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

LINUXOVÁ DISTRIBUCE PRO ŘÍDICÍ GATEWAY CHYTRÝCH BUDOV

LINUX DISTRIBUTION FOR SMART BUILDING GATEWAY

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Tomáš Nagy

AUTHOR

VEDOUCÍ PRÁCE — Ing. Lukáš Jablončík

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Tomáš Nagy ***ID:*** 231258

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Linuxová distribuce pro řídicí gateway chytrých budov

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit linuxovou distribuci pro gateway chytré budovy. Distribuce bude vycházet z běžné verze linuxu, např. Ubuntu, Debian, Fedora apod. V rámci teoretické části se student seznámí s postupy pro redukci systému a požadavky na gateway pro řízení budov. Dále provede analýzu dostupných řešení. V rámci praktické části bude provedena redukce systému, vložení potřebných balíků pro gateway a budou vytvořeny potřebné skripty. Student provede automatizaci procesu, prakticky ověří fungování systému na jednodeskovém počítači s architekturou x86, např. UpBoard. Dále student navrhne aplikaci prvků bezpečnosti potřebných pro zamýšlené použití.

DOPORUČENÁ LITERATURA:

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

[2] COOPER, M. Advanced Bash-Scripting Guide. Lulu.com, 2010. ISBN: 978-14-357-5218-4.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Lukáš Jablončík

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cieľom záverečnej práce bolo vytvoriť Linuxovú distribúciu pre gateway inteligentnej domácnosti. Práca je rozdelená na teoretickú a praktickú časť. V rámci teoretickej časti sa venovalo rozboru operačného systému Linux. Následne sa popisujú a porovnávajú jednodoskové počítače. V ďalšej časti sa pojednáva o problematike komunikačnej brány, o voľbe komunikačného protokolu a kontajnerizácii aplikácii. Výsledkom riešenia danej problematiky bol návrh postupu pridávania nástrojov na domácu automatizáciu, konštrukcia skriptov, ktoré túto problematiku automatizujú a návod na ich správne spustenie.

KĽÚČOVÉ SLOVÁ

Linux, Operačný systém, Ubuntu, SBC, UP Board, Komunikačná brána, Home Assistant, MQTT, Docker, Skript, Automatizácia

ABSTRACT

The goal of the thesis was to create a Linux distribution for a smart home gateway. The thesis is divided into theoretical and practical parts. The theoretical part dealt with the analysis of the Linux operating system. Subsequently, single board computers are described and compared. In the next part, the gateway issues, communication protocol selection and application containerization are discussed. As a result of addressing the issue, a procedure for adding tools for home automation, the construction of scripts that automate this issue and instructions for their proper execution were proposed.

KEYWORDS

Linux, Operating system, Ubuntu, SBC, UP Board, Gateway, Home Assistant, MQTT, Docker, Script, Automation

NAGY, Tomáš. *Linuxová distribuce pro řídicí gateway chytrých budov*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 73 s. Bakalárska práca. Vedúci práce: Ing. Lukáš Jablončík

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Tomáš Nagy
VUT ID autora:	231258
Typ práce:	Bakalárska práca
Akademický rok:	2022/23
Téma závěrečnej práce:	Linuxová distribuce pro řídicí gateway chytrých budov

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som sa poďakoval vedúcemu bakalárskej práce pánovi Ing. Lukášovi Jablončíkovi za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Operačný systém Linux reprezentuje svetovo najrozšírenejší operačný systém použitý v IoT (Internet of Things) komunikačných bránach. Dôvodom je jeho neustále vylepšovanie, rozširovanie a možnosť personalizácie podľa potreby používateľa.

Cieľom bakalárskej práce je naštudovanie si postupov pre redukciu systému, zoznámenie sa s požiadavkami komunikačnej brány a následná tvorba Linuxovej distribúcie pre komunikačnú bránu inteligentnej domácnosti. Táto distribúcia vychádza z bežnej verzie Ubuntu server, ktorá je upravená, aby vyhovovala štandardným potrebám pre riadenie inteligentnej domácnosti. Do systému sú ďalej pridané nástroje na správu budovy, ako Home Assistant, ktorý umožňuje v rámci jednej aplikácie ovládať zariadenia rôznych výrobcov. Celý proces tvorby je za použitia skriptov automatizovaný a je implementovaný návrh na redundanciu systému.

Práca je rozdelená na dve časti, teoretickú a praktickú. Kapitola [1] sa zameriava na popis operačného systému Linux. Predstavujú sa tu jeho najdôležitejšie časti a bližšie sa zoznamuje s distribúciami, ktoré by mohli slúžiť ako operačný systém pre komunikačnú bránu. V kapitole [2] sa popisujú jednodoskové počítače, ich rozdiely oproti stolovým počítačom a využitie v praxi. Taktiež je predstavený UP Board, ako nami zvolený hardvér pre komunikačnú bránu a jeho porovnanie s konkurenčnými alternatívami. Nasleduje kapitola [3], v ktorej sa porovnávajú dve konkurenčné platformy pre domácu automatizáciu a je zvolený najvhodnejší protokol pre prenos správ medzi zariadeniami. Zoznamuje sa tu aj s nástrojom Docker, ktorý za pomoci virtuálizácie uľahčuje inštaláciu a správu našich automatizačných programov. V praktickej časti, v kapitole [4] sa nachádzajú jednotlivé návody a postupy na prípravu vhodnej Linuxovej distribúcie, ktorá slúži ako základ komunikačnej brány. Taktiež sa v nej nachádza detailný popis automatizačných skriptov a pojednáva sa o možných komplikáciách pri inštalácii aplikácii.

1 Operačný systém Linux

Operačný systém OS Linux bol vytvorený Fínskym študentom Linusom Torvaldsom počas jeho štúdií na univerzite v Helsinkách v roku 1991. Jeho hlavným motívom bolo vytvoriť softvér, ktorý bude voľne dostupný pre každého. Kvôli jeho dostupnému zdrojovému kódu si Linux veľmi rýchlo obľúbili viaceré veľké organizácie a firmy, ktoré si ho začali prispôsobovať pre svoje potreby[1]. Najčastejšie sa Linux používa na serveroch, pretože je dostatočne bezpečný a škálovateľný.

Moderné Linuxové distribúcie sú podobné iným operačným systémom, ako napríklad macOS, či Windows, majú grafické používateľské rozhranie, softvér na prezeranie fotiek a videí, tabuľkový procesor a ďalšie nástroje, vďaka ktorým si Linux obľúbili aj bežní používatelia[2].

1.1 Časti operačného systému

V nasledujúcich pododdieloch sú teoreticky rozobrané najdôležitejšie časti operačného systému Linux. Zameriava sa tu na základnú problematiku zavádzačov, popisuje sa jadro a jeho rozdelenie, grafické používateľské rozhranie, aplikácie, ich inštalácia a systémový démoni.

1.1.1 Bootloader

Dáta operačného systému musia byť načítané do pamäte počítača, toto nám je umožnené pomocou **bootloadera** (zavádzač). Pri zapnutí, alebo reštartovaní počítača Basic Input/Output System (BIOS) prevedie základný test systému (POST), ktorý skúma, či jednotlivé hardvérové zariadenia fungujú správne. Následne je riadenie predané MBR, ktoré lokalizuje umiestnenie bootloadera. Po sprevádzkovaní požadovaných systémových prostriedkov, bootloader načíta kernel do pamäte RAM[3]. Napriek tomu, že existujú viaceré bootloadre v rôznych Linuxových distribúciách, medzi najviac obľúbené patria GNU GRUB, GRUB 2, LILO a LOADLIN[4].

1.1.2 Kernel

Kernel (jadro) je hlavnou súčasťou operačného systému Linux a je zároveň základným rozhraním medzi hardvérom počítača a jeho procesmi. Aktívne komunikuje medzi dvoma vrstvami a riadi systémové zdroje čo najefektívnejšie[5]. Po načítaní kernelu do chránenej časti pamäte, mu BIOS predá riadenie nad systémom, kernel následne načíta ďalšie časti operačného systému na dokončenie spustenia systému a predá riadenie používateľom skrz pracovnú plochu, alebo iné rozhranie, pričom

ostáva naďalej načítaný v pamäti. V prípade, že je kernel poškodený, alebo nie je načítaný správne, systém taktiež nebude pracovať správne, alebo vôbec[6].

Hlavné časti Linuxového jadra sú[7]:

- **Medziprocesová komunikácia** - procesy medzi sebou komunikujú pomocou rúr, zdieľanej pamäte, zámkov, semaforov a správ.
- **Správa pamäte** - jadro používa rôzne metódy, ako zabrániť procesom zasahovať do pamäte iného procesu, ako napríklad stránkovanie, segmentácia, fyzické a virtuálne adresovanie.
- **Virtuálny súborový systém** (VFS) - tento subsystém je zodpovedný za poskytovanie jednotného rozhrania na prístup k uloženým údajom naprieč rôznymi súborovými systémami a fyzickými pamäťovými médiami.
- **Správa zariadení** - na vykonávanie činnosti, potrebujú procesy prístup k perifériám pripojeným k počítaču, ktoré sú riadené kernelom za pomoci ovládačov zariadení. Ovládače tvoria rozhranie medzi jadrom a hardvérom, informujú jadro o tom, ako komunikovať a ovládať konkrétny hardvér.
- **Sieťový subsystém** - uľahčuje procesom a iným subsystémom prístup k sieti, bez nutnosti poznania fyzického zariadenia, alebo protokolu.

Typy kernelu (jadra)[8][9]:

- **Monolitické jadro** - služby užívateľov a jadra bežia v rovnakom pamäťovom priestore, čím je umožnený efektívnejší beh procesov, avšak vzniká tu závislosť medzi jednotlivými systémovými komponentami. Operačný systém Linux využíva tento typ jadra.
- **Mikrojadro** - je novší prístup, ktorý zahŕňa iba základné služby a zariadenia potrebné pre beh systému. Služby užívateľa a jadra bežia v rôznych pamäťových priestoroch, čím je dosiahnutá menšia veľkosť jadra oproti monolitickému.
- **Hybridné jadro** - kombinácia monolitického a mikrojadra. Rýchlosť je porovnateľná s monolitickým jadrom, modularita a stabilita s mikrojadrom.

Linuxové jadro je modulárne, čo umožňuje dynamicky vkladať a odstraňovať kód pri behu jadra[10]. **Moduly jadra** sú časti kódu, ktoré môžu byť na želanie načítané do jadra, bez nutnosti rekompilácie jadra, alebo reštartovania počítača. Existuje viacero výhod používania modulov, ako napríklad, ľahšia správa, jednoduchšia možnosť úpravy systémových porúch, šetrenie pamäťového miesta. Ak napríklad chyba v niektorom ovládači zabraňuje správnemu spusteniu systému, môže byť tento ovládač odstránený do doby opravenia chyby a následne spätne načítaný. Moduly jadra sa nachádzajú v adresári `lib/modules/`, poprípade `/usr/lib/modules` a majú príponu `.ko` (kernel object)[11].

1.1.3 Démoni (*daemons*)

Daemon (démon) je typ programu, ktorý vykonáva svoju činnosť na pozadí a bez vedomia užívateľa. V podstate sa jedná o program, ktorý čaká na objavenia sa konkrétneho stavu, alebo udalosti, ktorá spustí daného démona. Bežný užívateľ nevie ovládať periodickosť vykonávania démonov. Procesy démona obvykle končia na písmeno **d**, toto umožňuje jednoduché rozlišovanie medzi procesmi systému a procesmi démona. UNIX-ové systémy obsahujú množstvo démonov, ako napríklad `sshd`, ktorý spravuje SSH pripojenia, alebo `named`, ktorý spravuje DNS server a mnoho ďalších. V Linuxových systémoch sa démoni spúšťajú pri štarte systému. V adresári `/etc/init.d` sa nachádza množstvo shell-ových skriptov, pomocou ktorých je možné démony buď spúšťať, alebo zastavovať[12][13].

1.1.4 Grafické rozhranie

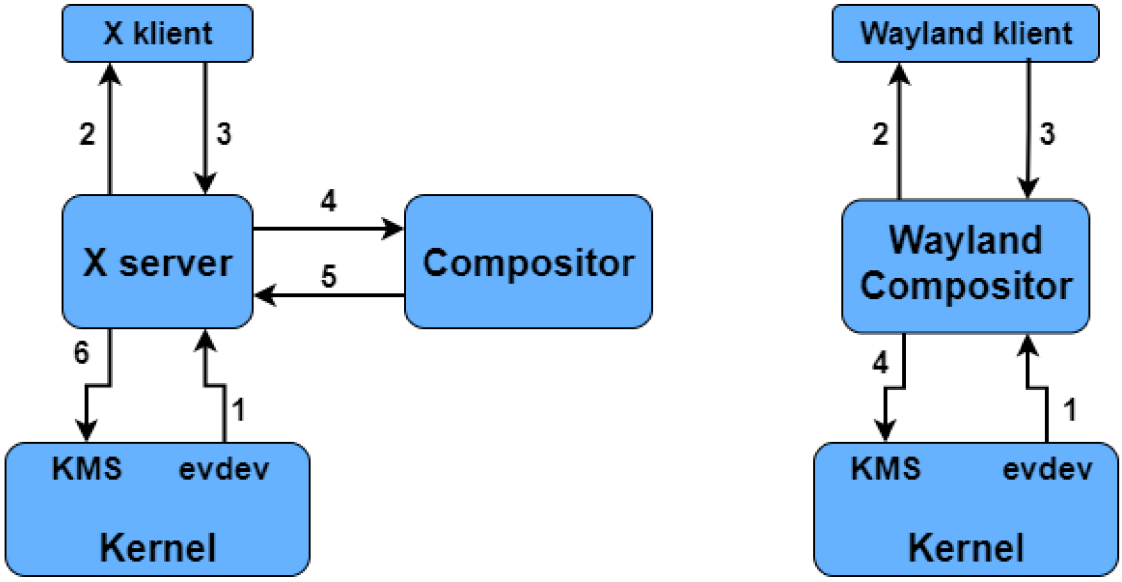
Grafické používateľské rozhranie (GUI) je formou používateľského rozhrania, ktoré umožňuje užívateľovi, za pomoci myši a klávesnice interagovať so systémom pomocou grafických prostriedkov namiesto príkazovej riadky[14]. Používateľské rozhranie a displej server sú pojmy, ktoré sa často zamieňajú, no nemali by. Displej server beží pod používateľským rozhraním, je to vlastne program, ktorého hlavnou úlohou je riadenie vstupu a výstupu svojich klientov z a do ostatných častí operačného systému a hardvéru. Na komunikáciu so svojimi klientmi využíva displej server svoj protokol. Displej server je základnou častou každého grafického používateľského rozhrania a sedí medzi jadrom systému a grafickým rozhraním.

Druhy displej servera[15]:

- **Xorg** - najviac využívaný displej server v Linuxových distribúciách. So svojimi klientskými aplikáciami komunikuje pomocou protokolu X11. Xorg je postavený na modely klient/server, čo znamená, že klienti môžu byť pripojený lokálne aj vzdialene.
- **Wayland** - komunitne riadený projekt, ktorého cieľom je nahradiť spomínaný Xorg s modernejším a bezpečnejším zobrazovacím systémom. Displej server Wayland-u sa nazýva `compositor` a aplikáciám poskytuje vyrovnávaciu pamäť pre každé okno, v tomto tkvie aj lepšie zabezpečenie oproti X11.

Desktopové prostredie v sebe zahŕňa množstvo komponentov ako ikony, panely s nástrojmi, tapety, ktoré sú základné piliere grafického používateľského rozhrania. Okrem toho obsahujú radu predinštalovaných aplikácií a služieb. Bez desktopového prostredia by boli užívatelia nútení pracovať s príkazovým riadkom a to zadávaním príkazov. Keďže sú desktopové prostredia jednoduché na prispôsobenie, užívateľovi je umožnené miešať aplikácie z rôznych desktopových prostredí, treba však

brať ohľad na veľkosť aplikácií a množstvo závislostí a predísť tak zahlteniu systému. Medzi najznámejšie desktopové prostredia patrí KDE, MATE, GNOME a Cinnamon[16].



Obr. 1.1: Architektúra Wayland vs. X server[17]

1.1.5 Aplikácie

Každé desktopové prostredie, ktoré prichádza s Linuxovou distribúciou, alebo ktoré sa užívateľ rozhodne nainštalovať, v sebe obsahuje množstvo vopred nainštalovaných aplikácií, avšak zďaleka nie všetky. Väčšina moderných Linuxových distribúcií prichádza s možnosťou si tieto aplikácie nainštalovať cez vstavaný obchod s aplikáciami, ktorý centralizuje a uľahčuje ich inštaláciu. Okrem inštalácie cez grafické prostredie je možné aplikácie inštalovať aj z príkazového riadka[18].

1.2 Distribúcie

Množina komponentov a procesy potrebné na inštaláciu týchto komponentov, ktoré sú potrebné pre dosiahnutie funkčného Linuxového systému sa nazýva **Linuxová distribúcia**. Moderné Linuxové distribúcie v sebe zahrňujú kernel (pre rôzne hardvérové architektúry môžu byť integrované rôzne verzie jadra), nástroje GNU (terminál), X server (displej server), desktopové prostredie (beží na X serveri pre poskytnutie grafického rozhrania) a ďalšie nástroje ako systém riadenia balíčkov[19].

Prakticky celý inštalačný proces linuxového systému, pri inštalácii konkrétnej distribúcie môže byť zautomatizovaný a prispôsobený užívateľovi. Výhodou Linuxových systémov je, že takmer všetky komponenty systému je možné zamieňať podľa potrieb užívateľa, alebo na dosiahnutie lepšieho behu systému.

1.2.1 Fedora

Fedora patrí medzi najznámejšie Linuxové distribúcie, časť tohoto úspechu pochádza od Red Hat Linux z ktorého vznikla. Napriek tomu, že obsahuje menej funkcií ako RHEL (Red Hat Enterprise Linux), patrí medzi najbezpečnejšie systémy a má bezplatné používanie. Fedora teda stojí na pomedzí dvoch cieľov, slúžiť ako komunitná verzia RHEL pre verejnosť a zároveň ako skúšobné prostredie nových metód a techník, ktoré môžu byť zahrnuté do Red Hat. Skupina komunitných vývojárov, zamestnancov Red Hat a dobrovoľníkov, ktorí sa zaoberajú inováciami, údržbou a podporou Fedory sa nazývajú Projekt Fedora. Z jednotlivých vydaní Fedory, ktoré sú už vylepšené a zabezpečené, vytvára Red Hat svoju vlastnú distribúciu Red Hat Enterpise Linux.

Fedora je podľa tvorcov distribúcia určená na bežné používanie, avšak má rôzne verzie pre rôzne využitie ako pre počítače, servery, IoT zariadenia, zamerané na beh v kontajnery, alebo cloudové prostredia. Má relatívne krátky šesť-mesačný cyklus vydávania aktualizácií pre zaistenie bezpečnosti a aktuálnosti systému[20].

Fedora podporuje dve hlavné architektúry od spoločnosti Intel a to 32-bitovú a 64-bitovú verziu, taktiež obsahuje špeciálne balíčky pre podporu ARM architektúry. Pre počítače má oficiálny zoznam podporovaných procesorov a pre servery má minimalizovanú variantu na zaistenie rýchleho chodu zariadenia. Čo sa týka hardvérovej podpory, Fedora obsahuje nástroje na používanie balíčkov `.rpm`, pre ktoré množstvo firiem vyvíja svoje ovládače a v prípade použitia neoficiálnych repozitárov sa naskytá možnosť využívania súkromných ovládačov pre zariadenia (sieťové, grafické karty). Z hľadiska bezpečnosti sa jedná o jeden z najbezpečnejších distribúcií, pretože SELinux, firewall a iné bezpečnostné funkcie sú povolené od prvotnej inštalácie[21].

1.2.2 Debian

Debian patrí medzi najznámejšie a najstaršie Linuxové distribúcie. Mnohí užívatelia považujú Debian za názornú ukážku komunitne riadenej distribúcie. V okruhu mnoho vývojárov a systémových správcov, patrí medzi veľmi obľúbenú distribúciu a to z dôvodu jej filozofie a spôsobu podpory. Debian sa považuje za predchodcu skoro polovice súčasných distribúcií, ako napríklad Ubuntu, Kali Linux, či Linux Mint.

Debian je určený na bežné používanie, avšak existujú verzie aj pre iné účely, ako serverová a IoT. Pre bežné vydania poskytuje aktualizácie po dobu približne jedného roku a pre vydanie LTS je podpora až po dobu piatich rokov. Debian si prisvojilo systémové vydania LTS od Ubuntu, kde každá stabilná verzia s podporou LTS dostáva hlavne bezpečnostné záplaty po dobu až päť rokov.

Hardvérová podpora Debianu leží na pomedzí Ubuntu a Fedory, napriek tomu, že Debian podporuje inštaláciu len voľného softvéru, je tu možnosť sťahovania aj proprietárneho softvéru z ich repozitárov.

