VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2023 Šimon Čecháček

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

CHYTRÝ VOIP ZVONEK

SMART VOIP DOORBELL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Šimon Čecháček

AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Caha

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Šimon Čecháček ***ID:*** 220908

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

NÁZEV TÉMATU:

Chytrý VoIP zvonek

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte technologie VoIP a protokol SIP. Seznamte se s prototypem jednoduchého VoIP komunikátoru na bázi jednodeskového počítače Raspberry Pi a obslužného programu napsaném v Python 3 vyvíjeném na Ústavu telekomunikací. Rozšiřte zařízení a jeho ovládací program o zpracování příchozích DTMF kódů, jejich kontrolu a podmíněné sepnutí relé. Zařízení rozšiřte o jednosměrný přenos videa. Proveďte revizi kódu ovládacího programu. Navrhněte krabičku na celé zařízení vytisknutelnou na 3D tiskárně. Pro VoIP komunikaci využijte vybraného poskytovatele (např. sip2sip.info). Vytvořený kód vystavte na GitHub. Proveďte analýzu bezpečnosti navrženého celku a identifikujte rizika nalezených zranitelností.

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

Vedoucí práce: Ing. Tomáš Caha

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

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Bakalářská práce se zaměřuje na rozšíření existujícího VoIP komunikátoru na jednodeskovém počítači Raspberry Pi, který využívá protokolu SIP. Práce přináší mimo jiné rozšíření o jednosměrný přenos videa a podmíněné sepnutí relé. Výsledný komunikátor je schopen uskutečnit hovor s oboustrannou komunikací, spustit definovaný kód po autorizaci volaného uživatele pomocí DTMF kódů a sepnout relé. Celé zařízení je uloženo v krabičce, vytisknuté na 3D tiskárně. IT bezpečnost navrženého řešení byla analyzována a kód i 3D model šasi jsou dostupné na GitHubu.

KLÍČOVÁ SLOVA

Zvonek, 3D Tisk, DTMF kódy, PJSIP, Python, Raspberry Pi, SIP, VoIP

ABSTRACT

This bachelor thesis focuses on the extension of an existing VoIP communicator on a single board computer Raspberry Pi, which uses the SIP protocol. The thesis introduces, among other things, an extension for one-way video transmission and conditional relay switching. The resulting communicator is able to make a call with two-way communication, execute a defined code using DTMF codes after authorization of the called user and switch a relay. The entire device is housed in a box printed on a 3D printer. The IT security of the proposed solution has been analyzed and the code and 3D model of the chassis are available on GitHub.

KEYWORDS

3D Printing, DTMF codes, PJSIP, Python, Raspberry Pi, SIP, VoIP, Doorbell

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

ČECHÁČEK, Šimon. *Chytrý VoIP zvonek*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 67 s. Bakalářská práce. Vedoucí práce: Ing. Tomáš Caha

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Šimon Čecháček
VUT ID autora:	220908
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	Chytrý VoIP zvonek

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné 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*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce, panu Ing. Tomášovi Cahovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále chci poděkovat také kolegům Bc. Aleši Michálkovi a Martinu Stejskalovi za asistenci při jemných činnostech (např. pájení), které jsou nad mé motorické schopnosti. Za zaškolení v práci s 3D modelovacím softwarem, zapůjčení 3D tiskárny a poradenství v oblasti elektrotechniky děkuji pánům Filipu Královi a Vladislavu Vitíkovi. Díky patří i společnosti miniTEL s.r.o. a jejímu jednateli Ing. Petru Soukupovi za ochotu a snahu pomoci při řešení softwarových problémů na poli SIP komunikace a práce s PJSIP knihovnou.

Obsah

Úvod **12**

1 Výběr hardwaru a softwaru **13**
1.1 Hardware . . . 13
1.1.1 Parametry výběru . . . 13
1.1.2 Specifikace modelu . . . 13
1.1.3 Raspberry Pi Zero 2W . . . 15
1.1.4 Zvolené zařízení . . . 15
1.1.5 Rozšiřující moduly . . . 16
1.1.6 Doplňkové vybavení . . . 17
1.2 Software . . . 17
1.2.1 Operační systém Raspberry Pi OS . . . 17
1.2.2 Programovací jazyk . . . 18
1.2.3 Klient softwarového telefonu . . . 18
1.2.4 Motion . . . 19
1.2.5 Technologie online hovorů . . . 20
1.2.6 Videohovory . . . 21
1.3 Postup vývoje a volba video řešení . . . 21
1.3.1 Překážky hardwarové . . . 21
1.3.2 Softwarové problémy . . . 22
1.3.3 Užité řešení . . . 22

2 Návrh softwarového řešení práce **28**
2.1 Diagram řešení . . . 28
2.2 Ovládací skript doorbell.py . . . 28
2.3 Doplňková třída lock.py . . . 32
2.4 Use-case diagram řešení . . . 34

3 Technologické postupy prototypování plastů a návrh šasi projektu **35**
3.1 Metody 3D tisku . . . 35
3.1.1 FUSED DEPOSITION MODELING (FDM) . . . 35
3.1.2 Stereolitografie (SLA) . . . 36
3.1.3 Volba metody tisku . . . 37
3.1.4 3D model šasi . . . 38

4 Postup instalace a kompilace softwarového řešení **41**
4.1 Instalace operačního systému a potřebných ovladačů . . . 41
4.2 Konfigurační soubory . . . 41

4.2.1 wpa_supplicant.conf 42
4.2.2 doorbell.cfg 43
4.2.3 .asoundrc ALSA konfigurace 43
4.3 Kompilace PJSIP s knihovnou PJSUA2 44
4.4 Vytvoření služby pro automatické spuštění 47
4.5 Konfigurace video funkcionalit 48
4.5.1 Instalace a konfigurace aplikace Motion 48

5 Montáž komponent do šasi **53**
5.1 Přední část šasi 53
5.2 Zadní část šasi 53
5.3 Finální montáž 54

6 Ověření funkčnosti **58**

7 Analýza z pohledu informační bezpečnosti **62**
7.1 Fyzické napadení přístroje 62
7.1.1 Odcizení zařízení 62
7.1.2 Únik konfiguračních souborů 62
7.1.3 Možnosti obrany 62
7.2 Útok na softwarové řešení 63
7.2.1 Sledování síťového provozu 63
7.2.2 Cílený útok na použité protokoly 64
7.2.3 Možnosti obrany 64

Závěr **65**

Literatura **66**

Seznam obrázků

1.1 Mikropočítač Raspberry Pi 4 s originálním popiskem 15
1.2 Mikropočítač Raspberry Pi Zero WH 24
1.3 Mikropočítač Raspberry Pi Zero 2W 25
1.4 ReSpeaker 2-Mics Pi HAT 26
1.5 Raspberry Pi kamera V2 27
2.1 Diagram softwarového řešení 28
2.2 Use-case diagram řešení 34
3.1 FDM Tiskárna Original Prusa MK4 36
3.2 FDM Tiskárna Anycubic Photon 37
3.3 FDM Tiskárna Bambu Lab X1-Carbon 38
3.4 Model šasi zvonku 39
3.5 Rozdělení modelu šasi 40
4.1 Uvodní obrazovka Raspi-config 42
4.2 Sekce interface Options v Raspi-config 51
4.3 Živý stream z kamery pomocí Motion 52
5.1 Instalace komponent do přední strany šasi 55
5.2 Rozložené šasi s komponenty 56
5.3 Zkompletované fyzické řešení zvonku 57
6.1 Záznam komunikace SIP protokolu, zachycený pomocí tcpdump . . . 58
6.2 Záznam RTP toku dat spolu s DTMF kódy 59
6.3 Konzole RPi: Registrace SIP 59
6.4 Administrace Odorik.cz - seznam přihlášených SIP klientů 60
6.5 Konzole RPi: Zahájení hovoru 60
6.6 Konzole RPi: Dokončení spojení hovoru 60
6.7 Konzole RPi: Přenos DTMF kódů a spuštění kódu po ověření 61
6.8 Wireshark záznam - příjem JPEG obrázků skrze HTTP 61

Seznam výpisů

2.1 doorbell.py -Volba tlačítka a konfiguračnáho souboru 29
2.2 doorbell.py - načtení konfigurace do prostředí knihovny 30
2.3 doorbell.py - Hlavní cyklus s knihovnou PJSUA2 31
2.4 lock.py 33
2.5 call.py - přidaný kód pro zpracování DTMF kódů a detekci stavu hovoru 33
4.1 wpa_supplicant.conf 43
4.2 doorbell.cfg 43
4.3 Konfigurační soubor .asoundrc 44
4.4 config_site.h 45
4.5 user.mak 45
4.6 voip.service 48

Úvod

Nové tisíciletí přineslo spoustu změn a jednou z nich je i čtvrtá průmyslová revoluce, takzvaný Průmysl 4.0. Hlavní myšlenkou je zaměření na optimalizaci a automatizaci výroby a všech procesů s tím spojených. Jde primárně o kontrolu kvality výrobků nebo snahu snižovat provozní náklady inteligentním řízením klimatizací a osvětlení provozů. Tento trend zjednodušení a automatizace opakovaných činností se přesouvá i mimo korporátní sféru. Pod marketingovými názvy jako chytrá domácnost nebo IoT se do bytových prostor instalují senzory vlhkosti, pohybu, deště či jiných podmínek. Spolu s chytrými spotřebiči, tedy zařízeními, která je možné vzdáleně spouštět a odečítat jejich stav, tak vzniká síť vzájemně komunikujících zařízení s velkou možností automatizací a pravidel. Nejčastějšími reakcemi bývá ovládání světel, pozicování rolet nebo zabezpečení objektů. Do poslední kategorie se řadí i předmět mé práce, chytrý zvonek využívající technologii VOIP pro oboustrannou hlasovou komunikaci, jednosměrné video a přenos kódu pro odemčení dveří.

V 1. kapitole se věnuji výběru vhodného zařízení a softwaru, který na něj následně aplikuji. Rozebírám zde také důvody a postup zvoleného způsobu přenosu videa. V kapitole druhé rozebírám samotné softwarové řešení a ukazuji vytvořené zdrojové kódy. Dále v kapitole 3. provádím rozbor vývoje šasi zvonku, které je vyrobeno z PLA plastu pomocí 3D tisku. Navazuji postupem instalace softwaru k dosažení mnou navrhovaného řešení v kapitole 4. a pokračuji popisem postupu vložení komponent do šasi v kapitole 5. Následně v kapitole 6. ukazuji ověření funkčnosti celého řešení. Poslední, sedmá kapitola se věnuje bezpečnostní analýze, včetně výčtu možných problémů a jejich řešení.

1 Výběr hardwaru a softwaru

Volba celkové platformy je pro efektivní řešení zadání velmi důležitá. Software i hardware musí být velmi dobře kompatibilní a umožňovat tak dlouhodobý běh bez nutných restartů zařízení nebo neustálých oprav. V této kapitole tedy rozebírám výběr nejvhodnější možné kombinace.

1.1 Hardware

V rámci jednoduššího postupu je vhodné si nejdříve zvolit samotné zařízení, na které následně navrhneme funkční a efektivní softwarové řešení. Výběr hardwaru je upřednostněn z důvodu větší složitosti. Díky velkému rozšíření VoIP technologií je totiž možné vytvořit funkční aplikaci snad pro každé častěji používané zařízení. Během práce se mi však podařilo narazit na problémy s kompatibilitou a na nestabilní funkce ve zvolené knihovně. Tyto nesrovnalosti tak vedly k lehké úpravě řešení oproti původnímu plánu.

1.1.1 Parametry výběru

Při volbě vhodného zařízení pro toto nasazení musíme brát v potaz hned několik parametrů. Použitá HW technologie musí být schopna nepřetržitého provozu i ve ztížených podmínkách. V exteriéru, běžné lokaci pro zvonek, je během roku velký rozsah teplot, což pro nás znamená volbu odolného zařízení s co nejmenší tepelnou stopou. Situaci ztěžuje i potřebná odolnost vůči vlhku, takže se musíme obejít bez aktivního chlazení.

Zvonek musí být také energeticky úsporný, aby měl i v horizontu let zanedbatelné provozní náklady a v případě požadavku na běh i při výpadku hlavního napájení zbytečně neplýtval uloženou energií v systému záložního zdroje.

Z hlediska portové výbavy je cílem dosáhnout co největších možností konektivity při zachování kompaktnosti. Klíčová je Wi-Fi konektivita pro snadnější instalace nebo možnost rozšíření o externí moduly, jako je kamera, reproduktor, klávesnice nebo modul pro kabelové připojení do datové sítě.

1.1.2 Specifikace modelu

Výslednou zvolenou možností je řada produktů **Raspberry Pi** z projektu britské neziskové organizace Raspberry Pi Foundation.

Tyto výrobky se vyznačují nejen vysokou efektivitou díky procesorům postaveným na architektuře ARM, ale i nízkou cenou, která je organizací záměrně držena při

zemi s cílem zpřístupnit rozumně výkonné mikropočítače velkému spektru uživatelů, a to včetně studentů a vzdělávacích organizací.

Je však nutno poznamenat, že v průběhu práce na projektu došlo k zásadnímu zhoršení situace na trhu s mikropočítači. Kvůli nedostatku čipů totiž výrazně poklesla výroba na straně Raspberry Pi Foundation a v roce 2023 tak není možné získat jakýkoliv z jejich modelů za běžnou cenu. Všechna zařízení jsou dostupná pouze na bazarových portálech v podobě nabídek od překupníků, a to za násobky běžných cen.

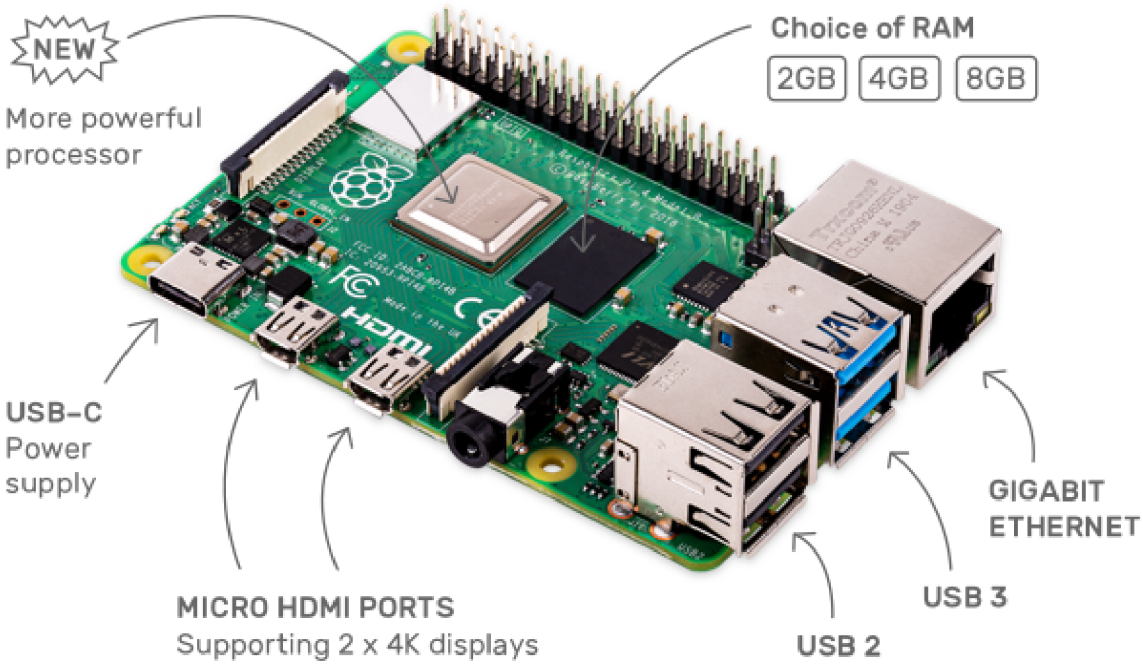
Možnosti konektivity záleží na vybraném modelu, ty se liší velikostí, spotřebou i cenou. Pro potřeby této práce přicházely v úvahu tři varianty.

Raspberry Pi 4

Aktuální vlajková loď řady zařízení RPI. Vzhledem k ceně kolem 1000 Kč[1] dosahuje s čtyřjádrovým Cortex procesorem skvělého poměru cena/výkon[2]. Ve vyšších variantách s větší kapacitou RAM jsou tyto modely využívány i pro domácí nasazení ESXi virtualizace. I při velikosti platební karty a výšce ne více než 2 centimetry poskytuje dva grafické výstupy, 1Gbit RJ45 pro připojení do LAN nebo plnohodnotné USB porty pro připojení periferií a další rozšíření funkcionality. Raspberry mikropočítače podporují i rozšíření pomocí rozšiřujících karet, nebo ovládání jiných zařízení, jako jsou relé. K tomu slouží GPIO *(General Input Output)* piny. U této nové generace došlo i k výměně napájecího konektoru. Běžné Micro-USB bylo nahrazeno novějším, bytelnějším a pohodlnějším konektorem typu USB-C.

Raspberry Pi Zero WH

Pi Zero je na opačném konci spektra produktové řady. Jedná se o zmenšenou, slabší variantu klasických Pi počítačů, zachovává si však všechny základní funkce a i přes slabší procesor a menší portovou vybavenost jde o reálně využitelné zařízení. Navíc, díky ceně kolem 450 Kč[3], jde o ideální zařízení pro nasazení do riskantnějších projektů, kde je vyšší šance ztráty zařízení. S nižším výkonem je spojená i značně nižší spotřeba. Kompromis ceny a výbavy se odehrál na poli USB vstupů, které jsou nyní jen ve formě Micro-USB s nutným použitím OTG redukce, a u síťové konektivity, kde zcela chybí LAN port a jsme tak odkázáni na další Micro-USB redukce, nebo Wi-Fi připojení. GPIO piny však zůstávají nedotčené a je možné je použít pro rozšiřovací karty nebo pro ovládání externí logiky. U modelů Zero zatím nedošlo k obměně napájecích konektorů, takže i zde je použito USB standardu Micro, nikoliv typ C.



Obr. 1.1: Mikropočítač Raspberry Pi 4 s originálním popiskem

Převzato z [1]

1.1.3 Raspberry Pi Zero 2W

Model Zero 2W je modernějším nástupcem Zero W. Primární změnou je výměna procesoru. Počítač si polepšil nejen zvýšením počtu jader z jednoho na tři, ale také přechodem na novější architekturu ARM8, díky které je viditelné výrazné zvýšení výkonu jednotlivých jader.

Jelikož šlo primárně o aktualizaci čipu, Pi Foundation vyvinula tento model jako plně kompatibilní náhradu původní verze Pi Zero. Portová výbava, velikost samotného zařízení a rozložení konektorů tak zůstaly beze změny.

1.1.4 Zvolené zařízení

Původně použitým zařízením bylo Raspberry Pi Zero první generace. Primárním důvodem byla menší velikost a dostatečné portové osazení pro toto použití, které vyžaduje pouze Wi-Fi, nebo LAN připojení a GPIO pro osazení rozšiřujícího modulu. Dalšími faktory při volbě pak byla nižší spotřeba elektrické energie a cena.

Během projektu však došlo k přechodu na model Zero 2W, a to kvůli nedostatečnému výkonu první generace. Tento deficit se projevoval na extrémně dlouhých časech kompilace knihoven, ale také při snaze implementovat video funkce pro splnění zadání práce. Výměna zařízení za novější model stejného formátu díky již zmíněné fyzické kompatibilitě nebyla zásadní překážkou a kromě lehké změny v kompilačních

parametrech tak šlo o velmi rychlou změnu, která ušetřila čas při vývoji kompletního softwarového řešení.

Navíc díky zvýšení výkonu zařízení necháváme prostor a rezervu pro případné rozšíření funkcionalit zvonku nebo přidání další, zcela odlišené role zařízení. Například může RPi sloužit nejen jako zvonek, ale i jako čipová jednotka pro kontroly vstupu do budovy nebo jako pohybový či teplotní senzor pro chytrou domácnost.

Volbou Pi Zero 2W sice získáváme úsporu financí a prostoru, ztrácíme však možnost použití LAN připojení, které by zajistilo lepší stabilitu komunikace a s POE *(Power Over Ethernet)* adaptérem i možnost provozovat datovou komunikaci spolu s napájením po jednom přívodním kabelu.

1.1.5 Rozšiřující moduly

Počítače Raspberry mohou být, jak jsem již ostatně zmínil v sekci výše, rozšiřovány přídavným vybavením. Tyto moduly jsou nejčastěji ve formátu desek s integrovanými obvody, dalšími konektory a specializovanými senzory. Připojují se na GPIO piny, které jim zabezpečují datovou komunikaci, fyzické upevnění a napájení. V případě vyšší energetické náročnosti přídavného zařízení je na tomto obsažen i USB port, který napájí jak modul samotný tak původní RPi. Všechna kompatibilní rozšíření se kvůli stylu umístění na zařízení nazývají **HAT**s, což v angličtině znamená čepice/klobouk.

Vyžadované parametry

Pro splnění zadaného cíle musí být zvolen vhodný doplňkový modul tak, aby splňoval potřebné parametry a doplňoval chybějící funkce a hardwarové vybavení.

V tomto konkrétním případě je třeba rozšířit primárně funkcionalitu v oblasti audia. Pi Zero totiž není v základu vybaveno žádným audio vstupem či výstupem. Obě tyto funkce pro správné fungování zvonku s oboustrannou komunikací logicky potřebujeme.

ReSpeaker 2-Mics Pi HAT

HAT použitý v mém řešení se nazývá ReSpeaker 2-Mics Pi HAT. Jeho zvolení proběhlo primárně na základě dostupnosti na fakultě a doporučení vedoucím práce. Jedná se o modul vyvinutý společností Seeed, která se zaměřuje na různé rozšiřující moduly a chytré senzory pro svět IoT.

V tomto případě se jedná o kartu, přidávající nejen pole stereo mikrofonů a zesílený výstup pro reproduktor, ale i další rozšiřující konektory Grove, které je možné

použít pro ovládání externí logiky nebo například relé[5]. Tyto proprietární konektory přímo od společnosti Seeed totiž předávají dál GPIO piny přímo z Raspberry. Tuto funkcionalitu využijeme pro připojení relé (simulace dveří objektu se zvonkem). Z přídavné desky také uplatníme mikrofony a výstup pro reproduktor. Na rozšíření je přítomno i tlačítko, to v rámci vývoje sloužilo jako spouštěč hovoru. Pro výsledné nasazení je však použito externí tlačítko, napojené do stejného Grove konektoru jako výše zmíněné relé.

Raspberry Pi kamera V2.1

Jelikož má výsledné zařízení podporovat i přenos videa, musíme jej vybavit kamerou pro jeho záznam. Volba padla na rozšíření přímo od výrobce mikropočítače. Důvodem zvolení tohoto modulu byla hardwarová podpora a dobrý poměr cena/výkon. Kamera se připojuje přímo do desky počítače skrze vyhrazený konektor a za cenu kolem 500 korun nabízí možnost až FUllHD videa. Takového rozlišení však v tomto projektu nedosáhneme.

Vedoucím práce byla pro projekt poskytnuta první generace doplňku, přesněji verze 1.2. Pro urychlení vývoje jsem však používal dvě sady mikropočítačů a svou vlastní kameru druhé generace, kterou jsem se rozhodl do výsledného zařízení nasadit kvůli lepší kvalitě obrazu i při nižším rozlišení výstupu.

1.1.6 Doplňkové vybavení

Ke splnění cíle potřebujeme i doplňkový externí hardware. Jde o malý reproduktor a napájecí adaptér. Obě zařízení byla sice dodána vedoucím práce, ale pro snadnější vývoj a vyšší kvalitu zvuku byla nahrazena jiným hardwarem. Primárně šlo o Micro-USB kabel s koncovkou do pravého úhlu a lehce větší, kvalitnější reproduktor.

1.2 Software

Na základě zvolené hardwarové platformy musíme nyní vybrat vhodný programovací jazyk a software na něm postavený. V potaz musíme brát míru optimalizace jazyka pro ARM, procesorovou architekturu použitou na RPi i úroveň a míru výběru v rámci doplňkového softwaru a knihoven.

1.2.1 Operační systém Raspberry Pi OS

Počítače Raspberry Pi podporují více operačních systémů. Byť nejčastěji nasazovaným je jeden z velké palety linuxových distribucí[7], existuje i varianta Microsoft Windows podporující ARMové procesory, Windows 10 for IoT.

Pro stanovené účely však použijeme oficiálně doporučovanou distribuci linuxového operačního systému Raspberry Pi OS z rodiny debian-based systémů. Jde o prostředí speciálně připravené pro práci s Pi produkty a obsahuje všechny potřebné ovladače pro správný běh komponent.

Operační systém lze nainstalovat ve 3 variantách. Čistý systém s grafickým rozhraním, systém s přidanými doporučenými aplikacemi nebo tzv. lite verze s minimem softwaru nutným pro běh.

Pro tento projekt byl nejprve užit Debian with desktop, tedy plná verze včetně standardních balíčků. Kvůli problémům s výkonem na Pi zero první generace však došlo k přechodu na verzi lite bez grafického rozhraní. Ani tato optimalizace však nebyla dostatečná, takže musela nastoupit již zmíněná změna hardwaru na druhou generaci. Zmenšená verze systému však zůstala i nadále.

Zvolený doplňkový hardware, zejména kamera a ReSpeaker HAT, omezil volbu verze operačního systému. Seeed studio totiž přestalo vydávat aktualizace ovladačů a Pi Camera v1 a v2 ztratila oficiální podporu. Posledním funkčním systém je ještě stále aktualizovaný, ale již ne aktuální Debian 10 „Buster“. Existují sice neoficiální snahy o podporu ovladačů komunitou vývojářů i pro novější Debian 11, nebylo však možné je nasadit bez kompromisů na stabilitě, která je pro tento projekt kritická.

1.2.2 Programovací jazyk

Jazyk, který použijeme pro napsání ovládacího softwaru je klíčovou volbou v rámci celého projektu. Pro dodržení požadavků musí náš kandidát mít schopnost fungovat v rámci zásad OOP *(objektově orientované programování)*, ale zároveň umožňovat jednoduché skriptování pro základní ovládací soubor, který budeme spouštět po startu systému.

Python 3

Platforma Raspberry preferuje jako hlavní jazyk pro práci Python. Vzhledem k tomu, že splňuje i výše stanovené podmínky a zároveň je pro něj dostupné velké množství knihoven[8], bude Python ve verzi 3 použit pro vytvoření ovládacího softwaru tohoto komunikátoru.

1.2.3 Klient softwarového telefonu

Výběr klienta pro uskutečnění telefonního hovoru je do značné míry ovlivněn doporučením vedoucího práce. I bez něj je však množství spolehlivě funkčních klientů

s podporou Python knihoven vcelku omezený. Dalším argumentem pro zvolený program byly i časté zmínky o jeho použití ve velkých projektech na veřejných fórech poskytovatelů služeb VoIP operátora.

PJSIP

PJSIP je open source VoIP klient napsaný v jazyce C. Je hojně využívaný i ve velkých telefonních systémech, jako jsou call centra nebo úřední infolinky. Toho dosahuje primárně integrací PJSIPu do frameworku Asterisk[10]. PJSIP podporuje kromě klasických hovorů i videohovory. Podpora videa sice byla dalším důvodem k jeho volbě, kvůli problémům s podporou hardwarové akcelerace a celkově nestabilitou videořešení PJSIP pro RPi však nakonec tato část funkcionality nebyla v projektu použita.

PJSUA

PJSUA je knihovna pro komunikaci s výše zmíněným programem PJSIP. V mé práci je použita novější varianta s vyšší abstrakcí, PJSUA2. Jde o high-level API, tedy komunikační rozhraní pro jiný software, které pracuje na úrovni příkazů pro PJSIP bez zbytečného odkrývání vnitřního fungování. Díky wrapperu pro Python přes něj můžeme vcelku pohodlně uskutečňovat hovory a přijímat potřebné kódy pro aktivaci funkce zámku.

PJSUA2 je už objektově orientovaná abstrakce nad původním PJSUA API a spojuje možnosti vytváření SIP sezení s možností ovlivňovat směrování hovorů a udržování SIP účtů online[11].

1.2.4 Motion

Jelikož nedošlo k použití video funkcí přímo v knihovně PJSIP, musíme vybrat i vyhrazené řešení pro přenos videa. Po dlouhém testování možností přenosu videa přímo v rámci hovoru nakonec padla volba na stabilnější přenos videa skrze HTTP MPEG stream. Tento přenos zajišťuje open-source projekt Motion, který je primárně určený pro sběr a následné zpracování videa z IP kamer, nebo fyzických, lokálně připojených kamer. Jak již název napovídá, software se specializuje na detekci pohybu na obraze. V tomto projektu však byla využita jen schopnost odesílat videostream při využití hardwarové akcelerace a jednoduché konfigurace.

Motion byl zvolen i kvůli velkým možnostem rozšíření funkcionalit projektu skrze detekce a následné reakce na zjištěné aktivity v obraze.

1.2.5 Technologie online hovorů

Voice over IP

Technologie Voice over IP, zkráceně VoIP je řešením přenosu živého telefonního hovoru přes sítě protokolu IP. Na rozdíl od hovoru vedeného po běžné, analogové telefonní síti je zvukový signál převáděn na digitální tok dat už na koncových zařízeních a je přenášen po lokální síti nebo internetu až k druhé straně, kde je opět převeden do zvukového signálu.

Hlavním důvodem nasazení této technologie v korporátním, ale i soukromém sektoru je možnost výrazných ekonomických úspor a velká flexibilita ve směrování, nahrávání a automatickém zpracování hovorů.

K propojením VoIP hovorů „ven“ do sítí mobilních operátorů slouží VoIP operátoři, kteří poskytují své ústředny a za zpravidla nižší ceny umožňují příchozí i odchozí hovory v rámci veřejné telefonní sítě.

VoIP operátor

Společností poskytujících služby VoIP operátora je na českém trhu velké množství. Pro tuto práci byla vybrána služba odorik.cz od společnosti miniTEL s.r.o. Toto řešení jsem zvolil z důvodu již existujících dobrých zkušeností s platformou a srozumitelné dokumentace, kterou ještě doplňuje ochotná podpora ze strany vývojářů aplikace. Nespornou výhodou je také funkce volání přímo v rámci sítě Odorik, které není zpoplatněno a je ideální pro testování funkčnosti klienta a hardwaru bez zbytečné finanční zátěže.

Protokol SIP

Nejčastěji používaným protokolem pro zahájení a správu internetových VoIP hovorů je protokol SIP *(Session Initiation Protocol)*. Jde o protokol na úrovni aplikační vrstvy s podobnou strukturou dotazů a odpovědí, jako má protokol HTTP. Celou specifikaci standardu nalezneme v dokumentu RFC 3261[12].

Jak již název napovídá, jde pouze o signalizační protokol, jeho úkolem je tedy vykomunikovat informace pro zahájení spojení a udržovat aktuální informace o stavu hovoru po jeho navázání. Zároveň slouží jako nositel dat níže popsaných DTMF kódů. O samotný přenos audio signálu se starají protokoly SDP *(Session Description Protocol)*[13] a RTP *(Real-Time Applications Protocol)*[14], kde SDP přenese detaily o vlastnostech zahajovaného streamu dat, který je následně v gesci RTP.

Alternativou protokolu SIP je protokol H.323, vyvinutý standardizační agenturou OSN pro telekomunikace. SIP je však pro svou komplexnost výrazně více používán a je nasazen i v řešení mého projektu.

1.2.6 Videohovory

Protokol SIP umožňuje správu nejen audio hovorů, ale i spojení s podporou videa. Princip funkcionality je stejný, jen se v komunikaci o dostupných možnostech spojení přenese i požadavek k návazní video přenosu a následně proběhne vyjednávání o dostupných rozlišeních a kodecích. Technické specifikace pak jsou již v režii protokolů SDP a RTP. Důležitá je i informace, že součástí této dohody o technickém řešení přenosu je i počet snímků za sekundu, které budou obě strany videohovoru posílat. Pokud není tento počet snímků dodržen, hovor se rozpadne. Tato vlastnost byla hlavním důvodem, proč se v rámci této práce video přesunulo na samostatný stream oddělený od hovorů. Raspberry totiž nebylo schopno dodržet stabilní počet snímků a hovor se tak po aktivaci videa rozpadl.

1.3 Postup vývoje a volba video řešení

Původním plánem pro přenos videa ze zvonku k majiteli bylo použít možnosti videa přímo v rámci VoIP telefonie. Výhodou by bylo zjednodušení práce se zařízením - uživatel zvedne hovor a rovnou vidí i osobu u zvonku, nebo také nižší náročnost na údržbu - méně užitého softwaru, který je nutno aktualizovat a překonfigurovávat.

Práci na tomto řešení bylo vynaloženo nemalé úsilí a šlo tak o podstatnou část času stráveného nad touto prací.

1.3.1 Překážky hardwarové

První limit, který se projevil už při snaze zlepšit audio kvalitu na nyní ve VoIP telefonii standardně používaný kodek PCMA byl výkon Raspberry Pi. V prvotní fázi totiž bylo užíváno Pi Zero WH, tedy první generace této řady, která disponovala pouze jedním jádrem procesoru staré verze architektury ARM. Nedostatečné kapacity procesoru se projevovaly nejen na neschopnosti stabilní práce s kvalitnějšími kodeky zvuku, ale i na extrémně dlouhé kompilaci knihovny PJSIP a její nástavby PJSUA2. Kompilace, která běžně zabrala i 4 hodiny, se navíc musela kvůli nedostatečné dokumentaci a neaktivní podpoře ze strany vývojářů mnohokrát opakovat.

Při aktivaci podpory během kompilace pak knihovna vytížila procesor na maximum i bez jakékoliv práce s videem nebo jeho použití v aktuálním hovoru. Po snaze provést optimalizace byla nakonec zvolena možnost přechodu na druhou generaci, tedy Pi Zero 2W.

Na novém zařízení již bylo možné knihovnu spustit, vykazovala však nestandardní softwarové potíže.

1.3.2 Softwarové problémy

Knihovna způsobovala těžko opravitelné problémy, kdy dvakrát provedená kompilace se stejnými parametry sice nehlásila žádné chyby, ale při spuštění knihovny vykazovala rozdílené chování a končila na jiných chybách a pádech běhu.

Po překonání většiny softwarových překážek se mi podařilo alespoň přenést jeden videosnímek ze zvonku ke klientovi na platformě windows. Toto spojení se však záhy rozpadlo, protože ani nový hardware nebyl schopen dodat dostatečné množství snímků za sekundu, aby naplnil očekávání druhé strany. PJSIP totiž sice umí provést kompilaci se zapnutou podporou pro ffmpeg, využití jeho kodeků s hardwarovou akcelerací ale v knihovně není řádně zadokumentováno a nebyl jsem schopen jej svými vlastními silami zprovoznit.

Než jsem se rozhodl zvolit alternativní řešení přenosu videa, zkusil jsem oslovit i experty na VoIP problematiku s žádostí o pomoc s knihovnou PJSIP a jejím užitím na mém hardwaru. Oslovil jsem tak například majitele společnosti Odorik, jejíchž služeb VoIP operátora jsem ve své práci využil. Jejich postoj potvrdil mé obavy o nedostatku výkonu a nastínil úvahy o případných problémech samotné knihovny PJSIP na HW užitém v mikropočítačích RPi. Bylo mi doporučeno se odklonit od přenosu videa skrze samotný VoIP hovor a soustředit se na zachování jeho stability a funkčnosti s hlasem.

Pokusil jsem se oslovit i zahraniční IT experty skrze platformu Stack Overflow s kompletním popisem mých problémů. Tento popis jsem umístil i na mailing list vývojářů knihovny. V obou případech však má vlákna zůstala bohužel bez odpovědi. Věřím tak, že jsem vyvinul maximální úsilí pro co nejpřesnější splnění zadání.

1.3.3 Užité řešení

Zvoleným řešením je použití kamery a grafického čipu mikropočítače k provozu webkamery, která neustále přenáší data do lokální sítě skrze protokol HTTP. Tento postup sice vytěžuje zařízení neustále, i mimo spuštěný hovor, ale dosahuje mnohem vyšší efektivity práce a tak i značně nižšího využití procesoru. Máme tak jistotu, že i při přenosu videa bude spolehlivě fungovat hlasový hovor a možnost poslat kód pro odemčení.

Díky této metodě přenosu je navíc video značně kvalitnější. Nemusí být totiž přenášeno s tak nízkou latencí, jakou vyžadují hlasové služby a je možné strávit více času přenosem a kódováním jednotlivých snímků. V moderních sítích je navíc časté řešení QoS, kde má VoIP telefonie vždy přednost před ostatními typy dat. Máme tedy jistotu, že i v případě slabšího připojení bude hlasový hovor spojen a může tak dojít ke komunikaci zvonícího s majitelem zvonku.

Zvolený software Motion je velmi rozšířený ve světě zabezpečovacích a pohybový systémů založených na Raspberry Pi. Jeho opensource licence nám umožňuje užití bez licenčních poplatků a jeho propracovanost nechává velký prostor pro zdokonalení nebo rozšíření mého projektu případným zájemcem. Běžná konfigurace pro start samotného přenosu je však velmi jednoduchá a i přes velkou komplexnost celého softwaru zabere jen pár minut a práci v pouze jednom souboru.

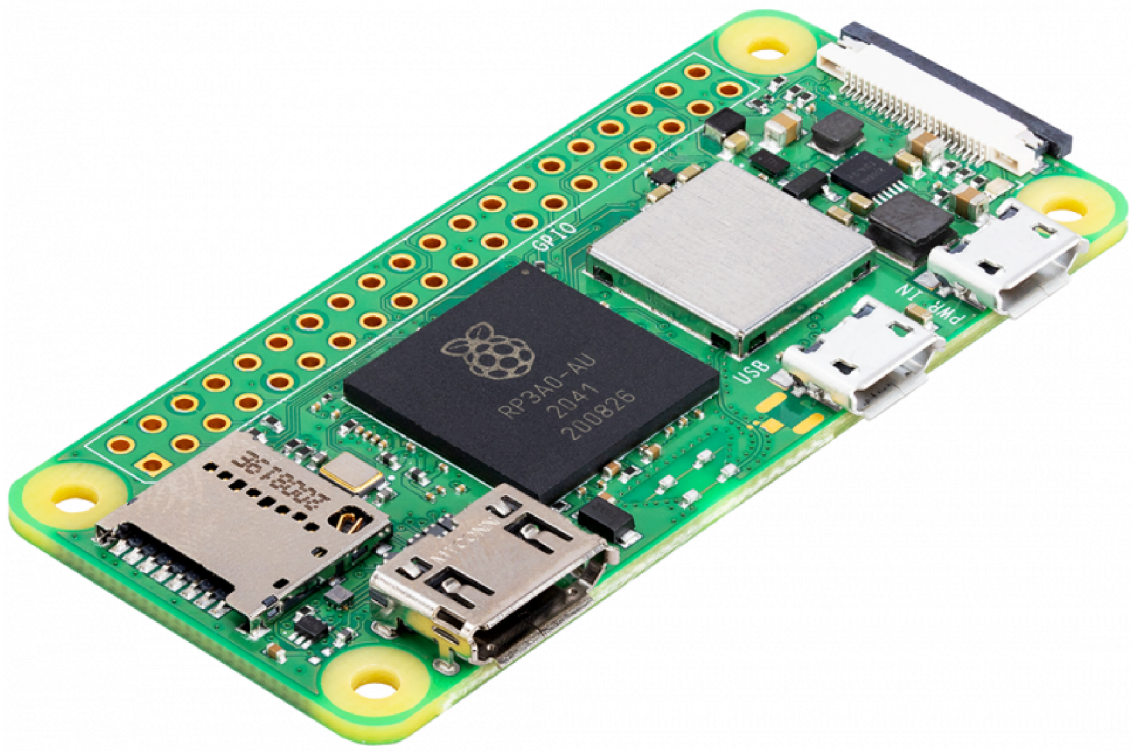
Jsem tedy přesvědčen, že zvolené řešení dostatečně supluje a v mnohém i překonává původně navrhovanou možnost.

DTMF kódy

Technologie kódů DTMF, česky často nazývaná jako kódová volba, je metoda signalizace stisknutého čísla nebo znaku během již zahájeného hovoru. V běžných telefonních sítích jsou tyto kódy přenášeny pomocí tónů na předem stanovené frekvenci, které jsou generovány koncovým zařízením, na němž došlo ke stisku číslice[15]. V případě hovorů přes VoIP jsou již tyto kódy přenášeny digitálně bez slyšitelného tónu, jako nositele. Kódy DTMF může v online hovoru přenášet jak protol RTP jako slyšitelné zvuky, tak protokol SIP promocí SIP INFO metody.

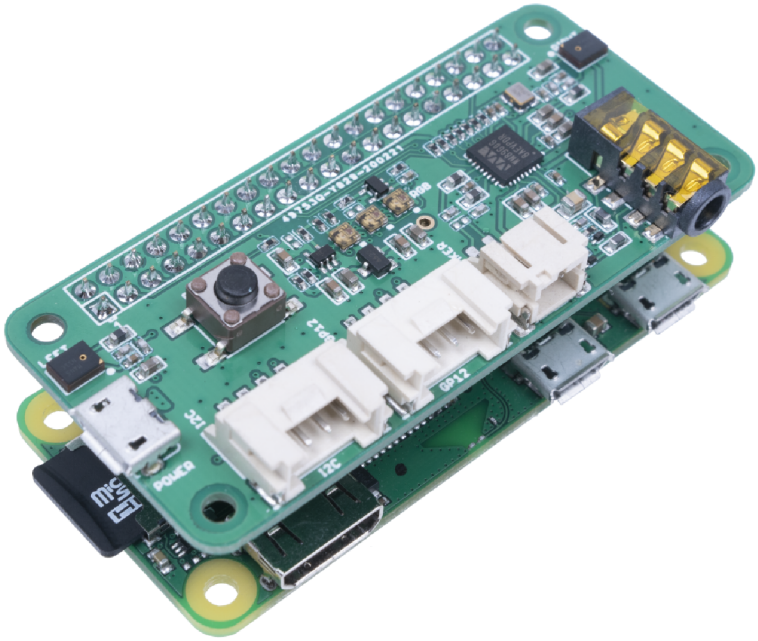
Obr. 1.2: Mikropočítač Raspberry Pi Zero WH

Převzato z [3]



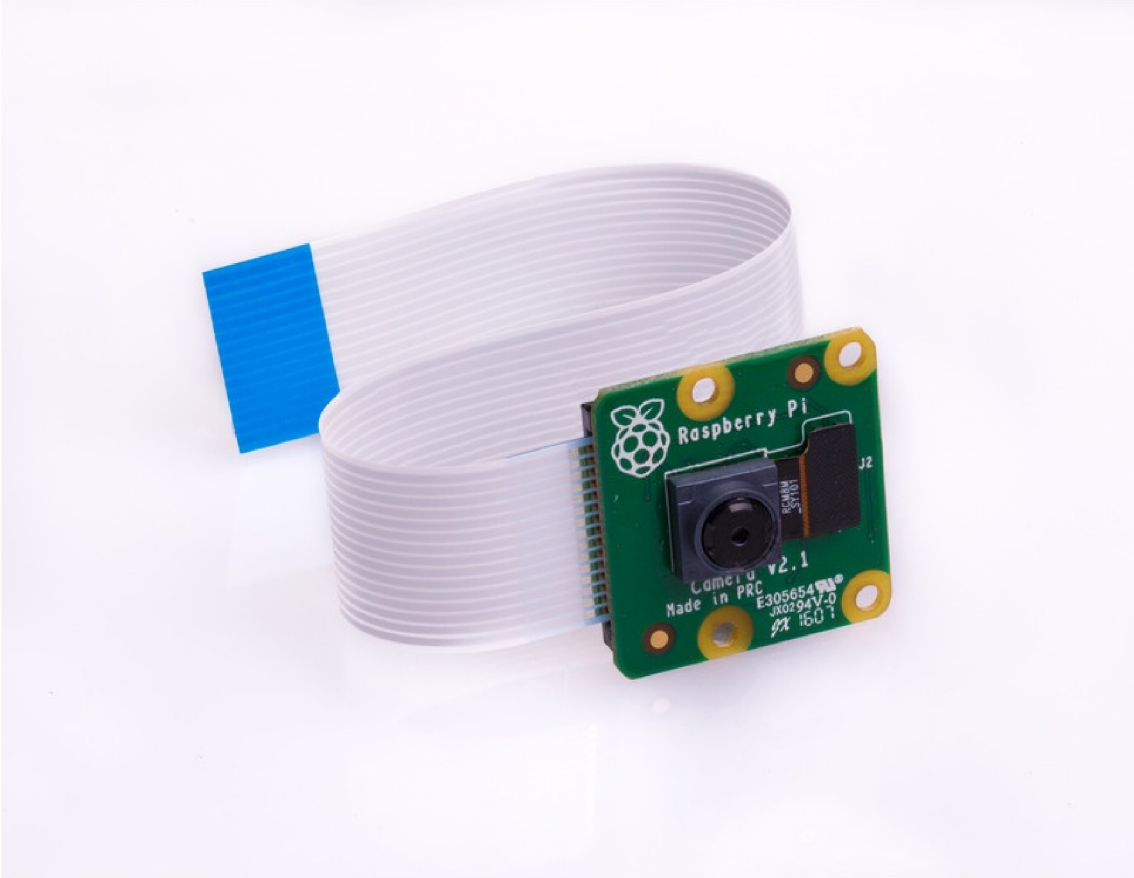
Obr. 1.3: Mikropočítač Raspberry Pi Zero 2W

Převzato z [4]



Obr. 1.4: ReSpeaker 2-Mics Pi HAT

Převzato z [5]



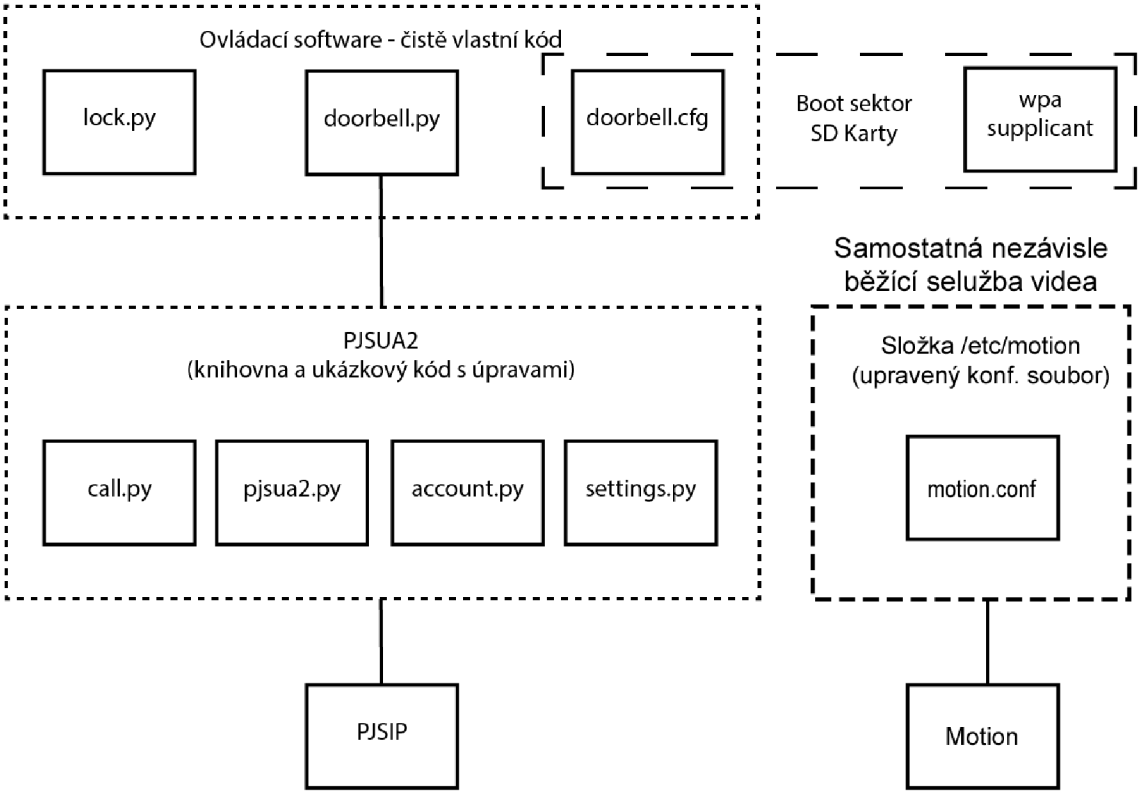
Obr. 1.5: Raspberry Pi kamera V2

Převzato z [6]

2 Návrh softwarového řešení práce

Softwarové řešení je složeno z vlastního ovládacího skriptu doorbell.py, přídavné třídy lock.py a konfiguračních souborů. Konfigurační soubory pak slouží pro nastavení systému a prostředí pro běh knihovny PJSUA.

2.1 Diagram řešení



Obr. 2.1: Diagram softwarového řešení

2.2 Ovládací skript doorbell.py

V této podkapitole provádím rozbor hlavního ovládacího skriptu, který je spuštěn při samotném startu systému a obstarává funkcionalitu a volání ostatních tříd či knihoven. Kód je vložen postupně, vždy s komentářem pro danou sekci. Kompletní softwarové řešení je dostupné na GitHub stránce `https://github.com/DJIronic/voipdoorbell`

```
BUTTON = 13
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
write=sys.stdout.write

configPath = sys.argv[1]
if(not exists(configPath)):
    print("Path to config is not set or is not valid path!")
    quit()
```

Výpis 2.1: doorbell.py - Volba tlačítka a konfiguračnáho souboru

V úvodu souboru je umístěna konfigurace tlačítka, které v případě stisku zahájí hovor. V tomto případě se jedná o tlačítko GPIO 13, tedy o externí tlačítko připevněné k obalu celého zařízení. GPIO PIN 13 je jeden ze dvou kontaktů, který je vyveden z ReSpeaker HAT jako součást Grove konektoru. Zároveň načítáme i konfigurační soubor, který je předáván jako argument při spuštění skriptu. Dochází ke kontrole, zda je cesta k souboru platná nebo neexistující.

```
# Load config
userConfig = configparser.ConfigParser()
userConfig.read(configPath)

# Prepare lock class
lockInst = lock.Lock(userConfig['lock']['passcode'])

# Accounts
accList = []
ep = endpoint.Endpoint()
ep.libCreate()
# Default config
appConfig = settings.AppConfig()

# Threading
appConfig.epConfig.uaConfig.threadCnt = 0
appConfig.epConfig.uaConfig.mainThreadOnly = True

# Config

appConfig.epConfig.uaConfig.maxCalls = 1
appConfig.epConfig.medConfig.channelCount = 1
# Set the channel count to 1 (mono)
appConfig.epConfig.medConfig.clockRate = 8000
```

```
appConfig.epConfig.medConfig.sndClockRate = 8000
# Set the sample rate to 8000 Hz
appConfig.epConfig.medConfig.ptime = 140
ep.libInit(appConfig.epConfig)
ep.transportCreate(appConfig.udp.type, appConfig.udp.config)


#configure the library
config = pj.AccountConfig()
config.idUri = userConfig['pjsua2']['id']
config.regConfig.registrarUri = userConfig['pjsua2']['registrarUri']
#config.regConfig.contactParams = "sip:736370@sip.odorik.cz"




config.presConfig.publishEnabled = True

cred = pj.AuthCredInfo()
cred.realm = userConfig['pjsua2']['realm']
cred.scheme = userConfig['pjsua2']['scheme']
cred.username = userConfig['pjsua2']['username']
cred.data = userConfig['pjsua2']['password']

config.sipConfig.authCreds.append(cred)

if not endpoint.validateSipUri(config.idUri):
    print("ERROR IN ID URI")

if not endpoint.validateSipUri(config.regConfig.registrarUri):
    print("ERROR IN REGISTRAR URI")

if not endpoint.validateSipUri(config.regConfig.contactParams):
    print("ERROR IN CONTANT PARAMS")

account = account.Account()
account.create(config)
```

Výpis 2.2: doorbell.py - načtení konfigurace do prostředí knihovny

Dalším krokem je parse samotného konfiguračního souboru, připravení lock třídy a následné vytvoření prostředí knihovny PJSUA2. Důležitou neměnnou částí konfigurace je vynucení běhu v rámci jednoho vlákna a omezení na maximálně jeden probíhající hovor. Pro správnou funkčnost na novějších verzí OS a Pi zero bylo nutné

specifikovat i vzorkovací frekvenci, počet audio kanálů a velikost mezipaměti. Tyto hodnoty se však lišily v závislosti nejen na modelu RPi, ale i kus od kusu ReSpeaker hatu a byly tak vždy zjišťovány metodou pokus-omyl pro danou kombinaci HW a verze OS. Následně předáváme parametry SIP serveru a cíle hovoru knihovně a ověřujeme, že zadané hodnoty jsou platné. Pokud ano, dojde k vytvoření instance účtu s přichystanou konfigurací.

```
# Start library
ep.libStart()
ep.libHandleEvents(10)


# Initialize an ongoing_call variable before the loop
ongoing_call = None

while True:
    input = GPIO.input(BUTTON)
    if input == GPIO.HIGH:  # Button is pressed
        # Check if there's an ongoing call
        if ongoing_call is None or ongoing_call.isCallDisconnected():
            print("No call detected, creating a new one to "
            + userConfig['lock']['targetVoipUri'])
            call_param = pj.CallOpParam()
            call_param.opt.audioCount = 1
            call_param.opt.videoCount = 0

            ongoing_call = call.Call(account, userConfig['lock']
            ['targetVoipUri'], lock=lockInst)
            ongoing_call.makeCall(userConfig['lock']['targetVoipUri'],
            call_param)
        else:
            print("There's an ongoing call, can't make a new one.")
        ep.libHandleEvents(10)
    ep.libHandleEvents(10)
    time.sleep(0.1)  # Add a short delay to avoid excessive CPU usage
```

Výpis 2.3: doorbell.py - Hlavní cyklus s knihovnou PJSUA2

V závěru spustíme celou knihovnu PJSUA2 a v cyklu čekáme na zahájení hovoru stiskem tlačítka. V případě aktivace dojde k hovoru a po jeho ukončení se vracíme zpět na začátek cyklu a čekáme na opakovaný stisk tlačítka. Hovor je tedy možné ihned po skončení předchozího vyvolat znovu. V kódu je implementován i systém kontroly již běžícího hovoru. Pokud je tedy již navazováno spojení, nebo už existuje hovor, stisk tlačítka vyvolá jen zápis logu do konzole a nepokusí se spustit nový

hovor. Touto funkcí se tak chráníme před dvojitým stisknutím nebo jiným druhem špatného zacházení s tlačítkem, které je dostupné veřejnosti.

2.3 Doplňková třída lock.py

Vyrobená třída `lock.py` je doplňkovou třídou pro `call.py`. Jejím zadáním je zpracovat příchozí DTMF kódy, uložit je do stacku a v případě potvrzení provést validaci zadaného kódu vůči hodnotě v konfiguračním souboru. Je-li zásobník shodný s předdefinovaným heslem, spustí se kód stanovený v metodě. V tomto případě jde o vypsání informace o spuštění do konzole a přepnutí relé skrze GPIO PIN12, který je stejně jako PIN pro tlačítko dostupný skrze Grove konektor na ReSpeaker HAT. Systém využívá DTMF kód s # pro potvrzení dokončení zadávání a ∗ pro vyprázdnění stacku – reset zadaného kódu.

```
import time
import RPi.GPIO as GPIO

class Lock:
    def __init__(self, passphrase):
        self.passphrase = passphrase
        GPIO.setup(12, GPIO.OUT)
        self.stack = []


    def unlock(self):
        GPIO.output(12, GPIO.HIGH)
        print("==================== unlocked ====================")
        time.sleep(5)
        self.lock()

    def lock(self):
        GPIO.output(12, GPIO.LOW)
        print("==================== locked ====================")

    def addDigit(self, digit):
        self.stack.append(digit)

    def receiveOk(self):
        passcode = ""
        passcode = passcode.join(self.stack)
        print("Passcode is " + passcode)
        if(passcode == self.passphrase):
            self.unlock()
        else:
```

```
            print("Wrong passcode")

        self.stack = []

    def receiveReset(self):
        print("Resetting")
        self.stack = []
```

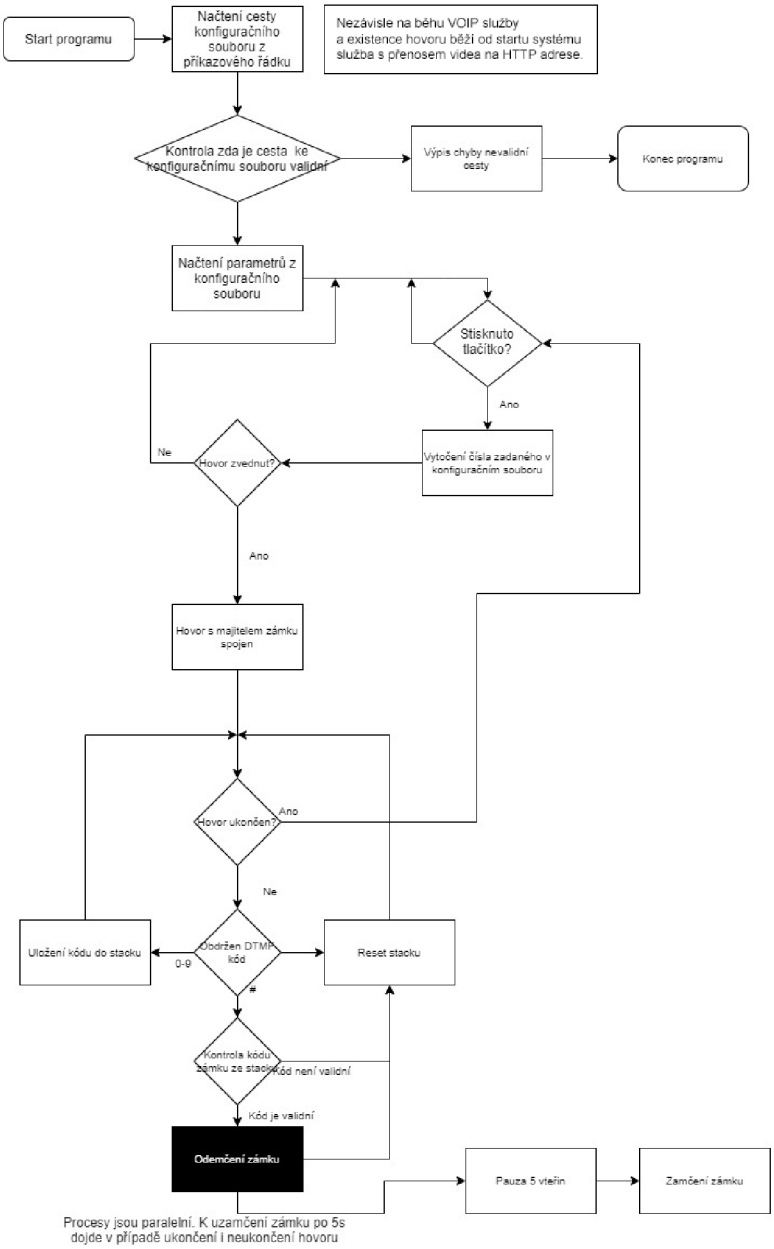
Výpis 2.4: lock.py

Pro fungování Lock třídy je třeba získavat data o DTMF kódech z probíhajícího hovoru. Třída call.py, která má průběh hovoru na starost, tak byla upravena, aby tyto kódy předávala. Zároveň byl do souboru přidán systém odpovědi na dotaz ohledně stavu hovoru. Tento prvek je použit pro výše zmiňovanou ochranu proti špatnému užití tlačítka. Přidaný python kód je uveden ve výpisu. Zbytek call.py byl vytvořen autorem knihovny.

```
    def onDtmfDigit(self, prm):
        print("Received DTMF digit: " + prm.digit)

        if(prm.digit == "#"):
            print("Sending OK")
            self.lockInst.receiveOk()
            return

        if(prm.digit == "*"):
            print("Sending reset")
            self.lockInst.receiveReset()
            return

        self.lockInst.addDigit(prm.digit)

    # Check for call disconnection to prevent
    #crashes when button is pressed repeatedly
    def isCallDisconnected(self):
        try:
         call_info = self.getInfo()
         return call_info.state in (pj.PJSIP_INV_STATE_DISCONNECTED,
         pj.PJSIP_INV_STATE_NULL)
        except pj.Error:
         return True
```

Výpis 2.5: call.py - přidaný kód pro zpracování DTMF kódů a detekci stavu hovoru

2.4 Use-case diagram řešení



Obr. 2.2: Use-case diagram řešení

3 Technologické postupy prototypování plastů a návrh šasi projektu

Dle zadání projektu má být celý zvonek uložen do šasi vlastního návrhu. V této kapitole tak popisuji design samotného obalu, osvětlím základy použité technologie a ukážu metodu instalace komponentů do výtisku. Výsledný 3D model je spolu se zdrojovým kódem zpřístupněn v Github repozitáři a je tak dostupný pro další uživatele.

Vzhledem ke stavu tohoto projektu, který je ve fázi prototypu, se jako nejvhodnější postup pro výrobu obalu jeví 3D tisk. Jeho největší výhodou je relativně rychlá a levná produkce plastových dílů v kusové výrobě. Pomocí 3D tisku jsme totiž schopni vyrobit v řádu hodin několik verzí modelu, a tak postupně na reálných výrobcích testovat správnost rozměrů, proveditelnost montáže nebo uživatelskou přívětivost. Navíc na rozdíl od metody například vstřikovacího lisu zde není nutná velká investice do formy, kterou navíc po jejím vyrobení již nelze razantně změnit.

3D tisk sice vyžaduje investici v podobě pořízení samotné tiskárny a tiskového materiálu, ale ve většina škol a zájmových zařízení jsou již tiskárnami vybaveny a dávají je k dispozici svým členům/studentům za dostupnou cenu pronájmu, nebo dokonce zdarma.

Výroba plastových výtisků se navíc stala v České republice oblíbenou volnočasovou aktivitou, a tak není těžké narazit na majitele tiskárny i mezi přáteli a rodinou. Tak tomu bylo i v mém případě a měl jsem tak pro svou práci dostupnou větší paletu různých 3D tiskáren i s odborným poradenstvím pouze za cenu materiálu.

3.1 Metody 3D tisku

Po rychlém vývoji této technologie je v současné době dostupných několik metod výroby plastových modelů. Každá z nich je vhodná pro jiný typ výrobků a nese si jiná pozitiva a negativa.

3.1.1 FUSED DEPOSITION MODELING (FDM)

FDM je v současné době nejrozšířenější metoda aditivní výroby. Princip je založen na postupném tavení tiskového materiálu v podobě tenkého vlákna. K jeho zahřátí probíhá v takzvaném extruderu, tedy tiskové hlavě, která se pohybuje nad povrchem tiskové podložky a postupně nanáší vrstvy materiálu až do dokončení požadovaného tisku.

Dle výrobce tiskárny se liší pohyblivost některých jejích částí. Například modely české společnosti **Prusa Research a.s.** pohybují hlavou po osách X a Z a tiskovou podložkou po ose Y (jak je vidět na 3.3). Jiní výrobci pak mají pro změnu lokace trysky pohyblivou jen hlavu s extruderem[17].

Výhodou této metody je široká paleta materiálů, ze kterých lze výrobky tisknout. Jde například o bioplast PLA, nebo pevnější materiály typu PETG/nylon, nebo dokonce materiály flexibilní, či rozpustné ve vodě, založené na bambusu.

Naopak nevýhodou je delší čas výroby a nutnost využívat v určitých situacích podpůrné konstrukce, které zvyšují cenu výsledného tisku vyšší spotřebou materiálu a uvádí v nutnost provádět opracování modelu po jeho vytištění – je nutno odstranit podpůrné systémy a zahladit stopy po jejich materiálu na výtisku.

Obr. 3.1: FDM Tiskárna Original Prusa MK4

Převzato z [16]

3.1.2 Stereolitografie (SLA)

SLA je nejstarší metodou aditivní výroby. Narozdíl od FDM používá místo strun materiálu tekutou pryskyřici, která reaguje na ultrafialové světlo. Nádoba s tiskovým materiálem je selektivně vystavována tomuto záření a postupně tak vznikají vrstvy ztvrdlého materiálu až po dosažení požadovaného modelu.

Velkou výhodou stereolitografie je vysoká přesnost výtisků a rychlost jejich vytvoření. Nevýhodou je však značně nižší výběr tisknutelných materiálů a technická omezení v typech modelů – některé modely by měly extrémní množství odpadového materiálu, nebo je pomocí SLA nelze tisknout vůbec. Dalším důvodem, proč není tato metoda tak rozšířená, je většinou menší tisková plocha a tím i menší maximální velikost tisknutého modelu. Každý výrobek se navíc musí pro větší pevnost po tisku nechat dále takzvaně „vytvrdit“ pomocí UV světla, což znamená další pořizovací náklady na další zařízení.

Obr. 3.2: FDM Tiskárna Anycubic Photon

Převzato z [18]

3.1.3 Volba metody tisku

Po srovnání všech faktorů a preferencí byla zvolena metoda FDM. Primárními důvody jsou její vyšší dostupnost kvalitnějšího vybavení, lehčí manipulace a také větší znalost této technologie pro mou osobu i odborný dohled.

Tisk byl prováděn na tiskárně Bambu Lab X1-Carbon, která se řadí mezi špičku na aktuálním trhu s tiskárnami. Toho dosahuje například automatickým vyrovnáváním podložky nebo automatickou detekcí chyb, obojí pomocí kamery a modelu

umělé inteligence. Její moderní softwarová výbava také ulehčuje práci s tiskárnou začátečníkům, a to bez ztráty prozatím neporažené rychlosti a kvality tisku. Oproti tiskárnám od Prusa Research jde o uzavřený model s kontrolou teplot a velmi spolehlivým podavačem materiálu, který umožňuje tisknout až 16 barvami/materiály najednou[19].

Obr. 3.3: FDM Tiskárna Bambu Lab X1-Carbon

Převzato z [20]

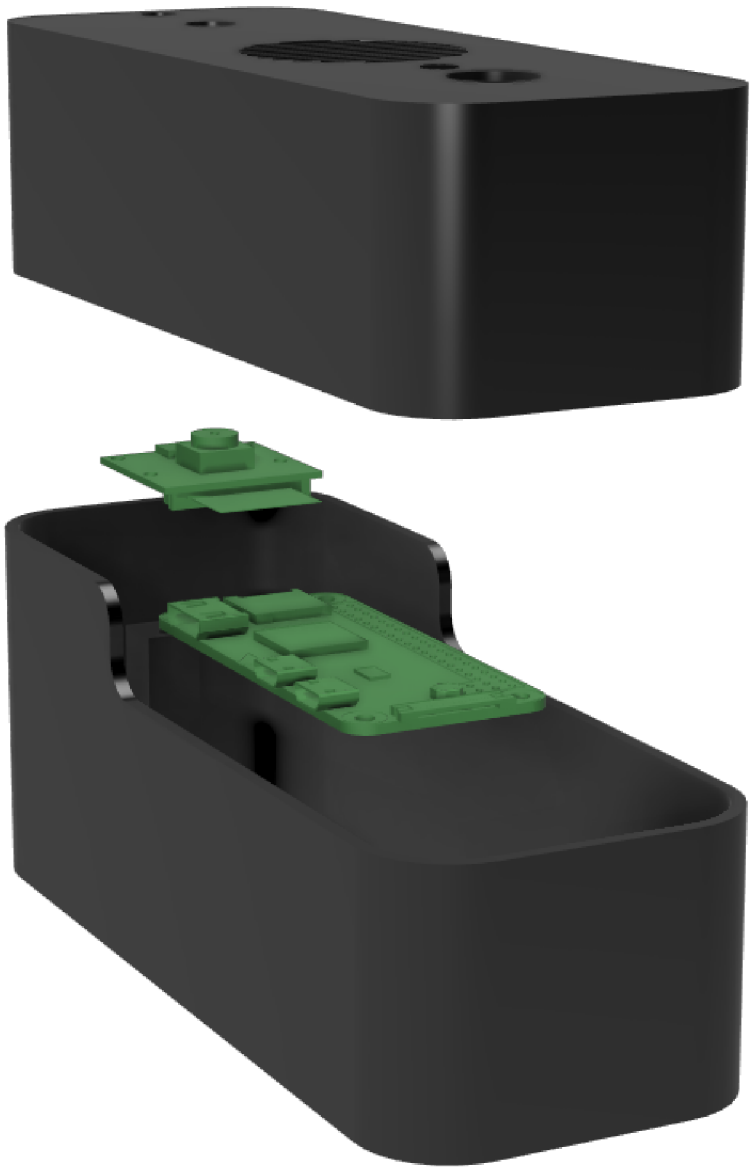
3.1.4 3D model šasi

Samotný 3D model určený pro tisk byl vytvořen v aplikaci Autodesk Fusion360. Software byl zvolen kvůli jednoduchosti a na základě doporučení vlastníka tiskárny.

Jde o design složený ze dvou dílů, které jsou při kompletaci zasunuty do sebe a zafixovány šroubem. Návrh má na horní straně zvonku stříšku, která pomáhá alespoň částečně chránit přes vlivy počasí a zároveň snižuje šance na oslepení kamery přímým slunečním svitem.

Model byl vytvářen maximálně jednoduše, a to nejen kvůli snaze ušetřit čas tisku, ale i kvůli mé zcela začátečnické úrovni vytváření tiskových a 3D modelů.

Obr. 3.4: Model šasi zvonku



Obr. 3.5: Rozdělení modelu šasi

4 Postup instalace a kompilace softwarového řešení

Celý proces nasazení je složen z několika stejně důležitých částí. Po instalaci operačního systému a ovladačů pro přidaný hardware je třeba zkompilovat samotnou aplikaci s knihovnou. Následně nasadíme hlavní python skript a vytvoříme z něj automaticky spouštěnou službu. Na závěr pak pro podporu videa nainstalujeme službu Motion a provedeme její konfiguraci.

4.1 Instalace operačního systému a potřebných ovladačů

Samotný operační systém je na instalaci jednoduchý. Stačí na SD kartu nahrát potřebné soubory a kartu vložit do Raspberry Pi. Vhodným nástrojem pro ulehčení této operace je `Raspberry Pi Imager`, který sám stáhne potřebné soubory, naformátuje SD kartu a provede překopírování souborů.

Po prvním spuštění je třeba nakonfigurovat systém pomocí příkazu sudo raspi-config. Je totiž třeba povolit interface kamery. Ten najdeme v sekci Interface Options (4.1) pod položkou Camera (4.2).

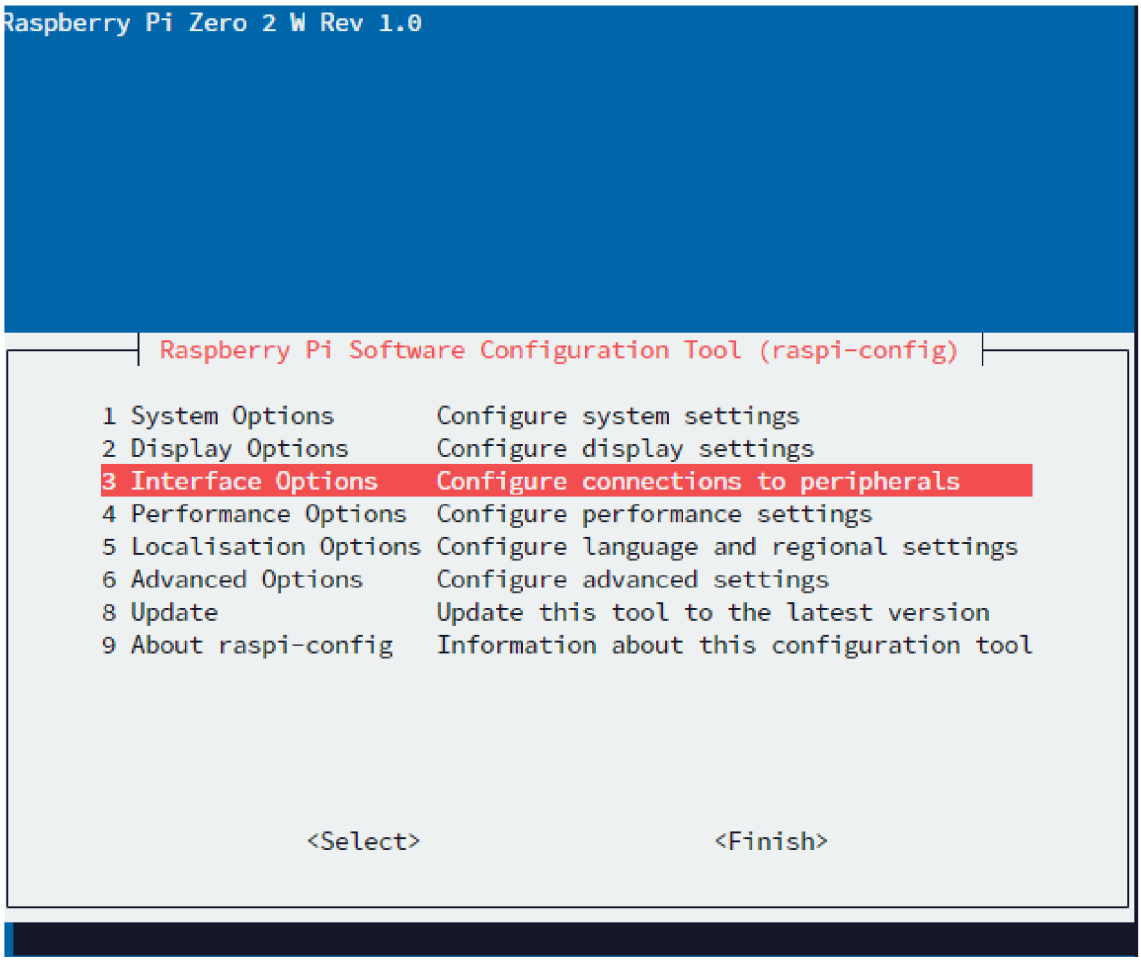
Po úspěšné úvodní konfiguraci pomocí příkazu `sudo raspi-config` je třeba nainstalovat ovladače pro modul ReSpeaker. To provedeme pomocí následujících příkazů.

```
$git clone https://github.com/respeaker/seeed-voicecard.git
$cd seeed-voicecard
$sudo ./install.sh
$sudo reboot now
```

4.2 Konfigurační soubory

Pro zjednodušení nasazení celé technologie jsou vytvořeny konfigurační soubory. Díky umístění v boot sektoru SD karty je možné k nim přistoupit i ze zařízení s OS Windows a konfigurační soubory tak upravit jednoduchým vyjmutím karty.

Nastavení pro software Motion není v této složce k dispozici. Nepočítá se s jeho častou změnou a skript, který by data kopíroval z /boot do /etc/motion by byl dalším slabým článkem narušujícím stabilitu. Motion totiž na rozdíl od našeho PJSIP řešení neumí číst konfiguraci přímo z boot složky.



Obr. 4.1: Uvodní obrazovka Raspi-config

4.2.1 wpa_supplicant.conf

Konfigurační soubor wpa_supplicant.conf je nyní systémem Raspberry Pi OS zpracováván zcela automaticky. Jak název naznačuje, jde o data pro linuxový balíček wpa supplicant, jehož úkolem je automatické připojení zařízení na WiFi síť s možností specifikace přesných parametrů. V našem případě však specifikujeme pouze stát lokace, SSID, tedy název sítě, a k němu příslušné heslo.

```
country=cz
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 scan_ssid=1
 ssid="MojeWIFI"
 psk="zvonek123heslo"
```

```
}
```

Výpis 4.1: wpa_supplicant.conf

4.2.2 doorbell.cfg

```
[pjsua2]
id = sip:123456@sip.odorik.cz
registrarUri = sip:sip.odorik.cz
realm = sip.odorik.cz
scheme = digest
username = 123456
password = Tdd5ss4

[lock]
passcode = 123
targetVoipUri = sip:541146112@sip.odorik.cz
```

Výpis 4.2: doorbell.cfg

Doorbell.cfg je hlavním konfiguračním souborem pro celé fungování zvonku. Sekce označená [pjsua2] specifikuje klíčové prostředí pro knihovnu PJSUA2, ovládající software PJSIP. Jedná se primárně o SIP server a uživatele pod kterým k SIP serveru přistupuje.

Sekce [lock] pak specifikuje heslo, které musí vlastník zadat, aby došlo k otevření, a cílové číslo, na které má RPi volat v případě aktivace hovoru. Veškeré citlivé hodnoty všech konfiguračních souborů jsou pro ukázku v tomto dokumentu změněny na neplatné údaje.

4.2.3 .asoundrc ALSA konfigurace

Pro správnou funkčnost audia na Raspberry Pi Zero 2W je nutné vytvořit konfiguraci správce softwarového mixeru ALSA. Tento soubor se umístí do domovského adresáře uživatele a zajistí, že je systém schopen správně pracovat s mikrofonem a reproduktorem ve stejný čas. Konfigurace na jiných modelech nebo revizích hardwaru či softwaru se bohužel může lišit.

```
pcm.!default {
  type asym
  playback.pcm {
```

```
        type plug
        slave.pcm "hw:0,0"
    }
    capture.pcm {
        type plug
        slave.pcm "hw:0,0"
    }
}

ctl.!default {
    type hw
    card 0
}
```

Výpis 4.3: Konfigurační soubor .asoundrc

4.3 Kompilace PJSIP s knihovnou PJSUA2

Pro použití PJSIPu je nutné nejdříve provést kompilaci programu s knihovnou PJSUA a potřebnými kodeky.

Prvním krokem kompilace je aktualizace repozitářů a stažení všech potřebných balíčků, které kompilace a všechny naše aplikace pro svůj běh vyžadují. V případě Raspberry Pi OS je použit správce balíčků apt.

```
$sudo apt-get update

$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 swig python3-dev -y


$sudo apt-get install libasound2-dev libssl-dev
libv4l-dev libsdl2-dev libsdl2-gfx-dev libsdl2-image-dev
libsdl2-mixer-dev libsdl2-net-dev libsdl2-ttf-dev
libx264-dev libavformat-dev libavcodec-dev
libavdevice-dev libavfilter-dev libavresample-dev
libavutil-dev libavcodec-extra libopus-dev
```

```
libopencore-amrwb-dev libopencore-amrnb-dev
libvo-amrwbenc-dev -y
```

Následuje stažení knihovny z GitHub repozitáře a rozbalení archivu.

```
$wget https://github.com/pjsip/pjproject/archive/2.10.tar.gz
$tar xfv 2.10.tar.gz
```

Ve vzniklé složce otevřeme podsložku `pjlib/include/pj` a vytvoříme soubor `config_site.h` pomocí softwaru `nano`.

```
$cd <slozka_z_archivu>/pjlib/include/pj
$nano config_site.h
```

Poté jej naplníme obsahem:

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

Výpis 4.4: config_site.h

V kořenovém adresáři knihovny vytvoříme soubor `user.mak` a pomocí `nano` vepíšeme následující data. Tím řekneme kompilátoru, aby knihovnu kompiloval pro naše zařízení a generaci procesoru.

```
export CFLAGS += -fPIC -mcpu=cortex-a53 -mfpu=neon-fp-armv8
-mfloat-abi=hard -mlittle-endian -munaligned-access -ffast-math

export LDFLAGS +=
```

Výpis 4.5: user.mak

Zůstaneme v kořenovém adresáři a spustíme přichystaný konfigurační skript:

```
$./configure --enable-shared --disable-libwebrtc
```

Kvůli malé velikosti RAM na RPi Zero 2W zvýšíme SWAP na 1024 MB, tedy využijeme větší část SD karty jako značně pomalejší operační paměť a umožníme běh kompilace. Po změně swapu je doporučován reboot.

```
$sudo dphys-swapfile swapoff
$sudo nano /etc/dphys-swapfile

#v souboru přepíšeme CONF_SWAPSIZE=100 na hodnotu CONF_SWAPSIZE=1024.

$sudo dphys-swapfile setup
$sudo dphys-swapfile swapon

#nyní je doporučován restart RPi
$sudo reboot
```

Nyní provedeme kompilaci, ještě bez python wrapperu.

```
$make dep
$make
$sudo make install

#pokud musíme kompilaci opakovat, je nutné před
#jejím spuštěním vymazat předchystaná data
#pomocí příkazů:

$make clean
$sudo make clean
$sudo make distclean
```

V adresáři `pjsip-apps/src/swig` upravíme soubor `Makefile` tak, aby se při kompilaci nezpracovával wrapper pro jazyky Java a C#.

```
$cd pjsip-apps/src/swig
$nano makefile
```

Následně řádek číslo 10 upravíme do stavu:

```
            LANG = python
```

Zůstaneme v adresáři `pjsip-apps/src/swig` a provedeme finální kompilaci knihovny pro Python 3. Následně dokončíme její instalaci, aby bylo možné ji v rámci kódu importovat jako `pjsua2`.

```
$make
$make install

#pokud musíme kompilaci opakovat,i zde je nutné před
#jejím spuštěním vymazat předchystaná data
#pomocí příkazů:

$make clean
$sudo make clean
$sudo make distclean
```

4.4 Vytvoření služby pro automatické spuštění

Služby systemd v rámci linuxu jsou podobné službám v rámci systému Windows. Lze nastavit jejich čas spuštění, uživatele a podmínky. Zde je vytvořena služba voip.service, která se aktivuje při startu systému po připojení k internetu. Služba vykoná aktivaci výše popsaného Python skriptu doorbell.py a specifikuje umístění konfiguračního souboru, v tomto případě ve složce boot.

```
$cd /etc/systemd/system
```

Vytvoření služby začneme vstupem do složky se všemi soubory služeb v systému. Použijeme příkaz `cd`.

```
$sudo nano voip.service
```

Následně pomocí `nano` vložíme níže uvedený obsah samotné služby. Soubor ponese název `voip.service`.

```
[Unit]
Description=SIP zvonek deamon
After=network-online.target
[Service]
Type=idle
ExecStart=/usr/bin/python3 /home/pi/doorbell.py "/boot/doorbell.cfg"
WorkingDirectory=/home/pi/
User=pi
[Install]
```

```
WantedBy=multi-user.target
```

Výpis 4.6: voip.service

Tímto jsme vytvořili konfiguraci služby. Nyní ji musíme vyzkoušet a aktivovat. Je nutné, aby všechny následující příkazy byly prováděny s povýšením práv, tedy s příkazem `sudo`.

```
$sudo systemctl daemon-reload
```

Příkaz `deamon-reload` provede znovunačtení seznamu všech služeb a zaznamená tak existenci naší nové služby pro příkaz `systemctl`.

```
$sudo systemctl start voip.service
$sudo systemctl enable voip.service
```

Po vytvoření služby a znovunačtení deamona systemctl můžeme příkazy `start` a `enable` naši novou službu nejdříve spustit a následně zaregistrovat do automatického startu při startu systému (po splnění podmínek určených v souboru služby).

V případě potřeby deaktivace automatického spouštění by se použil příkaz `systemctl` s parametrem `disable`. Při každé změně ve `voip.service` souboru je nutné provést i opětovné znovunačtení deamona příkazem `sudo systemctl daemon-reload`.

4.5 Konfigurace video funkcionalit

Po nastavení systému pro hovory je třeba provést konfiguraci videostreamu. Ten běží nezávisle na procesech pro audio. Výhodou tohoto řešení je, že obraz je dostupný i bez aktivního hovoru a naše zařízení tak lze použít jako bezpečnostní kameru nebo po napojení do chytré domácnosti (například `home-assistant.io`) lze kameru využívat pro detekci obličejů nebo SPZ. Jako HTTP stream jde navíc kamera přidat i do některých NVR a je ji tak možné nahrávat na externí úložiště pro případ, že by byl záznam potřeba.

4.5.1 Instalace a konfigurace aplikace Motion

Jako první provedeme instalaci aplikace Motion a nástrojů pro enkódování videa a práci s kamerou - ffmepg a video4linux:

```
$sudo apt-get update

$sudo apt-get install ffmpeg v4l-utils motion -y
```

Po nainstalování je třeba provést konfiguraci. Tu nalezneme v souboru `motion.conf` ve složce `/etc/motion`. Půjde primárně o změny rozlišení, framerate a nastavení samotného přenosu a encoderu. Případně zde můžeme provést i otočení kamery nebo upravit barvy či jas obrazu. Upravíme tedy následující řádky na stav:

```
#řádek 11
daemon on

#řádek 79
width 320

#řádek 82
height 240

#řádek 86
framerate 1

#řádek 461
stream_localhost off
```

Celý konfigurační soubor je uložen v repozitáři. Místo manuálních úprav je tak možné jej stáhnout a nahradit.

V případě zájmu je možné v závislosti na výkonnosti zařízení, kvalitě připojení a kvalitě kamery zvolit i jiné hodnoty rozlišení a počtu snímků. V ukázaném řešení jsem volil nejstabilnější cestu i pro slabší konektivity, protože zvonek bude pravděpodobně umístěn venku, daleko od nejbližšího Wi-Fi přípojného místa.

Po konfiguraci aplikace Motion je třeba nakonfigurovat i jejího deamona, neboli službu běžící na pozadí. Tuto část softwaru povolíme (tedy umožníme její automatický start se systémem) a následně manuálně spustíme.

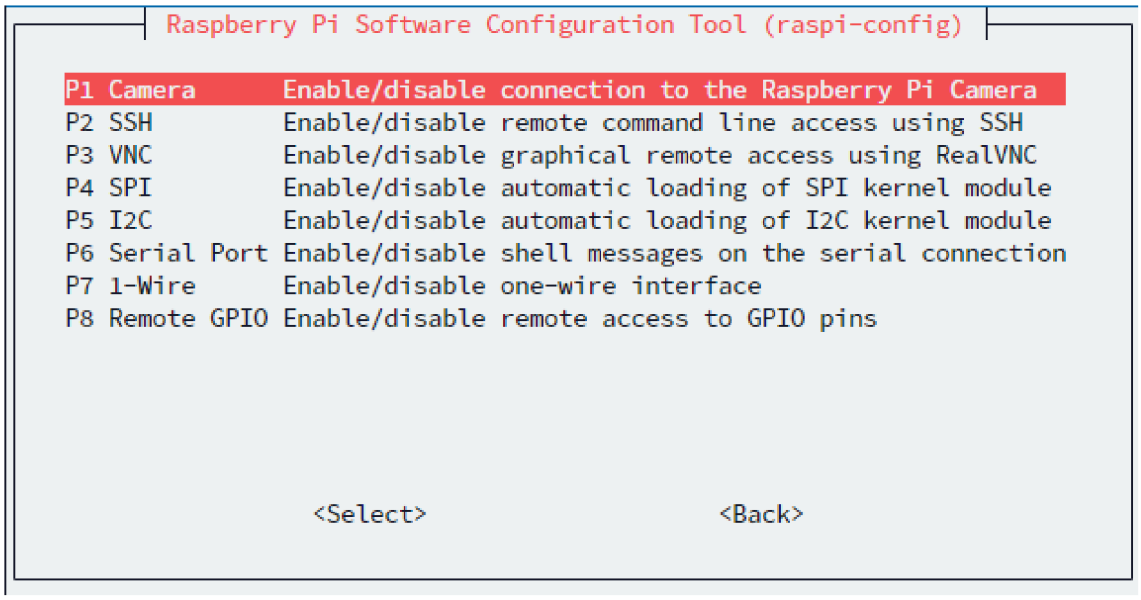
Služba je již předchystaná od vývojáře, takže ji nemusíme ručně vytvářet, jako u našeho PJSIP systému. Stačí tak pouze provést povolení příkazem:

```
sudo systemctl enable motion
```

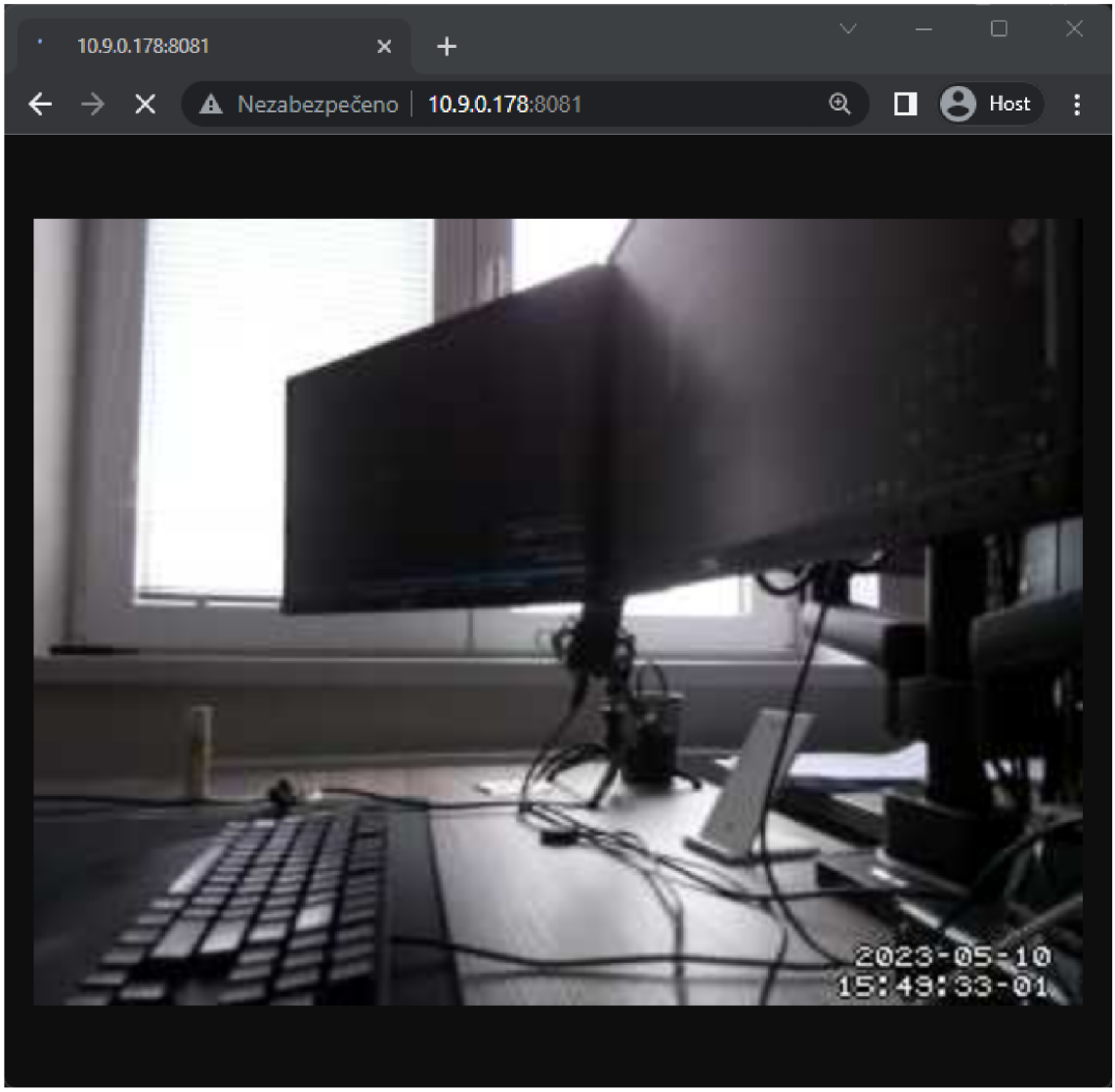
a následně službu manuálně spustit:

```
sudo systemctl start motion
```

Nyní by měl být na portu `8081` dostupný videopřenos z naší kamery. Pro jeho zobrazení stačí do webového prohlížeče zadat adresu `http://[RPI-IP]:8081` (4.3).



Obr. 4.2: Sekce interface Options v Raspi-config



Obr. 4.3: Živý stream z kamery pomocí Motion

5 Montáž komponent do šasi

Kompletace zvonku je proveditelná s běžně dostupnými materiály a vybavením. Každý z dílů šasi fixuje část komponent pomocí přichystaných slotů na šrouby. Při utahování šroubů je vždy nutné dbát zvýšené opatrnosti, aby nedošlo k utržení plastových úchytů přílišným utažením.

Vhodným postupem je alespoň zběžně otestovat softwarové řešení a předejít tak nutnosti demontáže z šasi v případě chyby v konfiguraci, která by zabránila přístup do zařízení skrze SSH, nebo nevyřešitelně omezila fungování zvonku.

5.1 Přední část šasi

Prvním krokem je montáž kamery do přední sekce modelu (5.1). Připravena jsou 4 místa, ale kamera bude díky své nízké váze stabilně ukotvena i dvěma krátkými šrouby o velikosti M2. Kameru doporučuji ještě před instalací opatřit datovým kabelem, jeho instalace po montáži kamery je totiž značně náročnější.

Pro instalaci reproduktoru jsem využil pevné kobercové pásky a podložil membránu mikrofonní pěnou. Důvodem pro toto rozhodnutí byly vibrace plastové mřížky modelu, které značně zhoršovaly kvalitu zvuku a znemožňovaly komunikaci. V úvahu přichází i využití oboustranné pěnové lepící pásky nebo lepidla. Mé řešení se však ukázalo jako dostatečně pevné i při méně šetrném zacházení a na rozdíl od jiných variant umožňuje lehčí demontáž reproduktoru nebo pěny při jejím znečištění nebo degradaci.

Jako poslední do této sekce nainstalujeme tlačítko pro spuštění hovoru. Jeho instalace je jednoduchá a provede se provlečením tlačítka skrze otvor v modelu a následným našroubováním matky dodané s tlačítkem. Tento spoj je nutno pevně utáhnout.

5.2 Zadní část šasi

Následně je třeba zapojit všechny potřebné konektory do desek RPi a ResSpeaker (5.2). Po vložení do boxu to totiž bude velmi obtížné. Reproduktor má svůj označený výstup, pro upravený konektor Grove využijeme port na ReSpeaker HAT s popiskem **GP12**. Je nutné připojit i kameru do slotu přímo na desce Raspbery Pi. Před umístěním obou PCB do šasi doporučuji i provléct a připojit Micro USB konektor.

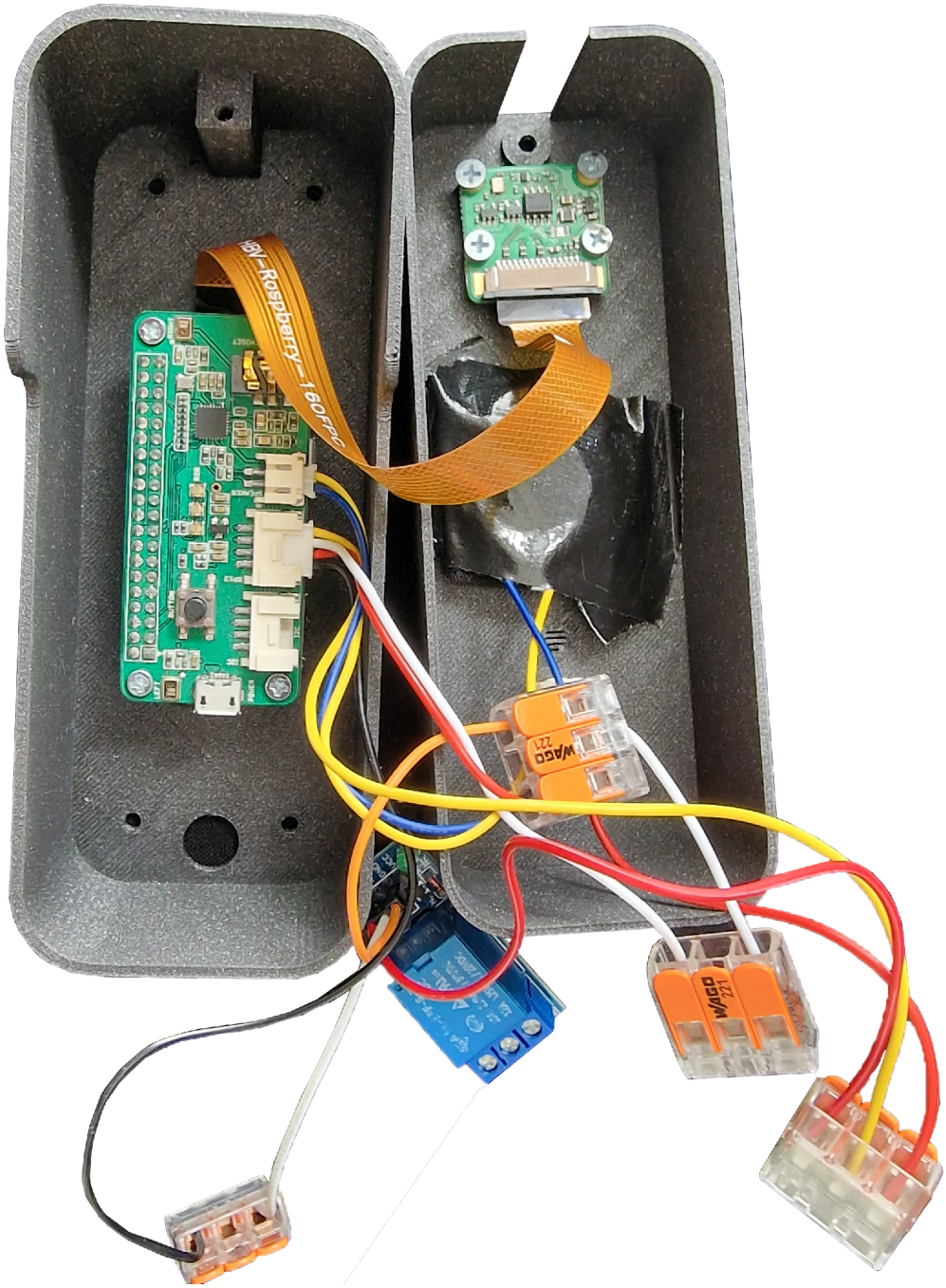
Po upevnění všech zásadních komponent je nutné umístit případné spoje nebo ostatní prvky do boxu tak, aby při spojení nedošlo k jejich poškození. V této práci je zvonek ve fázi prototypu, takže obsahuje velké množství svorek a nadbytečně

velkých součástek, jako je Micro USB konektor nebo relé. Z tohoto důvodu je i celý model větší, než by bylo v jakékoliv reálné produkci nutné.

5.3 Finální montáž

Po umístění všech součástek a všech případných spojů je možné spojit oba díly šasi dohromady. To je provedeno zasunutím přední části do části zadní. K finálnímu ukotvení pak slouží jeden, zepředu umístěný šroub v horní části zvonku. **Během spojování je třeba dbát zvýšené opatrnosti, aby nedošlo ke skřípnutí propojovacích vodičů pod nebo mezi částmi boxu.** Po dokončení fyzické montáže (5.3) je zvonek připraven k práci.

Obr. 5.1: Instalace komponent do přední strany šasi



Obr. 5.2: Rozložené šasi s komponenty

Obr. 5.3: Zkompletované fyzické řešení zvonku

6 Ověření funkčnosti

Ověření funkčnosti mého řešení si dovolím demonstrovat za pomoci odchycení paketů pomocí softwaru `tcpdump` a screenshotu z běhu programu, ukazujícího přijetí kódu a spuštění předdefinované akce pro odemčení. Pro ověření videa jsem již výše přiložil snímek obrazovky funkčního přenosu do okna prohlížeče, ale pro úplnost zde doložím záznam ze softwaru Wireshark a jeho příjem HTTP paketů.

Protocol	Length	Info
SIP/SDP	1206	Request: INVITE sip: [redacted]@sip.odorik.cz |
SIP	534	Status: 407 Proxy Authentication Required |
SIP	425	Request: ACK sip: [redacted]@sip.odorik.cz |
SIP/SDP	1395	Request: INVITE sip: [redacted]@sip.odorik.cz |
SIP	377	Status: 100 Trying |
SIP	650	Status: 180 Ringing |
SIP/SDP	907	Status: 200 OK (INVITE) |
SIP	495	Request: ACK sip:* [redacted]@ [redacted]:5060 |
SIP	460	Request: BYE sip: [redacted]@ [redacted]58444;ob |
SIP	460	Request: BYE sip: [redacted]@ [redacted]58444;ob |
SIP	460	Request: BYE sip: [redacted]@ [redacted]58444;ob |
SIP	438	Status: 200 OK (BYE) |
SIP	438	Status: 200 OK (BYE) |
SIP	438	Status: 200 OK (BYE) |

Obr. 6.1: Záznam komunikace SIP protokolu, zachycený pomocí tcpdump

Na obrázku 6.1 vidíme veškerou komunikaci SIP protokolu potřebnou pro jeden hovor. Jde o navázání spojení, přihlášení se k SIP serveru a zahájení hovoru.

Na následujícím screenshotu 6.2 je zachycen datový tok audia protokolem RTP. Zároveň si můžeme všimnout i přenosu DTMF kódů v rámci RTP EVENT. Vidíme, že datový tok audia probíhá celou dobu a prolíná se s přenosem DTMF. Audio nesmí být kódy přerušeno, takže jeho stream probíhá neustále.

Ve výpisu z konzole na obrázku 6.3 je právě dokončená registrace k SIP serveru. Vidíme, že i registrační komunikace probíhá přes UDP. Registrace končí statusem 200, tedy OK a je platná na 300 vteřin. Na posledním řádku vidíme průběh KeepAlive paketů, které se posílají jednou za 15 vteřin ze strany PJSIPu na servery Odoriku. V administraci na odorik.cz je možné kontrolovat klienty připojené k účtu SIP, jak lze vidět na obrázku 6.4.

Po spojení je možné navazovat hovory, přijímat DTMF kódy a případně spouštět kód po zadání kódu správného. Níže přikládám více obrázků, které funkčnost mého řešení dokazují.

```
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9254, Time=41760, Mark
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x2A1765CB, Seq=14026, Time=19886080
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9255, Time=41920
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9256, Time=42080
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9257, Time=42240
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9258, Time=42400
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x29AFA021, Seq=9259, Time=42560
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2 (end)
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2 (end)
RTP EVENT            58 Payload type=RTP Event, DTMF Two 2 (end)
RTP                 214 PT=ITU-T G.711 PCMA, SSRC=0x2A1765CB, Seq=14041, Time=19886880
```

Obr. 6.2: Záznam RTP toku dat spolu s DTMF kódy

```
--end msg--
12:06:20.850   pjsua_core.c  .RX 483 bytes Response msg 200/REGISTER/cseq=46741 (rdata0x23f0ad4) from UDP 81.31.45.51:5060:
SIP/2.0 200 OK
Via: SIP/2.0/UDP             :54575;rport=54575;branch=z9hG4bKPje3663db6-8693-4e47-b7dd-e5e624e377de;received=
From: <sip:      @sip.odorik.cz>;tag=8d0c8e86-2ab5-4fec-8739-4b093543b5e1
To: <sip:      @sip.odorik.cz>;tag=d9f55a8611ba6e9295acea97d3a19a47.5b28
Call-ID: 014f4d1e-3c40-4024-a6ee-f7cc41b59ef7
CSeq: 46741 REGISTER
Contact: <sip:      @            :58444;ob>;expires=89, <sip:                  :54575;ob>;expires=300
Content-Length: 0


--end msg--
12:06:20.854    pjsua_acc.c  ....SIP outbound status for acc 0 is not active
12:06:20.856    pjsua_acc.c  ....sip:      @sip.odorik.cz: registration success, status=200 (OK), will re-register in 300 seconds
12:06:20.857    pjsua_acc.c  ....Keep-alive timer started for acc 0, destination:81.31.45.51:5060, interval:15s
```

Obr. 6.3: Konzole RPi: Registrace SIP

Na screenshotu 6.7 je zobrazeno chování programu při přijetí DTMF kódu od vlastníka. V první variantě dojde k resetu, v druhé je kód potvrzen a akce spuštěna.

Na Wireshark záznamu (6.8) jsou vidět TCP pakety odeslané protokolem HTTP. V pravém dolním rohu pak lze vidět, že obsahem je JPEG obrázek o rozlišení 320x240 bodů.

Registrovaná SIP zařízení

stav	jméno	poslední registrace	registrace vyprší	veřejná IP	lokální IP	server port	protokol
	n/a	2021-12-08 16:51:37	2021-12-08 16:56:37	:54575 (info o IP)	(web zařízení)	5060	UDP

Detailní výpis posledního pokusu o registraci zde.

Obr. 6.4: Administrace Odorik.cz - seznam přihlášených SIP klientů

```
--end msg--
12:07:06.224   pjsua_core.c  .RX 335 bytes Response msg 100/INVITE/cseq=23153 (rdata0x23f0ad4) from UDP 81.31.45.51:5060:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP             :54575;rport=54575;branch=z9hG4bKPjc8a36beb-4785-456d-9c2a-10478c056d33;received=
From: sip:      @sip.odorik.cz;tag=98d50ba1-4239-4a19-8cfc-412dd97045ed
To: sip:      @sip.odorik.cz
Call-ID: b5f2b00a-1251-4939-8670-de78bfc65294
CSeq: 23153 INVITE
Content-Length: 0


--end msg--
12:07:07.041   pjsua_core.c  .RX 601 bytes Response msg 180/INVITE/cseq=23153 (rdata0x23f0ad4) from UDP 81.31.45.51:5060:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP             :54575;received=            ;rport=54575;branch=z9hG4bKPjc8a36beb-4785-456d-9c2a-10478c056d33
Record-Route: <sip:      @81.31.45.51;lr=on;ftag=98d50ba1-4239-4a19-8cfc-412dd97045ed;did=f446.909c1>
From: sip:      @sip.odorik.cz;tag=98d50ba1-4239-4a19-8cfc-412dd97045ed
To: sip:      @sip.odorik.cz;tag=as0b58786c
Call-ID: b5f2b00a-1251-4939-8670-de78bfc65294
CSeq: 23153 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH
Supported: replaces
Contact: <sip:*754860@89.185.255.38:5060>
Content-Length: 0
```

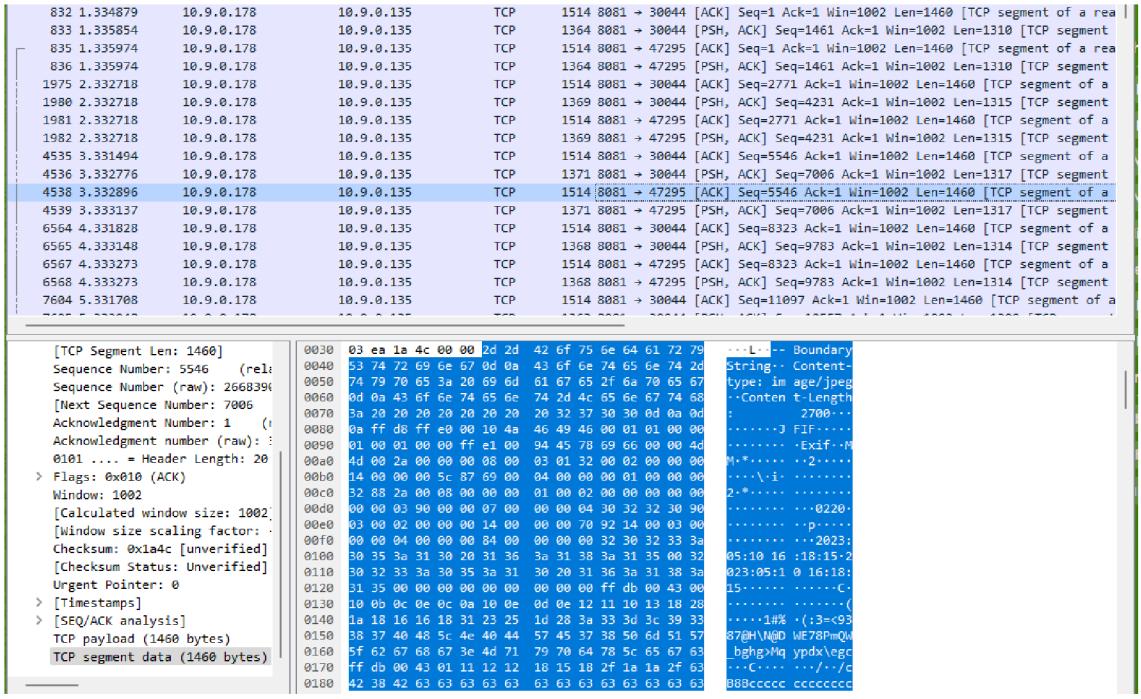
Obr. 6.5: Konzole RPi: Zahájení hovoru

```
12:07:36.990   inv0x24297cc  ....SDP negotiation done: Success
12:07:36.992  pjsua_media.c  .....Call 0: updating media..
12:07:36.995  pjsua_media.c  .......Media stream call00:0 is destroyed
12:07:37.000    pjsua_aud.c  ......Audio channel update..
12:07:37.002  strm0x2438724  .......VAD temporarily disabled
12:07:37.005  strm0x2438724  .......Encoder stream started
12:07:37.011  strm0x2438724  .......Decoder stream started
12:07:37.013  pjsua_media.c  ......Audio updated, stream #0: PCMA (sendrecv)
12:07:37.025    pjsua_aud.c !.....Conf connect: 0 --> 1
12:07:37.029   conference.c  ......Port 0 (default) transmitting to port 1 (sip:      @sip.odorik.cz)
12:07:37.032    pjsua_aud.c  .....Conf connect: 1 --> 0
12:07:37.035   conference.c  ......Port 1 (sip:754860@sip.odorik.cz) transmitting to port 0 (default)
12:07:37.053   pjsua_core.c  .....TX 455 bytes Request msg ACK/cseq=23153 (tdta0x243db44) to UDP 81.31.45.51:5060:
ACK sip:*      @89.185.255.38:5060 SIP/2.0
Via: SIP/2.0/UDP             :54575;rport;branch=z9hG4bKPj275483bd-53d9-4f1a-a8f8-ef0a56fbe623
Max-Forwards: 70
From: sip:     6@sip.odorik.cz;tag=98d50ba1-4239-4a19-8cfc-412dd97045ed
To: sip:      @sip.odorik.cz;tag=as0b58786c
Call-ID: b5f2b00a-1251-4939-8670-de78bfc65294
CSeq: 23153 ACK
Route: <sip:      @81.31.45.51;lr;ftag=98d50ba1-4239-4a19-8cfc-412dd97045ed;did=f446.909c1>
Content-Length:  0


--end msg--
12:07:37.243   sound_port.c  EC activated
12:07:37.647  strm0x2438724  VAD re-enabled

12:07:40.270    ec0x23f9e30  Buffer size adjusted from 1280 to 1112 (eff_cnt=926)
12:07:46.244 echo_suppress. !Echo suppressor updated at t=013.830s, echo tail=190 msec, factor min/avg=0.204/0.009
12:07:46.345    ec0x23f9e30  Buffer size adjusted from 1112 to 654 (eff_cnt=760)
```

