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

RASPBERRY PI - MODELOVÉ ÚLOHY

RASPBERRY PI - MODEL TASKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN KLOBUŠICKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Dr. Ing. PAVEL ZEMČÍK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Klobušický Martin**
Program: Informační technologie
Název: **Raspberry Pi - modelové úlohy**
Raspberry Pi - Model Tasks
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte zařízení Raspberry Pi a dokumentaci k němu (model Raspberry Pi 3B+), prostudujte i relevantní literaturu.
2. Navrhněte jednu nebo více modelových úloh (hra s neobvyklými periferiemi, domácí server, analýza zvuku apod.)
3. Zvolte a popište vhodný způsob implementace vybrané modelové úlohy (OS, software, vývojové prostředky, periferie) a popište vlastnosti.
4. Implementujte vybranou úlohu a demonstrujte její funkčnost.
5. Diskutujte dosažené vlastnosti a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

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

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cieľom tejto bakalárskej práce bolo zhrnutie aktuálnych riešení a projektov ktoré používajú Raspberry Pi a implementácia úloh ktoré by mohli byť riešené na Raspberry Pi. Úlohy by mali byť určené pre edukatívne a demonštračné účely. Prvou úlohou je framework pre spracovanie zvukových dát. Druhou úlohou je Informačný systém ktorý získava a zobrazuje dáta zo senzoru na meranie teploty a vlhkosti.

Abstract

The main goal of this bachelor thesis was to sumarize current solutions and projects that use Raspberry Pi and implementation of tasks that could be solved on Raspberry Pi. Task should be designed for educational and demonstration purposes. The first task is a framework for audio data processing. The second task is an information system that acquires and displays data from a sensor for measuring temperature and humidity

Klíčová slova

Raspberry Pi, spracovanie signálov, modelové úlohy, python

Keywords

Raspberry Pi, signal processing, model tasks, python

Citace

KLOBUŠICKÝ, Martin. *Raspberry Pi - modelové úlohy.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Raspberry Pi - modelové úlohy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením profesora Pavla Zemčíka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Klobušický
28. července 2020

Poděkování

Chcem poďakovať profesorovi Pavlovi Zemčíkovi za odborné rady a pomoc pri písaní práce.

Obsah

1 Úvod **2**

2 Platforma Raspberry **3**
2.1 Úvod do Raspberry 3
2.2 Vývojové nástroje pre Raspberry 7
2.3 Základ operačných systémov Raspberry 9
2.4 Základy jazyka Python v Raspberry 10

3 Raspberry vo výuke **12**
3.1 Kluby a akcie k problematike Raspberry 12
3.2 Herná konzola Raspberry - RetroPie 13
3.3 Spracovanie zvuku pomocou Raspberry - PiSound 14
3.4 Raspberry server na prehrávanie hudby - Pi MusicBox 15
3.5 Ovládanie Raspberry pomocou pohybu - Wavepad 16
3.6 Rozpoznávanie evidenčných čísiel vozidiel na Raspberry 17

4 Analýza a návrh riešení **18**
4.1 Analýza existujúcich riešení 18
4.2 Porovnanie dostupného hardware 19
4.3 Rozvedenie cieľa 20
4.4 Technické zadanie 21

5 Implementácia úloh **23**
5.1 Ciel frameworku 23
5.2 Príklad použitia 24
5.3 Popis riešenia frameworku pre prácu so zvukom 25
5.4 Popis riešenia systému s meteo stanicou 33
5.5 Testovanie 36

6 Záver **40**

Literatura **41**

Kapitola 1

Úvod

Prvé počítače na svete boli veľké ako celá miestnosť a ich výkon bol tak minimálny, že dnes by nestačil ani na spustenie operačného systému. S vývojom nových technológií sa však počítače začali zmenšovať a ich výkon stále stúpal. Dnes už existujú počítače o veľkosti kreditnej karty a jedným z takýchto počítačov je aj Raspberry Pi. Cieľom mojej práce bolo navrhnúť a popísať implementáciu niekoľkých modelových úloh ktoré by mohli byť riešené na Raspberry Pi a tieto úlohy následne implementovať.

Ako prvú úlohu som si vybral prácu so spracovaním zvuku. Podstatou úlohy je zachytenie vstupných dát, či už zo súboru alebo z mikrofónu, a následné predanie týchto dát do užívateľskej funkcie, kde má užívateľ prístup ku všetkým potrebným zdrojom. V tejto funkcii môže naprogramovať vlastné spracovanie dát v podobe filtru. Nato sa upravené dáta zapíšu do výstupného súboru. Druhou úlohou je domáci server so senzorom na meranie teploty a vlhkosti. Senzor v časových intervaloch zaznamenáva teplotu a vlhkosť a tieto dáta posiela na server, kde si ich užívateľ môže zobraziť. Táto úloha slúži najmä ako demonštrácia a ukazuje výhody v architektúre a veľkosti Raspberry Pi. Užívateľ má pripravený funkčný server a jednoducho rozšíriteľný informačný systém.

