Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Diplomová práce

Nadřazený systém pro správu garáže

Bc. Ondřej Červenka

Vedoucí práce: Ing. Martin Daňhel

30. listopadu 2017

Poděkování

Děkuji panu Ing. Martinu Daňhelovi za čas, který mi věnoval a zejména za cenné rady a odborné vedení mé diplomové práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 30. listopadu 2017 .....

Odkaz na tuto práci

Červenka, Ondřej. *Nadřazený systém pro správu garáže.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Seznam obrázků

Úvod

Cílem této práce je vytvořit nadřazený systém pro monitorování garážového komplexu. Výsledná aplikace bude komunikovat pomocí WiFi či Ethernetu s podřízenými systémy (zasílajícími údaje z čidel v garážích). Na základě získaných dat pak bude udržován stav jednotlivých garáží a vytvářena historii událostí.

Systém bude poskytovat webového rozhraní pro administraci. V tom bude možné přidávat a odebírat podřízené systémy, zobrazovat jejich stav a zaznamenané události.

Vzhledem k povaze zadání je nutné systém navrhnout s ohledem na zabezpečení přenášených informací před odposloucháváním či manipulací. Též je nutné autorizovat uživatele přistupjící do webového rozhraní.

Dalším důležitým požadavkem je snadná rozšiřitelnost o nové funkce. Systém by mělo být možné v budoucnu doplnit o možnost správy rozdílných podřízených systémů (například subsystémy pro sledování skladových zásob) či integraci s mobilní aplikací. Bude tedy potřeba navrhnout vhodné API pro předávání informací mezi systémem a jeho klienty.

V této práci se chci zaměřit na tvorbu aplikace na jedné konkrétní hardwarové platformě, jako například *Raspberry Pi*. Aplikace spolu s touto platformou by pak měla tvořit kompletní zařízení, které bude možné po základní konfiguraci (připojení do WiFi sítě, nastavení hesla) hned nasadit.

Výsledné řešení by však mělo být dostatečne nezávislé na zvolené platformě. Tudíž by neměl být problém spustit systém například na osobním počítači či virtuálním serveru.

V analytické části (1) práce tedy přiblížím proces výběru vhodné platformy, komunikačního protokolu a dalších prvků systému. Také stručně popíšu podřízený systém (garážové čidlo), se kterým budu dále pracovat. Další části mapují návrh (2) zařízení na základě této analýzy, jeho implemetaci (3) a testování (5).

Kapitola 1

Analýza

1.1 Struktura systému

Struktura celého systému je naznačena na obrázku 1.1. Podřízené systémy komunikují s nadřazeným na základě událostí. Nadřazený systém tyto události zpracovává a upravuje podle nich stav garáží v evidenci.

Zaznamenané události jsou také uchovávány v historii událostí, spolu s dalšími metadaty jako čas přijetí nebo původce.

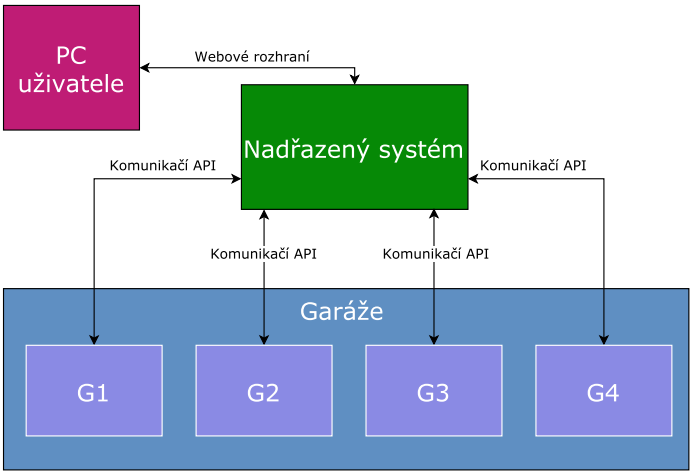
Komunikace mezi podřízeným a nadřazeným systémem je postavena na modelu *client/server*. Nadřazený systém provozuje server zvoleného protokolu (viz sekce 1.2), ke kterému se podřízené systémy připojují. Komunikaci tedy vždy iniciuje podřízený systém. S možností zasílání nevyžádaných zpráv podřízeným systémům v této práci nepočítám, mohl by to však být námět pro další rozšíření.

