
```

4.5.3 Stažení zdrojových kódů Chromium OS

Vytvoříme složku pro zdrojové kódy a přesuneme se do ní.

```
mkdir -p ${HOME}/chromiumos
cd ${HOME}/chromiumos
```

Nyní stáhneme repozitáře pro Git obsahující zdrojové kódy a manifest. Vše se uloží do nově vytvořené podsložky **.repo/**.

```
repo init -u https://chromium.googlesource.com/chromiumos/manifest.git --repo-url
https://chromium.googlesource.com/external/repo.git
```

V tomto kroku již provedeme samotné stažení zdrojových kódů. Příkaz repo synchronizuje lokální obsah s obsahem repozitářů. Při první synchronizaci funguje obdobně jako například git clone. Poté se synchronizují hlavně nové změny, takže

nemusíme stahovat pokaždé vše znova. Parametrem příkazu říkáme, z kolika repozitářů naráz chceme stahovat.

```
repo sync -j4
```

4.5.4 Kompilace Chromium OS

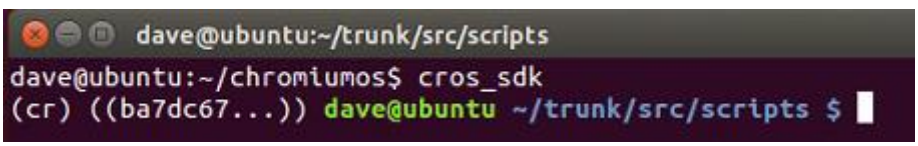
Konečně máme vše připraveno pro samotnou kompilaci, která se provádí uvnitř chrootu, a to změnou kořenového adresáře. Ten poté funguje jako jail sám pro sebe. Obsahuje vlastní kompilátor, vlastní nástroje (např. vlastní kopii shellu Bash, vlastní příkaz sudo atd.). Pro další postup předpokládám, že se nacházíme uvnitř složky, kterou jsme vytvořili pro synchronizaci zdrojových kódů.

```
cd ${HOME}/chromiumos
```

Následujícím příkazem stáhneme a nainstalujeme chroot. Stejným příkazem do něj vstoupíme, pokud je již nainstalovaný.

```
cros_sdk
```

Nyní jsme v chrootu, poznáme to podle speciálního „(cros-chroot)“ promptu. Jak si můžeme všimnout, většina složek v chrootu je namountována na složky vytvořené v naší složce se zdrojovými kódy. Samotný chroot je v podsložce **${home}/chromium/chroot.** Uvnitř této složky jsou systémové složky jako **/usr/bin** nebo **/etc.** Tyto složky naopak nejsou namountované a jsou lokálními kopiemi jedinečnými pro chroot, tudíž disponují vlastními verzemi programů a jsou úplně nezávislé na zbytku tohoto linuxového stroje. Všechny následující příkazy je třeba zadávat ze složky **`~/trunk/src/scripts`**.



Obrázek č. 19 - Cros_sdk chroot

V chrootu je nutné specifikovat, pro jaký stroj systém kompilujeme. Takzvaně vybrat desku. Jinak by spustitelné soubory nemusely být kompilovány pro správnou instrukční sadu. Seznam známých desek je umístěn v **~/trunk/src/overlays**. Ultrabook, pro který systém připravuji, má 64 bitový procesor.

```
export BOARD=amd64-generic
```

Můžeme inicializovat kompilátor pro zvolenou desku. Skript udělá defaultní sysroot ze složky **/build/${BOARD}**, stáhne potřebná data a připraví kompilaci.

```
./setup_board --board=${BOARD}
```

Následujícím skriptem si nastavíme heslo pro sdíleného uživatele „chronos“. Pod tímto účtem se můžeme v Chromium OS přihlásit do terminálu a odtud získat například super uživatelský přístup (sudo). Heslo je zašifrovaně uloženo v **/etc/shared_user_passwd.txt**.

```
./set_shared_user_password.sh
```

V tento moment nic nebrání samotné kompilaci. Nejprve zkompilujeme balíčky pro naši desku. Použitý skript v podstatě rozšiřuje standardní **make all** z Makefile systému.

```
./build_packages --board=${BOARD}
```

Po úspěšné kompilaci balíků můžeme přejít k finální kompilaci obrazu disku. Parametrem **base** určujeme, že chceme obraz nejvíce podobný oficiálnímu Chrome OS. Pro vývojářskou verzi je možné použít parametr **dev.**

```
./build_image --board=${BOARD} base
```

Obraz disku zkompilovaného systému je nyní připraven a uložen v **~/trunk/src/build/images/${BOARD}/versionNum/** (kde versionNum je číslo verze).

Skript se také postará o to, že na poslední zkompilovaný obraz vždy ukazuje symbolický link **~/trunk/src/build/images/${BOARD}/latest**. To použijeme v následujícím kroku, ve kterém si nahrajeme zkompilovaný systém na USB disk. Použil jsem USB disk o velikosti 32 GB, stačit by měl i disk podstatně menší kapacity.

```
cros flash usb:// ${BOARD}/latest
```

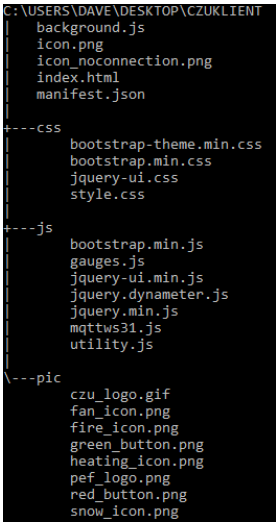
S připraveným USB diskem můžeme nabootovat systém Chromium OS na cílovém ultrabooku. Do příkazového terminálu je možno se dostat například tak, že po nabootování systému stisknete kombinaci kláves CTRL + ALT + F2. Nyní je možné se přihlásit jako uživatel „chronos“ a použít heslo, které jsme nastavili v dřívějším kroku. Zadáním následujícího příkazu lze Chromium OS nainstalovat na pevný disk počítače, což vymaže veškerý dosavadní obsah pevného disku.

```
/usr/sbin/chromeos-install
```

Celkem kompilace i na poměrně výkonném stroji a rychlém internetovém připojení zabrala kolem pěti hodin času. Kompilace operačního systému je proces velmi komplexní. Postup, který jsem zvolil, je dostatečný pro účely práce aniž by překračoval její rámec.

4.6 Vytvoření aplikace pro Chrome OS

4.6.1 Struktura aplikace



Obrázek č. 20 - Struktura aplikace

Struktura aplikace je naznačena na obrázku číslo 20. Celá struktura představuje takzvaný Chrome App package neboli balíček, který obsahuje veškeré součásti aplikace. Podložky aplikace jsou pojmenovány podle typů souborů v nich uložených. Podsložka \css tedy obsahuje soubory kaskádových stylů, podsložka \js soubory javascriptu a složka \pic použitá loga a obrázky. Kořenová složka celé struktury obsahuje ikony aplikace a dva důležité soubory, které jsou klíčové pro každou Chrome aplikaci.

Prvním z nich je soubor manifest.json. Jak je z přípony souboru patrné, jedná se o datový formát JSON. Soubor obsahuje datovou strukturu uchovávající důležitá metadata této aplikace. V nich je definována například verze aplikace, ikona (soubor icon.png), minimální požadovanou verzi Chrome OS pro běh aplikace. Velmi důležitou položkou je definice oprávnění. Aplikace využívá vyskakující notifikace, které je třeba povolit. Dále je třeba povolit přístup k externím souborům, jež aplikace využívá, jako jsou skripty (podsložka js) a css styly (podsložka css).

Druhým důležitým souborem je background.js. Tento skript se provede ihned po spuštění aplikace. Vytvořil jsem tedy událost, která po startu otevře hlavní okno samotné aplikace definované v souboru index.html.