Na internete je obrovské množstvo projektov, ktoré rôznym spôsobom ukazujú prednosti Raspberry. Dá sa použiť na všetko, od používania ako desktopový počítač až po zaujímavé projekty, čím je napríklad ovládanie chytrej domácnosti. Každý z týchto projektov je niečim výnimočný. Komplexné riešenia však väčšinou nebývajú úplne zadarmo, a vyžadujú obrovské množstvo externých knižníc, s ktorými môže mať Raspberry pri nesprávnej konfigurácií problém. Moje riešenie používa minimum externých knižníc a celé spadá pod MIT licenciu, takže každý môže program upraviť podľa seba a slobodne ho použiť.

V ďalšej kapitole, kapitole 2 som zhrnul vlastnosti Raspberry Pi, od jeho histórie až po architektúru. Nájde sa tu aj úvod do operačných systémov, ktoré Raspberry používa, a popis jazyku Python. Kapitola 3 obsahuje niektoré existujúce projekty, ktoré používajú Raspberry Pi. V kapitole 4 som analyzoval existujúce riešenia a na základe výsledkov navrhol vlastné príklady. Kapitola 5 obsahuje popis implementácie úloh a príklad ich použitia a testovanie.

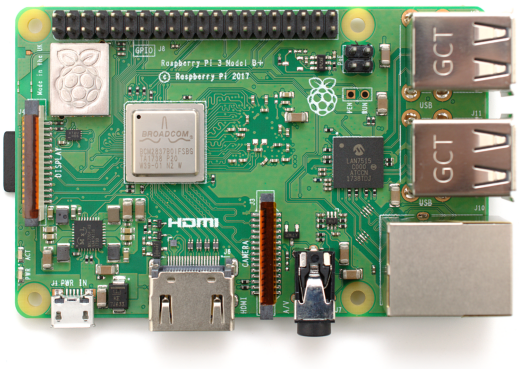
Kapitola 2

Platforma Raspberry

Obsahom tejto kapitoly je popis platformy Raspberry Pi. Kapitola obsahuje iba informácie, ktoré bezprostredne súvisia s mojou prácou. Obsah kapitoly sa nedá považovať za encyklopedický prehľad kvôli obmedzeniu maximálneho rozsahu práce.

2.1 Úvod do Raspberry

Raspberry Pi [7] je plne funkčný počítač s veľkosťou kreditnej karty. Aj napriek jeho veľkosti má všetky porty na pripojenie potrebných periférii. Má 4 USB-2 porty na pripojenie akéhokoľvek zariadenia na USB. Hneď pri USB portoch je Ethernet port na pripojenie Raspberry k sieti pomocou RJ45 konektoru. Za Ethernet portom sa nachádza 3.5mm AV jack, primárne slúži na pripojenie slúchadiel, má však aj špeciálnu funkciu. Keďže dokáže prenášať audio aj video signál, môže byť pripojený napríklad k projektoru pomocou TRRS adaptéru. Hneď vedľa sa nachádza CSI kamera konektor, kde môže byť zapojená kamera špeciálne určená pre Raspberry Pi. HDMI port ponúka pripojenie k monitoru alebo televízii vo veľmi dobrej kvalite. Vľavo od HDMI portu je mikro USB, ktoré slúži na pripojenie napájania k Raspberry. Rovnaké napájanie používajú aj mobilné telefóny, avšak odporúča sa použiť originál napájanie od Raspberry. Posledným konektorom je DSI, ktorý slúži na pripojenie Raspberry k dotykovému displeju.

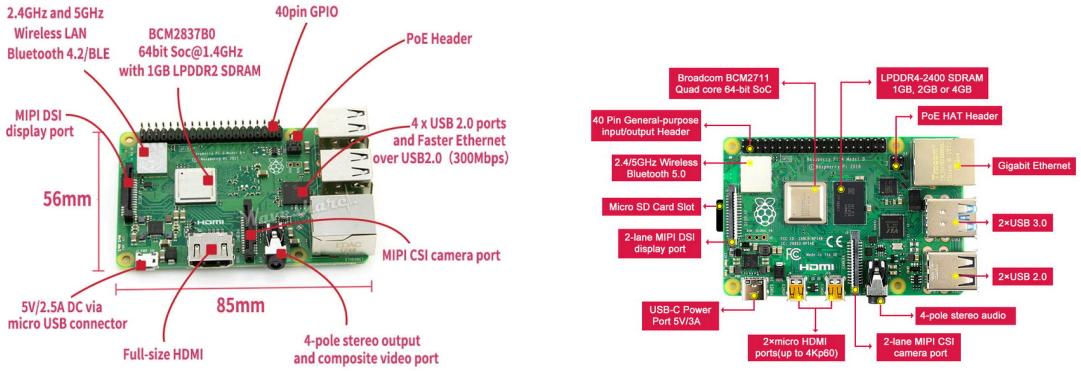


Obrázek 2.1: Raspberry Pi 3B+ [1]

[1] https://robu.in/product/raspberry-pi-3-model-b-bcm2837b0-soc-iot-poe-enabled/

- **SoC** (system-on-chip) - Je jadro celého Raspberry Pi, ktoré obsahuje CPU a GPU. Nachádza sa v strede Raspberry.
- **RAM** - Ďalší hlavný komponent Raspberry, ktorý uchováva dočasné dáta. V starších modeloch bol súčasťou SoC, ale od modelu 4 sa stal samostatným komponentom.
- **Komunikačný modul** - Na doske sa taktiež nachádza modul na komunikáciu pomocou Wifi a bluetooth.
- **PMIC** - Stará sa o rozdelenie energie, ktorá prichádza cez mikro USB do celého Raspberry.
- **správca USB portov** - Stará sa o funkčnosť USB portov a Ethernet portu.

Tento popis sa samozrejme môže líšiť v závislosti od modelu Raspberry Pi. Na obrázku 2.2 môžeme vidieť pár zásadných rozdielov medzi modelom 3 a modelom 4.



(a) Raspberry model 3

(b) Raspberry model 4

Obrázek 2.2: Raspberry Pi porovnanie modelov

V dnešnej dobe existuje už viac verzii Raspberry Pi. Úplne prvá generácia Raspberry Pi vznikla v roku 2012, a bolo to Raspberry Pi model B. Hneď rok na to vyšlo Raspberry model A. Momentálne je už dostupné aj Raspberry model 4, ktoré sa začalo predávať v roku 2019. Každá generácia Raspberry ponúka modely A a B. Hlavný rozdiel medzi nimi je v tom, že model A je kompaktná verzia modelu B. Má menej portov a menej funkcií, zväčša aj o trochu slabší hardware. Prehľad vydaných verzií Raspberry sa nachádza na obrázku 2.4.

Model	RPi 3 B	RPi 3B+	RPi 4 B
SoC typ	Broadcom BCM2837	Broadcom BCM2837B0	Broadcom BCM2711
CPU	4 x Arm Cortex-A53, 1.2GHz	4 x Arm Cortex-A53, 1.4GHz	4 x Arm Cortex-A72, 1.5GHz
RAM	1 GB	1 GB	1GB/2GB/4GB
GPU	Broadcom VideoCore IV	Broadcom VideoCore IV	Broadcom VideoCore VI
USB porty	4	4	4(2 x USB 3.0 + 2 x USB 2.0)
Ethernet	100Mbit/s Ethernet	Gigabit Ethernet (max. 300 Mbps)	Gigabit Ethernet (bez limitu)
WiFi	WiFi 802.11n	WiFi 802.11ac Dual Band	WiFi 802.11ac Dual Band
Bluetooth	4.1.	4.2 BLE	5.0 BLE
Výstup videa	HDMI/3.5mm Comp./DSI	HDMI/3.5mm Comp./DSI	mikro-HDMI/3.5mm Comp./DSI
Výstup zvuku	HDMI/3.5mm Composite	HDMI/3.5mm Composite	HDMI/3.5mm Composite
GPIO piny	40	40	40
Pamäť	MicroSD	MicroSD	MicroSD

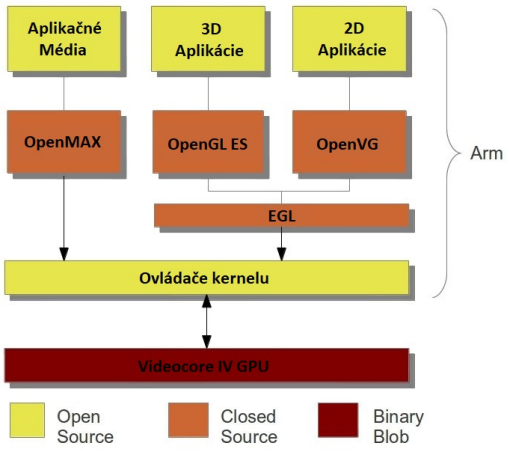
Obrázek 2.3: Hardware posledných verzií Raspberry

Prvá generácia Raspberry bola veľmi nenáročná a jej hardware bol oproti dnešným modelom veľmi slabý. Mala len jedno-jadrový 700MHz ARM11 procesor a 256-512 MB RAM. Druhá generácia už bola značne lepšia, mala štvor-jadrový 900MHz ARM Cortex-A7 procesor, 1GB RAM a rovnaké GPU a bola asi šesťnásobne rýchlejšia ako prvá generácia. Následne vyšla generácia Raspberry Pi Zero, ktorá neobsahovala väčšinu I/O portov a ani GPIO piny, bola však za výrazne menšiu cenu.