Další, kdo přistupuje do systému, je uživatel. Přes webové rozhraní může sledovat stav garáží a historii událostí. Také zde může spravovat klíče, které slouží pro přístup ke komunikačnímu API systému. Přístup do webového rozhraní je zabezpečen heslem.

1.1.1 Podřízený systém

Podřízený systém je zařízení umístěné v každé garáži, které sleduje stav okolí pomocí těchto senzorových vstupů:

- teplota,
- světelá intenzita (fotobuňka),
- detekce kouře,
- detekce pohybu,
- stav dveří.



Obrázek 1.1: Základní struktura systému

V případě překročení mezních hodnot se zařízení okamžitě hlásí nadřazenému systému. Kromě toho také v pravidelných intervalech odesílá kontrolní hlášení.

Vyhodnocení události je provedeno nadřazeným systémem. Podřízený systém tedy hlásí každou událost (například otevření dveří), aniž by nějak zkoumal její závažnost.

Základní požadavek na podřízený systém je schopnost komunikace přes Ethernet či WiFi pomocí protokolu zvoleného v sekci 1.2. Kromě toho může být hardware prakticky libovolný.

1.2 Výběr komunikačního protokolu

Nejdřív je nutné určit způsob komunikace, který bude systém používat. Díky tomu se budu při vybíraní platformy moci ujistit, že jsou dostupné vhodné knihovny a další software.

Nadřazený systém bude se svými klienty (monitorovací zařízení v jednotlivých garážích) komunikovat přes WiFi nebo Ethernet. Základem komunikace bude TCP/IP protokol, je však potřeba zvolit vhodný protokol z aplikační vrstvy OSI modelu, který na něm bude stavět.

1.2.1 Vlastní protokol

Jedna z možností je implementovat vlastní protokol pomocí TCP/IP socketů. Toto řešení se mi však nezdá příliš vhodné, neboť nepřináší žádné významné výhody, naopak se s ním pojí řada komplikací.

Pro vlastní protokol by bylo nutné vytvořit robustní server, který zvládá obsluhu více klientů najednou. Dále by vzhledem k citlivosti přenášených dat

bylo nutné implementovat nějakou formu šifrování. Tyto velmi obsáhlé problémy přitom řeší většina dnešních protokolů.

Další nevýhodou je nutnost implementace klientské části protokolu při vytváření nových zařízení spravovaných nadřazeným systémem. To do jisté míry omezuje jeho rozšiřitelnost.

1.2.2 HTTPS

Další možnost je využít ke komunikaci protokol HTTPS. V tomto případě by klienti komunikovali se sytémem pomocí HTTP metod jako například `get` nebo `post`.

Jelikož součástí požadavků na systém je i webové uživatelské rozhraní, bude v každém případě nutné použít webový server pro jeho provoz. Ten by pak bylo možné využít i k poskytnutí API pro komunikaci systému s garážovými čidly.

Vhodný webový server (jako například *Nginx*) zajistí vícevláknovou obsluhu všech klientů. Protokol se také postará o kryptografické zabezpeční přenášených dat, je však nutné získat certifikát k ověření pravosti serveru (viz sekci 1.2.2.1).

Certifikát bude potřeba zajistit i v případě, že komunikace s klienty nebude postavena na tomto protokolu. Je totiž nutné také zabezpečit webové rozhraní, například kvůli ověření identity uživatele. Nutnost pořízení certifikátu tedy nepředstavuje nevýhodu oproti jiným protokolům.

API realizované pomocí tohoto protokolu je poměrně snadno rozšiřitelné. Pro nově implementovanou operaci stačí definovat URL a případně formát přenášených dat.

