VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHYTRÝ VIDEO ZVONEK ZALOŽENÝ NA RASPBERRY PI

SMART VIDEO DOORBELL BASED ON RASPBERRY PI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADAM ONDREJKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZDENĚK MATERNA, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Ondrejka Adam**
Program: Informační technologie
Název: **Chytrý video zvonek založený na Raspberry Pi**
Smart Video Doorbell Based on Raspberry Pi
Kategorie: Vestavěné systémy

Zadání:

1. Proveďte rešerši existujících řešení pro video zvonek založený na Raspberry Pi.
2. Navrhněte vlastní řešení umožňující detekci přítomnosti osoby, streamování obrazu na multimediální centrum (např. OSMC) a posílání notifikací na mobilní zařízení.
3. Navržené řešení implementujte.
4. Proveďte testování řešení.
5. Zdrojové kódy a dokumentaci publikujte na GitHubu.
6. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle doporučení vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Materna Zdeněk, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Táto práca sa zaoberá návrhom a implementáciou vlastného riešenia inteligentného zvončeka založenom na Raspberry Pi. Navrhnuté riešenie narozdiel od komerčne dostupných, funguje plne lokálne bez internetového pripojenia a je open-source. Zvonček umožnuje audiovizuálny živý prenos, detekciu pohybu, notifikácie a mnoho ďalšieho. Riešenie je tiež prispôsobené pre použitie viacerých zvončekov zároveň, disponuje webovou stránkou a desktopovou aplikáciou na zobrazovanie notifikácií. Zvonček dokáže notifikácie po prepojení s aplikáciou Gotify doručovať aj na zariadenia s operačným systémom Android.

Abstract

This thesis is focused on the design and implementation of a smart doorbell based on Raspberry Pi. Unlike commercially available smart doorbells, the solution described in this thesis offers full functionality (motion detection, audiovisual streaming, and notifications) even without internet access. The doorbell operates fully locally and enables the user to use more doorbells at the same time. A web page and desktop app for displaying notifications was also implemented for the doorbell. After connecting to the android app Gotify, notifications can also be delivered to a device with the Android operational system.

Kľúčové slová

Raspberry Pi, picam, systemd, ffmpeg, RTMP, Nginx, Python, Gotify, Bootstrap, Electron

Keywords

Raspberry Pi, picam, systemd, ffmpeg, RTMP, Nginx, Python, Gotify, Bootstrap, Electron

Citácia

ONDREJKA, Adam. *Chytrý video zvonek založený na Raspberry Pi.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zdeněk Materna, Ph.D.

Chytrý video zvonek založený na Raspberry Pi

Prehlásenie

Prehlasujem, že som tuto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Zdeňka Materny, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Adam Ondrejka
11. mája 2021

Poďakovanie

Rád by som poďakoval vedúcemu mojej práce, Ing. Zdeňku Maternovi, Ph.D., za jeho rady, odporúčania, ľudský prístup a pomoc počas tvorby tejto práce. Ďakujem aj svojim rodičom za ich podporu počas celej doby štúdia a vecné rady do života, ktoré ma ním sprevádzajú a budú vždy sprevádzať. Rád by som taktiež poďakoval svojej skvelej priateľke Tereze Lapinovej za pomoc s gramatickou kontrolou a psychickou podporou počas tvorby tejto práce v tak neľahkej dobe ako je táto. Na záver by som rád poďakoval mojim priateľom, menovite Matej Soroka, Marek Molisch (Molitan), Teodor Žatko (Téčko), Ing. Zuzana Beníčková a budúcej sociologičke Sáre Šujanovej za ich priateľstvo, ochotu, výpomoc a kamarátsku podporu počas môjho štúdia na vysokej škole. S vami je život lepší, vďaka.

Obsah

1 Úvod **2**

2 Existujúce riešenia **3**
2.1 Komerčne dostupné riešenia 3
2.2 Kutilské riešenia 5

3 Návrh **7**
3.1 Raspberry Pi 7
3.2 Hardvérové komponenty a periférie zvončeka 9
3.3 Notifikácie 13
3.4 Streaming 17
3.5 systemd 20

4 Implementácia **21**
4.1 Pripojenie periférií 21
4.2 Systémové služby a konfigurácia 21
4.3 Hlavný riadiaci skript 23
4.4 Notifikácie 27

5 Testovanie **30**

6 Možné rozšírenia **32**
6.1 Rozšírenia zvončeka 32
6.2 Rozšírenia softvéru 34

7 Nedostatky riešenia **35**
7.1 Odozva 35
7.2 Absencia podpory duplexnej komunikácie 35
7.3 Napájanie adaptérom 35

8 Záver **37**

Literatúra **38**

A Testovanie **40**

B Gotify aplikácia **42**

C Dropbox **48**

Kapitola 1

Úvod

Cieľom tejto práce bolo vytvoriť vlastnú alternatívu inteligentného zvončeku založeného na Raspberry Pi. Práca popisuje analýzu existujúcich komerčne dostupných produktov (a ich prípadných nedostatkov), priebeh návrhu a implementácie vlastnej alternatívy, či problémy, na ktoré som počas tvorby tejto práce narazil.

Popis zahŕňa ako softvérovú realizáciu – zasielanie notifikácií a streaming, tak aj hardvérovú realizáciu – tou je zvonček (ktorého základ tvorí Raspberry Pi) disponujúci kamerou, pohybovým senzorom, digitálnym mikrofónom a tlačidlom na zvonenie. Zvonček taktiež poskytuje jednoduchú webovú stránku a desktopovú aplikáciu na ľahké a prehľadné zobrazenie najnovších notifikácií. Mnou vytvorené riešenie disponuje oproti komerčným zvončekom viacerými výhodami. Za hlavné pozitíva považujem plne lokálne fungovanie vrátane funkcionality notifikácií – zariadenie je funkčné aj pri výpadku internetového pripojenia, či otvorenosť modifikáciám alebo pridaniu novej požadovanej funkcionality technicky zdatnejším užívateľom.

Kapitola 2

Existujúce riešenia

Inteligentné zvončeky sú v dnešnej dobe pomerne populárnym produktom, ktorý však stojí nemalé peniaze. V tejto kapitole sú krátko popísane a porovnané najpopulárnejšie komerčne dostupné riešenia inteligentných zvončekov.

2.1 Komerčne dostupné riešenia

Najznámejším a veľmi rozšíreným inteligentným zvončekom je **Ring Doorbell**. Spoločnosť **Ring** bola pôvodne založená Jamiem Siminoffom pod menom Doorbot. Vo februári roku 2018 bola odkúpená známou spoločnosťou Amazon za približne 1 miliardu amerických dolárov [15]. Spoločnosť Ring ponúka svoj zvonček v štyroch variantách:

- Classic
- Plus
- Pro
- Elite

Drahšie varianty sú oproti lacnejším vybavené funkcionalitou ako napríklad funkcia *Pre-Roll*, ktorá umožňuje vidieť záznam 5 sekúnd pred tým, ako došlo k stlačeniu tlačidla [9]. Podľa internetového obchodu Alza[1] je najpredávanejším inteligentným zvončekom na lokálnom trhu v čase vypracovania tejto práce zvonček **Netatmo Doorbell**.

2.1.1 Nedostatky komerčných riešení

Hlavnou motiváciou k vypracovaniu vlastnej alternatívy je fakt, že žiadne z komerčne dostupných riešení:

- nie je open-source,
- neposkytuje oficiálne API[2] ani knižnicu,
- nie je plne lokálne.

[1] https://www.alza.sk/najpredavanejsie-najlepsie-inteligentne-zvonceky-s-kamerou/18867872.htm

[2] Application programming interface – rozhranie pre programovanie aplikácií

Komerčné riešenia inteligentných zvončekov nie sú podľa môjho subjektívneho názoru cenovo najprístupnejším produktom, keďže sa cena často pohybuje v tisícoch korún viz tabuľka 2.1. V prípade zvončeku Ring Doorbell je nepríjemným prekvapením nutnosť doplácať poplatky za funkcionalitu ako lokálne ukladanie záznamov na dobu dlhšiu ako 30 dní.

Obr. 2.1: Zvonček Ring Video Doorbell 3, prevzaté z [9].

Nakoľko SD[3] karty týmto zvončekom nie sú podporované, všetky záznamy sú uložené online. Prevažná väčšina komerčne dostupných zvončekov potrebuje k svojmu chodu vlastnú aplikáciu. Táto aplikácia je využitá pri komunikácii a doručovaní notifikácií, prípadne na prehliadanie záznamov zvončeka. K zvončeku **Ring Doorbell** je na internete dohľadateľných zopár tzv. *reverse-engineered* knižníc[4] pre jazyk **Python**[5]. Väčšina základných verzií zvončekov od rôznych výrobcov poskytuje zhruba podobnú funkcionalitu. Líšia sa často len v podpore štandardov bezdrôtového sieťového prenosu, prípadne v spôsobe napájania (adaptér alebo batérie, zriedka napájanie zo siete).

	Video Doorbell 3	Video Doorbell 3 Plus	Video Doorbell Elite	Netatmo Doorbell
výrobca	Ring	Ring	Ring	Netatmo
cena	4600 CZK	5100 CZK	10300 CZK	8600 CZK

Tabuľka 2.1: Cenové porovnanie jednotlivých verzií zvončeku Ring Doorbell a zvončeku Netatmo Doorbell, ceny sú prevzaté z [9], [23].

[3] Secure Digital
[4] https://github.com/tchellomello/python-ring-doorbell
[5] https://www.python.org/

2.2 Kutilské riešenia

Na internete je dostupných viacero kutilských alebo takzvaných *DIY*[6] riešení. Nasledujúci text popisuje niektoré z nich.

2.2.1 AI Smart Doorbell

Tento inteligentný zvonček, ktorý je taktiež založený na Raspberry Pi je dostupný v repozitári na stránke GitHub[7]. Využíva cloudovú platformu Microsoft **Azure** a protokol **MQTT**[8] (postavený na TCP/IP).

MQTT

Protokol MQTT je jednoduchý a nenáročný protokol, ktorý slúži k predávaniu správ medzi klientmi prostredníctvom centrálneho bodu. Tento centrálny bod sa nazýva *broker*. Protokol MQTT používa k prenosu protokol TCP[9] a návrhový vzor *publisher–subscriber*[10]. Existuje teda jeden broker, ktorý sa stará o výmenu správ a triedi ich do tzv. *topics*[11]. Zariadenie môže v danej téme:

- **publikovať** (publish) – posielať dáta brokerovi, ktorý ich následne triedi a predáva ďalším zariadeniam
- **odoberať** (subscribe) – zariadeniu, ktoré je prihlásené k odberu témy následne broker posiela všetky správy z danej témy

Jedno zariadenie môže v téme zároveň publikovať aj odoberať [19].

Microsoft Azure

Cloudová platforma Microsoft Azure umožňuje využitie rôznych služieb, aplikácii či API. Tento zvonček využíva Azure službu **Face**, ktorá umožňuje rozpoznávanie tvári. Na endpoint tohto API zvonček odošle fotografiu tváre, ktorá je ďalej službou Face spracovaná a vyhodnotená. Pomocou protokolu MQTT je možné do témy *doorbell/live* odosielať rôzne správy, ako napríklad:

- Správa s obsahom **live_30sec** – nahrá 30 sekundové video a uloží ho do nakonfigurovaného adresára
- Správa s obsahom **photo** – vytvorí fotografiu
- Správa s obsahom **{"new_person":"True","name":"meno"}** – pridá tvár (posledná vytvorená fotografia) do kolekcie tvári ktoré sú použité pri vyhodnocovaní službou Face v Microsoft Azure

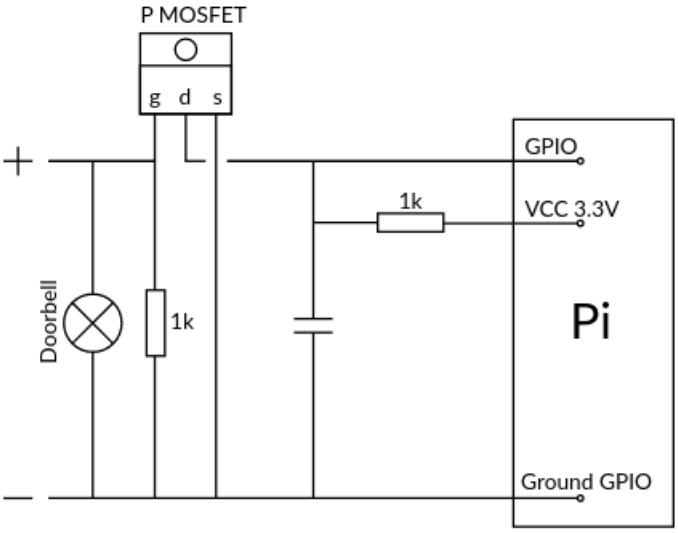
Tento zvonček je zaujímavým a moderným riešením, ktoré využíva relevantné technológie. Za silnú stránku považujem najmä integráciu so službou Face. Tento zvonček však (priamo) nepodporuje prenos na multimediálne centrum, nie je plne lokálny a neumožňuje detekciu pohybu (nemá pohybový senzor).

[6] Do It Yourself – Urob si sám
[7] https://github.com/Hurleyking/AI_Smart_PI_Doorbell
[8] Message Queuing Telemetry Transport
[9] Transmission Control Protocol – Protokol riadenia prenosu
[10] Vydavateľ–odberateľ
[11] Tém

2.2.2 Doorbell

Toto riešenie je taktiež dostupné v repozitári na stránke GitHub[12]. Zvonček je založený na Raspberry Pi *Zero* a nie je vybavený kamerou, mikrofónom ani pohybovým senzorom. Jeho úlohou je digitalizácia klasického analógového zvončeka a odosielanie notifikácií. K odosielaniu a príjmaniu notifikácií využíva službu **Pushover**. Notifikácia je odoslaná ak dôjde k stlačeniu tlačidla domového zvončeka – toto je zachytávané GPIO zbernicou Raspberry Pi (Sekcia 3.1.1).

Obr. 2.2: Raspberry Pi Zero pripojený k obvodu domového zvončeka, prevzaté z repozitára.



Obr. 2.3: Schéma obvodu, prevzaté z repozitára.

Aj keď zvonček neumožňuje audio-vizuálny prenos na žiadne zariadenie či multimediálne centrum, je toto riešenie zaujímave a hodné pozornosti, nakoľko neprináša nový funkčný celok ale rozširuje už existujúce zariadenie o ”inteligentnú” funkcionalitu ako notifikácie. V tomto prípade je nevýhodou opäť nutnosť pripojenia k internetu (služba Pushover).

[12] https://github.com/lesander/doorbell

Kapitola 3

Návrh

V tejto kapitole sú popísané technológie, ktoré boli vybraté pre implementáciu a realizáciu praktickej časti tejto práce. Cieľom je predstaviť čitateľovi platformu, na ktorej je zvonček postavený (Sekcia 3.1) a technológie ktoré boli použité pri implementácii funkcionality zvončeka. Implementácia zvončeka a jej návrh sa dá rozdeliť do dvoch logických celkov. Prvým je hardvérová časť, ktorá zahŕňa hardvérové komponenty a periférie zvončeka (Sekcia 3.2). Druhým je softvérová časť, ktorá sa primárne zaoberá funkcionalitou notifikácií (Sekcia 3.3) a živého prenosu (ďalej bude používaný anglický pojem *streaming*) (Sekcia 3.4). Streaming vyžadoval použitie webového servera **Nginx** (Sekcia 3.4.3), vytvorenie hlavného riadiaceho skriptu (Sekcia 4.3) a systémových služieb (Sekcia 3.5). Pre prácu s notifikáciami bol použitý open-source server **Gotify** (Sekcia 3.3.1).

3.1 Raspberry Pi

Raspberry Pi je cenovo dostupný a kompaktný jednodoskový počítač, nie väčší ako bežná platobná karta. Pôvodne bol vytvorený ako učebná pomôcka pre výučbu programovania a počítačových technológií, no všeobecne sa teší veľkému komerčnému úspechu. Svoje uplatnenie našiel hlavne vďaka svojej modularite, univerzálnosti a nízkej cene.

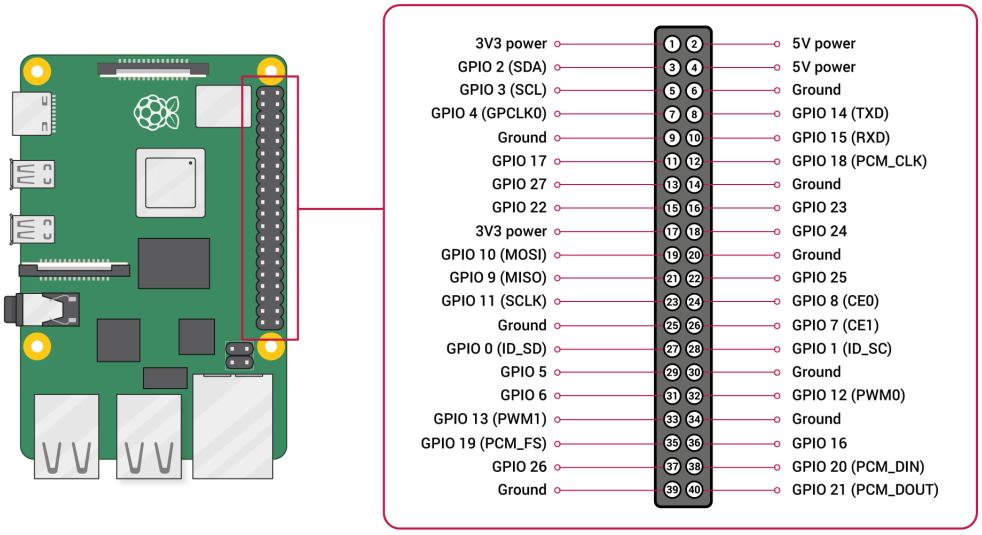
Obr. 3.1: Raspberry Pi 3 B+, prevzaté[1].

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Je taktiež veľmi populárnou platformou často využívanou pri tvorbe vstavaných systémov. Svoje uplatnenie tiež nachádza v novej a čoraz populárnejšej oblasti **IoT**[2]. Vývoj nových verzií naďalej prebieha aj v prítomnosti.

3.1.1 Hardvérová špecifikácia modelu

Pri tvorbe tejto práce bol použitý model B+ tretej generácie. Tento model disponuje štvorjadrovým 64-bitovým procesorom **Broadcom BCM2837B0, Cortex-A53** s frekvenciou 1.4GHz a pamäťou RAM o veľkosti 1 GB ktorú zdieľa s grafickým procesorom **Broadcom VideoCore IV**. Okrem gigabitového ethernet pripojenia podporuje **Raspberry Pi 3 B+** aj bezdrôtové WiFi pripojenie so štandardom `IEEE 802.11ac`[3], ktorý podporuje ako 2.4 GHz tak aj 5 GHz pripojenie. Pre účely implementácie bola využitá aj GPIO[4] zbernica, ktorá disponuje 40-timi pinmi viz Obrázok 3.2.



Obr. 3.2: Diagram GPIO zbernice na Raspberry Pi 3 B+, prevzaté z [7].

Zvonček využíva piny 1, 2, 6, 7 a 10. Prehľad použitých pinov a ich význam je možné nájsť v Sekcii 4.1. Piny sú programovateľné a pri práci s nimi som využil knižnicu **RPi.GPIO**[5] jazyka **Python**. Kompletná špecifikácia spolu s všetkými rozhraniami ktoré Raspberry Pi poskytuje je dostupná na stránkach výrobcu viz [6].

[2] Internet of Things - internet vecí
[3] https://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm
[4] General purpose input output - vstup a výstup pre všeobecné použitie
[5] https://pypi.org/project/RPi.GPIO/

3.1.2 Rasperry Pi OS

Operačný systém **Raspberry Pi OS** (pôvodne známy ako Raspbian) je operačný systém odvodený z linuxovej distribúcie **Debian**, oficiálne poskytovaný nadáciou **Raspberry Pi Foundation**. Raspberry Pi OS (Raspbian) je primárne určený pre jednodoskové počítače Raspberry Pi no jeho využitie nieje obmedzené iba na túto platformu. Je vysoko optimalizovaný pre procesory s architektúrou ARM, ktoré sú používané v modeloch Raspberry Pi. Ako hlavné desktopové prostredie používa *PIXEL*[6], ktoré tvorí modifikované prostredie *LXDE* a správca okien (window manager) *Openbox*.

3.1.3 Raspberry Pi OS Lite

Okrem štandardného operačného systému s desktopovým prostredím je možné využiť aj takzvanú *headless* verziu pre prípady, kedy užívateľ alebo výsledný produkt nepotrebuje k svojmu fungovaniu monitor (nieje prítomná pracovná plocha) či klávesnicu a myš. Výhoda tejto verzie spočíva v menšej spotrebe úložiska (300 MB oproti 1.3 GB) a výkonnostnej náročnosti. Táto varianta operačného systému je použitá aj pre účely tejto práce.

3.2 Hardvérové komponenty a periférie zvončeka

Zvonček má viacero hardvérových komponentov a periférií, ktoré sú súčasťou implementácie funkcionality. Klasický zvonček by mal predovšetkým obsahovať tlačidlo na zvonenie (Sekcia 3.2.4). Vytvorená implementácia zvončeku navyše disponuje pohybovým senzorom (Sekcia 3.2.2), kamerou (Sekcia 3.2.1), mikrofónom (Sekcia 3.2.3) a dodatočnou softvérovou implementáciou, ktorá ho robí "inteligentným". V tejto sekcii krátko popíšem jednotlivé komponenty a ich fungovanie. Za veľkú výhodu platformy Raspberry Pi považujem modularitu. Jednotlivé komponenty je ľahké časom doplniť, vymeniť za lepšie alebo iné tak, aby viac vyhovovali konkrétnym požiadavkám užívateľa (Kapitola 6).

3.2.1 Kamera

K zvončeku je pripojený širokouhlý kamerový modul pre Raspberry Pi zapožičaný fakultou.

Obr. 3.3: Širokouhlý kamerový modul, prevzaté[7].

[6] Pi Improved Xwindows Environment Lightweight
[7] www.distrelec.de/en/hd-wide-angle-camera-module-raspberry-pi-rpi-wwcam/p/30037327

Modul je kompatibilný so všetkými verziami Raspberry Pi a pripája sa pomocou 15 centimetrového *FFC*[8] konektoru viz Obrázok 3.3. Raspberry Pi disponuje samostatným portom pre modul kamery priamo na doske. Oproti bežnej kamere umožňuje širokouhlý kamerový modul až 160 stupňové zorné pole (pre klasický kamerový modul je to typicky len 72 stupňov). Kamera má štyri úchyty na pripevnenie k podkladu. Na zadnej strane modulu sa nachádza samolepiaci prvok, ktorý som použil pri uchytení kamery do stojatej polohy pre účely testovania (Kapitola 5).

3.2.2 Pasívny infračervený senzor

Pasívny infračervený senzorový modul alebo aj takzvaný **PIR**[9] (ďalej bude v texte používaná táto skratka) modul slúži k detekcii pohybu pred zvončekom. Tento modul bol taktiež spolu s kamerovým modulom zapožičaný fakultou. Okrem iného disponuje dvoma potenciometrami, ktoré umožňujú nastaviť citlivosť senzoru a odozvu medzi jednotlivými signálmi vysielanými senzorom (podrobnejšie viz Obr. 3.7). PIR modul sa pripája pomocou troch GPIO pinov, prehľad pinov a ich zapojenia je možné nájsť v Sekcii 4.1.

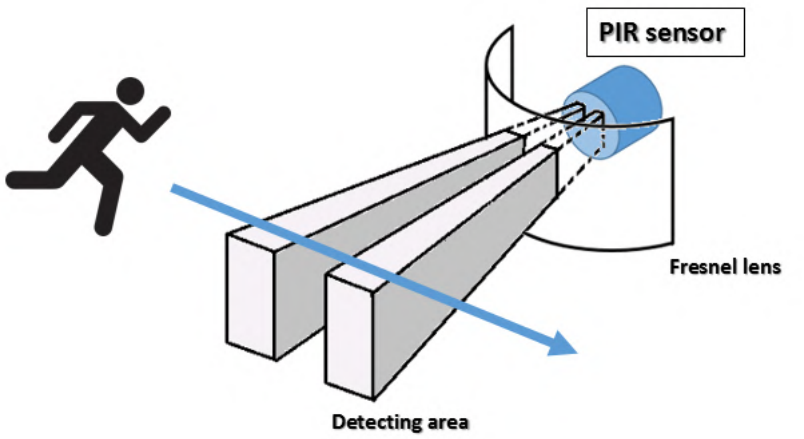
Obr. 3.4: PIR modul, prevzaté z [25].

Obr. 3.5: Senzor pod Fresnelovou šošovkou, prevzaté z [25].

Modul detekuje pohyb pomocou *pyroelektického* senzoru, ktorý je skrytý pod tzv. *Fresnelovou šošovkou*[10].

[8] Flexible Flat Cable
[9] Passive Infrared
[10] Fresnel lens

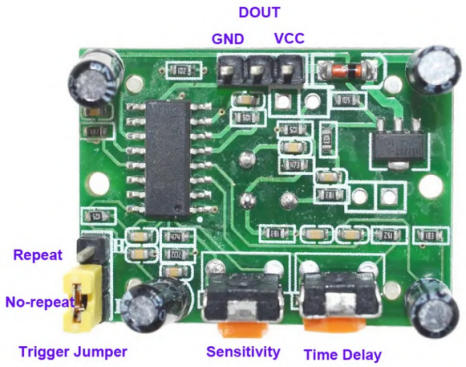
Pokiaľ do poľa detekcie senzoru vstúpi napríklad človek, teplo vyžarované jeho telom (v podobe infračerveného žiarenia) je zachytené pyroelektickým senzorom, ktorý následne generuje elektrický potenciál – toto sa nazýva pyroelektický jav. Fresnelova šošovka slúži k sústredeniu žiarenia z okolia na pyroelektický senzor [21].



Obr. 3.6: Princíp detekcie pohybu modulom, prevzaté z [12].

Jednotlivé piny modulu majú rôzne významy viz Obrázok 3.7. Okrem toho ponúka modul viaceré možnosti nastavenia buď pomocou potenciometrov alebo jumperu[11]:

- **Sensitivity** - citlivosť senzora
- **Time delay** - dĺžka trvania výstupu modulu (`GPIO.HIGH` na výstupnom pine)
- **Trigger jumper** - podľa umiestnenia jumperu nastavuje či sa má doba výstupu (*time delay*) obnoviť pri novej detekcii (tzv. *repeatable* režim[12]) [25]



Obr. 3.7: Možnosti nastavenia modulu, prevzaté z [12].

[11] Skratovacej prepojky
[12] Opakovateľný

3.2.3 Mikrofón

Pre prenos zvuku bol vybratý základný USB[13] mikrofón viz Obrázok 3.8. O synchronizáciu videa z kamery a zvuku z mikrofónu sa pre účely streamingu stará program, respektíve služba *picam* (Sekcia 3.4.4). Vybratý mikrofón považujem pre účely práce za vhodný najmä kvôli jeho malým rozmerom a jednoduchému zapojeniu.

Obr. 3.8: Použitý USB mikrofón, prevzaté z[14].

Mikrofón je však možné jednoducho vymeniť za iný alebo lepší. V skorších fázach vývoja pri prototypovaní bola namiesto tohto mikrofónu použitá USB web-kamera značky **Canyon** s vstavaným mikrofónom viz Obrázok 3.9.

Obr. 3.9: Webkamera, ktorá bola prvotne použitá namiesto USB mikrofónu.

3.2.4 Tlačidlo na zvonenie

Tlačidlo na zvonenie je kľúčovým prvkom každého zvončeka. V tejto práci bolo použité klasické spínacie tlačidlo a nepájivé pole. Tlačidlo využíva GPIO piny s fyzickým číslom 1 a 10, plus tzv. *pull-down* rezistor. Pri stlačení tlačidla a zopnutí obvodu, ktorý je pripojený k 3.3V pinu na Raspberry Pi je prečítaná v slučke obsluhy periférií hodnota `GPIO.HIGH` (logická 1). Keď však tlačidlo nieje stlačené, môže byť prečítané čokoľvek. Preto je nutné použiť tzv. *pull-down* rezistor, ktorý zaistí že pokiaľ je nieje obvod zopnutý na vstupe bude hodnota `GPIO.LOW` (logická 0) [7].

[13] Univerzálna sériová zbernica

3.2.5 Kryt

Na internete je možné nájsť viacero komerčne dostupných krytov pre Raspberry Pi, ktoré sú prispôsobené na použitie s kamerovým modulom a PIR modulom. Takéto puzdro môže tiež disponovať úchytom na stenu a prístupom k portom Raspberry Pi viz Obrázok 3.10.

Obr. 3.10: Puzdro prispôsobené na použitie PIR a kamerového modulu, prevzaté z [5].

Toto puzdro je nutné prispôsobiť pre montáž tlačidla, no napriek tomu je veľmi dobrým základom pre výrobu obalu zvončeka. Cena takéhoto krytu sa pohybuje okolo 400 korún[15]. Vo fáze návrhu bolo pôvodne plánované zahrnúť v implementácii aj toto puzdro. Nakoľko sa však verzia Raspberry Pi (Model 3B+) zapožičaného fakultou prestala výrobcom puzdra podporovať začiatkom marca tohto roku, ponechávam možnú implementáciu tejto časti pre užívateľa s vhodným a novším hardvérom (Raspberry Pi 4).

3.3 Notifikácie

Dôležitou funkciou zvončeka sú užívateľské notifikácie. Pokiaľ zvonček zachytí pohyb alebo niekto zazvoní, užívateľ obdrží notifikáciu.

3.3.1 Gotify

Základným stavebným kameňom celej funkcionality notifikácií je open-source server **Gotify**. Pre účely vytvárania a správy notifikácií poskytuje Gotify okrem webového rozhrania aj vlastné *REST*[16] API. Hlavnou výhodou Gotify je možnosť byť užívateľsky plne *self-hosted*[17]. Toto bolo aj motiváciou pri výbere technológií, keďže výsledný zvonček funguje plne lokálne aj bez internetového pripojenia[18]. Gotify taktiež poskytuje vlastnú aplikáciu pre mobilnú platformu Android. Vo webovom rozhraní Gotify je nutné vygenerovať prístupové tokeny. Tie sa delia na dva druhy – token klienta (prístup k notifikáciám) a token aplikácie (aplikácia je napríklad skript, ktorý notifikácie vytvára).

[15] https://thepihut.com/products/pir-camera-case-for-raspberry-pi-4-3

[16] Representational state transfer architektúra

[17] Server je spustený v lokálnej sieti užívateľa

[18] Pri výpadku internetového pripojenia nie sú záznamy zálohované online, no zvonček plne funguje, záznamy ukladá lokálne

3.3.2 Rôzna priorita notifikácií a formát JSON

S Gotify notifikáciami sa pracuje skrz REST API. Obmedzenú funkcionalitu ich správy, ako napríklad mazanie, nájdeme aj vo webovom rozhraní B.3. Na jednotlivé endpointy[19] sú klientmi posielané *HTTP*[20] požiadavky s rôznymi metódami. Gotify REST API má mnoho rôznych endpointov, pre zvonček a jeho funkcionalitu sú najdôležitejšie následovné:

- `/message` - všetky notifikácie
  - `GET` - vyžiadanie všetkých notifikácií
  - `POST` - vytvorenie novej notifikácie
- `/stream` - novovytvorená notifikácia[21]

Ak chce klient (napríklad zvonček) notifikáciu vytvoriť, pošle HTTP `POST` požiadavku na endpoint `/messages`[22]. Táto požiadavka obsahuje reťazec vo formáte *JSON*[23]. JSON je formát výmeny dát ľahko čitateľný človekom, ktorý je (aj napriek svojmu názvu) jazykovo nezávislý. Reťazec vo formáte JSON tvoria dve štruktúry - kolekcia dvojíc meno/hodnota a zoradený list hodnôt. V JSON reťazci, ktorý je v HTTP požiadavke odoslaný Gotify serveru, je možné špecifikovať okrem iného napríklad:

- `message` – špecifikuje obsah tela notifikácie
- `priority` – číselná priorita (čím väčšie číslo tým väčšia priorita)
- `title` – titulok (nadpis) notifikácie

Gotify toho však umožňuje oveľa viac, možné je napríklad aj odosielanie obrázkov – podrobnejšie sa o všetkých možnostiach dá dočítať v dokumentácii [4]. Notifikácie posielané zvončekom sa podľa udalosti, ktorá ich vyvolala delia na dva druhy:

- **Notifikácia s vysokou prioritou** – vyvolaná stlačením tlačidla a definovaná ako číslo 5
- **Notifikácia s nízkou prioritou** – vyvolaná detekciou pohybu PIR modulom a definovaná ako číslo 1

Vytváranie a odosielanie HTTP požiadaviek potrebných k upozorneniu užívateľa na vstup má na starosti hlavný riadiaci skript (Sekcia 4.3). Notifikácie sa však nielen vytvárajú, ale aj príjmajú a spracovávajú (Sekcia 3.3.3).

3.3.3 Zobrazovanie notifikácií

Notifikácií počas dňa môže byť viacero a je žiadúce, aby ich bolo možné prehľadne zobraziť. Bolo teda potrebné vytvorenie *HTML*[24] stránky s užívateľsky prívetivým vzhľadom, kde si tento prehľad môže užívateľ zobraziť.

[19] Prístupové body
[20] Hypertext Transfer Protocol – Hypertextový prenosový protokol
[21] Použitie tohto endpointu je špecifické pre desktopovú aplikáciu a jeho podrobnejší popis je možné nájsť v Sekcii 3.3.7
[22] Tomuto URI musí IP predchádzať adresa zariadenia, kde je spustený server Gotify a port nakonfigurovaný pre tento server. Výsledná URL môže teda vyzerať napríklad následovne http://localhost:8080/messages
[23] JavaScript Object Notation – JavaScriptová objektová notácia
[24] Hyper Text Markup Language – Hypertextový značkový jazyk

Pre tvorbu tejto stránky som si vybral *CSS*[25] framework **Bootstrap 4**. K stránke som taktiež vytvoril krátky skript v jazyku **JavaScript**, ktorý sa stará o automatickú aktualizáciu notifikácií a spracovanie odpovede zo servera Gotify, bez nutnosti opakovane načítavať stránku. Táto web stránka bola použitá aj pri tvorbe desktopovej aplikácie. Podrobnejšie sa o implementácii a formáte notifikácií píše v Sekcii 4.4.

3.3.4 Bootstrap 4

Bootstrap je populárny open-source framework slúžiaci k vytváraniu webových stránok a aplikácií, ktorý využíva HTML, JavaScript a CSS. Bootstrap je tzv. *mobile-first* framework - jeho komponenty sú navrhované primárne pre zobrazenie na rozlíšení bežného mobilného zariadenia. Poskytuje sadu predpripravených hotových štýlov, ktoré je možné aplikovať na jednotlivé HTML elementy, pričom nieje nutné vytvárať vlastné CSS štýly. Bootstrap nie je nutné inštalovať. Do hlavičky stačí vložiť element `<link>` s odkazom do *CDN*[26]. Všetky komponenty sú responzívne, a preto sa napríklad webová stránka či desktopová aplikácia (Sekcia 3.3.6) vždy zobrazuje korektne aj pri rôznych rozlíšeniach displeja. Pri tvorbe tejto sekcie som čerpal z [20] a [14].

3.3.5 JavaScript a AJAX

JavaScript je objektovo orientovaný skriptovací jazyk, syntaxou podobný jazyku **C**. JavaScript umožňuje prístup k jednotlivým *DOM*[27] elementom, ich vytváranie či manipuláciu. Technológia **AJAX**[28] umožňuje asynchrónne serverové volania. Server vráti odpoveď, ktorú môžeme v JavaScript kóde ďalej spracovať. Takto je možné manipulovať s jednotlivými *DOM* elementami bez nutnosti celú stránku znova načítavať.

3.3.6 Desktopová aplikácia

Pre zobrazenie prehľadu notifikácií a najmä možnosť vytvárania systémových notifikácií som navrhol a implementoval jednoduchú desktopovú aplikáciu. Pri tvorbe tejto aplikácie som použil populárny open-source softvérový framework **Electron**[29] a už vytvorenú webovú stránku (Sekcia 3.3.3). V tejto sekcii krátko popíšem spomínaný framework a správcu balíkov **npm**[30]. Podrobnejšie je implementácia aplikácie a systémových notifikácií popísaná v Sekcii 4.4.3. Táto aplikácia je tiež ľahko rozšíriteľná a jej možné rozšírenie je priblížené v Sekcii 6.2.1.

3.3.7 Protokol WebSocket

Pomocou protokolu **WebSocket** je možné posielať prostredníctvom jedného TCP pripojenia dáta oboma smermi zároveň. Protokol WebSocket zavádza socket[31], ktorý udržiava stály kanál spojenia so serverom. Pripojenie WebSocket spoznáme podľa prefixu *ws://* alebo *wss://* v adresnom riadku. WebSocket podporuje šifrované aj nešifrované spojenie.

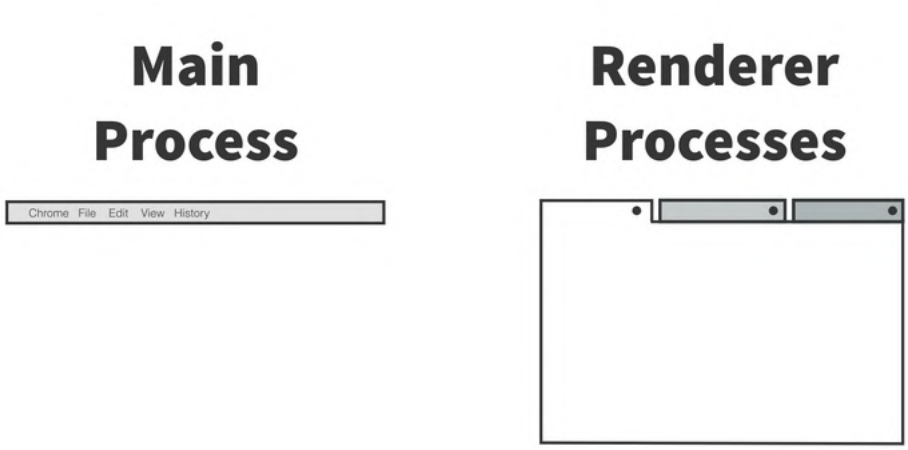
[25] Cascading Style Sheets – Kaskádové štýly
[26] Content Delivery Network – sieť doručenia obsahu
[27] Document Object Model – objektový model dokumentu
[28] Asynchronous JavaScript and XML
[29] https://www.electronjs.org/
[30] https://www.npmjs.com/
[31] Schránku

3.3.8 npm

Správca balíkov **npm**[32] je správca balíkov pre programovací jazyk JavaScript. Je tiež základným správcom balíkov pre softvérový systém **Node.js**[33]. Poskytuje *CLI*[34] klienta (`npm`) a online databázu dostupných balíkov (`npm registry`). K databáze sa pristupuje pomocou klienta, vyhľadávanie a prehliadanie balíkov je možné na webovej stránke npm. Tento nástroj bude slúžiť k správe závislostí desktopovej aplikácie, jej jednoduchú inštaláciu a spustenie.

3.3.9 Electron

Framework **Electron** je vyvíjaný spoločnosťou **GitHub** a slúži k vytváraniu natívnych aplikácií pomocou HTML, CSS a JavaScriptu. Electron kombinuje **Node.js** a jadro prehliadača **Chromium**. Electron vytvára grafické užívateľské prostredie pomocou kombinácie natívnych prvkov operačného systému (napríklad lišta) a HTML stránky. Umožňuje prácu so súborovým systémom alebo funkcionalitu **systémových notifikácií**. Funkcionalita systémových notifikácií bola hlavnou motiváciou vytvorenia aplikácie.



Obr. 3.11: Main a rendererovací proces, prevzaté z [18].

Aplikáciu v Electrone typicky tvoria dva typy procesov - *hlavný* a *renderovací proces*. Hlavný proces vytvára pre jednotlivé okná renderovacie procesy. Renderovací proces sa stará o vykresľovanie HTML stránky.

[32] Node Package Manager
[33] https://nodejs.org/en/
[34] Command Line Interface – rozhranie príkazového riadku

3.4 Streaming

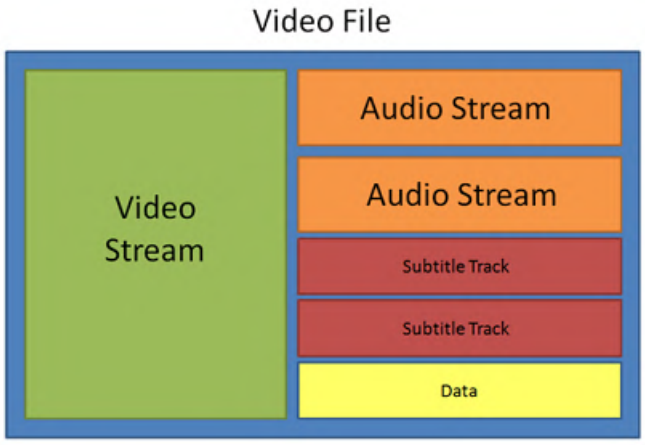
Pri stlačení tlačidla na zvončeku je užívateľovi odoslaná notifikácia o prebiehajúcom audiovizuálnom streame na multimediálne centrum - takto je možné vidieť, čo sa pred zvončekom deje. Nasledujúci text popisuje technológie využité pri implementácii funkcionality streamingu a potrebný teoretický základ.

3.4.1 Kontajner, formát kódovania a kodek

Predtým ako popíšem samotný streaming, je nutné vysvetliť spôsob, akým budú prenášané dáta uložené a spracovávané. Kontajner (obálkový formát) je súbor, ktorý uchováva video, audio, prípadne titulky a podobne. Video, audio a ostatné dáta, ktoré kontajner obsahuje sa nazývajú *streams* (tzv. prúdy). Existujú rôzne druhy kontajnerov, ako napríklad:

- `AVI`
- `MKV (Matroska)`
- `MPEG-TS`

Video a zvuková stopa v kontajneri môžu byť v rôznom formáte kódovania (dát). Niektoré formáty vyžadujú špecifický druh kontajneru, naopak niektoré kontajnery sú špecifické pre určité formáty [10], [26].



Obr. 3.12: Zjednodušený príklad kontajneru.

Formátov kódovania, je podobne ako kontajnerov, mnoho druhov a to napríklad:

- `H.264` – podpora hardvérového enkódovania aj dekódovania na Raspberry Pi [6]
- `MPEG-4` – podpora hardvérového dekódovania na Raspberry Pi [6]
- `WMV (Windows Media Video)`

Veľkosť súboru môže byť často priveľká, a preto je nutné dáta komprimovať. *Kodek* je zariadenie alebo softvér, ktoré môže pracovať ako kodér alebo dekodér. Ako kodér zaisťuje kodek komprimáciu (kompresiu) dát, teda zmenšenie ich objemu. Pri komprimácii je dôležité

poznať pojem *bit rate* - ten určuje koľko dát je uložených pre každú prehrávanú sekundu. Čim je väčší *bit rate*, tým menšia je komprimácia, čo má za výsledok vyššiu výslednú kvalitu no taktiež väčšiu veľkosť súboru. Komprimácia môže prebiehať buď bez strát (tzv. *lossless* komprimácia) alebo so stratami (tzv. *lossy* komprimácia)[27]. Dôležité je teda pochopiť rozdiely medzi týmito pojmami, ku ktorých zámene často dochádza. Nasledujúce riadky preto obsahujú zhrnutie tejto sekcie. Kontajner je formát súboru, ktorý uchováva video, audio a iné dáta. Dáta, ktoré kontajner obsahuje môžu byť v rôznom formáte kódovania. Kódovanie a kompresiu má na starosti kodek [24].

3.4.2 Protokol Real-Time Messaging Protocol

Protokol *RTMP*[35] je postavený na protokole *TCP* a slúži k prenosu videa, audia a iných dát po sieti v reálnom čase. Protokol RTMP funguje na princípe komunikácie medzi klientom a serverom. Pôvodne bol vyvinutý spoločnosťou **Macromedia**, ktorá bola neskôr odkúpená spoločnosťou Adobe. Protokol prenáša dáta v takzvaných fragmentoch (chunkoch[36]). Základná veľkosť fragmentu pre zvuk je 64 bajtov pre audio a 128 bajtov pre video. Veľkosť týchto fragmentov sa však môže počas prenosu dynamicky meniť po vyjednaním medzi klientom a serverom [17]. Existuje niekoľko alternatív protokolu RTMP. Nakoľko však program picam priamo podporuje tento protokol a v repozitári obsahuje aj predpripravenú konfiguráciu pre server Nginx [13], rozhodol som sa ho použiť taktiež.

3.4.3 Nginx

Protokol RTMP je klient-server protokol. Klientom bude samotný zvonček, ktorý pomocou kamery a mikrofónu zachytáva video a zvuk. **Nginx** (engine X) je populárny, voľne dostupný open-source webový server. Nginx bude v implementácii plniť úlohu RTMP servera. K samotnému Nginx je nutné nainštalovať a nakonfigurovať modul pre podporu RTMP prenosu. Modul aj návod k jeho konfigurácii je dostupný v oficiálnom repozitári[37]. Podrobnejší popis implementácie a konfigurácie serveru je možné nájsť v Sekcii 4.2.2.

3.4.4 picam

Pre zachytávanie a odosielanie videa a zvuku zvončekom som si vybral program **picam** vytvorený užívateľom **iizukanao** poskytovaný pod licenciou *GNU LGPL*. Zdrojové kódy a popis funkcionality sú dostupné v oficiálnom repozitári[38]. Program picam umožňuje využiť funkcionalitu kamerového modulu spôsobom kompatibilným s RTMP modulom pre server Nginx. Je v ňom možné jednoducho špecifikovať zariadenie pre zachytávanie zvukovej stopy, čo bolo dôležitým faktom pri výbere. Prvotne bola pre zvonček zvažovaná aj knižnica **picamera**[39]. Picam však umožňuje jednoduché ukladanie z už prebiehajúceho streamu. Predchádza tak problému, ktorý by mohol nastať pri snahe pristupovať ku kamerovému modulu dvoma procesmi zároveň (implementačný problém, na ktorý som narazil pri práci s knižnicou picamera). Podrobnejší popis tohto programu, využitých funkcií a použitia v implementácií je možné nájsť v Sekcii 4.2.1.

[35] Real-Time Messaging Protocol
[36] Blokoch
[37] https://github.com/arut/nginx-rtmp-module
[38] https://github.com/iizukanao/picam
[39] https://picamera.readthedocs.io/en/release-1.13/

3.4.5 FFmpeg

FFmpeg je multimediálny framework, ktorý zvláda kódovanie, dekódovanie, transkódovanie (prekódovanie), multiplexing, streamovanie a mnoho ďalšieho. Je open-source a kompilovateľný na mnohých platformách, ako napríklad *Linux, Windows, Mac OS X, Solaris*. FFmpeg poskytuje viaceré nástroje, ako napríklad `ffmpeg, ffplay, ffprob`.

ffmpeg

Nástroj `ffmpeg`, ktorý je súčasťou multimediálneho frameworku *FFmpeg* je veľmi rýchly konvertor zvuku a videa s ktorým sa pracuje skrz rozhranie príkazového riadku [3]. Tento nástroj dokáže ako svoj vstup využívať dáta zo živého prenosu, preto je vhodný aj pre účely implementácie. Podrobnejší popis tohto nástroja a ako je využívaný zvončekom sa nachádza v Sekcii 4.2.2 a 4.3.1.

3.4.6 Ukladanie záznamov

Pri zaznamenaní pohybu zvončekom je vytvorené krátke 10 sekundové video, ktoré umožňuje užívateľovi vidieť čo sa dialo po zachytení pohybu. Toto video je uložené lokálne a prípadne zálohované v cloudovej službe **Dropbox** pomocou jej oficiálne poskytovaného REST API. Ukladanie záznamov by však malo prebiehať v prvom rade lokálne, za účelom zachovania plnej funkčnosti aj bez internetového pripojenia. Podrobnejší popis implementácie ukladania a práce s Dropbox API je v Sekcii 4.3.1.

3.4.7 Dropbox

Dropbox je cloudová služba vytvorená a poskytovaná spoločnosťou **Dropbox**, ktorá sídli v Kalifornii. Dropbox okrem funkcionality dátového úložiska poskytuje aj rôzne nástroje pre kolaboráciu na zdieľanom obsahu. Okrem webového rozhrania disponuje aj desktopovou aplikáciou pre operačné systémy *Linux, Microsoft Windows a Mac OS X*. Dropbox disponuje vlastným REST API pre prácu so súbormi, ku ktorému je vytvorená aj oficiálna knižnica pre jazyk Python[40,41]. Pre prácu s Dropbox REST API je potrebné si v nastaveniach vytvoriť tzv. *aplikáciu*, v ktorej je následne možné upravovať jednotlivé práva pre prácu so súbormi skrz REST API. Pre autorizáciu využíva Dropbox moderný protokol **OAuth2** [2]. Dropbox navyše umožnuje nastaviť automatické mazanie súborov napríklad každých sedem dní.

3.4.8 OAuth2

Protokol **OAuth 2.0** je protokol slúžiaci k autorizácii API klientov. Zvonček pri práci s Dropbox REST API používa tzv. *Access token*[42], ktorý je nutné vygenerovať v nastaveniach aplikácie v Dropboxe po jej vytvorení. Tento token je pri posielaní HTTP požiadaviek na jednotlivé endpointy použitý k autorizácii prístupu k užívateľovým súborom v službe Dropbox. Pomocou Access tokenu sú teda aj špecifikované jednotlivé práva klienta [1]. Príklad vytvorenia Dropbox Acess tokenu je možné nájsť v prílohe C.1.

[40] https://github.com/dropbox/dropbox-sdk-python
[41] https://dropbox-sdk-python.readthedocs.io/en/latest/
[42] Prístupový token

3.5 systemd

Po zavedení jadra operačného systému je spustený program **init**, ktorého úlohou je spustiť *init skripty* (v prípade **systemd** spustiteľný binárny súbor) starajúce sa o chod systému a obsluhu *démonov*[43]. *Service manažér* je program starajúci sa o sledovanie a ovládanie procesov, ako napríklad ich:

- spúšťanie
- ukončovanie
- overovanie ich stavu (napríklad zlyhania)

Démon je proces, ktorý je spustený v pozadí bez priamej obsluhy od užívateľa. **Systemd** systém sa stará o chod démonov, v systemd tzv. *jednotiek* (units) [16]. Jednotky sa môžu deliť na viaceré typy, ako napríklad:

- Service jednotka[44]
- Mount jednotka[45]
- Target jednotka[46]

Systemd umožňuje užívateľovi vytváranie vlastných jednotiek, ktoré majú byť spustené po zavedení jadra operačného systému. Pre takúto službu je potrebné vytvoriť tzv. unit súbor, ktorý môže byť umiestnený napríklad do adresára `/etc/systemd/system`. Názvovou konvenciou pre systemd služby je meno jednotky (služby) v nasledujúcom tvare:

```
nazov_sluzby.service
```

V unit súbore je možné definovať rôzne premenné popisujúce závislosti služby (napríklad iné služby), typ služby alebo jej popis (anglicky description). Dôležitou možnosťou je premenná `Restart`, ktorá popisuje čo sa má stať ak sa proces služby ukončí, je zabitý a podobne. Takto je zaručené že aj pri zlyhaní alebo chybe sa systemd postará o jeho znovu-spustenie a beh bez zásahu užívateľa. Pre hlavný riadiaci skript bol využitý aj takzvaný **Watchdog** časovač ktorý sa stará o detekciu zacyklenia a prípadne potrebný reštart. Démon svoje korektné fungovanie ohlasuje periodicky správou `WATCHDOG=1`. Pokiaľ uplynie špecifikovaná doba (`WatchdogSec`) a počas nej Watchdog neobdržal žiadnu správu, je démon (služba) reštartovaný. Zvonček využíva štyri systémové služby.

- službu *gotify.service* pre server Gotify (Sekcia 4.2.3)
- službu *picam.service* pre program **picam** (Sekcia 4.2.1)
- službu *doorbell.service* pre **hlavný riadiaci skript** (Sekcia 4.3)

Webový server **Nginx** tiež využíva typ služba, ten je však vytvorený automaticky pri inštalácii a prvotnej konfigurácii (Sekcia 4.2.2). Vždy po vytvorení novej služby je potrebné použiť nasledujúce príkazy, ktoré nanovo načítajú závislosti a povolia beh služby:

```
systemctl daemon-reload
systemctl enable nazov_sluzby.service
```

Viac o systémových službách, ich konfigurácii a správe je možné nájsť napríklad tu [16].

[43] anglicky daemon
[44] Jednotka typu služba
[45] Jednotka starajúca sa o pripojenie súborového systému
[46] Jednotka typu cieľ

Kapitola 4

Implementácia

Jadro celej implementácie je tvorené jednotlivými systémovými službami a hlavným riadiacim skriptom `Doorbell.py`. V tejto kapitole popíšem implementáciu hlavného riadiaceho skriptu a vytvorených systémových služieb potrebných pre beh zvončeku.

4.1 Pripojenie periférií

Zvonček má viaceré periférie, ktoré sú pripojené rozličným spôsobom. Napríklad tlačidlo a PIR senzor sú pripojené ku GPIO zbernici Raspberry Pi. Prehľad pinov a ich význam viz tabuľka 4.1. Pri číslovaní je použité číslovanie podľa fyzického rozloženia pinov. Toto je v zdrojovom kóde nastavené nasledujúcim príkazom:

```
GPIO.setmode(GPIO.BOARD)
```

Taktiež je možné použiť číslovanie podľa výstupov pinov z procesora (`GPIO.BCM`[1]).

Tabuľka 4.1: Prehľad využitých GPIO pinov a ich význam

Komponent	*Číslo pinu a význam*		
Tlačidlo	PIN 1 – 3.3V	PIN 10 – OUT (GPIO.IN)	
PIR senzor	PIN 2 – 5V	PIN 7 – OUT (GPIO.IN)	PIN 6 – GND

Kamerový modul je pripojený pomocou FFC konektoru do tzv. *CSI*[2]. Ten je pred použitím potrebné povoliť v nastaveniach pomocou CLI aplikácie `raspi-config`. Mikrofón je k Raspberry Pi pripojený pomocou USB 2.0 portu.

4.2 Systémové služby a konfigurácia

Implementácia zvončeka zahŕňa tri resp. štyri systémové služby. Systémová služba pre obsluhu procesu hlavného riadiaceho skriptu sa nazýva `doorbell.service`. Podrobnejšie sa o tomto skripte píše v Sekcii 4.3. Služba `nginx.service` je vytvorená automaticky pri inštalácii, zvyšok je nutné vytvoriť manuálne. Jednotlivé unit súbory sú dostupné v repozitári spolu so zdrojovými kódmi[3].

[1] Broadcom SOC channel
[2] Camera Serial Interface
[3] https://github.com/yogiovic/pi_doorbell

4.2.1 picam.service

Služba *picam.service* spúšta a zabezpečuje beh programu **picam** s nasledujúcimi parametrami:

```
/cesta/ku/picam/picam --tcpout tcp://127.0.0.1:8181 --alsadev hw:1,0
```

Kde:

- `/cesta/ku/picam/`*picam* - je cesta ku binárnemu spustiteľnému súboru *picam* a je potrebné ju nakonfigurovať v unit súbore pre službu
- `-tcpout tcp://127.0.0.1:8181` - špecifikuje výstupnú sieťovú IP adresu, na ktorú sa budú dáta odosielať pomocou protokolu TCP
- `-alsadev hw:1,0` - špecifikuje z ktorého audio vstupu má byť nahrávaná zvuková stopa

4.2.2 nginx.service

Táto služba sa stará o chod webového serveru Nginx. Ten bolo po pridaní RTMP modulu `nginx-rtmp-module` nutné pred spustením nakonfigurovať. Konfigurácia je možná upravením konfiguračného súboru `nginx.conf` v adresári `/etc/nginx/`.

```
rtmp {
    server {
        listen 1935;
        chunk_size 4000;
        application doorbell {

            live on;

            exec_static /cesta/ku/ffmpeg -i tcp://127.0.0.1:8181?listen
            -c:v copy -ar 44100 -ab 40000
            -f flv rtmp://localhost:1935/doorbell/;
        }
    }
}
```

Obr. 4.1: Konfigurácia Nginx ako RTMP serveru[4].

Port číslo 1935 je štandardným RTMP portom[5],[6]. Okrem tohto nastavenia stojí za zmienku príkaz `exec_static`, ktorým server spúšta CLI nástroj `ffmpeg` (Sekcia 3.4.5).

[5] https://helpx.adobe.com/adobe-media-server/kb/ports-firewalls-flash-media-server.html
[6] https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=34

Nástroj ffmpeg webový server Nginx spúšta s nasledujúcimi parametrami, ktoré boli prevzaté z [13]:

- `-i tcp://127.0.0.1:8181?listen` - vstup z programu **picam** 4.2.1
- `-c:v copy` – kodek videa, ktorý má byť zachovaný identicky (copy) podľa kodeku vstupu
- `-ar 44100` – vzorkovacia frekvencia[7] v Hz
- `-ab 40000` – bit rate v kbit/s
- `-f flv rtmp://localhost:1935/doorbell/mystream;` – špecifikácia výstupu, na tejto adrese je stream dostupný v sieti

Viac informácií je možné nájsť v dokumentácií na oficiálnej stránke nástroja **ffmpeg**[3].

4.2.3 gotify.service

Systémová služba serveru Gotify. Gotify poskytuje k stiahnutiu spustiteľný binárny súbor `gotify-linux-arm-7`. Konfigurácia Gotify serveru je možná v súbore `config.yml` v adresári `/etc/gotify/`. V súbore je nutné vo vetve `stream: allowedorigins:` odkomentovať nasledujúci riadok:

```
# - "" # activate this for accessing from an electron app
```

Takto povolíme prístup ku Gotify aj desktopovej aplikácií tak, ako to naznačuje komentár v anglickom jazyku [4].

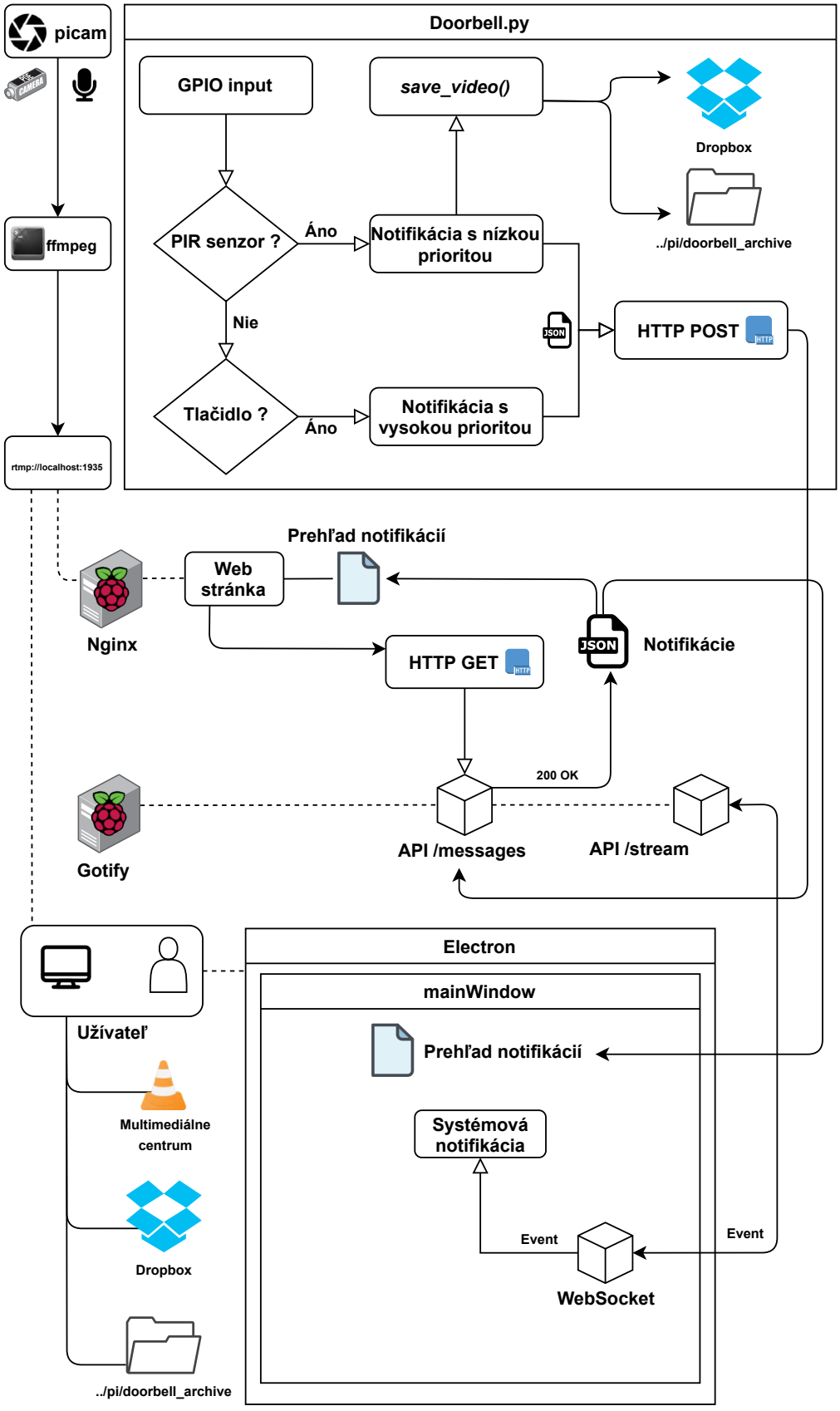
4.3 Hlavný riadiaci skript

Hlavný riadiaci skript `Doorbell.py` má za úlohu odchytávať udalosti na jednotlivých vstupoch GPIO zbernice na Raspberry Pi a spracovať ich. Za udalosť sa považuje detekcia pohybu pasívnym infračerveným čidlom alebo stlačenie tlačidla. Zdrojový kód obsahuje triedu `Doorbell`, ktorá zapuzdruje všetky atribúty a metódy potrebné pre chod a konfiguráciu zvončeka. Konštruktoru triedy sú pri vytvorení novej inštancie predané minimálne nasledujúce parametre:

- `self` - inštancia triedy
- `gotify_token` - prístupový token pre Gotify
- `dropbox_token` - prístupový token pre Dropbox
- `name` - názov zvončeku

Takto je možné lokálne vytvárať viacero zvončekov s príslušnými tokenmi a vlastným názvom. Vlastný názov zvončeku sa následne zobrazuje aj v prehľade notifikácií. Užívateľ, ktorý by sa rozhodol použiť viacero zvončekov zároveň, ich tak dokáže ľahko odlíšiť. Po zachytení udalosti je na nastavenú (prípadne základnú prednastavenú) dobu zablokovaná detekcia ďalšej udalosti. Takto predídeme nechceným notifikáciám v prípadoch, kedy napríklad pred zvončekom poletuje vták, ktorý opakovane vyvoláva vstup z PIR modulu.

[7] Počet meraní za určitú časovú jednotku – 44100 meraní za sekundu



Obr. 4.2: Diagram systému zvončeka.

Trieda `Doorbell` obsahuje následovné konfigurovateľné atribúty, ktoré špecifikujú túto dobu (v sekundách):

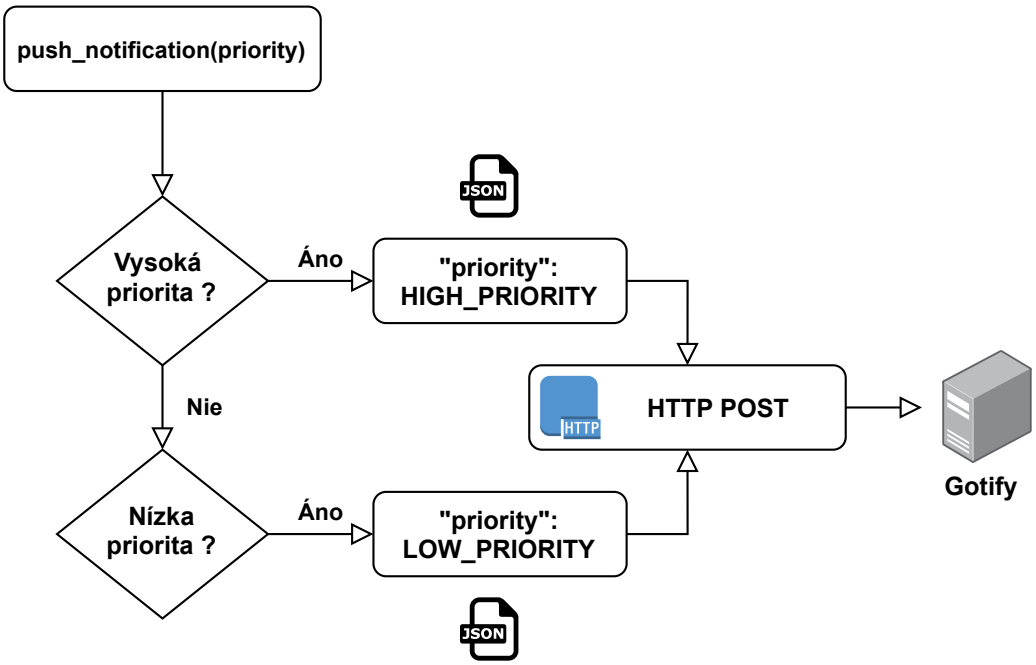
- `button_timeout` - doba, po ktorej uplynutí je znovu povolený vstup z tlačidla
- `pir_timeout` - doba, po ktorej uplynutí je znovu povolený vstup z pasívneho infračerveného senzoru

Pre obidva tieto atribúty sú implementované *getter* a *setter* metódy umožňujúce konfiguráciu viz Obrázok 4.3.

```
@property
def button_timeout(self):
    return self._button_timeout

@button_timeout.setter
def button_timeout(self, value):
    if value < 0.1:
        raise ValueError("Timeout below 0.1 is not allowed!")
    self._button_timeout = value
```

Obr. 4.3: Využitie dekorátorov v zdrojovom kóde pre getter a setter.

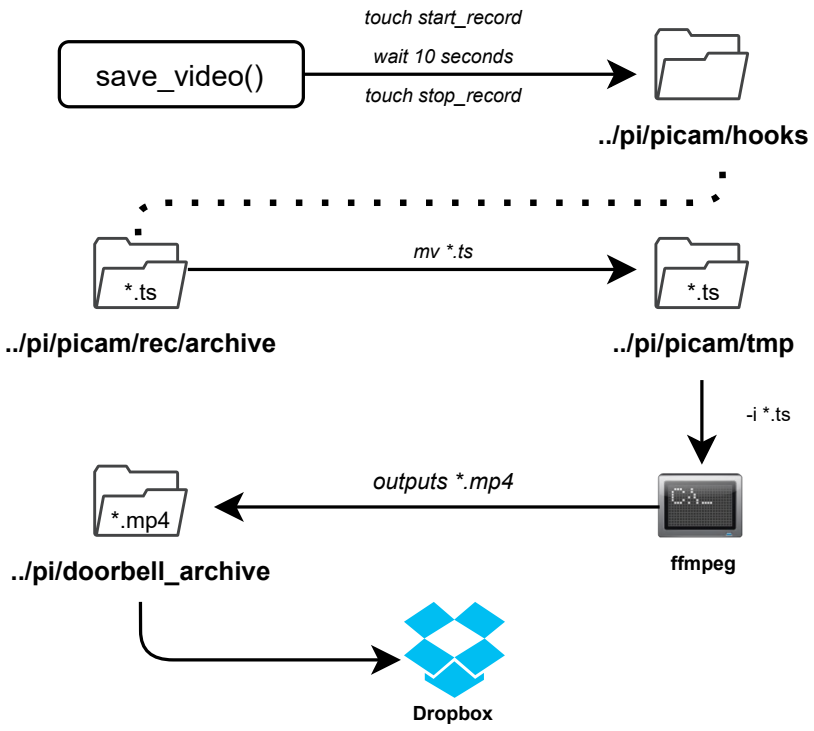


Obr. 4.4: Vývojový diagram vytvorenia notifikácie metódou `push_notification()`.

Po odchytení udalosti je zavolaná príslušná metóda na jej spracovanie. Jedná sa o metódy `handle_pir_event()` a `handle_button_event()`. Príslušná metóda zablokuje ďalšiu detekciu na nastavenú dobu a zavolá metódu
`push_notification(self, priority)`, ktorej parameter `priority` je zvolený na základe udalosti, ktorá vyvolala vytvorenie notifikácie. Notifikácia o detekcii pohybu má nižšiu prioritu ako notifikácia, ktorá oznamuje stlačenie tlačidla. Tieto hodnoty sú vopred definované v konštantách.

4.3.1 Ukladanie záznamov

Pri detekcii pohybu je vytvorené krátke 10 sekundové video. Toto video je uložené lokálne a ak je dostupné internetové pripojenie, je zálohované v cloudovej službe Dropbox. Nahranie videa do Dropboxu prebieha skrz oficiálne poskytované REST API. Program picam umožňuje spustenie nahrávania a uloženie nahrávky aj z práve prebiehajúceho streamu. Pre spustenie nahrávania stačí v adresári programu picam pomocou príkazu `touch` vytvoriť súbor `/hooks/start_record`. Na rovnakom princípe taktiež funguje aj zastavenie nahrávania – `touch /hooks/stop_record`.



Obr. 4.5: Diagram popisujúci vytvorenie nahrávky, prekonvertovanie nástrojom ffmpeg a zálohovanie metódou `save_video()`.

Výsledná nahrávka je premiestnená do pracovného adresára kde sa pomocou nástroja ffmpeg prevedie do súborového formátu MP4. Odtiaľ je následne presunutá do adresára archívu a prípadne zálohovaná v službe Dropbox.

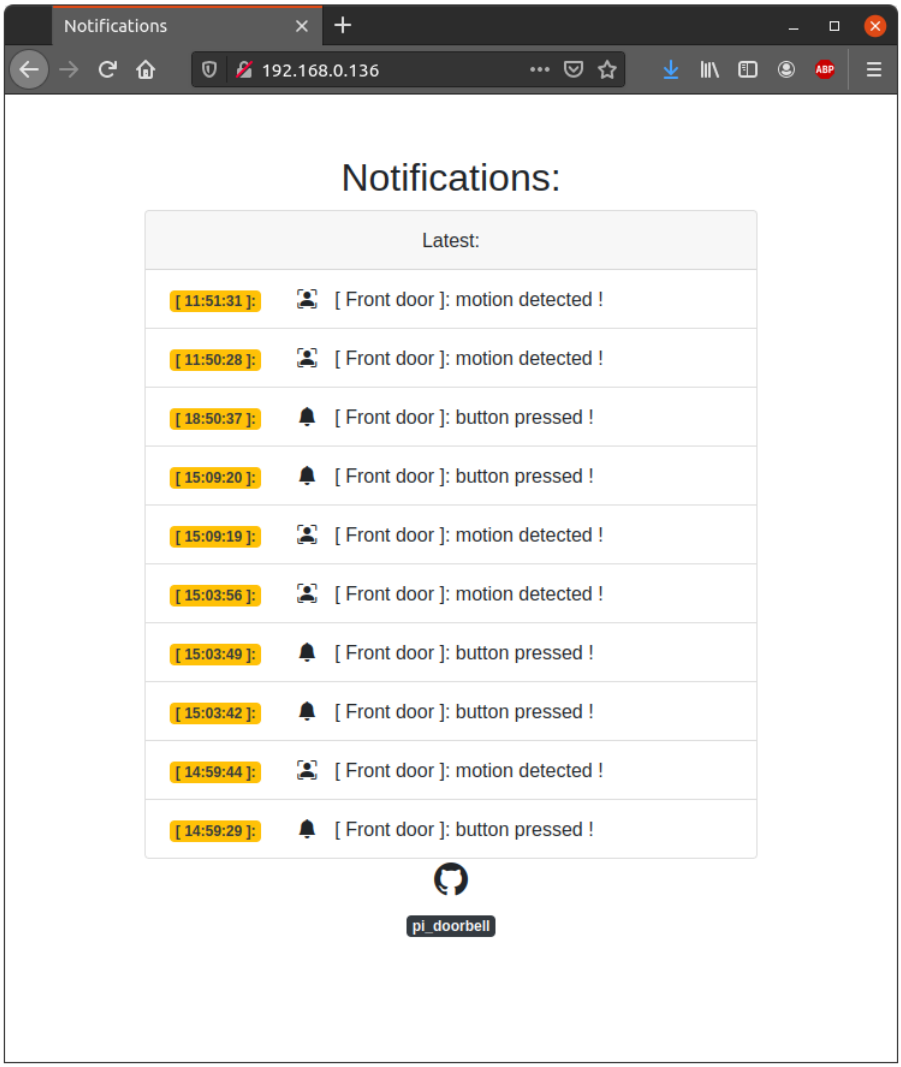
4.4 Notifikácie

Notifikácie sú dôležitou súčasťou zvončeka. Je možné ich zobraziť v prehľade na webovej stránke, v desktopovej aplikácii či pomocou oficiálnej Gotify aplikácie.

4.4.1 Webová stránka

Zvonček disponuje jednoduchou webovou stránkou a desktopovou aplikáciou s prehľadom desať najnovších notifikácii. V tejto sekcii je popísaný implementovaný webový prehľad a desktopová aplikácia, formát a zobrazenie notifikácií. Pri každej notifikácii sa nachádza:

- čas jej vytvorenia
- ikona
- názov zvončeku a krátka správa



Obr. 4.6: Webový prehľad notifikácií v prehliadači Firefox.

Podľa ikony je možné vidieť, či sa jedná o notifikáciu, ktorá bola vyvolaná detekciou pohybu alebo stlačením tlačidla zvončeka. Názov zvončeka je uvedený v hranatých zátvorkách a slúži k rozlíšeniu jednotlivých zariadení pokiaľ by užívateľ využíval viacero zvončekov súčasne. Za názvom nasleduje krátka správa v anglickom jazyku, ktorá spolu s ikonou popisuje o aký typ notifikácie sa jedná. Stránka na spodku tiež obsahuje odkaz na verejný repozitár so zdrojovými kódmi[8].

4.4.2 Aktualizačný skript

K webovej stránke je vytvorený krátky skript v jazyku JavaScript, ktorý sa stará o dynamickú aktualizáciu obsahu webovej stránky. Skript každých 5 sekúnd odošle HTTP GET požiadavku na `/message` endpoint serveru Gotify a obdrží odpoveď v podobe reťazca vo formáte JSON (Sekcia 3.3.2). Prehľad notifikácií je tvorený z následujúcich Bootstrap komponentov:

- `list-group` - prvok `<ul>` - zoznam všetkých notifikácií
- `list-group-item` - prvok `<li>` - samotná notifikácia

Skript po obdržaní odpovede reťazec rozparsuje, odstráni z prehľadu staré notifikácie a uloží si dáta o najnovších notifikáciách. Následne z dát vytvorí nové prvky pre jednotlivé notifikácie a pridá ich do prehľadu. Stránku teda nieje nutné opakovane načítať ale jej obsah sa mení dynamicky.

```
function getNotifications() {

    var xmlhttp = new XMLHttpRequest();
    var url = "http://[PI_ADDRESS]:[GOTIFY_PORT]/message?token=[TOKEN]";

    xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            var myArr = JSON.parse(this.responseText);
            parseResponse(myArr);
        }
    };

    xmlhttp.open("GET", url, true);
    xmlhttp.send();
    setTimeout(getNotifications, 5000);
}
```

Obr. 4.7: Príklad asynchrónneho serverového volania.

[8] https://github.com/yogiovic/pi_doorbell

4.4.3 Desktopová aplikácia

K zvončeku je vytvorená aj jednoduchá desktopová aplikácia vo frameworku Electron (Sekcia 3.3.9). Táto aplikácia využíva webovú stránku popísanú v predchádzajúcej kapitole. Navyše však ešte implementuje funkcionalitu systémových notifikácií.

4.4.4 Systémové notifikácie

Desktopová aplikácia po obdržaní informácie o vytvorení novej notifikácie vytvorí systémovú notifikáciu viz Obrázok 4.8.

```
const addr = 'ws://[PI_ADDRESS]:[GOTIFY_PORT]/stream?token=[TOKEN]'
const socket = new WebSocket(addr);

socket.addEventListener('message', function (event) {

    let notification = JSON.parse(event.data);
    let title = notification['title'];
    let time = notification['date'].split('T')[1].split('.')[0];
    let timestamp = "[ " + time + " ]" + ": ";
    let content = notification['message'];

    const myNotification = new Notification(title, {
        body: timestamp + content,
        requireInteraction: true
    });
});
```

Obr. 4.8: Príklad vytvorenia WebSocket a systémovej notifikácie v Electron aplikácii.

K upozorňovaniu na vznik novej udalosti (vytvorenie novej notifikácie) využíva aplikácia protokol `WebSocket` a Gotify REST API endpoint `/stream`. Pri vzniku obrdží reťazec vo formáte JSON, ktorý následne rozparsuje na jednotlivé časti ktoré budú tvoriť nadpis a telo správy notifikácie. Premenná `requireInteraction` definuje či má byť notifikácia schovaná až po kliknutí užívateľom.

4.4.5 Android notifikácie

Gotify poskytuje pre platformu Android taktiež oficiálnu aplikáciu[9]. Užívateľovi so zariadením, ktoré podporuje jej inštaláciu umožňuje využiť funkcionalitu notifikácií tohto zariadenia (napríklad mobilný telefón alebo tablet). Ako každý klient aj aplikácia potrebuje pre svoj beh používať vlastný vygenerovaný *clientToken* (Príloha B.4). Nevýhodou je absencia podpory iných platforiem, ktorú však rieši možnosť otvoriť si webový prehľad notifikácií. Android notifikácie (screenshoty) zo zvončeka je možné nájsť v prílohe B.2.

[9] https://play.google.com/store/apps/details?id=com.github.gotify

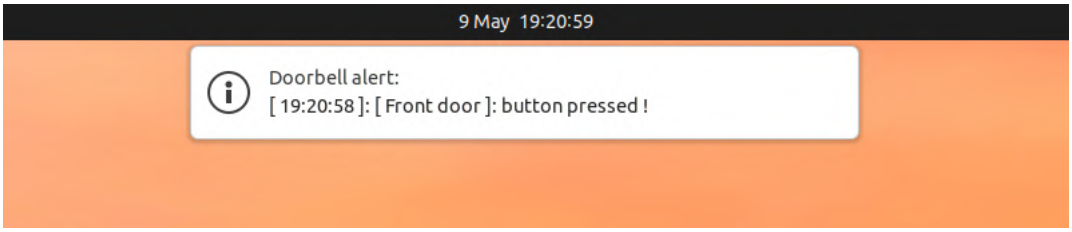
Kapitola 5

Testovanie

Zvonček (Raspberry Pi a periférie) bol pre účely testovania pripevnený k zárubni dverí viz Príloha A.1. Kamera bola umiestnená do výšky približne 180 centimetrov, ktorá sa aj pri rôznych výškach osôb ukázala ako vhodná. Zvonček bol následne používaný mojimi spolubývajúcimi (ďalej ako užívatelia), ktorým ďakujem za ich ochotu sa na testovaní podieľať. Otestované bolo nasledujúce:

- **Detekcia pohybu**
- **Zvonenie**
- **Zobrazovanie notifikácií**
- **Zahltenie zvončeka**
- **Streaming a ukladanie záznamov**

Interakcia so zvončekom je veľmi jednoduchá a intuitívna – jedná sa len o stlačenie tlačidla. Zobrazovanie prehľadu notifikácií bolo pri spätnej väzbe od užívateľov zhodnotené ako jednoduché a praktické. Trochu obtiažnejšia bola pre neskúseného užívateľa synchronizácia mobilnej aplikácie Gotify so zvončekom, nakoľko si vyžaduje vygenerovanie a zadanie prístupového tokenu. Aplikácia následne zobrazovala notifikácie priamo na mobilnom zariadení viz Príloha B.2. Desktopová aplikácia a zobrazovanie systémových notifikácií taktiež prebiehalo korektne, príklad systémovej notifikácie viz Obrázok 5.1.



Obr. 5.1: Screenshot systémovej notifkácie z desktopovej aplikácia zvončeka.

Pokus o zahltenie zvončeka bol neúspešný, zvonček po nakonfigurovanú dobu až do jej uplynutia ďalšie vstupy z periférií nezachytil.

OSMC

Zadanie ako príklad multimediálneho centra na ktoré bude zvonček streamovať obraz uvádza OSMC. **OSMC** je bezplatný a open-source operačný systém postavený na linuxovej distribúcii **Debian**[1] a projekte **Kodi**[2], určený pre platformy *Raspberry Pi*, *Apple TV* a *Vero*. OSMC primárne používa prehrávač médií Kodi ktorý bol použitý aj pri testovaní zvončeka. Pre zobrazenie streamu v Kodi je nutné vytvoriť textový súbor s príponou `.strm` ktorého obsahom je adresa streamu – `rtmp://[RASPBERRY_PI`[3]`]/doorbell/cam`. Tento súbor je k služí k prístupu ku streamu v multimediálnej knižnici prehrávača Kodi. Streaming fungoval korektne a kvalita zvukovej stopy z USB mikrofónu bola prijateľná. Značne lepšia kvalita zvuku však bola dosiahnutá pri použití USB web kamery ako mikrofónu. Ukladanie záznamov fungovalo podľa očakávaní – záznam sa vždy uložil lokálne do archívneho adresára a pokiaľ bolo dostupné internetové pripojenie bol zálohovaný aj v službe Dropbox. V službe Dropbox som si taktiež vyskúšal možnosť automatického mazania záznamov každých sedem dní. Pokiaľ zvonček nemal dostupné internetové pripojenie zálohovanie záznamu bolo neuspešné, no na chod zvončeka nemalo vplyv.

[0] Open Source Media Center – Open Source Mediálne Centrum
[1] https://www.debian.org/
[2] https://kodi.tv/
[3] IP adresa Raspberry Pi v sieti užívateľa

Kapitola 6

Možné rozšírenia

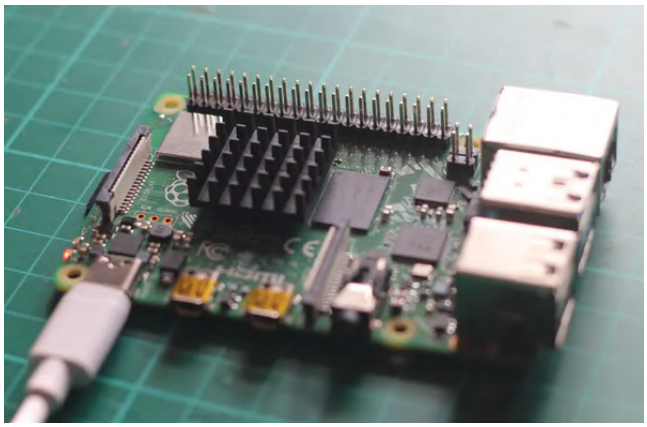
V tejto kapitole popíšem možné vhodné rozšírenia, ktoré môžu byť implementované v budúcnosti alebo v prípade nadviazania na prácu.

6.1 Rozšírenia zvončeka

Táto sekcia obsahuje popis možných rozšírení ktoré sú špecifické pre zvonček ako taký.

6.1.1 Chladenie Raspberry Pi

Raspberry Pi sa pri svojej prevádzke značne zahrieva. Maximálna teplota prevádzky je špecifikovaná ako 65°C (tzv. *soft limit*) respektíve 85°C (tzv. *hard limit*) [7]. Jednou z možností zníženia teploty Raspberry Pi je napríklad pridanie pasívneho chladiča, viz Obrázok 6.1.



Obr. 6.1: Raspberry Pi s pasívnym chladičom na procesore, prevzaté z [11].

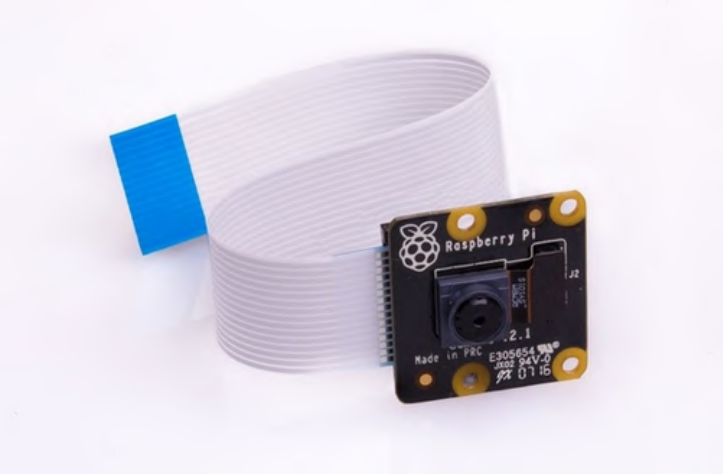
Ďalšou z možností je aktívne chladenie. Príklad aktívneho chladiča (s ventilátorom) je možné nájsť v sekcii 7.3.1. Aktívny chladič môže byť buď pripojený priamo k doske alebo je možné zakúpiť a použiť obal so vstavaným ventilátorom. Keďže však Raspberry Pi bolo zapožičané fakultou a nie moje vlastné, ponechávam pridanie chladiča ako možné rozšírenie.

6.1.2 Videokonferencia namiesto streamingu

Zadanie špecifikuje, že streaming by mal prebiehať na multimediálne centrum. Streaming na multimediálne centrum, ktorý by podporoval dvojsmernú komunikáciu (teda aj možnosť odpovedať človeku pred zvončekom), však prináša niekoľko implementačných problémov. Za hlavný považujem nutnosť ďalšieho zariadenia, ktoré bude prenášať zvuk (prípadne aj obraz) spať na zvonček a nutnosť pridania reproduktoru, čo navyšuje celkovú cenu riešenia. Možným rozšírením je však namiesto streamingu na multimediálne centrum, tak ako to upresňuje zadanie, využiť videokonferenčné služby, ako napríklad Google Duo alebo Jitsi. Pri stlačení tlačidla by tak došlo k spusteniu videohovoru medzi zvončekom respektíve človekom pred ním a užívateľom. Za nevýhodu tohto rozšírenia považujem fakt, že nie je lokálne a uberá tak zvončeku jeho hlavnú výhodu, ktorá spočíva práve v plne lokálnom fungovaní.

6.1.3 PiNoIR kamera

Zvonček využíva širokouhlý kamerový modul (Sekcia 3.2.1). Táto kamera sa však, ako každý modul, dá vymeniť za inú alebo lepšiu alternatívu. Takouto alternatívou môže byť napríklad kamera **PiNoIR** od oficiálneho výrobcu Raspberry Pi. Táto kamera sa vyznačuje absenciou filtru infračerveného žiarenia (NoIR - No Infrared) a je tak lepšie využiteľná pri zlých svetelných podmienkach. Pokiaľ užívateľ predpokladá, že zvonček bude umiestnený v časti, kde je počas celého dňa zlá viditeľnosť, je táto kamera určite vhodnejšia. Pre použitie v klasických svetelných podmienkach je však kvôli absencii infračerveného filtra skôr nevýhodou.



Obr. 6.2: Kamera PiNoIR, prevzaté[1]

6.1.4 Odomykanie a uzamykanie dverí pomocou serva

Ďalším možným rozšírením zvončeka a riadiaceho skriptu je doplnenie funkcionality ovládania mechanického serva, ktoré môže slúžiť napríklad k odomykaniu alebo uzamykaniu dverí. Toto môže prebiehať buď na základe užívateľského vstupu, ako napríklad stlačením tlačidla vo webovom rozhraní, alebo automaticky na základe detekcie tvári (Sekcia 6.1.5).

[1] https://www.raspberrypi.org/products/pi-noir-camera-v2/

6.1.5 Rozpoznávanie tváre

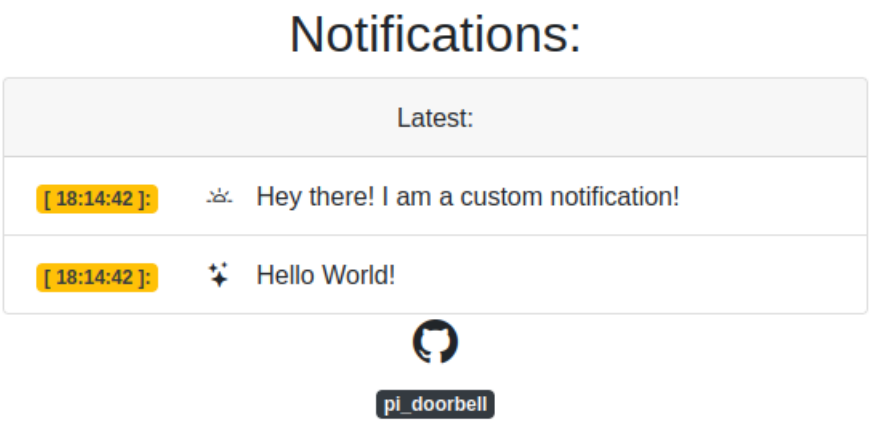
Existuje viacero bezplatných open-source knižníc zameriavajúcich sa na počítačové videnie, ako napríklad **OpenCV**. Na internete je dostupných viacero riešení pre Raspberry Pi, ktoré dokážu pomocou rôznych algoritmov rozoznávať tváre a identifikovať ich. Dobrým námetom na rozšírenie je prepojenie rozpoznávania tvárí s mechanickým servom tak, aby mohol zvonček automaticky odomykať dvere.

6.2 Rozšírenia softvéru

Rozšírenia v tejto sekcii sú všeobecného charakteru a nesúvisia priamo so zvončekom ako takým.

6.2.1 Electron Gotify klient

Electron aplikácia je vo svojej implementácii špecifická pre účely použitia so zvončekom. Nakoľko však spracováva odpoveď z Gotify REST API endopointu obecne, bolo by možné ju po úprave častí implementácie špecifických pre zvonček (ikony), použiť aj pre zobrazovanie prehľadu iných zariadení. Toto bolo aj otestované, viz Obrázok 6.3. Nakoľko však toto nepovažujem za priamo súvisiace so zadaním a implementáciou zvončeka, ponechávam to iba ako možné rozšírenie/iné využitie vytvoreného softvéru.



Obr. 6.3: Vytvorenie demonštračnej generickej notifikácie.

Kapitola 7

Nedostatky riešenia

Táto kapitola popisuje nedostatky zvončeka a ich možné riešenie buď v podobe rozšírenia alebo úpravy spôsobu použitia.

7.1 Odozva

Jedným z nedostatkov implementácie streamingu je odozva. Odozva však netvorí problém nakoľko komunikácia prebieha iba smerom od zvončeku k užívateľovi a nie naopak. Táto odozva je spôsobená nutnosťou načítania niekoľkých častí (blokov) streamu pred spustením jeho prehrávania – tzv. *buffering*.

7.2 Absencia podpory duplexnej komunikácie

Implementácia zvončeka podporuje pri komunikácii iba prenos od obrazu a zvuku od zvončeka k užívateľovi. Pri moderných inteligentných zvončekoch je podpora duplexnej komunikácie žiadúca a vhodná, avšak v prípade zvončeka, ktorý streamuje na multimediálne centrum a nie do mobilnej aplikácie, so sebou prináša niekoľko implementačných problémov. Obojsmerná komunikácia pri streamingu na multimediálnom centre vyžaduje zariadenie, ktoré bude zachytávať a streamovať užívateľa naspať na zvonček. Zvonček musí byť následne rozšírený o reproduktor či prípadne displej.

7.2.1 Riešenie

Riešenie obidvoch problémov by mohlo spočívať napríklad v podpore duplexnej komunikácie pomocou videokonferenčnej aplikácie ako Google Duo alebo Jitsi (Sekcia 6.1.2), kde prenos dát prebieha inak. Zadanie však špecifikuje streaming na multimediálne centrum, ktorého odovzvu sa mi podarilo znížiť na dobu troch sekúnd. Odozva podľa môjho názoru nie je zlá, no bola pri testovaní badateľná.

7.3 Napájanie adaptérom

Raspberry Pi resp. zvonček je napájaný Micro-USB adaptérom. Toto by pri použití v praxi mohlo predstavovať problém, napríklad z dôvodu nedostupnosti zástrčky či rôznych bezpečnostných dôvodov.

7.3.1 Riešenie

Raspberry Pi 3B+ je možné rozšíriť o prídavnú dosku **PoE HAT**, ktorá mu umožňuje využiť technológiu *PoE*[1]. Prídavná doska (viz Obrázok 7.1) taktiež obsahuje aktívny chladič procesora. Takto je možné pripojiť jedným káblom Raspberry Pi do LAN siete a zároveň ho napájať bez vlastného napájacieho zdroja. Cena takejto prídavnej dosky sa pohybuje okolo 20 eur alebo 500 českých korún [8].

Obr. 7.1: Raspberry Pi s prídavnou doskou PoE HAT, prevzaté z [8].

7.3.2 Power over Ethernet

Power over Ethernet alebo napájanie ethernetom je technológia prvýkrát použitá v roku 2000 firmou **Cisco**. Štandard `IEEE 802.3af` definuje dva typy PoE zariadení.

- *PSE* – Power Sourcing Equipment[2]
- *PD* – Powered Device[3]

PSE riadi množstvo energie, ktoré sa dodáva do PD až do vzdialenosti 100 metrov. PD je v tomto prípade Raspberry Pi. Sieť, v ktorej chceme PoE použiť, musí PoE podporovať. Tu je potrebné rozlíšiť dva pojmy:

- *Endspan* – napríklad switch[4], ktorý už v sebe má zabudovaný hardvér s podporou PoE, je základným stavebným prvkom PoE sietí
- *Midspan* – tzv. injektor, umiestnený medzi switchom a PD. Midspan sa typicky umiestňuje v blízkosti switchu a slúži ako PSE

Pokiaľ máme sieť už vytvorenú bez podpory PoE, je vhodné použiť injektor, ktorý bude slúžiť ako PSE. Endspan môže napájať ako dátové vodiče, tak aj volné páry vodičov. Midspan napája iba volný pár vodičov [22].

[1] Power over Ethernet – napájanie ethernetom
[2] Napájacie zariadenie
[3] Napájané zariadenie
[4] Prepínač

Kapitola 8

Záver

Cieľom tejto práce bolo navrhnúť a implementovať riešenie inteligentného zvončeka založenom na Raspberry Pi. Toto bolo splnené – zvonček funguje a po pridaní krytu a výbere vhodného spôsobu napájania môže byť nasadený do reálneho prostredia. Dôležitým faktorom bol výber použitých technológii. Boli vybraté technológie, ktoré sú open-source a umožnujú lokálne fungovanie. Celý systém je vytvorený tak, aby bol v budúcnosti ľahko modifikovateľný a jeho časti ľahko vymeniteľné. V práci som popísal problémy, na ktoré som narazil, či rozšírenia, ktoré by mohli byť v prípade nadviazania na túto prácu implementované. Tieto rozšírenia prinášajú novú funkcionalitu alebo poskytujú riešenie nedostatkov zvončeka. Za osobný prínos tejto práce považujem určite aj nové znalosti z oblasti živého prenosu videa či rôznych spôsobov ako sú tieto dáta uchované a prenášané. Zadanie práce bolo taktiež mojím prvým rozsiahlejším hardvérovým projektom, ktorý mi priniesol rozhľad v oblasti vstavaných systémov a inteligentných domácností. Pri práci som sa okrem iného zoznámil aj s webovými technológiami. Pre účely zvončeka bolo nutné nasadiť server, ktorý zastáva kľúčovú rolu v živom prenose dát a taktiež sprístupňuje užívateľovi webovú stránku. Zaujímavé bolo taktiež vytvorenie systémových služieb a pochopenie čo sa v skutočnosti deje pri štarte takéhoto systému – ako ho vytvoriť tak aby fungoval bez zásahu užívateľa aj po prípadnom zlyhaní. Prekvapivým zistením je taktiež fakt, že inteligentný domov nemusí stáť veľa peňazí. Počas analýzy rôznych riešení a návodov som na internete narazil aj na iné riešenia, nielen zvončekov, ale aj inteligentných domácich spotrebičov. Tie demonštrujú ako jednoducho pomocou napríklad Raspberry Pi vylepšiť svoju domácnosť a urobiť ju tak modernou a inteligentnou.

Literatúra

[1] *Access Tokens* [online]. Júl 2018 [cit. 2021-29-04]. Dostupné z: https://www.oauth.com/oauth2-servers/access-tokens/.

[2] *Dropbox API v2* [online]. 2021 [cit. 2021-29-04]. Dostupné z: https://www.dropbox.com/developers/documentation/http/documentation.

[3] *Ffmpeg Documentation* [online]. 2021 [cit. 2021-27-04]. Dostupné z: http://ffmpeg.org/ffmpeg.html.

[4] *Gotify Documentation* [online]. 2021 [cit. 2021-27-04]. Dostupné z: https://gotify.net/docs/.

[5] *PIR Camera Case for Raspberry Pi 4* [online]. 2021 [cit. 2021-25-04]. Dostupné z: https://thepihut.com/products/pir-camera-case-for-raspberry-pi-4-3.

[6] *Raspberry Pi 3 Model B+* [online]. 2021 [cit. 2021-20-04]. Dostupné z: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf.

[7] *Raspberry Pi Documentation* [online]. 2021 [cit. 2021-20-04]. Dostupné z: https://www.raspberrypi.org/documentation.

[8] *Raspberry Pi PoE HAT*. 2021 [cit. 2021-30-04]. Dostupné z: https://www.raspberrypi.org/products/poe-hat/.

[9] *The Ring Video Doorbells collection* [online]. 2021 [cit. 2021-25-04]. Dostupné z: https://eu.ring.com/pages/doorbells#compare-chart.

[10] AfterDawn.com. *Container* [online]. 2021 [cit. 2021-28-04]. Dostupné z: https://www.afterdawn.com/glossary/term.cfm/container.

[11] Alasdair, A. *Do You Need to Use a Fan for Cooling with the New Raspberry Pi 4?* 2019 [cit. 2021-30-04]. Dostupné z: https://www.hackster.io/news/do-you-need-to-use-a-fan-for-cooling-with-the-new-raspberry-pi-4-6d523ca12453.

[12] Fair, C. *What Are PIR Sensors and Why Do I Need Them?* [online]. IES Edmonton, 2018 [cit. 2021-24-04]. Dostupné z: https://iesedmonton.org/news/2018-5-22/what-are-pir-sensors-and-why-do-i-need-them.

[13] iizukanao. *Picam*. 2021 [cit. 2021-23-04]. Dostupné z: https://github.com/iizukanao/picam.

[14] Jahoda, B. *5minutový rychlokurs Bootstrapu* [online]. 2016 [cit. 2021-25-04]. Dostupné z: https://jecas.cz/bootstrap-rychlokurs.

[15] Kim, E. *Amazon buys smart doorbell maker Ring for a reported $1 billion* [online]. 2018 [cit. 2021-20-04]. Dostupné z: https://www.cnbc.com/2018/02/27/amazon-buys-ring-the-smart-door-bell-maker-it-backed-through-alexa-fund.html.

[16] Kníže k, J. *Nebojte se systemd: co to je a co umí?* Máj 2016 [cit. 2021-24-04]. Dostupné z: https://www.root.cz/clanky/nebojte-se-systemd-co-to-je-a-co-umi/.

[17] Krings, E. *What is RTMP? Real Time Messaging Protocol - What you Need to Know* [online]. Apríl 2021 [cit. 2021-26-04]. Dostupné z: https://www.dacast.com/blog/rtmp-real-time-messaging-protocol/.

[18] Lord, J. *Essential Electron*. 2021. Dostupné z: https://jlord.us/essential-electron/.

[19] Malý, M. *Protokol MQTT: komunikační standard pro IoT*. Internet Info, s.r.o., Jún 2016 [cit. 2021-01-05]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.

[20] Mark Otto, J. T. *Bootstrap* [online]. 2021 [cit. 2021-25-04]. Dostupné z: https://getbootstrap.com/.

[21] Michalec, L. *PIR detektor: skvělý sluha, ale zlý pán* [online]. 2013 [cit. 2021-24-04]. Dostupné z: https://vyvoj.hw.cz/automatizace/pir-cidlo-skvely-sluha-ale-zly-pan.html.

[22] Michalec, L. *Úvod do Power over Ethernet*. Júl 2020 [cit. 2021-30-04]. Dostupné z: https://automatizace.hw.cz/uvod-do-power-over-ethernet.html.

[23] *Netatmo Doorbell* [online]. 2021 [cit. 2021-20-04]. Dostupné z: https://www.alza.cz/netatmo-doorbell-d5530860.htm.

[24] Owen, A. *Video File Formats, Codecs, and Containers Explained: TechSmith* [online]. Marec 2021 [cit. 2021-28-04]. Dostupné z: https://www.techsmith.com/blog/video-file-formats/.

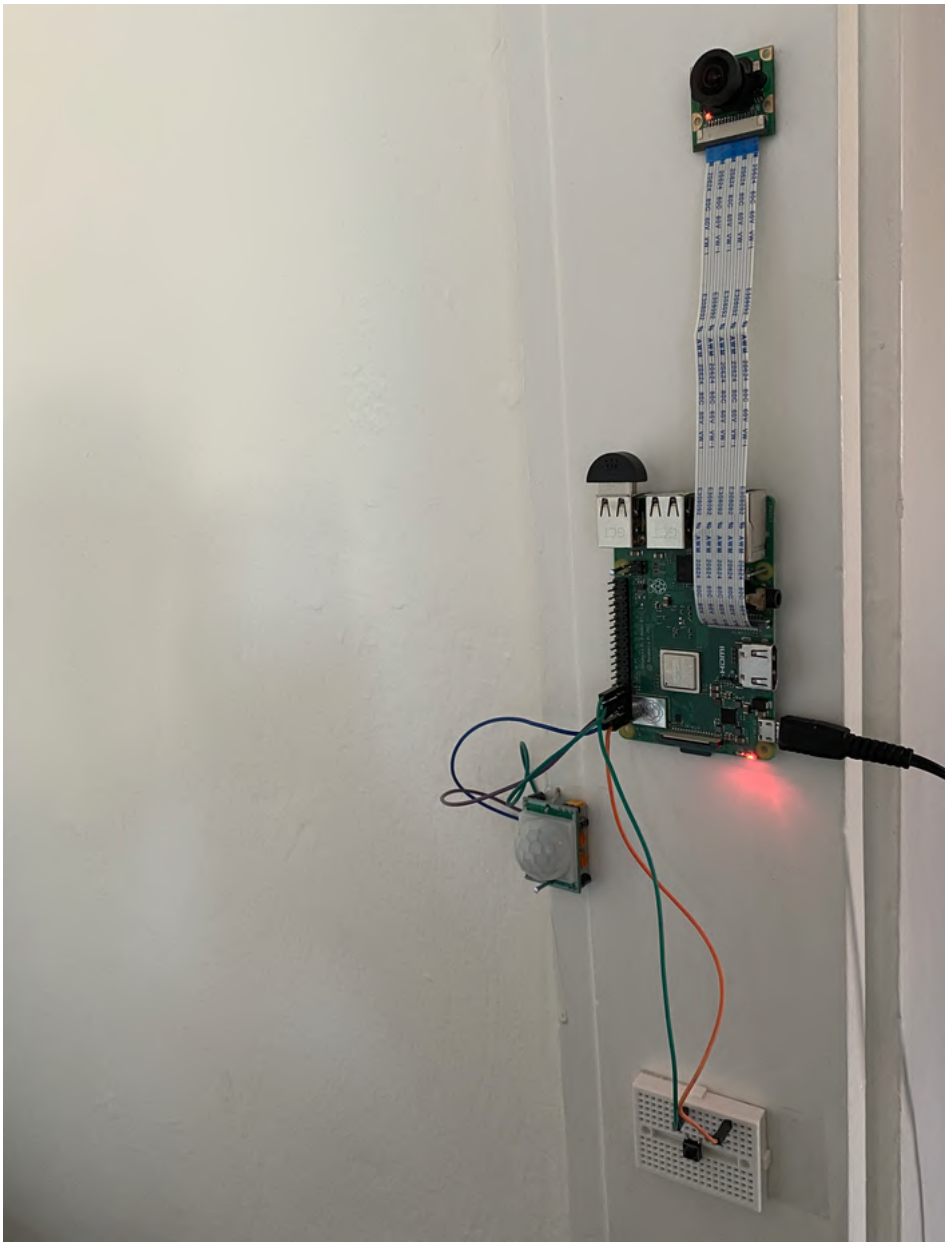
[25] Pelayo, R. *PIR Motion Sensor*. Microcontroller Tutorials, Máj 2019 [cit. 2021-24-04]. Dostupné z: https://www.teachmemicro.com/pir-motion-sensor/.

[26] Radim. *Digitální video - pojmy a software* [online]. December 2011 [cit. 2021-28-04]. Dostupné z: http://www.bitgoo.cz/digitalni-video-pojmy-a-software.php.

[27] Žáček, M. *Digitální video - přehled formátů a kódování* [online]. 2021 [cit. 2021-28-04]. Dostupné z: https://www.aldebaran.cz/onlineskola/etapy/video/formaty.html.

Príloha A

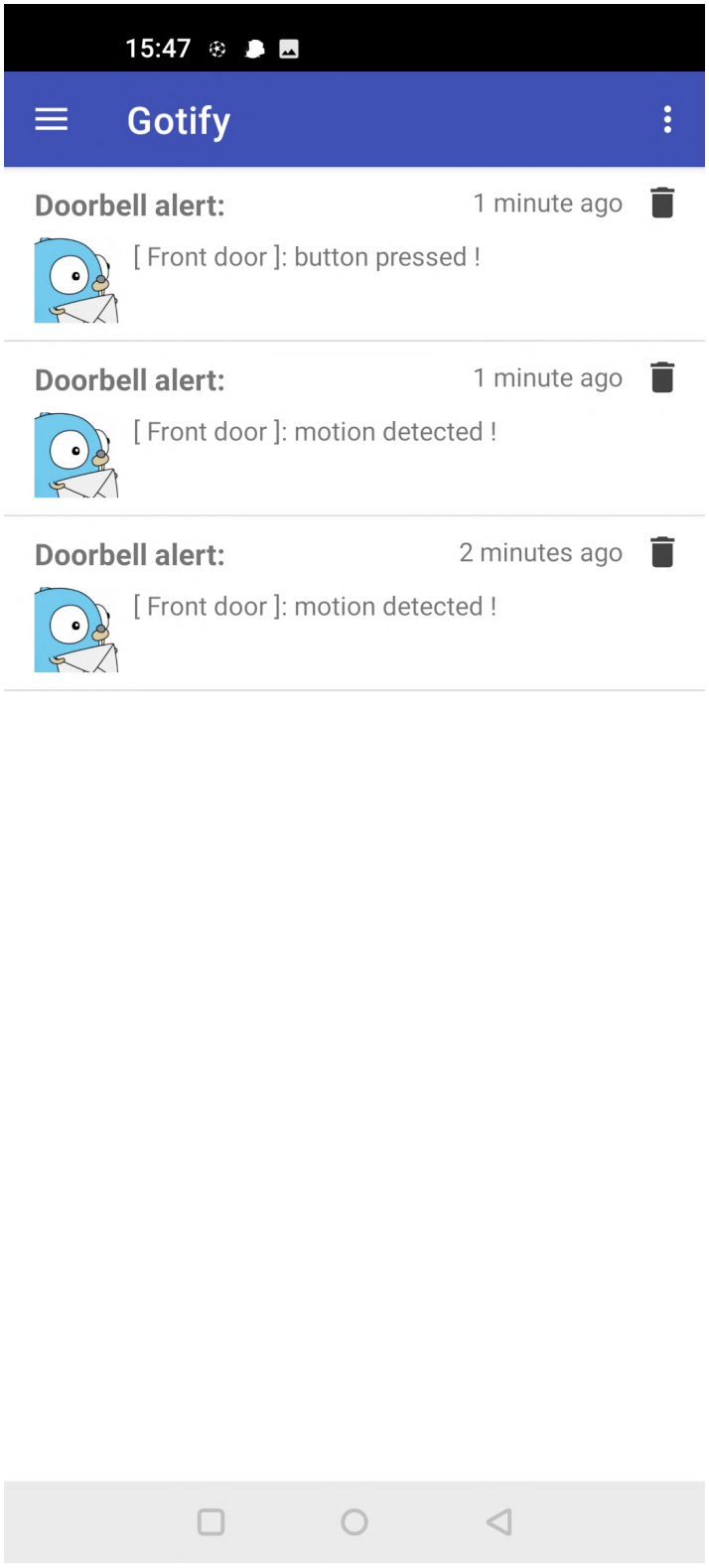
Testovanie



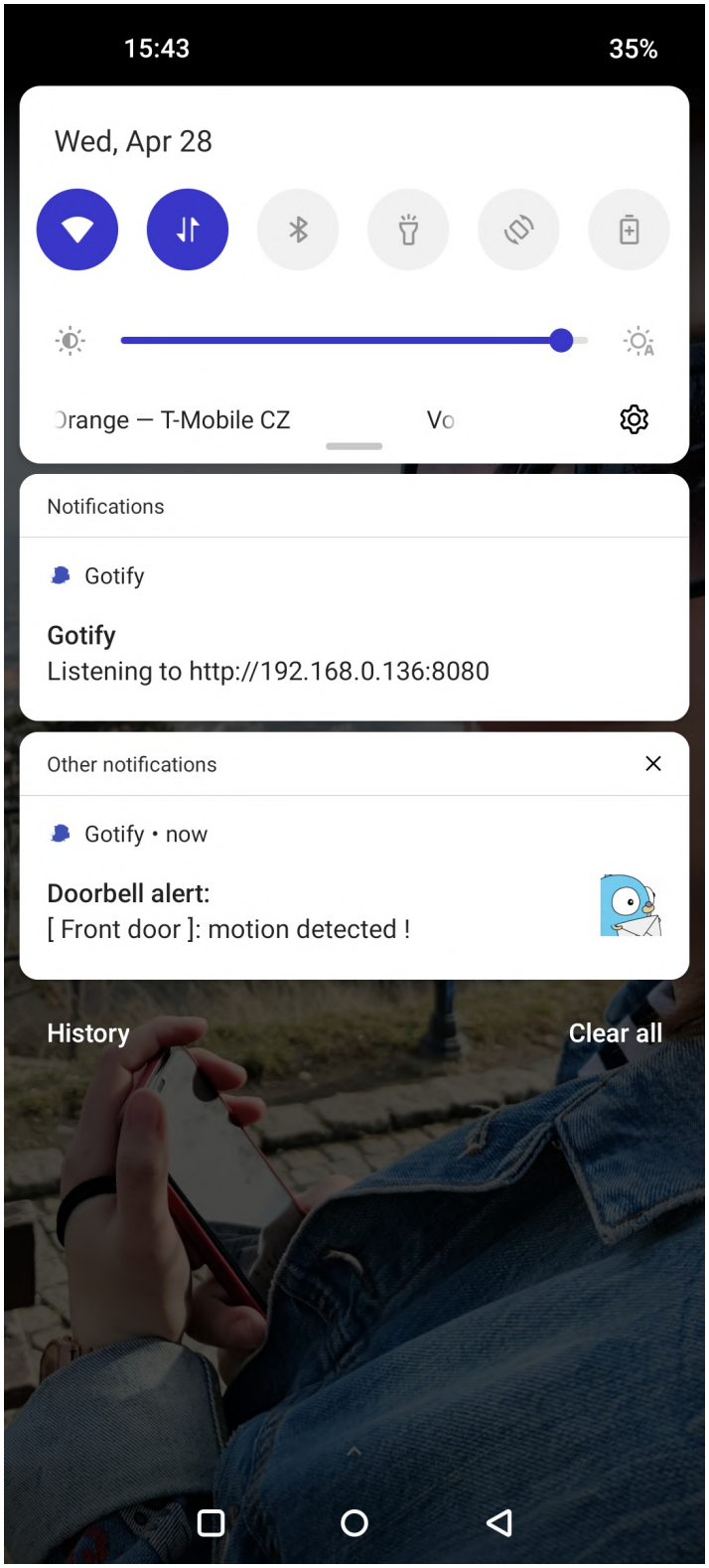
Obr. A.1: Zvonček s perifériami nasadený na zárubňu dverí pre účely testovania.

Príloha B

Gotify aplikácia



Obr. B.1: Rozhranie Gotify aplikácie



Obr. B.2: Push notifikácia z Gotify aplikácie

Obr. B.3: Webové rozhranie Gotify

Obr. B.4: Stránka pre vytváranie prístupových tokenov klientov

Obr. B.5: Stránka pre vytváranie prístupových tokenov aplikácií

Príloha C

Dropbox

Obr. C.1: Generovanie OAuth2.0 tokenu