Najviac hardvérových architektúr je podporovaných práve na Debiane. Oficiálne sú to, 32 a 64-bitové architektúry od Intel a AMD, ARM, MIPS, PowerPC, SPARC, S390. Čo sa týka neoficiálnej podpory hardvérových architektúr, sú taktiež dostupné na stiahnutie, väčšina z nich však nie je naďalej vyvíjaná. Z bezpečnostného hľadiska poskytuje prvotná inštalácia Debianu len základnú úroveň zabezpečenia a nejedná sa o tak zabezpečený systém ako Ubuntu, alebo Fedora. Samozrejme tu je možnosť zabezpečiť si Debian svojpomocne, alebo pomocou návodov[21].

1.2.3 Ubuntu

Ubuntu je najpoužívanejšia Linuxová distribúcia a kvôli jej jednoduchosti použitia je najviac užívateľsky prívetivá pre začiatočníkov. Ubuntu vzniklo ako presvedčenie o tom, že aj Linuxová distribúcia sa môže stať každodenným operačným systémom a konkurovať tak Windowsu a OS X. Ako základ pre distribúciu bol zvolený zdrojový kód Debianu, ktorý mal v tej dobe veľkú popularitu, avšak bol príliš zložitý na inštaláciu. Vývojári spoločnosti Canonical Ltd., ktorí Ubuntu vytvorili, sa snažili čo najviac zjednodušiť proces inštalácie, ako aj samotnú obsluhu OS.

Hlavné využitie Ubuntu je na bežné používanie, v poslednej dobe spoločnosť Canonical pracuje na tom, aby sa distribúcia dala nainštalovať na rôzne typy hardvéru. V súčasnosti má Ubuntu rôzne prevedenia pre rôzne prostredia, ako desktopová, serverová distribúcia, cloudové riešenie, IoT a mobilné zariadenia. Svojim užívateľom poskytuje deväť-mesačný cyklus vydávania aktualizácií pre bežné vydania a LTS (Long Term Suport) vydania sú aktualizované až po dobu piatich rokov.

Hardvérová podpora Ubuntu patrí k najlepším, bežné hardvérové prvky sú detegované automaticky. Kvôli popularite Ubuntu sa aj v prípade, že systém nerozpozná použitý hardvérový prvok, jednoduché nájsť alternatívny zdroj pre inštaláciu ovládača.

Pre Ubuntu je bežná podpora 32-bitových a 64-bitových architektúr od spoločností ako Intel a AMD. ARM a IBM Power8 sú oficiálne podporované na niektorých serverových vydaniach, zároveň je možné používať Ubuntu aj pre iné architektúry,

avšak je nutné použiť staršie, alebo neoficiálne vydania[21]. Ubuntu má už od prvotnej inštalácie povolené viaceré bezpečnostné mechanizmy. AppArmor je bezpečnostný modul, ktorý dokáže uzamknúť zraniteľné procesy, čím obmedzí škody, ktoré môžu byť spôsobené v prípade zneužitia týchto zraniteľností[22].

PPA (Pesonal Package Archive) je softvérový repozitár, pomocou ktorého vieme získať neštandardné aplikácie, alebo aktualizácie. Umožňuje vývojárom distribuovať svoj softvér a to bez nutnosti čakania na pridanie do oficiálnych repozitárov Ubuntu. Namiesto toho si vývojári vytvoria vlastný repozitár, do ktorého je možné nahrať zdrojové balíčky a požiadajú užívateľa, alebo zákazníka, aby si tento repozitár pridal medzi zdroje. Týmto spôsobom je užívateľovi zaručené, že dostane vždy aktuálnu verziu softvéru aj na staršom vydaní distribúcie.

Typy repozitárov v Ubuntu[23]:

- **Main** - softvér podporovaný spoločnosťou Canonical
- **Universe** - softvér, ktorý je udržiavaný komunitne
- **Restricted** - proprietárne ovládače
- **Multiverse** - softvér obmedzený autorskými právami

Súbor `/etc/apt/sources.list` je najdôležitejším faktorom pri pridávaní, alebo aktualizovaní aplikácií v Ubuntu. V tomto súbore systém uchováva informácie o všetkých repozitároch prítomných v systéme. Pri pridaní PPA repozitára sa neprepisuje súbor `/etc/apt/sources.list`, namiesto toho systém vytvorí dva súbory v adresári `/etc/apt/sources.list.d`, jedno s koncovkou `.list` a jedno s `.save`[24].

```
1 #pridanie PPA repozitára do zoznamu
2 sudo add-apt-repository <PPA_repo>
3
4 #aktualizácia dostupných balíčkov
5 sudo apt-get update
6
7 #inštalácia balíčka z PPA
8 sudo apt-get install <balicek_z_PPA>
```

Výpis 1.1: Príklad pridávania a inštalácie z PPA repozitára

2 Jednodoskové počítače

Jednodoskový počítač (SBC) je počítač, ktorý sa nachádza na jednej doske plošných spojov. Rozdiel medzi štandardným desktopovým počítačom je, že SBC majú všetky periférne zariadenia zabudované na jednej doske. Perifémymi zariadeniami sa myslia USB, sériové porty, Ethernetový port, audio/video výstup a ďalšie. Pridávanie ďalších vstupno-výstupných zariadení je možné pomocou rozširujúcich slotov, ktoré väčšina moderných jednodoskových počítačov obsahuje hneď viacero.

Oproti štandardnému počítaču, ktorý sa skladá z viacerých dosiek a komponentov, má SBC oveľa efektívnejší a kompaktnejší dizajn, keďže obsahuje len jednu dosku plošných spojov a do tejto dosky sú zabudované všetky komponenty. Medzi hlavné výhody SBC patrí nízka spotreba elektrickej energie, pretože obsahuje len komponenty potrebné na základnú prevádzku. Z rozmerov a výkonu jednodoskových počítačov tiež vypláva, že vytvárajú menej tepla ako štandardné počítače, týmto sa naskytá možnosť využitia v prostrediach, kde by mohla predstavovať problém teplota. Napokon, sú cenovo dostupnejšie než štandardné počítače a sú teda vhodnejšie do prostredí, v ktorých nie sú potrebné super-výkonné zariadenia[25].

Použitie jednodoskových počítačov[26]:

- **Domáce servery** - na dostatočne výkonnom SBC je možné spustiť vlastný server, napríklad emailový, webový, alebo multimediálny.
- **Desktopové použitie** - na základné použitie, ako je prehliadanie internetu, úprava a prezeranie dokumentov, jednoduchá práca s fotkami. Pre tieto využitia je možné nahradiť stolové počítače za jednodoskové.
- **IoT** - existuje veľké množstvo aplikácií pre vytvorenie vlastnej inteligentnej domácnosti, ako OpenHAB, alebo Home Assistant. Pomocou týchto aplikácií nainštalovaných na SBC je možné riadiť inteligentné zariadenia v domácnosti.
- **Vzdelávanie** - ako pomôcka pri výučbe odborných predmetov zameraných napríklad na automatizáciu a robotizáciu.

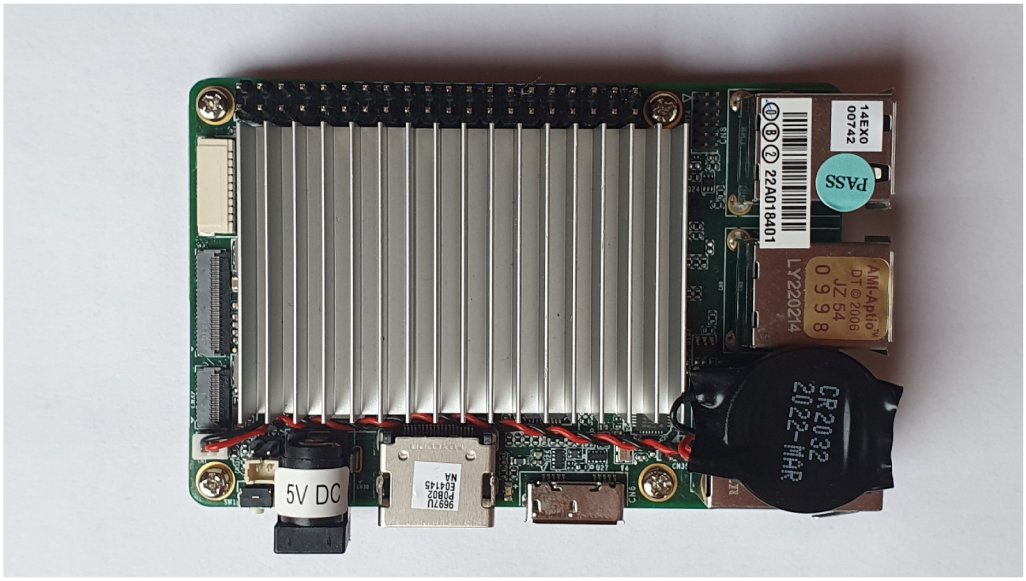
2.1 UP Board

UP Board je jednodoskový počítač o veľkosti kreditnej karty s nízkou spotrebou a vysokým výkonom. Je poháňaný štvorjadrovým procesorom Atom x5-Z8350 od spoločnosti Intel. Tento procesor beží na takte až 1.92GHz a má 64-bitovú architektúru. Grafický procesor ôsmej generácie Intel HD 400, si poradí aj s náročnejším 3D obsahom. Rozloženie každej dosky je totožné, zmeny sa prejavujú len pri ponuke pamätí RAM a eMMC. UP Board je možné osadiť 1GB/2GB/4GB DDR3L pamäťami RAM s frekvenciou 1600MHz a 16GB/32GB/64GB internej pamäte eMMC. UP Board obsahuje aj klasické periférie, konkrétne štyri USB 2.0, USB 3.0 OTG (On

The Go), ethernetový port podporujúci rýchlosť 1Gbps, HDMI port, DSI, CSI a ďalšie. Má 40-pinovú zbernicu GPIO (General Purpose Input Output), ktorá umožňuje pridávanie vlastných rozšírení. Napájanie je zabezpečené skrz 5.5mm jack konektor, pričom UP Board vyžaduje na prevádzku minimálne 5V a 3A, avšak odporučaný výkon zdroja je 5V a 4A. Procesor Intel Atom má tendenciu prehrievania pri stredne náročnom až náročnom používaní, preto je doska osadená hliníkovými chladičmi na hornej aj dolnej strane, ktoré zabezpečujú kvalitnú ventiláciu a sú nevyhnutné pre dlhodobé používanie. Prevádzková teplota, ktorá sa vďaka chladeniu pohybuje na úrovni od $0 - 60°C$ a prevádzková vlhkosť $10\% - 80\%$, predurčujú široké možnosti použitia UP Board vo viacerých prostrediach.

Čo sa týka bezpečnosti, UP Board disponuje bezpečnostnými funkciami Intel, ako Intel AES New Instructions, alebo technológia ochrany identity (IPT). Procesor tiež podporuje systém Android vo verzii 6.0, Microsoft Windows 10 v plnej verzii a rôzne Linuxové distribúcie. Séria UP Board sa vyrába pre rozličné domény a produkty, ako sú drony, inteligentné domácnosti, vzdelávanie, internet vecí (IoT), robotika, strojové videnie[27].

Praktická časť práce sa bude zameriavať na model UP Board vo verzií s 2GB RAM, 32GB internej pamäte a hliníkovým chladičom s rozmermi ($55 \times 35 \times 10$ mm).



Obr. 2.1: UP Board

Rozhodujúcim faktorom, pri výbere UP Board bola tiež veľká užívateľská základňa, technická podpora a online komunita, pozostávajúca z užívateľov a vývojárov, ktorí sa proaktívne podieľajú na zlepšení služieb UP a ich uvedenie do života.

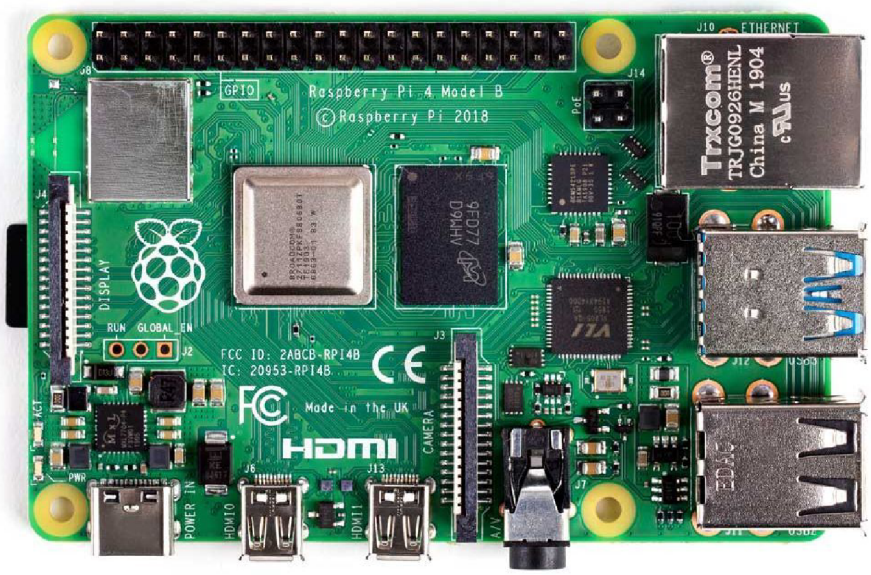
UP Board má tiež oficiálnu podporu pre vybrané verzie Ubuntu, čo z neho robí jednoznačnú voľbu v zamýšľanom použití.

2.2 Ďalšie jednodoskové počítače

Pri skúmaní najlepšieho jednodoskového počítača, ktorý by slúžil ako server na riadenie inteligentných budov, bolo nutné zvážiť viaceré faktory. Hlavným cieľom bolo vybrať SBC, ktoré by bolo dostatočné výkonné na spojazdnenie virtuálizácie a obsahovalo dostatočné množstvo vnútornej pamäte pre potreby inštalácie aplikácií. Ďalším dôležitým faktorom bola veľkosť operačnej pamäte, na vyhovenie požiadavok operačného systému a aplikácií. V prípade nedostatočnej veľkosti operačnej pamäte, by systém náhodne mrzol, alebo sa reštartoval, čo je pre potreby „servera“ neprijateľné. Možnosti rozšírenia do budúcnosti, boli taktiež prioritizované pri procese výberu jednodoskového počítača. V našom prípade sa jednalo hlavne o ethernetový port s rýchlosťou až 1Gbps, pre pripojenie inteligentných zariadení a o 40-pinovú zbernicu na pridávanie vlastných rozšírení. Rovnakú mieru dôležitosti má aj kompatibilita základnej architektúry a vybraného operačného systému. V našom prípade sa porovnávali tri Linuxové distribúcie, z ktorých bolo nakoniec vybrané Ubuntu pre jeho najvyššie kvality ako serverová distribúcia. Všetky vyššie zmienené faktory a ďalšie podstatné údaje sú vyobrazené v porovnávacej tabuľke 2.1.

2.2.1 Raspberry Pi 4

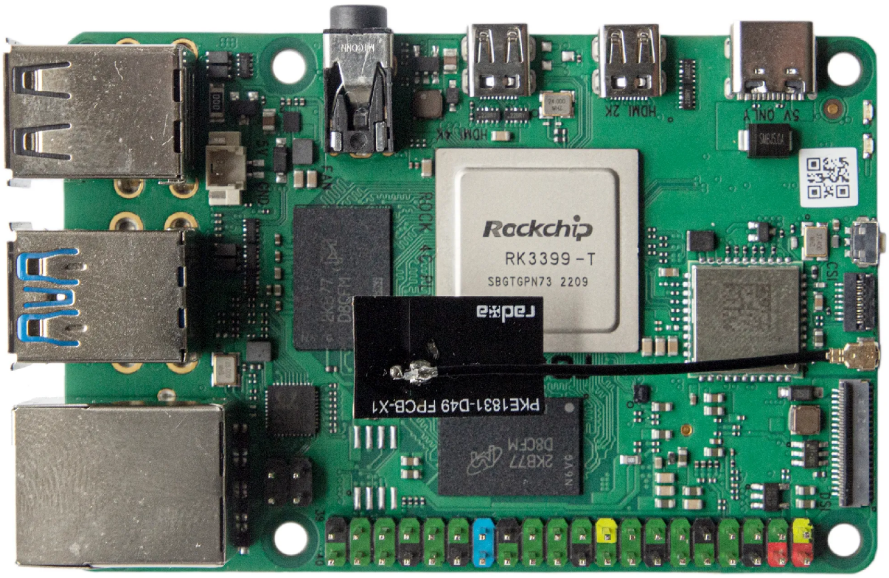
Raspberry Pi 4 je najnovšia varianta rady jednodoskových počítačov Raspberry Pi, od spoločnosti Raspberry Pi Foundation. Je poháňaný štvorjadrovým procesorom Cortext-A72 od spoločnosti Broadcom, ktorý dosahuje výkon 1.5GHz a je založený na architektúre ARM. Grafický procesor podporuje 4K video a dokáže súčasne utiahnuť až dva 4K monitory na 60 snímkach za sekundu. Raspberry obvykle neponúka vnútorné úložisko vo svojich zariadeniach, avšak je tu podpora pre mikro SD karty (úložisko dát, OS). V závislosti od modelu, sú k dispozícii rôzne varianty s operačnou pamäťou RAM a to 1GB/2GB/4GB/8GB LPDRR4. V základe Raspberry Pi 4 neobsahuje žiadny chladič. Prevádzková teplota dosahuje papierovú úroveň $0 - 50°C$, no v skutočnosti, pri väčšom zaťažení, môže dosahovať o niekoľko desiatok stupňov viac. Vo všeobecnosti sa nedoporučuje umiestňovať Raspberry Pi do vlhších prostredí, čo je nevýhoda oproti UP Board. V pomere ceny a výkonu nemá Raspberry Pi 4 takmer žiadnu konkurenciu, počiatočná cena pre základný model sa pohybuje na úrovni 40€, čo z neho robí ideálnu voľbu pre nadšencov[28].



Obr. 2.2: Raspberry Pi 4 Model B [29]

2.2.2 Rock Pi 4C

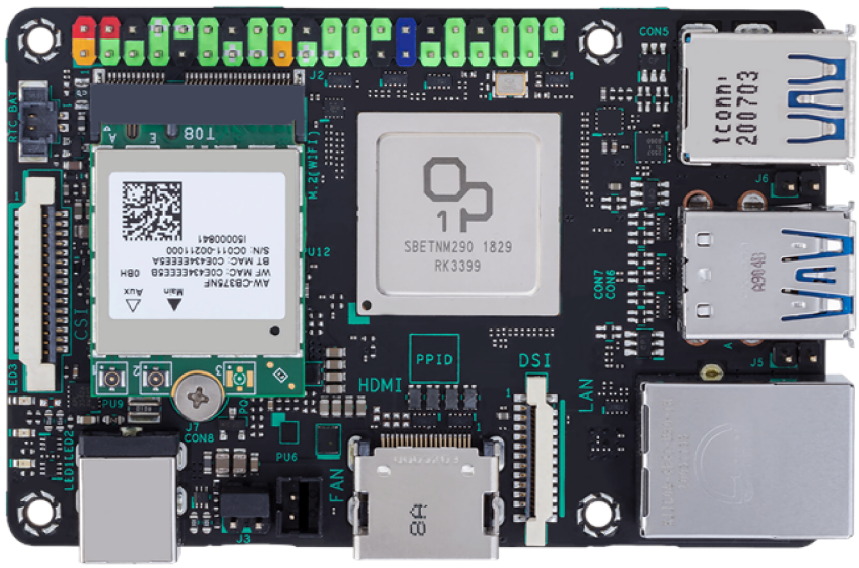
Jednodoskový počítač **Rock Pi 4C** je vysokovýkonná a multifunkčná doska, založená na výkonnom SoC RK3399-T od spoločnosti Rockchip. Rock Pi je vyvíjaný spoločnosťou OKdo v spolupráci s Radxa a predstavuje dokonalú alternatívu k Raspberry Pi. Je poháňaný štvorjadrovým procesorom Cortex-A53, v spolupráci s dvojjadrovým procesorom Cortex-A72 s taktom 1.8GHz. Architektúra ARM, na ktorej je aj tento procesor postavený, je bežná v segmente SBC. Vnútorné úložisko v Rock Pi 4 nenájdeme, avšak ponúka rozšírenie o mikro SD kartu s kapacitou až 128GB, alebo rozšírenie o M.2 SSD. Veľkosť operačnej pamäte, závisí od konkrétneho modelu, všeobecne sa však jedná o 64bitové dual-channel pamäte LPDDR4 s kapacitou 1GB/2GB/4GB. Prevádzková teplota Rock Pi je pri bežnej prevádzke v rozmedzí $0 - 50°C$, ale pri vyššom zaťažení sa môže vyšplhať na 80°C. Pre dlhodobú funkčnosť sa uvádza, že teplota CPU by nikdy nemala presiahnuť úroveň 85°C a je odporúčaná inštalácia externého chladenia[30]. Podobne ako pri Raspberry Pi, je tu taktiež podpora 4K rozlíšenia so 60 snímkami za sekundu. Obsahuje tiež rôzne kryptografické rozšírenia, ktoré prichádzajú s ARMv8 a technológiu zabezpečenia TrustZone. Rock Pi predstavuje výkonnejšiu variantu Raspberry, s natívnou podporou AI s akceleráciou GPU je skvelý pre aplikácie počítačového videnia, robotiky, výpočtov a ďalších. Nevýhodou oproti UP Board je napríklad menšia užívateľská základňa a horšia podpora[31].