Výhodou je také snadná implementace na straně klienta, tedy garážového čidla. Knihovny realizující klientskou část protokolu jsou dostupné na většině populárních platforem jako například *Arduino* (s Ethernet shieldem, oficiální knihovna *EthernetClient* [1]) nebo *ESP8266* (knihovna *esp8266wifi* [2]).

1.2.2.1 Certifikáty pro provoz HTTPS

Pro provoz HTTPS serveru lze použít například certifikáty certifikační autority Let's Encrypt, které jsou poskytovány zdarma. Kromě toho dodává Let's Encrypt také automatizačního klienta *Certbot* [3] pro snadné nasazení a aktualizaci jejich certifikátů. Bohužel certifikáty jsou vydávány pouze na doménu [4], což komplikuje použití v místní síti.

Jiná možnost je použití *self-signed* certifikátu. Tento certifikát není podepsaný žádnou certifikační autoritou, ale pouze vlastníkem certifikátu. Může tedy sloužit k šifrování komunikace (poskytuje veřejný klíč), ale je zranitelný vůči *man-in-the-middle* útoku [5].

Self-signed certifiát však lze použít k šifrování komunikace na uzavřené lokální síti, za předpokladu, že je server s certifikátem (přesněji s jeho soukromým klíčem) dostatečně zabezpečen [5].

Nevýhodou tohoto řešení je nedůvěra webových klientů (certifikát není podepsán certifikační autoritou), což by ovlivnilo přístup k uživatelskému rozhraní a API systému. V případě webového rozhraní by prohlížeč zobrazil varování o neznámém certifikátu. To by však mohl uživatel ignorovat.

Podřízené systémy by při zasílání požadavků museli přeskočit krok ověření totožnosti serveru. Jak toho dosáhnout v knihovně *Requests* pro Python (dostupné i pro *Raspberry Pi*), je naznačeno v ukázce 1.

```
>>> import requests
>>> r = requests.get('https://testserver/test', verify=False)
>>> r.status_code
200
```

Ukázka 1: Vytvoření HTTPS požadavku v knihovně *Requests*, bez verifikace serveru

1.2.2.2 Autentizace klientů na HTTPS

Přístup k API nadřazeného systému by měl být povolen pouze ověřeným klientům. Díky tomu bude možné zabránit například zasílání nepravdivých informací z neznámých zdrojů.

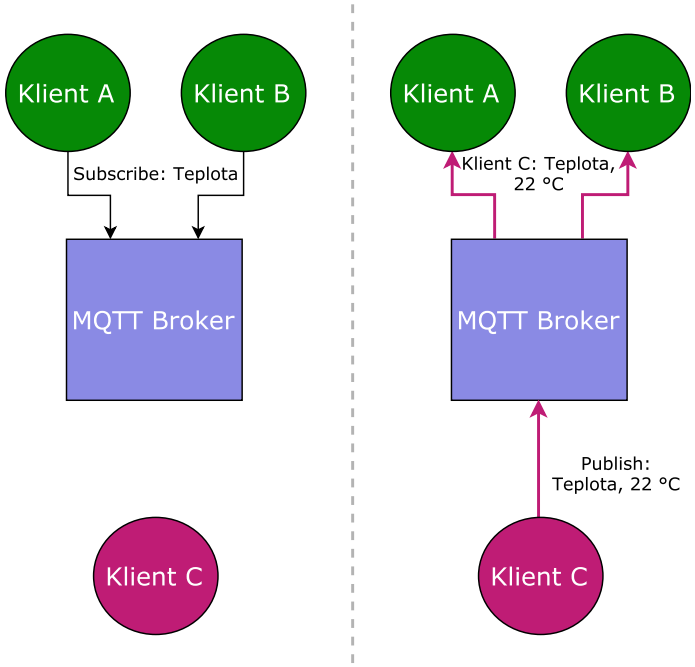
Jednoduchou autentizaci přes HTTPS lze realizovat například pomocí generování API klíčů. Pro každý podřízený systém bude vygenerován klíč, kterým se při zasílání požadavku systém prokáže. Seznam platných klíčů by byl udržován v databázi nadřazeného systému. Klíče by uživatel mohl přidávat nebo odebírat (například v případě odcizení podřízeného systému) pomocí webového rozhraní.

