Instalace a správa linuxových embedded terminálů

Bc. Jaroslav Gargulák

Diplomová práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jaroslav Gargulák**
Osobní číslo:	**A16147**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Instalace a správa linuxových embedded terminálů**
Téma anglicky:	**The Installation and Management of Linux Embedded Terminals**

Zásady pro vypracování:

1. **Vytvořte aplikaci komunikující s terminály pomocí rozhraní ethernet.**
2. **Umožněte instalaci a aktualizaci terminálů z databázového repozitáře.**
3. **Navrhněte databázový repozitář a databázi vyrobených terminálů.**
4. **Záznamy o terminálech aktualizovaných off-line aktualizujte po připojení k databázi.**
5. **Implementujte vyhledání terminálů na síti.**
6. **Umožněte zobrazení diagnostických informací z připojeného terminálu.**
7. **Definujte a zhodnoťte přínosy daného řešení.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **DOSTÁLEK, Libor a KABELOVÁ, Alena. Velký průvodce protokoly TCP/IP a systémem DNS. Brno : Computer Press, a.s., 2008. 978-80-251-2236-5.**
2. **Oracle Corporation. MySQL Documentation. [Online] 2017. https://dev.mysql.com/doc/.**
3. **PETERKA, Jiří. CZ.NIC. Báječný svět elektronického podpisu. [Online] 2010. https://knihy.nic.cz/files/edice/bajecny_svet_elektronickeho_podpisu_cznic.pdf.**
4. **PRICE, Jason. C# programování databází. Praha : Grada Publishing, a.s., 2005. 80-247-0982-1.**
5. **TROELSEN, Andrew W. C# a .NET 2.0 profesionálně. Vyd. 1. Brno: Zoner Press, 2006, 1197 s. Encyklopedie Zoner Press. ISBN 80-86815-42-0.**
6. **ALBAHARI, Joseph a ALBAHARI, Ben. C# 5.0 in a Nutshell: Fifth Edition. Sebastopol : O'Reily Media, 2012. ISBN 978-1-449-32010-2.**
7. **SSH Communications Security. SSH Communications Security. SSH (SECURE SHELL). [Online] 2017. https://www.ssh.com/ssh/.**
8. **Linux: dokumentační projekt. 4., aktualiz. vyd. Přeložil Lubomír PTÁČEK. Brno: Computer Press, 2007. ISBN 978-80-251-1525-1.**

Vedoucí diplomové práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **1. prosince 2017**

Termín odevzdání diplomové práce: **16. května 2018**

Ve Zlíně dne 11. prosince 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

L.S.

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 17.4.2018

.....
podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá návrhem a implementací systému pro instalace a aktualizace aplikací v linuxových embedded terminálech. Systém umožňuje bezpečnou instalaci aplikací, zobrazení diagnostických informací a správu databáze linuxových embedded terminálů. V teoretické části jsou popsány použité technologie síťového rozhraní a operačního systému Linux. V praktické části je popsán návrh celého systému a implementace jeho jednotlivých částí.

Klíčová slova: embedded, Linux, balíčkovací systém,

ABSTRACT

This diploma thesis deals with the proposal and implementation of the system for installation and updating of application in linux embedded terminals. This system enables the safety installation of applications, projection of diagnostic information and management of linux embedded terminals databases. The theoretical part describes the used technology of network interface and operating system Linux. The practical part describes the proposal of the whole system and implementation of its individual parts.

Keywords: embedded, Linux, package manager

Poděkování… patří každému, kdo měl trpělivost a byl se mnou až do chvíle, kdy píšu toto poděkování.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Embedded systémy jsou populárními typy informačních a řídicích systémů, se kterými se setkáváme stále častěji. Především v průmyslovém odvětví mají tyto systémy pevné zastoupení a tvoří skupinu životně důležitých částí výrobních linek nebo jednotlivých strojů. Na tyto počítače a jejich softwarové vybavení jsou kladeny vysoké nároky. Pracují v náročném průmyslovém provozu 24 hodin denně, 7 dní v týdnu, a proto je nutné zabezpečit snadnou údržbu a v maximální míře minimalizovat riziko poruchy.

Tato práce se věnuje linuxovým embedded dotykovým terminálům, které slouží jako operátorský panel a jako hlavní řídící počítač průmyslových zařízení. Terminály obsahují sofistikovaných software, který prochází různými fázemi vývoje. Uživatelé si často uvědomují požadavky na tento software, až při používání kompletního zařízení, a proto je běžné, že je software upravován až při provozu. Každá taková úprava vyžaduje i následnou instalaci, ale u embedded systémů se nemusí jednat vždy o rutinní úkon. Přestože instalace obvykle spočívá v přehrání několika souborů, stává se, že technik chybuje nebo se mu instalace vůbec nezdaří. Jakýkoliv problém při instalaci za provozu může znamenat především zdržení výroby a tím i nespokojenost zákazníka.

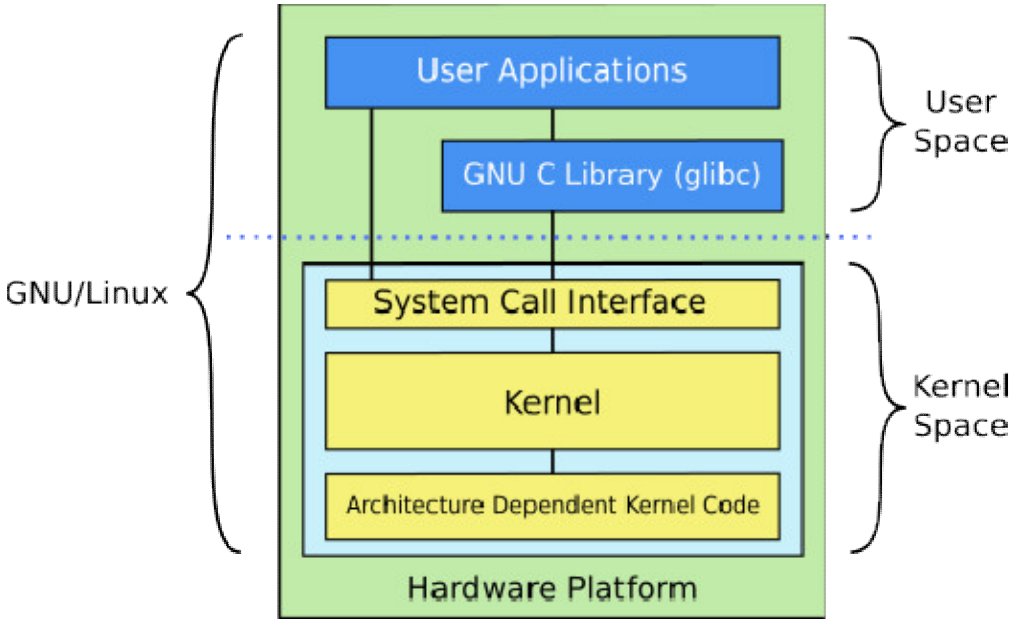
Téma této diplomové práce vzniklo jako snaha o zamezení problémů spojených s instalací aplikací do embedded terminálů. Cílem je vytvoření systému, který především usnadní instalaci těchto aplikací a zajistí správu databáze vyrobených terminálů, včetně informací o instalovaných aplikacích. Bude implementovat funkce pro jednoduchou instalaci ze softwarového repozitáře nebo lokálního balíčku a poskytne základní diagnostické informace o připojeném terminálu.

Systém najde využití při výrobě terminálů a také při servisních zásazích u zákazníka. Jeho hlavním přínosem bude zamezení manipulace s citlivými soubory a tím zvýšení spolehlivosti při instalacích. Dalším přínosem bude zajištění aktuálního přehledu o všech terminálech a jejich aplikacích, což bude výhodné zejména při poradenství. Nástroj, který byl již několikrát vyžadován servisními techniky, tak bude užitečným pomocníkem v náročném průmyslovém prostředí.

I. TEORETICKÁ ČÁST

1 LINUX

Je operační systém volně šiřitelný pod licencí GNU GPL. Byl vytvořen Linusem Torvaldsem jako studentský projekt a v současné době je velmi rozšířeným systémem napříč různými platformami, od velkých serverů až po spotřební elektroniku. Jeho obliba spočívá v tom, že Linux je bezpečný, stabilní a díky licenčním podmínkám jej může kdokoliv upravit a použít pro svou potřebu. [1]



Obr. 1: Struktura GNU/Linux [2]

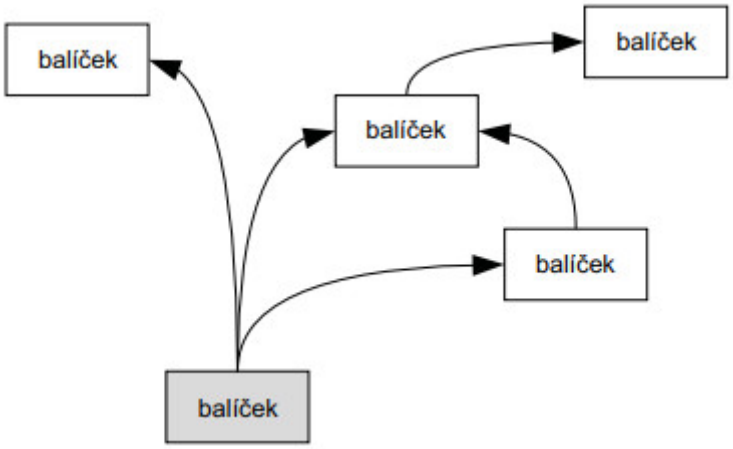
Pojem Linux představuje pouze jádro (kernel), které komunikuje s hardwarem a které splňuje standard POSIX. Je umístěno v chráněném paměťovém prostoru, který je oddělen od uživatelského paměťového prostoru. Spojení mezi jádrem a uživatelskými aplikacemi je zajištěno pomocí systémového volání. Aplikace mohou ke svému běhu využívat různých utilit a knihoven z projektu GNU. Jednou z často používaných je knihovna glibc. Ta poskytuje aplikacím základní funkce jako alokace paměti, práce se soubory nebo správa vláken. Díky své struktuře může být Linux přizpůsoben pro konkrétní použití a dokáže tak uspokojit běžné uživatele i zkušené programátory a správce systémů. [1] [3]

Systém Linux je nainstalován na embedded terminálu, který je využit k této práci. Terminál je vybaven procesorem s architekturou ARM, pro které jsou volně dostupné softwarové nástroje od společností Linaro. Tato společnost vznikla seskupením velkých výrobců těchto procesorů. [4]

1.1 Balíčkovací systémy

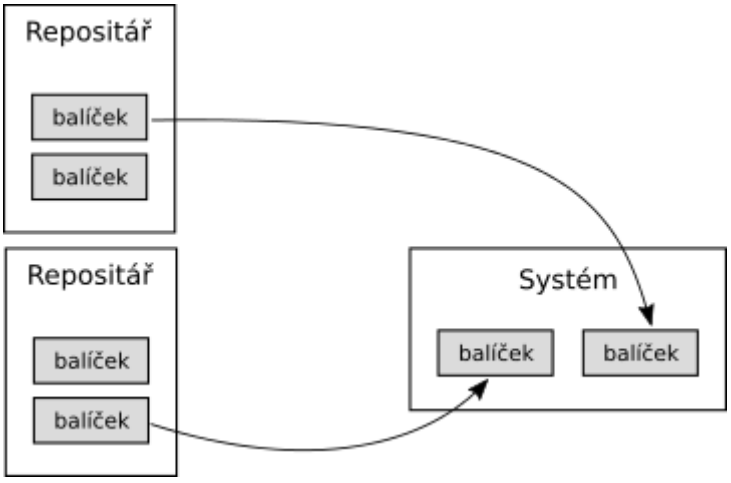
Balíčkovací systémy jsou typické pro linuxové distribuce. Jejich hlavním úkolem je usnadnění instalace, aktualizace programů a správa instalovaných balíčků v počítači. Obecně existují balíčky s binárními soubory nebo se zdrojovými kódy. V této práci jsou využívány pouze binární balíčky. Mezi hlavní výhody balíčkovacích systému patří:

- Vyhodnocení závislostí – v případě, kdy instalovaný program vyžaduje pro svou činnost další knihovny, umí balíčkovací systém najít odpovídající balíček a současně s hlavním programem balíček doinstalovat.



Obr. 2: Závislost mezi balíčky.

- Instalace z repozitářů – balíčky je možné instalovat z různých softwarových repozitářů, které jsou dostupné na internetu. Takový repozitář obvykle poskytuje tvůrce programu nebo tvůrce linuxové distribuce. Systém umožňuje přidat adresu repozitáře do svých „oblíbených“ a následně hlídat vydané aktualizace.



Obr. 3: Instalace balíčků z různých repozitářů.

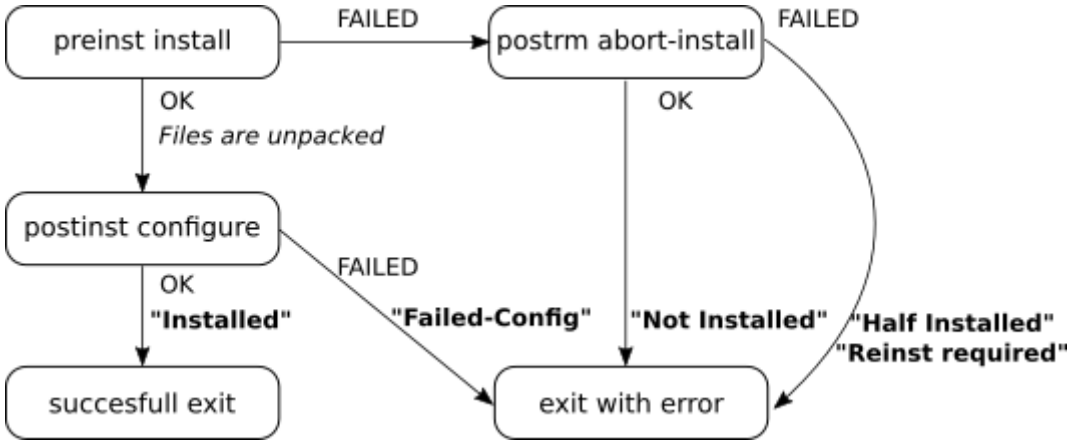
- Vyhledání – systém umožňuje efektivní vyhledávání balíčků v repozitářích.
- Bezpečnost – balíčky podporují elektronický podpis, systém může před samotnou instalací podpis ověřit a zjistit, zda není balíček podvržený.
- Více verzí najednou – obvykle u knihoven může být potřeba mít více verzí stejné knihovny, kvůli kompatibilitě s různými programy.