Obr. 2.3: Rock Pi 4 C [31]

2.2.3 Asus Tinker Board 2

Tinker Board 2 je jednodoskový počítač vyrábaný spoločnosťou Asus. S pomedzi vyššie spomínaných jednodoskových počítačov je Tinker Board 2 najvýkonnejší. Je poháňaný štvorjadrovým procesorom Cortex-A53 s taktom 1.5GHz, v spolupráci s dvojjadrovým procesorom Cortex-A72 s taktom 2.0GHz od spoločnosti Rockchip. 64-bitová architektúra ARMv8 mu zaručuje rovnaké výhody ako pri Rock Pi 4, avšak s vyšším taktom procesora sa umiestňuje výkonovo o úroveň vyššie. Využíva tiež heterogénnu technológiu spracovania big.LITTLE, ktorá spája dva odlišné procesory v jednom SoC. Výsledkom je automatické prideľovanie úloh a zabezpečenie jadier CPU pre každý proces. Operačné pamäte sú k dispozícii v dvoch variantách 2GB/4GB. Prevádzková teplota je rovnaká, ako UP Board, $0 - 60°C$, ale pre svoj výkonný procesor trpí častým prehrievaním, aj napriek hliníkovému chladeniu. Interné úložisko eMMC má malú kapacitu, len 16GB, avšak rozšírenie je možné za pomoci mikro SD karty. Tinker Board je tiež vybavený ochranou ASUS ESD Guard, ktorá predlžuje životnosť komponentov tým, že zabraňuje poškodeniu elektrostatickým výbojom. Vďaka natívnej podpore Androidu 10 a 11, môže Tinker Board 2 ponúknuť zaujímavé funkcie, ktoré nie sú bežne dostupné na konkurenčných jednodoskových počítačoch[32][33].



Obr. 2.4: Asus Tinker Board 2 [33]

	UP Board	Raspberry Pi 4	Rock Pi 4C	Tinker Board 2
SoC Výrobca	Intel	Broadcom	Rockchip	Rockchip
SoC Čip	Atom x5 Z8350	BCM2711	RK3399	RK3399
CPU Jadrá	4× Cherry Trail 1.92GHz	4× Cortex-A72 1.5GHz	2× Cortex-A72 1.8GHz, 4× Cortex-A53 1.4GHz	2× Cortex-A72 2.0GHz, 4× Cortex-A53 1.5GHz
GPU	Intel HD 400	Video Core VI	Mali T860MP4	Mali T860MP4
Pamäť RAM	1GB - 4GB DDR3L	1GB - 8GB LPDDR4	1GB - 4GB LPDDR4	2GB - 4GB LPDDR4
RAM Frekvencia	1600MHz	3200MHz	3200MHz	3200MHz
eMMC/flash	16GB - 64GB eMMC	-	-	16GB eMMC
Iné úložisko	-	Mikro SD	Mikro SD (až 128GB)	Mikro SD
WLAN	-	2.4GHz/5GHz (802.11ac)	2.4GHz/5GHz (802.11ac)	2.4GHz/5GHz (802.11ac)
LAN	1× GbE (RJ-45)	1× GbE (RJ-45)	1× GbE (RJ-45)	1× GbE (RJ-45)
USB 2.0	4×	2×	2×	-
USB 3.0	1× OTG	2×	1+1 OTG	3+1 OTG
Bluetooth	-	Bluetooth 5.0	Bluetooth 5.0	Bluetooth 5.0
Display	1× HDMI 1.4b, 1× DSI/eDP	2× µ-HDMI 2.0, 1× MIPI-DSI	1× HDMI 2.0, 1× MIPI-DSI	1× HDMI 2.0, 1× MIPI-DSI
Audio	1× I2S	1× 3.5mm jack	1× 3.5mm jack	1× I2S, 1× S/PDIF
Kamera	1× MIPI-CSI	1× MIPI-CSI	1× MIPI-CSI	1× MIPI-CSI
RTC	áno	nie	áno	áno
Rozšírenie	Pi 40	Pi 40	Pi 40	Pi 40
Podpora OS	Windows 10, ubilinux, Ubuntu, Yocto, Android 6.0	Raspberry Pi OS, Ubuntu, Manjaro, Gentoo, Kali, Fedora, PopOS	Ubuntu, Debian, Android 10	Debian, Android 10/11
Napájanie	5V/4A jack	5V/4A USB-C	9V/2A USB-C	5.5V/2.5A jack
Chladenie	pasívne	-	-	-
Rozmery	86.5×56.5 mm	88×58 mm	85×54 mm	85×56 mm
Cena	187 €	91 €	134 €	143 €

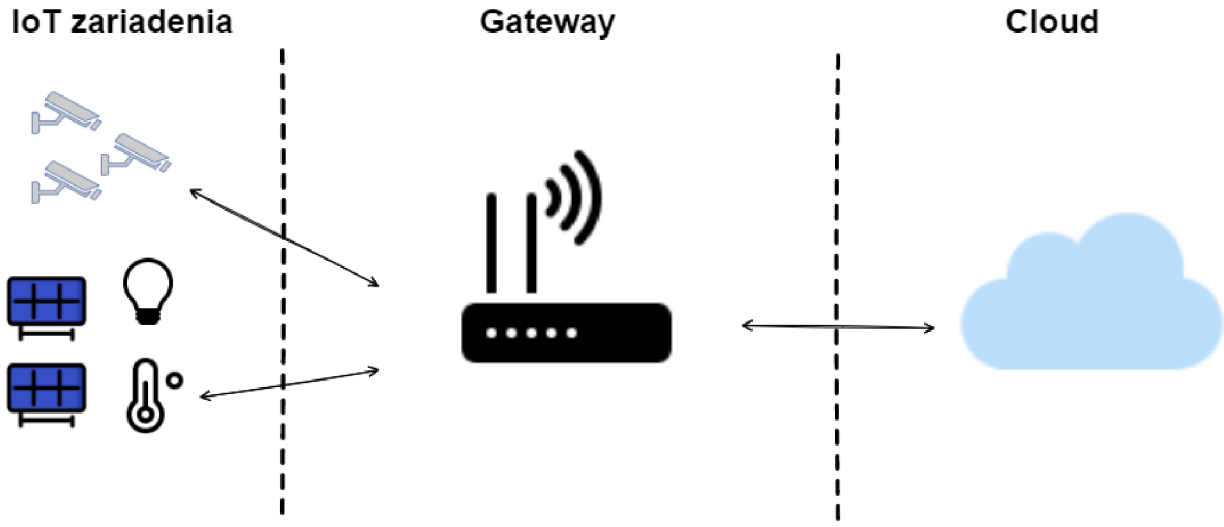
Tab. 2.1: Porovnanie UP Board s konkurenčnými SBC

3 Gateway

Komunikačná brána - gateway je fyzické zariadenie, alebo program, ktorý umožňuje komunikáciu, najčastejšie medzi inteligentnými zariadeniami a cloudom, prekladá komunikáciu a filtruje dáta na užitočné informácie. Dáta, ktoré sa medzi danými prostriedkami prenášajú, sú smerované cez komunikačnú bránu, ktorá sa správa ako smerovač a riadi dáta k ich destinácii. Moderné komunikačné brány vedia spracovať obojsmernú komunikáciu, čo znamená, že odchádzajúce toky správ o zariadeniach IoT sú odosielané do cloudu a prichádzajúce správy majú zväčša spravovací charakter (aktualizácie, správa zariadení).

Zariadenia IoT môžu zhromažďovať veľké množstvo údajov, ktoré si vyžadujú značné množstvo šírky pásma na odoslanie do cloudu. Mnohé komunikačné brány tak okrem smerovania aj predspracúvajú dáta. Výhodou tejto techniky je určitá forma agregácie a sumarizácie dát, čím sa znižuje množstvo dát, ktoré je nutné poslať ďalej, a tým sa znižuje spotreba šírky pásma, ako aj čas potrebný na prenos[34].

Z bezpečnostného hľadiska pridávajú komunikačné brány dodatočnú vrstvu ochrany siete a zariadení. Chránia dáta pri presune z a do zariadení a siete, zabezpečujú, že sa prenášajú len autorizované dáta. Existujú rôzne hrozby, ktorým sú komunikačné brány vystavené, ako manipulácia s hardvérom, DoS útoky, elevácia oprávnení, MITM útoky a spoofing. Vzhľadom na tieto hrozby nastáva nutnosť implementácie viacerých techník ochrany tejto brány. Najčastejšie sa používajú fiwerally, zabezpečený prístup, šifrovanie prenášaných a uložených dát, vzájomná autentifikácia brány a zariadení[35].



Obr. 3.1: Základný princíp komunikačnej brány

3.1 OpenHAB

OpenHAB - Open Home Automation Bus je projekt, ktorého cieľom je vytvoriť jednotné prostredie na riadenie inteligentnej domácnosti. Projekt založil v roku 2010 pán Kai Kreuzer, v súčasnosti projekt spadá pod neziskovú organizáciu OpenHAB Foundation.

OpenHAB je open source aplikácia, napísaná v jazyku Java, čo znamená, že je závislá len od virtuálneho stroja Java (JVM), ktorý je dostupný pre väčšinu platforiem. V prípade použitia na SBC ponúka OpenHAB svoj vlastný Linuxový klon OpenHABian, čo je kompletný obraz disku. Okrem tejto možnosti má aj inštalačný skript pre Linuxové distribúcie (založené na Debiane), macOS, Windows, Synology DiskStation, či Docker kontajner. Po inštalácii a spustení OpenHAB je používateľské rozhranie dostupné na `localhost:8080`. Pri prvotnom prihlásení je užívateľ vyzvaný k vytvoreniu administrátorského účtu.

OpenHAB používa nové unifikované rozhranie s názvom MainUI, pomocou ktorého je možné nastavovať všetky položky v OpenHAB. MainUI taktiež ponúka používateľom rôzne pohľady v závislosti od typu užívateľa. Bežný užívateľ, vidí všetky interaktívne časti používateľského rozhrania a tiež môže otvárať ďalšie aplikácie, avšak nevidí žiaden z administračných rozhraní a taktiež nemá prístup k administračným funkciám cez REST API. Administrátor, ako bolo zmienené vyššie, tento užívateľ musí byť vytvorený ako prvý. Administrátor má plnú správcovskú moc a zobrazujú sa mu možnosti, ako Nastavenia, Nástroje pre vývojárov, alebo Bočný panel pre vývojárov.

Interná štruktúra systému môže zo začiatku pôsobiť chaoticky a zložitejšie, ale vďaka jej prepracovanosti je prehľadná aj pri náročnejšej konfigurácii. Jedným zo základných pojmov je `thing` (vec). Veci sú objekty, ktoré môžu byť fyzicky pridané do systému a môžu poskytovať viacero funkcií (merajú teplotu a zároveň vlhkosť vzduchu). Zdôraznil by som, že veci nemusia byť fyzické zariadenia, ale môže sa jednať aj o webovú službu. Veci poskytujú `channels` (kanály), sú to vlastne funkcie, ktoré veci poskytujú. Napríklad, žiarovka môže mať kanál teploty farby, alebo webová služba kanál s teplotou, vlhkosťou a tlakom vzduchu. Na integráciu koncových prvkov sa využívajú `bindings` (väzby). Sú to zásuvné moduly, ktoré poskytujú spôsob prepojenia `items` (položiek) s fyzickými zariadeniami. Taktiež zovšeobecňujú špecifické komunikačné požiadavky zaradenia, aby sa s ním dalo pracovať jednoduchšie. Položky reprezentujú vstup (vlhkosť vzduchu), majú svoj stav a môžu dostávať príkazy. Prepojenie medzi vecami a položkami sa nazýva `link` (odkaz). Odkaz mapuje kanály na jednotlivé položky. Ak je položka prepojená s kanálom, znamená to, že schopnosť, ktorú položka poskytuje je dostupná cez konkrétny kanál.

Automatizácia procesov je možná pomocou vytvorenia pravidiel, ktoré sú definované užívateľom. Každé pravidlo obsahuje spúšťač, ktorý sa za určitých podmienok aktivuje a vyvolá skript na vykonanie preddefinovanej úlohy (vypnutie svetla). Bežne sa skripty a pravidlá píšu v jazyku Xbase, avšak je tu možnosť pridania doplnku Blockly. Blockly je vizuálny programovací jazyk, ktorý pozostáva z blokov, ktoré do seba zapadajú a môže byť dobrou voľbou pre začiatočníkov[36].

3.2 Home Assistant

Home Assistant je bezplatná a open source aplikácia zameraná na domácu automatizáciu s dôrazom na lokálne riadenie a súkromie. Prvá verzia Home Assistant vznikla ako voľne dostupná Python aplikácia, ktorú vytvoril pán Paulus Schoutsen a bola zverejnená na platforme GitHub v roku 2013.

Podporovaných platforiem na inštaláciu Home Assistant je naozaj veľa, ako, Windows, macOS, rôzne distribúcie Linuxu, SBC a NAS stanice.
Home Assistant ponúka viaceré možnosti inštalácie, medzi odporúčané patria[37]:

- **Home Assistant Operačný systém** - minimalizovaný OS, obsahujúci správcu doplnkov a Home Assistant jadra.
- **Home Assistant Kontajner** - samostatná inštalácia Home Assistant jadra do Docker kontajnera, bez doplnkov.
- **Home Assistant S dozorcom** - plnohodnotná inštalácia správcu doplnkov a jadra.
- **Home Assistant Jadro** - inštalácia pomocou Python prostredia, bez doplnkov.

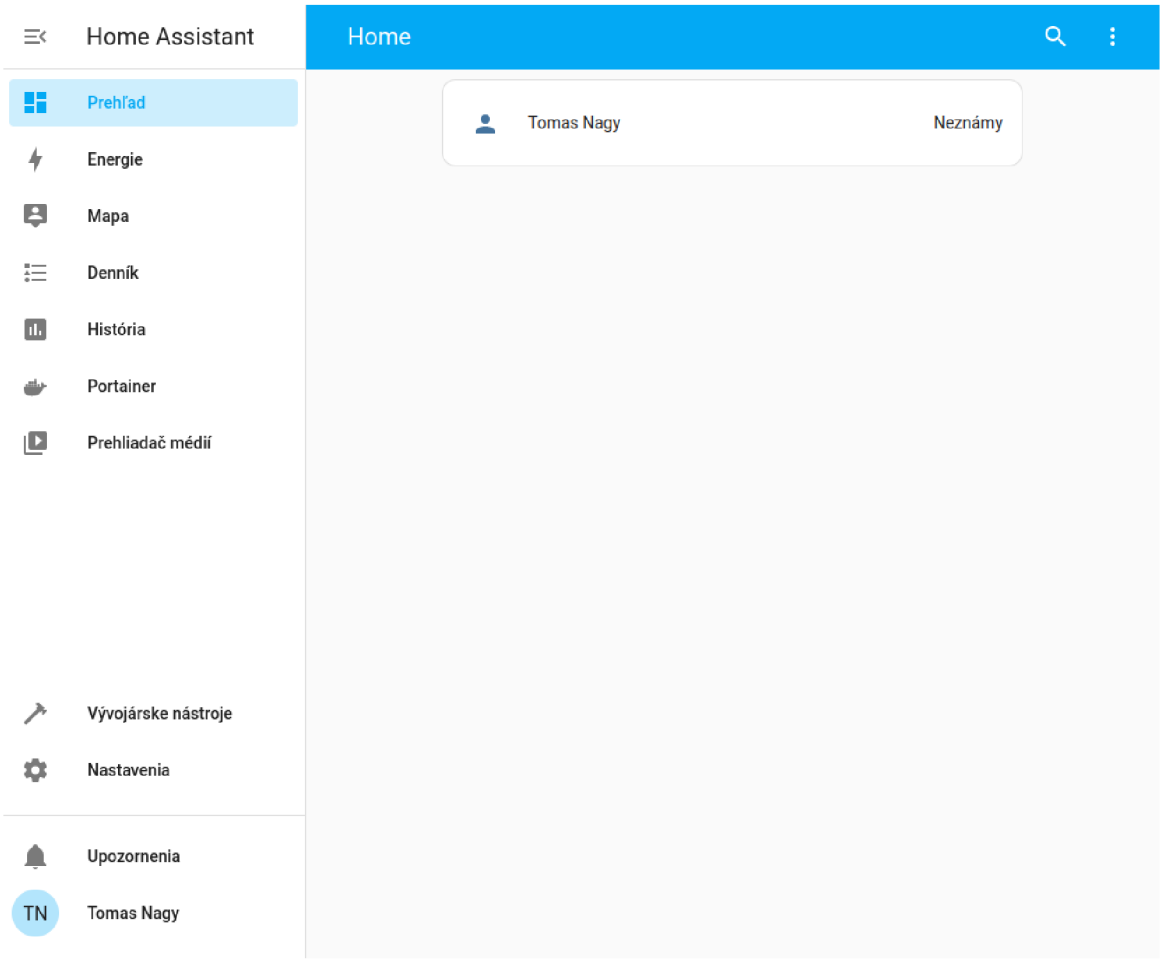
Po inštalácii a spustení Home Assistant je používateľské rozhranie dostupné na `localhost:8123`. Pri prvotnom prihlásení je užívateľ vyzvaný k vytvoreniu administrátorského konta. Pomocou tohto konta je možné spravovať všetky časti aplikácie, ako aj samotné zariadenia.

Interná štruktúra, podobne ako pri OpenHAB pozostáva z viacerých častí.
`Device` (zariadenie) je skupina entít, ktoré značia tú istú fyzickú, alebo logickú jednotku. `Entity` (entita) predstavuje jeden ovládací prvok zariadenia. Jedno zariadenie môže tým pádom poskytovať viacero entít na sledovanie funkcií zariadenia. Entity môžu mať rôzne `states` (stavy), ktoré sú závislé od typu entity. `Events` (udalosti) reprezentujú zmeny stavov entít. Výstup udalostí sa nazýva `service` (služba) a jedná sa o službu, ktorú entita poskytne pri zmene stavu. `Integrations` (integrácie) spájajú Home Assistant so zariadeniami a ich službami. Integrácia sa stará o prepojenie zariadení, alebo služieb, ktoré obnášajú špecifické požiadavky (špeciálne protokoly), do Home Assistant[38].

Home Assistant ponúka aj možnosť automatizácie, ktorá je riešená dvoma spôsobmi:

- Úpravou textového súboru `configuration.yaml`, je tu predpoklad znalosti jazyka YAML.
- Pomocou používateľského rozhrania, ktoré umožňuje vytvárať skripty v GUI.

Hlavné rozhranie pre Home Assistant sa nazýva Lovelace a slúži na ovládanie celého systému. Úprava používateľského rozhrania je možná graficky, alebo úpravou konfiguračného súboru[39].



Obr. 3.2: Webové rozhranie Lovelace v Home Assistant

3.3 Protokol MQTT

MQTT - Message Queuing Telemetry Transport je protokol prenosu správ, založený na mechanizme publish/subscribe. Je odľahčený, jednoduchý a navrhnutý tak, aby vyhovoval obmedzeniam CPU a šírke pásma. Tieto vlastnosti ho robia ideálnym pre použitie v stavanom prostredí, kde by poskytoval spoľahlivú a efektívnu metódu komunikácie. Je taktiež výbornou voľbou pre bezdrôtové a nespoľahlivé siete. Najčastejšie sa MQTT využíva v IoT zariadeniach a pri komunikácii stroj-stroj (M2M), kde je požadovaná nízka réžia pri prenose a odľahčená štruktúra, vzhľadom na pamäťové nároky[40].

Oproti modelu klient/server, kde klient komunikuje priamo so serverom, mechanizmus publish/subscribe oddeľuje klienta `publisher` (poskytovateľ správ), ktorý posiela správu, od klientov, ktorý dané správy prijímajú `subscribers` (odberatelia správ). Poskytovatelia a odberatelia sa nikdy priamo nekontaktujú, o svojej vzájomnej existencii ani nevedia. O spojenie medzi nimi sa stará `broker` (sprostredkovateľ), ktorého úlohou je distribúcia a prerozdeľovanie správ. Prerozdeľovaním správ je možné dosiahnuť určitú formu filtrácie, pri ktorej je možné riadiť a sledovať, ktorý odberateľ prijal danú správu. V prípade straty spojenia medzi sprostredkovateľom a odberateľom, si sprostredkovateľ uloží správy do vyrovnávacej pamäte a odošle ich, keď bude odberateľ opäť dostupný. MQTT je dátovo-agnostický, čo znamená, že správy môžu byť posielané ako textové reťazce, binárne dáta, alebo JSON, či XML.