Tyto klíče by také bylo nutné nahrát a uchovávat na podřízených systémech. Detaily tohoto procesu by záležely na platformě těchto systému. Například u *Arduina* by šlo klíč nahrát z uživatelova počítače pomocí sériové linky (s USB převodníkem) a udržovat ho v EEPROM.

1.2.3 MQTT

MQTT je komunikační protokol založený na modelu *publisher/subscriber*, vhodný pro použití v prostředí s omezenými zdroji (malý výkon procesoru, omezená paměť atd.) [6].

Komunikace mezi jednotlivými klienty v systému je zprostředkována pomocí centrály, nazývané *broker*. Ta spravuje adresy – *topics* – na kterých mohou klienti publikovat či odebírat zprávy.



Obrázek 1.2: Příklad struktury protokolu MQTT [7]

Na obrázku 1.2.3 tedy klienti A a B začnou odebírat (*subscribe*) *topic* „Teplota“. Když pak klient C pak publikuje zprávu na tuto adresu, *broker* se postará o doručení všem odebírajícím klientům.

Adresy je možné hierarchicky strukturovat. Lze tedy tvořit skupiny, například `/senzory/obyvak/teplota` nebo `/senzory/kuchyne/vlhkost`. Zprávy je nutné publikovat na jednoznačnou adresu, při odebírání je však možné použít modifikátory `+` a `#` pro specifikování skupiny adres. Modifikátor `+` odpovídá libovolnému jednomu stupni hierarchie, `#` pak libovolnému počtu libovolných stupňů. Pro odebírání všech senzorů vlhkosti lze tedy použít adresu `/senzory/+/vlhkost`. Všechna data by pak bylo možné odebírat na adrese `/senzory/#`. [7]

V případě této práce by tedy jak nadřazený systém, tak podřízené systémy byly klienty *brokeru*. Podřízené systémy by publikovali naměřená data, která by nadřazený systém odebíral. Samotný *broker* by pak mohl běžet souběžně s nadřazeným systémem na zvolené platformě (například open-source *broker* *Mosquitto* je dostupný na řadě platforem, včetně *Raspberry Pi* [8]).

Protokol podporuje tři možnosti QoS (*Quality of Service*) [6]:

- Nejvýše jedno doručení – tento mód pouze odešle zprávu, není zahrnut žádný opakovací mechanismus pro případ nedoručení.
- Alespoň jedno doručení – v tomto módu je zaručeno doručení zprávy, ta však může být doručena vícekrát.
- Přesně jedno doručení – zde je ošetřeno i duplicitní doručování zpráv.

Použití sofistikovanějších metod doručení má vliv na výkon, a proto se v některých případech vyplatí zvolit nižší úroveň QoS (například při posílání idempotentních zpráv). Pro tuto práci bych však pravděpodobně zvolil záruku přesně jednoho doručení.

1.2.3.1 Šifrování a autentizace na MQTT

V této části se budu zabývat prostředky pro šifrování komunikace, které jsou dostupné v *brokeru Mosquitto*.

První možnost je pro zabezpečení komunikace využít certifikáty, podobně jako u HTTPS. Zde by se pravděpodobně také využil *self-signed* certifikát (blíže popsaný v sekci 1.2.2.1). *Mosquitto* navíc také vyžaduje kořenový certifikát certifikační autority [9]. Při použití *self-signed* ceritifikátů by bylo nutné tuto autoritu vytvořit a používané certifikáty u ní podepsat (pro bližší informace viz [10]). Kořenový certifikát by také bylo nutné distribuovat klientům.

