VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 Luděk Páleník

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

VIRTUALIZACE ARCHITEKTURY ARM A VÝVOJ UNIVERZÁLNÍ ŘÍDICÍ PLATFORMY PRO EMBEDDED SYSTÉMY

VIRTUALIZATION OF ARM ARCHITECTURE AND DEVELOPMENT OF A UNIVERSAL CONTROL PLATFORM FOR EMBEDDED SYSTEMS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Luděk Páleník**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Kryštof Zeman**

SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Luděk Páleník ***ID:*** 138004
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Virtualizace architektury ARM a vývoj univerzální řídicí platformy pro embedded systémy

POKYNY PRO VYPRACOVÁNÍ:

V současné době probíhá řada diskuzí ohledně zpracování dat získaných z měřicích zařízení v rámci platformy zvané Internet věcí (Internet of Things). Jako možné řešení pro jednotky zpracovávající M2M (Machine-t--Machine) data se dnes jeví nízko-výkonové (embedded) zařízení, které umožňují běh Unixových operačních systémů. Vývojové kity těchto zařízení jsou však i dnes velmi drahé a proto se vývoj provádí ve virtualizovaných prostředích. V této bakalářské práci bude proto pozornost zaměřena virtualizaci prostředí ARM a na vývoj softwarového řešení (univerzální řídící platformy) zpracovávající přijatá data od senzorů / měřicích zařízení (s využitím protokolů Modbus, RS-485 a Protocol 104). Řešení bude realizováno v systému Linux, kde bude pomocí virtualizačního nástroje QEMU emulováno prostředí ARM, ve kterém bude v programovacím jazyce C/C++ realizována řídící platforma. Na závěr bude pro ověření výsledků použit vývojový kit CM-T43.

DOPORUČENÁ LITERATURA:

[1] PORTNOY, Matthew. 2012. Virtualization essentials. Indiana: Wiley. ISBN 978-1-118-17671-9

[2] LANGBRIDGE, James A. 2014. Professional embedded ARM development. Indiana: Wiley. ISBN 978-1-1-8-78894-3.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. Kryštof Zeman
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Bakalářská práce „Virtualizace architektury ARM a vývoj univerzální řídicí platformy pro embedded systémy“ se zabývá teoretickým rozborem IoT zařízení, M2M komunikací a virtualizačními technikami – zejména emulací různých architektur procesorů. Praktická část práce je zaměřena na implementaci upraveného emulovaného obrazu na modul CM-T43 od společnosti Texas Instruments. Pro emulaci je využit emulační nástroj QEMU instalovaný v prostředí operačního systému Linux. Dále se práce zabývá vývojem aplikace, která umožňuje čtení a zápis dat z/do jednotlivých čidel na bázi sériové linky (RS-232, RS-485) a Ethernetu. Aplikace je napsána v jazyce C a využívá protokol MODBUS.

KLÍČOVÁ SLOVA

IoT, M2M komunikace, QEMU, jazyk C, MODBUS, RS-485

ABSTRACT

Bachelor thesis "Virtualization of ARM architecture and development of a universal control platform for embedded systems" deals with theoretical analysis of IoT devices, M2M communications and virtualization techniques – especially emulation of different processor architectures. The practical part is implementation of the modified emulated image for CM-T43 module from Texas Instruments manufacturer. QEMU emulation tool installed on Linux operating system is used for emulation. Another part of this work is to develop an application that allows reading and writing of data from / to each sensor based on a serial line (RS-232, RS-485) and Ethernet. The application is written in language C and uses the MODBUS protocol.

KEYWORDS

IoT, M2M communication, QEMU, C language, MODBUS, RS-485

PÁLENÍK, Luděk *Virtualizace architektury ARM a vývoj univerzální řídicí platformy pro embedded systémy*: bakalářská práce. Místo: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 78 s. Vedoucí práce byl Ing. Kryštof Zeman

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Virtualizace architektury ARM a vývoj univerzální řídicí platformy pro embedded systémy“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Místo ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Kryštofu Zemanovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Místo ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Místo . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato bakalářská práce se věnuje Internetu věcí (IoT, Internet of Things), M2M (Machine-to-Machine) komunikaci, emulaci hardwaru a vývoji aplikace, která má sloužit pro komunikaci s čidly/senzory pracující na sériové lince či Ethernetu.

První kapitola pojednává o Internetu věcí. Termín Internet věcí popisuje zařízení, která v první řadě zpracovávají (přijímají) data v lokální síti, ale pro splnění podstaty IoT je data nutné distribuovat (odeslat) na centrální server, který je ve většině případů umístěn v Internetu (komunikace např. přes mobilní sítě LTE). Tato zařízení (často označovaná jako embedded/vestavěná) jsou dnes využívána v domácnostech, v průmyslu a v mnoha dalších odvětvích (elektronické zdravotnictví, logistika a jiné). Jejich počet vzrůstá rapidním tempem, zejména díky komfortu, který kvůli možné vzdálené správě či monitoringu nabízejí. Dále se tato kapitola věnuje komunikačním protokolům: CoAP (Constrained Application Protocol), SIP (Session Initiation Protocol), MQTT (Message Queue Telemetry Transport), AMQP (Advanced Message Queuing Protocol), DDS (Data Distribution Service) a MODBUS, které jsou pro IoT využívány.

Různá IoT zařízení mezi sebou mohou komunikovat, a tak vzniká nový pojem M2M komunikace. Tomuto pojmu se věnuje druhá kapitola. Popisuje zařízení, která jsou schopná komunikovat bez účasti člověka. Kapitola se také věnuje srovnání M2M komunikace a klasické H2H (Human-to-Human) komunikace. M2M lze vyžít k automatizaci a řízení procesů. S objemným nástupem M2M zařízení však nastává problém – zařízení zasílají data v intervalech, což může vést ke komplikacím v podobě nárazového zatížení sítě (zejména z pohledu přístupové rádiové části dnešních mobilních sítí).

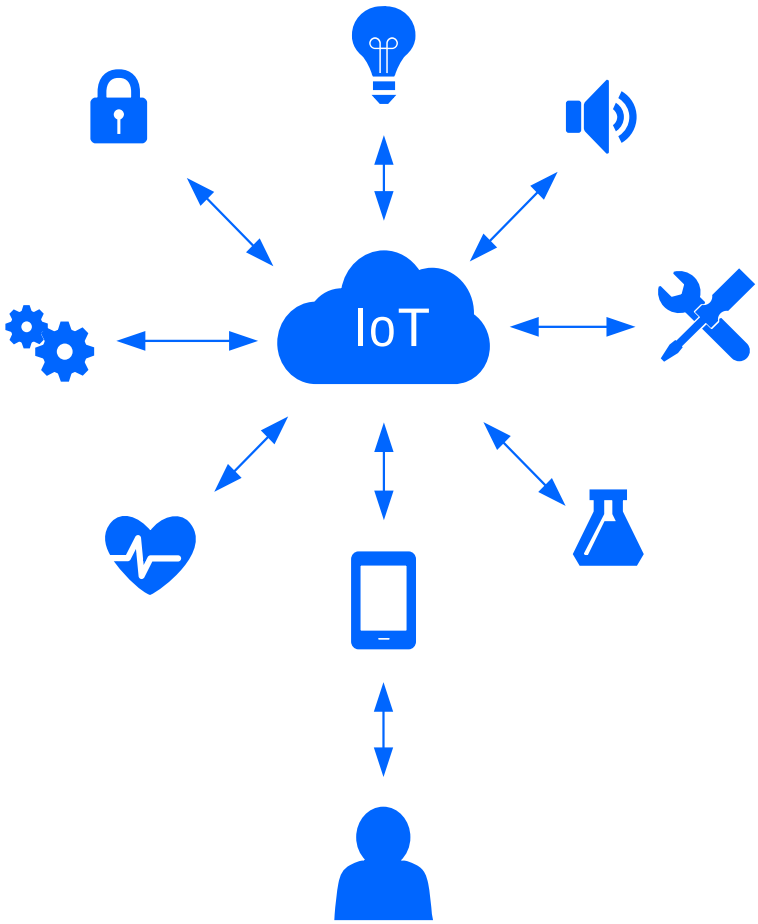
Vzhledem k nízkému výkonu zařízení a jejich počtu je lepší vývoj na emulovaném zařízení a následná distribuce do koncových zařízení. Virtualizačním technikám, typům virtualizace a emulaci se věnuje kapitola třetí.

Čtvrtá kapitola detailně popisuje emulátor QEMU, který byl pro projekt zvolen pro jeho schopnost emulovat širokou škálu procesorových architektur díky implementovanému křížovému překladači.

Pátá kapitola popisuje praktickou realizaci této bakalářské práce. Nejdříve je popsána instalace QEMU a emulace obrazu pro zvolený procesor, dále důležité části kódu aplikace mb_master, která vychází z protokolu MODBUS a slouží k zápisu a čtení dat z jednotlivých čidel/senzorů. Vytvořená aplikace pak umožňuje realizaci „IoT scénáře“, a to komunikaci od senzoru až směrem k vzdálenému serveru.

1 INTERNET OF THINGS

Internet of Things (IoT), česky Internet věcí je termín, který se obecně používá pro označení vestavěného (embedded) zařízení připojeného k Internetu pomocí kabelového, WLAN (Wireless Local Area Network) bezdrátového připojení, nebo datového připojení mobilní sítě. Specifickou vlastností těchto zařízení je schopnost prostřednictvím Internetu sdílet data. Počet fyzických objektů připojených k Internetu vzrůstá v dnešní době nevídaným tempem. Příkladem takových objektů může být systém HVAC (Heating, Ventilating and Air Conditioning; česky vytápění, ventilace a klimatizace), sloužící k monitorování a kontrole systémů v chytrých domácnostech. Další oblasti, kde lze IoT využít jsou např. doprava, zdravotnictví, průmyslová automatizace či nouzové reakce na přírodní nebo člověkem způsobené katastrofy, kdy je složité situaci vyhodnotit lidským faktorem [1], [2].



Obr. 1.1: Možnosti IoT

Na obrázku 1.1 lze vidět, kam až dnes IoT sahá. Nejvýše v hierarchii je člověk, který vlastní komunikační zařízení (chytrý telefon, tablet, laptop). Pomocí Internetu je možné vzdáleně monitorovat zdravotní stav pacienta, uzavřít průtok vody do bazénu, pokud čidlo zjistí kalnou vodu, vzdáleně nastavit pračku nebo doma v pracovně vypnout světlo.

1.1 Historie

Určitým předchůdcem IoT zařízení byl systém SCADA (Supervisory Control And Data Acquisition; česky dispečerské řízení a sběr dat). Nejednalo se o úplný řídící systém, ale o systém s „dohledem“. První generace systému byla vyvinuta v 60. letech 20. století pro napájecí systémy a využívala centralizované výpočetní systémy (hlavní počítač byl připojen do všech koncových jednotek). Druhá generace v 90. letech změnila a modernizovala architekturu. Byl zaveden otevřený distribuovaný systém, který umožnil rozeslat jeden příkaz na více stanic. Zavedením lokální sítě LAN (Local Area Network), a tudíž i síťové vrstvy a protokolu TCP/IP, bylo možné jednotlivá zařízení sledovat a řídit na interní síti. 3. generace umožňovala připojení k Internetu a plnou integraci firemní sítě [3], [4], [5].

První koncept IoT sahá do 80. let 20. století. V roce 1982 byl představen návrh „chytrého“ zařízení, které bylo připojeno k Internetu a kontrolovalo množství nápojů na skladě a jejich výstupní teplotu [6].

V roce 1990 byl k Internetu pomocí PC (Personal Computer) připojen toustovač, který bylo možné vzdáleně vypnout či zapnout [7].

Největší průlom IoT činí rok 1999, kdy byl poprvé představen termín „Internet of Things“. Bylo to na prezentaci Kevina Ashtona na MIT (Massachusetts Institute of Technology) ve Spojených státech amerických. Ashton představil projekt Auto–ID Labs, což byl systém na sledování zboží pomocí štítku s jednoznačným RFID (Radio Frequency Identification) identifikátorem, umístěným na dané položce. Pomocí Internetu bylo možné pohyb zboží sledovat [7], [8], [9].

Kromě RFID se začal využívat standard IEEE 802.11 (Wi-Fi). Ten umožnil bezdrátové připojení objektů, snížily se náklady za použité technologie a montáž.

V 70. letech se začala vyvíjet technologie čárkových kódů a QR (Quick Response) kódů. Tyto technologie nám dnes umožňují díky různým aplikacím pro mobilní telefony či tablety kód načíst a zobrazit například detailní informace o produktu.

Později vznikly technologie jako ZigBee, které bylo určeno pro malé objekty s nízkou spotřebou – standard 802.15.4 WPAN (Wireless Personal Area Network) nebo Bluetooth, které vzniklo v roce 1998 a dnes je využíváno spoustou zařízení (tablety, chytré telefony, chytré hodinky, chytrá obuv apod.) [9].

1.1.1 Důvod nástupu

IoT je schopné zvýšit efektivitu podnikání, využívat inteligenci z širokého sortimentu zařízení, zlepšit činnost a zvýšit spokojenost zákazníků [1]. Očekáváno je zlepšení veřejné bezpečnosti, dopravy a zdravotnictví. Spektrum využití IoT je veliké, nicméně zmíněny budou tři nejperspektivnější směry:

- **Komunikace**
  Internet věcí může sdělovat informace osobám a systémům. Může se jednat o vyhodnocení stavu zařízení nebo o data ze senzorů, které jsou schopné sledovat životní funkce člověka. Pro mnohé firmy může být zajímavé sledovat pohyb zboží či např. nákladních automobilů, které zboží doručují. GPS (Global Positioning Sytem) systém je schopen zaznamenat informace o pozici nákladního automobilu, a tato data pak pomocí Internetu sdílet [10].
- **Automatizace a řízení**
  Jeden z hlavních důvodů pro nástup IoT je úspora energií, lepší synchronizace procesů v průmyslu a v dnešní době zejména Smart metering/Home Automation (sběr dat a řízení) pro domácnosti. IoT objekty jsou schopny sdílet pomocí Internetu data. Tuto výhodu lze využít např. v logistice či dopravě – v případě, že řídící jednotka automobilu vyhodnotí určitou závadu, a tato závada brání automobilu v dalším pohybu, tak IoT zařízení vyzve danou společnost vlastnící servisní vozidlo k výjezdu za účelem opravy poškozeného automobilu [10].
- **Úspora nákladů**.
  Mnoho společností se zaměřuje na úsporu nákladů. Díky IoT může společnost ušetřit náklady poskytnutím aktuálních dat ze zařízení např. určité kompetentní osobě, která na základě získaných informací provede kroky nezbytně nutné k záchraně daného zařízení či systému. Senzory mohou také měřit data, jako jsou jízdní vlastnosti a rychlost, s cílem snížit náklady paliva a opotřebení spotřebního materiálu. Nová inteligentní měřicí zařízení pro domácnosti a podniky mohou také poskytovat data, která pomáhají lidem pochopit spotřebu energie a příležitosti pro úsporu nákladů. V mnoha případech je možné určitá zařízení dálkově ovládat. Lze například vzdáleně nastavit klimatizaci v daném prostředí (kancelář) [10].

1.2 Protokoly Aplikační vrstvy použité pro IoT

Mnoho IoT standardů je navrženo s cílem zjednodušit a usnadnit práci programátorům aplikací a poskytovatelům služeb. Exponenciální růst IoT zařízení dal podnět pro vznik různých skupin komunikačních protokolů (v podstatě protokolů pracujích na aplikační vrstvě) na podporu rozvoje IoT pod vedením následujících společností: World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPCglobal, Institute of Electrical and Electronics Engineers (IEEE) a European Telecommunications Standards Institute (ETSI) [1].

V této kapitole jsou probrány protokoly: CoAP (Constrained Application Protocol), SIP (Session Initiation Protocol), MQTT (Message Queue Telemetry Transport), AMQP (Advanced Message Queuing Protocol), DDS (Data Distribution Service), MODBUS a závěr této kapitoly je věnován srovnávací tabulce jednotlivých protokolů.

1.2.1 Constrained Application Protocol (CoAP)

CoAP je jedním z nejnovějších protokolů aplikační vrstvy vyvinutý společností IETF. Je to síťově orientovaný protokol. Vychází z protokolu HTTP (Hypertext Transfer Protocol). Jako transportní protokol však využívá UDP (User Datagram Protocol) narozdíl od HTTP, které využívá protokol TCP (Transmission Control Protocol). Je založen na architektuře REST (Representational State Transfer). REST umožňuje klientům a serverům vystavit a používat webové služby. CoAP umožňuje zařízením o nízké spotřebě a nízkých výpočetních a komunikačních schopnostech využití REST interakcí (GET, POST, PUT a DELETE) díky schopnosti měnit délku UDP datagramu [1], [11].

1.2.2 Session Initiation Protocol (SIP)

SIP je textově orientovaný protokol využívaný pro signalizaci v internetové telefonii. Využívá port 5060 a protokoly UDP a TCP. Umožňuje zahájení, změnu a ukončení relace. Funkcemi protokolu jsou: vyhledání uživatele, zjištění schopností uživatele, dostupnost uživatele, nastavení relace, změna relace. Hlavními prvky sítě jsou UAC (User Agent Client) a UAS (User Agent Server). UAC generuje a odesílá žádosti a přijímá a zpracovává odpovědi. UAS žádosti přijímá a zpracovává a generuje odpovědi, které jsou odesílány zpět. Zprávy SIP se dělí na 2 kategorie:

- Žádosti (INVITE – zahájení či modifikace relace, ACK – potvrzení o přijetí finální odpovědi na zprávu ACK, OPTIONS – dotázání schopností na UAS, BYE – ukončení relace, CANCEL – zrušení nevyřízené žádosti, REGISTER – registrace současné polohy klienta).