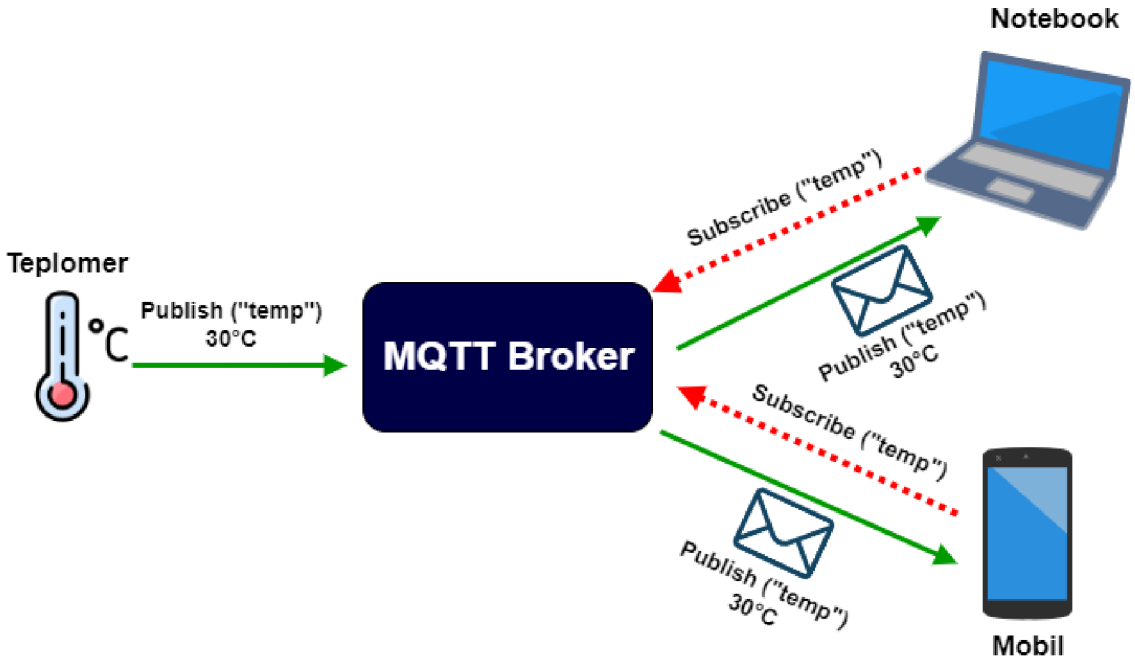
MQTT má tiež vlastnú sadu pojmov, jedným z najdôležitejších je `topics` (téma), téma je reťazec (UTF-8), ktorý je využívaný sprostredkovateľom na filtráciu správ pre klientov. Témy môžu byť rozdelené na viacero úrovní, každá úroveň je oddelená lomítkom (`/`). Klient môže do témy prispieť správou pomocou `publish` (publikovanie), alebo byť `subscriber` (odberateľ) danej témy. V prípade, že klient dané téma odberá, vidí všetky správy, ktoré sú publikované ostatnými klientmi. Pri odoberaní určitých tém, má klient na výber odoberať, buď konkrétnu tému (`domov/prízemie/kuchyňa/osvetlenie`), alebo použiť zástupné znaky na odoberanie viacerých tém súčasne.

MQTT používa dvojicu zástupných znakov[41]:

- **Jedna úroveň (+)**, zastupuje jednu úroveň v téme. Napríklad, odberateľ témy *domov/prízemie/+/osvetlenie* bude dostávať informácie o osvetlení v hociktorej miestnosti na prízemí.
- **Viac úrovní (#)**, môže byť použitý výlučne ako posledný znak témy. Odberateľ témy *domov/prízemie/#* bude dostávať informácie od všetkých poskytovateľov na prízemí.

Pôvodne bol protokol MQTT vyvinutý, ako malý a efektívny protokol na prenos dát cez nespoľahlivé linky, tým pádom bezpečnosť nebola jedným zo základných požiadaviek pri jeho implementácii. Existujú však rôzne bezpečnostné techniky, ktoré zabezpečujú MQTT[42]:

- **Zabezpečenie siete** - Fyzicky bezpečná sieť, alebo pripojenia realizované prostredníctvom VPN.
- **Prihlasovacie meno a heslo** - MQTT podporuje prihlasovanie pomocou mena a hesla, avšak tieto záznamy sú prenášané v otvorenom formáte. Vo veľa prípadoch sa meno a heslo používa, nie ako bezpečnostný faktor, ale ako spôsob zabránenia nechcenej komunikácie.
- **SSL/TLS** - MQTT sa spolieha na transportný protokol TCP, ktorý nie je šifrovaný, na porte 1883. Na šifrovanie spojenia sa využíva protokol TLS (SSL), ktorý používa štandardizovaný port 8883 pre bezpečnú komunikáciu. TLS je odporúčaná forma zabezpečenia komunikácie prostredníctvom MQTT.



Obr. 3.3: MQTT publish/subscribe model[43]

3.3.1 MQTT-SN

MQTT Sensor Network je protokol prenosu správ pre vstavané zariadenia, ktoré nepodporujú TCP/IP siete. Využíva rovnaký mechanizmus (publish/subscribe) prenosu správ ako štandardné MQTT, avšak primárnym prenosovým protokolom je

UDP. Najčastejšie sa MQTT-SN využíva v prostrediach Zigbee a Z-Wave.
Medzi hlavné rozdiely oproti štandardnému MQTT patrí:

- Klienti vedia nadviazať spojenie s gateway, bez počiatočnej konfigurácie.
- Menšia potrebná šírka pásma, kvôli prepracovanej veľkosti paketov.
- Preddefinované `topicsID` a krátke `topic name`, sú dostupné bez registrácie k broker.
- Protokol UDP nepotrebuje udržiavať neustále spojenie medzi klientmi, čím sa znižuje réžia komunikácie.
- Podpora „spiacich klientov“, kedy sa správy ukladajú do vyrovnávacej pamäte gateway a doručia sa po prebudení.
- Podpora QoS úrovne 3, čo v tomto prípade znamená, že sa nevyžaduje vytvorenie počiatočného spojenia pre vysielanie.

Najväčšou nevýhodou oproti bežnému MQTT je nedostatočná podpora a nedostupnosť kvalitného brokera (sprostredkovateľa)[44].

3.3.2 Mosquitto

Mosquitto je open source MQTT broker, ktorý poskytuje serverové a klientské implementácie protokolu MQTT. Mosquitto podporuje všetky verzie protokolu MQTT (5.0, 3.1.1 a 3.1). Vďaka jeho jednoduchosti a kompaktnosti je vhodný na použitie na všetky typy zariadení (desktop, servery, jednodoskové počítače). Mosquitto je dostupné na inštaláciu na viacerých platformách, ako Windows, macOS a rôzne distribúcie Linuxu[45].

Projekt Mosquitto ponúka dva typy MQTT klientských služieb príkazovej riadky, `mosquitto_sub` a `mosquitto_pub`. Tieto nástroje sú vynikajúce na vykonávanie rýchlych testov a riešenie problémov. `Mosquitto_pub` je klient, ktorý publikuje správu na určité téma. `Mosquitto_sub` naopak dané téma odoberá, prijíma a vypisuje prijatú správu[46].

3.4 Docker

Docker, v súčasnosti často skloňovaný pojem, jedná sa o open source platformu používanú pri, vývoji, nasadení a správe aplikácií v kontajneroch. Kontajnery sú štandardizované spustiteľné komponenty, ktoré obsahujú zdrojový kód, systémové nástroje, knižnice a konfiguračné súbory potrebné na spustenie danej aplikácie v akomkoľvek prostredí. Na vytváranie kontajnerov, do ktorých sa ukladajú aplikácie, využíva Docker virtualizáciu. Tento koncept sa zdá byť podobný s virtuálnymi strojmi (VM), keďže oba využívajú izolované virtuálne prostredia, avšak kontajnery sú rýchlejšie, efektívnejšie a menšie ako plnohodnotné virtuálne stroje.

Obľúbenosť Dockeru je podnietená hlavne kompatibilitou medzi rôznymi operačnými systémami (Windows, macOS, viaceré Linuxové distribúcie), ako aj prepracovanými návodmi a dokumentáciou[47].

3.4.1 Docker Engine

Docker Engine je považovaný za jadro Dockeru. Ide o odľahčené behové prostredie a aplikáciu klient-server, ktorá vytvára s spravuje kontajnery. Docker Engine sa skladá z troch hlavných komponent[48]:

- **Server** - daemon (`dockerd`), beží na pozadí a stará sa o vytváranie a spravovanie kontajnerov.
- **Rest API** - špecifikujú rozhrania, ktorými môžu programy komunikovať a riadiť démona.
- **Klient** - (`docker`) umožňuje používateľom interagovať s démonom prostredníctvom API (príkazy, skripty).

Zoznamy a stavy kontajnerov sú zobrazené po zadaní príkazu `docker ps`, ktorý je považovaný za najzákladnejší príkaz pri skúmaní stavov kontajnerov. Zobrazí základné informácie o kontajneroch, ako ID, obraz, príkaz na spustenie, dátum vytvorenia, meno, porty (ak sú otvorené). Každý kontajner môže byť v jednom zo siedmych stavov[49]:

- **Created** - kontajner bol vytvorený, ale nie spustený.
- **Restaring** - reštartujúci sa, napríklad pri vzniku chyby.
- **Running** - naznačuje úspešné spustenie Dockera.
- **Paused** - kontajner bol pozastavený (údržba).
- **Removing** - vo fáze odstraňovania, tento stav naznačuje, že sa jedná o väčší kontajner, alebo chybu pri odstraňovaní.
- **Exited** - kontajner ukončil svoju činnosť úspešne.
- **Dead** - nefunkčný kontajner (dá sa len odstrániť).

```
1 docker ps -a
```

Výpis 3.1: Výpis všetkých kontajnerov

3.4.2 Docker Images

Pre každý kontajner je definovaný image (obraz), z ktorého sa spúšťa. Image je šablóna s inštrukciami pre vytváranie Docker kontajnerov. Docker image obsahuje zdrojový kód, knižnice, nástroje a ďalšie súbory potrebné na beh aplikácie. Pri spustení image, sa kontajner stáva jeho inštanciou, týmto sa docieli možnosť spúšťania viacerých kontajnerov z rovnakého image. Najväčším verejným úložiskom imageov

je cloudová služba Docker Hub, ponúka veľké množstvo imageov na rôzne využitie (webové servery, upravené Linuxové distribúcie). Na Docker Hub môžu nahrávať svoje dáta ľubovoľní užívatelia, toto umožňuje útočníkom nahrať svoje okopírované image (obsahujúci škodlivý softvér), s cieľom oklamať používateľov, že pochádza z oficiálneho zdroja. Z pohľadu bezpečnosti sa odporúča vyberať image, ktoré sú označené ako oficiálne a preveriť ich obsah. Ďalším spôsobom získania imageov je tvorba vlastných pomocou Dockerfile. Táto metóda je výhodnejšia, pretože umožňuje tvorbu personalizovaných imageov, podľa požiadaviek užívateľa[50].

Kontajnery sa spúšťajú príkazom `docker run`, ktorý je možné doplniť o viaceré argumenty. Parameter `-d` spustí kontajner na pozadí (štandardne sa spúšťa na popredí), `--name` nastaví meno pre daný kontajner, `-p` mapuje hosťovský port na port kontajnera a `-v` špecifikuje umiestnenie konfiguračných súborov. V prípade, že image nie je dostupný, Docker Engine sa ho pokúsi stiahnuť a následne spustiť kontajner.

```
1 #spustenie kontajnera s konkrétnymi argumentmi
2 docker run -d --name <meno_kontajnera> -p <port_h:
3 port_k> -v <cesta_k_config>:/config <nazov_image>
4
5 #zastavenie kontajnera
6 docker stop <meno_kontajnera>
```

Výpis 3.2: Ukážka základných príkazov v Dockeri

3.4.3 DockerFile

Ako už bolo zmienené, ďalším spôsobom získania imageov je vytvorenie vlastných. **Dockerfile** je textový konfiguračný súbor, ktorý obsahuje inštrukcie na vytváranie imageov a zároveň celý proces automatizuje. Pri vytváraní súboru Dockerfile, je zvyklosťou tento súbor umiestniť do samostatného priečinka so všetkými súbormi, ktoré budú pri vytváraní použité. Docker vo svojej terminológii toto umiestnenie nazýva ako `build context`. Z bezpečnostného hľadiska nie je možné zahrnúť súbory mimo špecifickú zložku a to hlavne z dôvodu, použitia stiahnutého Dockerfile napríklad z Docker Hub. Toto opatrenie zaručuje, že pri zostavovaní image sa doň nedostane žiadny citlivý súbor užívateľa[51].

Nasledujúca ukážka približuje problematiku vytvárania vlastného Dockerfile. Ide o jednoduchý Dockerfile pre vytvorenie ssh servera v kontajneri.

```
1 FROM ubuntu:20.04
2 RUN apt update && apt install openssh-server -y
3 RUN useradd -rm -d /home/upuser -g root -G sudo
4 -u 1000 upuser
5 RUN echo 'upuser:passworrd' | chpasswd
6 RUN service ssh start
7 EXPOSE 22
8 CMD ["/usr/sbin/sshd", "-D"]
```

Výpis 3.3: Ukážka jednoduchého Dockerfile pre ssh server

Každý Dockerfile musí začať inštrukciou `FROM`, ktorá špecifikuje rodičovský image. V tejto ukážke sa vychádzalo z Linuxovej distribúcie Ubuntu vo verzii 20.04, ak by sme za `:` nedefinovali takzvaný „tag“ 20.04, builder by automaticky použil `latest` (v našom prípade najnovšiu verziu Ubuntu). `RUN` definuje spustenie príkazu cez shell, udáva aký proces bude bežať vo vnútri kontajnera pri jeho spustení. Inštrukcia `EXPOSE` informuje Docker, že kontajner pri behu počúva na špecifickom porte, ak protokol nie je špecifikovaný, používa štandardne TCP. Príkaz `CMD` špecifikuje inštrukciu, ktorá sa má vykonať pri spustení Docker kontajnera, jej hlavným účelom je spúšťanie programu potrebného v kontajneri[52].

Príkaz `docker build` zostavuje image z Dockerfile a z build context. Argument `-t` sa používa pre pomenovanie image. Mená pre image majú predpísaný vzor na ich tvorbu, najprv je nutné zadať meno image a verziu. Meno image špecifikuje pomenovanie, ktoré bude niesť daný kontajner, verzia (tag) nemá predpísanú konkrétnu konvenciu a je na autorovi ako ju zvolí, od mena sa oddeľuje dvojbodkou a pri nešpecifikovaní sa predvolene použije `latest`. Bodka na konci príkazu udáva, že Dockerfile je umiestnený v aktuálnom adresári a nie je nutné špecifikovať celú cestu.

```
1 docker build -t <meno_image>:<tag> .
```

Výpis 3.4: Príkaz docker build

3.4.4 Docker Compose

Docker Compose je nástroj od vývojárov Dockeru, pre centrálnu správu viacerých kontajnerov. Príkaz `docker run` ako je zobrazené v ukážke 3.2, v sebe zahŕňa špecifikácie portov, nastavenie mien, cestu ku konfiguračnému súboru, parametre a ďalšie, preto býva mnohokrát veľmi rozsiahli. V prípade nutnosti spustenia a prepojenia viacerých kontajnerov sa stáva príkaz `docker run` nepohodlným. Konfiguračný

súbor `docker-compose.yaml` obsahuje všetky potrebné nastavenia pre fungovanie a komunikáciu kontajnerov s vonkajším prostredím, alebo medzi sebou. Konfiguračný súbor je vo formáte YAML, ktorého hlavnou výhodou je jednoduchá čitateľnosť pre ľudí. Prednosťou tejto metódy je skutočnosť, že Docker Compose dokáže vyčítať konfiguráciu z konfiguračného súboru a uľahčuje prácu správcom, ktorí sú za pomoci pár príkazov schopní spúšťať, zastavovať a sledovať kontajnery súčasne[53].

```
#spustí kontajnery z konfiguračného súboru,
#parameter -d spustí kontajner na pozadí
docker-compose up -d

#zastaví a odstráni kontajnery a siete vytvorené
#pomocou docker-compose up
docker-compose down

#výpis kontajnerov a ich vlastností
docker-compose ps
```

Výpis 3.5: Príklady použitia docker-compose

Pre správnu funkčnosť nástroja Docker Compose je pri spustení, potrebné nachádzať sa v adresári s konfiguračným súborom. Názov konfiguračného súboru `docker-compose.yaml` je preddefinovaný a z dôvodu funkčnosti a kompatibility je vždy nutné ho nazvať takto.

```
version: '3.0'
services:
    homeassistant:
        container_name: homeassistant
        image: "home-assistant/home-assistant:stable"
        volumes:
          - /opt/homeassistant/config:/config
          - /etc/localtime:/etc/localtime:ro
        restart: unless-stopped
        privileged: true
        network_mode: host
```

Výpis 3.6: Ukážka súboru docker-compose.yaml pre HomeAssistant

4 Linuxová distribúcia pre gateway

Nasledujúca kapitola sa zameriava na tvorbu samotnej Linuxovej distribúcie, ktorá poslúži ako hlavný riadiaci prvok gateway. Prvá časť pojednáva o prerekvizitách potrebných k samotnej inštalácii systému, o manuálnom pozmeňovaní systému a pridávaní aplikácií slúžiacich na riadenie. Následne sú popísané možné problémy pri práci s Docker a VPN. Nakoniec je predstavená možnosť automatizovanej tvorby celej distribúcie s vysvetlením funkčnosti redundancie systému.

Ešte pred samotnou inštaláciou je nutné mať k dispozícii USB s kapacitou aspoň 8GB a stiahnutú serverovú distribúciu Ubuntu 20.04.5 z oficiálnej stránky `https://ubuntu.com/download/server#downloads`.

Po úspešnom stiahnutí potrebného obrazu, je nevyhnutné tento ISO obraz napáliť na USB. Pri napaľovaní na hosťovskom systéme Windows sa typicky volí súborový systém FAT32, pretože nie všetky Linuxové distribúcie podporujú NTFS. Najčastejšie používané nástroje na napaľovanie sú:

- **Rufus** - používaný pre hosťujúci systém Windows.
- **Balena Etcher** - používaný v Linuxových systémoch.

Pri prvotnom spustení je potrebné okrem samotného SBC, USB s obrazom Ubuntu a napájania, mať k dispozícii, internetové pripojenie, klávesnicu a obrazovku. Pre napájanie UP Board je doporučené použiť zdroj s výkonom 5V/4A. Firmware v UP Board je nastavený tak, aby pri spustení kontroloval, či je pripojené bootovateľné USB, ak áno, pokúsi sa z neho načítať systém, ak nie, pokračuje štandardné spustenie systému.V prípade, že na UP Board už máme nainštalovaný operačný systém, je potrebné toto USB odstrániť, aby po sa po reštarte zariadenia neprepísal. Inštalácia Ubuntu server je detailnejšie popísaná v prílohe A.

4.1 Inštalácia SSH servera

Možnosť vzdialene ovládať server a jeho funkcie je v dnešnej dobe samozrejmosťou, preto sa pri konfigurácii UP Board využije práve táto alternatíva. Ďalšou výhodou vzdialeného spojenia je, že Ubuntu server štandardne neponúka grafické užívateľské rozhranie, ktoré môže byť v niektorých prípadoch výhodné. Z dôvodu bezpečnosti a autentickosti sa použije šifrované spojenie pomocou SSH (Secure Shell Protocol). SSH server je možné nainštalovať priamo pri inštalácii Ubuntu, ako voliteľnú položku, alebo doinštalovať do funkčného systému. Inštalácia prebieha jednoducho pomocou nasledujúceho príkazu.

```
1 sudo apt-get install openssh-server
```

Overenie funkčnosti SSH servera je možné skrz príkaz `systemctl status sshd` (viď obrázok 4.1).
Povoliť a spustiť SSH server je potrebné v prípade nefunkčnosti ihneď po inštalácii, jednoducho následujúcimi príkazmi.

```
1 sudo systemctl enable ssh
2 sudo systemctl start ssh
```

```
upuser@upboard-server:/$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor prese>
     Active: active (running) since Sat 2022-12-03 13:30:41 UTC; 31min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 702 (sshd)
      Tasks: 1 (limit: 1569)
     Memory: 5.5M
     CGroup: /system.slice/ssh.service
             └─702 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
```

Obr. 4.1: Ukážka funkčného SSH servera

4.2 Inštalácia UP Board kernelu

UP Board oficiálne podporuje Linuxové jadro vo verzii 5.4.0, ktoré je overené na viacerých jednodoskových počítačoch od firmy UP.
Po inštalácii serverovej distribúcie Ubuntu a následného reštartu zariadenia sa pridá oficiálny PPA repozitár UP Board, v ktorom sa nachádza aj spomínané jadro.
Nasledujúci príkaz vytvorí súbor v adresári `/etc/apt/sources.list.d`. Pri pridávaní vlastných zdroj je doporučované použiť tento adresár a nie hlavný konfiguračný súbor `/etc/apt/sources.list`.

```
1 sudo add-apt-repository ppa:up-division/5.4-upboard
```

Ďalej po pridaní potrebného repozitára, je potrebné stiahnuť a aktualizovať informácie o balíčku zo všetkých nakonfigurovaných zdrojov.

```
2 sudo apt update
```

Po aktualizovaní informácií o dostupných balíčkoch sa prejde k odstráneniu starej verzie jadra. Nasledujúci príkaz odstráni všetky súčasti jadra, ktoré začínajú slovom `linux-` a končia na `generic`, zástupný znak `.*` priradí ľubovoľný reťazec medzi vyššie spomínané slová. Pri výzve „Prerušiť odstránenie jadra“ je nutné vybrať „NIE“, inak by celý proces skončil.

```
3 sudo apt-get autoremove --purge 'linux-.*generic'
```

Následne je možné nainštalovať nové jadro (5.4.0). Ubuntu vo verzii 18.04 a 20.04 zdieľajú rovnaké jadro, preto je v následujúcom príkaze použitá verzia 18.04. Vychádza to čisto z pomenovania súboru v repozitári UP a funkčnosť systému to nijak neovplyvní.

```
4 sudo apt-get install linux-generic-hwe-18.04-5.4-upboard
```

Príkaz `dist-upgrade` okrem funkcie `upgrade` dokáže efektívne spracovať zmeny v závislostiach balíčkov. Toto zahŕňa, odstránenie nepotrebných závislostí, riešenie konfliktov medzi balíčkami, ktoré vznikli v dôsledku zmien závislostí. Taktiež sa používa pri inštalácii novej verzie jadra. `Update-grub` hľadá súbory v `/boot`, ktoré začínajú na „vmlinuz-“ a vytvorí záznamy typu grub pre každý. Týmto zaručíme, že pri bootovaní zariadenia sa použije novo nainštalované jadro.

```
5 sudo apt dist-upgrade -y
6 sudo update-grub
```

```
7 sudo reboot
```

Reštartovaním zariadenia sa prevedú vykonané zmeny v systéme, ktorými sa zavedie nové jadro vo verzii 5.4.0.

```
upuser@upboard-server:~$ uname -a
Linux upboard-server 5.4.0-1-generic #0~upboard5-Ubuntu SMP
Fri Jan 7 11:53:57 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

Obr. 4.2: Kontrola verzie jadra

4.3 Inštalácia Docker a Docker Compose

Pred samotnou inštaláciou je doporučená aktualizácia apt balíčkov, v tomto prípade to ale nie je nutné, pretože celý systém bol aktualizovaný pri inštalácii jadra.
Ďalej je potrebná inštalácia závislostí, ktoré potrebuje Docker pre svoju funkčnosť. Väčšina z týchto závislostí je predinštalovaná v Ubuntu a nie je nutná ich dodatočná inštalácia.

```
sudo apt-get install \
   ca-certificates \
   curl \
   gnupg \
   lsb-release
```

Oficiálny Docker GPG kľúč sa pridáva do `/usr/share/keyrings`, tento adresár je odporúčaným umiestnením pre konvertované súbory GPG, pretože je predvoleným umiestnením ostatných kľúčov v systéme.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg
   | sudo gpg --dearmor -o /usr/share/keyrings/docker-
   archive-keyring.gpg
```

Pre správnu funkčnosť sa nastaví repozitár pre stabilnú verziu nástroja Docker (možno použiť aj „beta“, alebo „experimentálnu“ verziu).

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/
   usr/share/keyrings/docker-archive-keyring.gpg] https
   ://download.docker.com/linux/ubuntu $(lsb_release -cs
   ) stable" | sudo tee /etc/apt/sources.list.d/docker.
   list > /dev/null
```

Po pridaní požadovaného repozitára, je potrebné stiahnuť a aktualizovať informácie o balíčku zo všetkých nakonfigurovaných zdrojov a následne prejsť k inštalácii samotného Docker Engine.

```
sudo apt update
sudo apt-get install docker-ce docker-ce-cli
   containerd.io
```

Či všetko funguje, je možné overiť spustením kontajnera „Hello World“. Ak inštalácia prebehla v poriadku a Docker je správne nastavený, kontajner zobrazí správu a automaticky sa vypne.

```
upuser@upboard-server:/opt$ sudo docker run hello-world
[sudo] password for upuser:

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Obr. 4.3: Overenie funkčnosti pomocou „Hello World“

Inštalácia Docker Compose prebieha obdobne ako Docker Engine, najskôr sa stiahne najnovšia stabilná verzia.

```
sudo curl -L "https://github.com/docker/compose/releases
  /download/1.29.2/docker-compose-$(uname␣-s)-$(uname␣-
  m)" -o /usr/local/bin/docker-compose
```

Nastavia sa povolenia pre stiahnuté súbory, aby ich bolo možné spúšťať.

```
sudo chmod +x /usr/local/bin/docker-compose
```

Týmto sa nainštaloval Docker Compose a príkazom `docker-compose --version` je možné overiť jeho funkčnosť.

```
upuser@upboard-server:/opt$ docker-compose -version
docker-compose version 1.29.2, build 5becea4c
```

Obr. 4.4: Overenie funkčnosti Docker Compose

4.4 Inštalácia Home Assistant pomocou Docker Compose

Pred začiatkom inštalácie je nutné zvoliť adresár, v ktorom sa budú nachádzať konfiguračné súbory. Pre účely tejto práce bol zvolený adresár `/opt`, v ktorom bude vytvorený konfiguračný súbor pre Docker Compose.

```
cd /opt
sudo vim docker-compose.yaml
```

Konfiguračné informácie sú pridané do tohoto súboru, konkrétne prevedenie konfiguračného súboru pre inštaláciu Home Assistant je zobrazené vo výpise 4.1.
Po uložení konfigurácie je možné spustiť Docker pomocou príkazu `docker-compose up -d`. Pred spustením príkazu je potrebné uistiť sa, že sa nachádzame v adresári `/opt`, pretože je potrebné ho spustiť z rovnakého adresára, ako v ktorom je uložený súbor „docker-compose.yaml“. Pre výpis spustených kontajnerov je možné použiť príkaz:

```
sudo docker ps
```

4.5 Inštalácia Wireguard VPN pomocou Docker Compose

Wireguard je open-source VPN, známy pre svoju jednoduchosť a ľahké používanie. Od konkurencie sa najviac odlišuje svojim prístupom k bezpečnosti. Wireguard nezakladá konfigurácie zabezpečenia medzi klientom a serverom, tie sú už preddefinované. Jedná o službu bez spojenia, pretože nie je potrebné pripájať sa, respektíve opätovne sa pripájať na server, vďaka čomu je tento protokol významne rýchlejší než jeho konkurencia.

Pred samotnou inštaláciou je potrebné vytvoriť adresár na mieste, kde sa budú nachádzať súbory pre VPN. Z dôvodu konzistencie je zvolený adresár `/opt`. Po vytvorení adresára sa upraví hlavný konfiguračný súbor `docker-compose.yaml`, do ktorého je pridaný kód na inštaláciu Wireguard.

```
mkdir /opt/wireguard-server
sudo vim docker-compose.yaml
```

```
version: '3.0'
services:
    wireguard:
        image: lscr.io/linuxserver/wireguard:latest
        container_name: wireguard
        cap_add:
            - NET_ADMIN
            - SYS_MODULE
        environment:
            - SERVERURL=192.168.10.34
            - SERVERPORT=51820
            - INTERNAL_SUBNET=10.13.13.0/24
            - ALLOWEDIPS=0.0.0.0/0
        volumes:
            - /opt/wireguard-server/config:/config
            - /lib/modules:/lib/modules
        ports:
            - 51820:51820/udp
        sysctls:
            - net.ipv4.conf.all.src_valid_mark= 1
        restart: unless-stopped
```

Výpis 4.1: Redukovaný docker-compose.yaml na inštaláciu Wireguard VPN

Po uložení konfigurácie sa príkazom `docker-compose up -d` stiahne a nastaví požadovaný kontajner, tento príkaz je nutné spúšťať v adresári s konfiguračným súborom. Medzi dôležité parametre tohoto výpisu patria napríklad `NET_ADMIN, SYS_MODULE`, ktoré dodávajú kontajneru dodatočné oprávnenia v systéme. Konkrétne môže ísť o inštaláciu modulov jadra a upravovanie sieťového rozhrania. Wireguard sa v skutočnosti spúšťa vo vnútri Linuxového jadra, preto je taktiež potrebné pripojiť priečinok `/lib/modules`.
Či inštalácia prebehla v poriadku je možné overiť príkazom `docker ps`, ktorý vypíše základné informácie o spustených kontajneroch.

Pre pripojenie klienta k VPN serveru je potrebné zo serveru skopírovať konfiguračný súbor klienta do klientskej stanice. Klient musí mať na svojom počítači už nainštalovaný Wireguard (`sudo apt install wireguard`). Následne je možné napríklad pomocou SCP stiahnuť daný súbor ku klientovi.

```
1 scp USER@192.168.10.34:/opt/wireguard-server/config/
    peer_1/peer_1.conf ~/Desktop
2 mv peer_1.conf /etc/wireguard/wg0.conf
```

Pre automatické pripájanie sa k VPN serveru je potrebné u klienta zadať nasledujúce príkazy, ktoré povolia pripájanie sa pri štarte systému a spustia sa samotnú službu.

```
1 sudo systemctl enable wg-quick@wg0.service
2 sudo systemctl start wg-quick@wg0.service
```

Pre manuálne pripojenie k tunelu sa zadávajú príkazy:

```
1 #spustenie
2 sudo wg-quick up wg0
3 #zastavenie
4 sudo wg-quick down wg0
```

4.5.1 Overenie funkčnosti VPN tunela

Na overenie funkčnosti VPN servera sa použije príkaz `ip route get „ip-addr"`. Overovať budeme adresou `8.8.8.8`, čo je IP adresa Googla. Pred samotnou skúškou je nutné zastaviť VPN server a následne zadať príkaz.

```
ubuntu-pc@ubuntu:~$ ip route get 8.8.8.8
8.8.8.8 via 192.168.10.1 dev ens33 src 192.168.10.35 uid 1000
    cache
```

Obr. 4.5: Smerovanie bez VPN

Z obrázka je zrejmé, že klient pristupuje ku Google cez základné sieťové rozhranie `ens33`. V tomto prípade žiadne spojenie cez VPN server neprebieha.

Pri spustení VPN servera (`sudo wg-quick up wg0`), je viditeľné, že sieťová prevádzka je smerovaná cez virtuálne rozhranie `wg0`, čo predstavuje VPN server.

```
ubuntu-pc@ubuntu:~$ ip route get 8.8.8.8
8.8.8.8 dev wg0 table 51820 src 10.13.13.2 uid 1000
    cache
```

Obr. 4.6: Smerovanie s aktívnym VPN

4.5.2 Možné problémy s Wireguard

V prípade nefunkčnosti služby, vzniká potreba na klientovi nainštalovať balíček `resolvconf`, ktorý spravuje informácie o mennom serveri.
Na strane s Docker kontajnerom je potrebné overiť, aký menný server (DNS) beží vrámci Dockera (`cat /etc/resolv.conf`). Taktiež treba overiť DNS hostiteľského servera. Ak je v hociktorom prípade predvolená IP adresa 127.0.0.53 (predstavuje lokálnu DNS cache), nebude spojenie fungovať. Najlepším riešením v tomto prípade je prepísať hostiteľský `/etc/resolv.conf`. Keďže súbor `resolv.conf` je len symbolický odkaz na `/run/systemd/resolve/stub-resolv.conf` (127.0.0.53), je potrebné zmeniť tento odkaz na `/run/systemd/resolve/resolv.conf`, ktorý predstavuje zoznam reálnych DNS serverov.

```
1 sudo ln -sf /run/systemd/resolve/resolv.conf /etc/resolv
    .conf
```

Po tomto kroku sa skontroluje záznam v `/etc/resolv.conf` a v prípade platného DNS servera je zapotreby reštartovať službu Docker.

Na klientskej strane treba skontrolovať záznam v `/etc/wireguard/wg0.conf`, ktorý predstavuje konfiguračný súbor klienta a v prípade potreby opraviť pole DNS na platnú IP z `resolv.conf`.

4.6 Automatizovaná tvorba Linuxovej distribúcie

Pred samotným spustením skriptu je potrebné previesť inštaláciu systému Ubuntu server spôsobom uvedeným v prílohe A. Ukážka inštalácie prebieha vo virtuálnom prostredí VMware, avšak bodom 10 je inštalácia takmer totožná s fyzickým zariadením. Priebeh skriptu je časovo náročnejší proces, hlavne z dôvodu výkonového obmedzenia jednodoskového počítača. Výmena Linuxového jadra a inštalácia potrebných aplikácii a závislostí zaberá približne $10 - 15$ minút. Vykonávanie zálohy a tvorba nového zavádzača trvá okolo 25 minút. Pri zarátaní času potrebného na inštaláciu systému a následného vykonávania skriptu sa táto doba predlžuje na približne 1 hodinu.

Postup spustenia skriptu:

1. V domovskom adresári užívateľa je zadaný príkaz: `git clone https://github.com/TololoSK/ubuntu-upboard.git`, ktorý stiahne všetky potrebné súbory a skripty.
2. Príkazom `sudo chmod +x reboot.sh` je udelené oprávnenie na spustenie súboru, čo umožní jeho spustenie ako programu, respektíve skriptu.
3. Následne je možné skript spustiť: `sudo ./reboot.sh`

V rámci automatizácie procesu vytvárania personalizovanej Linuxovej distribúcie bolo vytvorených viacero skriptov v Bash a Python, ktoré na seba nadväzujú a ich činnosť je systematická. Prvým a zároveň najdôležitejším skriptom je **`reboot.sh`**, ktorý na začiatku kontroluje splnenie požiadaviek pre inštaláciu systému a následne vykonáva samotné pridávanie potrebných balíčkov. Druhý skript **`copy.sh`**, ako jeho názov napovedá, prevádza predovšetkým kopírovanie systému na záložnú partíciu. Nasledujúci skript kontroluje stav napätia na svojich pinoch a reaguje na ich zmenu. Posledný skript **`pressed.sh`** je spustený v prípade stlačenia tlačítka a vykonáva automatické pre-bootovanie do novej partície. Taktiež bol vytvorený redukovaný súbor pre GRUB **`grub.cfg`**, ktorý obsahuje len nami potrebné záznamy pre bootovanie.

4.6.1 Základ automatizácie

Základom pre automatizáciu bol skript **`reboot.sh`**, ktorý sa skladá z približne 175 riadkov a je možné ho rozdeliť na 3 hlavné časti.

Prvá časť pozostáva z definície premenných, ktoré budú často využívané pri behu skriptu a z kontroly parametrov potrebných pre úspešný priebeh skriptu. Ako prvé sa kontroluje predchádzajúci beh skriptu pomocou „príznakového súboru“, ktorý sa vytvára len na označenie toho, či bola/nebola vykonaná určitá akcia. Následne skript kontroluje požiadavky na veľkosti jednotlivých partícii, ktoré sú nevyhnutné pre beh skriptu. Najskôr je porovnávaná veľkosť 2. partície so statickou hodnotou 512MB v

bajtovom prevedení, viz. výpis 4.2. Existencia tejto partície umožňuje logicky oddeliť nami používaný bootleader od zvyšku systému. Následne skript hľadá, či existujú 2 partície, ktoré sú na byte presne rovnako veľké. Tento krok je z dôležitý z pohľadu redundancie systému, keďže pri neskoršom behu skriptu medzi týmito partíciami prebieha kópia z jednej na druhú. Posledný kontrolný blok overuje, či je partícia *mmcblk0p3* pripojená (*mounted*) ako koreňový súborový systém (**/**).

```
1 part_size=$(sudo parted /dev/mmcblk0 unit B print | awk
    '/^ 2/{print $3}' | sed 's/B//')
2   if [[ $part_size -gt 536870912 ]]; then
3       echo "/dev/mmcblk0p2 partition is at least 512MB"
4   else
5       exit 1
6   fi
```

Výpis 4.2: Ukážka kontroly veľkosti partície v bajtoch

V druhej časti skriptu sú vytvorené služby, ktoré zaisťujú, že skripty budú aktívne aj po reštarte zariadenia a zaisťujú ich beh na pozadí. Výpis 4.3 približuje telo služby **reboot-sh.service**, ktorá spúšťa skript **reboot.sh** po reštarte. Najdôležitejšími časťami tejto služby sú:

- `After` - určuje poradie, v ktorom sa služby spúšťajú, zaisťuje, že sa služba nespustí, kým nebudú k dispozícii sieťové a grafické rozhrania.
- `ExecStart` - určuje konkrétny príkaz (skript), ktorý sa má spustiť.
- `WantedBy` - zaisťuje, že sa služba spustí po dosiahnutí zadaného cieľa.

Taktiež tu prebieha automatická výmena jadra na verziu 5.4.0, ktoré pochádza priamo od výrobcov UP Board. Tento proces je totožný s postupom v sekcii 4.2. Nakoniec je vytvorený takzvaný „príznakový súbor“, ktorý zaistí, že po reštarte nebudú vyššie popísané kroky znova vykonané a zariadenie sa reštartuje pre zavedenie nového jadra.

```
1 [Unit]
2 Description=service for reboot.sh
3 After=network.target graphical.target
4
5 [Service]
6 ExecStart=/home/upuser/ubuntu-upboard/reboot.sh
7
8 [Install]
9 WantedBy=graphical.target
```

Výpis 4.3: Ukážka služby reboot-sh.service

Tretia časť pozostáva z príkazov, ktoré sú vykonané po reštarte zariadenia. Ide o inštaláciu požadovaných balíčkov ako Docker, Docker Compose, Pip, Wireguard a balíček na správu 40-pinovej GPIO zbernice RPi.GPIO. Je vytvorený adresár pre ukladanie užívateľských dát v Docker, vytvorí a naplní sa súbor *docker-compose.yaml*, ktorý slúži na efektívnu správu viacerých kontajnerov. Odstráni sa služba **`reboot-sh.service`**, ktorej ďalší beh v systéme nie je potrebný a slúžila len pre oživenie skriptu pri prvom reštarte. Vytvorí a povolí sa služba **`copy-sh.service`**, ktorej účelom je spúšťať skript **`copy.sh`**. V tretej časti sa tiež pripojí druhá partícia, na ktorú bude nainštalovaný nový bootloader.

- `-root-directory` - špecifikuje názov prípojného bodu pre oddiel, na ktorom je potrebné vytvoriť nový adresár /boot/grub a naplniť ho súbormi GRUB.
- `-efi-directory` - je potrebné pre inštaláciu GRUB spôsobom, ktorý je kompatibilný s firmvérom UEFI.
- `/dev/mmcblk0` - určuje jednotku, na ktorú sa nainštaluje zavádzač.

Na konci sa za pomoci príkazu `source` spúšťa skript pre kopírovanie systému na záložnú partíciu.

```
1 sudo grub-install --root-directory=/mnt/mmcblk0p2 --efi-
    directory=/boot/efi /dev/mmcblk0
```

Výpis 4.4: Ukážka inštalácie GRUB na mmcblk0p2

4.6.2 Redundancia systému a možnosť bootovania

O tvorbu záložnej partície a o pridanie možnosti bootovania medzi vytvorenými partíciami sa stará skript **`copy.sh`**.

Pred začiatkom vykonávania skriptu sa kontroluje existencia „príznakového súboru“, ktorý zaisťuje jednorázový beh skriptu na konkrétnej partícii. V prípade, že tento súbor neexistuje, vykoná sa naplnenie premenných `TO` a `FROM`, pričom `TO` značí partíciu, na ktorú je potrebné vykonať zálohu a `FROM` je naopak partícia, ktorú je potrebné zálohovať. Je vykonaná úplná kópia systému na záložnú partíciu. Tento proces je časovo náročnejší a zaberá v priemere $20 - 25$ minút.

```
1 sudo dd if=$FROM of=$TO
```

Výpis 4.5: Ukážka príkazu na kopírovanie

Po skončení príkazu na kopírovanie systému je potrebné vygenerovať nový Univerzálny Identifikátor (UUID) súborového systému, aby bolo možné partície od seba odlíšiť a umožniť bootovanie z ktorejkoľvek z nich. Samotný príkaz na tvorbu UUID vyžaduje kontrolu súborového systému, respektíve kontrolu, ktorá hľadá a opravuje nekonzistencie v súborovom systéme, ako poškodené bloky, ktoré by mohli viesť k

strate dát. Týmto príkazom je `sudo e2fsck -f $TO -y`, ktorý s parametrom `-f` „force“ vykoná opravu a kontrolu bez nutnosti interakcie s užívateľom. Následne sú naplnené premenné pre jednotlivé UUID, ktoré sú v ďalších krokoch nahraté do `grub.cfg` súborov a súboru `/etc/fstab`. Z dôvodu kopírovania celého systému je taktiež nesmierne dôležité prepísať UUID v súbore `/etc/fstab` na záložnej partícii, aby bolo možné, v prípade potreby nabootovať z tejto partície. Účelom partície `mmcblk0p2` je slúžiť, ako hlavná pre boot systému. Z tohoto dôvodu je na ňu nahraný personalizovaný súbor `grub.cfg`, v ktorom sa príkazmi uvedenými nižšie nahradia ukazovatele na vygenerované a validné UUID.

```
1 sudo sed -i "s/<incidcator1-here>/$uuid2/g" /mnt/
    mmcblk0p2/boot/grub/grub.cfg
2 sudo sed -i "s/<incidcator2-here>/$uuid1/g" /mnt/
    mmcblk0p2/boot/grub/grub.cfg
```

Nakoniec je vytvorený „príznakový súbor“, ktorý zabráni ďalšiemu spusteniu skriptu na aktuálnej partícii.

4.6.3 Redundancia pomocou tlačítka

V tejto časti bol vytvorený Python skript **`button.py`**, ktorého úlohou je monitorovanie stavu tlačítka.

Knižnica RPi.GPIO je Python knižnica, ktorá poskytuje modul na ovládanie GPIO zbernice. Tento jednoduchý skript využíva túto knižnicu pri nastavovaní parametrov. Konkrétne nastavuje pin 15 ako vstup (*input*), vďaka čomu bude čítať stav tlačidla pripojeného na tento pin. `GPIO.add_event_detect()` je metóda, ktorou Python skript môže zistiť a reagovať na zmeny GPIO pinov. Táto metóda berie viacero argumentov:

- `15` - deteguje udalosti na pin 15.
- `GPIO.BOTH` - detekcia signálu na nábežnej a zostupnej hrane.
- `callback` - funkcia, ktorá je volaná pri pri zmene stavu signálu.
- `bouncetime` - ignoruje udalosti, ktoré sú kratšie ako definovaný čas v ms.

```
1 GPIO.add_event_detect(15, GPIO.BOTH, callback=pushed,
    bouncetime=600)
```

Výpis 4.6: Ukážka konkrétnej implementácie GPIO.add_event_detect()

Funkcia `pushed` načíta vstupný stav tlačidla a v prípade, že je tlačidlo stlačené (state==1), zaznamená aktuálny čas do premennej. Pri pustení tlačidla sa porovná zaznamenaný čas s aktuálnym a v prípade, že táto doba presiahla 5 sekúnd, je pomocou funkcie `subprocess.call()` spustený bash skript **`pressed.sh`**.

4.6.4 Automatické bootovanie na záložnú partíciu

V prípade poškodenia niektorej systémovej časti je potrebný efektívny mechanizmus, ktorý umožňuje bez zbytočného a zdĺhavého hľadania problémov sfunkčniť celý systém naraz. Pri tejto časti zohráva dôležitú rolu „záložná“ partícia, ktorá predstavuje klon pôvodnej plne funkčnej systémovej partície.

Pre tento účel bol vytvorený skript **pressed.sh**, ktorý je možné vyvolať stlačením tlačidla, alebo manuálnym spustením. Na začiatku sa naplnia premenné `TO` a `FROM`, ktorých hodnota je buď 0, alebo 1 v závislosti od partície, na ktorej systém beží. Čísla 0 a 1 označujú číslo záznamu (*munuentry*) v súbore `grub.cfg` (číslovania sa začína od 0). Tieto záznamy označujú, ktorý systém bude spustený pri nasledujúcom reštarte zariadenia a sú bežne používané pri dualbootoch. Premenná `TO` značí partíciu, ktorá má byť nastavená ako bootovacia v nasledujúcich krokoch a `FROM` značí aktuálnu partíciu. Príkazom z výpisu 4.7, sa nastaví ukazovateľ na konkrétne *menuentry* a reštartom zariadenia sa prevedie spustenie systému zo zvolenej partície. Podstatou tohoto kroku je zaistenie redundancie systému, pričom zvolená partícia `TO` sa stane hlavnou (systémovou) a `FROM`, ktorá slúžila ako systémová pred chybou, bude použitá ako záložná. Tento krok je zaistený opätovným spustením skriptu **copy.sh** popísaného v kapitole 4.6.2. Pri každom ďalšom reštarte zariadenia sa boot prevádza z aktuálnej systémovej partície. Tento postup pre redundanciu systému, v prípade poruchy systémovej časti je možné aplikovať vždy v prípade potreby, pričom sa vykonáva zmena aktívnej partície na záložnú a záložnej na aktívnu.

```
1 sudo grub-set-default --boot-directory=/mnt/mmcblk0p2/
    boot $TO
```

Výpis 4.7: Ukážka manuálneho nastavenia boot partície

Záver

V bakalárskej práci sa zaoberalo tvorbou Linuxovej distribúcie, ktorá bola implementovaná na jednodoskovom počítači a slúžila ako komunikačná brána inteligentnej domácnosti. Distribúcia vychádzala z bežnej distribúcie Ubuntu server vo verzii 20.04. Ako hardvér brány bol vybraný UP Board, ktorý pre jeho rýchlosť, užívateľskú podporu a cenu, predčil konkurenčné produkty.

Práca obsahuje všeobecnú charakteristiku operačného systému Linux. Sú v nej popísané jeho najdôležitejšie časti a sú predstavené a porovnané tri distribúcie, ktoré by mohli byť zvolené ako hlavný operačný systém komunikačnej brány. Po teoretickom rozbore vyplynulo, že ako serverová distribúcia má najviac predností práve Ubuntu, ktorá je oficiálne podporovaná aj samotným UP Boardom. Porovnanie jednodoskových počítačov neprebiehalo len z papierových hodnôt a hrubého výkonu, ale aj na základe užívateľskej, technickej podpory a online komunity, ktorá zohráva dôležitú rolu pri riešení problémov. Ako nástroj pre domácu automatizáciu bola zvolená platforma Home Assistant, ktorá umožňuje pomerne jednoduchú integráciu inteligentných zariadení do unifikovaného systému. Virtualizačný nástroj Docker významne zjednodušil inštaláciu aplikácií, ako Home Assistant a umožnil ich použitie na nami zvolenom hardvéri.

V rámci prípravy distribúcie, bol kladený dôraz na zostavenie návodov, ktoré špecificky popisovali jednotlivé kroky inštalácie. Bola opísaná inštalácia SSH servera, ktorý sme ako užívateľ potrebovali na vzdialené riadenie brány. Bol taktiež vypracovaný návod, na inštaláciu oficiálne podporovaného Linuxové jadra na UP Board, ktoré by malo lepšie optimalizovať systémové zdroje. Inštaláciou Docker a Docker Compose sme docielili kompatibilitu s platformou Home Assistant, ktorá je použitá ako riadiace centrum inteligentnej domácnosti. Pridanie Wireguard VPN umožní bezpečnú správu komunikačnej brány cez sieť.

Výstupom práce je systém, ktorý v sebe obnáša aplikácie potrebné pre funkčnú komunikačnú bránu. Systém bol segmentovaný a má znaky redundancie. Automatizácia procesu inštalácie aplikácii a segmentácie užívateľom zjednoduší prácu a umožní nasadenie systému do reálneho prostredia. Experimentom, pomocou senzoru teploty bolo potvrdené, že systém v skutočnosti spĺňa požiadavky komunikačnej brány a tým pádom považujem zadanie za splnené.

Literatúra

[1] Britannica, The Editors of Encyclopaedia. *Linus Torvalds* [online]. 2022. [cit. 14. 10. 2022]. Dostupné z URL: <https://www.britannica.com/biography/Linus-Torvalds>

[2] Red hat. *What is Linux?* [online]. 2022. [cit. 14. 10. 2022]. Dostupné z URL: <https://opensource.com/resources/linux>

[3] IONOS. *What is a bootloader and how does it work?* [online]. 2022. [cit. 14. 10. 2022]. Dostupné z URL: <https://www.ionos.com/digitalguide/server/configuration/what-is-a-bootloader/>

[4] TECHMINT. *4 Best Linux Boot Loaders.* [online]. 2021. [cit. 14. 10. 2022]. Dostupné z URL: <https://www.tecmint.com/best-linux-boot-loaders/>

[5] Red Hat. *What is the Linux kernel?* [online]. Február 27, 2019. [cit. 15. 10. 2022]. Dostupné z URL: <https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel/>

[6] TechTarget. *What is a kernel?* [online]. August 2022. [cit. 15. 10. 2022]. Dostupné z URL: <https://www.techtarget.com/searchdatacenter/definition/kernel>

[7] LibreTexts. *Linux Kernel Subsystem.* [online]. Apríl 19, 2021. [cit. 16. 10. 2022]. Dostupné z URL: <https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/06%3A_Kernel_Module_Management/1.03%3A_Linux_Kernel_Subsystem>

[8] Mohita Madaan. *Kernel and its types: Operating System.* [online]. Jún 24, 2022. [cit. 16. 10. 2022]. Dostupné z URL: <https://www.naukri.com/learning/articles/kernel-and-its-types-operating-system/>

[9] GeeksforGeeks. *Kernel in Operating System.* [online]. Marec 8, 2022. [cit. 16. 10. 2022]. Dostupné z URL: <https://www.geeksforgeeks.org/kernel-in-operating-system/>

[10] LibreTexts. *Kernel Modules.* [online]. Október 4, 2022. [cit. 20. 10. 2022]. Dostupné z URL: <https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/06%3A_Kernel_Module_Management/2.04%3A_Kernel_Modules>

[11] ROBERT, Love. *Linux Kernel Development*. 2nd ed. Indianapolis: Novell Press, 2005. ISBN 0-672-32720-1.

[12] Brandon Jones. *Everything about Daemons in Linux.* [online]. Máj 5, 2021. [cit. 20. 10. 2022]. Dostupné z URL: <https://www.fosslinux.com/46765/daemon-linux.htm>

[13] java T point. *Linux Daemon.* [online]. 2021. [cit. 20. 10. 2022]. Dostupné z URL: <https://www.javatpoint.com/linux-daemon>

[14] LibreTexts. *GUI Configuration.* [online]. Júl 20, 2022. [cit. 20. 10. 2022]. Dostupné z URL: <https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/08%3A_How_to_Manage_System_Components/2.04%3A_GUI_Configuration>

[15] Bobby Borisov. *Xorg, X11, Wayland? Linux Display Servers And Protocols Explained.* [online]. Apríl 2, 2022. [cit. 20. 10. 2022]. Dostupné z URL: <https://linuxiac.com/xorg-x11-wayland-linux-display-servers-and-protocols-explained/>

[16] LibreTexts. *GUI Desktop Environments.* [online]. Júl 22, 2022. [cit. 20. 10. 2022]. Dostupné z URL: <https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/08%3A_How_to_Manage_System_Components/2.05%3A_GUI_Desktop_Environments>

[17] Wayland. *Wayland Architecture.* [online]. 2022. [cit. 12. 11. 2022]. Dostupné z URL: <https://wayland.freedesktop.org/architecture.html>

[18] Elvis Plesky. *What is Linux? An in-depth introduction.* [online]. Február 4, 2019. [cit. 21. 10. 2022]. Dostupné z URL: <https://www.plesk.com/blog/various/what-is-linux/>

[19] Chris Hoffman. *What Is a Linux Distro, and How Are They Different from One Another?* [online]. September 23, 2016. [cit. 3. 11. 2022]. Dostupné z URL: <https://www.howtogeek.com/132624/htg-explains-whats-a-linux-distro-and-how-are-they-different/>

[20] Forrest Stroud. *Fedora Linux.* [online]. Júl 2, 2021. [cit. 3. 11. 2022]. Dostupné z URL: <https://www.webopedia.com/definitions/fedora/>

[21] CASTRO, Jose Dieguez. *Introducing Linux Distros: choose the right Linux distributions for you needs.* [New York, NY]: Apress, [2016]. ISBN 978-1-4842-1393-3.

[22] Chris Hoffman. *What Is AppArmor, and How Does It Keep Ubuntu Secure?* [online]. September 28, 2016. [cit. 3. 11. 2022]. Dostupné z URL: <https://www.howtogeek.com/118222/htg-explains-what-apparmor-is-and-how-it-secures-your-ubuntu-system/>

[23] Abhishek Prakash. *Using PPA in Ubuntu Linux [Complete Guide].* [online]. Máj 7, 2021. [cit. 27. 10. 2022]. Dostupné z URL: <https://itsfoss.com/ppa-guide/>

[24] GeeksforGeeks. *Using PPA in Linux* [online]. Jún 2, 2022. [cit. 29. 10. 2022]. Dostupné z URL: <https://www.geeksforgeeks.org/using-ppa-in-linux/>

[25] Arun KL. *What Are Single Board Computers (SBCs)? And, Why You Should Buy Single Board Computers?* [online]. 2022. [cit. 29. 10. 2022]. Dostupné z URL: <https://thesecmaster.com/what-are-single-board-computers-sbcs-and-why-you-should-buy-single\protect\@normalcr\relax-board-computers/>

[26] Moel Long. *What is a Single-Board Computer?* [online]. Február 10, 2021. [cit. 29. 10. 2022]. Dostupné z URL: <https://www.electromaker.io/blog/article/what-is-a-single-board-computer>

[27] UP. *UP Board Series* [online]. 2022. [cit. 29. 10. 2022]. Dostupné z URL: <https://up-board.org/up/specifications/>

[28] Raspberry Pi Foundation. *Raspberry Pi 4 Computer Model B* [online]. [cit. 1. 12. 2022]. Dostupné z URL: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf>

[29] The PiHut. *Raspberry Pi 4 Model B* [online]. 2022. [cit. 1. 12. 2022]. Dostupné z URL: <https://thepihut.com/products/raspberry-pi-4-model-b?src=raspberrypi&variant=41005997392067>

[30] Radxa. *Rock Pi 4 SE Product Brief* [online]. 2022. [cit. 1. 12. 2022]. Dostupné z URL: <https://www.gotronic.fr/pj2-fiche-technique-2613959-radxa-rs114se-r2d4w2s0ib-rock-4-se-4-gb-6-x-15-g
pdf>

[31] OKdo. *OKdo ROCK 4 Model C* [online]. 2022. [cit. 1. 12. 2022]. Dostupné z URL: <https://www.okdo.com/p/okdo-rock-4-model-c-4gb-single-board-computer-rockchip-rk3399-t-arm-cortex-a

[32] Whitson Gordon. *Asus Tinker Board 2S Review* [online]. Máj 21, 2021. [cit. 1. 12. 2022]. Dostupné z URL: <https://www.pcmag.com/reviews/asus-tinker-board-2s>

[33] Asus. *Tinker Board 2* [online]. 2022. [cit. 1. 12. 2022]. Dostupné z URL: <https://tinker-board.asus.com/product/tinker-board-2.html>

[34] Brien Posey. *IoT gateway* [online]. Marec 2022. [cit. 3. 11. 2022]. Dostupné z URL: <https://www.techtarget.com/iotagenda/definition/IoT-gateway>

[35] Sravani Bhattacharjee. *How Gateways Can Secure IoT Architectures* [online]. Apríl 9, 2020. [cit. 3. 11. 2022]. Dostupné z URL: <https://www.mouser.mx/blog/how-gateways-secure-iot-architectures>

[36] openHAB Community a openHAB Foundation. *Welcome to openHAB* [online]. 2022. [cit. 4. 11. 2022]. Dostupné z URL: <https://www.openhab.org/docs/>

[37] Paulus Schoutsen. *Welcome to openHAB* [online]. Máj 26, 2020. [cit. 5. 11. 2022]. Dostupné z URL: <https://www.home-assistant.io/blog/2020/05/26/installation-methods-and-community-guides-wiki/>

[38] Home Assistant. *Glossary* [online]. 2022. [cit. 5. 11. 2022]. Dostupné z URL: <https://www.home-assistant.io/docs/glossary/>

[39] Home Assistant. *Documentation* [online]. 2022. [cit. 5. 11. 2022]. Dostupné z URL: <https://www.home-assistant.io/docs/>

[40] Corrine Bernstein, Kate Brush, Alexander S. Gillis. *MQTT (MQ Telemetry Transport)* [online]. Január 2021. [cit. 10. 11. 2022]. Dostupné z URL: <https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport>

[41] The HiveMQ Team. *MQTT Topics, Wildcards, Best Practices - MQTT Essentials: Part 5* [online]. August 20, 2019. [cit. 10. 11. 2022]. Dostupné z URL: <https://www.hivemq.com/blog/>

[42] The HiveMQ Team. *HiveMQ - MQTT Security Fundamentals* [online]. Máj 11, 2015. [cit. 10. 11. 2022]. Dostupné z URL: <https://www.hivemq.com/mqtt-security-fundamentals/>

[43] OPC Router. *What is MQTT? A practical introduction* [online]. 2022. [cit. 10. 11. 2022]. Dostupné z URL: <https://www.opc-router.com/what-is-mqtt/>

[44] Steve Cope. *Introduction to MQTT-SN (MQTT for Sensor Networks)* [online]. 2023. [cit. 12. 3. 2023]. Dostupné z URL: <http://www.steves-internet-guide.com/mqtt-sn/>

[45] Bhagvan Kommadi. *Mosquitto: MQTT?* [online]. Február 4, 2020. [cit. 11. 11. 2022]. Dostupné z URL: <https://medium.com/@bhagvankommadi/mosquitto-mqtt-2a352bd8f179>

[46] Mosquitto, Eclipse Foundation. *An open source MQTT broker* [online]. 2022. [cit. 11. 11. 2022]. Dostupné z URL: <https://mosquitto.org/>

[47] Sofija Simic. *What is Docker?* [online]. September 16, 2021. [cit. 12. 11. 2022]. Dostupné z URL: <https://phoenixnap.com/kb/what-is-docker>

[48] Farhan Hasin Chowdhury. [online]. Február 1, 2021. [cit. 12. 11. 2022]. Dostupné z URL: <https://www.freecodecamp.org/news/the-docker-handbook/#what-is-a-container>

[49] Eric Goebelbecker. *Docker: How to List Every Container and More.* [online]. Február 8, 2022. [cit. 12. 11. 2022]. Dostupné z URL: <https://www.cloudbees.com/blog/docker-how-to-list-every-container-and-more>

[50] Alexander S. Gillis. *What is a Docker image?* [online]. Máj 2022. [cit. 12. 11. 2022]. Dostupné z URL: <https://www.techtarget.com/searchitoperations/definition/Docker-image>

[51] WebSupport. *Docker – 2. Dockerfiles a images.* [online]. November 26, 2020. [cit. 17. 11. 2022]. Dostupné z URL: <https://www.websupport.sk/podpora/kb/docker-dockerfiles-a-images/>

[52] Docker Inc. *Best practices for writing Dockerfiles.* [online]. 2022. [cit. 17. 11. 2022]. Dostupné z URL: <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/>

[53] WebSupport. *Docker – 5. Compose.* [online]. November 6, 2020. [cit. 18. 11. 2022]. Dostupné z URL: <https://www.websupport.sk/podpora/kb/docker-compose/>

Zoznam symbolov a skratiek

AI	Artificial Intelligence
ARM	Advanced RISC Machine
CPU	Central Processing Unit
DNS	Domain Name System
DoS	Denial of Service
eMMC	embedded MultiMediaCard
POST	Power On Self Test
MBR	Master boot record
GPU	Graphics Processing Unit
RAM	Random Access Memory
GB	Gigabyte
GRUB	GRand Unified Bootloader
GUI	Graphical User Interface
IoT	Internet of Things
LILO	Linux Loader
LOADLIN	Load Linux
LTS	Long Term Support
MITM	Man-in-the-middle
OS	Operating system
QoS	Quality of Service
SBC	Single-board Computer
SCP	Secure Copy Protocol
SSH	Secure Shell Protocol
TLS	Transport Layer Security

VFS Virtual File System

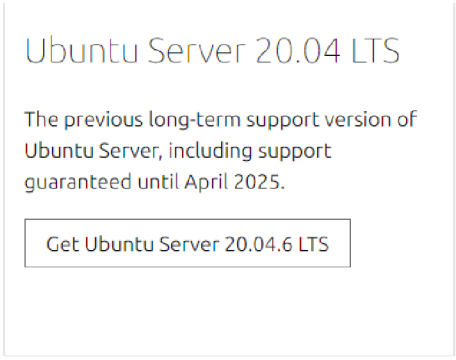
VM Virtual Machine

VPN Virtual Private Network

Zoznam príloh

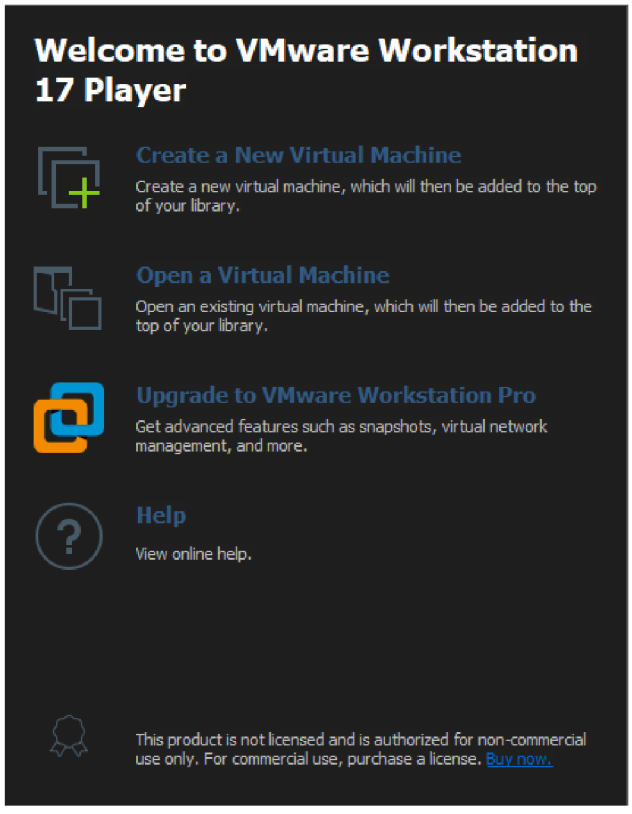
A Inštalácia Ubuntu server 20.04 vo VMware Workstation

1. Pred začiatkom inštalácie je potrebné stiahnuť Ubuntu server vo verzii 20.04 zo stránky `https://ubuntu.com/download/server#downloads`.



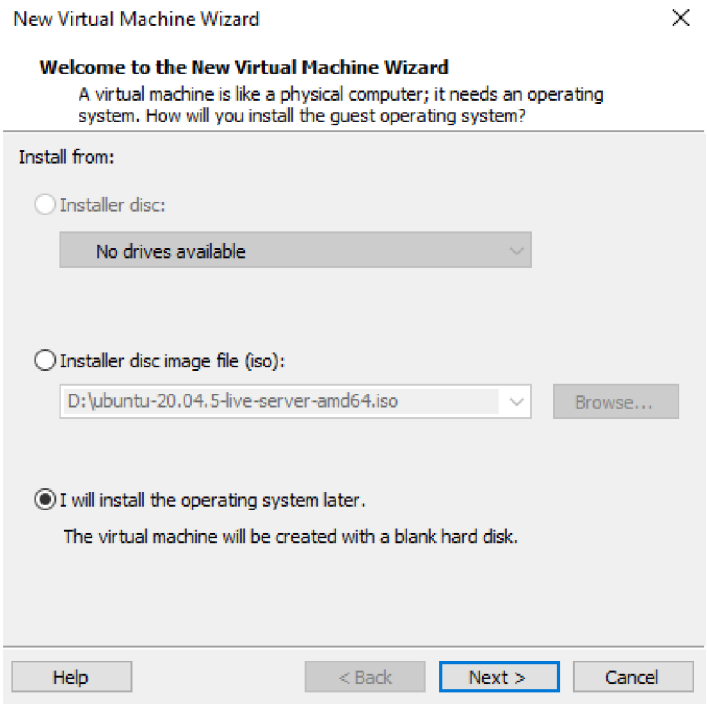
Obr. A.1: Krok 1.

2. Vo WMware zvolíme „Create a New VM“.



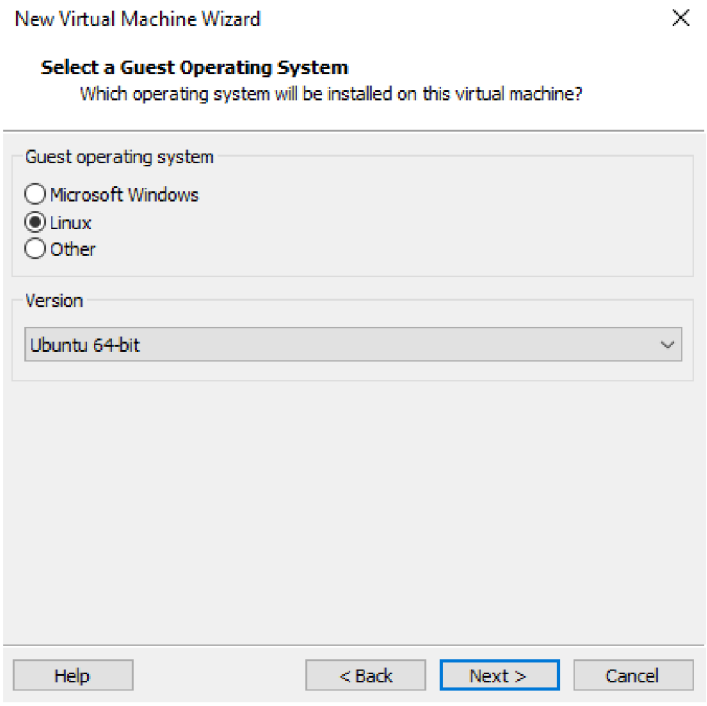
Obr. A.2: Krok 2.

3. Je zaznačená možnosť „I will install the OS later“ a pokračuje sa s „Next“.



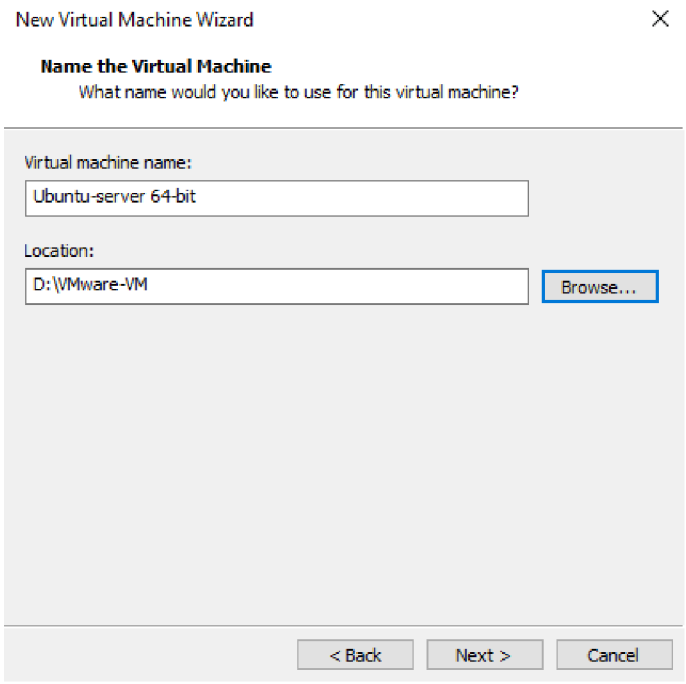
Obr. A.3: Krok 3.

4. Ako operačný systém je zvolený „Linux“, verzia „Ubuntu 64-bit“.



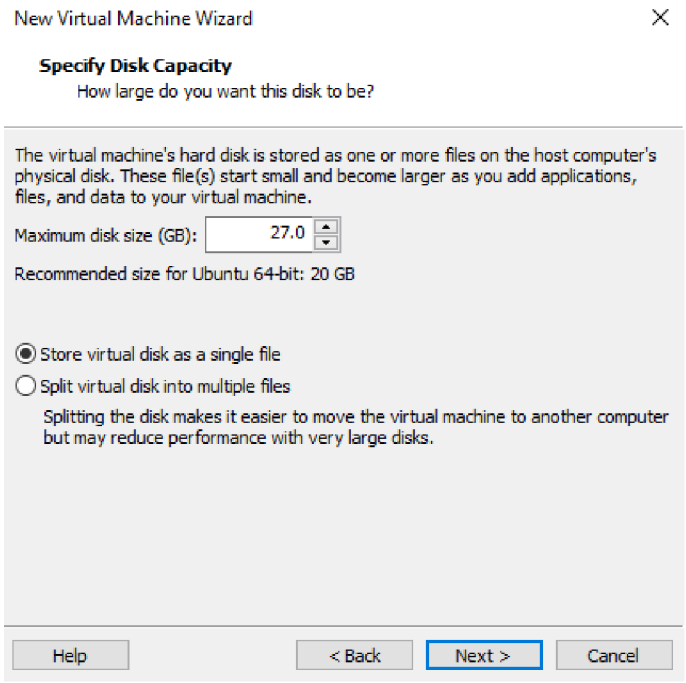
Obr. A.4: Krok 4.

5. V tomto kroku je pomenovaný virtuálny stroj a je vybrané umiestnenie uloženia.



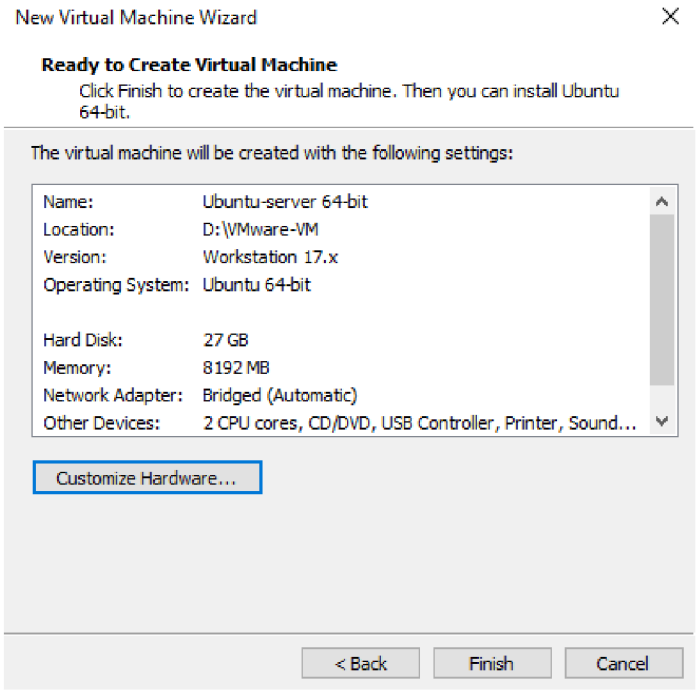
Obr. A.5: Krok 5.

6. Je zvolená potrebná veľkosť disku. Doporučených je minimálne 2.5GB.



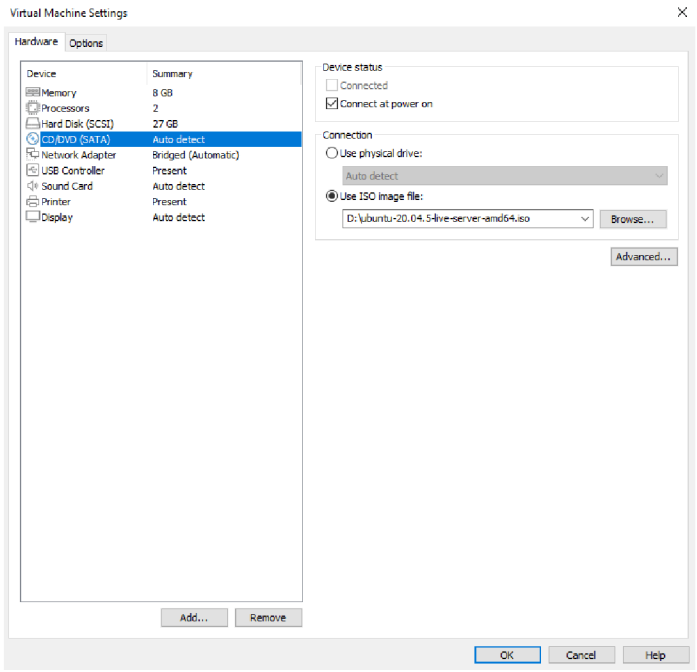
Obr. A.6: Krok 6.

7. V tomto kroku je možné prispôsobiť hardvérové požiadavky systému.



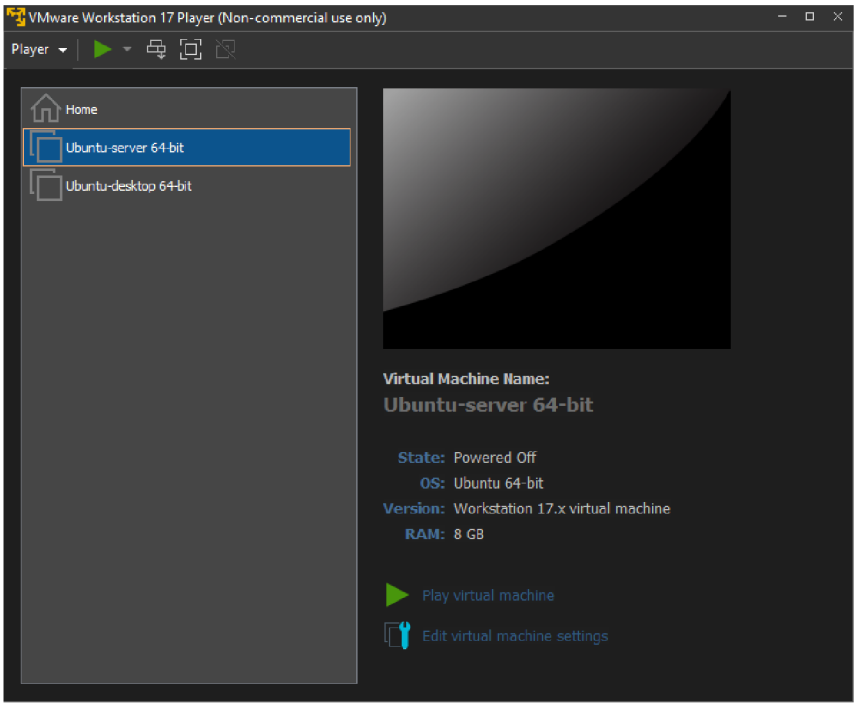
Obr. A.7: Krok 7.

8. Na pravej strane je zvolená voľba „Use ISO image file“ a je vybraný obraz disku na inštaláciu.



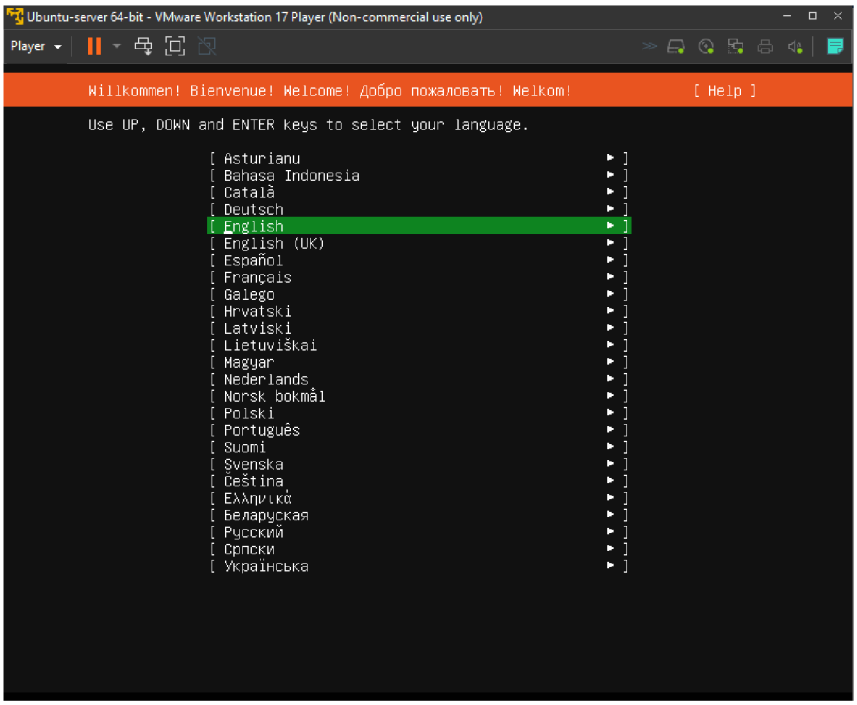
Obr. A.8: Krok 8.

9. Zvolíme „Ubuntu-server 64-bit“ a zeleným tlačítkom je spustená inštalácia.



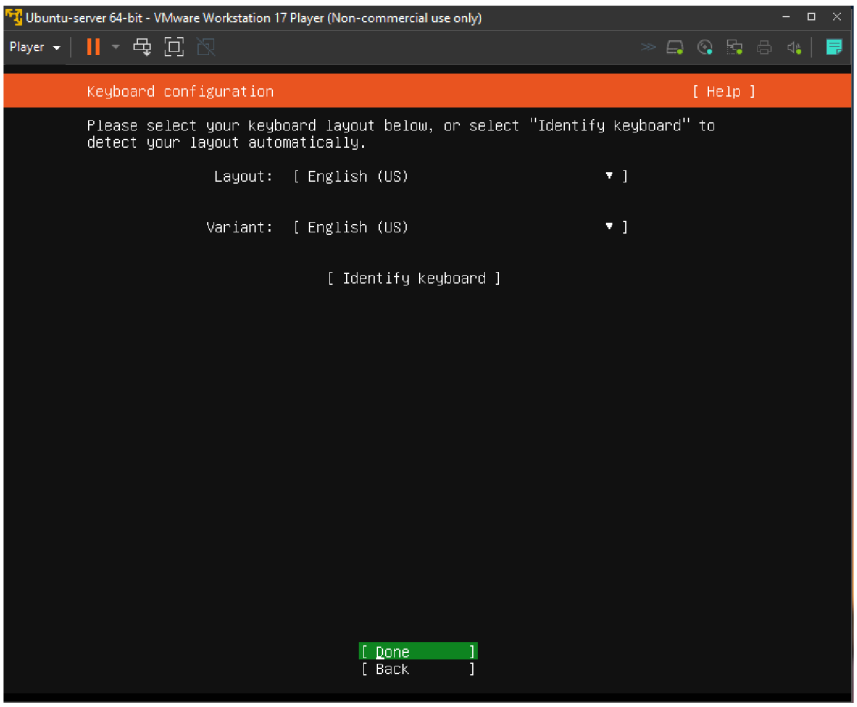
Obr. A.9: Krok 9.

10. Voľba predvoleného jazyka systému (English).



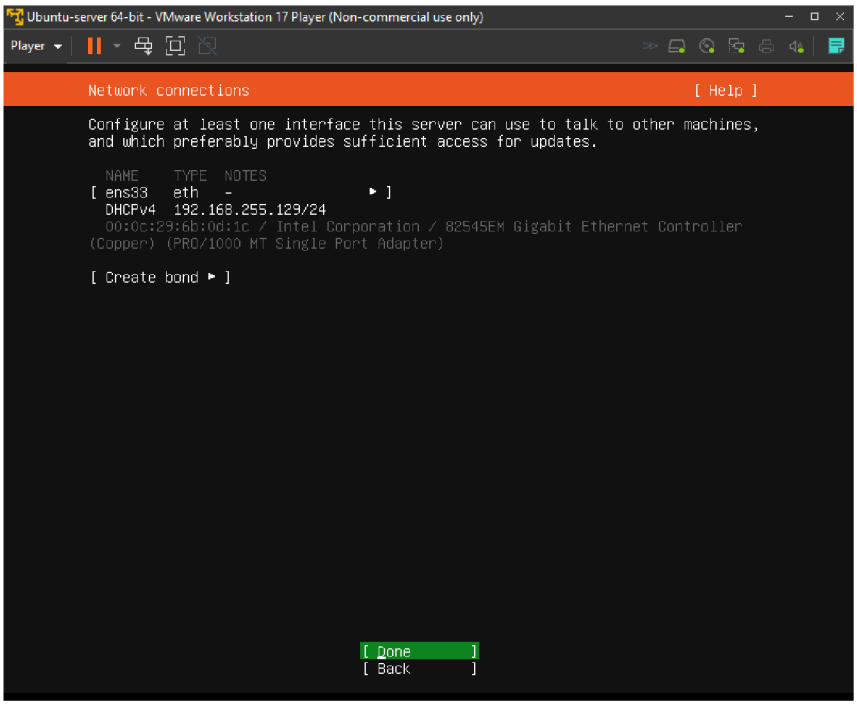
Obr. A.10: Krok 10.

11. Voľba jazyka klávesnice (English US).



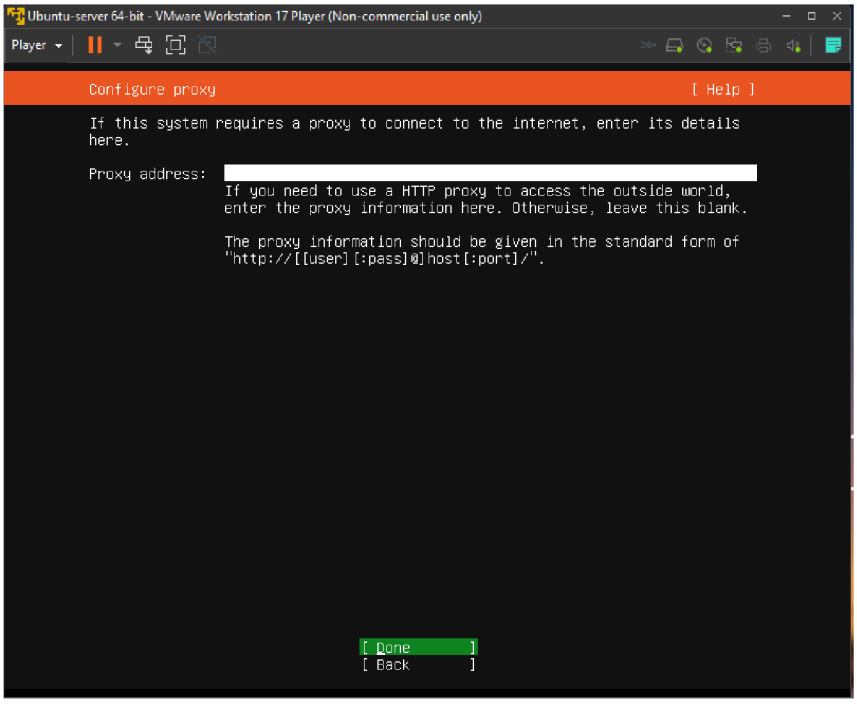
Obr. A.11: Krok 11.

12. Po pridelení IPv4 adresy DHCP serverom klikneme na „Done“.



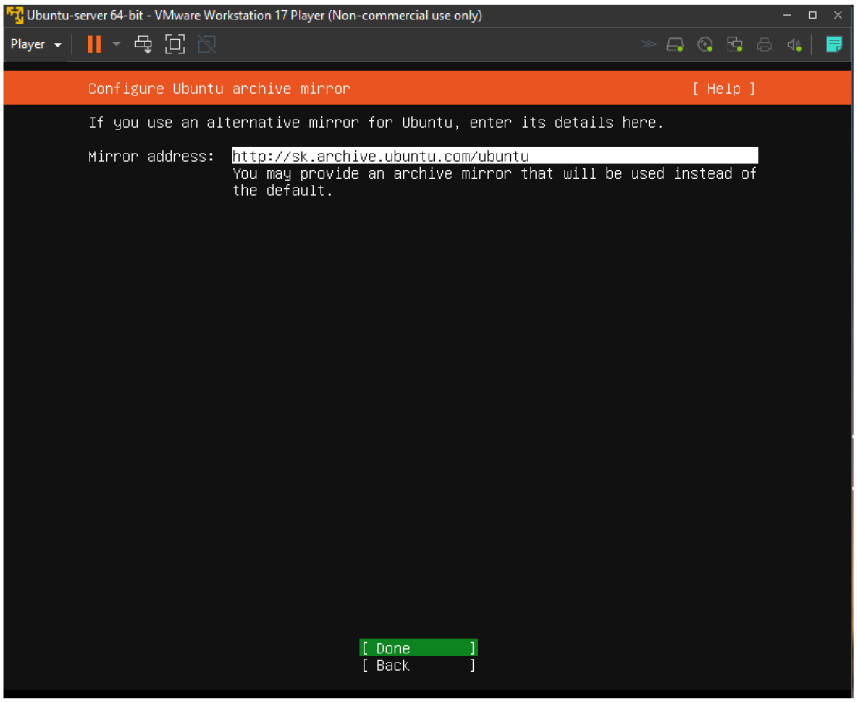
Obr. A.12: Krok 12.

13. Adresu proxy je možné v tomto prípade vynechať, klikneme na „Done“.



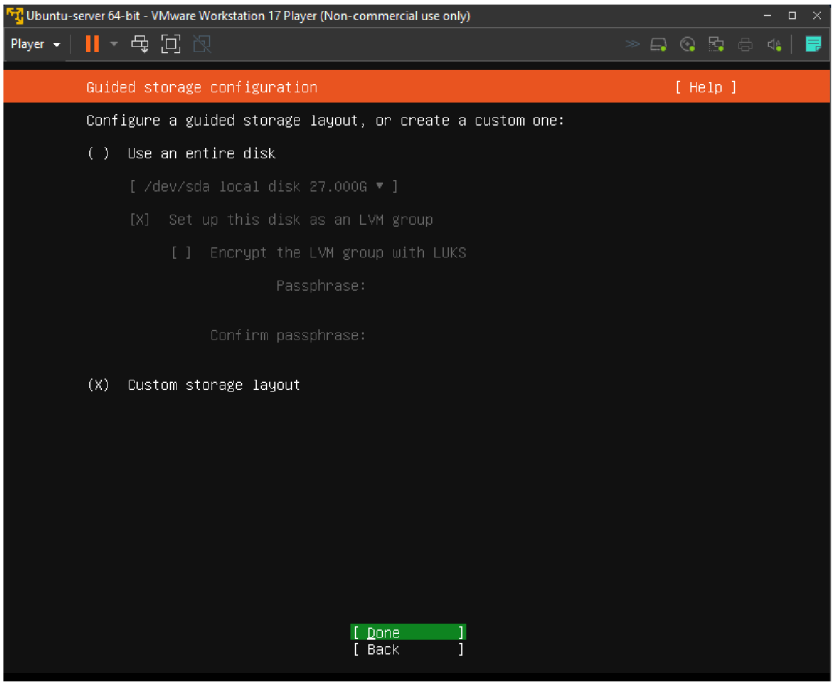
Obr. A.13: Krok 13.

14. Adresy zrkadla je možno ponechať na predvolenom, klikneme na „Done“.



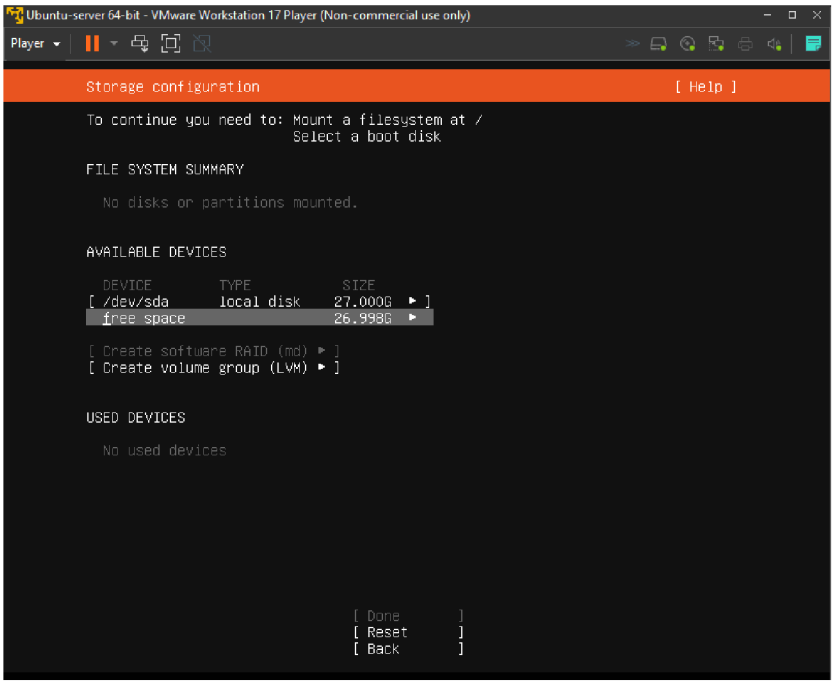
Obr. A.14: Krok 14.

15. Je zvolená možnosť „Custom storage layout“, ktorá nám umožní vlastné rozvrhnutie veľkostí partícii.



Obr. A.15: Krok 15.

16. Po zvolení „free space“ je vybraná možnosť „Add GPT partition“.



Obr. A.16: Krok 16.

17. Ako prvá partícia je `sda2` o veľkosti 512MB. Jediným účelom tejto partície je uchovanie vlastného (redukovaného) zavádzača. Pri tomto kroku je dôležité, aby bola dodržaná jej veľkosť, aspoň 512MB a taktiež sa za miesto pripojenia vyberá „Leave unmounted“.

Obr. A.17: Krok 17.

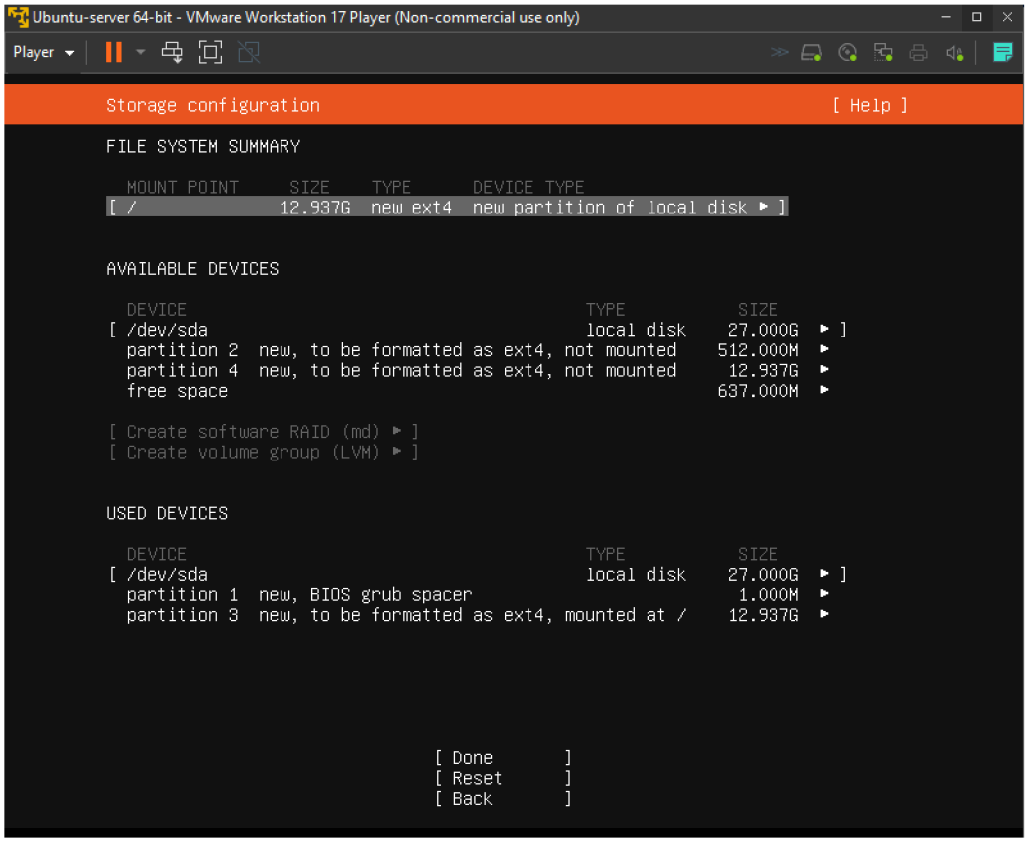
18. Druhá partícia `sda3` je systémová, na tejto partícii bude bežať primárny operačný systém s aplikáciami. Na jej veľkosti nezáleží, avšak je potrebné myslieť na fakt, že záložná partícia musí mať rovnakú veľkosť ako primárna. Ako prípojný bod je zvolený `/` (root).

Obr. A.18: Krok 18.

19. Tretia partícia `sda4` je záložná. Jej veľkosť musí korešpondovať s `/` (root) partíciou. Ponecháva sa ako nepripojená.

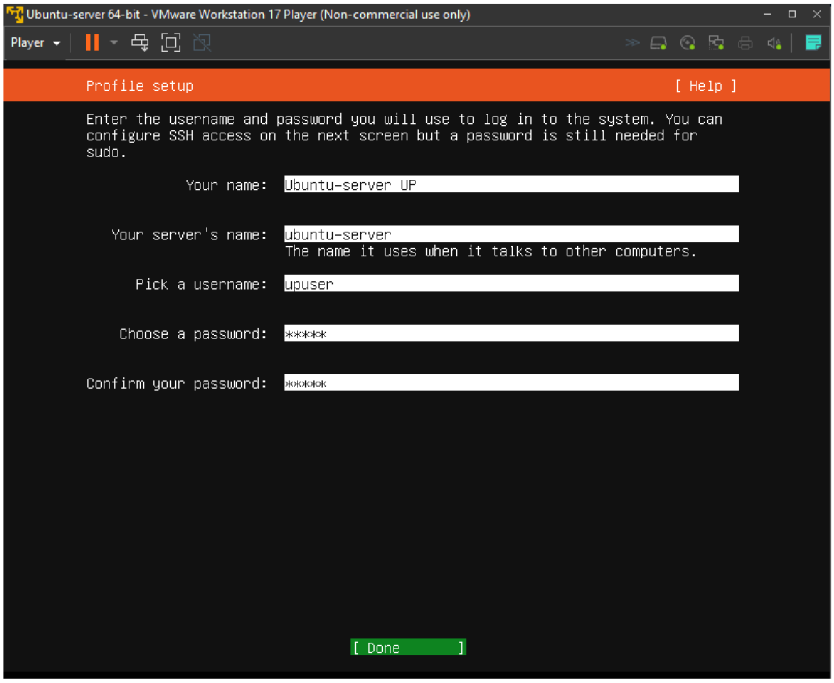
Obr. A.19: Krok 19.

20. Ukážka vytvorených partícii. Po správnom prevedení vyššie popísaných krokov je možné kliknúť na „Done“.



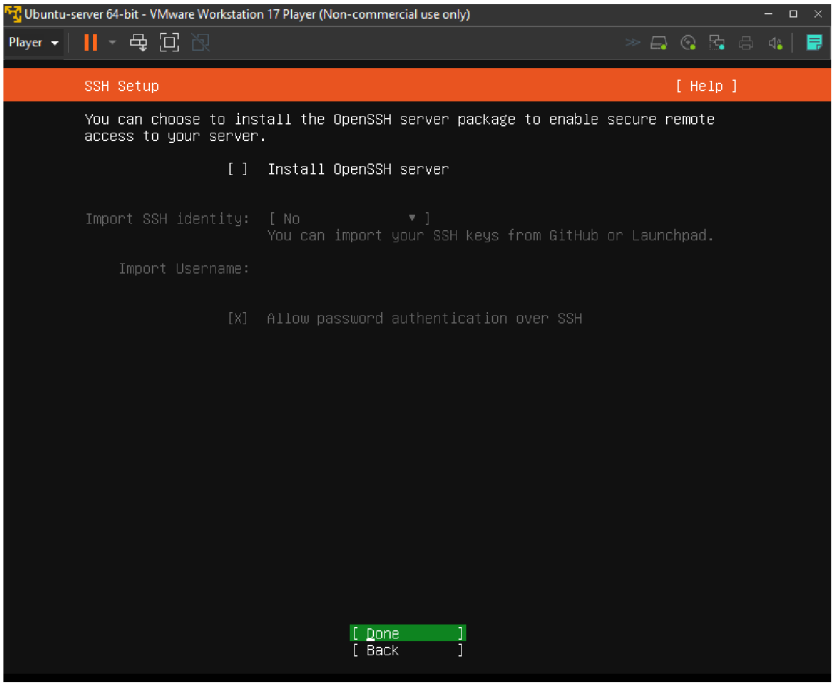
Obr. A.20: Krok 20.

21. Pri nastavení profilu je zvolené užívateľské meno a heslo.



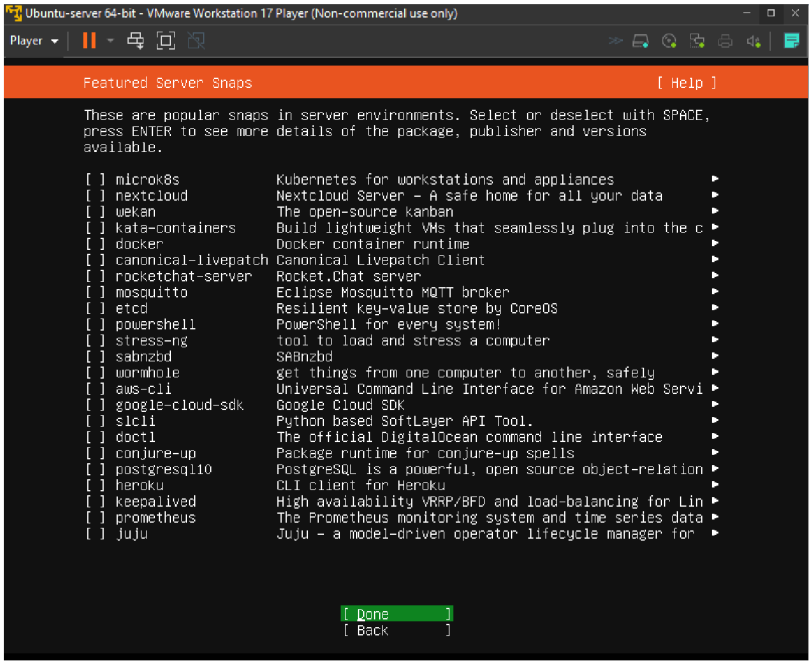
Obr. A.21: Krok 21.

22. Pokračuje sa bez inštalácie SSH servera, len kliknutím na „Done“.



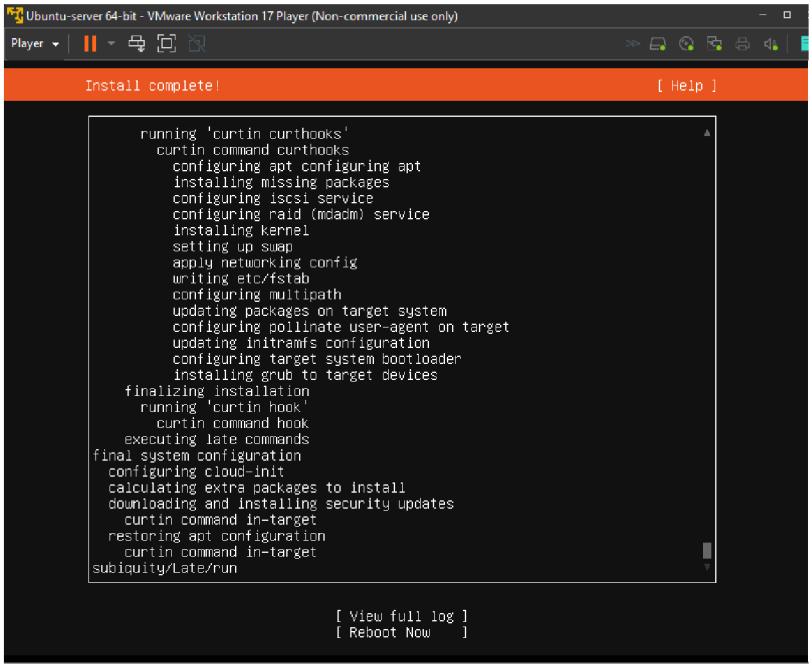
Obr. A.22: Krok 22.

23. Inštalácia propagovaných serverových balíčkov nie je v nami vytváranom systéme potrebná. Kliknutím na „Done“ je spustená inštalácia systému.



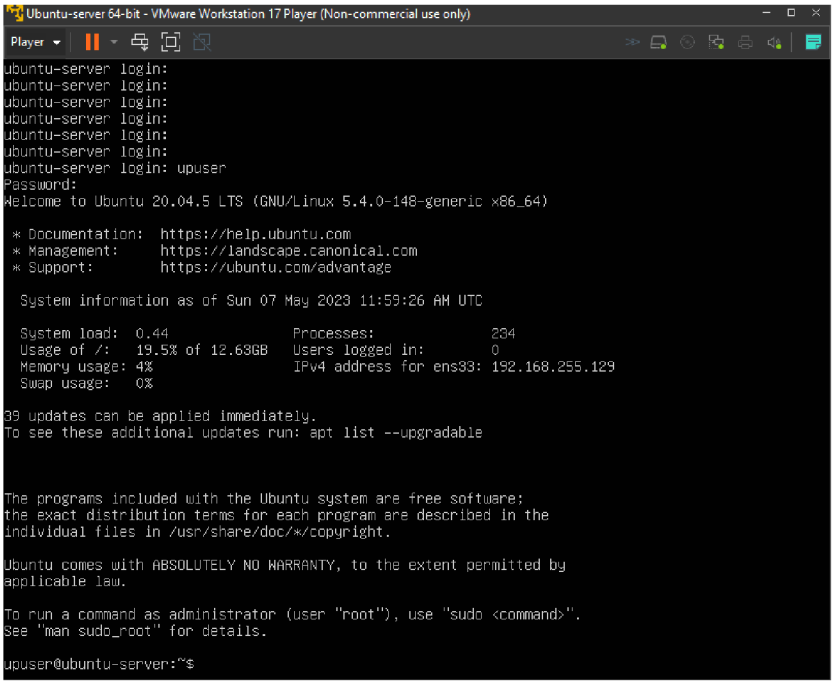
Obr. A.23: Krok 23.

24. Po skončení inštalácie je zvolená možnosť „Reboot Now“.



Obr. A.24: Krok 24.

25. Po spustení systému je zadané užívateľské meno a heslo a systém je pripravený k užitiu.



Obr. A.25: Krok 25.