Kromě certifikátů lze pro šifrování použít i PSK (*pre-shared key*). V tom případě *broker* a jeho klienti používají pro zašifrování komunikace společný klíč (známý jak klientovi, tak *brokeru*). Různí klienti přitom mohou mít různé klíče. [11]

Bohužel podpora PSK v MQTT klientech není příliš rozšířená. PSK je možné použít v knihovně *libmosquitto*, určené pro C/C++ (s vazbami pro Python). U této knihovny se mi však podařilo najít pouze manuálovou stránku (viz [12]), bez informací o jejím dalším vývoji či udržování. Modul poskytující vazby do Pythonu byl nicméně předán projektu *Paho* [13].

Paho poskytuje implementace MQTT klientů pro mnoho platforem (včetně například *Arduina* [14]). Dokumentace klientů pro C++ a Python však možnost šifrování pomocí PSK vůbec nezmiňuje [15] [16].

Tyto možnosti lze použít i k autentizaci klientů *brokeru*. Při použití certifikátů lze v konfiguračním souboru *Mosquitta* zvolit `require_certificate`. Poté bude od klienta vyžadován certifikát prokazující jeho totožnost. Při použití PSK lze k autentizaci využít sdílený klíč (*broker* odmítne klienty s neplatnými klíči). Kromě toho je možno použít také autentizaci pomocí uživatelského jména a hesla, která je součástí MQTT protokolu, případně klienty neověřovat vůbec (a pouze šifrovat komunikaci). [11]

1.2.4 Závěr výběru protokolu

V sekcích 1.2.2 a 1.2.3 jsem se blíže podíval na dva poměrně rozšířené protokoly aplikační vrstvy, které by bylo možné použít pro tvorbu nadřazeného systému.

Pokud by mezi požadavky na systém bylo zahrnuto zasílání nevyžádaných zpráv podřízeným systému (jak je zmíněno v sekci 1.1), zvolil bych pravděpodobně protokol MQTT. V tom je tato funkcionalita velmi snadno implementovatelná – stačí aby podřízené systémy odebíraly *topic*, na kterém by nadřazený systém publikoval zprávy.

Jelikož se však v této práci zabývám systémem, který zprávy pouze přijímá a zaznamenává, rozhodl jsem se pro HTTPS. Nasazení tohoto protokolu je o něco snazší (není potřeba na zařízení instalovat *broker* a zařizovat certifikační autoritu – stačí *self-signed* certifikát) a s jeho použitím mám více zkušeností. Také částečně uvolní požadavky na volbu platformy (webový server bude potřeba v každém případě, při volbě HTTPS jako komunikačního protokolu mezi systémy tedy nebude nutný žádný další software).

Každopádně bude mým cílem navrhnout výslednou aplikaci tak, aby rozhraní pro podřízené systémy realizované pomocí HTTPS bylo možné snadno nahradit MQTT rozhraním.

K zabezpečení komunikace (včetně webového rozhraní) použiju *self-signed* certifikát. Hlavní důvod je požadavek na použití v místní síti, bez zaručeného přístupu k internetu. Toto rozhodnutí nemá vliv na návrh a implementaci systému, pouze na jeho nasazení.

Pokud má provozovatel systému k dispozici doménu a veřejnou IP, může k zabezpečení použít certifikát podepsaný důvěryhodnou autoritou (vázaný na doménu). Stačí pouze v konfiguračním souboru webového serveru nahradit *self-signed* certifikát podepsaným certifikátem. Není tedy nutné provádět změny v kódu aplikace.

1.3 Ukládání dat

1.4 Výběr platformy

Pro realizaci systému je nutné zvolit vhodnou platformu. Jelikož je cílem práce vytvořit fyzické zařízení, rozhodl jsem se jako základ použít některý z jednodeskových počítačů, které jsou v dnešní době na trhu. Tyto počítače bývají cenově velmi dostupné a zároveň poskytují dostatečný výkon a podporu pro provoz systému.

Při výběru počítače byla nejdůležitejším kritériem podpora softwaru potřebného k implementaci monitorovacího systému.