- Odpovědi (1xx – dočasný stav, 2xx – úspěch, 3xx – přesměrování, 4xx – chyba klienta, 5xx – chyba serveru, 6xx – globální chyba) [12].

1.2.3 Message Queue Telemetry Transport (MQTT)

MQTT je protokol, který funguje na principu vydávání a odběru zpráv. Zaměřuje se na propojení embedded zařízení a sítí s aplikacemi a middlewarem. Základními prvky MQTT architektury jsou: vydavatelé, zprostředkovatelé (servery) a odběratelé. MQTT využívá model klient/server, kde všechny snímače jsou typu klient a pomocí TCP protokolu se připojují k serveru. Zprávy jsou doručovány na adresu s tématem. Klienti se mohou přihlásit k odběru více témat. Každý klient odebírající dané téma obdrží každou zprávu k němu publikovanou. Zprostředkovatel se stará o bezpečnost kontrolou oprávnění vydavatelů a odběratelů [1].

1.2.4 Advanced Message Queuing Protocol (AMQP)

AMQP je otevřený protokol aplikační vrstvy pro IoT se zaměřením na prostředí zpráv (princip vydávání a odběru zpráv). Vyžaduje spolehlivý transportní protokol TCP pro přenos. Komunikace je zpracována dle dvou hlavních prvků:

- Výměny – používají se ke směrování zpráv do příslušných front.
- Fronty zpráv – zde mohou být uloženy zprávy, které jsou rozesílány přijímačům [1].

1.2.5 Data Distribution Service (DDS)

DDS je protokol typu vydávání-odběr zpráv pro M2M komunikaci v reálném čase. Ve srovnání s protokoly jako MQTT nebo AMQP, DSS využívá multicasting, poskytuje lepší spolehlivost a Quality of Service (QoS). DSS podporuje 23 zásad QoS, pomocí kterých může být pokryta celá řada případů např. bezpečnost, priorita a spolehlivost. DDS architektura je definována dvěma vrstvami:

- Data-Centric Publish-Subscribe (DCPS) – zodpovídá za poskytování informace odběratelům.
- Data-Local Reconstruction Layer (DLRL) – volitelná vrstva, která slouží jako rozhraní funkčnosti pro DCPS vrstvu [1].

1.2.6 Protokol MODBUS

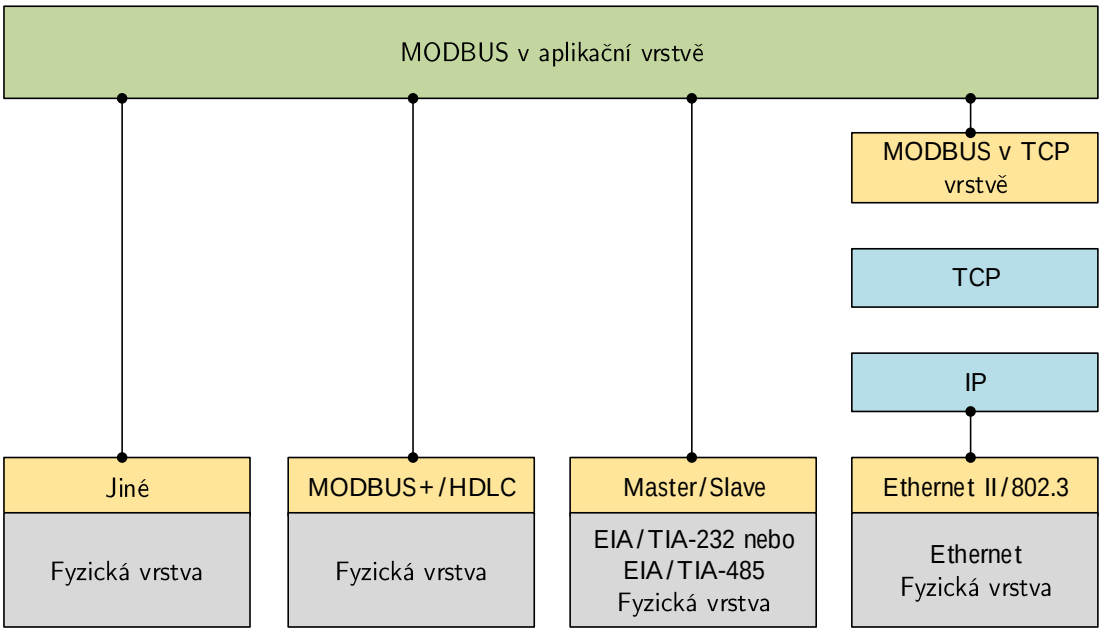
Cílem této práce je vytvoření aplikace, která bude moci číst či zapisovat data jednotlivých čidel/senzorů na bázi sériové linky nebo Ethernetu. Vzhledem k tomu, že MODBUS je protokol používaný v průmyslu a je primárně určený k tomuto typu komunikace, je vhodným kandidátem pro budoucí práci s aplikací, a proto mu bude věnována zvýšená pozornost.

MODBUS je protokol aplikační vrstvy ISO/OSI (International Standards Organization Open Systems Interconnection) modelu pro zasílání zpráv, který poskytuje komunikaci typu klient/server mezi zařízeními připojenými na různé typy sběrnic nebo sítí. V současné době jsou využívány následující [13]:

- **TCP/IP** přes Ethernet.
- **Asynchronní sériový přenos** pomocí různých médií (EIA/TIA-232-E, EIA-422, EIA/TIA-485-A, optické vlákno, rádiové vlny atd.).
- **MODBUS PLUS** – vysokorychlostní síťový přenos.

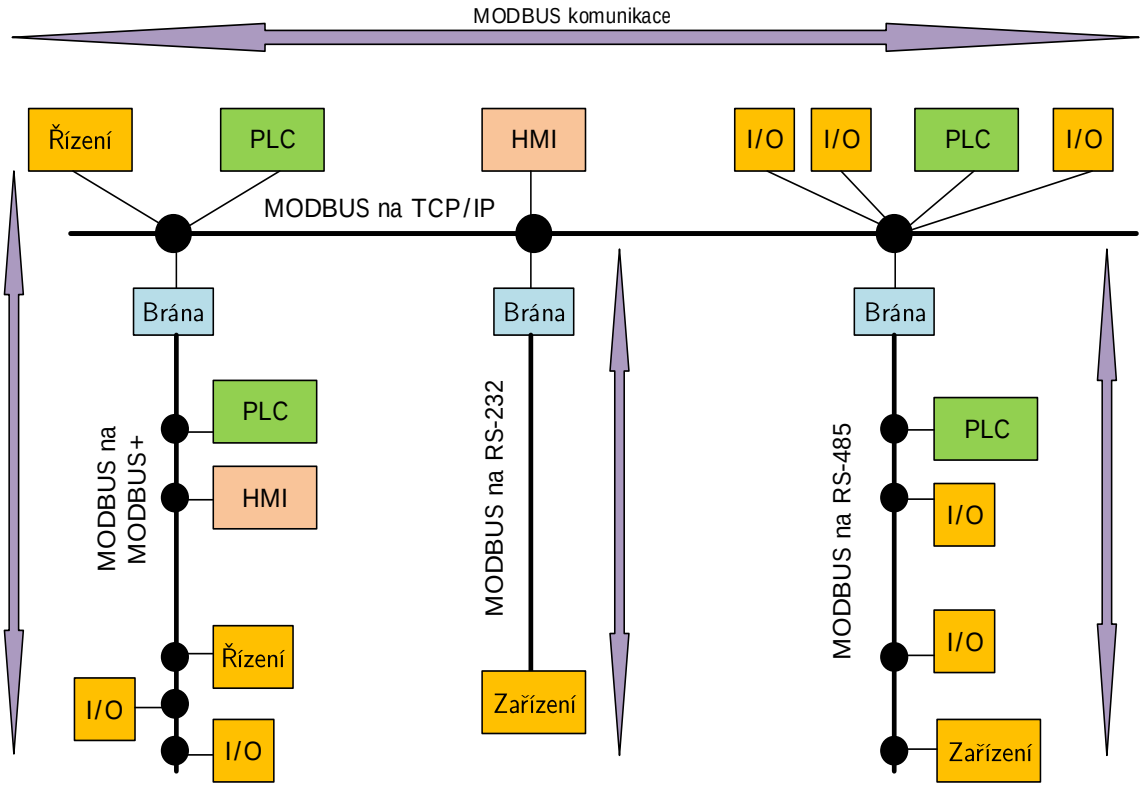
V průmyslu se MODBUS stal standardem roku 1979. V současné době podpora pro jednoduché konstrukce MODBUS neustále roste. Používá vyhrazený port 502 a komunikuje na základě požadavků a odpovědí [13].

Na obrázku 1.2 je znázorněna návaznost jednotlivých standardů či protokolů s aplikační vrstvou MODBUSu.



Obr. 1.2: MODBUS ve vrstvách ISO/OSI modelu [13]

Protokol MODBUS umožňuje komunikaci ve všech typech síťových architektur viz obrázek 1.3.

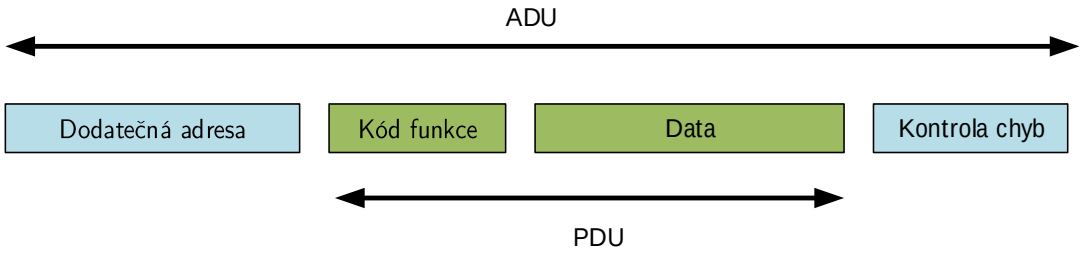


Obr. 1.3: Příklad síťové architektury MODBUSu [13]

Všechny typy zařízení jako jsou: PLC (Programmable Logic Controller), HMI (Human Machine Interface), ovládací panel, I/O zařízení a jiné) mohou použít protokol MODBUS k zahájení vzdáleného přístupu. Stejný typ komunikace může být prováděn jak na sériové lince tak na Ethernetu. Brány umožňují komunikaci mezi několika typy sběrnic a MODBUSem, který běží na TCP/IP [13].

Rámec protokolu MODBUS

MODBUS definuje datovou jednotku PDU (Protocol Data Unit). Mapování protokolu na konkrétní sběrnice nebo sítě mohou zavést některé dodatečné pole aplikační datové jednotky ADU (Application Data Unit). Znázornění rámce MODBUSu na obrázku 1.4 [13].



Obr. 1.4: Základní rámec protokolu MODBUS [13]

Komunikace

Datová aplikační jednotka v síti MODBUS je postavena ze strany klienta, který inicializuje MODBUS transakce. Funkce oznamuje serveru, jaký druh činnosti se bude vykonávat. ADU naváže formát žádosti iniciované klientem.

Pole „Kód funkce“ v PDU jednotce je kódováno v jednom bajtu. Platné kódy jsou v rozmezí 1-255 dekadicky (rozsah 128 až 255 je vyhrazen a slouží k odpovědi výjimek). Je-li zpráva odeslána z klienta na server, pole „Kód funkce“ informuje server, jaký druh činnosti se bude vykonávat. Kód funkce "0" není platný. Podřazené kódy funkcí jsou přidány do jiných funkčních kódů za účelem definování více akcí.

Datové pole zpráv odeslaných z klienta na server obsahuje dodatečné informace, které server používá pro definici kódu funkce. Datové pole může mít nulovou délku v některých druzích žádostí. V takovém případě server nevyžaduje žádné dodatečné informace. Kód funkce sám určuje akci.

Pokud nedojde k žádné chybě související s požadavkem MODBUS funkce, ADU datové pole při odpovědi ze serveru na klienta obsahuje požadované údaje. Pokud se chyba související s požadavkem MODBUS funkce objeví, pole obsahuje chybový kód s výjimkou, že aplikační server může použít k určení další opatření [13].

Tabulka 1.1 srovnává vlastnosti jednotlivých popsaným protokolů, aby čtenáři byla přiblížena jejich specifikace a možnosti využití.

Tab. 1.1: Srovnání IoT protokolů

<table>
<tr><td colspan="2">Aplikační protokol</td><td>DDS</td><td>CoAP</td><td>AMQP</td><td>MQTT</td><td>SIP</td><td>MODBUS</td></tr>
<tr><td rowspan="4">Infrastruktura protokolů</td><td>Směrovací protokoly</td><td colspan="6">RLP</td></tr>
<tr><td>Síťová vrstva</td><td colspan="4">6LoWPAN</td><td colspan="2">IPv4/IPv6</td></tr>
<tr><td>Linková vrstva</td><td colspan="6">IEEE 802.15.4, IEEE 802.11, IEEE 802.3</td></tr>
<tr><td>Fyzická vrstva</td><td>LTE-A</td><td colspan="2">EPC Global</td><td>IEEE 802.15.4</td><td>IMS</td><td>EIA/TIA-485/232</td></tr>
</table>

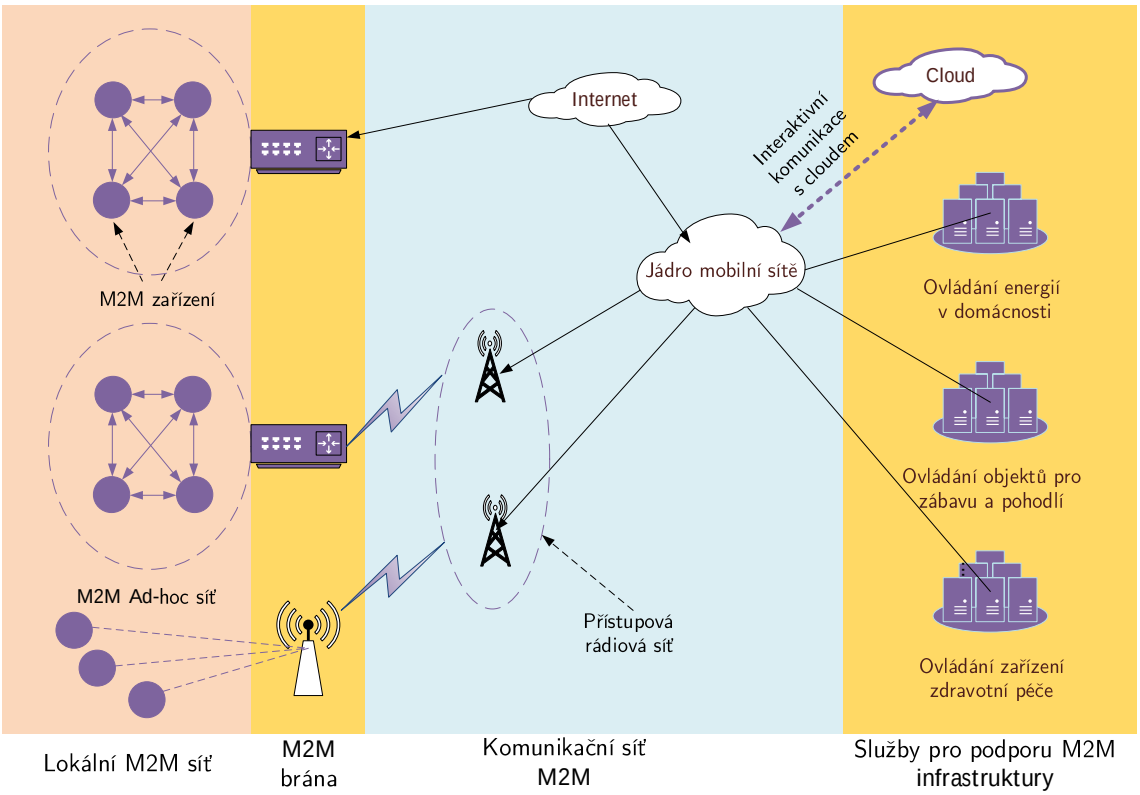
2 MACHINE-TO-MACHINE (M2M)

Vzhledem k čím dál rychlejšímu technologickému vývoji jsou v různých oblastech dnešního života aplikována zařízení, která jsou schopna plnit svou funkci bez účasti člověka. IoT a M2M termíny, označující tato zařízení, jsou v dnešní době člověku známější než dříve. M2M je dnes již nedílnou součástí Internetu věcí. Tvoří specifickou oblast informačních a telekomunikačních technologií/služeb [14].

M2M s praktickou částí této práce úzce souvisí. Výsledná aplikace, sloužící pro komunikaci s čidly, bude univerzální, a to jak pro uživatele, tak pro automatizovaný systém. Z toho vyplývá, že účast člověka není nutná. Proto je tato kapitola podrobněji věnována M2M komunikaci, použitým bezdrátovým protokolům a k nim odpovídajícím drátovým alternativám.

2.1 M2M komunikace

Jak je z názvu patrné, jde o komunikaci mezi dvěma nebo více zařízeními bez nutné účasti člověka (pokud není brána v potaz údržba a kontrola zařízení). M2M komunikace umožňuje mezi jednotlivými zařízeními (čidla, senzory) sdílet data a podle potřeby regulovat daná nastavení. Typicky se může jednat o komunikaci mezi určitým druhem senzoru a vyhodnocovací jednotkou (např. automatizované nastavování teplot v chladícím systému, kdy teploměr odesílá data do vyhodnocovací jednotky (ústředny), a ta na základě přijatých dat spíná nebo vypíná chlazení). Propojení je dnes většinou bezdrátové, ale i použití kabelového (v této práci použit MODBUS po kabelovém vedení) či optického rozvodu není vyloučeno [15].



Obr. 2.1: Obecné schema M2M komunikace [15]

2.1.1 Historie

Počátky M2M technologie sahají do poloviny 20. století. Tehdejší zařízení byla schopná komunikovat pouze jednosměrně a byla primárně využívána ke sběru dat. Tato zařízení našla uplatnění v NASA (National Aeronautics and Space Administration), kde byla snaha vyvinout automatizovanou M2M inteligenci pro kosmické mise. Dále také např. v meteorologii.

V roce 1995 firma Siemens předvedla GSM modul M1, který umožnil bezdrátovou komunikaci po síti GSM (Global System for Mobile Communications) [16], [17].

K dalšímu rozšiřování došlo s rozvojem GSM sítě, GPS, sítě WiMAX (Worldwide Interoperability for Microwave Access) a HSPA (High Speed Packet Acces).

V posledních letech dochází k neustálému rozšiřování M2M díky stávajícím i novějším síťovým technologiím, ke kterým patří zejména Wi-Fi, NFC (Near Field Communication), RFID, Bluetooth, a v současné době také sítě 4. generace - LTE (Long Term Evolution) [16].

Důvod nástupu

M2M je základní komunikační síť spojující senzory, stroje a objekty. Tržní příležitosti tím přímo souvisí s konektivitou z uvedených zařízení. Tento trh je zase poháněn větším makroekonomickým vývojem na trzích, které se spoléhají na dostupnosti M2M sítí [19]. IoT a M2M mohou pomoci ušetřit finanční i časové prostředky a dokázat ekonomiku také obohatit. Chytré domácnosti se rozrůstají čím dál více právě proto, že šetří člověku peníze. Zařízení a senzory mohou chytře sledovat spuštění levnějšího tarifu elektřiny a podle toho mohou spínat domácí spotřebiče, nebo se může jednat o lékařské vyšetření tlaku pacienta na dálku, ale také celkové řízení senzorů a čidel ve výrobním procesu bez toho aniž by člověk musel na místo fyzicky vyjet.

2.2 Specifické vlastnosti M2M provozu

Vlastnosti provozu se pro jednotlivé technologie liší. Mají různé požadavky na oblasti využití, rozdílné přenosové rychlosti, či frekvenční pásma. Potřeba systematického snížení spotřeby energií (elektřina, voda, plyn) přináší poptávku po rychlých, chytrých a online zařízeních pro velké společnosti, pro menší firmy i jednotlivce. Tato komunikační architektura včetně snímačů, akčních členů a komunikačních technologií je známá jako M2M nebo MTC (Machine-Type-Communication). Ve srovnání s tradiční H2H (Human-to-Human), což je klasický lidmi generovaný datový provoz, M2M technologie masivně narůstá. Předpokládá se, že nasazení M2M komunikace v plném rozsahu by mohlo mít velký dopad na H2H komunikaci, vzhledem k nepřipravenosti mobilních sítí a množství M2M zařízení. Mohlo by dojít ke kolapsu mobilních sítí v důsledku vytížení přístupové rádiové části [19]. Hlavní rozdíly popisuje tabulka 2.1.

Tab. 2.1: Porovnání M2M a H2H [19]

Hlavní rozdíly	**H2H**	**M2M**
Množství	Počet bezdrátově připojených zařízení neustále narůstá (telefony, tablety, notebooky), nicméně rostoucí potenciál M2M jen těžko překoná.	Společnost CISCO odhaduje, že v roce 2018 bude připojeno již 2 miliardy M2M zařízení – přechod na IPv6.
Napájení	Údržba napájení je prováděna člověkem. Nutnost nabíjení, či výměna zdroje (baterie).	Zařízení musí být koncipováno tak aby byl nutný minimální nebo žádný zásah člověka (dobíjení baterie pomocí okolních vlivů – světlo, vibrace atd.).
Objem dat	Obsazena významná šířka pásma, přenos velkých objemů dat.	Postačí minimální šířka pásma, data obvykle v řádu desítek bytů.
Velikost sítě	Závisí na vzdálenosti komunikujících stanic.	Velké množství připojených zařízení, odesílajících malý objem dat v pravidelných intervalech může způsobit exponenciální růst provozu sítě, popřípadě její kolaps.
Zpoždění	V určitých typech služeb (internetová telefonie) může zpoždění způsobit problémy v komunikaci. Existují však třídy tolerance.	Některé aplikace M2M komunikace probíhá v reálném čase. Závisí to na kritičnosti přenášených dat. Tolerance se u takových služeb snižuje (eHealth).

2.2.1 Intervaly zasílání dat

Intervaly zasílání datových jednotek se mohou výrazně lišit. Jejich doba se mění v závislosti na důležitosti přenášené informace. Intervaly mohou být v řádu sekund, minut i hodin. Pro zasílaní naměřených dat z vodoměrů na centrálu se používá výrazně delších intervalů. Příkladem může být systém BONEGA – v odpočtovém období je interval vysílání 20 - 24 s. Těchto mimořádně krátkých intervalů je docíleno díky efektivní spotřebě. Ostatní dny v roce je četnost vysílání každé 4 minuty [18].

V M2M provozu rozlišujeme 3 typy zasílání dat [19]:

- **Pravidelné aktualizace**
  K tomuto typu provozu dochází pokud zařízení vysílají zprávy o změnách stavu v centrální jednotce v pravidelných intervalech (např. chytré odečty vodoměrů). Intervaly řízené událostmi – odeslání dat je vyvoláno nějakou událostí např. nouzové oznámení zdravotního stavu pacienta [19].
- **Data na žádost**
  Tento typ provozu se využívá především tam, kde je zapotřebí velký přenos dat nebo přenos dat v reálném čase. Může se jednat o zjištění zdravotního stavu pacienta, nebo o poplašnou zprávu, která informuje o blížící se vlně tsunami. Nejdříve je vyslána žádost a poté jsou obdržena data. Některé informace např. hlášení o pozici zařízení nebo příjem firmware na zařízení v reálném čase neprobíhá.
- **Výměna důležitých dat**
  Tento poslední typ datového provozu nastává po události, a to po jednom z výše uvedených druhů odesílání dat. To zahrnuje všechny případy, kdy se větší množství dat vyměňuje mezi jednotlivými čidly a serverem. Tento provoz klade důraz na komunikační kanál směrem od prvku sítě (uplink), velikost dat může být konstantní i proměnlivá [19].

2.2.2 Používané komunikační prokotoly

Zigbee (IEEE 802.15.4)

Je bezdrátová komunikační technologie v sítích PAN (Personal Area Network). Díky Ad-hoc směrování je umožněna komunikace i na větší vzdálenosti. Technologie se díky nízkým odběrům elektrické energie využívá právě v průmyslu a senzorových sítích. Využívá bezlicenční ISM (Industrial, Scientific and Medical) pásmo o frekvenci 868 MHz a přenosovou rychlost 20 kb/s pro Evropu a 2,4 GHz a přenosovou rychlost 250 kb/s celosvětově. Energie potřebná pro přenos dat je 20 mW [19], [20].

Bluetooth Low energy (BLE)

Je bezdrátová komunikační technologie v sítích PAN zaměřená na nové aplikace ve zdravotnictví, fitness, zabezpečení atd. V porovnání s klasickým Bluetooth poskytuje BLE výrazně nižší spotřebu energie při zachování podobného komunikačního dosahu. Pracuje na frekvenci 2,4 GHz a energie potřebná pro přenos dat činí 10 mW [20], [22].

Wireless M-bus

Je komunikační protokol využívaný v průmyslu. Používá se zejména k dálkovému přenosu dat z měřičů (odběr energií). Komunikace probíhá v bezlicenčním pásmu okolo frekvence 868 MHz (2 kanály 868,3 a 868,95 MHz). Je využita hvězdicová topologie, kde množství měřících jednotek/snímačů přenáší data k jedné centrální jednotce (koncentrátoru). Je tedy využívána komunikace typu klient-server, kde koncentrátor funguje na principu serveru a veškerá čidla/senzory jsou klienti. Energetická náročnost se pohybuje okolo 25 mW pro přenos dat [20]. Bezdrátová verze M-bus vychází z původního standardu M-bus. U obou verzí M-bus se nevyužívá adresování, data se rozlišují na základě sériového čísla zařízení.

Low-Power WiFi IEEE 802.11ah

Může být uchazečem v M2M sítích. Všudypřítomné Wi-Fi sítě se mohou pochlubit obrovskou popularitou s více než 2 miliardy přístupovými body. Je založen na standardu IEEE 802.11 [19]. Navrhovaná architektura se skládá ze tří hlavních bloků:

- mikrořadiče s nízkou spotřebou energie,
- Wi-Fi modulu,
- připojených senzorů.

Rozšířený ovladač je řízen mikroprocesorem a přizpůsobuje senzory k režimu s nízkou spotřebou energie [21].

2.2.3 Spolehlivost vs. rychlost zasílání

Některá zařízení věnují pozornost bezpečnosti našeho života. Jedná se například o zařízení využívaná v elektronickém zdravotnictví. Další zařízení jsou klíčovými prvky v oblasti inženýrských infrastruktur. Mezi ně patří například zařízení detekující fáze a napětí, jističe atd. v chytrých sítích Smart Grid. Na taková zařízení jsou kladeny velké provozní nároky, zejména na odezvu a rychlost zasílání potřebných dat a jejich ověření. Na odečty energií se takové nároky nekladou, data není zapotřebí ověřovat, protože nenesou kritickou informaci. Inteligentní měřiče obvykle odesílají data dávkově v řádech sekund až milisekund. Měřiče udávají různé hodnoty střídy vysílání (1 %, 0,1 % nebo dokonce 0,01 %). Výsledná střída všech měřičů se odvíjí z počtu zařízení vysílajících v tentýž okamžik. Inteligentní sítě se vyvíjí a je pravděpodobné, že komunikace jednotlivých měřičů bude častější a cykly se zvýší [23], [24].

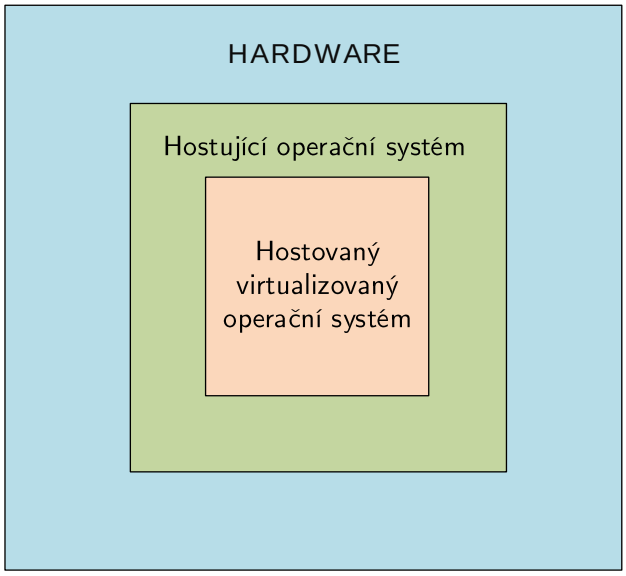
3 VIRTUALIZAČNÍ PLATFORMY

V dnešní době se stále více mluví o virtualizaci. Virtualizace je termín, kterým se označuje abstrakce počítačových zdrojů: virtuální paměť, virtualizace úložiště a další periferie [25]. Umožňuje provozovat více operačních systémů a aplikací na jednom fyzickém hardwaru ve stejnou dobu. Virtualizace může zvýšit informačním technologiím pružnost, flexibilitu a škálovatelnost. Snižuje investiční a provozní náklady, poskytuje vysokou dostupnost aplikací a zvyšuje produktivitu [26]. V této kapitole budou vysvětleny základní pojmy a bude popsána historie, typy virtualizace, rozdíly v použitých technologiích a virtualizační programy, které se v dnešní době využívají nejčastěji.

3.1 Základní pojmy

Zde budou vysvětleny dva základní pojmy, které s virtualizací úzce souvisí, názorně je popisuje obrázek 3.1.

1. **Hostující/hostitelský systém**
   Hostující systém je ten operační systém (OS), který poskytuje prostředky k virtualizaci (HW, SW).
2. **Hostovaný systém**
   Hostovaný systém je naopak ten systém, který prostředky k virtualizaci čerpá.



Obr. 3.1: Základní pojmy virtualizace

3.2 Historie

Virtualizace je termín, který se poprvé objevil v 60. letech 20. století. IBM v roce 1960 představila sálový počítač CP-40, který využíval hardwarovou virtualizaci na úrovni procesoru. V roce 1967 byla vyvinuta první verze hypervizoru. Druhá verze hypervizoru (CP-67) byla představena v roce 1968 a umožňovala sdílení paměti mezi virtuálními stroji [27]. Od počátku 80. do 90. let se začaly rapidně snižovat ceny za hardwarové komponenty, což vedlo k navýšení prodeje osobních počítačů a virtualizace byla na pár let opomíjena [28], [29], [30], [31]. S narůstajícím prodejem se také rapidně zvýšil odběr elektrické energie a tuto situaci bylo následně nutné řešit. Od konce 90. let až do konce první dekády nového milénia byly vyvinuty technologie společností VMware, Citrix, KVM (Kernel-based Virtual Machine) nebo Microsoft. Průlomové byly roky 2005 a 2006, kdy společnosti Intel a AMD začaly implementovat do svých procesorů technologie podporující virtualizaci [32].

3.3 Hypervizor

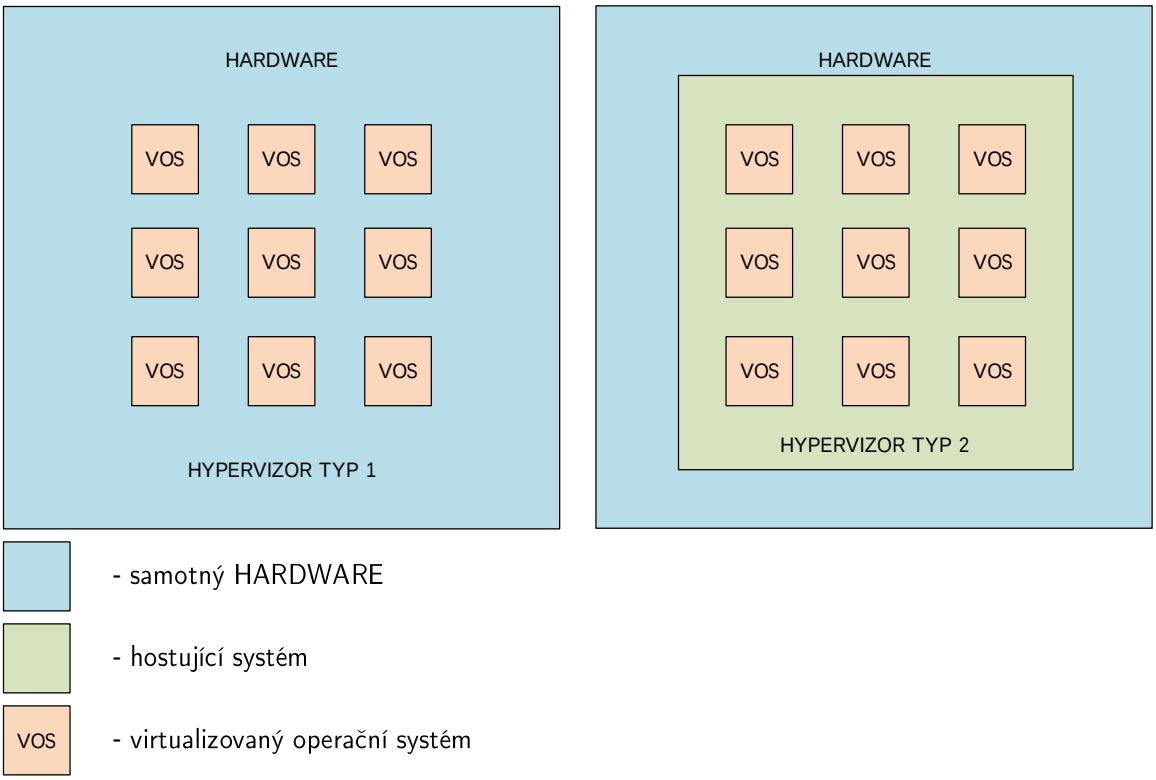
Hypervizor nebo take Virtual Machine Monitor (VMM) je softwarový nástroj, který umí vytvářet a provozovat virtuální stroje. Hypervizor, který spouští jeden nebo více virtuálních strojů je definován jako hostitel a každý virtuální stroj spuštěný na hypervizoru je definován jako host. Hypervizor řídí hostované operační systémy s virtuální operační platformou a řídí i vykonávání těchto hostovaných operačních systémů. Více instancí různých operačních systémů může sdílet virtualizované hardwarové prostředky [33]. Hypervizory jsou děleny na 2 typy:

Bare metal hypervizor/nativní (typ 1)

Tento typ hypervizoru lze spustit přímo na hardwaru hostitele za účelem správy a manipulace hostovaných operačních systémů. Hostovaný operační systém běží jako proces na hostiteli. Příkladem jsou Xen Hypervisor, VMware ESX/ESXi nebo např. KVM [33].

Hostovaný hypervizor (typ 2)

Tento hypervizor lze spustit na běžném operačním systému, stejně jako ostatní počítačové aplikace/programy. Příkladem mohou být Java Virtual Machine (JVM), VirtualBox nebo VMware Workstation [33]. Obrázek 3.2 znázorňuje oba zmíněné typy hypervizorů.



Obr. 3.2: Typy hypervizorů

3.4 Způsoby virtualizace

V informačních technologiích se používá několik druhů virtualizace. Některé umožní virtualizovat celý stroj, další umožní oddělit „systém v systému“ tak, že se chová jako samostatný systém. V této podkapitole budou popsány způsoby, které jsou používány různými výrobci virtualizačních či emulačních programů.

3.4.1 Emulace

Při emulaci dochází k simulaci kompletního fyzického stroje, který dovoluje běh jakéhokoliv nemodifikovaného operačního systému na zcela odlišné architektuře procesoru. Tento způsob se využívá při vývoji software pro platformy, které nejsou fyzicky dostupné. Mezi typické zástupce patří např. QEMU (Quick Emulator) [34].

3.4.2 Plná virtualizace

U plné virtualizace jsou simulovány všechny součásti počítače pro stejný druh procesoru. Hostovaný operační systém je oddělen od fyzické vrstvy a nemůže tím pádem

rozlišit, že nemá přístup k fyzickému hardwaru. Všechny aplikace jsou spuštěny jen na virtuálním hardwaru. Operační systém ani programové vybavení (aplikace) nepotřebují žádné modifikace. To nachází uplatnění pro vývojáře, kteří si potřebují otestovat vyvíjený produkt na několika různých operačních systémech. Mezi nejznámější aplikace patří: VMware Workstation a VMware Server [32], [34], [35], [36].

3.4.3 Paravirtualizace

Za použití plné virtualizace je téměř nemožné dosáhnout plného výkonu. Mimo fyzického hardwaru je totiž nutné emulovat i řadu operací (práce s pamětí, většina instrukcí procesoru, přístup na disk atd.). U paravirtualizace není nutné simulovat hardware a provádí se jen částečná abstrakce virtuálního prostředí, čímž lze dosáhnout vyššího výkonu s menší režií provozovaného systému. Jádro hostovaného operačního systému je však nutné modifikovat [32], [35], [36].

3.4.4 Virtualizace na úrovni operačního systému

Je režim, při kterém virtuální stroje (aplikace) sdílí systémové prostředky fyzického stroje a virtualizovaný systém pro svůj běh využívá jádro hostujícího operačního systému. To umožňuje spouštět více na sobě nezávislých a vzájemně izolovaných virtuálních počítačů. Ty jsou označovány jako kontejnery. Realizace je provedena pomocí hostitelského operačního systému, na kterém běží zprostředkovatel (virtualizační vrstva). Ten zajistí virtualizaci prostředí daného operačního systému [36].

3.4.5 Parciální virtualizace

Probíhá virtualizace částí hardwaru, zejména adresního prostoru. Podporuje sdílení zdrojů a izolaci procesů, ale nedokáže oddělit instance hostovaných OS. I když obecně nelze mluvit o virtuálním stroji, jedná se o významný přístup z historického hlediska. Byl použit např. u IBM CP/CMS [36].

3.5 Virtualizace dle způsobu využití

V této sekci budou popsány oblasti, ve kterých se virtualizace používá. Dále bude popsáno, za jakým účelem se virtualizace používá.

3.5.1 Virtualizace serverů

Virtualizace má největší smysl v oblasti serverových řešení. Většina fyzických serverů není často využívána ani z poloviny, což je neefektivní jak z hlediska využití

hardwaru, tak i energií. Zavedením virtualizace je možné vytvořit na fyzickém stroji takový počet virtuálních strojů, aby byl výpočetní výkon fyzického stroje plně využit [26], [37], [38].

3.5.2 Virtualizace desktopů

Virtualizací desktopů se rozumí kompletního provoz prostředí desktopu na centralizovaném serveru. Vzhledem k tomu, že většina výpočtů probíhá na serveru, k provozu stačí jednoduchá klientská zařízení („tenký“ klient), na kterých může virtualizované prostředí běžet. Důvody desktopové virtualizace mohou být čistě finanční, bezpečnostní a jiné. Uplatnění najde např. u společností, které mají větší množství identických systémů pracovních stanic [32], [39], [40].

3.5.3 Virtualizace úložiště

Mezi vrstvou virtuálních serverů a vrstvou fyzických disků existuje vrstva virtuálních úložišť. Tato virtuální úložiště jsou dostupná virtuálním serverům a tváří se jako cílové úložiště [32].

3.6 Současné virtualizační programy

V této podkapitole budou stručně popsány současné virtualizační techniky. Nebude se jednat o kompletní výčet virtualizačních technik na trhu, ale o popis nejvyužívanějších a nejznámějších. Emulační program QEMU bude použit v praktické části, a tak zde bude zmíněn jen okrajově a jeho detailní popis bude v následující kapitole.

3.6.1 VMware

Společnost VMware byla založena roku 1998 a získala patent na virtualizační technologie. Dnes je kvůli jejímu masivnímu použití a všudypřítomnosti považována za lídra na trhu s virtualizací [41].

3.6.2 VMware Workstation

Je hostovaný hypervizor, který se zaměřuje na provoz virtuálních strojů na desktopových systémech s operačními systémy Linux a Windows. Podporuje 3D akceleraci a připojení USB (Universal Serial Bus) zařízení. Je hojně využíván vývojáři, kteří potřebují vyzkoušet vyvíjený produkt na více operačních systémech [42].

3.6.3 VMware ESX/ESXi Server

Je hostující hypervizor, který je pro virtualizované stroje nejvyšším arbitrem. Po verzi 4.1 byl ESX server přejmenován na ESXi. Je postaven na linuxovém jádře. Virtuální stroje jsou spravovány nástrojem vSphere, který je součástí produktu ESX/ESXi. Lze ho stáhnout zadáním IP adresy hypervizoru do adresního pole webového prohlížeče. Program vSphere je možné nainstalovat pouze na stanice s operačním systémem Windows. Nejnovější verze ESXi podporuje až 128 virtuálních procesorů, 4 TB virtuální operační paměti RAM (Random Access Memory), zařízení USB 3.0, až 62 TB na jeden virtuální disk [43].

3.6.4 VirtualBox

VirtualBox je univerzální virtualizační program pro x86 hardware, využívající plnou/nativní virtualizaci. Je vydáván jako OpenSource software v souladu s podmínkami GNU General Public License (GPL) a je zdarma ke stažení. V současné době VirtualBox funguje na systémech Windows, Linux, Macintosh a Solaris a podporuje velké množství hostovaných operačních systémů Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8), DOS/Windows 3.x, Linux (2.4, 2.6 a 3.x), Solaris a OpenSolaris, OS/2 a OpenBSD. VirtualBox je aktivně vyvíjen a má stále rostoucí seznam funkcí, podporované operační systémy [44].

3.6.5 Microsoft Server Hyper-V

Microsoft se při vývoji tohoto produktu spoléhal na hardwarovou podporu virtualizace v procesorech a naprogramoval celý systém od začátku. První beta verze byla vydaná jako součást Windows Serveru 2008. Později byla dostupná ve verzi Windows Server 2012/2012 R2 a také zdarma ke stažení pro systémy Windows 8/8.1. Podpora hostů ve verzi 2012/2012 R2 je zatím jen pro produkty z řad Windows, a to pro serverové i desktopové systémy. Nejnovější Hyper-V už slibuje možnost bootovat i různé Linuxové systémy (Ubuntu 14.04 a novější, SUSE Linux Enterprise Server 12 a novější, Red Hat Enterprise Linux 7.0 a novější, and CentOS 7.0 a novější). Pomocí technologie Hyper-V je možné vytvářet a spravovat virtualizované výpočetní prostředí pomocí technologie virtualizace, která je součástí Windows Serveru. Aplikace Hyper-V nainstaluje důležité součásti a volitelně může nainstalovat nástroje pro správu. Důležité součásti obsahují hypervizor systému Windows, službu Správa virtuálních počítačů technologie Hyper-V, zprostředkovatele rozhraní virtualizace WMI (Windows Management Instrumentation) a další součásti virtualizace [45], [46], [47], [48].

3.6.6 Kernel-based Virtual Machine (KVM)

KVM je řešení plné virtualizace pro Linux a hardwarové platformy x86 obsahující rozšíření podpory procesoru (Intel VT nebo AMD-V). Skládá se ze zatížitelného modulu jádra „kvm.ko“, který poskytuje virtualizační infrastrukturu jádra a specifické moduly procesoru „KVM-intel.ko“ nebo „KVM-amd.ko“. Používáním KVM je možné provozovat množství virtuálních strojů běžících s neupravenými obrazy Linuxu nebo Windows. Každý virtuální stroj má vlastní virtualizovaný hardware: síťovou kartu, disk, grafický adaptér, atd. KVM je open source software. K virtualizaci a podpoře hardwarové používá nástroj QEMU, bez něhož není schopen fungovat [49], [50].

3.6.7 Linux-Vserver

Linux-VServer poskytuje virtualizaci pro GNU/Linux na úrovni operačního systému. Toho je dosaženo izolací úrovně jádra. To umožňuje provozovat několik virtuálních jednotek najednou. Tyto jednotky jsou dostatečně izolovány, aby byla zaručena požadovaná bezpečnost. Tento způsob ale nijak neomezuje efektivní využití dostupných prostředků. Linux-VServer umožňuje virtualizovat servery na vrstvě operačního systému. To může znamenat rozdělení jednoho fyzického serveru na několik malých výpočetních oddílů (kontejnerů). Každý takový oddíl vypadá a chová se jako skutečný server, z hlediska svého majitele. Na unixových systémech je tato technologie založena na standardním mechanismu chroot (Change Root) neboli změna kořenového adresáře [51], [52].

3.6.8 OpenVZ

OpenVZ je virtualizační kontejnerový nástroj pro Linux. OpenVZ vytváří bezpečnější izolované Linux kontejnery na jednom fyzickém serveru. Ty umožňují lepší využití serverů a zajištění, že aplikace nebudou v konfliktu. Každý kontejner funguje stejně jako samostatný server. Jednotlivé kontejnery lze restartovat nezávisle na sobě. Funguje tedy na stejném principu jako Linux-VServer. OpenVZ je bezplatný open source software, dostupný pod licencí GNU GPL [52], [53].

3.6.9 QEMU (Quick EMUlator)

QEMU je otevřený hostovaný hypervizor a emulátor, který uživateli poskytuje hardwarovou virtualizaci. Nástroj QEMU umí emulovat CPU (Central Processing Unit) prostřednictvím dynamických binárních překladů a poskytuje sadu modelů přístrojů, které umožňují provozovat nejrůznější nemodifikované operační systémy. Nejčastěji

se tedy používá pro emulace různých operačních systémů, potřebných při vývoji určitého produktu a potřebě vyzkoušet tento produkt na více operačních systémech [58]. Je podporován systémy Linux, Microsoft Windows, Macintosh a některými UNIX systémy. QEMU bude použito jako softwarový nástroj pro bakalářskou práci, proto bude více rozebráno v následující kapitole.

4 QEMU QUICK EMULATOR

QEMU, jako jediný z výše uvedených softwarových programů (viz strana 33) umí emulovat různé platformy/procesory. Proto byl vybrán k implementaci emulovaného hardwaru a softwarového vybavení, které bude možné přenést na fyzické zařízení. Je licencován pod GNU General Public License. V současné době disponuje QEMU velkou podporou vývojářů, aktualizace jsou téměř pravidelné a nové verze vychází i několikrát za měsíc. Aktuální verze (volně ke stažení) je 2.5.0-rc2 a vyšla 27. listopadu 2015 [59]. V této kapitole bude popsáno: v jakých režimech umí QEMU pracovat, požadavky na balíčky, výhody a nevýhody QEMU.

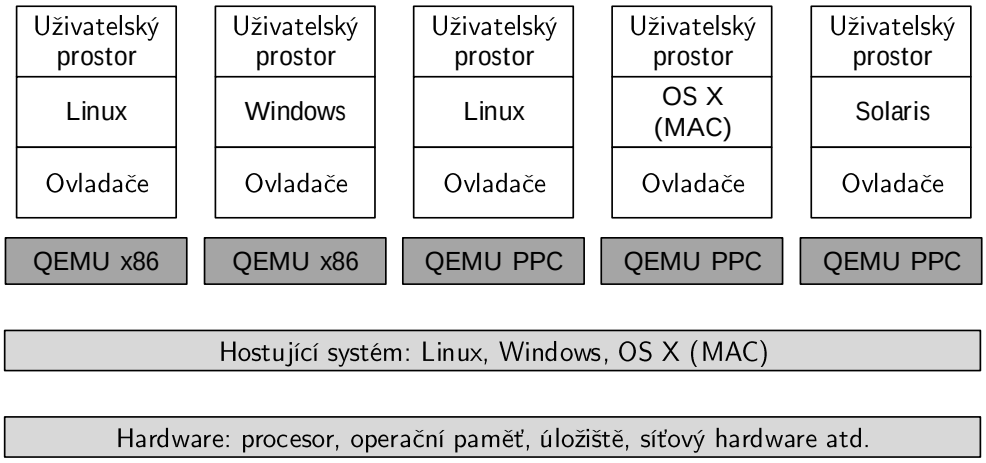
4.1 Režimy QEMU

QEMU umí pracovat ve dvou hlavních režimech:

1. **Uživatelská emulace**
   Tento režim funguje pouze na OS Linux nebo Darwin/Macintosh. V tomto módu je uživateli umožněno spouštět aplikace určené pro jinou platformu [59]. Hlavním principem uživatelské emulace je křížový překladač viz níže.
2. **Systémová Emulace**
   V tomto režimu QEMU emuluje úplný operační systém, včetně periferií. To může být využito k virtualizaci několika hostovaných počítačů na jednom systému. Lze emulovat mnoho hostovaných operačních systémů např.: Linux, Solaris, Microsoft Windows, DOS, BSD. Podporuje emulaci procesorů různých instrukčních sad např.: ISA PC, PREP, MIPS, ARMv7, ARMv8, PowerPC, SPARC, ETRAX CRIS, x86 a x86_64 atd. [59].

4.2 Architektura

Na obrázku 4.1 je znázorněna architektura QEMU pomocí jednotlivých vrstev. Od hardwaru, přes hostující operační systém až po jednotlivé hostované systémy, které jsou nainstalovány pomocí QEMU.



Obr. 4.1: Architektura QEMU [60]

4.3 Požadavky na balíčky

QEMU lze nainstalovat na UNIXové/Linuxové systémy i na Windows systémech. Tato práce bude věnována implementaci pouze v systému Linux. Existují 2 způsoby instalace. První způsob zahrnuje stažení balíčku QEMU z webových stránek projektu, jeho následné rozbalení a instalaci. Druhý způsob (značně jednodušší) zahrnuje otevření příkazové řádky a zadání příkazu, který se však může napříč linuxovým distribucím lišit. Instalaci knihoven většinou systém nabídne sám [55], [56].

Příkaz 4.1: Instalace pro systémy Debian „rodiny“ - Ubuntu a jiné

```
sudo apt-get install qemu
```

Příkaz 4.2: Instalace pro systémy RedHat „rodiny“ - CentOS a jiné

```
yum install qemu
```

Příkaz 4.3: Instalace pro Gentoo

```
yast -i qemu
```

Příkaz 4.4: Instalace pro Arch Linux

```
sudo pacman -S qemu
```

Příkaz 4.5: Instalace pro Macinosh

```
brew install qemu
```

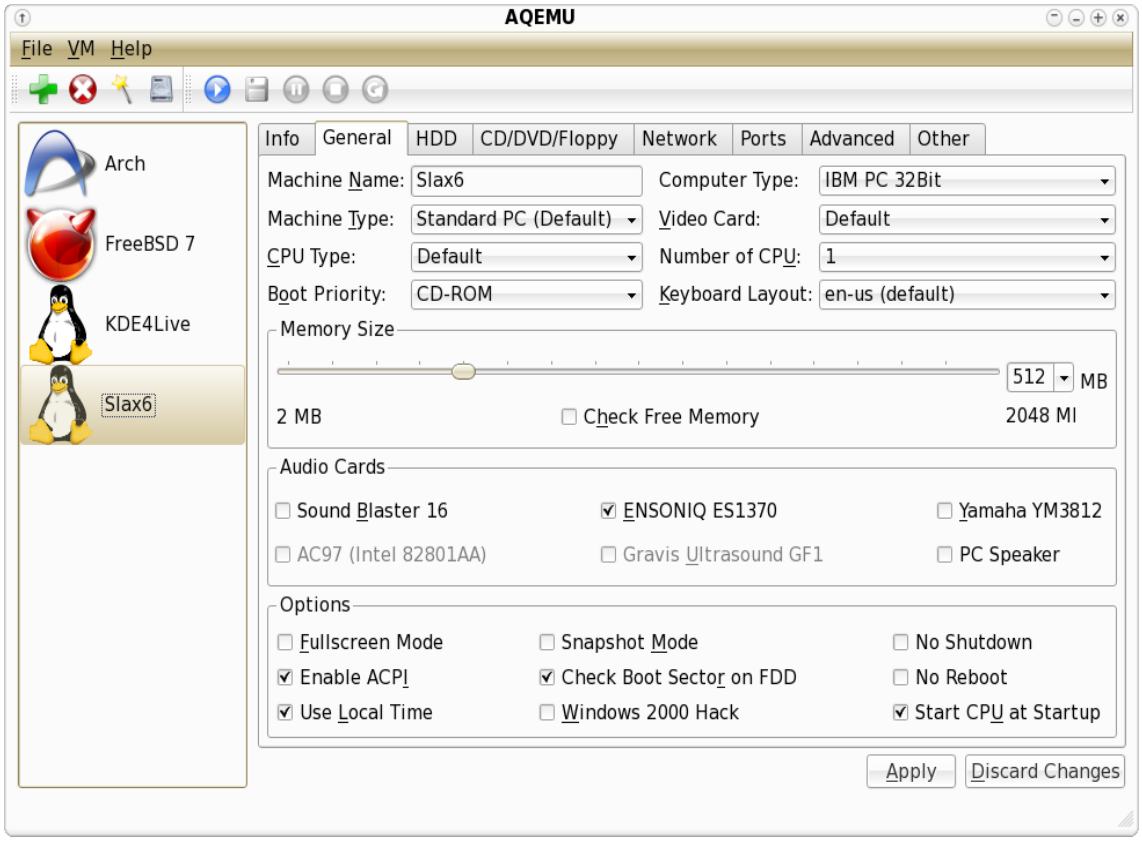
4.4 Výhody QEMU

Emulátor QEMU nabízí spoustu výhod z nichž nejvýznamnější jsou:

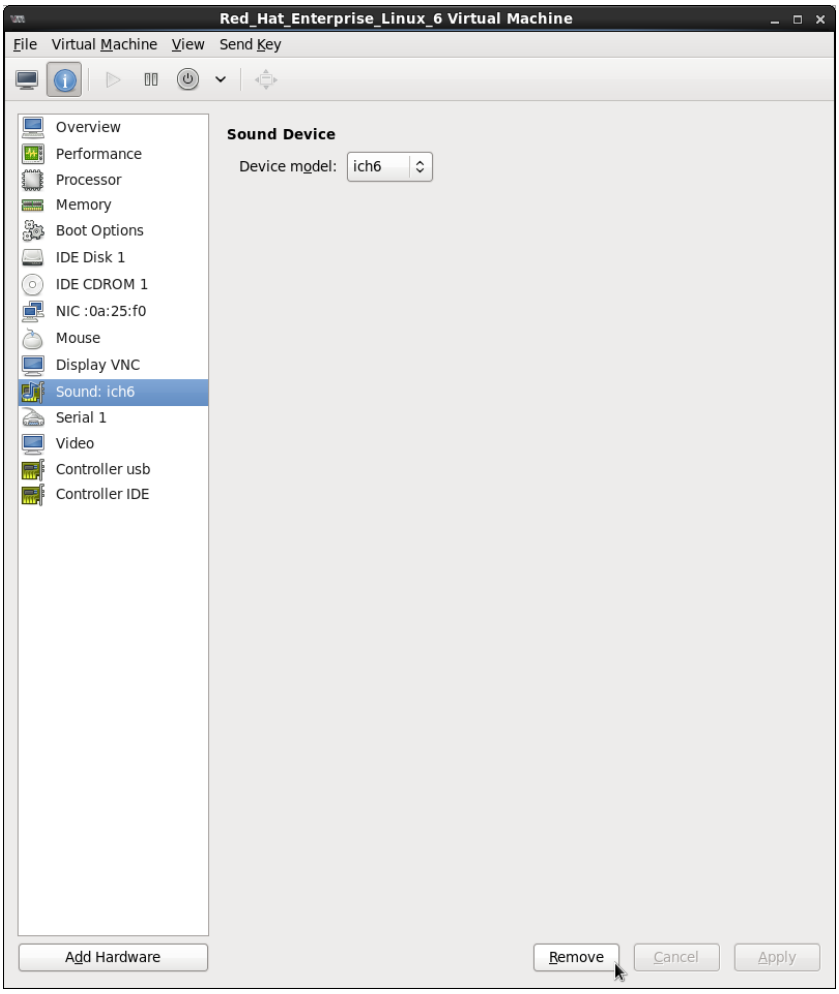
1. **Křížový překladač – CrossCompiler**
   Křížový překladač je překladač, který je schopen přeložit či spustit spustitelný kód, určený pro jinou platformu, na stávajícím (hostujícím) systému [58]. Lze tedy např. vyzkoušet program určený pro systém Android na PC s hostujícím systémem Linux. Tato vlastnost může být z pochopitelných důvodů využita v oblasti vývoje Embedded zařízení. Pomocí křížového překladače je možné emulovat zařízení s kompletní hardwarovou konfigurací [61]. Na tuto virtuální hardwarovou konfiguraci lze následně nainstalovat virtuální OS, čímž získáme kompletní fyzický prvek i s programovým vybavením (např. emulované Raspberry Pi se systémem Raspbian). Výhodou křížového překladače je také možnost emulované zařízení vyzkoušet pod záštitou výkonnějšího hardwaru. Embedded zařízení často nemají nijak vysoký výkon a kompilace některých kódu může být pro taková zařízení velice náročná, zdlouhavá a z těchto důvodů také velmi nepraktická. Překlad je obstarán silnějším hardwarem a po zajištění stability systému lze jednoduše celý virtuální systém přenést na fyzické zařízení, které bylo předtím pouze emulováno.
2. **Úspora finančních prostředků**
   Z výše uvedených důvodů lze jednoduše odvodit další nespornou výhodu, a sice úsporu financí. Program QEMU je volně dostupný, takže odpadá nutnost nákupu. Emulací zařízení odpadá nutnost zakoupení zařízení. Lze tedy zařízení nejdříve emulovat, vyzkoušet jeho chování, odladit chyby v programu a poté ho pořídit.
3. **Nezávislost na hostujícím operačním systému**
   Jako hostující operační systém pro QEMU lze zvolit jak většinu UNIX-ových systémů tak i systémy Windows a vzhledem k možnosti využití emulace celého počítače lze pod QEMU spustit prakticky libovolný operační systém [59].
4. **Emulace hardwaru**
   Jak již bylo výše uvedeno, QEMU umí emulovat širokou škálu procesorů, což tvoří výhodu pro vývojáře aplikací určených pro jinou platformu. S tím souvisí také rychlost, která je při překladu na výkonnějším hardwaru nedocenitelná [59].

4.5 Nevýhody QEMU

Za malou nevýhodu může být považována absence defaultního GUI (Graphical User Interface). Nicménně lze doinstalovat například AQEMU nebo Virt-manager, který rozšíří QEMU o grafické rozhraní viz obrázky 4.2, 4.3. I když grafické prostředí dovoluje uživateli mnoho možností, tak příkazovou řádku nemůže nikdy plně nahradit.



Obr. 4.2: program AQEMU



Obr. 4.3: program Virt-manager

4.6 Chroot vs. Virtualizace

V unixových systémech existuje rozšířený typ „virtualizace“. Chroot je na unixových operačních systémech název pro operaci, která pro spuštěné aplikace i jejich potomky mění kořenový adresář. Aplikace uvnitř chrootu si tak myslí, že jsou obklopeny vlastním systémem. Nejedná se tedy o virtualizaci jako takovou. Typickým příkladem jsou Linux-VServer nebo OpenVZ (viz strana 35). Principem je pomocí chrootu vytvořit kontejnery, do kterých je umístěn daný systém. Hostujícímu systému se hostovaný systém jeví jako proces. Není možné přidělit jednotlivým kontejnerům virtuální paměť, či procesor, jak je to u jiných technologií, protože jednotlivé kontejnery sdílí stejné výpočetní zdroje s hostujícím systémem. Dalším příkladem může být systém Android, který je také založen na linuxovém jádře. Systém je nainstalován do fyzického hardwaru (např. telefonu), ale pro veškeré uživatelské aplikace je vytvořen chroot. Uživatel tak nemá právo zasahovat do systému [52], [57].

5 PRAKTICKÁ REALIZACE

Tato část práce se věnuje výběru vhodného vestavěného zařízení. Dále bude odůvodněn výběr daného operačního systému, který bude do zařízení implementován. Bude popsána architektura procesorů ARM, instalace QEMU a připravení emulovaného systému, který bude určen pro daný typ procesoru.

5.1 Výběr zařízení

Zařízení, které je bude vybráno musí splňovat následující požadavky:

1. dostatečný výkon,
2. malé rozměry IoT krabičky,
3. možná komunikace se senzory pomocí RS-485 nebo Ethernetu,
4. průmyslové řešení.

Možná řešení:

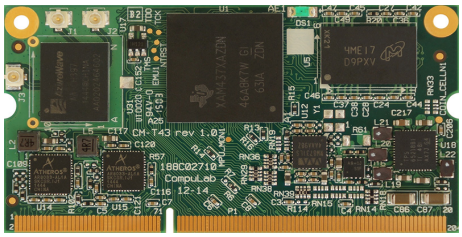
1. Raspberry pi – v určitých odvětvích průmyslu nelze použít kvůli vnějším vlivům magnetického vlnění (např. frekvenční měniče), u nejnovějšího modelu vzniká problém s přehříváním při vyšší zátěži – **nevhodný pro řešení práce**.
2. Banana pi, Orange pi – čínské vestavěné desky, nejistá podpora systémů, problém sehnat v rozumné době v Evropě (dlouhá doba dovozu), v určitých odvětvích průmyslu, podobně jako u Raspberry, nelze použít – **nevhodný pro řešení práce**.
3. Zásuvný modul CM-T43 – průmyslové řešení, malé rozměry, teplotně odolnější (-40 až 85 °C), jistá dodávka počtu kusů od společnosti Texas Instruments – **vhodný pro řešení práce**.

5.2 Modul CM-T43 - Texas Instruments

V této sekci bude popsán zásuvný modul, pro který byl sestavován obraz operačního systému Debian/armhf a deska, která slouží jako vývojový kit, do níž může být modul implementován.

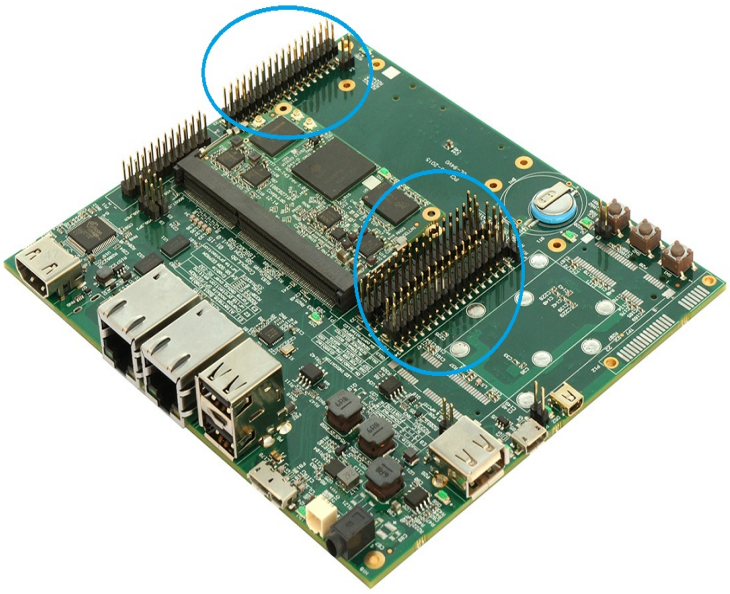
Zásuvný modul CM-T43 viz obrázek 5.1 může být osazen procesory:

- **Texas Instruments Sitara AM4379 ARM Cortex-A9, 1GHz NEON™ SIMD a VFPv3**.
- **Texas Instruments Sitara AM4376 ARM Cortex-A9, 800MHz NEON™ SIMD a VFPv3 NEON™ SIMD a VFPv3**.



Obr. 5.1: Zásuvný modul CM-T43

Může disponovat operační pamětí RAM **128MB – 1GB, DDR3-800**. Na modul CM-T43 bude později přenesen celý operační systém Linux Debian.



Obr. 5.2: Deska pro zásuvný modul – SBC-T43

Originální deska SBC-T43 viz obrázek 5.2 je osazena porty USB, MMC / SDIO, sériovými UART (Universal Asynchronous Receiver Transmitter), debuggovacím portem RS-232 apod. Fyzické sériové porty RS-485 a RS-232 na originální desce nejsou, ale dají se implementovat pomocí USB modulu, který se připojí k desce. Systému se to poté jeví jako fyzický sériový port. Na místech, vyznačených modrými kroužky, se nachází GPIO (General-purpose input/output) nožičky, mezi nimiž se nachází UART.

5.3 Debian Jessie 8.2

Jako hostující operační systém byla vybrána linuxová distribuce Debian. OS Debian byl v projektu použit z následujících důvodů:

- velká podpora vývojářů,
- tvoří jednu z největších „rodin“ linuxových distribucí na světě,
- je nenáročný,
- je svobodný, skládá se ze základního programového vybavení a dalších nástrojů, kterých je k provozu počítače třeba
- stabilita systému [62].

5.3.1 Instalace Debianu

Debian je volně šiřitelný a dostupný systém a je možné ho stáhnout z oficiálních webových stránek https://www.debian.org/. Současná verze Debianu je 8.2 - Jessie s verzí jádra 3.16. I přes vhodnost testing větve pro desktop byla jako hostující systém zvolena stable verze. Instalace spočívá v několika krocích:

1. První krok je stažení příslušného obrazu systému z webových stránek Debian. Pro projekt je zvolena grafická nástavba XFCE (XForms Common Environment), která je nenáročná na běh systému a je stabilnější a svižnější než např. KDE (K Desktop Environment) či GNOME (GNU Network Object Model Environment).
2. Po stažení se vytvoří instalační médium (USB nebo CD). V závislosti na použitém hostujícím operačním systému lze využít např.: rufus.exe ve Windows systémech nebo unetbootin (Windows, Linux i Macintosh systémy).
3. Zavedení média a instalace systému.

Samotná instalace Debianu už je vhodná pro uživatele se znalostí Linuxových systému.

5.4 ARM

Termín ARM označuje architekturu procesorů, které disponují nízkým odběrem elektrické energie. Používá se zejména v mobilních zařízeních či vestavěných systémech. ARM vývoj byl započat ve Velké Británii ARM Holdings v 80. letech 20. století. První „rodina“ ARM procesorů nesla název ARM1 a architektura byla nazvána ARMv1. Dnes využívané „rodiny“ jsou např.: Cortex, StrongARM nebo Xscale, které využívají různé architektury (ARMv4, ARMv6, ARMv7 a jiné) [63], [64]. Architektura ARM se neustále vyvíjí a Debian proto nabízí tři základní ARM porty pro nejlepší podporu širokého spektra systémů:

1. **Debian/armel** cílí na starší 32 bitové ARM procesory bez hardwarové podpory výpočtů s plovoucí desetinnou čárkou [65].
2. **Debian/armhf** funguje jen na novějších 32 bitových ARM procesorech, které mají implementovanou architekturu alespoň ARMv7. Debian/armhf využívá tyto rozšířené možnosti a vyšší výkon nových modelů [65].
3. **Debian/arm64** funguje na 64 bitových ARM procesorech s implementací architektury alespoň ARMv8 [65].

5.5 Instalace QEMU

Instalace QEMU na distribuci Debian spočívá v zadání příkazu:

Příkaz 5.1: Instalace QEMU

```
root@ntblenovo:# apt-get install qemu
```

Potřebné balíky k instalaci QEMU se doinstalují automaticky.

5.5.1 Vytvoření obrazu v QEMU

Nejdříve bylo nutné zjistit specifikace desky z důvodu nastavení emulovaného obrazu pro konkrétní platformu. Modul CM-T43 - TI AM437x System-on-Module|Computer-on-Module je osazen procesorem ARM Cortex-A9 (viz strana 42) a pamětí o velkosti 1GB. Dále je možné postupovat dle následujících kroků:

Příkaz 5.2: Vytvoření adresáře

```
root@ntblenovo:# mkdir /home/uzivatel/Dokumenty/
                Debian-armhf-Wheezy
root@ntblenovo:# cd Debian-armhf-Wheezy
```

Vytvoření adresáře na souborovém systému pro umístění daných souborů. Aktuální verze Debianu pro armhf, nazvaná Jessie ještě není plně vyvinuta. Z tohoto důvodu byla vybrána stabilní verze Wheezy, kterou je možné stáhnout dle následujících příkazů:

Příkaz 5.3: Stažení potřebných souborů

```
root@ntblenovo:# wget http://ftp.debian.org/debian/dists/
                wheezy/main/installer-armhf/current/
                images/vexpress/netboot/
                vmlinuz-3.2.0-4-vexpress
```

```
root@ntblenovo:# wget http://ftp.debian.org/debian/dists/
                wheezy/main/installer-armhf/current/
                images/vexpress/netboot/initrd.gz
```

Příkaz 5.4: Vytvoření virtuálního disku pro QEMU

```
root@ntblenovo:# qemu-img create -f qcow2 debian.img 8G
```

Tento příkaz vytvoří obraz disku o velikosti 8GB (dle velikosti SD kardy implementované v modulu).

Příkaz 5.5: Instalace systému do vytvořeného obrazu

```
root@ntblenovo:# qemu-system-arm -m 1024M -sd debian.img \
-M vexpress-a9 -cpu cortex-a9 \
-kernel vmlinuz-3.2.0-4-vexpress -initrd initrd.gz \
-append "root=/dev/ram" -no-reboot
```

Příkaz *qemu-system-arm* definuje jaká architektura procesoru bude emulována, parametr *-m* je „memory“ a přiřazuje 1024MB paměti RAM, parametr *-sd* definuje typ úložiště (sd karta) a následně je zaveden obraz již vytvořeného disku. Parametr *-M* „machine“ definuje jaký typ zařízení bude emulován, *-cpu* upřesňuje typ procesoru. Parametrem *-kernel vmlinuz-3.2.0-4-vexpress* zavedeme jádro linuxu a ramdisk *-initrd initrd.gz*. Parametr *-append "root=/dev/ram"* připojuje RAM disk *initrd* při bootování do uvedene cesty. Po tomto nastavení je spuštěna instalace identická s jakoukoliv instalací Debianu.

5.5.2 Spuštění obrazu v QEMU

Zde bude pomocí několika příkazů znázorněno spuštění již nainstalovaného systému z vytvořeného obrazu.

Příkaz 5.6: Spuštění obrazu v QEMU

```
root@ntblenovo:# modprobe nbd
```

Tento příkaz zavede modul *nbd* do jádra hostujícího systému. NBD (Network Block Device) je protokol, který umožní přistupovat k souborům obrazu a také se systému jeví jako blokové zařízení, ke kterému lze přistupovat.

Příkaz 5.7: Připojení obrazu do zařízení */dev/nbd0*

```
root@ntblenovo:# qemu-nbd -c /dev/nbd0 /home/uzivatel/Dokumenty/
                Debian-armhf/debian.img
```

Příkaz 5.8: Výpis zařízení */dev/nbd*

```
root@ntblenovo:# fdisk -l nbd*
Disk /dev/nbd0: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000e3348

Device     Boot    Start     End  Sectors Size Id Type
/dev/nbd0p1         2046 15624191 15622146 7,5G 5 Extended
/dev/nbd0p5         2048 11718655 11716608 5,6G 83 Linux
/dev/nbd0p6     11720704 15624191 3903488 1,9G 82 Linux swap /
   Solaris
```

Příkazem fdisk s parametrem *-l nbd** se vypíše seznam virtuálních diskových oddílů, které byly při instalaci vytvořeny. Oddíl */dev/nbd0p1* označuje celý „fyzický“ disk. Oddíl */dev/nbd0p5* je systémový a oddíl */dev/nbd0p6* je tzv. SWAP (odkládací prostor). V závislosti na vytíženosti systému je možné později tento oddíl odstranit.

Příkaz 5.9: Připojení souborového systému hostujícího operačního systému do souborového systému hostitelského operačního systému

```
root@ntblenovo:# mount /dev/nbd0p5 /mnt/
```

Příkaz 5.10: Kopírování ramdisku z adresáře */mnt/boot/initrd.img-3.2.0-4-vexpress* do adresáře */home/uzivatel/Dokumenty/Debian-armhf-Wheeze/*

```
root@ntblenovo:# cp /mnt/boot/initrd.img-3.2.0-4-vexpress
                /home/uzivatel/Dokumenty/Debian-armhf-Wheeze/
```

Příkaz 5.11: Kopírování jádra z adresáře */mnt/boot/vmlinuz-3.2.0-4-vexpress* do adresáře */home/uzivatel/Dokumenty/Debian-armhf-Wheeze/*

```
root@ntblenovo:# cp /mnt/boot/vmlinuz-3.2.0-4-vexpress
                /home/uzivatel/Dokumenty/Debian-armhf-Wheeze/
```

Příkaz 5.12: Odpojení hostujícího souborového systému od hostitelského

```
root@ntblenovo:# umount /mnt
```

Příkaz 5.13: Bootování instalovaného operačního systému

```
root@ntblenovo:# qemu-system-arm -m 1024M -sd debian.img \
-M vexpress-a9 -cpu cortex-a9 \
-kernel vmlinuz-3.2.0-4-vexpress -initrd
initrd.img-3.2.0-4-vexpress\
-append "root=/dev/mmcblk0p1" -no-reboot
```

Nyní je zavedeno jiné (instalované) jádro a ramdisk. To nyní slouží k zavedení nainstalovaného systému na rozdíl od zavedení původního jádra a ramdisku, které sloužily pro instalaci.

5.5.3 Instalace grafického prostředí, podpůrných programů a knihoven

Pro další manimulaci s QEMU obrazy bude použit program Virt-manager viz 4.3. Pro jeho správnou funkčnost je zapotřebí doinstalovat knihovu *libvirt-bin* a balíček *virtinst*. To se provede pomocí:

Příkaz 5.14: Instalace Virt-manager a podpůrného programového vybavení

```
root@ntblenovo:# apt-get install virt-manager libvirt-bin virtinst
```

kde ***libvirt*** je knihovna, která obsahuje moduly pro různé hypervizory a ***virtinst*** je software, obsahující sadu příkazů, které slouží pro manipulaci, vytváření a klonování virtuálních strojů, využívá knihovnu *libvirt*.

5.6 Aplikace

V této sekci je práce zaměřena na programování aplikace, která bude následně spuštěna v modulu CM-T43. Je požadováno, aby program uměl:

1. vyčíst data z konkrétního zařízení používající RS-485/RS-232/Ethernet,
2. zapsat data do konkrétního zařízení používající RS-485/RS-232/Ethernet,
3. později by měla aplikace fungovat jako daemon (program, který běží dlouhodobě na pozadí systému), který se bude volat přímo ze systému.

Aplikace bude využívat průmyslový protokol MODBUS (viz strana 19), který splňuje uvedené požadavky a bude psána v jazyce C.

5.6.1 Aplikace mb_master

Aplikace byla nazvána jako mb_master. Mb proto, že využívá protokol MODBUS a master proto, že se výsledný program v modulu bude chovat jako master z pohledu

sériové linky. Aby celý program fungoval správně a aby bylo možné jej zkompilovat, bylo nutné doinstalovat některé knihovny, které nebyly v systému nativně.

1. **modbus.h**
   modbus.h je knihovna obsahující protokol MODBUS a pro chod programu je nepostradatelná.
2. **argtable2.h**
   argtable2.h je knihovna sloužící k parsování příkazů z terminálu/příkazové řádky.

Příkaz 5.15: Instalace libmodbus

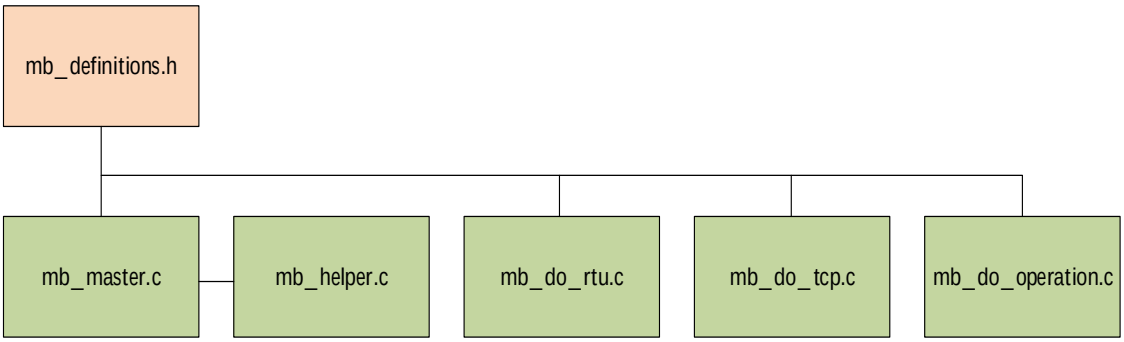
```
root@ntblenovo:# apt-get install libmodbus5
```

Příkaz 5.16: Instalace argtable2

```
root@ntblenovo:# apt-get install libargtable-dev
```

Samotný kód byl rozdělen do šesti dílčích částí. Rozdělení je popsáno níže a znázorněno graficky na obrázku 5.3.

1. **mb_definitions.h** – knihovna s definicemi funkcí,
2. **mb_helper.c** – pomocné testy,
3. **mb_do_operation.c** – výběr funkce MODBUSu,
4. **mb_do_rtu.c** – funkce pro rtu kontext,
5. **mb_do_tcp.c** – funkce pro tcp kontext,
6. **mb_master.c** – obsahuje třídu „main“.



Obr. 5.3: Rozdělení aplikace **mb_master**

V knihovně **mb_definitions.h** se pomocí příkazu „#define“ nadefinovaly hodnoty funkcí MODBUSu, chybových hlášení, návratových kódů a dalších konstant, které by nebylo vhodné definovat staticky kvůli vznikající nepřehlednosti v kódu. Po pádu programu se ukáže návratová hodnota chyby, pomocí které je snadnější určit na čem byl program nekorektně ukončen. Knihovna dále obsahuje prototypy funkcí, které jsou v programu použity. Celý obsah knihovny je k nalezení v přílohách viz strana 74.

C kód 5.17: knihovna mb_definitions.h

```
/**
 * CONST DEFINES
**/
                .
                .
                .
struct item_t {
       int fst;
       int snd;
};

uint16_t int2uint16(int i);
uint8_t int2uint8(int i);

int do_operation(modbus_t *ctx, int function, struct item_t item,
           int count, int count_count, int *data, int data_count);

int do_rtu(const char *device, int mode, int baudrate, char parity,
           char stopbits, int modbus_address, int function,
           struct item_t item, int count, int count_count,
           int *data, int data_count);

int do_tcp(struct arg_str *address, struct arg_int *port,
           int modbus_address, int function, struct item_t item,
           int count, int count_count, int *data, int data_count);

int tests();
```

Některé funkce MODBUSu používají speciální datové typy **uint8_t** a **uint16_t**. To kvůli zadávání bitů (8-bitové číslo) a registrů (16-bitové číslo). Proto se musela vytvořit funkce, která „ořeže“ klasický datový typ **int**(32-bitové číslo) na 8 popřípadě 16-bitové číslo. Kód byl vložen do samostného souboru **helper.c**.

C kód 5.18: helper.c

```c
uint16_t int2uint16(int i) {
        return (uint16_t) (i & 0x0000ffff);
}
uint8_t int2uint8(int i) {
        return (uint8_t) (i & 0x000000ff);
}
/**
 * tests
**/
int tests() {
        assert(int2uint16(0x12345678) == 0x5678);
    assert(int2uint8(0x12345678) == 0x78);
        return 1;
}
```

Pro zadávání jednotlivých funkcí MODBUSu byla vytvořena funkce **do_operation**, jejíž kód je samostatně umístěn do souboru **mb_do_operation.c**. Funkce obsahuje nový MODBUS kontext a deklarace proměnných, které jsou dále ve funkci **do_operation** použity.

C kód 5.19: Funkce do_operation

```c
int do_operation(modbus_t *ctx, int function, struct item_t item, int count,
                    int count_count, int *data, int data_count) {
                .
                .
                .
```

Funkce MODBUSu, které jsou v programu použity pomocí libmodbus jsou:

1. čtení bitů,
2. čtení vstupních bitů,
3. čtení registrů,
4. čtení vstupních registrů,
5. zápis jednoho bitu,
6. zápis jednoho registru,
7. zápis bitů,
8. zápis registrů,
9. zápis a čtení registrů.

U deváté funkce – **zápis a čtení registrů** zároveň nastal problém. Funkce 1-8 vykonávají v jeden moment právě jednu operaci (čtení nebo zápis). U každé funkce, se zadává parametr „-i nebo - -item“, což je parametr určující startovací adresu,

od které se začíná číst/zapisovat. Funkce **zápis a čtení registrů** však potřebuje tyto parametry dva, protože se určuje startovací položka pro čtení a startovací položka pro zápis. Byla proto stvořena struktura item_t. Díky ní je možné oddělit první a druhou položku. Všechny funkce, nepoužívající tuto druhou položku byly ošetřeny, aby nebylo parametr možné vůbec zadat.

C kód 5.20: Struktura item v knihovně mb_definitions.h

```
struct item_t {
       int fst;
       int snd;
};
```

Další částí aplikace je funkce na **do_rtu** , která určuje kontext sériové linky. Kód funkce je umístěn v souboru **mb_do_rtu.c**.

C kód 5.21: Funkce do_rtu

```
int do_rtu (const char *device, int mode, int baudrate, char parity,
            char stopbits, int modbus_address, int function,
            struct item_t item, int count, int count_count,
            int *data, int data_count) {
               .
               .
               .
```

Pro funkci **do_rtu** jsou POVINNÉ parametry:

- device – zařízení, se kterým se bude komunikovat,
- stopbits – počet stopbitů je povinným parametrem, protože bez něj nelze komunikaci navázat a nastavovat defaultní hodnotu nemá význam,
- modbus_addres – je povinný parametr právě jen pro rtu a udává adresu koncového „slave“ zařízení,
- function – výběr funkce pro danou relaci,
- item – počáteční adresa, od které se bude daná funkce provádět,
- count – počet jednotek (bitů/registrů), které se mají vyčíst (pouze pro **čtecí** funkce a pro funkci **zápis a čtení registrů**),
- data – počet jednotek (bitů/registrů), které se mají zapsat (pouze pro **zapisovací** funkce a pro funkci **zápis a čtení registrů**).

Pro funkci **do_rtu** jsou VOLITELNÉ parametry:

- mode – parametr mode určuje mód sériové linky, buď RS-232 nebo RS-485, nastavovat ho jako povinný parametr nemá význam, protože ne všechny embedded zařízení musí být vybaveny těmito sériovými porty, tudíž ani defaultní

hodnota není dále nastavena,

- baudrate – nastaveno defaultně na 9600 baudů,
- parity – parita – nastavena defaultně žádná parita.

V kódu je implementován přehled, který informuje uživatele/systém, co zadal.

C kód 5.22: Přehled do_rtu

```
        printf ("Serial device:      %s\n", device);
        printf ("Serial mode:        %d\n", mode);
        printf ("Baud rate:          %d\n", baudrate);
        printf ("Number of stopbits: %d\n", stopbits);
        printf ("Type of parity:     %d\n", parity);
        printf ("Modbus address:     %d\n", modbus_address);
        printf ("Type of function:   0x%02x", function);
        printf ("Item number (first): %d\n", item.fst);
        printf ("Item number (second): %d\n", item.snd);
        printf ("Items count requested: %d\n", count);
        printf ("Data count supplied: %d\n", data_count);
        printf ("*****************************************\n");
```

Stejným způsobem je vytvořena funkce **do_tcp** v souboru **mb_do_tcp.c**.

C kód 5.23: Funkce do_tcp

```
int do_tcp (struct arg_str *address, struct arg_int *port,
            int modbus_address, int function, struct item_t item,
            int count, int count_count, int *data, int data_count) {
               .
               .
               .
```

Pro funkci **do_tcp** jsou POVINNÉ parametry:

- function – výběr funkce pro danou relaci,
- item – počáteční adresa, od které se bude daná funkce provádět,
- count – počet jednotek(bitů/registrů), které se mají vyčíst (pouze pro **čtecí** funkce a pro funkci **zápis a čtení registrů**),
- data – počet jednotek(bitů/registrů), které se mají zapsat (pouze pro **zapisovací** funkce a pro funkci **zápis a čtení registrů**).

Pro funkci **do_tcp** jsou VOLITELNÉ parametry:

- address – IP adresa je nastavena defaultně na „0.0.0.0“, tudíž naslouchá na IP všech adresách,
- port – nastaven defaultně na port 1502 (MODBUS port),

- modbus adress – tento parametr má pro tcp význam pouze tehdy, když bude použito nějaké síťové zařízení, které bude dále používat síť sériové linky.

C kód 5.24: Přehled do_tcp

```
        printf ("IP address:          %s\n", IP_chaddr);
        printf ("IP port:             %d\n", IP_chport);
        printf ("Modbus address:      %d\n", modbus_address);
        printf ("Type of function:    0x%02x\n", function);
        printf ("Item number (first): %d\n", item.fst);
        printf ("Item number (second): %d\n", item.snd);
        printf ("Items count requested: %d\n", count);
        printf ("Data count supplied: %d\n", data_count);
        printf ("*********************************************\n");
```

Poslední částí aplikace je soubor **mb_master.c**, který obsahuje třídu „main“. Vůbec nejdůležitější parametr celého programu je výběr vrstvy. Tento parametr rozhoduje, zda bude volána funkce **do_rtu** nebo **do_tcp**. Je zde podmínkou ošetřeno, aby aplikace při nezadání tohoto parametru byla bez výjimky ukončena, protože by dále neměla význam.

C kód 5.25: Ošetření chybějícího parametru pro výběr vrstvy

```
            if (layer->count == 0) {
            exitcode = EXIT_NO_LAYER_ERR;
            fprintf(stderr, "Exit: (%d) - No layer selected!\n", exitcode);
            goto exit;
    }
```

Dále zde kód obsahuje struktury knihovny **argtable2.h**, pomocí níž byla stvořena nabídka „- -help“. Zejména kvůli tomu, že aplikace má být univerzální. Nelze předpokládat, že ji bude používat pouze některý z automatizovaných systémů, ale je možné, že najde koncového zájemce i mezi uživateli. Nabídka „- -help“ je k nalezení v přílohách viz strana 76 .

Celková kompilace programu se v systému Linux prováděla na základě souboru **makefile**

Příkaz 5.26: makefile

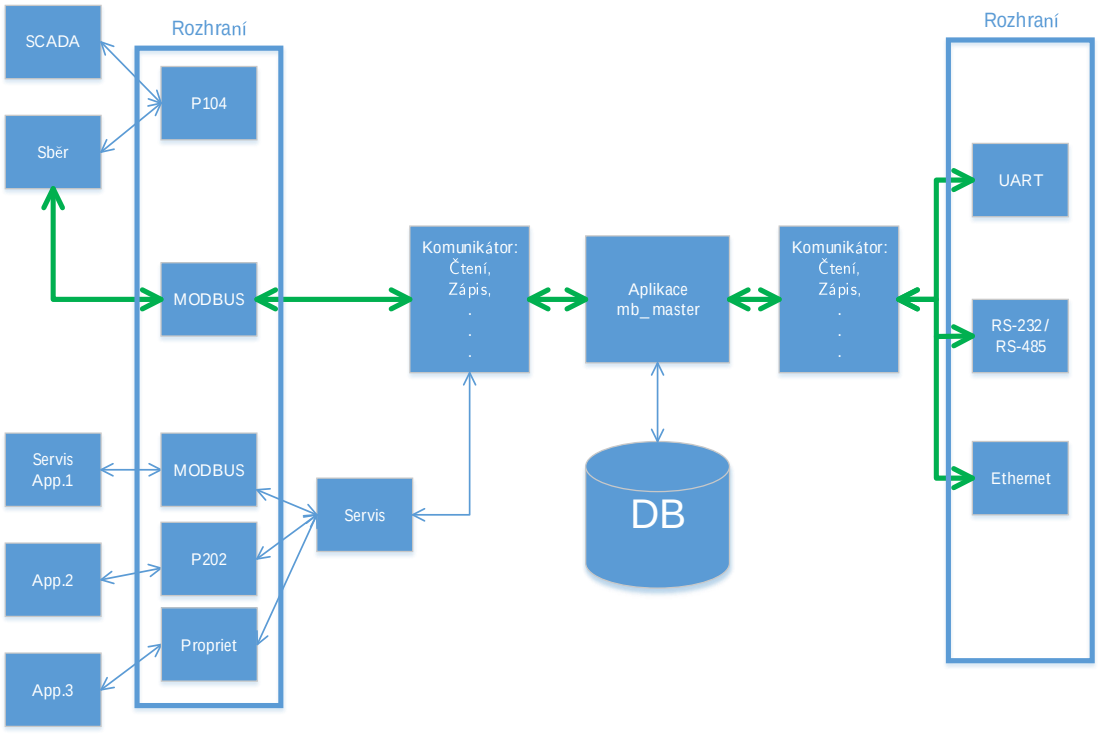
```
all: mb_master
mb_master: mb_master.c
	gcc -g --std=c99 -l argtable2 -l modbus -Wall
	mb_master.c mb_helper.c mb_do_operation.c mb_do_rtu.c
	mb_do_tcp.c -o mb_master
```

a příkazu:

Příkaz 5.27: make

```
root@ntblenovo:# make
```

5.6.2 Architektura



Obr. 5.4: Architektura použití aplikace

Na obrázku 5.4 je znázorněn komunikační řetězec vytvořeného řešení. Princip komunikace vede skrze MODBUS a jeho funkce. Blok „Komunikátor: čtení, zápis“

reprezentuje výběr funkce MODBUSu. Blok „Aplikace mb_master“ je komunikátorem mezi jednotlivými koncovými zařízeními a uživatelem či systémem. Příkaz může přijít od SCADY i od uživatele a musí obsahovat skupinu parametrů (viz níže), která rozhodne jaká operace se provede, na jakém rozhraní atd. Aplikace **mb_master** byla testována pomocí aplikace „diagslave“, která je zdarma dostupná na webových stránkách `http://www.modbusdriver.com/diagslave.html`. Aplikace byla testována v režimu sériové linky i Ethernetu.

Sériová linka

Za pomocí dvou, navzájem propojených, převodníků USB->RS-485, viz obrázek 5.5, byly testovány veškeré parametry, nastavitelné pro sériovou linku (baudrate, parita, stopbity atd.). Následujícím příkazem byla ověřena jejich přítomnost v systému.

Příkaz 5.28: Výpis adresáře /dev/ttyUSB*

```
root@ntblenovo:~# ls /dev/ttyUSB*
/dev/ttyUSB0 /dev/ttyUSB1
```

Obr. 5.5: Převodník z USB na RS-485

Následujícím příkazem byla aplikace „diagslave“ s parametry pro výběr sériové linky, žádnou paritou, baudratem 9600 baudů a jedním stopbitem. Pro komunikaci bylo nutné nastavit tyto parametry na obou stranách stejně.

Příkaz 5.29: Příkaz na straně aplikace diagslave

```
root@ntblenovo:~/testing/linux# ./diagslave -m rtu -b9600 -p none -s1
/dev/ttyUSB1
```

```
diagslave 2.12 - FieldTalk(tm) Modbus(R) Diagnostic Slave Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration: address = -1, master activity t/o = 3.00
Serial port configuration: /dev/ttyUSB1, 9600, 8, 1, none

Server started up successfully.
Listening to network (Ctrl-C to stop)
. . .
```

Příkaz 5.30: Příkaz na straně aplikace mb_master – zápis bitů

```
root@ntblenovo:~/mb_master_app# ./mb_master -lr -d /dev/ttyUSB0 -i0
-b9600 -D1 -D1 -D0 -D1 -a1 -f0x0f -s1
do_rtu
Serial device:       /dev/ttyUSB0
Serial mode:         0
Baud rate:           9600
Number of stopbits:  1
Type of parity:      0
Modbus address:      1
Type of function:    0x0fItem number (first): 0
Item number (second): 0
Items count requested: 0
Data count supplied: 4
**********************************************************
None parity
WRITE BITS
Data value: 1
Data value: 1
Data value: 0
Data value: 1
Result of writing: 1
Result of writing: 1
Result of writing: 0
Result of writing: 1
```

Příkaz 5.31: Reakce aplikace diagslave

```
Slave  1: writeCoils from 1, 4 references
```

Příkaz 5.32: Příkaz na straně aplikace mb_master – čtení bitů

```
root@ntblenovo:~/mb_master_app# ./mb_master -lr -d /dev/ttyUSB0 -i0
-b9600 -c6 -a1 -f0x01 -s1
do_rtu
Serial device:       /dev/ttyUSB0
Serial mode:         0
Baud rate:           9600
Number of stopbits:  1
Type of parity:      0
Modbus address:      1
Type of function:    0x01Item number (first): 0
Item number (second): 0
Items count requested: 6
Data count supplied: 0
*********************************************************
None parity
READ BITS
Result of reading: 1
Result of reading: 1
Result of reading: 0
Result of reading: 1
Result of reading: 0
Result of reading: 0
```

Příkaz 5.33: Reakce aplikace diagslave

```
Slave  1: writeCoils from 1, 4 references
Slave  1: readCoils from 1, 6 references
```

Aplikace **mb_master** všemi testy pomocné aplikace **diagslave** prošla bez komplikací, a tudíž je plně schopna operovat na sériové lince.

Ethernet

U Ethernetu probíhalo testování podobně jako pro sériovou linku, ale s potřebou zadání výrazně méně parametrů. Aplikace **mb_master** má předdefinované hodnoty pro port a pro IP adresu (0.0.0.0:1502), které tedy pro testování není nutné zadávat.

Parametry je možné změnit pomocí **-I** nebo **- -IP_addr** pro změnu IP adresy a **-P** nebo **- -IP_port** pro změnu portu.

Příkaz 5.34: Příkaz na straně aplikace diagslave

```
root@ntblenovo:~/testing/linux# ./diagslave -m tcp -p1502
diagslave 2.12 - FieldTalk(tm) Modbus(R) Diagnostic Slave Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration: address = -1, master activity t/o = 3.00
TCP configuration: port = 1502, connection t/o = 60.00

Server started up successfully.
Listening to network (Ctrl-C to stop)
. . .
```

Příkaz 5.35: Příkaz na straně aplikace mb_master – zápis registrů

```
root@ntblenovo:~/mb_master_app# ./mb_master -lt -i0 -f0x10 -D9999
-D358 -D245
do_tcp
IP address:          0.0.0.0
IP port:             1502
Modbus address:      0
Type of function:    0x10
Item number (first): 0
Item number (second): 0
Items count requested: 0
Data count supplied: 3
*****************************************************
WRITE REGISTERS
Data value: 9999
Data value: 358
Data value: 245
Result of writing: 270f
Result of writing: 166
Result of writing: f5
```

Příkaz 5.36: Reakce aplikace diagslave

```
validateMasterIpAddr: accepting connection from 127.0.0.1
Slave  0: writeHoldingRegisters from 1, 3 references
```

Příkaz 5.37: Příkaz na straně aplikace mb_master – čtení registrů

```
root@ntblenovo:~/mb_master_app# ./mb_master -lt -i0 -f0x03 -c4
do_tcp
IP address:          0.0.0.0
IP port:             1502
Modbus address:      0
Type of function:    0x03
Item number (first): 0
Item number (second): 0
Items count requested: 4
Data count supplied: 0
*****************************************************
READ REGISTERS
Result of reading: 270f
Result of reading: 166
Result of reading: f5
Result of reading: 0
```

Příkaz 5.38: Reakce aplikace diagslave

```
validateMasterIpAddr: accepting connection from 127.0.0.1
Slave  0: writeHoldingRegisters from 1, 3 references
Slave  0: readHoldingRegisters from 1, 4 references
```

Z důvodu komplikace v podobě dvojitého parametru startovací položky (**-i** nebo **- -item**) se provedlo také testování funkce **zápis a čtení registrů** .

Příkaz 5.39: RPříkaz na straně aplikace mb_master – zápis a čtení registrů

```
root@ntblenovo:~/mb_master_app# ./mb_master -lt -i0 -f0x17 -i1 -D9999
-D358 -D245 -c4
do_tcp
IP address:          0.0.0.0
IP port:             1502
Modbus address:      0
Type of function:    0x17
```

```
Item number (first): 0
Item number (second): 1
Items count requested: 4
Data count supplied: 3
*****************************************************
Selected WRITE AND READ REGISTERS
Data value: 9999
Data value: 358
Data value: 245
Result of writing: 270f
Result of writing: 166
Result of writing: f5
Result of reading 166
Result of reading f5
Result of reading 0
Result of reading 0
```

Příkaz 5.40: Reakce aplikace diagslave

```
validateMasterIpAddr: accepting connection from 127.0.0.1
Slave  0: writeHoldingRegisters from 1, 3 references
Slave  0: readHoldingRegisters from 2, 4 references
```

Aplikace **mb_master** všemi testy pomocné aplikace **diagslave** také zde prošla bez problémů. Je tedy plně schopna operovat i na Ethernetu. Aplikace **mb_master** byla při testování stabilní a díky ošetření chybného zadání parametrů (zadání neexistujícího, nesprávného parametru pro dané rozhraní, či naopak nezadání potřebného parametru, nenalezení daného zařízení apod.) je připravená k použití.

6 ZÁVĚR

Bakalářská práce se v teoretické části zabývala Internetem věcí a M2M komunikací (historie, způsoby využití, komunikační protokoly apod.). Dále byly popsány virtualizační platformy, zejména emulační program QEMU.

V praktické části, kde byl použit právě emulační program QEMU, byla nejdříve emulována architektura procesoru a následně byl přizpůsoben obraz hardwarovým požadavkům modulu CM-T43 od společnosti Texas Instruments.

Dále byla praktická část práce zaměřena na programování aplikace mb_master, která vychází z protokolu MODBUS a funguje jako zprostředkovatel zápisu nebo čtení informací jednotlivých čidel, připojených k sériové lince nebo Ethernetu. Při programování se objevilo několik komplikací se zadáváním parametrů, zejména u operace „zápis a čtení registrů“. V této operaci jsou zapotřebí dva parametry pro zvolení počáteční položky, z nichž jedna určuje od jaké položky se začne číst a druhá od které se začne zapisovat. Tohoto bylo docíleno vytvořením speciální struktury, která tyto položky rozlišuje. Všechny ostatní funkce MODBUSu musely být od tohoto druhého parametru ošetřeny. Dalším problémem byl výběr módu sériové linky (RS-232 nebo RS-485). Nastavovat parametr módu jako povinný nemělo význam, protože ne každé vestavěné zařízení disponuje vývodem pro tato rozhraní. Tento parametr je tedy volitelný, protože v průmyslu se ne zřídka používají převodníky, které to řeší na bázi hardware, a tudíž to není třeba řešit pomocí programového vybavení nebo pomocí úpravy jádra systému.

Přenos dat byl otestován pomocí speciálního převodníku z USB na RS-485 a také pomocných testů, které nabízí knihovna MODBUSu.

V budoucnu by se dala aplikace rozšířit do podoby daemona, který by bylo možné volat přímo ze systému.

LITERATURA

[1] AL-FUQAHA, Ala, Mohsen GUIZANI, Mehdi MOHAMMADI, Mohammed ALEDHARI a Moussa AYYASH. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys [online]. 2015, 17(4): 2347-2376 [cit. 2015-11-29]. DOI: 10.1109/COMST.2015.2444095. ISSN 1553-877x. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7123563

[2] Internet věcí (Internet of Things). KODYS. INTERNET VĚCÍ (INTERNET OF THINGS) [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.kodys.cz/internet-of-things.html

[3] DANEELS, A. a W. SALTER. WHAT IS SCADA? [online]. Italy, 1999 [cit. 2015-11-29]. Dostupné z: http://cds.cern.ch/record/532624/files/mc1i01.pdf

[4] NING CAI, JIDONG WANG a XINGHUO YU. SCADA system security: Complexity, history and new developments. 2008 6th IEEE International Conference on Industrial Informatics [online]. IEEE, 2008, (1): 569-574 [cit. 2015-11-29]. DOI: 10.1109/INDIN.2008.4618165. ISBN 978-1-4244-2170-1. ISSN 1935-4576. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4618165

[5] HAYASHI, H., Y. TAKABAYASHI, H. TSUJI a M. OKA. Rapidly increasing application of Intranet technologies for SCADA (supervisory control and data acquisition system). IEEE/PES Transmission and Distribution Conference and Exhibition [online]. IEEE, 2002, (1): 22-25 [cit. 2015-11-29]. DOI: 10.1109/TDC.2002.1178254. ISBN 0-7803-7525-4. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1178254

[6] Internet of Things Done Wrong Stifles Innovation. PALERMO, Frank. Informationweek [online]. 2014 [cit. 2015-11-28]. Dostupné z: http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/internet-of-things-done-wrong-stifles-innovation/a/d-id/1279157

[7] History of the Internet of Things. Postcapes [online]. 2013 [cit. 2015-11-28]. Dostupné z: http://postscapes.com/internet-of-things-history

[8] That 'Internet of Things' Thing: In the real world, things matter more than ideas. ASHTON, Kevin. Http://www.rfidjournal.com/ [online]. 2009 [cit. 2015-11-28]. Dostupné z: http://www.rfidjournal.com/articles/pdf?4986

[9] SURESH, P., J. Vijay DANIEL, V. PARTHASARATHY a R. H. ASWATHY. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. 2014 International Conference on Science Engineering and Management Research (ICSEMR) [online]. IEEE, 2014, (1): 1-8 [cit. 2015-11-29]. DOI: 10.1109/ICSEMR.2014.7043637. ISBN 978-1-4799-7613-3. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7043637

[10] An Introduction to the Internet of Things (IoT): Part 1. of "The IoT Series". LOPEZ RESEARCH LLC. Cisco [online]. San Francisco, CA 94123, 2013 [cit. 2015-11-28]. Dostupné z: http://www.cisco.com/web/solutions/trends/iot/introduction_to_IoT_november.pdf

[11] Constrained Application Protocol for Internet of Things. CHEN, Xi. Constrained Application Protocol for Internet of Things [online]. 2014 [cit. 2015-11-29]. Dostupné z: http://www1.cse.wustl.edu/~jain/cse574-14/ftp/coap/

[12] ČÍKA, Petr. Multimediální služby [online]. Vysoké učení technické v Brně, 2012 [cit. 2015-11-29]. ISBN 978-80-214-4443-0.

[13] MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3. *Http://www.modbus.org/* [online]. 2012 [cit. 2016-05-23]. Dostupné z: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[14] YAN ZHANG, RONG YU, SHENGLI XIE, WENQING YAO, YANG XIAO a M GUIZANI. Home M2M networks: Architectures, standards, and QoS improvement. IEEE Communications Magazine [online]. 2011, 49(4): 44-52 [cit. 2015-11-29]. DOI: 10.1109/MCOM.2011.5741145. ISSN 0163-6804. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5741145

[15] CHEN, Min, Jiafu WAN, Sergio GONZALEZ, Xiaofei LIAO a Victor C.M. LEUNG. A Survey of Recent Developments in Home M2M Networks. IEEE Communications Surveys [online]. 2014, 16(1): 98-114 [cit. 2015-11-29]. DOI: 10.1109/SURV.2013.110113.00249. ISSN 1553-877x. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6674156

[16] Machine to Machine. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2015-11-29]. Dostupné z: https://en.wikipedia.org/wiki/Machine_to_machine

[17] GSM-Modul M1. Computerwoche [online]. 1996 [cit. 2015-11-29]. Dostupné z: http://www.computerwoche.de/a/gsm-modul-m1,1105147

[18] Bytové vodoměry s bezdrátovým přenosem dat BONEGA. *Http://www.bonega.cz* [online]. 2013 [cit. 2016-05-27]. Dostupné z: http://www.bonega.cz/vodomery/

[19] ANTÓN-HARO, Carles a Mischa DOHLER. Machine-to-machine (M2M) Communications: Achitecture, Performance and Applications. Cambridge, CB22 3HJ, UK: Woodhead Publishing Limited is an imprint of Elsevier, 2015. ISBN 978-1-78242-102-3.

[20] HOSEK, Jiri, Pavel MASEK, Dominik KOVAC a Franz KROPFL. M2M gateway: The centerpiece of future home. 2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) [online]. IEEE, 2014, : 190-197 [cit. 2015-11-29]. DOI: 10.1109/ICUMT.2014.7002101. ISBN 978-1-4799-5291-5. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7002101

[21] NGUYEN, Minh-Son a Quan LE-TRUNG. Low-power and cost-effective wifi sensor motes for wireless embedded Internet applications. 2014 International Conference on Advanced Technologies for Communications (ATC 2014) [online]. IEEE, 2014, : 441-445 [cit. 2015-11-29]. DOI: 10.1109/ATC.2014.7043428. ISBN 978-1-4799-6956-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7043428

[22] Low Energy [online]. BLUETOOTH. [cit. 2015-11-29]. Dostupné z: http://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

[23] Key Issues in Perspective: SMART METERS and DATA ACCURACY [online]. A SMARD GRID. A POWERFUL FUTURE. 2012 [cit. 2015-11-29]. Dostupné z: https://www2.dteenergy.com/wps/wcm/connect/eb0a40f8-c578-46cc-b20a-cd4c4d083826/AMI-Smart+Meters+and+Data+Accuracy.pdf?MOD=AJPERES

[24] Smart meters [online]. EMFS.INFO. [cit. 2015-11-29]. Dostupné z: http://www.emfs.info/sources/meters/smart/

[25] FENG, Shaochong, Yanqiang DI a Yuanchang ZHU. Design and Realization of Virtualization-Based Basic Simulation Computing Resource Management Platform. 2010 Fifth Annual ChinaGrid Conference [online]. IEEE, 2010, (18): 271-274 [cit. 2015-11-28]. DOI: 10.1109/ChinaGrid.2010.18. ISBN 978-1-4244-7543-8. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5562868

[26] Virtualization Technology. Virtualization [online]. 2015 [cit. 2015-11-26]. Dostupné z: http://www.vmware.com/virtualization

[27] IBM and HP virtualization: A comparative study of UNIX virtualization on both platforms. MILBERG, Ken. IBM and HP virtualization [online]. 2009 [cit. 2015-11-28]. Dostupné z: http://www.ibm.com/developerworks/aix/library/au-aixhpvirtualization

[28] Memory price. MCCALLUM, John C. Jcmit [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.jcmit.com/memoryprice.htm

[29] Trends in the cost of computing. AI IMPACTS. AI Impacts [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://aiimpacts.org/trends-in-the-cost-of-computing/

[30] Average Historic Price of RAM. Statistic Brain Research Institute [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.statisticbrain.com/average-historic-price-of-ram/

[31] Historical cost of computer equipment. Google [online]. 2006 [cit. 2015-11-28]. Dostupné z: http://answers.google.com/answers/threadview/id/771396.html

[32] RUEST, Danielle a Nelson RUEST. Virtualizace: podrobný průvodce. Vyd. 1. Brno: Computer Press, 2010, 408 s. ISBN 978-80-251-2676-9.

[33] AJILA, Samuel A. a Omhenimhen IYAMU. Efficient Live Wide Area VM Migration with IP Address Change Using Type II Hypervisor [online]. San Francisco, California, USA, 2013 [cit. 2015-11-28]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6642495&tag=1. Akademická práce. Carleton University 1125 Colonel By Drive, Ottawa, ON Canada.

[34] Virtualizace (opět) jako paradigma pro datacentra. GÖSSEL, František. Novell [online]. 2007 [cit. 2015-11-28]. Dostupné z: http://www.novell.cz/cs/aktuality/technicke-clanky/virtualizace-opet-jako-paradigma-pro-datacentra.html

[35] MATYSKA, L. Techniky virtualizace počítačů (2): Zpravodaj ÚVT MU. Techniky virtualizace počítačů (2) [online]. 2007, XVII(3): 9-12 [cit. 2015-11-28]. ISSN 1212-0901. Dostupné z: http://webserver.ics.muni.cz/bulletin/articles/545.html

[36] KOVARI, A. a P. DUKAN. KVM & OpenVZ virtualization based IaaS Open Source Cloud Virtualization Platforms: OpenNode, Proxmox VE [online]. Subotica, Serbia, 2012 [cit. 2015-11-28]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6339540&tag=1. Akademická práce. College of Dunaújváros/Institute of Informatics.

[37] PORTNOY, Matthew. Virtualization essentials. 1. st ed. Indianapolis: Wiley Pub., Inc., 2012. ISBN 11-181-7671-5.

[38] WOLF, Chris a Erick M. HALTER. Virtualization: from the desktop to the enterprise. 1.st ed. New York: Distributed in U.S. by Springer-Verlag New York, 2005. ISBN 978-159-0594-957.

[39] GOLDEN, Bernard a Erick M. HALTER. Virtualization for dummies: from the desktop to the enterprise. 1.st ed. Hoboken, N.J.: Wiley, 2008. ISBN 04-701-4831-4.

[40] Virtualizace desktopů: Budoucnost firemního desktopu. T-SYSTEMS CZECH REPUBLIC A.S. Virtualizace desktopů [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.t-systems.cz/produkty-a-reseni/virtu-ln-desktopy/604918_1/blobBinary/pdf-ps.pdf

[41] VMware se již pošesté lídrem v hodnocení Magic Quadrant pro oblast virtualizační infrastruktury serverů x86 [online]. 2015 [cit. 2016-05-29]. Dostupné z: http://bit.ly/1WUgwAw

[42] VMware Workstation Pro. VMware. VMware Workstation Features, Multiple OS, Run Linux on Windows [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.vmware.com/cz/products/workstation/features.html

[43] VSphere ESXi Bare-Metal Hypervisor. VMware. VMware ESXi [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.vmware.com/cz/products/esxi-and-esx/overview

[44] Oracle VM VirtualBox. VirtualBox [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://www.virtualbox.org/

[45] TechNet Blog CZ/SK. TECHNET. Microsoft Hyper-V [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://blogs.technet.com/b/technetczsk/p/microsoft-hyper-v.aspx

[46] Microsoft Hyper-V Server 2012 R2 a Hyper-V Server 2012. TECHNET. Microsoft Hyper-V Server 2012 R2 a Hyper-V Server 2012 [online]. 2015 [cit.

2015-11-28]. Dostupné z: https://technet.microsoft.com/cs-cz/library/hh833684.aspx

[47] Přehled technologie Hyper-V. Přehled technologie Hyper-V [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://technet.microsoft.com/cs-cz/library/hh831531.aspx

[48] What's new in Hyper-V on Windows Server 2016 Technical Preview. TECHNET. What's new in Hyper-V on Windows Server 2016 Technical Preview [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://technet.microsoft.com/en-us/library/dn765471.aspx

[49] KVM. LINUX-KVM. Kernel Virtual Machine [online]. 2008 [cit. 2015-11-28]. Dostupné z: http://www.linux-kvm.org/page/Main_Page

[50] KVM - Debian Wiki. DEBIAN. KVM [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://wiki.debian.org/KVM

[51] Overview - Linux-VServer. Overview [online]. 2013 [cit. 2015-11-28]. Dostupné z: http://linux-vserver.org/Overview

[52] Virtualizace v Linuxu – Wikiknihy. WIKIKNIHY. Virtualizace v Linuxu [online]. 2009 [cit. 2015-11-28]. Dostupné z: https://cs.wikibooks.org/wiki/Virtualizace_v_Linuxu

[53] OpenVZ Virtuozzo Containers Wiki. OpenVZ [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://openvz.org/Main_Page

[54] USENIX '05 — Technical Paper, FREENIX Track. BELLARD, Fabrice. QEMU, a Fast and Portable Dynamic Translator [online]. 2005 [cit. 2015-11-28]. Dostupné z: https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/bellard/bellard_html/

[55] Libvirt: The virtualization API. The virtualization API [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://libvirt.org/index.html

[56] QEMU/Installing QEMU - Wikibooks, open books for an open world. QEMU/Installing QEMU [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://en.wikibooks.org/wiki/QEMU/Installing_QEMU

[57] Chroot - Debian Wiki. Chroot [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://wiki.debian.org/cs/chroot

[58] QEMU. QEMU open source processor emulator [online]. 2015 [cit. 2015-11-29]. Dostupné z: http://wiki.qemu.org/Main_Page

[59] QEMU Emulator User Documentation. QEMU Emulator User Documentation [online]. 2005 [cit. 2015-11-29]. Dostupné z: http://qemu.weilnetz.de/qemu-doc.html

[60] K.P, Dileep, A. RAGHAVENDRA RAO, Suman M, Devesh G a S.V. SRIKANTH. Verification of Linux Device Drivers using Device Virtualization [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7100338&tag=1

[61] YU, Yang, Qixian CAI a Sen GUO. A new construction method of common embedded cross compiler tool based on newlib. 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems [online]. IEEE, 2010, (3): 817-819 [cit. 2015-12-15]. DOI: 10.1109/ICICISYS.2010.5658328. ISBN 978-1-4244-6582-8. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5658328

[62] Debian: The universal operating system. Debian [online]. 2015 [cit. 2015-12-15]. Dostupné z: https://www.debian.org/

[63] List of ARM microarchitectures. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2015-12-15]. Dostupné z: https://en.wikipedia.org/wiki/List_of_ARM_microarchitectures

[64] Debian: ArmHardFloatPort [online]. 2015 [cit. 2015-12-14]. Dostupné z: https://wiki.debian.org/ArmHardFloatPort

[65] Debian: Instalace systému Debian GNU/Linux 8 na architektuře armhf [online]. 2015 [cit. 2015-12-14]. Dostupné z: https://www.debian.org/releases/stable/armhf/ch02s01.html.cs

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADU	Application Data unit
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DCPS	Data-Centric Publish-Subscribe
DDS	Data Distribution Service
DLRL	Data-Local Reconstruction Layer
EPC	Electronic Product Code
ETSI	European Telecommunications Standards Institute
GNOME	GNU Network Object Model Environment
GPL	General Public License
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HMI	Human Machine Interface
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
HVAC	Heating Ventilation and Air-Conditioning
H2H	Human-to-Human
Chroot	Change root
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IoT	Internet of things
ISM	Industrial, Scientific and Medical

ISO/OSI	International Standards Organization Open Systems Interconnection
JVM	Java Virtual Machine
KDE	K Desktop Environment
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LTE	Long-Term Evolution
MIT	Massachusetts Institute of Technology
MQTT	Message Queue Telemetry Transport
MTC	Machine-Type-Communication
NASA	National Aeronautics and Space Administration
NBD	Network Block Device
NFC	Near Field Communication
OS	Operační systém
PAN	Personal Area Network
PDU	Protocol Data Unit
PC	Personal Computer
PLC	Programmable Logic Controller
QEMU	Quick Emulator
QoS	Quality of Service
QR	Quick Response
RAM	Random Acces Memory
REST	Representational State Transfer
RFID	Radio Frequency Identification
RLP	Radio Link Protocol
SCADA	Supervisory Control and Data Acquisition

SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
USB	Universal Serial Bus
VMM	Virtual Machine Monitor
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless LAN
WMI	Windows Management Instrumentation
WPAN	Wireless Personal Area Network
W3C	World Wide Web Consortium
XFCE	XForms Common Environment
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks

SEZNAM PŘÍLOH

A PŘÍLOHY

A.1 Obsah hnihovny mb_definitions.h

C kód A.1: knihovna mb_definitions.h

```c
#include <argtable2.h>

#ifndef MB_DEFINITIONS_H
#define MB_DEFINITIONS_H

/**
 * CONST DEFINES
**/
#define DATA_BITS                                (8)
#define MAX_REGISTERS                            (0x7d0)
#define MAX_BITS                                 (0x7d0)
#define DEFAULT_IP_ADDR                          ("0.0.0.0")
#define DEFAULT_IP_PORT                          (1502)
#define IP_ADDR_LEN                              (15)
#define RESPONSE_sec                             (1)
#define RESPONSE_usec                            (20000)

/**
 * ERR DEFINES
**/
#define ERR_OK                                   (1)
#define ERR_TRUE_OR_FALSE                        (-1)
#define ERR_TOO_BIG                              (-2)
#define ERR_WRONG_DATA_COUNT                     (-3)
#define ERR_INVALID_NUMBER_OF_PARITY             (-4)
#define ERR_INVALID_NUMBER_OF_STOPBITS           (-5)
#define ERR_NEW_CTX_FAILED                       (-6)
#define ERR_SERIAL_LINE                          (-7)
#define ERR_DIFFERENCE_SET_AND_ACTUAL_MODE       (-8)
#define ERR_CONNECT_FAILED                       (-9)
#define ERR_INVALID_FUNCTION_NUMBER              (-10)
#define ERR_TOO_MANY_DATACOUNT                   (-11)
#define ERR_INVALID_FUNCTION_PARAMETER           (-12)
#define ERR_UKNOWN_RTU_MODE                      (-13)
#define ERR_MISSING_PARAMETER                    (-14)

#define ERR_READING_BITS                         (-101)
#define ERR_READING_INPUT_BITS                   (-102)
#define ERR_READING_REGISTERS                    (-103)
```

```
#define ERR_READING_INPUT_REGISTERS             (-104)
#define ERR_WRITING_BIT                         (-105)
#define ERR_WRITING_REGISTER                    (-106)
#define ERR_WRITING_BITS                        (-107)
#define ERR_WRITING_REGISTERS                   (-108)
#define ERR_WRITING_AND_READING_REGISTERS       (-109)

/**
 * EXITCODE DEFINES
**/
#define EXIT_OK                                 (0)
#define EXIT_INSUFFICIENT_MEMORY_ERR            (1)
#define EXIT_PARSING_ERR                        (2)
#define EXIT_NO_LAYER_ERR                       (3)
#define EXIT_INVALID_PARAMETER_ERR              (4)
#define EXIT_MISSING_PARAMETER_ERR              (5)
#define EXIT_TCP_LAYER_ERR                      (6)
#define EXIT_RTU_LAYER_ERR                      (7)
#define EXIT_UKNOWN_LAYER_ERR                   (8)
/*
 * MB FUNCTIONS DEFINES
**/
#define MB_F_READ_BITS                          (0x01)
#define MB_F_READ_INPUT_BITS                    (0x02)
#define MB_F_READ_REGISTERS                     (0x03)
#define MB_F_READ_INPUT_REGISTERS               (0x04)
#define MB_F_WRITE_BIT                          (0x05)
#define MB_F_WRITE_REGISTER                     (0x06)
#define MB_F_WRITE_BITS                         (0x0F)
#define MB_F_WRITE_REGISTERS                    (0x10)
#define MB_F_WRITE_AND_READ_REGISTERS           (0x17)

struct item_t {
        int fst;
        int snd;
};

uint16_t int2uint16(int i);
uint8_t int2uint8(int i);

int do_operation(modbus_t *ctx, int function, struct item_t item,
           int count, int count_count, int *data, int data_count);

int do_rtu(const char *device, int mode, int baudrate, char parity,
           char stopbits, int modbus_address, int function,
           struct item_t item, int count, int count_count,
```

```
            int *data, int data_count);

int do_tcp(struct arg_str *address, struct arg_int *port,
            int modbus_address, int function, struct item_t item,
            int count, int count_count, int *data, int data_count);

int tests();

#endif /* MB_DEFINITIONS_H */
```

A.2 Nabídka „- - help“

Příkaz A.2: Výpis nabídky „- - help“

```
./mb_master --help
Usage: mb_master
--help                          Print this help and exit.
--version                       Print version information and exit.

-l, --layer=<string>            A layer to use: (r)tu or (t)cp.
                                REQUIRED initialization parameter!
-a, --modbus-address=<int>      Address of modbus slave.
                                REQUIRED parameter for rtu and OPTIONAL
                                for tcp!
-i, --item=<int>                Adress of first item.
                                REQUIRED parameter for both of layers.
                                For 0x17 function is first -i parameter
                                for writing and second -i for reading.
-c, --count=<int>               Number of items from reading function.
                                REQUIRED parameter for both of layers if
                                is used "read or read/write" functions:
                                0x01, 0x02, 0x03, 0x04 or 0x17
-D, --data=<int>                Data to write
                                REQUIRED parameter for both of layers if
                                is used "write or read/write" functions:
                                0x05, 0x06, 0x0F, 0x10 or 0x17
-f, --function=<int>            MODBUS functions:
                                REQUIRED parameter for both of layers.
                                  -f 0x01 is "modbus_read_bits"
                                  -f 0x02 is "modbus_read_input_bits"
                                  -f 0x03 is "modbus_read_registers"
```

```
                                -f 0x04 is "modbus_read_input_
                                   registers"
                                -f 0x05 is "modbus_write_bit"
                                -f 0x06 is "modbus_write_register"
                                -f 0x0F is "modbus_write_bits"
                                -f 0x10 is "modbus_write_registers"
                                -f 0x17 is "modbus_write_and_read_
                                   registers"
-I, --IP_addr=<string>          IP address.
                                OPTIONAL parameter only for tcp.
                                Default IP address is 0.0.0.0
-P, --IP_port=<int>             IP port.
                                OPTIONAL parameter only for tcp.
                                Default IP port is 1502
-d, --device=<string>           Physical serial device [e.g. ttyUSB0].
                                REQUIRED parameter for rtu.
-p, --parity=<int>              Type of parity.
                                OPTIONAL parameter only for rtu.
                                0 is none (default), 1 is odd and 2 is
                                even.
-m, --mode=<int>                Set serial mode.
                                OPTIONAL parameter only for rtu.
                                2 for RS-232 or 4 for RS-485.
-s, --stopbits=<int>            Number of stopbits.
                                REQUIRED parameter only for rtu.
                                1 or 2.
-b, --baudrate=<int>            Baud rate.
                                OPTIONAL parameter only for rtu.
                                Default is 9600.
```

B OBSAH PŘILOŽENÉHO CD

Na disku CD se nachází čtyři adresáře.

1. **Dokumentace_Compulab_Modul_CM-T43** – tento adresář obsahuje dokumentaci k modulu CM-T43,
2. **el_verze_práce** – obsahuje elektronickou verzi bakalářské práce ve formátu PDF,
3. **mb_master_app** – obsahuje zdrojové kódy aplikace **mb_master**,
4. **testing** – osahuje testovací aplikaci **diagslave**.