Obr. 6.6: Konzole RPi: Dokončení spojení hovoru

```
--end msg--
12:07:37.243   sound_port.c  EC activated
12:07:37.647  strm0x2438724  VAD re-enabled

12:07:40.270    ec0x23f9e30  Buffer size adjusted from 1280 to 1112 (eff_cnt=926)
12:07:46.244 echo_suppress. !Echo suppressor updated at t=013.830s, echo tail=190
12:07:46.345    ec0x23f9e30  Buffer size adjusted from 1112 to 654 (eff_cnt=760)
Received DTMF digit: 1
Received DTMF digit: 2
Received DTMF digit: 3
Received DTMF digit: *
Sending reset
Resetting
Received DTMF digit: 1
Received DTMF digit: 2
Received DTMF digit: 3
Received DTMF digit: #
Sending OK
Passcode is 123
==================== unlocked ====================
==================== locked ====================
```

Obr. 6.7: Konzole RPi: Přenos DTMF kódů a spuštění kódu po ověření



Obr. 6.8: Wireshark záznam - příjem JPEG obrázků skrze HTTP

7 Analýza z pohledu informační bezpečnosti

Chytrý zvonek, jakožto zařízení připojené do sítě s možností komunikovat, umístěné na vnější straně budovy může představovat bezpečnostní riziko a možný vstupní bod pro případného útočníka. V následujících bodech analyzuji možné problémy spojené s instalací zařízení tohoto typu do míst v dosahu neznámých osob.

7.1 Fyzické napadení přístroje

7.1.1 Odcizení zařízení

V případě, že dojde k fyzickému vniknutí do zařízení zvonku postaveného na v této práci popisovaném zařízení, může dojít k odcizení celého mikropočítače, nebo k byť i dočasnému vyjmutí SD karty a přečtení jejího obsahu.

Kromě ztráty finanční v případě odcizení zařízení tak může dojít i k úniku interních informací, které je možné zneužít.

7.1.2 Únik konfiguračních souborů

Z důvodu jednoduchého nastavení jsou v boot sektoru úložiště uloženy běžně čitelné konfigurační soubory samotného VOIP spojení, a před první startem i nešifrované přístupy k síti Wi-Fi. Při odcizení těchto dat může útočník získat přístup k vašemu účtu u VOIP operátora nebo údaje od linky zneužít k hovorům na účet vlastníka zařízení a tím mu způsobit nemalou finanční ztrátu.

V případě přečtení souboru wpa_supplicant.conf získá útočník přístup k Wi-Fi síti, kterou zařízení používá pro přístup k internetovému poskytovateli služeb. Tento soubor se už sice v novějších verzích operačního systému po importu maže z boot sektoru, stále je však dostupný přímo v konfiguračních složkách systému. Tato importovaná verze sice uchovává heslo v zašifrované podobě, ale i tak může vydat podstatné informace, jako je SSID sítě nebo certifikát v případě firemních/školních sítí.

Navíc po přidaní video přenosu může útočník po infiltraci do sítě užít kamery k získávání dat o chování obyvatel a vytvoření ucelenějšího profilu a časů, kdy je budova bez přítomnosti osob.

7.1.3 Možnosti obrany

Kromě použití dostatečně připevněného zařízení v dostatečně pevném obalu je na místě se chránit i vůči výše uvedeným útokům po vniknutí do zvonku.

Je vhodné jako VOIP poskytovatele použít službu oddělující účty SIP klientů od účtu s přístupem do administrace poskytovatele. Stejně tak je přínosné využít služeb operátora s možností omezení maximální částky fakturace nebo operátora využívajícího kreditový systém. Tím je možné výrazně omezit finanční ztrátu v případě zneužití linky.

V rámci ochrany dat při úniku souboru wpa_supplicant.conf doporučuji využít vytvoření vlastní, vyhrazené Wi-Fi (a VLAN) sítě pro zařízení chytré domácnosti. V této síti je pak možné omezit komunikaci v rámci lokálního prostředí jen na nutná zařízení (centrály domácí automatizace a podobně) a zároveň limitovat možnost připojení se dle whitelistu MAC adres (s nahlášením incidentu pokusu o připojení neschválené adresy na e-mail nebo pomocí notifikace). Tím dojde k zamezení vniku do domácí vnitřní sítě a whitelist MAC adres výrazně ztíží připojení cizího zařízení i se znalostí hesla.

Dalším krokem nastavitelným u většiny routerů je možnost skrytí SSID. Tento postup sice ztíží nastavení některých prvků sítě, ale zároveň výrazně omezí falešné pokusy o přihlášení způsobené omylem (a tím omezí počet falešných notifikací o pokusu nepovolených MAC) a zároveň ztíží nalezení sítě bez získání konfiguračního souboru.

Vůči zneužívání internetové přípojky je pak možno se bránit omezením rychlosti připojení k internetu. Většina IoT zařízení má naprosto minimální datové přenosy a vystačí si s konektivitou, která je pro běžného uživatele zcela nedostatečná a demotivující.

Ne na všech domácích routerech a síťových zařízeních je možné nastavit všechna výše uvedená doporučení. Každá část zvlášť je však určitým zvýšením bezpečnosti, a tak doporučuji se vždy snažit co nejvíce přiblížit uváděným návrhům.

7.2 Útok na softwarové řešení

7.2.1 Sledování síťového provozu

Další variantou útoku je možnost cíleně útočit na zařízení a na něm použité síťové protokoly a aplikace. Útočník například může vyčíst síťovou konfiguraci (jak je již popsáno výše), následně kartu vrátit zpět a získané údaje použít pro sledování a odchyt datových přenosů zvonku.

SIP a jeho přidružené protokoly hovoru jsou v základním stavu, stejně jako protokol HTTP, nešifrované. V případě získání přístupu do sítě je tak možné útokem Man-In-The-Middle odchytávat pakety hovorů a ty tak odposlouchávat.

7.2.2 Cílený útok na použité protokoly

Stejně tak se může útočník snažit zaútočit na zranitelnosti objevené v použitých protokolech a aplikacích a získat tak vzdálený přístup k zařízení. Ten pak může použít k vytvoření opakované možnosti otevření objektu nebo vyřazení zvonku z provozu. V případě kombinace se špatně nastavenou sítí (jak je popsáno v bodě 7.1.2) může útočník získat přístup do lokální sítě skrze napadený zvonek nebo použít získanou konektivitu k útokům na vnější cíle zapojením Raspberry Pi do svého botnetu.

7.2.3 Možnosti obrany

Obrana je veskrze stejná, jako u fyzického odcizení, a to primárně ve snaze omezit Wi-Fi síť na síť jen pro IoT zařízení a v případě dostatečně výkonného hardwaru a schopného softwaru nasadit šifrovanou verzi protokolu SIP.

Pro obranu vůči zneužití videopřenosů je možné zvážit nasazení autentifikace (např. skrze lokální reverzní proxy na RPi), nebo omezení přístupu k web serveru jen na určité adresy ve vnitřní síti. I tyto limitace se však dají překonat.

Poslední navrhovaná úprava – šifrování protokolu SIP pomocí certifikátů nebyla v rámci práce nasazena z důvodu výkonnostních a softwarových limitů řešení. Nutno však podotknout, že zvolený VoIP operátor tuto technologii podporuje.

Závěr

Výsledné řešení zvládá plnit základní cíl. Zařízení je schopno po stisku tlačítka uskutečnit hovor s oboustrannou komunikací a po autorizaci volaného uživatele i spustit předem definovaný kód.

Hovory je možné vyvolat i opakovaně bez nutnosti zásahu do běhu zvonku samotného. Konfigurace pro hovory i připojení k síti je načítaná ze souborů uložených v prostoru SD karty, čitelných i ze systému Microsoft Windows (díky formátování FAT32) a je tak umožněna vcelku pohodlná změna konfigurace celého řešení. Raspeberry Pi má nakonfigurovanou možnost plně automatické obnovy provozu po výpadku napájení nebo restartu systému zcela bez zásahu uživatele. Tlačítko má navíc ošetřené i opakované stisky nebo špatné zacházení ze strany uživatele.

Při hovoru lze odemknout dveře (sepnout relé) pomocí zadání správného číselného kódu, který se odešle pomocí DTMF kódů.

Videopřenos je proveden mimo VoIP hovory pomocí HTTP JPEG streamu. Jeho zobrazení tak není závislé na samotném hovoru a konfigurace obrazu je značně flexibilnější s umožněním změn rozlišení nebo rotace obrazu na základě konektivity či výkonu užitého zařízení. Díky použití aplikace Motion je pro zkušené uživatele ponechán prostor si nastavit pokročilé možnosti a funkce a tím značně rozšířit funkcionalitu mého řešení.

Celý zvonek je uložen do šasi, vyrobeného pomocí technologie 3D tisku. Model je navržen tak, aby co nejvíce ochránil zařízení před počasím a snaží se také chránit obraz kamery vůči slunečnímu svitu. Tvarem navíc maximálně napodobuje běžné video zvonky.

Literatura

[1] Buy a Raspberry Pi 4 Model B — Raspberry Pi. *Raspberry Pi* [online]. Cambridge: Raspberry Pi (Trading) Limited., 2020 [cit. 2021-12-03]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[2] Raspberry Pi 4 Tech Specs. *Raspberry Pi* [online]. Cambridge: Raspberry Pi (Trading) Limited., 2020 [cit. 2021-12-03]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[3] PRENNER, Michal. Raspberry Pi Zero WH. *RPIshop.cz: Váš dodavatel Raspberry Pi* [online]. Roudné: RPishop.cz, 2017 [cit. 2021-12-03]. Dostupné z: https://rpishop.cz/zero/685-raspberry-pi-zero-wh-4250236816296.html

[4] PRENNER, Michal. Raspberry Pi Zero 2 W. *RPIshop.cz: Váš dodavatel Raspberry Pi* [online]. Roudné: RPishop.cz, 2017 [cit. 2023-10-05]. Dostupné z: https://rpishop.cz/raspberry-pi-zero/4311-raspberry-pi-zero-2-w-5056561800004.html

[5] ReSpeaker 2-Mics Pi HAT - Seeed Studio. *Seeed Studio* [online]. Shenzhen: Seeed Studio [cit. 2021-12-03]. Dostupné z: https://www.seeedstudio.com/ReSpeaker-2-Mics-Pi-HAT.html

[6] PRENNER, Michal. Raspberry Pi kamera V2. *RPIshop.cz: Váš dodavatel Raspberry Pi* [online]. Roudné: RPishop.cz, 2017 [cit. 2023-10-05]. Dostupné z: https://rpishop.cz/mipi-kamerove-moduly/329-raspberry-pi-kamera-modul-v2.html

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

[8] PILGRIM, Mark. *Ponořme se do Python(u) 3: Dive into Python 3*. Praha: CZ.NIC, c2010. CZ.NIC. ISBN 978-80-904248-2-1.

[9] The logo of Raspberry Pi OS. *Wikimedia Commons: File:Raspberry Pi OS Logo.png* [online]. Cambridge: Raspberry Pi Foundation, 2020, 28. května 2020 [cit. 2021-12-03]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/d/d1/Raspberry_Pi_OS_Logo.png

[10] *Asterisk: open source framework for building communications* [online]. Markham, Canada: Sangoma Technologies Corporation, 2021 [cit. 2021-12-03]. Dostupné z: https://www.asterisk.org/

[11] *PJSUA2 Documentation* [online]. Spojené královstvi: Teluu, 2014 [cit. 2021-12-04]. Dostupné z: https://www.pjsip.org/docs/book-latest/html/index.html

[12] *RFC 3261: SIP: Session Initiation Protocol* [online]. Reston, USA: The Internet Society, 2002 [cit. 2021-12-03]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3261

[13] *RFC 4566: SDP: Session Description Protocol* [online]. Reston, USA: The Internet Society, 2006 [cit. 2021-12-03]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4566

[14] *RFC 3550: RTP: A Transport Protocol for Real-Time Applications* [online]. Reston, USA: The Internet Society, 2003 [cit. 2021-12-03]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3550

[15] *DTMF Signaling: EECS20N: Signals and Systems* [online]. Berkeley, CA, US: UC Berkeley EECS Dept., 2011 [cit. 2021-12-04]. Dostupné z: https://ptolemy.berkeley.edu/eecs20/week2/dtmf.html

[16] *Original Prusa MK4: 3D tiskárny Original Prusa přímo od Josefa Průši* [online]. Praha: Prusa Research a.s., 2023 [cit. 2023-05-10]. Dostupné z: https://www.prusa3d.com/cs/produkt/original-prusa-mk4-2/

[17] FDM: ČVUT Fakulta strojní. *Fakulta Strojní ČVUT v Praze* [online]. Praha: ČVUT, 2023, 2023 [cit. 2023-05-10]. Dostupné z: https://www.fs.cvut.cz/ostatni/sekce-3dtisk/3dtisk/moznosti-3dtisk/rozdeleni-technologii-3dtisk/fdm-3dtisk/

[18] *Anycubic Photon: ANYCUBIC 3D PRINTING* [online]. Shenzhen: Shenzhen Anycubic Technology Co., 2017 [cit. 2023-05-10]. Dostupné z: https://www.anycubic.com/products/anycubic-photon-3d-printer

[19] *Bambu Lab X1-Carbon Combo 3D Printer: Bambu Lab EU* [online]. Pflugerville (United States): Bambu Lab, 2023 [cit. 2023-05-10]. Dostupné z: https://eu.store.bambulab.com/products/x1-carbon-combo

[20] *Bambu Lab Carbon X1 Combo: Septillion Co., Ltd* [online]. Bangkok: Septillion Co., 2023 [cit. 2023-05-10]. Dostupné z: https://www.septillion.co.th/en/product/bambu-lab-carbon-x1-combo/