[5]

Napříč různými linuxovými distribucemi existuje několik druhů balíčkovacích systémů. Pro embedded zařízení je možné využívat balíčkovací systém Debianu.

1.1.1 Balíčkovací systém Debian

Balíčkovací systém z distribuce Debian je vyspělým systémem používaným například i v oblíbené distribuci Ubuntu. Balíček tohoto systému je soubor typu archiv *.deb*, který obsahuje dva typy souborů. Prvním z nich jsou soubory, které budou instalovány do systému. Druhým typem jsou řídící soubory. Mezi řídícími soubory se musí vždy nacházet soubor *control* s popisem daného balíčku a volitelně zde můžou být skripty *preinst*, *postinst*, *prerm* a *postrm*, které jsou spouštěny během instalace. [5] [6]



Obr. 4: Průběh instalace balíčku Debian [6]

1.2 Linuxový embedded terminál

Terminál, se kterým se pracuje v této práci, je výrobkem firmy SVCS Process Innovation. Je vybaven displejem o velikosti 10 palců s kapacitní dotykovou vrstvou a moderním ARM procesorem I.MX6 od firmy NXP.

Základní parametry terminálu:

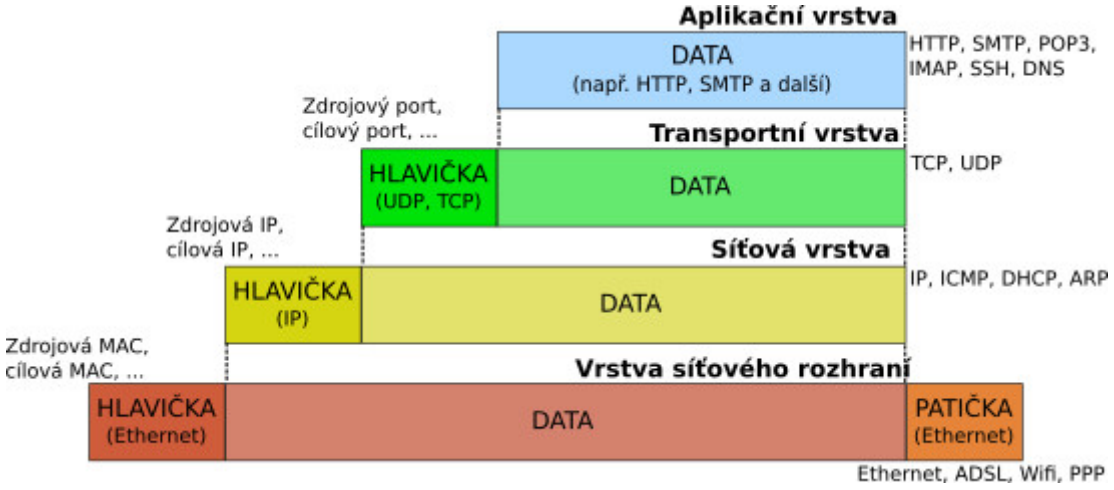
- CPU NXP™ i.MX 6 ARM® Cortex™-A9
- 1GB DDR3 SDRAM
- 4 GB eMMC Flash
- 1x Ethernet
- 2x RS485
- 4x USB 2.0
- -40..+85°C

Terminál je běžně používán jako hlavní řídící počítač pro průmyslová zařízení. Pomocí komunikačních portů je schopen komunikovat s hardwarovými moduly a řídit technologické procesy dle přání zákazníků. Na terminálu běží operační systém Linux se specifickou aplikací pro konkrétní zařízení.

Většina komunikace s terminálem bude řešena pomocí rozhraní ethernet a protokolu SSH.

2 SÍŤOVÝ MODEL TCP/IP

Model vznikl na konci 60. let v USA pro potřeby počítačové sítě ARPANET. Nyní je implementován ve všech běžných výpočetních zařízeních a je používán po celém světě u drátových i bezdrátových sítí. Model je koncipován jako čtyřvrstvá datagramová služba pro síťovou komunikaci, kde každá vrstva má na starosti určitou úlohu při přenosu. Nižší vrstvy se snaží o co nejrychlejší doručení datagramu bez zajištění spolehlivosti a vyšší vrstvy se starají o spolehlivost přenosu, tj. potvrzování nebo opakování vysílání. [7]



Obr. 5: Síťový model TCP/IP [8]

2.1 Vrstva síťového rozhraní

Její implementace je závislá na typu přenosové technologie. Vrstva má na starosti příjem a vysílání datových rámců po fyzickém přenosovém médiu. Data k odeslání, která jsou dodána z vyšší vrstvy, zapouzdřuje do MAC rámců a odesílá. Příchozí data, která jsou doručena po přenosovém médiu, vybaluje z MAC rámců, provádí kontrolu chyb a předává vyšší vrstvě ke zpracování. Vrstva je schopna MAC rámec adresovat uvnitř lokální sítě. [7] [9]

2.2 Síťová vrstva

Vrstva je nezávislá na přenosové technologii a zajišťuje, aby se rámce dostaly od odesílatele k příjemci. Je schopna provádět směrování a předávat rámce mezi sítěmi. K tomu využívá protokol IP, který data z vyšší vrstvy zapouzdřuje do rámce a připojuje k němu cílovou a zdrojovou IP adresu. [7] [9]

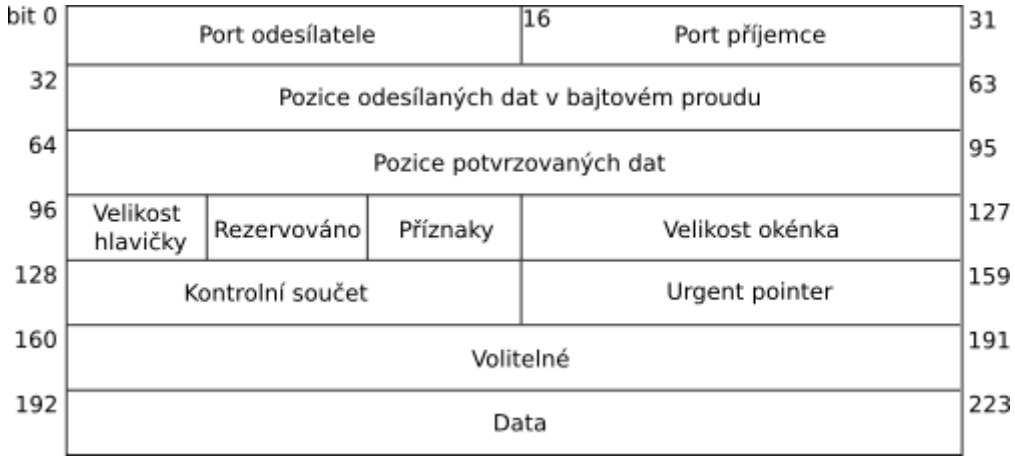
Významným protokolem této vrstvy je i protokol ICMP, který je používán k zasílání informací při nestandardních situacích jako například vypršení TTL (Time to Live) datagramu. Protokol je rovněž používán pro ověření dostupnosti zařízení na síti, příkaz je známý jako ping. [7]

2.3 Transportní vrstva

Úkolem vrstvy je zajištění přenosu dat mezi konkrétními aplikacemi. Každá aplikace v rámci jednoho počítače může obsadit jeden nebo více očíslovaných portů. Příjemce je pak schopen rozpoznat, které aplikaci jsou data určena a na který port zaslat odpověď. Transportní vrstva je nejčastěji realizována protokolem TCP nebo UDP. [9]

2.3.1 Protokol TCP

TCP (Transmission Control Protocol) je využíván transportní vrstvou k zajištění spolehlivého přenosu dat. Spolehlivost je zajišťována důmyslným potvrzováním přenesených bloků dat a je podmínkou navázání spojení před zahájením přenosu. Přenos je realizován proudově. Vyšší vrstva může do transportní vrstvy předávat data bajt po bajtu. Nicméně TCP neodesílá každý bajt ihned, ale ukládá data do vyrovnávací paměti a po naplnění nebo na příkaz odesílá celý blok najednou. [7]



Obr. 6: Struktura TCP datagramu [7]

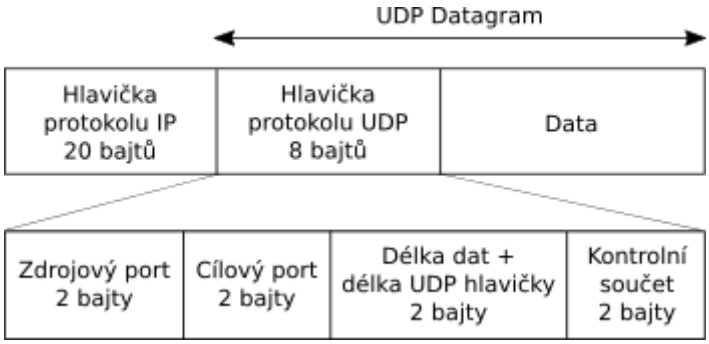
- Pozice odesílaných dat v bajtovém proudu – pozice prvního odesílaného bajtu z celkového bajtového proudu ve směru odesílání.
- Pozice potvrzovaných dat – používáno při potvrzování, obsahuje číslo pozice očekávaného bajtu z celkového bajtovém proudu.
- Velikost hlavičky – celková velikost TCP hlavičky v násobcích 32 .

- Příznaky – slouží k ovlivnění přenosu jako navázání spojení, ukončení spojení nebo potvrzování.
- Velikost okénka – počet bajtů, které je příjemce schopen přijmout k právě přijatým.
- Urgent pointer – umožňuje označit část dat jako přednostní.

[7]

2.3.2 Protokol UDP

Protokol UDP (User Datagram Protocol) je „nespolehlivou“ službou na transportní vrstvě síťového modelu. Na rozdíl od protokolu TCP nezaručuje doručení datagramu protější straně, nezaručuje doručení více datagramů ve správném pořadí a funguje jako nespojovaná služba, tj. nevyžaduje před zahájením přenosu navázání spojení. Od vyšší vrstvy vždy očekává kompletní blok dat, který je odesílán najednou. V případě použití UDP přebírá odpovědnost za kvalitu přenosu aplikační program. Na první pohled nevýhodné vlastnosti jsou užitečné v případech, kdy je se ztrátami datagramu počítáno a znovu vysílání by byla jen ztráta času, např.: online hry nebo VOIP. [10] [7]



Obr. 7: Struktura UDP datagramu [7]

Velkou výhodou UDP protokolu je jeho jednoduchost a možnost vysílat broadcast a multicast data. Vysílání UDP broadcastů bude využito v této práci pro vyhledání terminálů na síti.

2.4 Aplikační vrstva

Je nejvyšší vrstvou v síťovém modelu. Obvykle je tvořena konkrétními aplikacemi, které zprostředkovávají uživateli síťové služby jako email, webový prohlížeč a spoustu dalších. Ke své činnosti většinou využívají aplikačních protokolů, které jsou dále závislé na protokolech TCP nebo UDP v transportní vrstvě. [9]

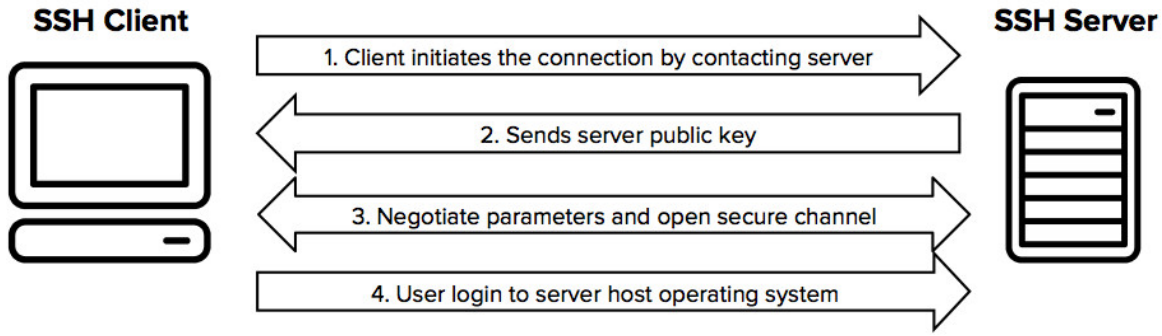
SSH	Bezpečná komunikace mezi dvěma počítači - příkazová řádka, přenos dat.
HTTP	Výměna hypertextových dokumentů ve formátu HTML
POP3	Stahování emailových zpráv z poštovního serveru
FTP	Nezabezpečený přenos souborů.
DHCP	Automatická konfigurace počítačů připojených do sítě.

Tab. 1: Příklady aplikačních síťových protokolů [9]

2.4.1 Protokol SSH

Protokol SSH poskytuje silně zabezpečenou komunikaci mezi dvěma počítači a lze jej chápat jako zabezpečenou alternativu protokolu telnet. SSH je ve své základní podobě příkazového řádku vhodné zejména pro vzdálenou administraci serverů. Pomocí SSH jsou implementovány protokoly SFTP a SCP sloužící pro přenos souborů mezi počítači, nebo může být s jeho pomocí vytvořen tunel pro vzdálené ovládání plochy. [11]

SSH pracuje jako klient-server, kde server je zodpovědný především za autentizaci klienta a klient je zodpovědný za ověření totožnosti serveru a navázání spojení.



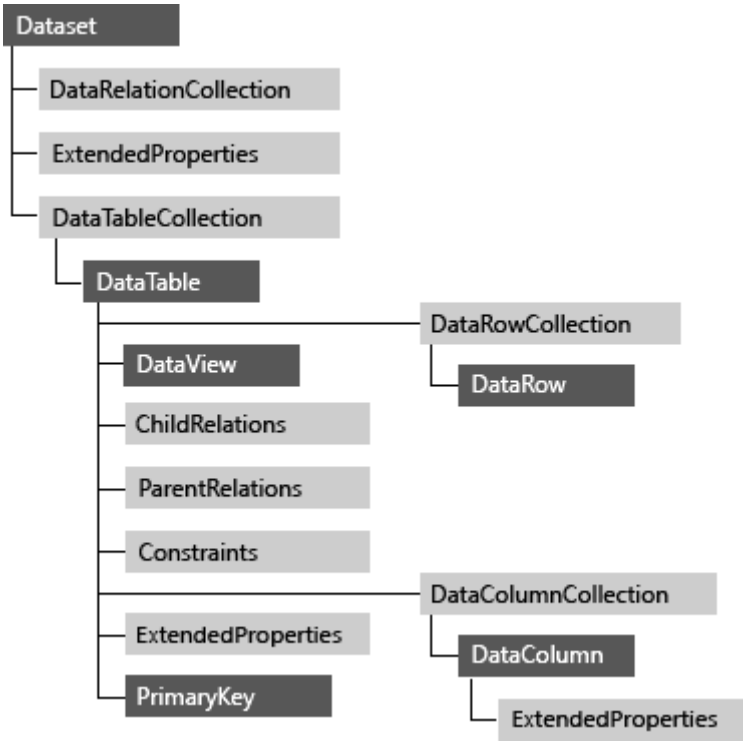
Obr. 8: SSH navázání spojení [12]

Klient vytváří TCP spojení se serverem a server následně odesílá informaci o verzi protokolu a svůj veřejný klič. Klient vybírá kompatibilní verzi protokolu a ověřuje totožnost serveru pomocí veřejného klíče. Porovnává jej s klíčem, který má uložen z předchozího připojení. Pokud by klíč nebyl stejný, může to znamenat, že se klient připojuje na server, který se pouze vydává za požadovaný server. Pokud je kontrola úspěšná, server a klient se dohodnou na symetrickém klíči, který bude použit k šifrování komunikace. Dalším krokem je autentizace klienta. To je možné pomocí hesla nebo pomocí klíčových párů. Po autentizaci klienta je spojení navázáno a může být zahájena komunikace. Veškerá další komunikace je šifrována šifrou AES a integrita dat ověřována hashovacím algoritmem SHA-2. [12] [13]

3 KOMPONENTA ADO.NET

Komponenta ADO.NET, která je součástí frameworku .NET, bude využita pro přístup k firemní databázi a k XML souboru pro ukládání offline instalací. Komponenta umožňuje jednotný přístup k různým datovým zdrojům pomocí rozhraní, které musí poskytovatel datového zdroje implementovat. Tímto řešením je oddělena vrstva manipulace s daty od vrstvy přístupu k datům a aplikace tak může používat jednotný přístup k různým datovým zdrojům. [14]

Framework .NET poskytuje pro práci s daty komponentu *DataSet*, která umí vnitřně vytvořit tabulky, relace mezi tabulkami a nastavit primární klíče v tabulkách. *DataSet* lze používat i jako lokální databázi s ukládáním do XML souboru. Spojení mezi datovým zdrojem a komponentou *DataSet* poskytuje komponenta *DataAdapter*, která používá pro komunikaci s datovým zdrojem své vnitřní *Command* objekty. Ke komunikaci s datovým zdrojem lze využít i komponentu *DataReader*, který vytváří z datového zdroje datový proud. [15] [16]



Obr. 9: Architektura ADO.NET [15]

II. PRAKTICKÁ ČÁST

4 MOTIVACE A NÁVRH ŘEŠENÍ

Embedded terminály a jejich aplikace jsou hlavním řídícím prvkem mnoha průmyslových zařízení vyráběných firmou SVCS Process Innovation. Správně fungující aplikace jsou u takových zařízení kritickým parametrem. S přibývajícím počtem různých typů zařízení, přibývá také počet různých aplikací a aktualizací instalovaných do terminálů. Potřeba aktualizace aplikace může být vynucena potřebou opravení chyby, nebo skutečností, kdy doba mezi výrobou zařízení a uvedením do provozu u zákazníka bývá v některých případech i větší než jeden rok. Za tuto dobu je obvykle vydána nová verze aplikace a servisní technik, který u zákazníka uvádí zařízení do provozu, musí provést i aktualizaci terminálu. Bohužel informace o aktualizaci se zpět do firmy obvykle nedostane a v případě pozdějších řešení dotazů od zákazníků, je nutné si typ zařízení a verzi aplikace od zákazníka vyžádat.

Instalace a hlavně aktualizace můžou být mnohdy pro servisní techniky nepříjemným zážitkem s nejistým výsledkem, a proto je nutné se problémem instalací a hlavně aktualizací zabývat. Systém vytvořený v rámci diplomové práce by měl zvýšit pohodlí a bezpečnost při instalacích aplikací do terminálů a zároveň odstranit problém s chybějícími informacemi o terminálech dodaných zákazníkům.

4.1 Funkční požadavky

- Aplikace, která umožní nainstalovat balíček s aplikací do terminálu.
- Balíček s aplikací bude možné instalovat přímo z repozitáře - online.
- Balíček bude možné nainstalovat i offline mimo dosah repozitáře.
- Aplikace umožní vyhledat terminály na síti.
- Informace o připojeném terminálu budou automaticky uloženy do databáze.
- Bude zajištěna aktuálnost databáze i při offline instalaci.

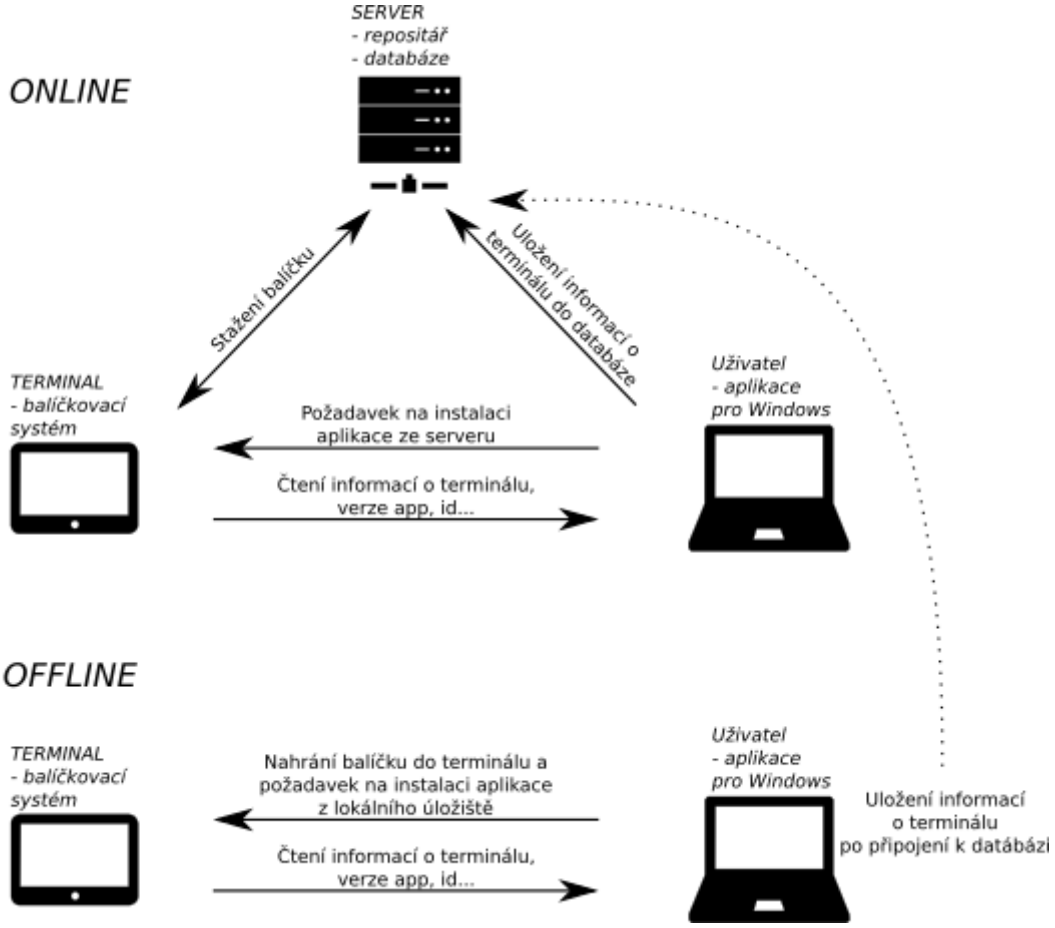
4.2 Systémové požadavky

- Instalace aplikací bude realizována linuxovým balíčkovacím systémem v terminálu.
- Vydávání povelu pro instalace bude pomocí aplikace na Windows.
- Aplikace se bude připojovat pomocí ethernetu.
- Aplikace bude komunikovat s databází na firemním serveru.

4.3 Topologie systému

Celá aplikace pro terminál se skládá z několika málo souborů, proto byla uvažována varianta vlastního řešení balíčku a jeho instalace. Balíček by obsahoval soubory s popisem umístění a uživatelská aplikace by dle nějakého popisu soubory nahrála do terminálu. Při dalších úvahách však vyvstávaly otázky ohledně zabezpečení balíčků proti poškození, nebo ošetření při přerušení instalace.

V linuxových systémech se pro účely instalací běžně používají balíčkovací systémy, které řeší zabezpečení i řadu dalších věcí spojených s instalací, jako závislosti nebo různé akce při aktualizaci. Proto byl i pro tuto práci vytipován balíčkovací systém, který je vhodný pro jednodušší systémy jako je embedded zařízení. Následně byla s ohledem na požadavky vytvořena topologie, která pokrývá jak online instalaci z repozitáře, tak offline instalaci mimo dosah repozitáře.



Obr. 10: Topologie systému

Veškeré instalace na terminálu budou prováděny pouze přes balíčkovací systém. Povely pro zahájení instalace budou balíčkovacímu systému zadávány přes aplikaci na počítači uživatele. Při online instalaci, kdy je dostupná komunikace se serverem, uvidí uživatel všechny dostupné balíčky v repozitáři na serveru a bude tak moci instalovat balíčky přímo z tohoto repozitáře. Na serveru bude spuštěn databázový systém, kde budou uchovány informace o vyrobených terminálech a instalovaných balíčcích. Tyto informace bude ukládat uživatelská aplikace, jakmile bude provedena nějaká změna.

Bude-li uživatel mimo dosah serverového repozitáře, bude moci provádět takzvané offline instalace. Stejný balíček, který se nachází v repozitáři, bude mít uživatel ve svém počítači a aplikace mu umožní balíček nahrát do terminálu. Následně aplikace vyšle příkaz pro instalaci z lokálního úložiště terminálu. Informace o offline instalaci budou ukládány na počítači uživatele a jakmile se připojí k databázi, budou tyto informace aktualizovány.

4.4 Návrh GUI linuxových terminálů

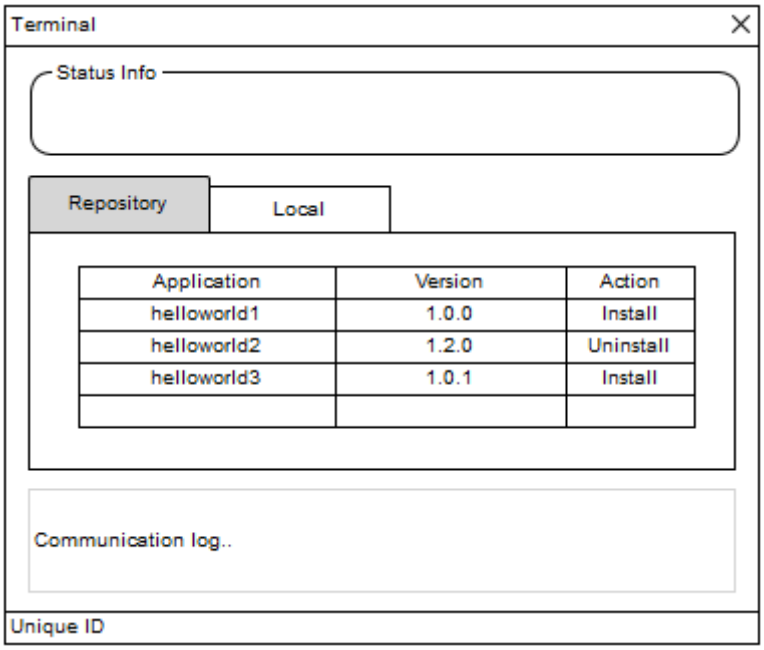
V rámci bakalářské práce Řízení a monitoring hardwarových modulů, byla vytvořena aplikace, která slouží jako konfigurační nástroj pro hardwarové moduly firmy SVCS Process Innovation. Tyto moduly tvoří společně s terminálem kompletní řídicí systém průmyslových zařízení. Proto se nabízí, aby tato aplikace byla rozšířena i o podporu terminálů. [17]

Aplikace je napsaná v jazyce C# a je spustitelná na Windows, kde je instalován framework .NET verze 4 a vyšší. Komunikace s moduly je realizována pomocí sériového portu, který je vytvořen FTDI převodníkem USB-RS485. Pro komunikaci s terminály bude nutné aplikaci rozšířit o komunikaci pomocí rozhraní ethernet a protokolu SSH. Protože framework .NET standardně nepodporuje protokol SSH, bude nutné vytipovat vhodnou knihovnu. Dále bude nutné implementovat funkce pro vyhledávání terminálů na síti a komunikaci s databází. [17]

Na rozdíl od hardwarových modulů, bude možné přidat ručně adresy terminálů pro případ, kdy by automatické vyhledávání nebylo úspěšné. Kvůli bezpečnosti nebude možné jakékoliv konzolové zadávání komunikačních příkazů.

S uživatelského pohledu budou terminály do aplikace zapracovány ve stejném duchu jako hardwarové moduly. Bude možné připojit více terminálu najednou a každý připojený terminál bude mít své vlastní dialogové okno, ve kterém budou zpracovány veškeré

funkce. Protože každý terminál bude mít svůj vlastní komunikační kanál, bude zobrazen výpis komunikace přímo v okně terminálu. Dále zde budou dostupné veškeré diagnostické informace jako doba běhu, volné místo na disku apod. Ve stejném okně bude zobrazen i výpis repozitáře a možnost offline instalace. [17]



Obr. 11: Drátěný model dialogu terminálu

5 REPOZITÁŘ BALÍČKŮ NA SERVERU

Repozitář představuje adresářovou strukturu, která je zpřístupněna webových serverem. Pro embedded terminály byl na firemním serveru vytvořen repozitář s následující strukturou:

Obr. 12: Struktura repozitáře [18]

5.1 Vytvoření binárního balíčku

Balíčkovací systém OPKG umí pracovat s balíčky Debianu (.deb). Pro tvorbu těchto balíčků je možné využít utilitu `dpkg-deb`. Balíček lze sestavit z adresáře, ve kterém se nachází adresář `DEBIAN` a kompletní adresářová struktura programu. Adresářová struktura programu bude vytvořena při instalaci v místě, které je definováno v konfiguračním souboru `opkg.conf` v položce `dest`. V adresáři `DEBIAN` musí být umístěn soubor `control`, který popisuje daný balíček. [6]

Obr. 13: Adresářová struktura balíčku

Soubor `control` binárního balíčku (balíček bez zdrojových kódů) musí povinně obsahovat následující informace:

`Package`	Název balíčku, který bude používán pro instalaci a odinstalaci. Měl by obsahovat pouze jméno (bez komentářů, bez verze). Musí být pouze malými písmeny a začínat alfanumerickými znaky. Dovoleny jsou znaky (a-z), číslice (0-9) a znaménka plus (+), mínus (-).
`Version`	Verze aplikace.
`Architecture`	Architektura, pro kterou je aplikace zkompilována.
`Maintaner`	Kontakt na autora.
`Description`	Podrobnější popis aplikace.

Tab. 2: Struktura souboru control v balíčku .deb [6]

```
Package: helloworld
Version: 1.0.0
Architecture: armv71
Section: extras
Priority: standard
Maintainer: Company s.r.o.
Description: Hello World Application
```

Obr. 14: Příklad souboru control v balíčku

Mimo povinných informací může soubor obsahovat i nepovinné informace jako závislosti na jiných balíčcích, název balíčku obsahující zdrojové kódy nebo velikost instalace. [6]

Z připraveného adresáře lze sestavit balíček příkazem:

```
dpkg-deb --build name_ver_arch
```

Takto sestavený balíček již může být nahrán do repozitáře nebo může být šířen pro offline aktualizaci. Posledním krokem je vytvoření metadat v repozitáři.

5.2 Vytvoření metadat repozitáře

Při každé změně balíčku v repozitáři je nutné vygenerovat nový popisný soubor `Packages.gz`. Tento soubor je stahován klientem OPKG. Jsou v něm popsány všechny balíčky v repozitáři včetně kontrolních součtů, které klient využívá ke kontrole balíčku po stažení. Soubor `Packages` je možné vygenerovat příkazem:

```
dpkg-scanpackages –m path/to/repo > Packages
```

Přepínač `–m, --multiversion` slouží k povolení více verzí jednoho balíčku v repozitáři. Standardně není možné mít v repozitáři více balíčku se stejným jménem.

Vygenerovaný soubor je potřeba zabalit pomocí `gzip` a poté nahrát do repozitáře.

[19]

6 BALÍČKOVACÍ SYSTÉM TERMINÁLU

Distribuce Linuxu instalovaná v terminálech standardně neobsahuje žádný balíčkovací systém. Vzhledem k tomu, že se jedná o jednoúčelové embedded zařízení, jehož distribuce Linuxu vychází z projektu Yocto, byl doinstalován klient balíčkovacího systému OPKG (Open Package Management), který je rovněž součástí projektu Yocto. [20]

6.1 Kompilace klienta OPKG

Klient OPKG je volně šiřitelný ve formě zdrojových kódů a je nutné jej pro danou platformu zkompilovat. To lze provést na standardní i686 platformě s přepínačem pro platformu arm-linux. Ke kompilaci byly využity překladače od společnosti Linaro. [21]

```
./configure --host=arm-linux --prefix=/usr/local/opkg
CC=/usr/bin/arm-linux-gnueabihf-gcc
LD=/usr/bin/arm-linux-gnueabihf-ld
RANLIB=/usr/bin/arm-linux-gnueabihf-ranlib
CFLAGS="-I/usr/arm-linux-gnueabihf/include"
LDFLAGS="-L/usr/arm-linux-gnueabihf/lib -larchive"
```

Obr. 15: Konfigurace křížové kompilace OPKG

Se stejnou konfigurací je možné zkompilovat knihovnu libArchive, která je pro kompilaci OPKG potřebná. [22]

Výsledkem kompilace je spustitelný soubor opkg, knihovny a manuálové soubory.

Obr. 16: Adresářová struktura sestaveného klienta OPKG

Všechny soubory je nutné nahrát do terminálu a poté je možné začít klienta OPKG používat.

Zkompilovaný klient OPKG je součástí přílohy P1.

6.2 Konfigurace klienta OPKG

Klient je konfigurován pomocí jednoho souboru *opkg.conf*, jehož umístění lze předat parametrem při spuštění. V souboru je nutné nadefinovat především adresu serveru s repozitářem a případně další parametry.

`src/gz local file:///path/to/packages`	Cesta lokálního úložiště balíčků
`src/gz repo http://server-addr.com/repo`	Adresa serverového repozitáře
`arch archType 200`	Preferovaná architektura. Číslo definuje prioritu pro případ, kdy balíček bude dostupný pro více architektur.
`dest place /home/user`	Definuje místo kam je možné nainstalovat balíček. Lze určit při instalaci.

Tab. 3: Přehled konfiguračních příkazů [23]

```
arch all 100
arch armv71 200
src/gz binary-armv7 http://192.168.1.5/repo/dists/release/binary-armv71
dest root /home/root
option lists_dir /var/opkg-lists
```

Obr. 17: Příklad konfiguračního souboru OPKG

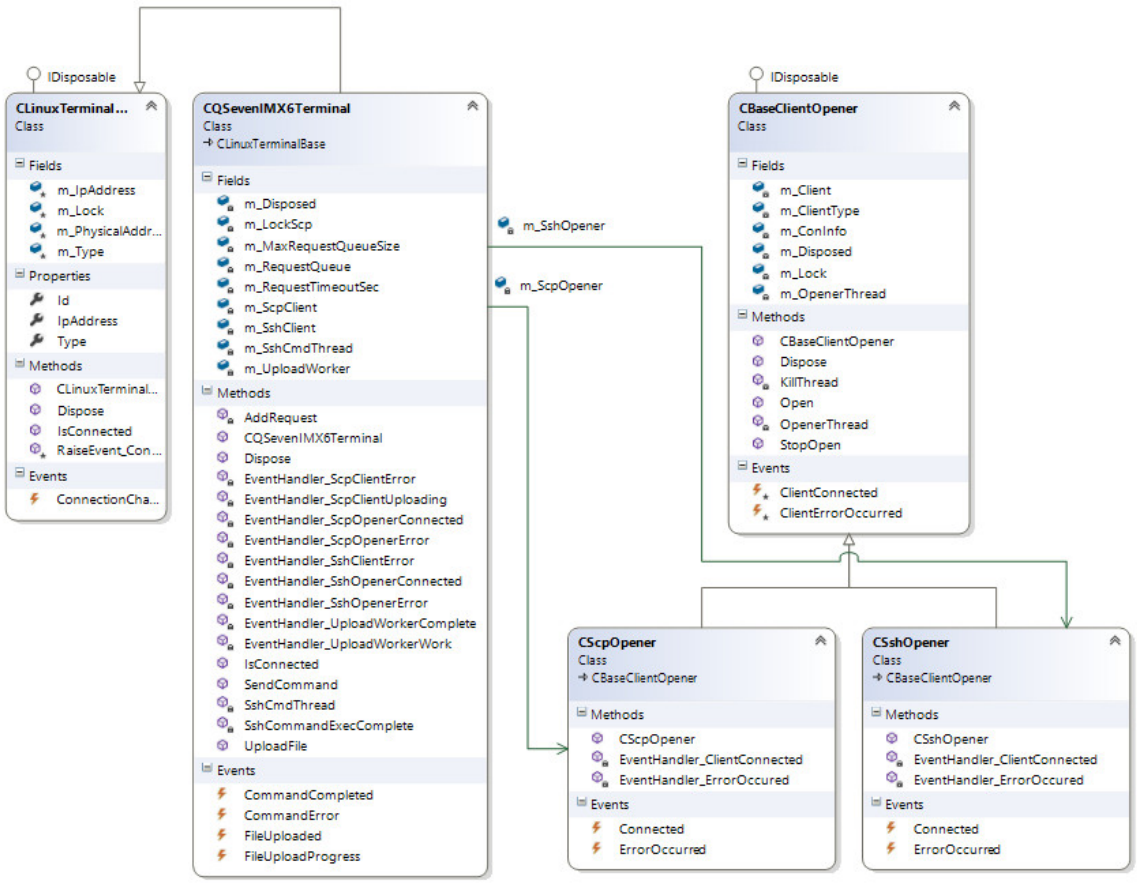
Kompletní konfigurační soubor se nachází v příloze P1.

7 IMPLEMENTACE PODPORY TERMINÁLŮ

Aplikace, která byla vytvořena v rámci bakalářské práce Řízení a monitoring hardwarových modulů, byla rozšířena o podporu terminálů. Je napsaná v jazyce C# a základem rozšíření je objekt terminálu implementovaný třídou *CLinuxTerminalBase.* Jedná se o rodičovskou třídu všech linuxových terminálů, které by se mohly v budoucnu objevit. Třída obsahuje jedinečný identifikátor terminálu, typ terminálu definovaný potomkem, IP adresu pro komunikaci a jednu událost, která informuje o změně připojení. Instance terminálů jsou uchovány v kolekci typu list a jsou zobrazovány na hlavním formuláři ve zvláštním layoutu určeném pro linuxové terminály. [17]

7.1 Třída terminálu IMX6

Implementace IMX6 terminálu, je realizována třídou *CQSevenIMX6Terminal,* která dědí výše uvedenou rodičovskou třídu. Poskytuje metody pro vyslání SSH příkazu a nahrání souboru pomocí SCP. Dále poskytuje události pro signalizaci dokončení SSH příkazu a signalizaci dokončení nahrávání souboru.



Obr. 18: Diagram základních tříd implementace terminálu

7.1.1 SSH a SCP připojení

Základ připojení aplikace k terminálu tvoří knihovna SSH.NET, která je volně šiřitelná pod licencí MIT, je kompletně napsaná v jazyce C# a bez využití kódu třetích stran. Knihovna umožňuje vytvořit SSH klienta a provádět SSH příkazy. Dále umožňuje přenosy souborů pomocí SCP a SFTP a podporuje přihlašování pomocí hesel i pomocí klíčů. [24]

Pro otevření SSH připojení byla vytvořena třída *CBaseClientOpener*, která je rodičem třídy *CSshOpener.* Třída slouží pro permanentní udržení připojení jehož parametry přijímá v konstruktoru. Je tvořena vláknem, které po zavolání metody *Open* začne navazovat připojení. Pokud server není dostupný, vyvolává se událost *ClientErrorOccured* a další pokus o spojení proběhne automaticky po uběhnutí intervalu, který je nastaven parametrem při volání metody *Open*. Podaří-li se připojení úspěšně navázat, dojde k vyvolání události *ClientConnected* a reference na objekt připojeného klienta je předána v parametru události. Po vyvolání události je vlákno uspáno, dokud není opět zavolána metoda *Open*. Volání metody *Open* je navázáno na událost *ErrorOccured*, která je vyvolána připojeným klientem při rozpadnutí spojení. Touto konstrukcí je zajištěno, že spojení s terminálem bude automaticky navázáno vždy, jakmile to bude možné.

SCP připojení je vytvářeno jen před přenosem souboru, jelikož je využíváno jen při lokální instalaci balíčku. Po přenosu je opět uzavíráno. Princip připojení je stejný jako v případě SSH. Je implementováno třídou *CScpOpener*, která dědí třídu *CBaseClientOpener*.

7.1.2 Zpracování SSH příkazů

Zaslání SSH příkazu je možné iniciovat zavoláním metody *SendComand* nad objektem terminálu, přičemž lze vybírat pouze z množiny příkazů definovaných výčtem. Příkazy jsou zpracovávány stylem producent-konzument, jsou řazeny do fronty a postupně zpracovávány vláknem *SshCmdThread*. Po zpracování příkazu jsou vyvolány události *CommandCompleted* nebo v případě chyby *CommandError*. Pokud se nachází ve frontě další příkazy ke zpracování, vlákno okamžitě přejde k jejich zpracování. V opačném případě je vlákno uspáno.

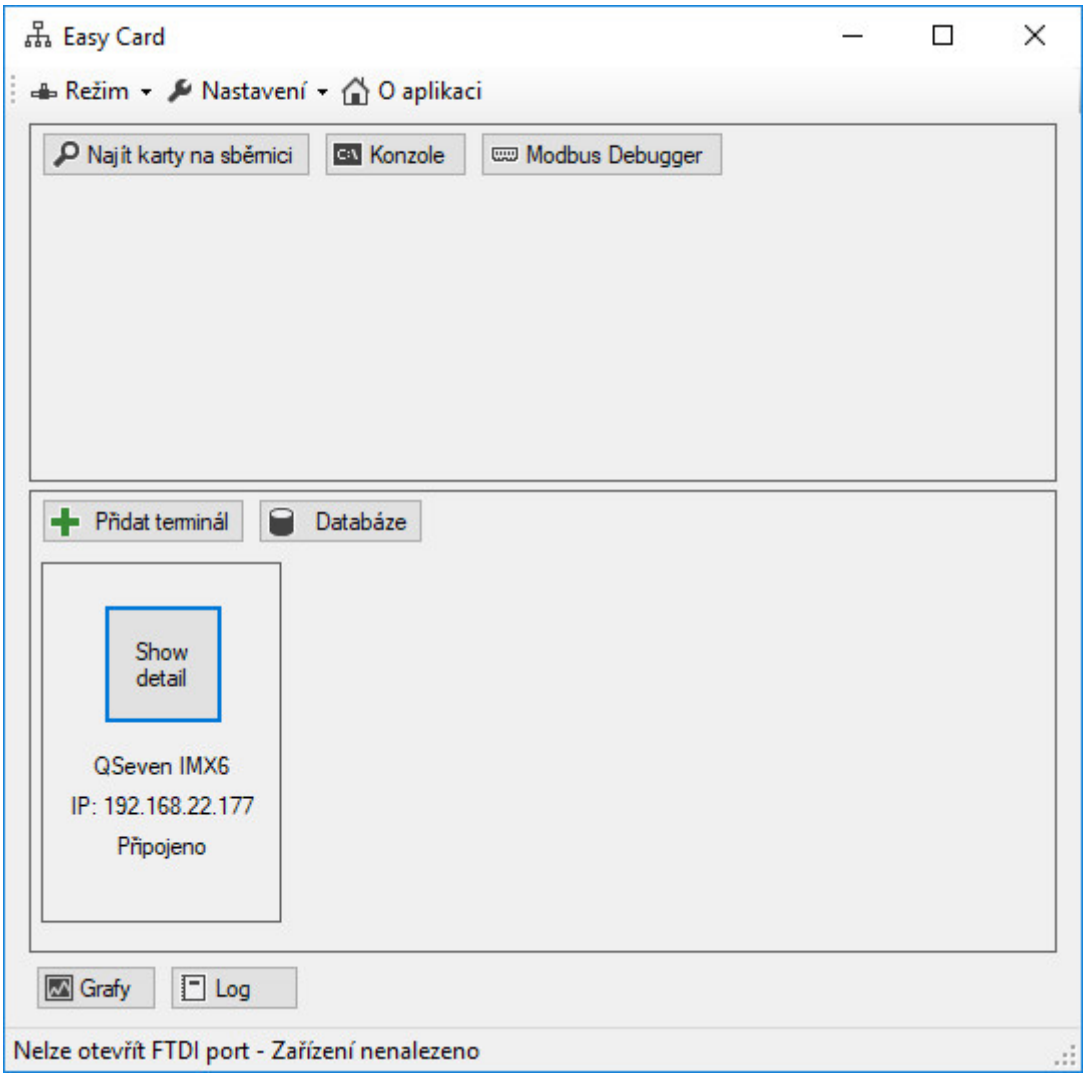
7.1.3 Nahrávání souboru

Je realizováno pomocí komponenty *BackgroundWorker*, která umožňuje vykonávat úlohy na samostatném vlákně. Úlohu nahrávání lze spustit zavoláním metody *UploadFile* nad objektem terminálu, ovšem pouze pokud neběží jiná úloha. Metodě je nutné předat

parametry o souboru, který má být nahrán a cílové umístění. Tyto parametry jsou dále předány spouštěnému vláknu. To se postará o otevření SCP spojení, nahrání souboru do terminálu a uzavření spojení. Během nahrávání je SCP klientem volána událost *Uploading*, která je dále předávána objektem terminálu a je použita k implementaci progres baru.

7.2 GUI linuxových terminálů

GUI (Grafické uživatelské rozhraní) původní aplikace obsahovalo na hlavním formuláři pouze jeden layout pro zobrazení připojených hardwarových modulů. Protože v tomto layoutu lze zobrazit pouze zařízení, které jsou potomky třídy *CHwCardObj*, byl hlavní formulář rozšířen o další layout, pro zobrazení terminálů, které jsou potomky třídy *CLinuxTerminalBase*. Uživatel může upřednostnit velikost jednoho nebo druhého layoutu pomocí rozdělovače, do kterého jsou oba layouty umístěny. [17]

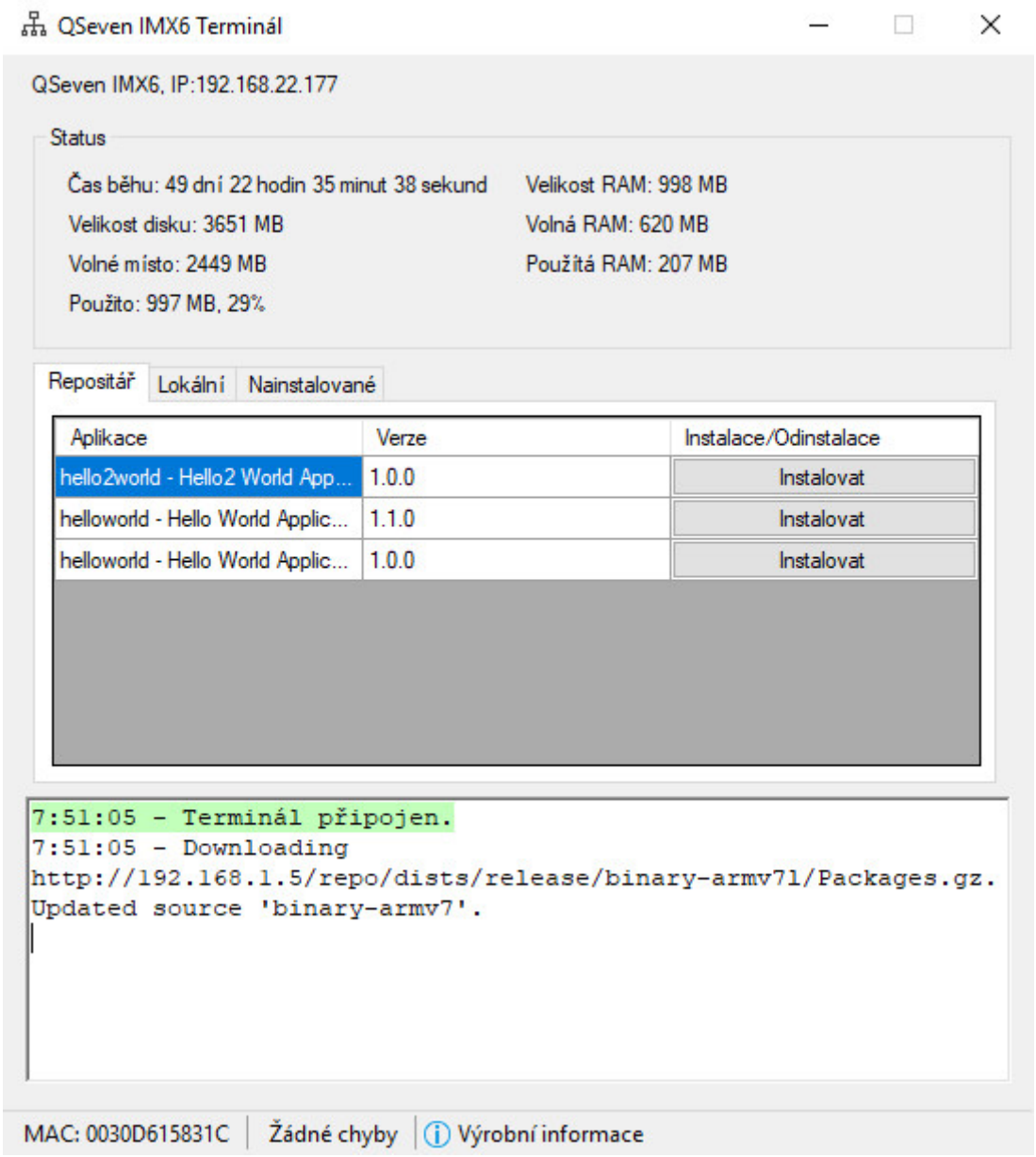


Obr. 19: Layout pro zobrazení linuxových terminálů

7.2.1 Formulář terminálu IMX6

Na hlavním formuláři jsou všechna zařízení zobrazena obecně, tedy jako obecný hardwarový modul nebo obecný linuxový terminál. Po kliknutí na tlačítko detail, je zobrazen formulář dle typu zařízení. [17]

Formulář pro terminál IMX6 je složen ze tří hlavních částí – výpis diagnostických informací, výpis komunikace a záložky pro aktualizaci a instalaci balíčků.



Obr. 20: Formulář terminálu IMX6

7.2.1.1 Diagnostika

Diagnostická data jsou čtena pouze pokud je otevřen formulář. Pomocí časovače jsou zasílány SSH příkazy, které jsou uvedeny v tabulce Tab. 4. Odpovědi na příkazy jsou upravovány na čitelnou podobu a zobrazeny v části status.

`cat /proc/uptime`	Čas, jak dlouho je spuštěn systém Linux.
`df | grep /dev/root`	Informace o velikosti a obsazení flash paměti.
`free -m | grep Mem`	Informace velikosti a obsazení paměti RAM

Tab. 4: SSH příkazy pro čtení diagnostických informací.

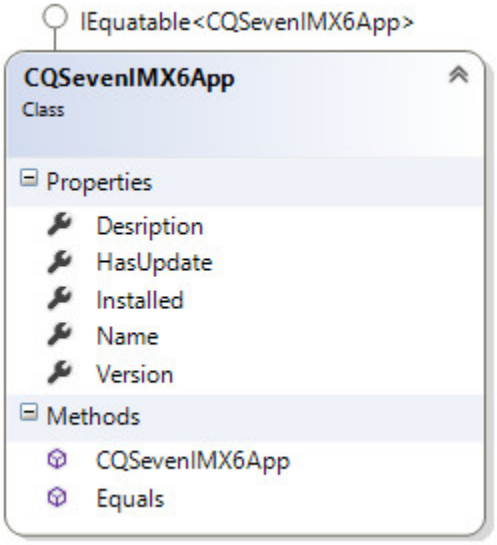
7.2.1.2 Instalace a aktualizace balíčků

Část pro instalace balíčků je rozdělena do tří záložek – repozitář, lokální a nainstalované. Na záložce repozitář jsou zobrazeny balíčky dostupné přímo z repozitáře, který je spuštěn na firemním serveru. Záložka lokální slouží k instalaci balíčku z počítače uživatele a na záložce nainstalované je zobrazen seznam balíčků nainstalovaných v terminálu. Aktualizace obsahu záložek je prováděna pomocí následujících příkazů.

`opkg –f /etc/opkg.conf update`	Provede aktualizaci seznamu dostupných balíčků v repozitáři.
`opkg –f /etc/opkg.conf list`	Vypíše seznam dostupných balíčků v repozitáři.
`opkg –f /etc/opkg.conf list-installed`	Vypíše seznam nainstalovaných balíčků.
`opkg –f /etc/opkg.conf list-upgradable`	Vypíše seznam balíčků, pro které je dostupná novější verze v repozitáři.
`opkg –f /etc/opkg.conf info`	Vypíše kompletní informace o instalovaných balíčcích včetně data instalace.

Tab. 5: SSH příkazy pro výpisy informací o balíčcích.

Z výsledků příkazů uvedených v Tab. 5 je sestaven list balíčků, který je typu *BindingList* a implementuje rozhraní pro propojení s komponentou *DataGrid*. Při správném propojení komponenta automaticky aktualizuje výpis při změně dat v listu. Prvky listu jsou objekty třídy *CQSevenIMX6App*, kde každý objekt představuje jeden balíček. Vlastnosti balíčku, jako *Installed* nebo *HasUpdate*, jsou použity ke skrytí nebo zobrazení instalačních tlačítek na komponentě *DataGrid*. Kvůli porovnání balíčků a omezení duplicit ve výpisu, třída implementuje Interface *IEquatable*. [25]



Obr. 21: Třída balíčku.

Pomocí instalačních tlačítek, lze balíček nainstalovat, odinstalovat nebo aktualizovat. Každé z těchto tlačítek generuje příslušný SSH příkaz:

`opkg -f /etc/opkg.conf install app_name`	Nainstaluje balíček. App_name může obsahovat jméno balíčku v repozitáři, nebo cestu k lokálnímu balíčku.
`opkg -f /etc/opkg.conf remove app_name`	Odstraní nainstalovaný balíček.
`opkg -f /etc/opkg.conf upgrade app_name`	Aktualizuje nainstalovaný balíček.

Tab. 6: SSH příkazy pro instalaci balíčků.

Informace o výsledku zpracování příkazu, je doručena pomocí události. Aby po instalaci balíčku byla zajištěna aktualizace výpisu balíčků, je na událost navázáno čtení listu instalovaných balíčků.

8 VYHLEDÁNÍ TERMINÁLŮ NA SÍTI

Jelikož existuje více možností pro vyhledání zařízení na sítí a každé má svá úskalí, bylo nutné zvolit nejvhodnější řešení pro tuto aplikaci.

Během práce byla testována utilita nmap, která je poměrně populární v oboru skenování sítě. Umožňuje oskenovat vybrané porty celé podsítě a dává na výběr spoustu typů skenování. Například UDP, ICMP Echo nebo TCP SYN. Nmap bohužel není standardní součástí Windows a vyžaduje instalaci. Proto nebyl vybrán. [26]

Další uvažované řešení spočívalo ve vytvoření skriptu na terminálu, který by po připojení k síti zajistil odeslání identifikace na firemní server. Aplikace by si informace o dostupných terminálech stahovala pouze ze serveru. Toto řešení je úsporné na síťový provoz, nicméně nepokrývá možnost, kdy se terminál nachází mimo firmu.

Poslední možností, která byla nakonec zvolena, je všesměrové vysílání UDP datagramů z aplikace. Nevýhodou všesměrového vysílání je, že může být blokován firemními firewally. Výhodou je, že v odpovědi lze získat podrobnější informace o zařízení a poměrně jednoduchým způsobem lze kontaktovat všechna zařízení na síti. Vzhledem k tomu, že instalace balíčků bude probíhat převážně mimo firemní síť, bylo zvoleno toto řešení.

8.1 Zpracování UDP požadavků na terminálu

Aby byl terminál detekovatelný pomocí všesměrového vysílání UDP, je nutné mít spuštěný nějaký proces, který bude naslouchat na domluveném portu a při příchozí zprávě odpoví odesílateli. K tomuto účelu byl vytvořen démon, který je automaticky spouštěn po startu operačního systému. Jeho jediným úkolem je naslouchat na portu a v případě příchozí UDP zprávy na tuto zprávu odpovědět. Démon je pojmenován *svcs-idudpd*.

8.1.1 Linuxový démon

Linuxový démon je proces, který zpracovává nějakou úlohu na pozadí systému. Obvykle má uzavřené standardní vstupy a výstupy, je odpojen od terminálu a je spouštěn po startu systému. V jazyce C je možné obyčejný program změnit na démona pomocí volání několika funkcí knihovny glibc během spouštění programu. [27] [3]

8.1.1.1 Inicializace

Pokud je z terminálu spuštěn nějaký proces, je tento terminál obsazen, dokud není proces ukončen. Aby bylo možné terminál uvolnit a současně nechat proces běžet, je nutné vytvořit novou kopii běžícího procesu pomocí funkce `fork()`. Funkce vytvoří novou kopii adresného prostoru procesu, který dále poběží na pozadí. Rodičovský proces může být ukončen a tím dojde k návratu do terminálu. [28]

```
pid_t pid = fork();

if( pid > 0 )
{
   exit( 0 );              /* Ukončení rodičovského procesu */
}
```

Obr. 22: Fork a ukončení rodičovského procesu

V dalším kroku se nastaví nové session id pomocí funkce `setsid()`. Linux sdružuje pod jedním session id procesy spuštěné z jednoho terminálu. Aby nedošlo k ukončení procesu při odhlášení od terminálu, je potřeba session id změnit a vyloučit tak proces z původní session. Dále proces registruje zachytávání signálů SIGHUP a SIGTERM pro své restartování a ukončení. [27] [28]

Aby nedocházelo k náhodným problémům při odpojování disků po spuštění procesu démona, je dobré změnit jeho pracovní adresář na bezpečné definované místo, například na „`/`". To lze udělat zavoláním funkce `chdir()`. [27] [28]

Pokud všechny uvedené kroky proběhly v pořádku, proces jako poslední krok při spouštění uzavře standardní vstupy a výstupy STDIN, STDOUT a STDERR. Protože pracuje na pozadí, tyto kanály nepotřebuje. Pokud by některá funkce předpokládala jejich otevření, lze je přesměrovat do `/dev/null`. [29]

```
close( STDIN_FILENO );   // Descriptor 0
close( STDOUT_FILENO );  // Descriptor 1
close( STDERR_FILENO );  // Descriptor 2

open( "/dev/null", O_RDONLY );
open( "/dev/null", O_WRONLY );
open( "/dev/null", O_RDWR );
```

Obr. 23: Přesměrování standardních vstupů a výstupů do /dev/null

Zavoláním funkce `open()` hned po zavření standardních vstupů a výstupů je docíleno jejich přesměrování, protože open vždy vrací nejnižší volný deskriptor. [29]

8.1.1.2 Činnost

Jakmile je proces bez chyby inicializovaný, je připraven vykonávat svou činnost. V tomto případě obsadí první volný port z rozsahu čísel definovaných v konfiguračním souboru a čeká na UDP požadavek pro identifikaci. Celý systém je procesem zatěžován jen minimálně, jelikož veškerá činnost je řízena pouze Qt signály a odpověď s identifikací je sestavena pouze jednou při spuštění procesu. Odpověď je v ASCII formátu a obsahuje informaci o výrobci, procesoru a verzi Linuxu. Informace o procesoru je čtena ze souboru /proc/cpuinfo a informace o verzi Linuxu je čtena ze souboru /proc/version.

```
VN:SVCS Process Innovation s.r.o
HW:Freescale i.MX6 Quad/DualLite (Device Tree)
SW:Linux version 4.1.15+gb63f3f5 (gcc version 6.2.0)
SU:BVz79327tk1/vZ6IzKYhiKAcsHM=
```

Obr. 24: Odpověď s identifikací terminálu

Na konec odpovědi je připojen kontrolní součet SHA256 převedený do formátu Base64 a celá odpověď je zašifrována symetrickou šifrou AES.

Pro AES šifrování je použit následující klíč a inicializační vektor:

Šifrovací klíč	`byte[] { 0x61,0x9c,0x8a,0x1c,0xaf }`
Inicializační vektor	`byte[] { 0x5a,0x3c,0x65,0x11,0x22 }`

Tab. 7: Klíč a inicializační vektor pro AES šifrování

Odpovědi jsou zasílány pouze, pokud jsou příchozí požadavky ve stejném formátu jako odpověď. Tedy zašifrovány šifrou AES s uvedenými klíči a s korektním kontrolním součtem na konci zprávy. Obsahem požadavku musí být řetězec „`SU:SVCS IDENTIFICATION`“.

```
CM:SVCS IDENTIFICATION
SU:rC2AeRQ8aQ05/na5Ay6fiekMF5xQ
```

Obr. 25: Požadavek na identifikaci terminálu

8.1.1.3 Automatické spuštění

V linuxové distribuci terminálu je použit pro zavádění systému Systém V init. Tento systém používá pro automatické spouštění procesů skripty napsané ve skriptovacím jazyce bash. Proto byl pro spouštění démona vytvořen skript *svcs-idudpd*, který je umístěn na terminálu v adresáři */etc/init.d/*. Tento skript umožňuje démona spustit, ukončit a restartovat.

```
NAME=svcs-idudpd
DAEMON=/usr/bin/$NAME
CONFIG=/etc/$NAME.cfg
SCRIPTNAME=/etc/init.d/$NAME

DAEMON_ARGS="${PORTNUMBERLOW} ${PORTNUMBERHI}"
#
# Function that starts the daemon/service
#
do_start()
{
        $DAEMON $DAEMON_ARGS
}
```

Obr. 26: Funkce pro spuštění démona ve spouštěcím skriptu.

Po nakopírování skriptu do terminálu byly aktualizovány odkazy na spouštěcí skripty pomocí příkazu:

```
update-rc.d svcs-idudpd default
```

Spouštěcí skript je umístěn v příloze P1.

8.1.1.4 Konfigurace

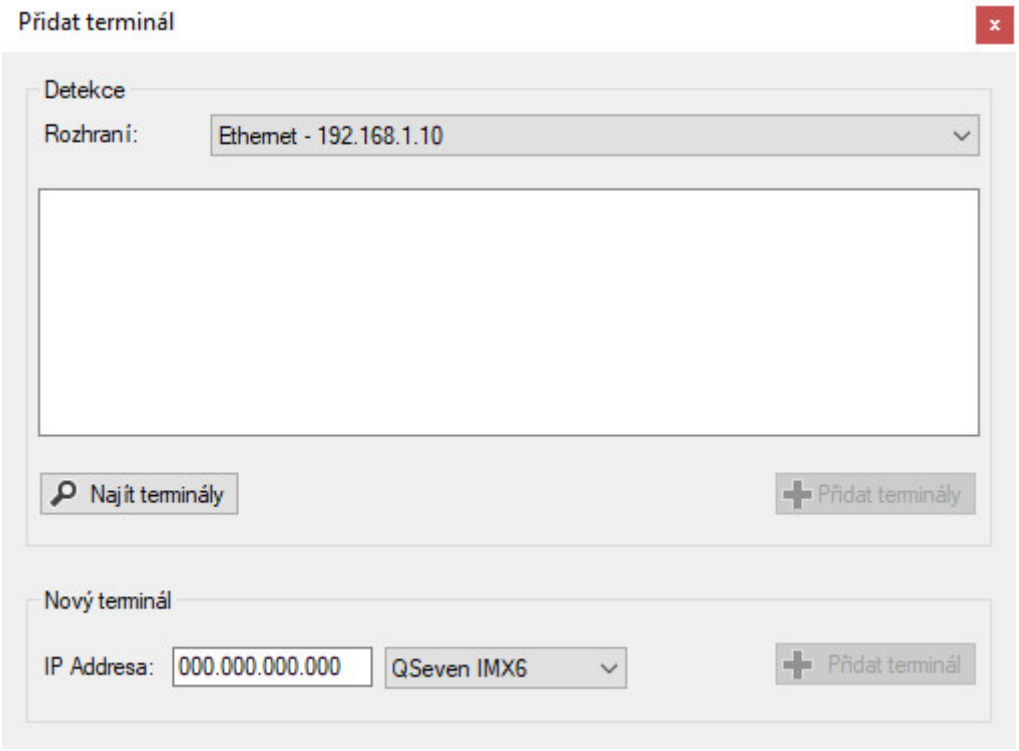
Pro konfiguraci portů, na kterých bude démon naslouchat, byl vytvořen konfigurační soubor *svcs-idudpd.cfg*, který je umístěn v adresáří */etc*. V tomto souboru lze definovat rozsah čísel portů, ze kterých démon vybere první volný. Číselné hodnoty portů jsou předány parametrem při spuštění démona.

```
# Listening on this port.
PORTNUMBERLOW=49300
PORTNUMBERHI=49305
```

Obr. 27: Rozsahy portů pro naslouchání.

8.2 GUI vyhledávání linuxových terminálů

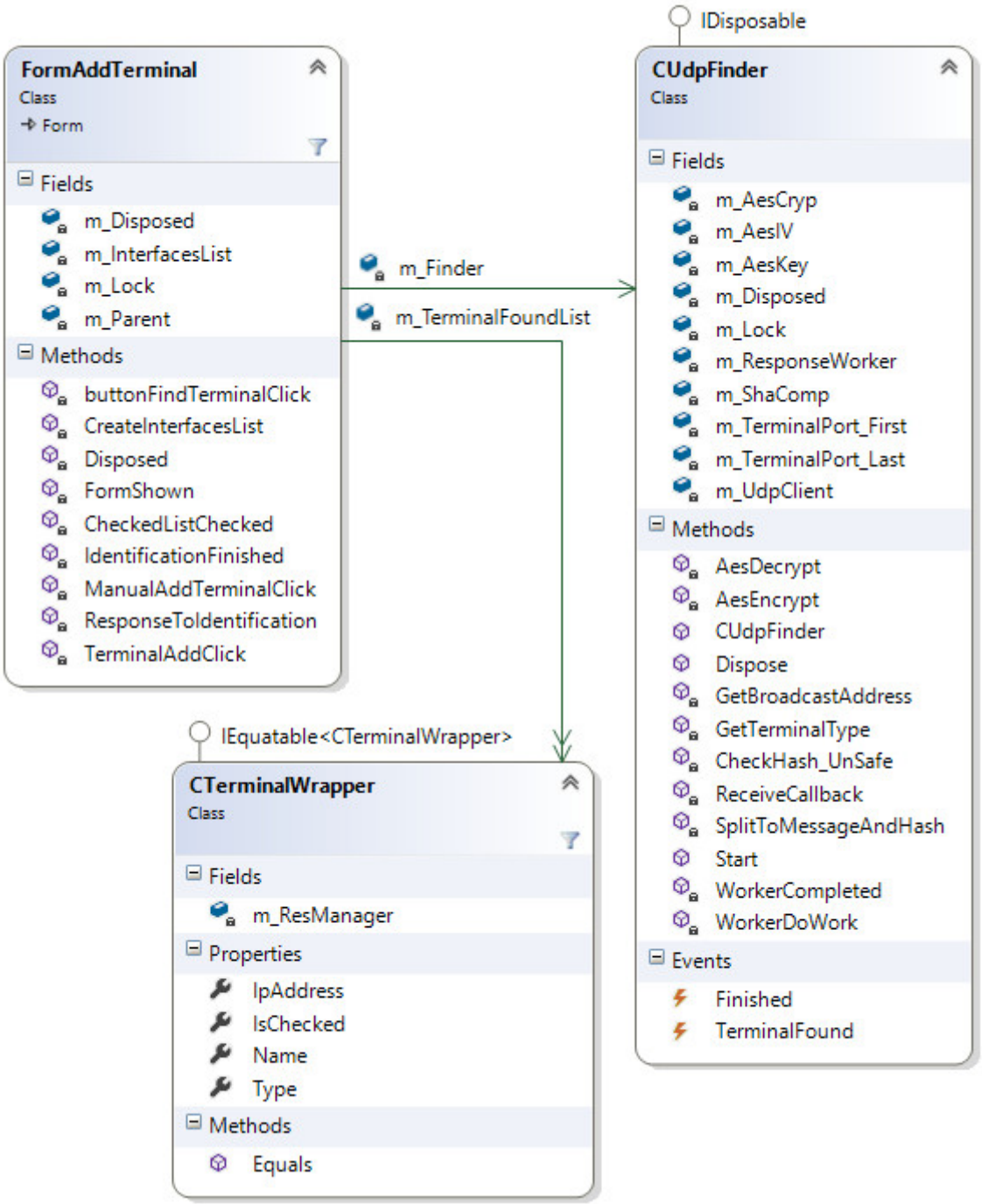
Je realizováno pomocí samostatného formuláře přístupného z hlavního okna aplikace. Uživatel se může pokusit o automatické vyhledání terminálů skrze zvolené síťové rozhraní, nebo v případech, kdy je na síti blokováno všesměrové UDP vysílání, může IP adresu terminálu zadat ručně.



Obr. 28: Formulář pro vyhledání terminálů.

Pro automatické vyhledávání je v rámci formuláře vytvořen objekt třídy *CUdpFinder*, který zajišťuje vysílání požadavků a zpracování odpovědí. Automatické vyhledávání lze zahájit stiskem tlačítka najít terminály, které volá metodu Start a jako parametr je předávána reference na síťové rozhraní. Metoda sestaví zprávu a zahájí všesměrové vysílání skrze zvolené síťové rozhraní. Zpráva je vysílána na porty terminálu 49300 až 49305 na nichž může naslouchat linuxový démon. Po odeslání je spuštěno vlákno řízené komponentou *BackgroundWorker*, které čeká na příchozí odpovědi. Pokud nějaká odpověď dorazí, proběhne dešifrování AES, ověří se přítomnost známých znaků a ověří se kontrolní součet. Následně je dekódován typ terminálu a vyvolána událost *TerminalFound*. Pokud po dobu tří sekund nedorazí žádná další zpráva, vlákno ukončí svoji činnost. Ukončení vyhledávání je signalizováno událostí *Finished*.

Z detekovaných terminálů je sestaven list objektů třídy *CTerminalWrapper,* kde každý objekt uchovává IP adresu a typ terminálu. Po dokončení vyhledávání uživatel z listu vybere terminály, které chce přidat a až po stisku tlačítka přidat jsou vytvořeny nové instance terminálů.



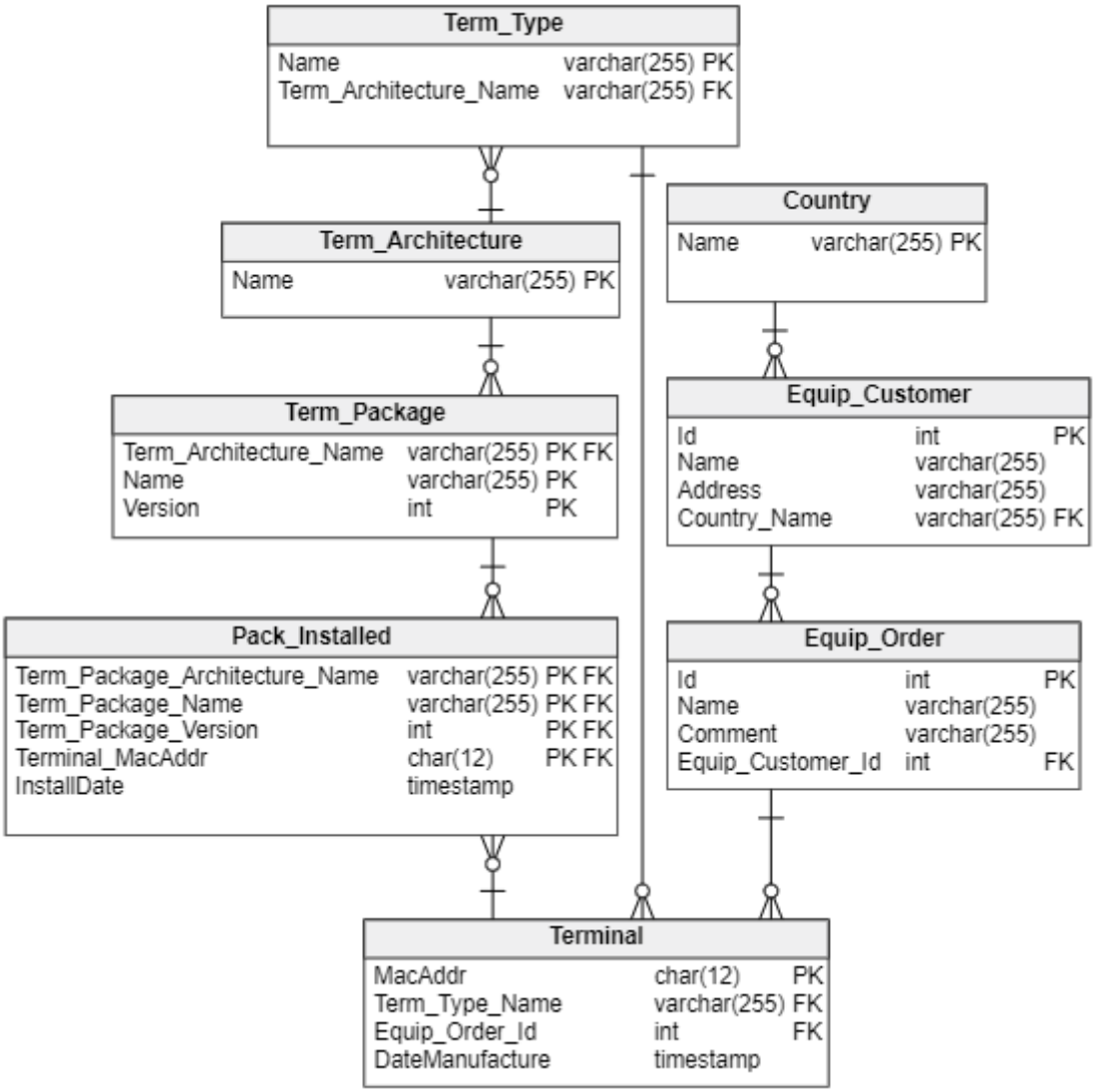
Obr. 29: Diagram tříd vyhledávání terminálů

9 DATABÁZE TERMINÁLŮ

Před samotným návrhem databáze, byly nejdříve vytipovány požadavky na informace, které mají být uchovávány.

- Typ terminálu s definicí architektury – pro budoucí odlišení více druhů terminálů.
- Nainstalované balíčky – možno instalovat více balíčků do jednoho terminálu.
- Datum výroby.
- Přiřazení terminálu ke konkrétní zakázce.
- Informace o zakázce – komentář, zákazník, adresa, země.

9.1 Návrh databáze



Obr. 30: Návrh databáze

9.1.1 Popis tabulek

- Term_Type – typy terminálů, které budou v budoucnu vyráběny.
- Term_Architecture – architektury terminálů, je součásti typu terminálů.
- Term_Package – balíčky dostupné ze serverového repozitáře. Balíček je jednoznačně identifikován jménem, verzí a architekturou terminálu.
- Pack_Installed – nainstalované balíčky s datem instalace. Instalace je jednoznačně identifikována balíčkem z tabulky Term_Package a MAC adresou terminálu.
- Terminal – vyrobené terminály. Terminál je jednoznačně identifikován MAC adresou. Dále má přiřazen typ, id zakázky a datum výroby.
- Country – země, odkud může pocházet zákazník.
- Equip_Customer – zákazníci, kterým bylo dodáno nějaké zařízení s terminálem.
- Equip_Order – zakázky.

9.2 Komunikace s databází

Pro tuto práci bude využíván firemní databázový server, na kterém je používána databáze MySQL. V budoucnu je plánován přechod na databázi MariaDB, která je plně kompatibilní s MySQL. MariaDB nabízí mnohá vylepšení a především aktivní řešení bezpečnostních chyb a vydávání aktualizací.

Pro připojení k MySQL databázi pomocí .NET aplikace, je potřeba nainstalovat konektor, který implementuje rozhraní ADO.NET. A pro usnadnění vývoje a prohlížení databáze přímo z Visual Studia, je možné doinstalovat doplněk MySQL for Visual Studio integration, který od verze 6.7 již není součástí konektoru. [30] [31]

Komunikaci s databází je možné realizovat více přístupy, pro tuto práci bylo uvažováno použití jedné ze tříd My*SqlDataReader* nebo My*SqlDataAdapter*. První z nich nabízí rychlejší přístup a menší spotřebu paměti, protože udržuje v paměti vždy jen jeden záznam. Ale při jeho použití je nutné se starat o otevření a uzavření připojení a neposkytuje zpětný zápis do databáze. SqlDataAdapter poskytuje komfortnější přístup za cenu vyšší paměťové režie. Zajišťuje automatické otevření a uzavření spojení a poskytuje jednoduché metody pro zpětnou aktualizaci dat do databáze. Protože navržená databáze má jednoduchou strukturu a výsledky dotazů nebudou obsahovat velké počty záznamů, byl pro komunikaci zvolen *SqlDataAdapter*. [32]

Sestavení a volání SQL dotazů zajišťuje třída *CLinuxTerminalDatabase.* Reference na objekt této třídy je předávána všem objektům, které potřebují získat nebo aktualizovat databázová data. Dotazy, do nichž je předávána nějaká hodnota, jsou vytvořeny pomocí zástupných jmen. Na rozdíl od přímého zápisu příkazu, jsou vstupní data zpracována pouze jako literálové hodnoty. Tento přístup efektivně zabraňuje SQL injection. [33]

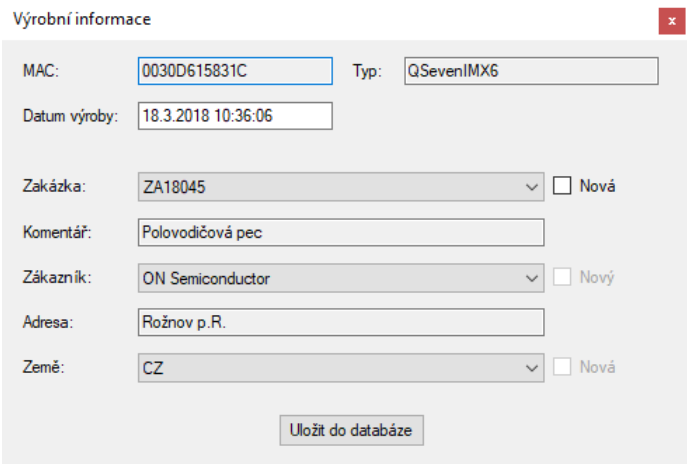
```
MySqlCommand Cmd = new MySqlCommand(
       "INSERT INTO Equip_Order ( Name, Comment, CustomerId ) " +
       "VALUES ( @OrderName, @Comment, @CustomerId);"
       , DbConnection );

Cmd.Parameters.AddWithValue( "@OrderName",  RowData["Name"] );
Cmd.Parameters.AddWithValue( "@Comment",    RowData["Comment"] );
Cmd.Parameters.AddWithValue( "@CustomerId", RowData["CustomerId"] );
```

Obr. 31: Přiklad SQL příkazu s použitím zástupných jmen.

9.3 Vkládání výrobních informací

Výrobní informace jako číslo zakázky, zákazník nebo datum výroby, je možné změnit prostřednictvím formuláře, který lze zobrazit po připojení k terminálu. Tento formulář umožňuje vybrat z existujících položek, nebo umožňuje vytvořit novou položku. Existující položky jsou nabízeny ze seznamu, který je pouze pro čtení a zobrazuje data z datového zdroje – tabulky vytvořené SQL dotazem. Při volbě nové položky, je zrušena vlastnost pouze pro čtení a upravený text je ukládán do lokální kolekce formuláře. Všechny nové záznamy jsou vloženy do databáze až při stisku tlačítka uložit.



Obr. 32: Formulář pro vkládání výrobních informací.

Uvedené informace jsou ukládány pouze do firemní databáze a budou sloužit především pro identifikaci terminálů při kontaktu se zákazníkem a pro celkový přehled.

9.4 Lokální úložiště offline instalací.

V případech, kdy je pomocí uživatelské aplikace instalován balíček a není k dispozici připojení k databázi, používá aplikace své lokální úložiště ve formě XML souboru. Jakmile je připojení k databázi funkční, jsou informace o provedené instalaci uloženy do databáze. Tuto funkci implementuje třída *QSevenIMX6LocalDb*, která je potomkem třídy *CLinuxTerminalDatabase.*

Třída pro lokální ukládání používá *DataSet*, který obsahuje tabulku terminálů a tabulku instalovaných balíčků se stejnou strukturou jako v databázi. *DataSet* implicitně umožňuje svůj obsah uložit nebo načíst z XML souboru.

```
<LocalDataset>
  <InstalledPackage>
    <TerminalMac>0030D615831C</TerminalMac>
    <PackName>helloworld</PackName>
    <PackVersion>65792</PackVersion>
    <PackArch>armv7l</PackArch>
    <InstallDate>2018-03-29T06:20:22+02:00</InstallDate>
  </InstalledPackage>
  <InstalledPackage>
    <TerminalMac>0030D615831C</TerminalMac>
    <PackName>hello2world</PackName>
    <PackVersion>65536</PackVersion>
    <PackArch>armv7l</PackArch>
    <InstallDate>2018-03-29T06:24:50+02:00</InstallDate>
  </InstalledPackage>
</LocalDataset>
```

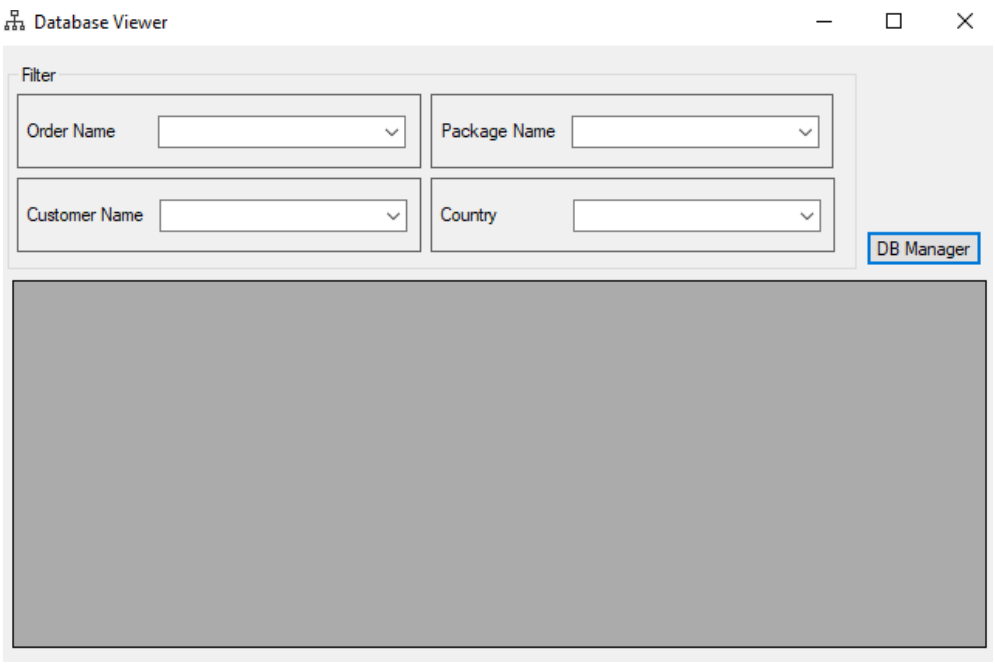
Obr. 33: XML soubor s lokálně uloženými informacemi o instalacích.

Pro správnou funkci lokálního ukládání, je nutné vždy při změně připojení aktualizovat seznam terminálů metodou *SetTerminalList*. Metoda pro každý terminál zaregistruje událost příchozích příkazů a změny připojení. Na základě těchto událostí jsou prováděny záznamy do lokální databáze. Třída dále využívá příkaz opkg-info, který je zasílán ihned po připojení terminálu k síti. Příkaz vrací informace o všech instalovaných balíčcích, které jsou rovněž ukládány do lokálního úložiště. Díky tomu, je vždy pouhým připojením terminálu zajištěna aktualizace databáze nejaktuálnějšími informacemi o všech instalovaných balíčcích.

O kontrolu dostupnosti připojení a následnou aktualizaci se stará vlákno, které je spouštěno každých pět sekund. Vlákno nejprve prochází lokální tabulku terminálů a následně lokální tabulku instalovaných balíčků. Pokud se nějaký ze záznamů nenachází v databázi, je vložen a následně je celý lokální dataset smazán.

9.5 Procházení databáze

Prohlížeč databáze je vytvořen ve zvláštním okně, které je přístupné přímo z hlavního formuláře. Jedná se o jednoduché okno s komponentou datagrid, kterému je jako datový zdroj přiřazena tabulka vytvořená z SQL dotazu. Prohlížeč umožňuje nastavit filtry na číslo zakázky, jméno balíčku, jméno zákazníka a zemi odkud zákazník pochází. Sestavení příslušného SQL dotazu zajišťuje metoda *GetTerminalsFiltered*, která jako parametr přijímá filtry definované slovníkovou kolekcí.



Obr. 34: Formulář pro prohlížení databáze.

10 PŘÍNOSY SYSTÉMU A JEHO DALŠÍ VÝVOJ

Ve firmě SVCS Process Innovation, která používá embedded terminály k řízení průmyslových zařízení, byly aplikace do těchto terminálů instalovány stylem "ručního" nahrávání souborů. Tento způsob je nekomfortní a nevhodný zejména pro nezkušené techniky. Systém vyvinutý v této diplomové práci se snaží instalace aplikací zjednodušit, omezit chybovost a evidovat informace o všech terminálech.

Od systému jsou očekávány následující přínosy:

- Možnost šíření aplikací pomocí balíčků.
- Zjednodušení instalace aplikací do terminálů.
- Vyšší zabezpečení díky kontrolním součtům celého balíčku.
- Zamezení manipulace s citlivými soubory.
- Možnost řízení aktualizací pomocí skriptů.
- Evidence vyrobených terminálů a instalovaných balíčků v databázi.
- Zajištění aktuálnosti databáze i po aktualizaci na novou verzi balíčku.

Prvních pět bodů je spolehlivě zajištěno pomocí linuxového balíčkovacího systému, který se stará o rozbalení, kontrolu a instalaci balíčků. Aktuálnost databáze je závislá především na uživateli, který musí být připojen k firemní databázi. Protože systém umožňuje balíčky instalovat i mimo firemní síť, existuje zde riziko, že se informace o instalaci do databáze nedostane. Pro tento případ aplikace ukládá informace na počítači uživatele a záleží pouze na něm, zda se připojí k databázi a dovolí informace aktualizovat.

10.1 Další vývoj systému

Balíčkovací systém, který byl na terminálu spuštěn, otevírá cestu k instalaci balíčků z flash disku. Tím by došlo k dalšímu zjednodušení instalací. Dále by bylo možné systém rozšířit o evidenci hardwarových modulů, které s terminálem tvoří kompletní řídicí systém. Těmto modulům se věnovala bakalářská práce Řízení a monitoring hardwarových modulů. A v neposlední řadě by bylo vhodné vylepšit GUI pro instalace balíčků a prohlížení databáze. [17]

ZÁVĚR

Během této práce byl vytvořen systém, jehož hlavním přínosem je omezení manipulace s citlivými soubory, které jsou nahrávány do terminálu. Dále, zajištění aktuálního přehledu o terminálech včetně instalovaných aplikacích a možnost jednoduchým způsobem instalovat a aktualizovat aplikace v embedded terminálech. Systém je navržen tak, že i pro nezkušeného uživatele nebude instalace překážkou.

Práce byla zahájena výběrem vhodného systému pro instalaci aplikací. Jednou z možností bylo vytvoření vlastního návrhu. Tato varianta nebyla zvolena, protože s instalací můžou být spojeny další navazující úkony při aktualizacích nebo při zajištění kompatibility. Jelikož se na Linuxu běžně používají balíčkovací systémy, byl vytipován jednodušší systém pro použití na embedded zařízeních.

Po seznámení se s balíčkovacím systémem a jeho typem balíčků, byl na firemním serveru vytvořen softwarový repozitář s testovacími balíčky. Dále byla na serveru zřízena databáze pro ukládání informací o instalacích a vyrobených terminálech.

Pro instalace aplikací pomocí balíčků, bylo nutné ze zdrojových souborů přeložit balíčkovacího klienta, který bude spustitelný na embedded terminálech. Klienta bylo nutné přeložit s přepínačem pro platformu arm-linux. Po pár neúspěšných pokusech se překlad podařil a bylo možné otestovat první instalaci pomocí tohoto systému.

K ovládání balíčkovacího klienta, byla zvolena aplikace, která byla vytvořena v rámci bakalářské práce. Tuto aplikaci bylo nutné rozšířit o několik funkčností. Jako první byla vybrána knihovna pro SSH komunikaci a bylo implementováno komunikační jádro. Po úspěšně navázaném spojení s terminálem byla plně implementována podpora instalací a aktualizací aplikací pomocí balíčkovacího klienta. Dále byla do aplikace přidána komunikace s firemní databází a byl vytvořen systém pro dočasné ukládání informací do lokálního úložiště. Jakmile je dostupná firemní databáze, aplikace informace aktualizuje do firemní databáze.

Systém je již testován ve firemním prostředí a postupně bude aplikován na vyráběné terminály. Pokud se osvědčí a nebude vykazovat nedostatky, bude postupně uvolněn i pro použití firemními techniky.

SEZNAM POUŽITÉ LITERATURY

1. *Linux: dokumentační projekt. 4., aktualiz. vyd. Přeložil Lubomír PTÁČEK.* Brno : Computer Press, 2007. ISBN 978-80-251-1525-1.

2. **Jones, M.Tim.** IBM developerWorks. *Anatomy of the Linux kernel.* [Online] 6. 6 2007. [Citace: 2. 2 2018.] https://www.ibm.com/developerworks/library/l-linux-kernel/.

3. The GNU C Library (glibc). *What is glibc?* [Online] [Citace: 2. 2 2018.] https://www.gnu.org/software/libc/.

4. **Došek, Roman.** *OS Linux na platformě ARM.* 2013 : Univerzita Tomaše Bati ve Zlíně, 2013.

5. **Jelínek, Lukáš.** *Vytváříme vlastní distribuci linuxu.* místo neznámé : COMPUTER PRESS, 2010. 978-80-251-2433-8.

6. Debian Policy Manual. *Debian Policy Manual.* [Online] 27. 12 2017. [Citace: 8. 2 2018.] https://www.debian.org/doc/debian-policy/.

7. **Peterka, Jiří.** Archiv článků a přednášek Jiřího Peterky. [Online] [Citace: 23. 1 2018.] http://www.earchiv.cz/a93/a303c110.php3.

8. Wikipedie. *TCP/IP.* [Online] [Citace: 2. 2 2018.] https://cs.wikipedia.org/wiki/TCP/IP.

9. Fakulta elektrotechniky a informatiky, Univerzita Pardubice. *studijní materiály - FEI - [UPCE].* [Online] [Citace: 5. 2 2018.] http://www.fei.doevil.cz/?dir=.%2Ffei%2F2_rocnik%2F%5BZS%5D+IPS1+-+Site+1%2FTeorie+LEARN.

10. **DOSTÁLEK, Libor a KABELOVÁ, Alena.** *Velký průvodce protokoly TCP/IP a systémem DNS.* Brno : Computer Press, a.s., 2008. 978-80-251-2236-5.

11. DSL.cz. *SSH – bezpečné používání vzdáleného počítače a kopírování dat.* [Online] [Citace: 29. 1 2018.] http://www.dsl.cz/jak-na-to/jak-na-ssh.

12. **SSH Communications Security.** SSH Communications Security. *SSH (SECURE SHELL).* [Online] 2017. https://www.ssh.com/ssh/.

13. **PETERKA, Jiří.** CZ.NIC. *Báječný svět elektronického podpisu.* [Online] 2010. https://knihy.nic.cz/files/edice/bajecny_svet_elektronickeho_podpisu_cznic.pdf.

14. **PRICE, Jason.** *C# programování databází.* Praha : Grada Publishing, a.s., 2005. 80-247-0982-1.

15. Microsoft docs. *ADO.Net Overview.* [Online] 30. 3 2017. https://docs.microsoft.com/cs-cz/dotnet/framework/data/adonet/ado-net-overview.

16. **ALBAHARI, Joseph a ALBAHARI, Ben.** *C# 5.0 in a Nutshell: Fifth Edition.* Sebastopol : O'Reily Media, 2012. 978-1-449-32010-2.

17. **Gargulák, Jaroslav.** *Řízení a monitoring hardwarových modulů.* UTB ve Zlíně, Fakulta aplikované informatiky, 2016. .

18. Wiki Debian. *Debian Repository Format.* [Online] 13. 11 2017. [Citace: 22. 12 2017.] https://wiki.debian.org/DebianRepository/Format.

19. Ask Ubuntu. *How to create a local APT repository?* [Online] 30. 6 2012. [Citace: 22. 12 2017.] https://askubuntu.com/questions/170348/how-to-create-a-local-apt-repository.

20. yocto project. *Opkg.* [Online] [Citace: 2. 2 2018.] https://www.yoctoproject.org/tools-resources/projects/opkg.

21. Linaro. [Online] [Citace: 22. 12 2017.] https://www.linaro.org/downloads/.

22. libarchive. *Multi-format archive and compression library.* [Online] [Citace: 2. 2 2018.] https://www.libarchive.org/.

23. OpenWrt. *OPKG Package Manager.* [Online] [Citace: 22. 12 2017.] https://wiki.openwrt.org/doc/techref/opkg.

24. Github. *SSH.NET.* [Online] [Citace: 10. 02 2018.] https://github.com/sshnet/SSH.NET.

25. **TROELSEN, Andrew W.** *C# a .NET 2.0 profesionálně.* Brno : Zoner Press, 2006. 80-86815-42-0.

26. NMAP.ORG. [Online] [Citace: 02. 02 2018.] https://nmap.org/.

27. stack overflow. *Creating a daemon in Linux.* [Online] 29. 12 2016. [Citace: 17. 2 2018.] https://stackoverflow.com/questions/17954432/creating-a-daemon-in-linux.

28. sallyx.org. *Démoni.* [Online] 10. 10 2014. https://www.sallyx.org/sally/c/linux/daemon.

29. die.net. [Online] [Citace: 17. 2 2018.] https://www.die.net/.

30. **Oracle Corporation.** MySQL Documentation. [Online] 2017. https://dev.mysql.com/doc/.

31. MySQL. *Download Connector/Net.* [Online] [Citace: 3. 4 2018.] https://dev.mysql.com/downloads/connector/net/.

32. **Coehoorn, Joel.** stack overflow. *SqlDataAdapter vs SqlDataReader.* [Online] 4. 11 2009. https://stackoverflow.com/questions/1676753/sqldataadapter-vs-sqldatareader.

33. Microsoft docs. *Configuring Parameters and Parameter Data Types.* [Online] 30. 3 2017. https://docs.microsoft.com/cs-cz/dotnet/framework/data/adonet/configuring-parameters-and-parameter-data-types.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASCII	American Standard Code for Information
DHCP	Dynamic Host Configuration Protocol
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
MAC	Media Access Control
POP3	Post Office Protocol
SCP	Secure Contain Protect
SFTP	Secure File Transfer Protocol
SHA	Secure Hash Algorithm
SIGHUP	Signal Hang Up
SIGTERM	Signal Terminate
SSH	Secure Shell
TCP	Transmission Control Protocol.
TTL	Time to Live
UDP	User Datagram Protocol.
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje:

fulltext.pdf	Diplomová práce
arm-linux	Balíčkovací klient s knihovnou libArchive, který je spustitelný na linuxovém embedded terminálu.
IdentityDaemon	Zdrojové kódy démona pro identifikaci linuxového embedded terminálu, včetně konfiguračního a spouštěcího skriptu.
EasyCard	Zdrojové kódy uživatelské aplikace pro komunikaci s linuxovým embedded terminálem.