Kapitola 2

Návrh

Kapitola 3

Implementace

```
# ... code here ...

import numpy as np

def foo(a):
    print(a)

class FooBar:
    def __init__(self):
        self.b = 10

a = [1, 2, 3, 4]
for i in a:
    if i == 2:
        print("hello world")
```

Ukázka 2: Testovací listing

Kapitola 4

Nasazení

tady bude neco vo nasazovani na RPI (tj rozjet apache, vygenerovat certifikaty atd., viz `https://github.com/ggljzr/mi-dip-impl/tree/master/deployment`)

Kapitola 5

Testování

Závěr

Literatura

[1] Arduino: Web Client. 2015, [cit. 2017-10-25]. Dostupné z: https://www.arduino.cc/en/Tutorial/WebClient

[2] Grokhotkov, I.: esp8266wifi – Client Example. 2017, [cit. 2017-10-25]. Dostupné z: https://github.com/esp8266/Arduino/blob/master/doc/esp8266wifi/client-examples.rst

[3] Electronic Frontier Foundation: Certbot – About. [cit. 2017-10-18]. Dostupné z: https://certbot.eff.org/about/

[4] Electronic Frontier Foundation: Let's Encrypt – Frequently Asked Questions. 2017, [cit. 2017-11-07]. Dostupné z: https://letsencrypt.org/docs/faq/

[5] Wallen, J.: When are self-signed certificates acceptable for businesses? 2017, [cit. 2017-11-08]. Dostupné z: https://www.techrepublic.com/article/when-are-self-signed-certificates-acceptable-for-businesses/

[6] Lampkin, V.: What is MQTT and how does it work with WebSphere MQ? 2012, [cit. 2017-10-25]. Dostupné z: https://www.ibm.com/developerworks/mydeveloperworks/blogs/aimsupport/entry/what_is_mqtt_and_how_does_it_work_with_websphere_mq?lang=en

[7] Jaffey, T.: MQTT and CoAP, IoT Protocols. 2014, [cit. 2017-11-12]. Dostupné z: https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

[8] Newsom, C.: Mosquitto Message Broker. 2016, [cit. 2017-11-16]. Dostupné z: https://github.com/mqtt/mqtt.github.io/wiki/Mosquitto-Message-Broker

[9] Light, R.: mosquitto.tls – Mosquitto Manual. [cit. 2017-11-20]. Dostupné z: https://mosquitto.org/man/mosquitto-tls-7.html

[10] Nguyen, J.: OpenSSL Certificate Authority. 2015, [cit. 2017-11-20]. Dostupné z: https://jamielinux.com/docs/openssl-certificate-authority/

[11] Light, R.: mosquitto.conf – Mosquitto Manual. [cit. 2017-11-20]. Dostupné z: https://mosquitto.org/man/mosquitto-conf-5.html

[12] Light, R.: libmosquitto – Mosquitto Manual. [cit. 2017-11-21]. Dostupné z: https://mosquitto.org/man/libmosquitto-3.html

[13] Eclipse Foundation: Python – Mosquitto Documentation. [cit. 2017-11-21]. Dostupné z: https://mosquitto.org/documentation/python/

[14] Eclipse Foundation: Embedded MQTT C/C++ Client Libraries. [cit. 2017-11-21]. Dostupné z: http://www.eclipse.org/paho/clients/c/embedded/

[15] Eclipse Foundation: Paho C++ Documentation. [cit. 2017-11-21]. Dostupné z: http://www.eclipse.org/paho/files/cppdoc/index.html

[16] Eclipse Foundation: Paho Python Documentation. [cit. 2017-11-21]. Dostupné z: https://pypi.python.org/pypi/paho-mqtt

Příloha **A**

Seznam použitých zkratek

API

EEPROM

IP

HTTP Graphical user interface

HTTPS Graphical user interface

MQTT Graphical user interface

OSI

PSK

QoS

TCP/IP Graphical user interface

URL

USB

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS