VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Tomáš Bičák

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

JEDNOÚČELOVÝ VOIP KOMUNIKÁTOR
DEDICATED VOIP COMMUNICATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Bičák**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Caha**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Tomáš Bičák ***ID:*** 155737
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Jednoúčelový VoIP komunikátor

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s technologií VoIP a protokolem SIP. Vytvořte prototyp zařízení, které po stisknutí tlačítka naváže hlasové spojení s určitým účastníkem. Pro VoIP komunikaci využijte vybraného poskytovatele (např. sip2sip.info). Vytvořený kód vystavte pod licencí MIT na GitHub.

DOPORUČENÁ LITERATURA:

[1] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[2] PILGRIM, M. Ponořme se do Python(u) 3. CZ.NIC, 2010. 435 s. ISBN: 978-80-904248-2-1.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 18.8.2020

Vedoucí práce: Ing. Tomáš Caha

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Cílem bakalářské práce bylo sestrojit VoIP komunikátor. Tento VoIP komunikátor je postaven na zařízení Raspberry Pi Zero s externí zvukovou kartou. Na zařízení Raspberry Pi je nainstalován operační systém Raspbian. Jako softwarový základ pro VoIP komunikaci je použita knihovna PJSIP. Vytvořené programové vybavení umožňuje automaticky přijmout hovor a vytvořit hovor na definované adresy. Hovor je uskutečněn po stisku tlačítka.

KLÍČOVÁ SLOVA

SIP, Raspberry Pi Zero WH, ReSpeaker 2 Mics Pi HAT, VoIP, PJSIP

ABSTRACT

The aim of the bachelor thesis was to build a VoIP communicator. This VoIP communicator is built on a Raspberry Pi Zero device with an external sound card. Raspbian operating system is installed on the Raspberry Pi. The PJSIP library is used as the software basis for VoIP communication. The created software allows you to automatically answer a call and make a call to defined addresses. The call is made after pressing the button.

KEYWORDS

SIP, Raspberry Pi Zero WH, ReSpeaker 2 Mics Pi HAT, VoIP, PJSIP

BIČÁK, Tomáš. *Jednoduchý VoIP komunikátor*. Brno, 2020, 59 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Tomáš Caha

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Jednoduchý VoIP komunikátor“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

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

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato bakalářská práce se věnuje sestavením jednoduchého VoIP komunikátoru, na jednodeskovém počítači je nainstalován program na VoIP komunikaci. Tento program je ovládán pomocí skriptů napsaných v programovacím jazyce Python.
Raspberry Pi je osazeno speciálním zařízením, které slouží jako zvuková karta a obsahuje tlačítko. Pomocí tlačítka se ovládá celé zařízení.

Tato bakalářská práce navazuje na semestrální projekt, kde se hledaly a zkoušely programy na VoIP (Voice over Internet Protocol) technologie na zařízení Raspberry Pi Zero WH.

Technologie VoIP je technologie, která umožňuje přenos digitalizovaného hlasu prostřednictvím počítačové sítě. Technologie VoIP využívá sadu protokolů a kodeků ke komunikaci. K navázání a ukončení hovorů ve VoIP se používá protokol SIP (Session Initiation Protocol).

K samotnému testování VoIP komunikátoru je použito zařízení Raspberry Pi Zero, které vyvinula britská společnost Raspberry Pi Foundation v roce 2017.
Raspberry Pi je nejrozšířenější jednodeskový počítač na světě. Je využíván jak pro studijní účely, tak i v domácnostech na jednoduchou automatizaci nebo jako levný počítačový server na kterém jsou spuštěny potřebné služby.

Na Raspberry Pi je použit programovací jazyk Python. Programovací jazyk Python je vysokoúrovňový skriptovací programovací jazyk, který byl vyvinut v roce 1991. Nabízí dynamickou kontrolu datových typů a podporuje různé programovací styly včetně objektově orientovaného, imperativního, procedurálního nebo funkcionálního stylu. Python byl vyvíjen jako open source projekt, který nabízí zdarma instalační balíky pro většinu běžných operačních systémů.

V 1. kapitole bakalářské práce je popsána technologie VoIP a zařízení Raspberry Pi Zero WH s možnostmi připojení externích modulů. Tato kapitola se nachází na straně 13. V 2. kapitole je popsána příprava zařízení. Je zde popsána instalace operačního systému, instalace rozšiřující zvukové karty a kompilace programového balíčku PJSIP. Tato kapitola se nachází na straně 24. Kapitola 3 je zaměřena na návrh obslužného programu. Zde jsou popsány vytvořené skripty. Tato kapitola se nachází na straně 32. Kapitola 4 je zaměřena na ověření funkčnosti a možného využití daného zařízení. Předposlední kapitola se nachází na straně 40. Kapitola 4 je zaměřena na zhodnocení celé práce. Závěr práce se nachází na straně 42. Zdrojové kódy se nachází na straně 49.

1 Použité technologie software a hardware

Tato část se zabývá teoretickým popisem všech částí zařízení na kterém je bakalářská práce realizována. Nejprve je zde popsána technologie VoIP a protokol SIP. Následně je zde popsáno zařízení Raspberry Pi.

1.1 Technologie Voice over Internet Protocol

VoIP je technologie, která umožňuje přenos digitalizovaného hlasu pomocí paketů protokolů UDP (User Datagram Protocol), TCP (Transmission Control Protocol) nebo IP (Internet Protocol). Využívá se pro telefonování prostřednictvím Internetu, intranetu nebo jiného datového spojení. K přenosu multimediálních dat využívá sadu protokolů. Mezi tyto protokoly patří signalizační protokoly (SIP) a protokoly zajišťující přenos multimediálních dat (RTP, SRTP) [13].

Mezi nejvíce používané signalizační protokoly patří SIP, H.323 nebo IAX (Inter-Asterisk eXchange). Pro přenos multimediálních dat se používá protokol RTP nebo zabezpečená verze SRTP.

Důležitou součástí VoIP jsou takzvané pobočkové ústředny, koncová zařízení zpravidla VoIP telefony, ale může se v těchto sítích objevit i takzvaná Gateway (brána). Gateway slouží v těchto sítích na propojení IP telefonie a ISDN (Integrated Services Digital Network) sítě. Pobočkové ústředny se starají o registraci uživatelů a zprostředkování hovorů.

Na trhu existuje celá řada výrobců hardwarových ústředen, ale i softwarových ústředen. Poslední součástí jsou koncová zařízení, zpravidla to bývá hardwarový telefon, ale může to být i softwarový telefon (aplikace) nainstalovaný na PC nebo na chytrém mobilním zařízení.

Protokol UDP (User Datagram Protocol) patří do protokolů TCP/IP a je alternativou protokolu TCP. Tento protokol poskytuje nespojitou službu, kde není potřeba před samotnou komunikací vytvořit cestu mezi účastníky. Je zde riziko, že se data po cestě mezi účastníky ztratí. Tento protokol se hodí na použití u technologie VoIP lépe než protokol IP [13].

Session Intiation Protocol (SIP) je signalizační protokol pro sestavení, dohled a rozpad obecných relací (point-to-point nebo multipoint). Je nezávislý na přenášeném obsahu relací. Prostřednictvím protokolu RTP umožňuje přenos multimediálních dat, zpráv a telefonních hovorů.

SIP protokol je poměrně jednoduchý a podobá se HTTP (Hypertext Transfer Protocol) relacím založených na zprávách request/response. Navíc jde o lidsky čitelný text-based formát dat. Podrobnosti o přenášeném obsahu (čísla portů, kodeky,

protokoly, šifrování) jsou komunikovány a vyjednávány protokolem SDP (Session Description Protocol).

Koncepce protokolu SIP přenáší komunikační schopnosti na koncová zařízení a sám protokol se soustředí výhradně na řízení komunikace, sestavování, dohled a rozpad relací, což je opačný přístup než mají tradiční telekomunikační sítě, kde koncová zařízení postrádají veškerou inteligenci. Jak název protokolu napovídá, jedná se o protokol relační, ale protože TCP/IP žádnou relační vrstvu nerozeznává, patří SIP mezi protokoly aplikační vrstvy a pracuje nad transportními protokoly UDP (většinou) či TCP [13].

Adresace v SIP

SIP URI mají různé formy (obecně sip:user@domain) a mohou obsahovat i telefonní čísla.

Podpora jak webové adresace, tak telefonních čísel umožňuje IP komunikaci bez větších problémů přecházet mezi telefonní sítí a Internetem. Uživatelé na kterékoli síti tak mohou komunikovat s kýmkoli na telefonní síti nebo na Internetu, aniž by museli vyměnit svá stávající zařízení. IP zařízení s podporou SIP (telefony, počítače) mohou komunikovat přímo, pokud znají URL (Uniform Resource Locator) druhé strany [12].

Signalizace

Zprávy protokolu SIP jsou dvojího druhu. A to žádost a odpověď. V tabulce 1.1 jsou nejčastější posílané žádosti v protokolu SIP.

Tab. 1.1: Odesílané žádosti protokolu SIP

Žádost	Popis
REGISTER	registrace účastníka na SIP Proxy serveru
INVITE	zahájení komunikace o plánované nové relaci
ACK	potvrzení zahájení relace
CANCEL	přerušení zahajovaní relace ještě před jejím navázáním
BYE	ukončení probíhající relace
OPTIONS	požádá o informace o možnostech vzdálené strany, aniž by se sestavilo volání

Odpovědi mají tvar trojmístného čísla. Tento návratový kód označuje výsledek žádosti obdobně, jak je tomu například v protokolu HTTP. Lze je rozdělit do šesti skupin podle první číslice. V tabulce 1.2 jsou tyto chyby popsány.

Tab. 1.2: Chybová hlášení protokolu SIP

Číslo chyby	Proces	Popis
1XX	průběh	průběh bez problémů, ale není ukončen přenos
2XX	úspěch	krok ukončen bez problémů
3XX	přesměrování	krok probíhá, ale ještě se v souvislosti s ním něco očekává
4XX	chyba klienta	požadavek je chybný a nemůže být serverem zpracován
5XX	chyba serveru	požadavek je zřejmě v pořádku, ale chyba je na straně serveru
6XX	fatální chyba	zcela fatální chyba, kterou nelze jakkoliv zpracovat

Nejběžnější odpovědi jsou „200 OK“ (úspěšné provedení žádosti), „302 Moved Temporarily“ (běžné přesměrování) a „404 Not Found“ (volaný uživatel se nenachází na daném serveru).

Registrace

Registrace uživatele se provádí odesláním žádosti REGISTER na UAS (User Agent Server). Žádost REGISTER obsahuje v hlavičce protokolu pole *TO:* kde je obsažena identita uživatele, respektive SIP URI registrujícího se uživatele. Pole To: společně s polem Contact: vytvoří záznam v registračním serveru, který se následně předán lokalizační službě. Při správné registraci se odesílá potvrzení `200 OK`, to znamená, že registrace proběhla úspěšně. [12]

Navázání a ukončení hovoru

Navázání spojení hovoru se nazývá „three-way handshake“. Klient A, který se chce spojit s klientem B, vyšle požadavek `INVITE`, kde je obsažen popis nabízeného spojení. Pokud klientovi B přijde zpráva v pořádku, tak odešle zprávu `100`. Tato odpověď je chápána jako vyzváněcí tón. Když klient B přijme hovor, tak se odešle zpráva `200 OK`. Tato zpráva obsahuje IP adresu, čísla portů a kodeky, na kterých bude klient očekávat data. V posledním kroku navazování komunikace odešle klient A klientovi B zprávu `ACK`, která potvrzuje přijetí odpovědi od klienta B. Po přijetí zprávy `ACK` začíná přenos audia, popřípadě videa.

Pro ukončení hovoru pošle jeden z účastníků protistraně požadavek `BYE`, který slouží na ukončení hovoru. Protistrana odpoví potvrzovací zprávou `200 OK` a hovor je tímto ukončen.

Spojení obvykle probíhá za účasti SIP serverů, kdy uživatel nemusí znát adresu volaného. Klient A pošle požadavek `INVITE`, který se dotazuje SIP serveru, ke kterému klient B náleží. SIP server je obvykle spárován s databází kontaktů. Jakmile

server zjistí aktuální adresu a polohu volaného klienta, tak tuto adresu posílá zpátky klientu A. Klient A pak rozhodne, zda naváže spojení s klientem B nebo ne. Pokud je SIP server v módu *PROXY*, tak upraví zprávu `INVITE` a pošle ji klientovi B. Klient B odešle zprávu `200 OK` na SIP server, odkud mu přišla žádost INVITE. Zpráva `200 OK` je po přijetí na SIP server přeposílána na klienta A. Po přijetí zprávy `200 OK` klientem A se navazuje spojení mezi volanými účastníky.

SIP ústředna

Je zařízení, na kterém je uloženo nastavení zařízení a profily uživatelů. Tyto ústředny mohou být v lokální síti nebo na internetu. Internetové ústředny jsou brány jako služba, kterou poskytuje nějaká společnost. Tyto služby jsou buď volně dostupné a nemusí se za ně platit, nebo jsou placené [13] .

V lokální síti se nejběžněji používá Linuxový server, na který je nahrána SIP ústředna. Nejčastěji se používá SIP ústředna od společnosti Asterisk, která je volně šiřitelná a nejsou za ni účtovány poplatky. Mezi její výhody patří možnost bohatého nastavení a přizpůsobení potřebám daného zákazníka. Dále je tu možnost ukládání výpisů hovorů a monitorování hovorů na svém vlastním hardware. V případě telefonování v lokální síti není potřeba připojení k internetu. Mezi nevýhody tohoto systému se řadí pořizovací cena hardwaru, nutnost provést vlastní nastavení a zálohovat si systém. Je tu nutnost řešit aktualizace a zabezpečení celého systému.

Na rozdíl od lokálních ústředen jsou internetové ústředny dobře zabezpečené. Uživatel se nemusí starat o zabezpečení a nemusí se také starat o aktualizace systému nebo zálohování systému. Internetoví poskytovatelé nabízejí tyto ústředny řešené přímo na míru zákazníka. Nevýhodou je nutnost stálého připojení k internetu. Bez připojení k internetu nebude fungovat telefonie ani v lokální síti.

Ústředna `SIP2SIP.info`

`SIP2SIP.info` je projekt od společnosti AG-Project. Je to komunikační služba v reálném čase pro audio, video, chat, přenos souborů a vícestranné konference založené na signalizaci SIP a souvisejících protokolech (RTP, MSRP (Message Session Relay Protocol) a XCAP (XML Configuration Acces Protocol)). Služba je otevřena a koresponduje s externími doménami SIP a XMPP (Extensible Messaging and Presence Protocol). Celá správa a registrace účtu se provádí přes internetové stránky `SIP2SIP.info` [16].

Pro registraci je nutno vyplnit jméno účtu, heslo, e-mail a údaje o uživateli přesněji jeho jméno a příjmení. Po registraci se odešle potvrzovací e-mail na zadanou adresu. Po obdržení potvrzovacího e-mailu se už stačí jen přihlásit k vytvořenému účtu.

Správa účtu se provádí přes webové stránky, které jsou uživatelsky přívětivé a dá se v nich rychle orientovat. Po přihlášení na stránkách `SIP2SIP.info` se objeví všechny potřebné informace. Dále jsou zde informace o zařízení, na kterém je daný uživatel přihlášen, je tu i možnost náhledu do historie volání, přesměrování hovorů, DNS (Domain Name System), přidávání nebo odebírání kontaktů. Dále se dá nastavit automatický příjem hovorů.

1.2 Zařízení Raspberry Pi

Raspberry Pi bylo vyvinuto britskou společností Raspberry Pi Foundation v roce 2012 s cílem podpořit výuku informatiky na školách a seznámit studenty s tím, jak mohou počítače ovládat různá zařízení. Raspberry Pi je název malého jednodeskového počítače. Existují i jiné jednodeskové počítače, které vychází z Raspberry Pi, jsou to například Banana Pi, Khadas, Odroid a další. Dále je v textu Raspberry Pi nahrazeno jeho zkratkou RPi.

Někdy se RPi přirovnává k Arduinu, což je také jednodeskový počítač, který je založen na procesorech ATmega od společnosti Atmel.

Na trhu existuje více variant RPi, které se od sebe liší velikostí a použitým procesorem. Na RPi je použit mikroprocesor z rodiny ARM (Ashton Raggatt McDougall). Výkon RPi v dnešní době dosahuje výkonu srovnatelného s dnešními mobilními telefony. Lze říci, že nejmodernější RPi 4 dosahuje výkonu přenosného počítače. V tabulce 1.3 na straně 21 je porovnání všech vlastností minulých a současných verzí RPi [3].

1.2.1 Využití Raspberry Pi

Nejčastěji se Raspberry Pi používá jako základ pro automatizaci například domácností, nebo se používá v počítačových sítích jako úložiště dat nebo jako server, na kterém běží různé služby (DHCP (Dynamic Host Configuration Protokol), DNS (Domain Name System), Pi-Hole (Blokování internetové inzerce) a atd.). RPi se dá v domácnosti využít jako multimediální centrum s operačním systémem LibreELEC. Jako automatizační jednotka se může využít na dálkové ovládání žaluzií, garážových vrat. Nebo může být připojen k 3D tiskárně a sloužit jako printserver. Lze RPi využít jako Web server, který pracuje v lokální síti nebo je připojen k internetu. Na internetu je mnoho projektů realizovaných pomocí RPi.

1.2.2 Raspberry Pi Zero

RPi Zero je nejmenší, nejlevnější a nejúspornější z RPi. Toto RPi se hodí k využití tam, kde je potřeba málo místa s velkým potenciálem. Zařízení je poháněno mikroprocesorem o taktu 1 GHz. Na trhu existují tři varianty RPi Zero a to v první variantě bez GPIO sběrnice a WiFi modulu. Druhá varianta obsahuje WiFi modul a neobsahuje také sběrnici GPIO. A poslední varianta obsahuje GPIO sběrnici a WiFi modul [4].

1.2.3 Raspberry Pi 3

RPi model 3 se na trhu objevuje ve variantě B a B+, která se od sebe liší v použité síťové kartě a ve výkonu procesoru. První na trh se dostala varianta RPi 3 B, která měla jako první 64bitový mikroprocesor s taktem 1,2 GHz a obsahovala WiFi kartu se standartem 802.11n. Novější model RPi 3 B+, tak má vyšší výkon procesoru (1,4 GHz), síťová karta využívá maximální rychlosti přenosu 1 Gb/s a WiFi karta podporuje standart 802.11ac. Oproti původní variantě je možné už připojit Raspberry Pi PoE HAT, který dokáže napájet RPi pomocí ethernetového portu [5].

1.2.4 Raspberry Pi 4

RPi 4 je nejnovější varianta Raspberry, která přišla na trh s novými použitými konektory. Změnil se konektor napájení, původně RPi využívalo microUSB konektor a nově je využíváno USB C. Nová verze má dva video výstupy. Změnil se video výstup, který byl konektor HDMI (High-Definition Multimedia Interface) na microHDMI. Původní verze měla 4 konektory USB ve standartu 2.0 u nové verze byly 2 konektory USB 2.0 nahrazeny standartem 3.0. Nově je nabízeno RPi ve variantách s různou velikostí operační paměti, a to o velikostech 2, 4 a 8 Gb. RPi 4 má oproti variantě 3 B+ větší takt mikroprocesoru (1,5 GHz) a podporuje maximální rozlišení video výstupu 4Kp60 [6].

1.2.5 Operační systém Raspberry Pi

Operační systém pro RPi je speciálně upravený operační systém pro architekturu ARM. Z tohoto důvodu nelze na RPi nainstalovat jiný operační systém. Operační systémy, které podporují architekturu ARM jsou optimalizovány tak, aby bylo dosaženo co nejvyššího výkonu a stability sytému. Systémy, které jsou schopny pracovat na RPi jsou na bázi Linuxu, ale lze i použít operační systém od společnosti Microsoft [3].

Raspbian

Raspberry Pi OS neboli Raspbian je operační systém založen na linuxové distribuci Debian. Systém byl vydán v roce 2012 s velkým množstvím doporučených balíčků, které měli za úkol usnadnit a zpříjemnit práci uživatele. Tento operační systém je nejvíce rozšířený mezi uživateli RPi pro svoji jednoduchost a přehlednost. Na oficiálních stránkách je více verzí tohoto operačního systému. Lze ho stáhnout ve verzi Lite, která neobsahuje grafické rozhraní a celé ovládání se provádí pomocí příkazového řádku. Tato verze je doporučována pro zkušenější uživatele. Dále je dostupná verze, která je s grafickým rozhraním a poslední verze je s doporučeným softwarem [3].

Ubuntu Mate

Ubuntu Mate je operační systém založen na uživatelsky nejrozšířenějším operačním systému Ubuntu. Tato verze operačního systému pro RPi je speciálně upravená, tak aby poskytovala uživateli kompletní a známé prostředí pro stolní počítače. Tento operační systém bohužel nejde spustit na zařízeních RPi ZERO a RPi 1 model A i B [7].

Kali Linux

Kali linux je linuxová distribuce odvozena od operačního systému Debian. Je navržen pro digitální forenzní analýzu a penetrační testy na prolomení hesel. Má více než 600 předinstalovaných programů na testování.. Tento operační systém byl primárně vyvinut na testování bezpečnosti počítačových sítí. Doporučuje se pro velmi zkušené uživatele.

Arch Linux

Arch Linux je primárně vyvíjen pro procesory s architekturou x86_64 a i386. Tento operační systém klade důraz na minimalistický a snadno přizpůsobitelný systém. Lze ho přizpůsobit k jakémkoli použití. Doporučuje se především pro zkušené uživatele.

OSMC

OSMC (Open Source Media Center) je bezplatný multimediální přehrávač založený na operačním systému Linux. Byl představen již v roce 2004 a umožňuje přehrávat videa z lokálního úložiště dat nebo z internetu. Tento operační systém je primárně určen pro zařízení Vero, ale dá se využít i na RPi. Je založen na multimediálním přehrávači KODI [8].

LibreELEC

LibreELEC je operační systém pro RPi, který z tohoto zařízení vytvoří multimediální centrum. Je postaven na multimediálním přehrávači KODI. Celý systém je dobře optimalizovaný a je možné ho ovládat pomocí dálkového ovladače od televize, pokud je RPi spojeno pomocí HDMI kabelu s televizí. Pro tento operační systém slouží rozšiřující balíčky [9].

PiNet

PiNet je operační systém, který je navržen pro školy a školící organizace k nastavení a správě sítí Raspberry Pi. Slouží k replikaci systémů, které existují pro sitě Microsoft. Všechen software je volně dostupný a je vytvořen pedagogy z celého světa [10].

RISC OS

Operační systém RISC OS není postaven na jádru operačního systému Linux nebo Unix. Považuje se za první operační systém pro architekturu ARM již v roce 1987. Tento operační systém s aplikacemi dosahoval velikosti 6 MB. Původně byl vyvinut pro mnohem pomalejší procesory a se spojením s RPi dosahuje tento systém rychlé odezvy na požadavky uživatele.

Windows 10 IOT

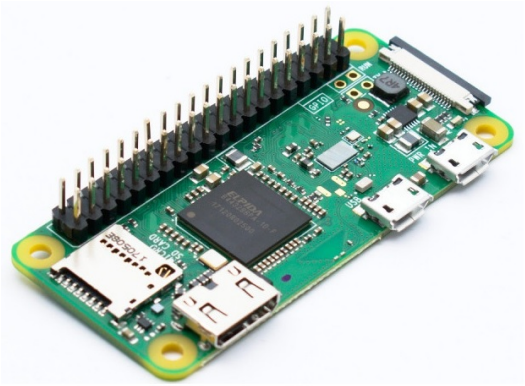
Windows 10 IOT je operační systém od společnosti Microsoft. Tento operační systém je určen pro jednoúčelové využití. Lze ho využít v bankomatech, herních automatech, tiskových zařízeních a ve zdravotních přístrojích, které sdílí informace pomocí cloudových služeb. Z operačního systému Windows 10 používá jen jádro systému. Je bez grafického rozhraní a tento operační systém není prodáván koncovým spotřebitelům, ale je prodáván primárně OEM (Original Equipment Manufacture) výrobcům, kteří jej následně upravují a prodávají koncovým uživatelům [11].

1.2.6 Rozšiřující moduly

Raspberry Pi je ve světě dost rozšířené a existují na něj různé rozšiřující moduly, které vytvořila společnost Raspberry Pi Foundation. Na trhu existují moduly, které vyvíjí různé společnosti. Na RPi je možné dokoupit různé moduly, které rozšíří daná zařízení o další možnosti využití. Lze dokoupit různé dotykové obrazovky různých velikostí jak od společnosti Raspberry, tak i od ostatních výrobců. Dále se dá dokoupit zvuková karta, tepelný snímač a automatizační karta, která obsahuje relé. Z RPi se dá vytvořit pomocí těchto rozšiřujících modulů i 3D tiskárna, nebo domácí multimediální centrum s televizní kartou.

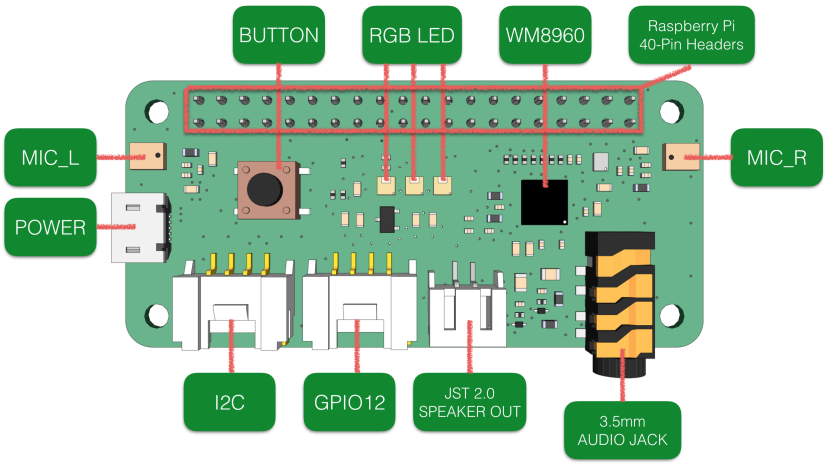
Tab. 1.3: Přehled všech verzí Raspberry Pi

Typ	Generace	Architektura	Procesor	SDRAM	USB	Síť	Výkon
Model A	1	ARMv6 (32-bit)	700 MHz single-core	256 MiB	1		300 mA
Model A	1+	ARMv6 (32-bit)	700 MHz single-core	512 MiB	1		200 mA
Model B	1	ARMv6 (32-bit)	700 MHz single-core	512 MiB	2	10/100 Mbit	700 mA
Model B	1+	ARMv6 (32-bit)	700 MHz single-core	512 MiB	4	10/100 Mbit	600 mA
Model B	2	ARMv7 (32-bit)	900 MHz 32-bit quad-core	1 GiB	4	10/100 Mbit	800 mA
Model B	3	ARMv8 (64/32-bit)	1,2 GHz 64-bit quad-core	1 GiB	4	10/100 Mbit, 802.11n	200 mA
Zero	PCB 1.3	ARMv6 (32-bit)	700 GHz single-core	512 MiB	1		160 mA
Zero	W	ARMv6 (32-bit)	1 GHz single-core	512 MiB	1	802.11n	160 mA
Model B	3+	ARMv8 (64/32-bit)	1,4 GHz 64-bit quad-core	1 GiB	4	10/100/1000 Mbit, 802.11ac	459 mA
Model B	4	ARMv8 (64-bit)	1,5 GHz 64-bit quad-core	1, 2, 4, 8 GiB	4	10/100/1000 Mbit, 802.11ac	3A



Obr. 1.1: Raspberry ZERO WH [4]

1.2.7 ReSpeaker 2 Mics Pi HAT



Obr. 1.2: Rozložení ReSpeaker 2 Mics Pi HAT [15]

ReSpeaker 2 Mics Pi HAT je rozšiřovací karta s dvěma mikrofony pro Raspberry Pi, která je určena pro AI a hlasové aplikace. Pomocí ní jsme schopni realizovat projekt, protože na desce Raspberry Pi Zero není žádný mikrofon nebo audio výstup. Deska je osazena čipem WM8960 a na stereo kodeku s nízkým výkonem. K dispozici jsou 2 mikrofony na obou stranách desky pro shromažďování zvuků. Dále obsahuje také 3 led diody, jedno tlačítko a pro výstup zvuku je zde zabudován 3,5mm audio jack nebo JST 2.0 Speaker Out. Deska dále obsahuje rozhraní Grove, pomocí kterého se dá rozšířit o další možnosti využití. Deska je napájena pomocí GPIO (General-purpose input/output) sběrnice přímo z Raspberry Pi, nebo se může napájet pomocí microUSB (Universal Serial Bus) konektoru, který je integrován na desce. Pomocí

tohoto konektoru se posléze napájí i samotné Raspberry Pi. Celkové rozložení je vyobrazeno na obrázku 1.2. Pro účely bakalářské práce se využívají mikrofony, tlačítko a audio výstup pomocí 3,5mm audio jacku [15].

2 Příprava realizačního prostředí

Tato část bakalářské práce je zaměřena na přípravu zařízení. První část této kapitoly je zaměřena na výběr zařízení a ústředny, na kterém bude tato bakalářská práce realizována. Druhá část kapitoly je zaměřena na instalaci operačního systému na paměťovou kartu. Předposlední část této kapitoly je zaměřena na instalaci a konfiguraci rozšiřujícího modulu Respeaker 2 Mics Pi HAT. Poslední část této kapitoly je zaměřena na stažení, kompilaci a ověření funkčnosti PJSIP.

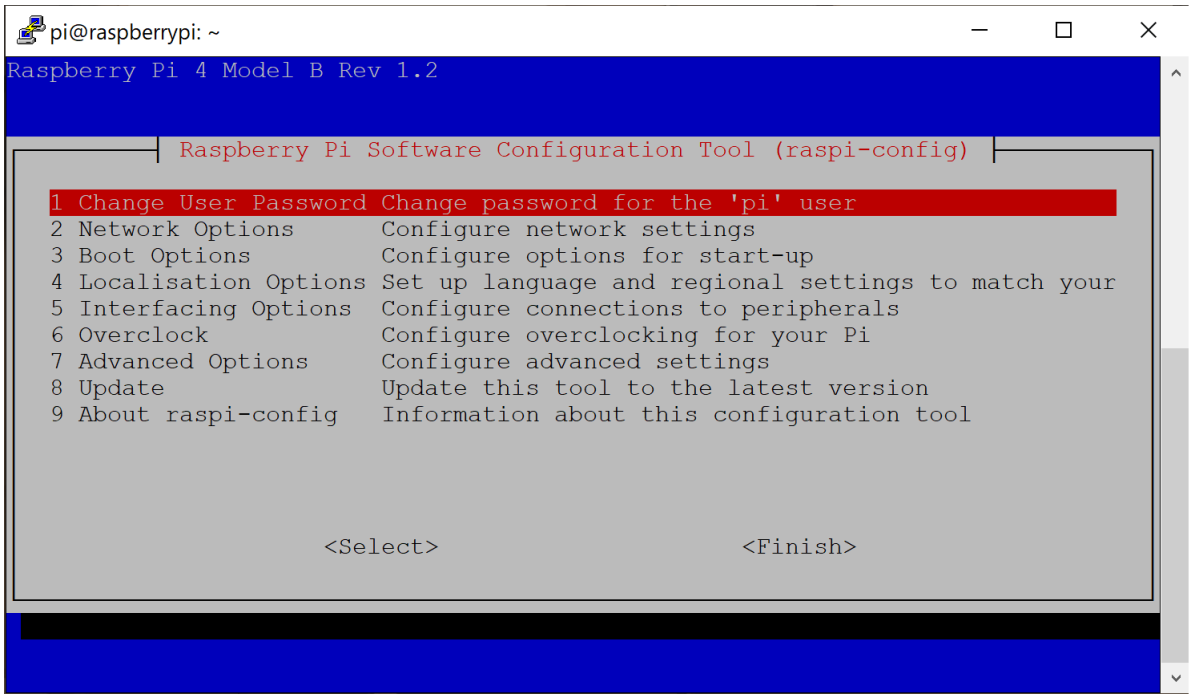
Jako zařízení bylo zvoleno RPi Zero WH, které je nejmenší z rodiny RPi. Zařízení RPi Zero WH je popsáno na straně 18. Jelikož toto zařízení neobsahuje zvukové vstupy ani výstupy, tak byla zvolena externí zvuková karta ReSpeaker 2 Mics Pi HAT. Tato zvuková karta je popsána na straně 22. Jako ústředna byla zvolena ústředna od AG-Project `SIP2SIP-info`, která je popsána na straně 16.

2.1 Instalace operačního systému

Pro samotnou instalaci operačního systému je nutné mít RPi, SD kartu a počítač s připojením na internet, odkud se stáhne nejnovější verze Raspbianu. Pomocí počítače se stáhne ze stránek výrobce instalační soubor a spustí se instalace. Pomocí aplikace Raspberry Pi Imager se zapíše nejnovější operační systém na paměťovou kartu. Nejdříve se musí vybrat verze operačního systému, to se provede tím, že se klikne na *CHOOSE OS* a vybere se Raspberry Pi OS (others). Po kliknutí se zobrazí nabídka, ze které se vybere Raspberry Pi OS Lite (32-bit). Tato verze operačního systému je bez grafického rozhraní a další nastavení se provádí pomocí příkazového řádku.

Verze operačního systému bez grafického rozhraní je volena z důvodu, že zvolené RPi nebude připojeno k žádnému zobrazovacímu médiu. Dále se musí vybrat paměťová karta, kam se zvolený operační systém nainstaluje. Nakonec se zmáčkne tlačítko *WRITE*, které nainstaluje daný operační systém na paměťovou kartu. Po skončení tohoto procesu se paměťová karta z počítače ještě nevyjme, protože je nutné na ni vložit ještě dva soubory. Tyto soubory se vloží na disk boot. První z těchto souborů se jmenuje *ssh* a nemá žádnou příponu. Tento soubor slouží k povolení vzdáleného připojení pomocí ssh na RPi.

Druhý soubor je jmenuje *wpa_supplicant.conf* a slouží k nastavení připojení RPi k bezdrátové síti. Jeho struktura je vypsaná ve výpisu 2.1.



Obr. 2.1: Nástroj na konfiguraci zařízení Raspberry Pi

Výpis 2.1: Výpis souboru wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=CZ

network={
  ssid="NAZEV-SSID"
  psk="HESLO"
  key_mgmt=WPA-PSK
}
```

Po uložení těchto souborů do disku boot se paměťová karta vyjme z počítače a vloží se do RPi. Po vložení paměťové karty do RPi se připojí napájení. Jelikož k RPi není připojeno žádné zobrazovací médium, tak se na RPi bude připojovat vzdáleně pomocí programu Putty. Před připojením k RPi je nutné si zjistit IP adresu zařízení. To se provede pomocí programu Angry IP Scanner, který slouží ke skenování připojených zařízení do dané počítačové sítě. Po zjištění IP adresy RPi je nutné spustit program Putty a do něj zadat IP adresu RPi a SSH port. SSH pracuje v základním nastavení na portu 22, který se zadá do kolonky Port. Po zadání příslušných údajů stačí jen zmáčknout tlačítko Open a počkat na sestavení spojení mezi PC a RPi. Při

sestavování spojení vyskočí tabulka s informacemi o RSA (Public-key cryptosystem) klíči a je nutné kliknout na „Accept a save“. Toto potvrzení způsobí, že při příštím přihlašování se už nebude muset ověřovat RSA klíč.

Pro přihlášení je nutné zadat uživatelské jméno a heslo. Uživatelské jméno je nastaveno v základu na: „pi“ a heslo na: „raspberry“. Po přihlášení je nutné zadat tyto příkazy:

Výpis 2.2: Příkazy na instalaci operačního systému

```
1 $ sudo apt-get update
2 $ sudo apt-get upgrade -y
3 $ sudo raspi-config
```

Příkazy, které jsou uvedené na prvním a druhém řádku zapříčiní, že systém se aktualizuje a nainstalují se nejnovější balíčky. Po provedení těchto příkazů je systém připraven na další krok. Může být proveden ještě jeden příkaz, který je uveden na třetím řádku, pomocí kterého se může nastavit uživatelské jméno, heslo, název zařízení v síti a další parametry.

2.2 Instalace rozšiřujícího modulu ReSpeaker 2 Mics Pi HAT

Rozšiřující modul Respeaker 2 Mics Pi Hat se připojuje k RPi pomocí 40pinového konektoru GPIO. Přes tento konektor se rozšiřující modul napájí a probíhá přes něj samotná komunikace. Tato podkapitola je rozdělena do dalších částí podle toho jak se rozšiřující modul zprovozňoval. První část se zabývá přípravou RPi před samotnou instalací rozšiřujícího modulu. V další části je popsána samotná instalace ReSpeakeru a ověření jeho funkčnosti. Poslední část této podkapitoly se zabývá úpravou spouštěcího souboru.

2.2.1 Příprava Raspberry Pi

Tento krok se provede jen tehdy, pokud je ReSpeaker instalován na již používaný RPi. Před samotnou instalací je nutné provést dva příkazy, které aktualizují RPi. A to jsou příkazy uvedené níže.

Výpis 2.3: Příkazy na přípravu operačního systému

```
1 $ sudo apt-get update
2 $ sudo apt-get upgarede -y
```

Příkaz uveden na prvním řádku aktualizuje všechny používané repozitáře. Druhým příkazem se nainstalují nejnovější verze používaných aplikací a systém je připraven na další krok a to je instalace potřebného softwaru pro ReSpeaker.

2.2.2 Instalace softwaru ReSpeaker

Instalaci softwaru pro ReSpeaker vyžaduje nainstalovat Git, který umožní stahovat soubory z GitHubu. Instalace Gitu se provede příkazem které je uveden na prvním řádku.

Výpis 2.4: Příkazy na instalaci ReSpeaker

```
1 $ sudo apt-get install git -y
2 $ git clone https://github.com/respeaker/seed-voicecard.
    git
3 $ cd seed-voicecard
4 $ sudo ./install.sh
5 $ sudo reboot
6 $ aplay -l
7 $ arecord -l
8 $ sudo nano /boot/config.txt
```

Samotná instalace je rozdělena do více kroků. V prvním kroku se stáhne z GitHubu seed-voicecard, kde jsou obsaženy všechny potřebné soubory pro samotnou instalaci. Stažení se provede příkazem, který je uveden na druhém řádku. Po stažení je nutné si otevřít složku seed-voicecard a to je provedeno příkazem, který je uveden na třetím řádku. Po otevření složky je nutné zadat příkaz na instalaci všech potřebných podprogramů. Instalace se provede pomocí příkazu, který je uveden na čtvrtém řádku. Po provedení instalace je nutné provést restart systému. Restart systému se provede pomocí příkazu, který je uveden na pátém řádku. Po restartu je nutné se znovu přihlásit do systému. Pomocí šestého příkazu se ověří, zda připojenou zvukovou kartu systém vidí.Tento příkaz vypíše všechny připojené zvukové karty. Příkazem na sedmém řádku se ověří viditelnost všech záznamových zařízení. Tento příkaz vypíše seznam všech připojených zařízení, které mají schopnost zaznamenat zvuk. Pokud systém vidí připojenou zvukovou kartu ReSpeaker, tak je tento krok poslední, co se instalace týče. Poté už je pouze upraven spouštěcí soubor.

2.2.3 Úprava spouštěcího souboru

Posledním krokem v instalaci rozšiřující karty je úprava spouštěcího souboru. Tato úprava spouštěcího souboru způsobí, že systém bude při startu používat rozšiřující

modul ReSpeaker 2 Mics Pi HAT pro přehrávání zvuku a jeho zachycení. K tomuto kroku je nutné provést úpravu souboru, který se nachází na disku boot. Úprava souboru se provede příkazem který, je uveden na osmém řádku.

V tomto souboru se pouze upraví řádek, který odkazuje na *dtparam=audio=on*, tento řádek se nachází v dolní části souboru. Koncový soubor *config.txt* po úpravě bude vypadat takto:

Výpis 2.5: Výpis souboru config.txt

```
# Enable audio (loads snd_bcm2835)
dtparam=audio=off
dtoverlay=i2s-mmap
dtoverlay=seed-2mic-voicecard
```

Po změně tohoto souboru je nutné provést restart systému, aby se zapsaly všechny změny a systém se spustil s již upraveným nastavením. Restart systému se provede příkazem na pátém řádku. Po restartu je nutné ověřit, zda se provedené změny zapsaly správně. Toto se provede příkazem na šestém řádku. Tento příkaz znovu vypíše seznam zařízení na přehrávání zvuku s tím rozdílem, že po provedené změně se v tomto seznamu objeví jen jedna zvuková karta. Poslední příkaze, které je nutné provést je zobrazen na sedmém řádku. Tento příkaz vypíše seznam všech zařízení na zaznamenávání zvuku. Pokud se úprava souboru *config.txt* provedla správně, tak se vypíše jen jedno zařízení. Tímto je zvuková karta ReSpeaker 2 Mics Pi Hat připravena na používání.

2.3 Programový balíček PJSIP

Tato podkapitola se zabývá popisem PJSIP a následného stažení a implementaci na RPi. PJSIP je volně dostupná a open source multimediální knihovna, která implementuje základní protokoly pro multimediální komunikaci. Je založena na protokolech SIP, RTP a SDP. Kombinuje signalizační protokol SIP s funkcemi NAT (Network Address Translation) a vytváří tak vysoce kvalitní multimediální komunikační API (Application Programing Interface), které se dá využít na jakémkoliv systému.

2.3.1 Příprava systému na stažení PJSIP

Tento krok se provede jen tehdy, pokud se bude PJSIP stahovat a kompilovat na již používané RPi. Před samotným stahováním PJSIP je nutné systém RPi aktualizovat a to se provede následujícími příkazy:

Výpis 2.6: Příkazy na přípravu operačního systému

```
1 $ sudo apt-get update
2 $ sudo apt-get upgarede -y
```

Po provedení těchto příkazů se systém aktualizuje a následně se může přejít na samotné stažení PJSIP na RPi.

2.3.2 Stažení a kompilace PJSIP

Před samotným stažením PJSIP je nutné na RPi nainstalovat Git. Pokud tento nástroj už používáte, tak se tento krok přeskočí a je možné stáhnout PJSIP. Nástroj Git se nainstaluje pomocí prvního příkazu uvedeného ve výpisu 2.8. Pro pozdější správné spuštění PJSIP je nutné doinstalovat potřebné balíčky. Instalace všech potřebných balíčků se provede tímto příkazem:

Výpis 2.7: Příkaz na stažení všech potřebných balíčků

```
$ sudo apt-get install libssl-dev libasound2-dev
   libasound2-doc libasound2-plugins libasound2 libasound
   -dev build-essential automake autoconf libtool
   libpulse-dev libsamplerate0-dev libcommoncpp2-dev
   libccrtp-dev libzrtpcpp-dev libdbus-1-dev libasound2-
   dev libdbus-c++-dev libyaml-dev libpcre3-dev libgsm1-
   dev libcelt-dev alsa-base alsa-utils alsa-tools opus-
   tools libopus-dev libpcre3-dev libcommoncpp2-dev
   libdbus-1-dev libyaml-dev libv4l-dev libvo-amrwbenc-
   dev -y
```

Po této instalaci je nutné na GitHubu stáhnout nejnovější verzi PJSIP. Při tvorbě této práce byl využit PJSIP ve verzi 2.10. V této verzi byly provedeny úpravy skriptů na kompilaci. Stažení PJSIP se provede pomocí příkazu, který je uveden na druhém řádku.

Výpis 2.8: Příkazy na kompilaci PJSIP

```
1 $ sudo apt-get install git -y
2 $ wget https://github.com/pjsip/pjproject/archive/2.10.
     tar.gz
3 $ tar xfv 2.10.tar.gz
4 $ cd pjproject-2.10
5 $ nano config_site.h
6 $ cd ~/pjproject-2.10
7 $ nano user.mak
```

```
8  $ cd pjsip-apps/src/swig
9  $ make
10 $ sudo make install
11 $ python3 python/test.py
12 $ cd ~/pjproject-2.10/pjsip-apps/bin
```

Po stažení PJSIP je nutné provést rozbalení. Rozbalení tohoto souboru se provede příkazem, který je uveden na třetím řádku. Nyní je potřeba otevřít vytvořenou složku *pjproject-2.10* čtvrtým příkazem. Po otevření složky je nutné provést úpravu souboru *config_site.h*, který se nachází v podsložce *pjlib/include/pj*. Úprava souboru se provede pátým příkazem. Po otevření tohoto souboru je nutné ho upravit.

Výpis 2.9: Výpis souboru config_site.h

```
#define PJMEDIA_AUDIO_DEV_HAS_ALSA      1
#define PJMEDIA_AUDIO_DEV_HAS_PORTAUDIO 0
#define PJMEDIA_HAS_VIDEO         0
```

Dále je nutné upravit soubor *user.mak*, který se nachází v kořenovém adresáři *pjproject-2.10*. Do tohoto adresáře se vrátíme pomocí šestého příkazu. Úprava souboru *user.mak* se provede sedmým příkazem. Pro použití PJSIP na RPi je nutné upravit tento soubor tak, aby vypadal podle tohoto vzoru:

Výpis 2.10: Výpis souboru user.mak

```
# You can create user.mak file in PJ root directory to
   specify
# additional flags to compiler and linker. For example:
export CFLAGS += -fPIC -mcpu=arm1176jzf-s -mfpu=vfp
-ffast-math -mfloat-abi=hard -mlittle-endian -munaligned-
   access
-DPJ_HAS_IPV6=1
export LDFLAGS +=
```

Po úpravě těchto souborů je nutné rozšířit odkládací prostor. Tento prostor je možné rozšířit pomocí těchto příkazů:

Výpis 2.11: Příkazy na rozšíření odkládacího prostoru

```
1 $ sudo swapoff
2 $ sudo dd if=/dev/zero of=/tempswaú bs=1024 count=1000k
3 $ sudo mkswap /tempswap
4 $ sudo swapon /tempswap
5 $ free -m
```

Příkaz *free -m* vypíše velikost volného místa v paměti. Kdyby se odkládací prostor nerozšířil, tak by se kompilace neprovedla správně a musela by se swapovací paměť dodatečně rozšířit. Před samotnou kompilací je nutné se dostat do podadresáře */pjsip-apps/src/swig*. Tento podadresář se otevře osmým příkazem z výpisu 2.8. Po otevření podadresáře se musí spustit dva příkazy, které provedou kompilaci PJSIP. tyto příkazy jsou na řádku devět a deset z výpisu 2.8

Příkaz *make* je časově náročný a vyžaduje větší odkládací prostor. Po provedení příkazů je kompilace dokončena. Pro ověření funkčnosti stačí zadat jedenáctý příkaz z výpisu 2.8, který vytvoří testovací hovor. Pokud se kompilace provedla správně, tak testovací hovor naváže spojení a v zápětí se ukončí.

Aby vše fungovalo správně je nutné upravit soubor *daemon.conf*, který se nachází v adresáři */etc/pulse*. Tento soubor je nutné upravit podle tohoto vzoru, který se nachází v příloze A.3 na straně 50.

Lze provést ještě jeden test, zda byla kompilace úspěšná. Je nutné se přepnout do složky *pjsip-apps/bin*. Do této složky se dá přepnout pomocí posledního příkazu z výpisu 2.8.

Test se provede pomocí příkazu z výpisu 2.12, který spustí aplikaci PJSIP. V následujícím příkazu je nutno nahradit *XXXX* přihlašovacími údaji z použité ústředny SIP2SIP.info.

Výpis 2.12: Příkaz na ověření hovoru

```
./pjsua-armv6l-unknown-linux-gnueabihf --id sip:XXXX
--registrar sip: XXXX --realm sip2sip . info --username
   XXXX
--password XXXX --contact sip: XXXX
--outbound sip: proxy.sipthor.net 5060 --thread -cnt 1
--nameserver 8.8.8.8 --nameserver 8.8.4.4 --publish
--clock-rate 8000 --add-codec pcma --dis-codec opus
   /48000/2
--dis-codec AMR-WB/16000/1 --dis-codec AMR/8000/1
```

3 Návrh obslužného programu

Tato část bakalářské práce je zaměřena na tvorbu obslužného programu a jeho popis. Požadavkem bakalářské práce je vytvoření prototypu zařízení, které při zmáčknutí tlačítka vytvoří hovor na zvoleného účastníka. Tedy je potřeba vytvořit skript, který po zmáčknutí tlačítka provede požadovanou funkcionalitu. Návrh obslužného programu je rozdělen do více částí. V první části se řeší zprovoznění tlačítka, které je zabudované na rozšiřující desce ReSpeaker 2 Misc Pi HAT, které je připojeno k RPi pomocí sběrnice GPIO. Další část této kapitoly se zabývá vytvořením samotných skriptů, které slouží k ovládání PJSIP. V předposlední části je popis skriptu, který přesouvá soubor *log.txt* na webový server. V poslední části jsou popsány vytvořené webové stránky. V dokumentaci je barevně odděleno, co vytvořil autor bakalářské práce a to, co převzal z internetu. Převzaté řádky kódu jsou označeny zeleně a kódy vytvořené autorem jsou označeny modře.

Tab. 3.1: Barevné rozlišení textu zdrojových kódů

Barva textu	Význam barvy
Černý	příkazy zadávané do terminálu
Modrý	kód vytvořený autorem práce
Zelený	původní kód použitého software*

* – Použit volně šiřitelný programový kód.

3.1 Instalace hardwarového tlačítka

Hardwarové tlačítko se nachází na rozšiřující desce ReSpeaker 2 Mics Pi HAT, která je připojena pomocí 40pinového konektoru GPIO. Tlačítko je připojeno k této sběrnici na pinu 17. Na ověření funkčnosti tlačítka je nutné doinstalovat balíček k programovacímu jazyku Python. Instalace tohoto balíčku se provede prvním a druhým příkazem z výpisu příkazů 3.1.

Výpis 3.1: Příkazy na instalaci tlačítka

```
1 $ sudo apt-get install python-pip -y
2 $ sudo pip install rpi.gpio -y
3 $ python tlacitko.py
```

Po nainstalování balíčku *rpi.gpio* je nutné vytvořit jednoduchý skript, který bude vypisovat do konzole informaci o aktivnosti stisknutého tlačítka. Na internetu je více

skriptů, které jsou volně dostupné a slouží ke stejnému nebo podobnému účelu. Celý skript je v příloze A.4 na straně 52.

Na začátku je nutné importovat knihovny, které se využívají. Následně je zde nadeklarované tlačítko s přiřazeným pinem sběrnice GPIO. V dalším kroku je nutné nastavit sběrnici GPIO. v *GPIO.BCM* se nastavuje číslování pinů podle Broadcom čipu. Lze místo *BCM* využít i *BOARD*, které nastavuje číslování pinů v kruhu. Doporučuje se místo *BOARD* využívat *BCM*, kvůli přehlednosti výstupních pinů. V dalším kroku je nutné nastavit tlačítko tak, aby bylo vstupním elementem. V poslední části kódu je vytvořen while cyklus (nekonečná smyčka), který testuje zda je tlačítko rozepnuté nebo sepnuté.

Vytvořený kód se spustí pomocí posledního příkazu z výpisu 3.1. Po spuštění se do terminálu začne vypisovat zda je tlačítko sepnuté nebo rozepnuté. V skriptu je nastavena nekonečná smyčka, a proto se ukončení tohoto programu provede zmáčknutím kombinace kláves *Ctrl + c*, tím dojde k přerušení skriptu.

3.2 Vytvoření programového vybavení

Tato sekce se zabývá vytvořením skriptů. Celkem jsou vytvořeny tři skripty, které mají různé úkoly. Tyto skripty se jmenují: *runscript.sh*, *copyfile.py*, *call.py* a *copylog.py*.

3.2.1 Skript *runscript.sh*

Skript *runscript.sh* má za úkol spuštění dvou skirptů a to *copyfile.py* a *call.py*. V první části skriptu *runscript.sh* je deklarace proměnných, které se využívají při spuštění skriptu *call.py*. Před samotným spuštěním skriptu *call.py* se spustí skript *copyfile.py*.

Výpis 3.2: Skript runscript.sh

```
#!/bin/bash

ID="sip:zvonek1@sip2sip.info"
REGISTRAR="sip:sip2sip.info"
REALM="sip2sip.info"
USERNAME="zvonek1@sip2sip.info"
PASSWORD="zvonek123456"
CONTACT="sip:zvonek1@sip2sip.info"
OUTBOUND="sip:proxy.sipthor.net:5060"
THREAD="1"
NAMESERVER="8.8.8.8"
NAMESERVER2="8.8.4.4"
CLOCKRATE="8000"
ADDCODEC="pcma"
DISCODEC="opus/48000/2"
DISCODEC2="AMR-WB/16000/1"
```

```
DISCODEC3="AMR/8000/1"

python copyfile.py
python call.py --id sip:zvonek1@sip2sip.info --registrar sip:sip2sip.info --realm
    sip2sip.info --username zvonek1 --password zvonek123456 --contact sip:
    zvonek1@sip2sip.info --outbound sip:proxy.sipthor.net:5060 --thread-cnt 1 --
    nameserver 8.8.8.8 --nameserver 8.8.4.4 --publish --clock-rate 8000 --add-codec
    pcma --dis-codec opus/48000/2 --dis-codec AMR-WB/16000/1 --dis-codec AMR/8000/1
     --client sip:zvonek2@sip2sip.info --client2 sip:zvonek3@sip2sip.info
```

Aby se spustil skript *runscript.sh*, tak je nutné provést úpravu souboru *rc.local*. To se provede příkazem z výpisu 3.3.

Výpis 3.3: Příkaz na úpravu souboru rc.local

```
1 $ sudo nano /etc/rc.local
```

Je nutné se v tomto souboru dostat na konec a před *exit 0* zadat „/boot/runscript.sh &“. Tímto se dosáhne spuštění požadovaného skriptu po spuštění systému.

Výpis 3.4: Soubor rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

/boot/runscript.sh "&"

exit 0
```

3.2.2 Skript *copyfile.py*

Skript *copyfile.py* má za úkol překopírování souboru *wpa_supplicant.conf* z adresáře *etc/wpa_supplicant* do disku *boot*.

Výpis 3.5: Skript copyfile.py

```
import subprocess as sub
import shlex
```

```
process = sub.Popen(shlex.split("sudo cp /etc/wpa_supplicant/wpa_supplicant.conf /
    boot"),
                    stdout=sub.PIPE,
                    universal_newlines=True)

while True:
    output = process.stdout.readline()
    print((output.strip()))

    return_code = process.poll()
    if return_code is not None:
        print('RETURN CODE', return_code)
        for output in process.stdout.readlines():
            print(output.strip())
        break
```

Překopírování souboru *wpa_supplicant.conf* se provádí z toho důvodu, že po startu systému se tento soubor překopírovává do jiné složky a už nejde naleznout na disku *boot*. Soubor *wpa_supplicant.conf* nese informace o nastavení připojení k WiFi síti.

3.2.3 Skript *parser.py*

Skript *parser.py* byl vytvořen, za účelem otestování předávání parametrů mezi dvěma skripty. Tento skript je nadále ještě využíván v skriptu *call.py*.

Výpis 3.6: Skript parser.py

```
#!/usr/bin/env python3

import sys
import argparse



class ParseArguments():

    def parse(self, args=sys.argv[1:]):
      print("parse")
      parser = argparse.ArgumentParser()

      parser.add_argument("--id")
      parser.add_argument("--registrar")
      parser.add_argument("--realm")
      parser.add_argument("--username")
      parser.add_argument("--password")
      parser.add_argument("--contact")
      parser.add_argument("--outbound")
      parser.add_argument("--thread-cnt")
      parser.add_argument("--nameserver", action="append")
      parser.add_argument("--publish", action="store_true") # arg
      parser.add_argument("--clock-rate")
      parser.add_argument("--add-codec")
      parser.add_argument("--dis-codec", action="append")
    parser.add_argument("--client")
```

```
    parser.add_argument("--client2")

      options = parser.parse_args(args)
      return options


def main():
  print("def")
  p = ParseArguments()
  args = p.parse()

  print(args)
  print(args.nameserver)


if __name__ == '__main__':
  main()
```

3.2.4 Skript *call.py*

Skript *call.py* je částečně převzat a upraven, aby splňoval všechny požadavky na správnou funkčnost bakalářské práce. Celý skript se nachází v příloze A.8 na straně 55. Zde budou popsány části, které byly upraveny. Do skriptu *call.py* byla přidána i jedna funkčnost navíc.

V první části skriptu byla přidána část na import parametrů ze skriptu *run-script.sh*. Tímto importem se importuje informace o uživateli a další potřebné parametry k registraci uživatele k ústředně. Dále se importují adresy uživatelů, od kterých se automaticky přijímá hovor.

Výpis 3.7: Předávání parametrů

```
import argparse

class ParseArgumennts():
    def parse(self,args=sys.argv[1:]):
        parser= argparse.ArgumentParser()


        parser.add_argument("--id")
        parser.add_argument("--registrar")
        parser.add_argument("--realm")
        parser.add_argument("--username")
        parser.add_argument("--password")
        parser.add_argument("--contact")
        parser.add_argument("--outbound")
        parser.add_argument("--thread-cnt")
        parser.add_argument("--nameserver", action="append")
        parser.add_argument("--publish", action="store_true")
        parser.add_argument("--clock-rate")
        parser.add_argument("--add-codec")
        parser.add_argument("--dis-codec", action="append")
        parser.add_argument("--client")
        parser.add_argument("--client2")
```

```
        options = parser.parse_args(args)
        return options

parser = ParseArgumennts()
args = parser.parse()
```

Dále je zde upraven kód na příjem hovorů. Zde byla celá část kódu předělána tak, že při příchozím hovoru se testuje zda příchozí hovor je od účastníka, který je nadefinován pro automatický příjem.

Výpis 3.8: Automatický příjem hovorů

```
# Notification on incoming call
    def on_incoming_call(self, call):
        global current_call
        if current_call:
            call.answer(486, "Busy")
            return
        client1=args.client
        client2=args.client2
        if call.info().remote_uri == client1:
            current_call = call
            call_cb = MyCallCallback(current_call)
            current_call.set_callback(call_cb)
            current_call.answer(100)
        else call.info().remote_uri == client2
            current_call = call
            call_cb = MyCallCallback(current_call)
            current_call.set_callback(call_cb)
            current_call.answer(100)
```

Následně je zde přidaná funkčnost na zachytávání čísel zadaných na číselníku. Tento příjem je ve zdrojových kódech označen jako dtmf. Tyto čísla se po zachycení vypisují do konzole.

Výpis 3.9: Zachytávání čísel zadaných na číselníku

```
    def on_dtmf_digit(self,digits):
        print(digits)
```

Registrace uživatele k ústředně bylo nutné upravit tak, že je tato registrace uživatele prováděna pomocí předávaných parametrů. Před úpravou byla registrace nastavena na statické hodnoty.

Výpis 3.10: Registrace uživatele

```
acc = lib.create_account(pj.AccountConfig (username=args.username, password=args.
    password, domain=args.realm, proxy=args.outbound))
```

Výpis 3.11: Nastavení logfile

```
lib.init(log_cfg = pj.LogConfig(level=LOG_LEVEL,filename='/home/pi/log.txt',callback
    =log_cb))
```

Poslední část, kterou bylo nutno upravit je hlavní smyčka. Tuto smyčku bylo nutno upravit, tak že reaguje na zmáčknutí tlačítka. Po zmáčknutí tlačítka začne volat předem definovaný uživatel, který se nadefinoval v skriptu *runscript.sh*.

Výpis 3.12: Menu

```
# Menu loop
    while True:
        print "My SIP URI is", my_sip_uri
        input = GPIO.input(BUTTON)
        if input :
            lck = lib.auto_lock()
            current_call = make_call(address = args.client)
            del lck

        else:
            os.system('python copylog.py')
```

Celý skript se nachází v příloze A.8 na straně 55.

3.2.5 Skript *copylog.py*

Skript *copylog.py* má za úkol překopírování souboru *log.txt* z RPi do složky *www/file* na vzdáleném webovém serveru.

Výpis 3.13: Skript copylog.py

```
import subprocess as sub
import shlex

process = sub.Popen(shlex.split("sudo scp /home/pi/log.txt pi@WEBSERVERIP:/home/pi/
    www/file"),
                    stdout=sub.PIPE,
                    universal_newlines=True)

while True:
    output = process.stdout.readline()
    print((output.strip()))

    return_code = process.poll()
    if return_code is not None:
        print('RETURN CODE', return_code)
        for output in process.stdout.readlines():
            print(output.strip())
        break
```

3.3 Webové stránky zobrazující historii hovorů a stav zařízení

Webové stránky jsou vytvořeny pomocí programovacího jazyka Python. K jejich tvorbě byl použit balíček Django. Na webové stránce se nalézají dvě tlačítka. První tlačítko ověřuje dostupnost zařízení. Druhé tlačítko vypisuje historii volání. Tyto funkce jsou vytvořeny jako dva skripty, které se můžou použít zvlášť.

Dostupnost zařízení se ověřuje pomocí jednoduchého ping testu, při kterým se pošle jeden dotaz a čeká se na odpověď. Pokud obdrží odpověď, tak se vypíše, že je zařízení dostupné. Pro obdržení dotazu je nastaven časový limit.

Výpis 3.14: Skript ping.py na ověření dostupnosti zařízení

```
import os

hostname = "192.168.0.1"
response = os.system("ping " + hostname + " /n 1")


if response == 0:
  test = True
  print ('Zvonke je dostupny')
else:
  test = False
  print ('Zvonek je nedostupny!')
```

Výpis historie volání se provádí pomocí stahování dat z použité ústředny `SIP2SIP.info`. Přístup k této ústředně je řešen jako volání funkcí pomocí url adresy. Pomocí této url adresy se dá stáhnout historie volání nebo změnit nastavení klienta. Data, které se přijímají, jsou ve formátu JSON. Tyto data jsou následně vyobrazeny.

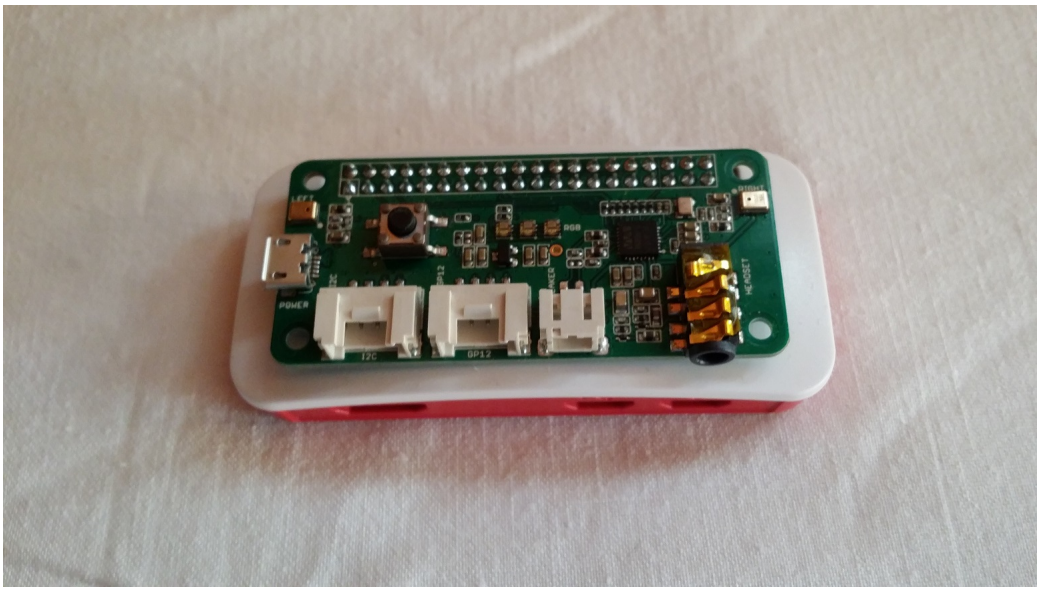
Výpis 3.15: Skript data.py na stahování historie z `SIP2SIP.info`

```
from builtins import print

import requests
from requests.auth import HTTPBasicAuth


response = requests.get("https://enrollment.sipthor.net/settings.phtml?action=
    get_history&realm=sip2sip.info",
                        auth=HTTPBasicAuth('zvonek1','Silent5Killer'))
data=response.json()
print(type(data))
print(data)
```

4 Ověření funkčnosti



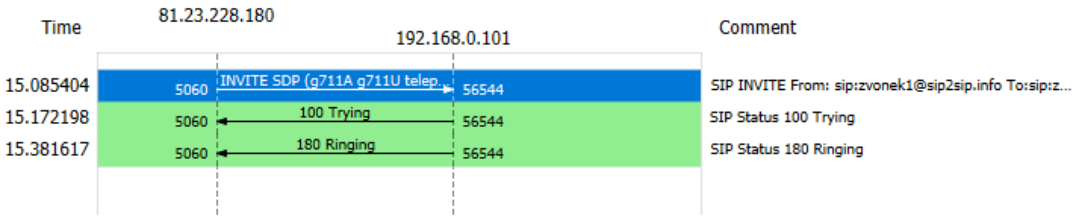
Obr. 4.1: ReSpeaker 2 Mics Pi HAT na Raspberry Pi Zero

Tato kapitola se zabývá ověřením funkčnosti celého projektu. Testování funkčnosti bylo prováděno postupně, jak byla bakalářská práce realizována. První testovanou věcí na zvukové kartě bylo ověření, zda dokáže přijímat a vydávat zvukové signály. Celé zvukové testování bylo pomocí Alsamixéru. Další testovanou částí byl programový balíček PJSIP. Testování programového balíčku PJSIP bylo provedeno pomocí druhého testovacího příkazu, který se nachází na straně 31. Po spuštění tohoto příkazu se v terminálu objeví menu programu PJSIP. Pomocí tohoto menu se navolí vytvoření hovoru a zadá se adresa volaného. Pomocí tohoto programu byla ověřena správná funkčnost PJSIP na RPi. Na obrázku 4.1 je vyobrazeno zařízení RPi s externí zvukovou kartou Respeaker 2 Mics Pi HAT, která obsahuje hardwarové tlačítko.

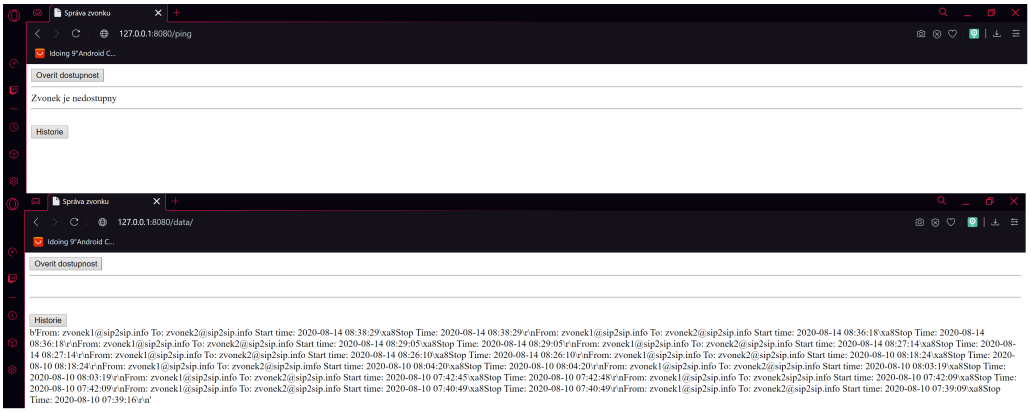
Další část ověření funkčnosti je zaměřeno na vytvořené skripty. První skript, který byl ověřen, byl na kopírování souboru *wpa_supplicant.conf*. Tento skript byl ověřen na vícero zařízeních RPi. Po odzkoušení skriptu *copyfile.py* byl vytvořen skript na automatické spuštění při zapnutí systému. Tento skript se jmenuje *runscript.sh*. Poslední skript, který byl vytvořen a odzkoušen se jmenuje *call.py*.

Celý vytvořený program funguje tak, že při spuštění systému se spustí automatický skript, který překopíruje soubor *wpa_supplicant.conf* na disk *boot* a následně spustí skript *call.py*. Tento skript zaregistruje nadefinovaného uživatele k SIP ústředně a čeká na zmáčknutí tlačítka.

Po zmáčknutí tlačítka se vytvoří hovor na účet volaného. Ověření funkčnosti celého projektu bylo provedeno tak, že na mobilní zařízení byl nainstalován program na SIP komunikaci a byl registrován druhý uživatel. Následně bylo zařízení RPi připojeno do elektrické sítě a po spuštění systému se zmáčklo tlačítko a byl vytvořen testovací hovor k druhému uživateli. Vytvoření hovoru je zachyceno na obrázku 4.2.



Obr. 4.2: Zachycení vytvoření hovoru pomocí programu Wireshark



Obr. 4.3: Vytvořené webové stránky

Závěr

Tato bakalářská práce měla za cíl vytvořit prototyp jednoduchého VoIP komunikátoru. Při tvorbě bylo použito zařízení RPi Zero WH, ke kterému byl připojen rozšiřující modul ReSpeaker 2 Mics Pi HAT. Jako operační systém byl využit systém nabízený společností Raspberry Pi Foundation v nejnovější verzi s označením Buster. Instalace a nastavení operačního systému proběhlo bez problému. Po prvním spuštění se zjistilo, že vytvořený soubor (*wpa_supplicant.conf* na disku *boot* se přesunul do složky */etc/wpa_supplicant*. Tímto přesunutím bylo znemožněno následné úpravy souboru, tudíž byl vytvořen skript, který tento soubor zkopíroval zpátky na disk *boot* a tím umožnil následné úpravy tohoto souboru.

Při instalaci rozšiřující karty ReSpeaker 2 Mics Pi HAT bylo zjištěno, že nejnovější operační systém využívá jádro systému ve verzi 4.19 a zvuková karta má podporu pro jádro systému 3.18. Ale s tímto nebyl problém, protože při instalaci potřebných balíčků se z internetu stáhly všechny potřebné soubory, které umožňují práci této zvukové karty na jádru systému 4.19 a novějších verzí. Nejnovější verze jádra systému je 5.4. Po instalaci byla ověřena funkčnost zvukové karty.

Po otestování zvukové karty probíhalo samotné stažení PJSIP a následná kompilace. Před samotnou kompilací bylo potřeba stáhnout a nainstalovat potřebné balíčky tak, aby PJSIP fungoval správně. Po nainstalování balíčků bylo nutno stáhnout nejnovější verzi PJSIP a upravit konfigurační balíčky. Při kontrole konfiguračních balíčků bylo zjištěno, že nejnovější verze 2.10 má opravené konfigurační balíčky a tudíž nebyla potřeba je upravovat oproti verzi 2.9. Byly provedeny dvě kompilace. První kompilace byla bez problémů, ale při druhé kompilaci se vyskytly problémy. Tyto problémy byly způsobeny malým odkládacím oddílem, tudíž bylo nutno tento prostor rozšířit. Po rozšíření prostoru byla provedena druhá kompilace. Po provedení obou kompilací byla vyzkoušena funkčnost PJSIP. Při ověřování funkčnosti byl zjištěn problém se zvukem, ale ten byl následně opraven. Bylo potřeba opravit soubor *daemon.conf*. Po opravě tohoto souboru nebyly zjištěny další problémy.

Po kompilaci a otestování PJSIP přišly na řadu spouštěcí skripty. První skript, který byl vytvořen byl skript, který kopíruje soubor *wpa_supplicant.conf*. Kopírovací skript byl vytvořen a následně byla odzkoušena funkčnost. Posléze byl vytvořen hlavní spouštěcí skript, který na začátku volal jen kopírovací skript. Nakonec byl vytvořen skript, který má na starosti zaregistrování účastníka a vytvoření hovoru, čekání na příchozí hovor a výpis dtmf kódu do konzole. S tímto skriptem byly problémy kvůli nepochopení činnosti předvytvořených skriptů a následné implementaci všech potřebných součástí. Při tvorbě byly využity návody a popisy všech funkcí na stránkách výrobce. Tyto stránky byly nepřehledné a některé funkčnosti nebyly vůbec popsány. Tvorba tohoto skriptu je velmi náročná i pro zkušené programá-

tory. Všechny vytvořené skripty jsou vystaveny na GitHubu pod licencí MIT. Odkaz na stránky, kde se nachází vytvořené kódy `https://github.com/Devil811/VoIP-communicator`. Byly také vytvořeny webové stránky, kde se nachází historie volání.

Další možné využití této bakalářské práce je jako inteligentní domovní zvonek, který je zabudovaný v krabičce s reproduktorem a s relé, které otevře dveře při zadání určité kombinace dtmf kódu.

Literatura

[1] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[2] PILGRIM, M. Ponořme se do Python(u) 3. CZ.NIC, 2010. 435 s. ISBN: 978-80-904248-2-1.

[3] Raspberry Pi: About us. Raspberry Pi: About us [online]. [cit. 2020-08-10]. Dostupné z: https://www.raspberrypi.org/about/

[4] Raspberry Pi Zero. Raspberry Pi Zero [online]. [cit. 2020-06-08]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-zero/

[5] Raspberry Pi 3. Raspberry Pi 3 [online]. [cit. 2020-06-08]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[6] Raspberry Pi 4. Raspberry Pi 4 [online]. [cit. 2020-06-08]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[7] Ubuntu. Ubuntu Mate [online]. [cit. 2020-06-08]. Dostupné z: https://ubuntu-mate.org/ports/raspberry-pi/

[8] OSMC: About. OSMC: About [online]. [cit. 2020-08-10]. Dostupné z: https://osmc.tv/about/

[9] LibreELEC: About. LibreELEC: About [online]. [cit. 2020-08-10]. Dostupné z: https://libreelec.tv/about/

[10] PiNet. PiNet [online]. [cit. 2020-06-08]. Dostupné z: http://pinet.org.uk/

[11] Windows pro Internet věcí. Windows [online]. [cit. 2020-06-08]. Dostupné z: https://developer.microsoft.com/cs-cz/windows/iot/

[12] HAVELKA, Ondřej. DETEKCE VOIP APLIKACÍ VE STATISTIKÁCH SÍŤOVÉHO PROVOZU. Brno, 2009. Bakalářská práce. VUT Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádník.

[13] BĚLÍK, David. PROPRIETÁRNÍ VOIP PROTOKOLY VÝROBCŮ POBOČKOVÝCH ÚSTŘEDEN. Brno, 2011. Bakalářská práce. VUT Fakulta informačních technologií. Vedoucí práce Ing. Pavel Šilhavý, Ph.D.

[14] DANKO, Martin. MODELOVÁNÍ KVALITY SLUŽEB V POČÍTAČOVÝCH SÍTÍCH. Brno, 2012. Diplomová práce. VUT Fakulta informačních technologií. Vedoucí práce Ing. Ondřej Ryšavý, Ph.D.

[15] ReSpeaker 2 Mics Pi HAT. RPishop.cz [online]. [cit. 2020-06-08]. Dostupné z: https://rpishop.cz/zvukove-karty/1173-respeaker-2-mics-pi-hat.html

[16] Sip2sip.info. AG Project: about [online]. [cit. 2020-08-10]. Dostupné z: https://ag-projects.com/categories/services/

Seznam symbolů, veličin a zkratek

MSRP Message Session Relay Protocol

XCAP XML Configuration Acces Protocol

XMPP Extensible Messaging and Presence Protocol

ISDN Integrated Services Digital Network

SSH Secure Shell

ARM Ashton Raggatt McDougall

GPIO General-purpose input/output

USB Universal Serial Bus

SRTP Secure Real-time Transport Protocol

HTTP Hypertext Transfer Protocol

URL Uniform Resource Locator

IAX Inter-Asterisk eXchange

SRTP Secure Real Time Protocol

IP Internet Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

H.323 Kodek

open source Otevřený software

UAC User Agent Clien

UAS User Agent Server

DHCP Dynamic Host Configuration Protokol

DNS Domain Name System

Pi-Hole Blokování internetové inzerce

RPi Raspberry Pi

VoIP	Voice over Internet Protocol
SIP	Session Initiation Protocol
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
RSA	Public-key cryptosystem
API	Application Programing Interface
NAT	Network Address Translation
HDMI	High-Definition Multimedia Interface
OEM	Original Equipment Manufacture

Seznam příloh

A Zdrojové kódy

Tab. A.1: Barevné rozlišení textu zdrojových kódů

Barva textu	Význam barvy
Černý	příkazy zadávané do terminálu
Modrý	kód vytvořený autorem práce
Zelený	původní kód použitého software*

* – Použit volně šiřitelný programový kód.

Výpis A.1: Soubor config_site.h

```
#define PJMEDIA_AUDIO_DEV_HAS_ALSA      1
#define PJMEDIA_AUDIO_DEV_HAS_PORTAUDIO 0
#define PJMEDIA_HAS_VIDEO       0
```

Výpis A.2: Soubor user.mak

```
export CFLAGS += -fPIC -mcpu=arm1176jzf-s -mfpu=vfp
-ffast-math -mfloat-abi=hard -mlittle-endian
-munaligned-access -DPJ_HAS_IPV6=1
export LDFLAGS +=
```

Výpis A.3: Soubor deamon.conf

```
; daemonize = no
; fail = yes
; allow-module-loading = yes
; allow-exit = yes
; use-pid-file = yes
; system-instance = no
; local-server-type = user
; enable-shm = yes
; enable-memfd = yes
; shm-size-bytes = 0
; lock-memory = no
; cpu-limit = no

; high-priority = yes
; nice-level = -11

; realtime-scheduling = yes
; realtime-priority = 5

; exit-idle-time = 20
; scache-idle-time = 20

; dl-search-path = (depends on architecture)

; load-default-script-file = yes
; default-script-file = /etc/pulse/default.pa

; log-target = auto
; log-level = notice
; log-meta = no
; log-time = no
; log-backtrace = 0

; resample-method = speex-float-1
; enable-remixing = yes
; enable-lfe-remixing = no
; lfe-crossover-freq = 0

; flat-volumes = yes

; rlimit-fsize = -1
; rlimit-data = -1
; rlimit-stack = -1
; rlimit-core = -1
; rlimit-as = -1
; rlimit-rss = -1
; rlimit-nproc = -1
; rlimit-nofile = 256
; rlimit-memlock = -1
; rlimit-locks = -1
; rlimit-sigpending = -1
; rlimit-msgqueue = -1
; rlimit-nice = 31
; rlimit-rtprio = 9
; rlimit-rttime = 200000

; default-sample-format = s16le
; alternate-sample-rate = 48000
```

```
; default-sample-channels = 2
; default-channel-map = front-left,front-right

; default-fragments = 4
; default-fragment-size-msec = 25

; enable-deferred-volume = yes
; deferred-volume-safety-margin-usec = 8000
; deferred-volume-extra-delay-usec = 0

high-priority = no
realtime-scheduling = no
resample-method = trivial
default-sample-rate = 48000
```

Výpis A.4: Skript tlacitko.py

```
import RPi.GPIO as GPIO
import time

BUTTON = 17

GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN)

while True:
    state = GPIO.input(BUTTON)
    if state:
        print("off")
    else:
        print("on")
time.sleep(1)
```

Výpis A.5: Skript runscript.sh

```
#!/bin/bash

ID="sip:zvonek1@sip2sip.info"
REGISTRAR="sip:sip2sip.info"
REALM="sip2sip.info"
USERNAME="zvonek1@sip2sip.info"
PASSWORD="zvonek123456"
CONTACT="sip:zvonek1@sip2sip.info"
OUTBOUND="sip:proxy.sipthor.net:5060"
THREAD="1"
NAMESERVER="8.8.8.8"
NAMESERVER2="8.8.4.4"
CLOCKRATE="8000"
ADDCODEC="pcma"
DISCODEC="opus/48000/2"
DISCODEC2="AMR-WB/16000/1"
DISCODEC3="AMR/8000/1"

python copyfile.py
python call.py --id sip:zvonek1@sip2sip.info --registrar sip:sip2sip.info --realm
    sip2sip.info --username zvonek1 --password zvonek123456 --contact sip:
    zvonek1@sip2sip.info --outbound sip:proxy.sipthor.net:5060 --thread-cnt 1 --
    nameserver 8.8.8.8 --nameserver 8.8.4.4 --publish --clock-rate 8000 --add-codec
    pcma --dis-codec opus/48000/2 --dis-codec AMR-WB/16000/1 --dis-codec AMR/8000/1
     --client sip:zvonek2@sip2sip.info --client2 sip:zvonek3@sip2sip.info
```

Výpis A.6: Skript copyfile.py

```
import subprocess as sub
import shlex

process = sub.Popen(shlex.split("sudo cp /etc/wpa_supplicant/wpa_supplicant.conf /
    boot"),
                    stdout=sub.PIPE,
                    universal_newlines=True)

while True:
    output = process.stdout.readline()
    print((output.strip()))

    return_code = process.poll()
    if return_code is not None:
        print('RETURN CODE', return_code)
        for output in process.stdout.readlines():
            print(output.strip())
        break
```

Výpis A.7: Skript parser.py

```
#!/usr/bin/env python3

import sys
import argparse



class ParseArguments():

    def parse(self, args=sys.argv[1:]):
      print("parse")
      parser = argparse.ArgumentParser()

      parser.add_argument("--id")
      parser.add_argument("--registrar")
      parser.add_argument("--realm")
      parser.add_argument("--username")
      parser.add_argument("--password")
      parser.add_argument("--contact")
      parser.add_argument("--outbound")
      parser.add_argument("--thread-cnt")
      parser.add_argument("--nameserver", action="append")
      parser.add_argument("--publish", action="store_true") # arg
      parser.add_argument("--clock-rate")
      parser.add_argument("--add-codec")
      parser.add_argument("--dis-codec", action="append")
    parser.add_argument("--client")
    parser.add_argument("--client2")

      options = parser.parse_args(args)
      return options


def main():
  print("def")
  p = ParseArguments()
  args = p.parse()

  print(args)
  print(args.nameserver)


if __name__ == '__main__':
  main()
```

Výpis A.8: Skript call.py

```
# $Id$
#
# SIP call sample.
#
# Copyright (C) 2003-2008 Benny Prijono <benny@prijono.org>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#

import sys
import pjsua as pj
import argparse
import RPi.GPIO as GPIO
import os.system

BUTTON = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON,GPIO.IN)

class ParseArgumennts():
    def parse(self,args=sys.argv[1:]):
        parser= argparse.ArgumentParser()


        parser.add_argument("--id")
        parser.add_argument("--registrar")
        parser.add_argument("--realm")
        parser.add_argument("--username")
        parser.add_argument("--password")
        parser.add_argument("--contact")
        parser.add_argument("--outbound")
        parser.add_argument("--thread-cnt")
        parser.add_argument("--nameserver", action="append")
        parser.add_argument("--publish", action="store_true")
        parser.add_argument("--clock-rate")
        parser.add_argument("--add-codec")
        parser.add_argument("--dis-codec", action="append")
        parser.add_argument("--client")
        parser.add_argument("--client2")
        options = parser.parse_args(args)
        return options


LOG_LEVEL=3
current_call = None
```

```
# Logging callback
def log_cb(level, str, len):
    print str,

# Callback to receive events from account
class MyAccountCallback(pj.AccountCallback):

    def __init__(self, account=None):
        pj.AccountCallback.__init__(self, account)

    # Notification on incoming call
    def on_incoming_call(self, call):
        global current_call
        if current_call:
            call.answer(486, "Busy")
            return
        client1=args.client
        client2=args.client2
        if call.info().remote_uri == client1:
            current_call = call
            call_cb = MyCallCallback(current_call)
            current_call.set_callback(call_cb)
            current_call.answer(100)
        else call.info().remote_uri == client2
            current_call = call
            call_cb = MyCallCallback(current_call)
            current_call.set_callback(call_cb)
            current_call.answer(100)


# Callback to receive events from Call
class MyCallCallback(pj.CallCallback):

    def __init__(self, call=None):
        pj.CallCallback.__init__(self, call)

    # Notification when call state has changed
    def on_state(self):
        global current_call
        print "Call with", self.call.info().remote_uri,
        print "is", self.call.info().state_text,
        print "last code =", self.call.info().last_code,
        print "(" + self.call.info().last_reason + ")"

        if self.call.info().state ==
        pj.CallState.DISCONNECTED:
            current_call = None
            print 'Current call is', current_call

    # Notification when call's media state has changed.
    def on_media_state(self):
        if self.call.info().media_state ==
        pj.MediaState.ACTIVE:
            # Connect the call to sound device
            call_slot = self.call.info().conf_slot
            pj.Lib.instance().conf_connect(call_slot, 0)
            pj.Lib.instance().conf_connect(0, call_slot)
            print "Media is now active"
        else:
```

```
            print "Media is inactive"

    def on_dtmf_digit(self,digits):
        print(digits)

# Function to make call
def make_call(uri):
    try:
        print "Making call to", uri
        return acc.make_call(uri, cb=MyCallCallback())
    except pj.Error, e:
        print "Exception: " + str(e)
        return None


parser = ParseArgumennts()
args = parser.parse()

lib = pj.Lib()

try:

    lib.init(log_cfg = pj.LogConfig(level=LOG_LEVEL,filename='/home/pi/log.txt',
    callback=log_cb))


    transport = lib.create_transport(pj.TransportType.UDP,pj.TransportConfig(0))
    print "\nListening on", transport.info().host,
    print "port", transport.info().port, "\n"


    lib.start()


    acc = lib.create_account(pj.AccountConfig (username=args.username, password=args
    .password, domain=args.realm, proxy=args.outbound))


    if len(sys.argv) > 1:
        lck = lib.auto_lock()
        current_call = make_call(sys.argv[1])
        print 'Current call is', current_call
        del lck

    my_sip_uri = "sip:" + transport.info().host + ":" + str(transport.info().port)

    # Menu loop
    while True:
        print "My SIP URI is", my_sip_uri
        input = GPIO.input(BUTTON)
        if input :
            lck = lib.auto_lock()
            current_call = make_call(address = args.client)
            del lck

        else:
            os.system('python copylog.py')
```

```
    # Shutdown the library
    transport = None
    acc.delete()
    acc = None
    lib.destroy()
    lib = None

except pj.Error, e:
    print "Exception: " + str(e)
    lib.destroy()
    lib = None
```

Výpis A.9: Skript copylog.py

```
import subprocess as sub
import shlex

process = sub.Popen(shlex.split("sudo scp /home/pi/log.txt pi@WEBSERVERIP:/home/pi/
    www/file"),
                    stdout=sub.PIPE,
                    universal_newlines=True)

while True:
    output = process.stdout.readline()
    print((output.strip()))

    return_code = process.poll()
    if return_code is not None:
        print('RETURN CODE', return_code)
        for output in process.stdout.readlines():
            print(output.strip())
        break
```

Výpis A.10: Skript ping.py na ověření dostupnosti zařízení

```
import os

hostname = "192.168.0.1"
response = os.system("ping " + hostname + " /n 1")


if response == 0:
  test = True
  print ('Zvonke je dostupny')
else:
  test = False
  print ('Zvonek je nedostupny!')
```

Výpis A.11: Skript data.py na stahování historie z `SIP2SIP.info`

```
from builtins import print

import requests
from requests.auth import HTTPBasicAuth


response = requests.get("https://enrollment.sipthor.net/settings.phtml?action=
    get_history&realm=sip2sip.info",
                        auth=HTTPBasicAuth('zvonek1','Silent5Killer'))
data=response.json()
print(type(data))
print(data)
```