```
chrome.app.runtime.onLaunched.addListener(function() {
  chrome.app.window.create('index.html', {
    id: 'main',
    bounds: { width: 620, height: 500 }
  });
});
```

Obrázek č. 21 - Událost vytvářející hlavní okno aplikace

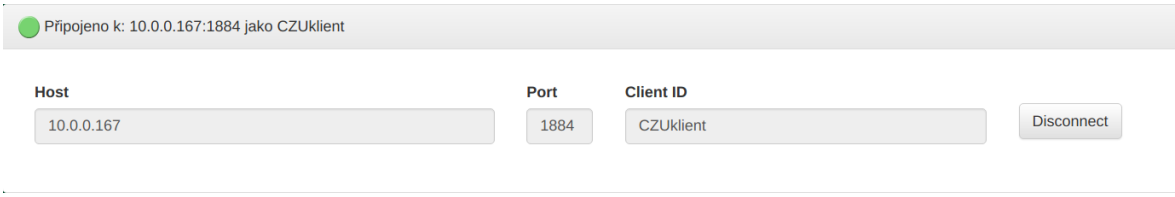
4.6.2 Grafické uživatelské rozhraní (GUI) – Bootstrap

Grafické uživatelské rozhraní jsem vytvořil pomocí frameworku Bootstrap s několika vlastními modifikacemi. Bootstrap vychází z webových technologií HTML a CSS. Na těchto technologiích jsou postaveny návrhářské šablony pro vytváření formulářů, tlačítek, navigačních menu a dalších komponent rozhraní. Díky tomu lze pomocí jednoduché konvence vkládat interaktivní prvky a kompletně graficky zpracované elementy. Většina elementů použitých v této práci vychází z frameworku Bootstrap a jsou definovány v souboru index.html pomocí nastavení příslušných tříd. Stylizace elementů, které jsem upravoval potřebám aplikace nebo sám vytvářel, je definována v souboru css/style.css. Samotné grafické rozhraní jsem nastyloval do typických barev ČZU, tedy tmavě zelené barvy s bílými prvky v popředí.

Aplikaci dominuje vrchní navigační lišta, nesoucí logo univerzity spolu s logem fakulty. Navigační lišta je stylovaná třídami navbar a navbar-inverse, u kterých jsem přestyloval základní parametry jako barvy a pozicování. Lišta je zafixovaná na vrchní okraj stránky pomocí třídy navbar-fixed-top a zůstane tam i při skrolování.

Tělo aplikace jsem nakódoval pomocí takzvaného Bootsrap Gridu. To je stylovací mříž, která umožňuje pozicování až 12 individuálních sloupců na jednom řádku. S celým obsahem těla aplikace tedy nakládám podobně jako s tabulkou.

Formulář na prvním řádku slouží k připojení k MQTT Brokeru. Zde je možnost zadat adresu Brokeru a TCP komunikační port. Poslední možností je vyplnit Client ID, tedy identitu, pod kterou toto připojení Broker eviduje. Kliknutím na tlačítko zahájíme připojení k zadanému Brokeru. Stav připojení je znázorněn v hlavičce formuláře. Červeným puntíkem a nápisem Nepřipojeno, nebo puntíkem zeleným s nápisem Připojeno. Funkci tohoto formuláře jsem chtěl původně zabudovat pevně do aplikace a zobrazovat pouze status připojení, nicméně samostatný formulář se mi osvědčil jako praktičtější řešení.

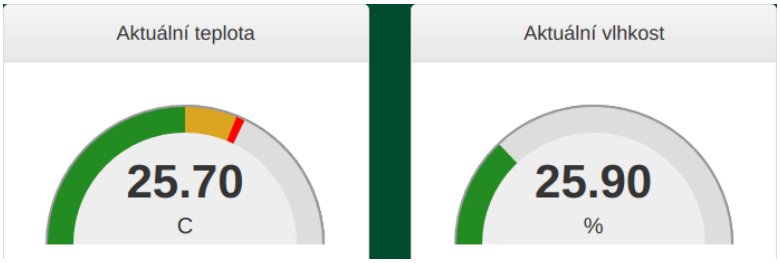


Obrázek č. 22 - Formulář připojení k MQTT Brokeru

Pomyslný druhý řádek mřížky jsem rozdělil na tři nezávislé sloupce. První a druhý sloupec slouží k zobrazení přijímaných dat z MQTT Brokeru v reálném čase. Pro grafický efekt vykreslení naměřených hodnot jsem využil jquery.dynameter plugin. Tento plugin umožňuje v reálném čase zobrazovat naměřené hodnoty v půlkruhové výseči s barevně definovanými intervaly podle aktuální hodnoty. V souboru js/gauges.js jsem vytvořil instance objektů pro ukazatel teploty a vlhkosti, tyto objekty jsou přímo svázány s příslušnými formuláři pro ně určenými. V tabulce číslo 10 jsem uvedl parametry, jež jsem nastavil pro barevné vykreslování. [32]

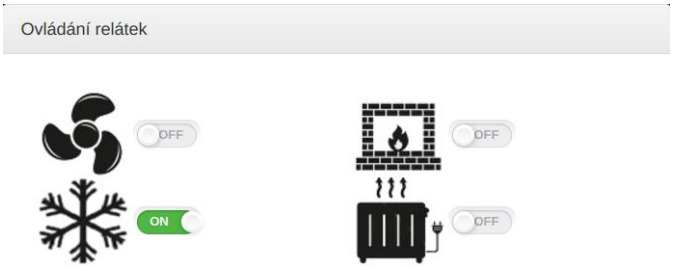
Vzduch	**Minimum**	**Maximum**	**Zelená**	**Žlutá**	**Červená**
Teplota	0 °C	40 °C	[0 °C;20 °C]	[20 °C;25 °C]	[25 °C;40 °C]
Vlhkost	0 %	100 %	[0 %;40 %]	[40 %;60 %]	[60 %;100 %]

Tabulka č. 10 - Rozsahy grafického znázornění měřených hodnot



Obrázek č. 23 - Grafického znázornění naměřených hodnot

Poslední, třetí sloupec druhého řádku, slouží k ovládání relé na HW čidle. Formulář pro ovládání obsahuje čtyři tlačítka. Dvě z nich jsou plně funkční, další dvě slouží hlavně ilustračně, jelikož na HW čidle jsou zapojena jen 2 relé. Každému tlačítku jsem pro efekt nastyloval animaci přepnutí pomocí CSS ve smyslu zapnuto – zelená, vypnuto – šedivá. Každé tlačítko má také přidělenu ikonu elektrospotřebiče, jež by eventuálně mohlo spínat.

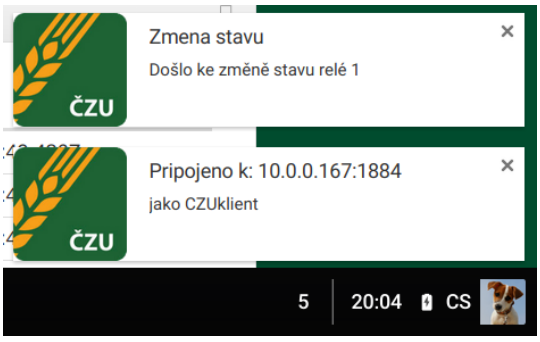


Obrázek č. 24 - Grafický ovládací panel pro spínání silnoproudých relé

Jako další prvky jsem se rozhodl implementovat výpisy komunikace tak, aby bylo zřetelně vidět, jak komunikace probíhá. Na třetím řádku pomyslné mřížky jsem umístil formulář s poslední přijatou komunikací v tématech, která klient odebírá od MQTT Brokera. Formulář obsahuje tabulku se třemi sloupci. První z nich reprezentuje odebírané vlákno, druhý obsah přijaté zprávy v tomto vláknu a třetí sloupec čas, kdy zpráva dorazila. V pravém horním rohu formuláře jsem umístil tlačítko pro zabalení celého formuláře tak, aby výpis nebyl vidět.

Úplně poslední implementovaný formulář má stejný formát jako formulář předchozí. Nicméně, oproti předchozímu, tento formulář obsahuje celou historii přijatých zpráv, a to v pořadí, v jakém je klient od MQTT Brokeru obdržel. Tento formulář také obsahuje tlačítko pro vymazání výpisu historie zpráv.

Pro ještě větší grafickou názornost jsem se rozhodnul implementovat funkci, jež využije systém notifikací z Chrome OS API. Každá důležitá akce je tedy reflektovaná malým upozorněním v rohu obrazovky tak, aby uživateli neunikla, i když zrovna nemá klientskou aplikaci aktivní. Každá tato notifikace zůstane zobrazena deset sekund.



Obrázek č. 25 - Použití notifikačního API v Chrome OS aplikaci

4.6.3 MQTT komunikace - Eclipse Paho

K vytvoření funkční komunikace s MQTT Brokerem pomocí WebSockets jsem použil javascriptové API z projektu Eclipse Paho. Toto open-source API umožňuje použít verzi MQTT 3.1.1 a je uloženo v souboru js/mqttws31.js. Všechny obslužné metody, které aplikace využívá, jsem vytvořil v souboru js/utility.js. Pro ladění a diagnostiku aplikace jsem využil výpisu do konzole. [31]

Většina interní funkcionality aplikace se skrývá za tlačítkem Connect, sloužícím k připojení k MQTT Brokeru. Při kliknutí na tlačítko se vyvolá funkce connectionToggle(), tato funkce vyhodnotí, zda je již klient připojen nebo nikoliv.

Pokud klient není připojen k MQTT Brokeru, vykoná se funkce connect(). To je asi nejdůležitější funkce celé aplikace. V těle funkce se vytvoří instance objektu MQTT WebSockets klienta z Paho API s předanými parametry z formulářů připojení. Dalším, velice důležitým prvkem této funkce, je nastavení takzvaných handlerů u vzniklé instance klienta. Tyto handlery principiálně vycházejí z Paho API, ale jejich definici je třeba přesně implementovat tak, aby bylo jasné, které funkce má klient v různých situacích volat.

```
function connect(){
    var hostname = document.getElementById("hostInput").value;
    var port = document.getElementById("portInput").value;
    var clientId = document.getElementById("clientIdInput").value;
    client = new Paho.MQTT.Client(hostname, Number(port), clientId);
    console.info('Connecting to Server: Hostname: ', hostname, '. Port: ', port, '. Client ID: ', clientId);

    // nastavení callback handlerů
    client.onConnectionLost = onConnectionLost;
    client.onMessageArrived = onMessageArrived;

    var options = {
      invocationContext: {host : hostname, port: port, clientId: clientId},
      onSuccess: onConnect,
      onFailure: onFail
    };
    // připojení k MQTT Brokeru
    client.connect(options);
    var statusSpan = document.getElementById("connectionStatus");
    statusSpan.innerHTML = 'Pripojuji...';
}
```

Obrázek č. 26 – Funkce connect() pro připojení Chrome OS aplikace k MQTT Brokeru

Prvním z callback handlerů je onConnectionLost(). Tato funkce se využije v situaci, kdy klient z nějakého důvodu ztratí připojení k MQTT Brokeru. Funkce změní stav aplikace, odblokuje zašedlé formuláře, změní ikonku indikace připojení a vypíše příslušnou chybovou hlášku a notifikaci.

```
function onConnectionLost(responseObject) {
  if (responseObject.errorCode !== 0) {
    console.log("Connection Lost: " + responseObject.errorMessage);
  }
  connected = false;
  setFormEnabledState(false);
     document.getElementById("connectionLED").src="pic/red_button.png";
     var statusSpan = document.getElementById("connectionStatus");
  statusSpan.innerHTML = "Ztraceno připojení k serveru: " + responseObject.errorMessage;
 popupNotification("Ztraceno připojení k serveru" ,responseObject.errorMessage,"icon_noconnection.png");
}
```

Obrázek č. 27 - Funkce onConnectionLost() je volána při ztrátě připojení k MQTT Brokeru

Další funkce pro příslušný handler je onMessageArrived(). Toto je asi nejsložitější funkce aplikace, jelikož zpracovává veškeré příchozí zprávy. Funkce obstará výpis historie přijatých zpráv do příslušných formulářových oken a zpracuje odebíraná témata pomocí sady podmínek. Dále funkce aktualizuje měřená data teploty a vlhkosti vzduchu na grafické vizualizaci aplikace, pokud obdrží zprávu o změně stavu relé, tak zobrazí notifikaci s touto informací.

```
function onMessageArrived(message) {
  console.log('Message Recieved: Topic: ', message.destinationName, '. Payload: ', message.payloadString, '. QoS: ', message.qos);
  console.log(message);
  var messageTime = new Date().toISOString();
  // Tabulka historie přijatých zpráv
  var table = document.getElementById("incomingMessageTable").getElementsByTagName('tbody')[0];
  var row = table.insertRow(0);
  row.insertCell(0).innerHTML = message.destinationName;
  row.insertCell(1).innerHTML = safe_tags_regex(message.payloadString);
  row.insertCell(2).innerHTML = messageTime;

  if(message.destinationName=="node-1/temperatureC-1"){
    $myTemperatureMeter.changeValue(safe_tags_regex(message.payloadString));
  } else if(message.destinationName=="node-1/humidity-1"){
    $myHumidityMeter.changeValue(safe_tags_regex(message.payloadString)) ;
  } else if(message.destinationName=="node-1/relay-1state"){
   if(Number(safe_tags_regex(message.payloadString))!=relay1)   {
   popupNotification("Zmena stavu", "Došlo ke změně stavu relé 1","icon.png");
   }
  relay1 = Number(safe_tags_regex(message.payloadString));
   document.getElementById("relayinput1").checked = relay1;
 } else if(message.destinationName=="node-1/relay-2state"){
  if(Number(safe_tags_regex(message.payloadString))!=relay2)   {
   popupNotification("Změna stavu", "Došlo ke změně stavu relé 2","icon.png");
   }
  relay2 = Number(safe_tags_regex(message.payloadString));
     document.getElementById("relayinput2").checked = relay2;
  }

  if(!document.getElementById(message.destinationName)){
      var lastMessageTable = document.getElementById("lastMessageTable").getElementsByTagName('tbody')[0];
      var newlastMessageRow = lastMessageTable.insertRow(0);
      newlastMessageRow.id = message.destinationName;
      newlastMessageRow.insertCell(0).innerHTML = message.destinationName;
      newlastMessageRow.insertCell(1).innerHTML = safe_tags_regex(message.payloadString);
      newlastMessageRow.insertCell(2).innerHTML = messageTime;
  } else {
      // Tabulka posledních přijatých zpráv
      var lastMessageRow = document.getElementById(message.destinationName);
      lastMessageRow.id = message.destinationName;
      lastMessageRow.cells[0].innerHTML = message.destinationName;
      lastMessageRow.cells[1].innerHTML = safe_tags_regex(message.payloadString);
      lastMessageRow.cells[2].innerHTML = messageTime;
  }
}
```

Obrázek č. 28 - Funkce onMessageArrived() spravuje přijaté zprávy od MQTT Brokeru

Funkce callback handleru onConnect() se provede v momentě, kdy se klientovi úspěšně podaří navázat spojení s MQTT Brokerem. Funkce nastaví zelený indikátor připojení, vypíše zprávu o úspěšném připojení, zašedne formuláře s parametry připojení tak, aby nešly během aktivního připojení měnit a hlavně přihlásí klienta k odběru témat od MQTT Brokeru.

```
function onConnect(context) {
  console.log("Client Connected");
  var statusSpan = document.getElementById("connectionStatus");
  statusSpan.innerHTML = "Připojeno k: " + context.invocationContext.host + ':'
  connected = true;
  setFormEnabledState(true);
    // Odebírání témat z HW čidla
    popupNotification("Pripojeno k: " + context.invocationContext.host + ':' +
    var qos = '0';
    client.subscribe('node-1/temperatureC-1', {qos: Number(qos)});
    client.subscribe('node-1/humidity-1', {qos: Number(qos)});
    client.subscribe('node-1/relay-1state', {qos: Number(qos)});
    client.subscribe('node-1/relay-2state', {qos: Number(qos)});
    document.getElementById("connectionLED").src="pic/green_button.png";
}
```

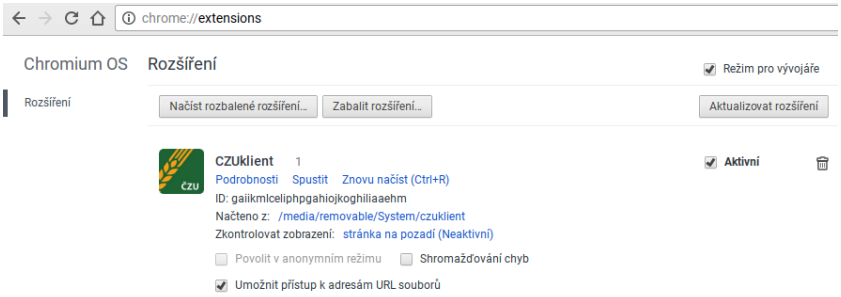
Obrázek č. 29 - Funkce přihlašující Chrome OS klienta k odběru vybraných témat z MQTT Brokeru

Poslední callback handler, funkce onFail(), je přesným opakem předchozí funkce. Nastane v momentě, kdy se klientovi nepodaří navázat spojení s MQTT Brokerem. Funkce vypíše důvod neúspěšného připojení a indikuje ho červenou stavovou ikonou.

K odesílání dat o sepnutí relé slouží funkce relay1function() a relay2function(). Obě jsou analogické. Pro jejich vyvolání jsem vytvořil EventListener, což je metoda, která zavolá příslušnou funkci při kliknutí na tlačítko. V momentě, kdy se tak stane, tak funkce odešle do příslušného vlákna negaci svého aktuálního stavu. Určitě je na místě podotknout, že klient změní stav relé až poté, co ho obdrží potvrzený od HW čidla. Tím se zamezí nekonzistenci mezi klientem a HW čidlem v případě nedoručení příkazu.

4.6.4 Instalace aplikace do Chromium OS

Instalaci vytvořené aplikace do operačního systému Chromium OS jsem provedl v takzvaném vývojářském režimu, abych ji nemusel publikovat veřejně. Jako první krok je potřeba tento režim povolit. To je možné provést v prohlížeči Chromium, konkrétně v modulu „rozšíření“, dostupném na URL chrome://extensions. Po zaškrtnutí políčka „Režim pro vývojáře“ se zobrazí volba „Načíst rozbalené rozšíření“. Pomocí této volby již stačí jen ukázat do kořene struktury aplikace, jak jsem jí popsal v bodě 4.6.1, a aplikace se nainstaluje.

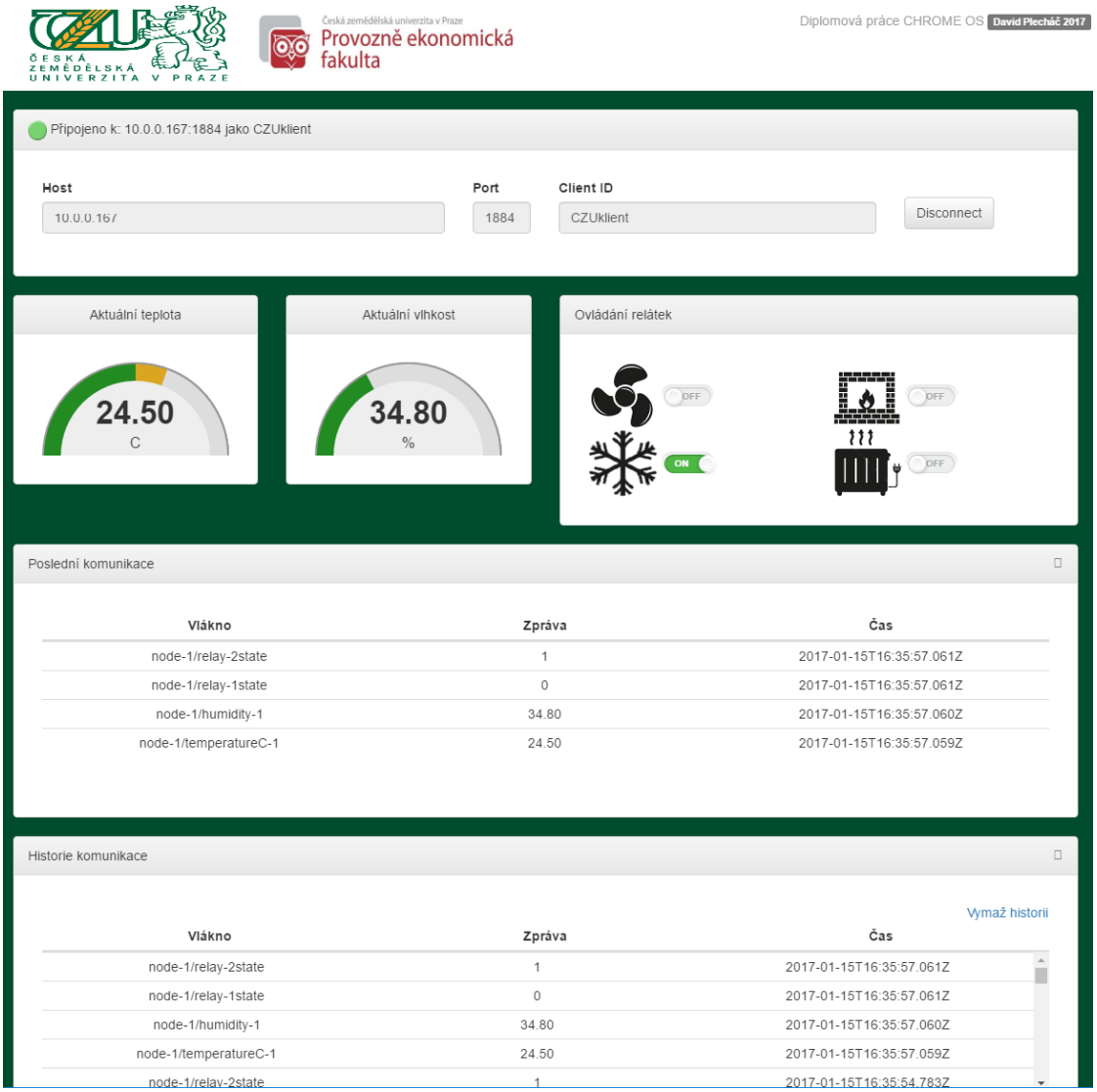


Obrázek č. 30 - Nainstalovaná aplikace v Chromium OS

Po úspěšné instalaci můžeme aplikaci spustit buď volbou „Spustit“, nebo pomocí ikony na hlavní listě, takzvané „poličce“.

Obrázek č. 31 - Ikona aplikace pro rychlé spuštění na hlavní liště Chromium OS

Na obrázku 32 je spuštěná aplikace. Aplikace je připojená k MQTT Brokeru, což signalizuje první formulář pomocí zeleného puntíku a hlášky „připojeno k:“. Ihned po připojení začne aplikace přijímat naměřená data z HW čidla, ty pak graficky znázorňuje. Ve formuláři s ovládacími tlačítky je sepnuté druhé relé. V tabulce „Poslední komunikace“ jsou poslední nejaktuálnější zprávy ze všech témat odebíraných z MQTT Brokera. V poslední tabulce „Historie komunikace“ je zobrazen výpis všech zpráv ze všech odebíraných témat, v pořadí ve kterém zprávy přišly.



Obrázek č. 32 - Spuštěná Chrome OS aplikace po připojení k MQTT Brokeru

5 Výsledky a diskuse

5.1 MQTT Broker

MQTT Broker tak, jak jsem ho nakonfiguroval v kapitole 4.3, fungoval pro účely této práce skvěle. Jediným citelnějším problémem byla instalace podpory WebSockets pro spojení s Chrome OS aplikací. Bez úspěchu jsem vyzkoušel několik různých postupů, ale až tento, který zde popisuji, byl zcela legitimní. Pro účely práce jsem si vystačil se základním nastavením. MQTT Broker Mosquitto nabízí i další rozsáhlé možnosti konfigurace, které by řešily konkrétní funkční požadavky, ale dle mého názoru jsou již mimo rámec této práce.

Pokud by byla aplikace používána přes internet, zcela jistě by bylo nutné implementovat autentizaci a šifrování SSL/TSL. Provedl jsem sérii penetračních testů a zjistil jsem, že přenos dat je bez šifrování zcela nezabezpečený a že probíhá výhradně v plaintextu. Sledováním TCP streamu jsem byl schopen celou komunikaci odchytávat. Domnělý hacker s touto znalostí by mohl jednoduše simulovat ovládací příkazy Chrome OS klienta a spínat silnoproudá relé, nebo jinak škodit, což by mohlo být značně nebezpečné.

Obrázek č. 33 - Ukázka nezabezpečené komunikace pomocí sledování TCP streamu v programu Wireshark

5.2 Hardwarové čidlo

Hardwarové čidlo pracovalo spolehlivě asi měsíc v kuse bez jakéhokoli problému. Používal jsem ho ke spínání pokojové lampy a pásku s LED Diodami. Také bezpečně měřilo teplotu v pokoji, ve kterém bylo umístěno. Vzhledem k tomu, že jsem použil levná čínská relé, určitě bych nedoporučoval vytěžovat je víc než na polovinu jejich nominálního výkonu v odporové zátěži. Nejlepším řešením by bylo, aby stávající 5V relé spínalo stykač a až ten aby spínal silnoproudé zařízení. Drobná nevýhoda současného řešení je, že si zařízení nepamatuje a neobnoví svůj aktuální stav v případě výpadku proudu. To by se dalo vyřešit ukládáním aktuálního stavu a jeho následného obnovení z microSD karty nebo EPROM paměti Arduina. Dalšími vylepšeními by mohl být čistě bezdrátový přenos v podobě nahrazení ethernet shieldu bezdrátovou WiFi kartou a připojením alkalických baterií jako napájecího zdroje. Dále by byla nasnadě možnost připojit další periferie jako hardwarová tlačítka nebo LCD display.

5.3 Instalace operačního systému Chromium OS

Kompilace operačního systému je velmi komplexní záležitost. Jak z hlediska výpočetní náročnosti, tak nároků na připojení k internetu. Operační systém Chromium OS v podobě, v jaké jsem ho zkompiloval, byl naprosto dostatečný pro otestování funkčnosti

aplikace a její používání. První problém, na který jsem narazil, byl po probuzení systému z režimu spánku, poté se systém stal místy nestabilní a bylo nutné ho restartovat. To přisuzuji tomu, že jsem ke kompilaci použil nejnovější rozpracované zdrojové kódy. Na místě by tedy bylo použití nějaké starší, stabilnější verze. Druhým problémem se ukázala nefunkčnost bezdrátové síťové karty. Řešením by pravděpodobně bylo přeprogramování dostupného linuxového ovladače karty pro potřeby Chromimum OS a jeho zahrnutí do kompilace.

5.4 Aplikace pro Chrome OS

Aplikaci tenkého klienta pro Chrome OS jsem vytvořil pomocí technologií a specifik, které tento operační systém vyžaduje. Pro aplikaci jsem vytvořil uživatelské grafické rozhraní, stylizované do barev ČZU. Aplikace splňuje bezpečnostní požadavky pro Chrome OS aplikaci (Content Security Policy) a je připravena pro distribuci do Internetového obchodu Chrome.

Námětem pro další vylepšení aplikace by mohla být například propracovaná práce s daty. Tím myslím jejich ukládání nebo načítání z databáze, grafování, nebo také jejich vyhodnocování. Spínání relé by mohlo být podmíněno právě výstupu práce s daty nebo například časovači. Za zmínku by stálo zatím experimentální použití API pro spuštění Chrome OS aplikací na mobilním operačním systému Android.

6 Závěr

Na základě stanovených cílů jsem vyhodnotil charakteristiky, možnosti vývoje, distribuce a monetizace operačního systému Google Chrome OS a aplikací pro něj. Pomocí syntézy všech poznatků jsem určil vhodné nasazení operačního systému na trhu výpočetní techniky.

Společnosti Google se podařilo vytvořit platformu nabízející opravdu komplexní služby. Všechny tyto služby jsou dostupné v rámci webového prohlížeče, s jehož uživatelským účtem jsou do jisté míry svázané. Tento trend je patrný plošně u veškeré nabídky moderního software. Většina nového software nabízí ovládací rozhraní ve webovém prohlížeči. Díky tomuto faktu se pro většinu uživatelů změnilo používání počítače jako takové. Ve většině případů uživatelé, po nastartování operačního systému, využívají pouze webový prohlížeč ke konzumaci online obsahu. Pro drtivou většinu těchto uživatelů je robustní operační systém, jako například Microsoft Windows, zcela zbytečnou záležitostí. Této skutečnosti společnost Google využila a vytvořila operační systém Chrome OS, jehož hlavní komponentou je právě jejich webový prohlížeč Chrome. Tento operační systém je distribuován spolu s hardwarem jako OEM. Tento hardware se vyznačuje velmi dobrou hardwarovou výbavou a velmi nízkou cenou, až třikrát nižší oproti konkurenci.

Celý operační systém je silně cloudově orientovaný. Nedisponuje tudíž velkým lokálním výpočetním výkonem. Není také kompatibilní s ostatními platformami a nelze tedy na něm spouštět jiné aplikace než pro něj přímo vytvořené nebo webové. Tento problém je možné řešit pomocí virtualizace. Velké virtualizační platformy jako Citrix a VMware vyvinuly klienty pro Chrome OS zajišťující vzdálený přístup k virtuálním strojům, na kterých je možno provozovat libovolnou platformu s potřebnými aplikacemi. Chrome OS je nadále ignorován společností Microsoft a jejich virtualizační platformou Hyper-V. Dokonce neexistuje ani oficiální Chrome OS klient podporující protokol RDP, využívaný k vzdálené správně Windows. Tuto nepodporu od výrobce platformy je možno řešit pomocí podporovaného software třetích stran, například společnosti TeamViewer.

Kromě virtualizace zajišťující nezávislost na platformě je, hlavně pro firemní použití, nutná centralizovaná správa organizační struktury. Doménu lze na Chrome platformě spravovat velice efektivně pomocí nástroje Chrome management console. Tento nástroj umožňuje administrátorovi domény široké nastavení bezpečnostních politik,

přístupových práv, instalaci aplikací a distribuci připravených nastavení do počítačů v doméně vybavených Chrome OS. Nevýhodou je nemožnost spravovat zařízení s Chrome OS v doméně existující na jiné platformě, jako například Microsoft Active Directory.

Abych ověřil teoretické poznatky a mohl na nich postavit praktickou část práce, potřeboval jsem funkční stroj s operačním systémem Chrome OS. Jak jsem již popsal, tento systém je distribuován pouze společně s hardwarem. Kupovat chromebook mi nepřišlo příliš akademicky přínosné, proto jsem se rozhodl předvést postup zprovoznění Chromium OS na běžném notebooku. Chromium OS je open-source projekt, který stojí za komerčním operačním systémem Chrome OS. Chromium OS jsem zkompiloval pomocí zdrojových kódů přímo z veřejně dostupných repozitářů společnosti Google. Tímto postupem jsem zjednodušeně demonstroval proces, který musí podstoupit výrobce hardwaru s cílem odladit Chrome OS pro své výrobky.

Na základě teoretických poznatků o vývoji Chrome OS aplikací jsem navrhl a naprogramoval aplikaci tenkého klienta pro ovládání hardwarového čidla přes internet pomocí protokolů TCP/IP. Aplikace je kompletně připravena na distribuci v internetovém obchodu Chrome Web Store, nicméně jejím účelem bylo pouze prakticky prozkoumat možnosti Chrome OS. Aplikaci jsem stylizoval do typických barev univerzity. Hlavní funkcí aplikace je grafické znázornění měřených hodnot teploty a vlhkosti, jež jsou přenášeny z hardwarového čidla. Aplikace obsahuje i grafické rozhraní k vzdálenému spínaní silnoproudých relé na hardwarovém čidlu. Toto čidlo jsem postavil na platformě Arduino pomocí několika čínských součástek. Firmware jsem naprogramoval v jazyce C. Pro výměnu zpráv jsem zvolil komunikační protokol MQTT, ten používá například Facebooková aplikace Messenger. Tím jsem dosáhl velkého uživatelského komfortu. Uživateli mojí aplikace stačí připojit zařízení k internetu a poté ho může ovládat odkudkoli na světě. Provedl jsem vyhodnocení jednotlivých částí praktické práce. Krom návrhů na budoucí vylepšení a zdůraznění úskalí jsem zdůraznil nutnost použití šifrování a autentizace k zajištění bezpečnosti celého řešení.

Operační systém Google Chrome OS si dle mého názoru bude nadále získávat trh s výpočetní technikou. Sám využívám svůj notebook hlavně jako terminál. I pro komerční sféru dává Chrome OS smysl. V době veřejných zakázek orientovaných na nejmenší cenu má obrovskou výhodu v bezkonkurenčně nejnižší pořizovací ceně při zachování srovnatelných parametrů. Mojí vytvořenou aplikací jsem dokázal, že je pro tento systém

možné naprogramovat téměř libovolnou aplikaci. Díky virtualizaci a správě domény může Chrome OS do velké míry zastoupit drtivou většinu kancelářských, netechnologicky orientovaných strojů ve všech úkonech, které jejich uživatelé potřebují.

7 Seznam použitých zdrojů

[1] ANTO, Y. Chrome OS and Secret of Google. Saarbrücken: Lambert Academic Publishing. 2012. 271 pp. ISBN 978-3-659-17127-7.

[2] MILLER, M. My Google Chromebook. Indianapolis, Ind.: Que. 2012. 271 pp. ISBN 07-897-4396-5.

[3] ROOT, G. Cloud Computing with Google Chrome, 1 edition. Seatle:Amazon. 2013. 116 pp. ISBN 978-1483902258.

[4] ŠIKA, M. Virtuální počítač: praktická řešení pro domácí uživatele. Vyd. 1. Brno: Computer Press. 2011. 256 str. ISBN 978-80-251-3334-7.

[5] STROSS, Randall E. Planeta Google: o troufalém plánu jedné firmy organizovat všechno, co známe. Vyd. 1. Brno: Computer Press, 2009, 296 s. ISBN 978-80-251- 2412-3.

[6] BELCHIN, Moises a Patricia JUBERIAS. Web programming with Dart. New York, New York: Apress, 2015. Expert's voice in Web development. ISBN 148420557X.

[7] KARCHŇÁK, Daniel. 15 let Googlu: internetový gigant rok za rokem. Živě.cz [online]. 2013 [cit. 2015-09-28]. Dostupné z: 15 let Googlu: internetový gigant rok za rokem Více na: http://www.zive.cz/clanky/15-let-googluinternetovy-gigant-rok-za-rokem/sc-3-a-170399/

[8] Google Chrome: webový prohlížeč [online]. [cit. 2017-03-20]. Dostupné z: https://www.google.com/chrome/browser/desktop/index.html

[9] MACICH, Jiří. Chromebook a Chrome OS na vlastní kůži: webový prohlížeč [online]. 2013 [cit. 2017-03-20]. Dostupné z: https://www.root.cz/clanky/chromebook-a-chrome-os-na-vlastni-kuzi/?utm_expid=.Ltrejb6WR_K2RfpRj7BD6A.0&utm_referrer=https%3A%2F%2Fwww.root.cz%2Fn%2Fchrome-os%2F%3Fpi%3D3

[10] KASÍK, Pavel. Přichází Chromebook.: Pro studenty za 339 korun měsíčně i s internetem [online]. 2011 [cit. 2017-03-20]. Dostupné z: http://technet.idnes.cz/prichazi-chromebook-pro-studenty-za-339-korun-mesicne-i-s-internetem-1pi-/notebooky.aspx?c=A110511_200725_tech-a-trendy-nb_pka

[11] GOLDMAN, Joshua. Asus Chromebox review: Very good Chrome OS starting point [online]. 2011 [cit. 2017-03-20]. Dostupné z: https://www.cnet.com/products/asus-chromebox/review/

[12] Google Chromecast [online]. [cit. 2017-03-20]. Dostupné z: https://www.google.com/intl/en_us/chromecast/?utm_source=chromecast.com

[13] Test Asus Chromebit CS10: Počítač ako čokoládová tyčinka [online]. 2016 [cit. 2017-03-20]. Dostupné z: http://www.zive.sk/clanok/113172/test-asus-chromebit-cs10-pocitac-ako-cokoladova-tycinka

[14] The Chromium Projects: Chromium OS [online]. [cit. 2017-03-20]. Dostupné z: https://www.chromium.org/chromium-os

[15] Chromebooks for Work: Chromebooks are easy to manage and run top business apps. [online]. [cit. 2017-03-20]. Dostupné z: https://www.google.com/intl/cs/chrome/education/devices/features-management-console.html

[16] Chromebooks for Education: The Chromebook Family. A range of fast, portable computers for students and teachers. [online]. [cit. 2017-03-20]. Dostupné z: https://www.google.com/intl/cs/chrome/education/devices/features-management-console.html

[17] Chrome OS based devices in the Enterprise: Client virtualization [online]. 2015 [cit. 2017-03-20]. Dostupné z: http://www.intel.de/content/dam/www/public/emea/xe/en/pdf/intel-chrome-os-white-paper-2015-edition-final-v1.4.pdf

[18] JavaScript APIs: Stable APIs [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/api_index

[19] Chrome Apps Architecture: Stable APIs [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_architecture

[20] Chrome App Lifecycle: How the lifecycle works [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/app_lifecycle

[21] Content Security Policy [online]. [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/apps/contentSecurityPolicy

[22] Publish in the Chrome Web Store [online]. In: . [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/webstore/publish

[23] Monetizing Your Chrome Web Store Item [online]. In: . [cit. 2017-03-20]. Dostupné z: https://developer.chrome.com/webstore/money

[24] MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. [online]. In: . [cit. 2017-03-20]. Dostupné z: http://mqtt.org/

[25] Eclipse Mosquitto™ is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 [online]. In: . [cit. 2017-03-20]. Dostupné z: https://mosquitto.org/

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

[27] Arduino UNO [online]. In: . [cit. 2017-03-20]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardUno

[28] W5100 Datasheet [online]. In: . [cit. 2017-03-20]. Dostupné z: https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf

[29] Digital-output relative humidity & temperature sensor/module DHT22 [online]. In: . [cit. 2017-03-20]. Dostupné z: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[30] Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. [online]. In: . [cit. 2017-03-20]. Dostupné z: http://getbootstrap.com/

[31] Eclipse Paho JavaScript Client [online]. In: . [cit. 2017-03-20]. Dostupné z: http://www.eclipse.org/paho/clients/js/

[32] LEI, Tze. Jquery-dynameter: A jQuery plugin that transforms your empty DIV into a semi-circular meter with dynamic value-update support. [online]. In: . 2014 [cit. 2017-03-20]. Dostupné z: https://github.com/tlei123/jquery-dynameter

8 Přílohy

[1] DP_David_Plecháč_Chrome_OS_Czuklient [online]. In: . [cit. 2017-03-20]. Dostupné z: https://is.czu.cz

[2] DP_David_Plecháč_Chrome_OS_HWčidlo_firmware [online]. In: . [cit. 2017-03-20]. Dostupné z: https://is.czu.cz