Family	Model	Form Factor	Ethernet	Wireless	GPIO	Released	Discontinued
Raspberry Pi	B	Standard[a]	Yes	No	26-pin	2012	Yes
	A		No			2013	Yes
	B+		Yes		40-pin	2014	
	A+	Compact[b]	No			2014	
Raspberry Pi 2	B	Standard[a]	Yes	No		2015	
Raspberry Pi Zero	Zero	Zero[c]	No	No		2015	
	W/WH			Yes		2017	
Raspberry Pi 3	B	Standard[a]	Yes	Yes		2016	
	A+	Compact[b]	No			2018	
	B+	Standard[a]	Yes			2018	
Raspberry Pi 4	B (1 GiB)	Standard[a]	Yes (Gigabit Ethernet)	Yes		2019[31]	Yes[1]
	B (2 GiB)						
	B (4 GiB)						
	B (8 GiB)					2020	

Obrázek 2.4: Prehľad modelov Raspberry Pi [2]

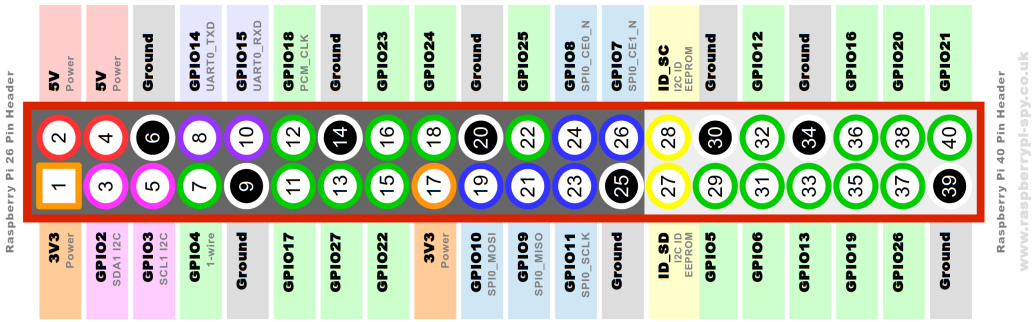
Postupne, ako vychádzali novšie generácie, pribúdali aj nové funkcie [3] a stúpala cena. Je preto lepšie si rozmyslieť, na aké účely bude Raspberry použité, kvôli výberu generácie a modelu, ktoré bude najlepšie v pomere cena/výkon. Raspberry používa ako operačný systém Raspbian, ktorý odporúčajú výrobcovia. Môže však bežať na v podstate akomkoľvek systéme založenom na Linuxe. Základným ovládačom, ktorý používa, je VideoCore IV GPU. Je označovaný ako "binary blob", čo znamená, že je to uzavretá časť binárneho kódu v open source systéme. Je načítaný do GPU pri bootovaní systému. Aplikačné programy volajú knižnice OpanMax, OpenGL ES a OpenVG, a tie následne volajú funkcie Linuxového jadra, ktoré komunikuje s VideoCore IV. Video aplikácie používajú OpenMAX, 3D aplikácie používajú OpenGL ES a 2D aplikácie používajú OpenVG.



Obrázek 2.5: Architektúra Raspberry Pi [3]

[2] https://en.wikipedia.org/wiki/Raspberry_Pi

Jednou z najväčších výhod, ktoré Raspberry Pi ponúka sú GPIO piny [2]. GPIO piny môžu nadobúdať hodnoty logickej "0"alebo "1", vďaka čomu môžu slúžiť ako vstup alebo výstup pre pomocný hardware, ako napríklad teplotné senzory, led diódy, tlačítka a rôzne iné veci. Vďaka GPIO má Raspberry obrovské množstvo využití, od správy chytrej domácnosti až po vytvorenie mobilného telefónu alebo rôznych robotov. Na doske sa nachádzajú dva 5V piny a dva 3V3 piny, ktoré sa nedajú konfigurovať. Zvyčajne je na pin privedené buď vysoké napätie (3V3), alebo nízke napätie (OV alebo zem). K tomu, aby sa predišlo k poruche by napätie, ktoré má byť privedené na každý pin, malo byť v logickom rozsahu tak, ako to udáva špecifikácia. Ak však privedieme vyššie, alebo naopak nižšie napätie, môže dôjsť k poruche celého čipu. Platí aj to, že keď privedieme napätie, ktoré sa nedá považovať za vysoké (3V3) ani za nízke (0V), môže nastať neočakávané chovanie.



Obrázek 2.6: GPIO piny [4]

Vďaka GPIO pinom sa Raspberry značne odlišuje od klasických počítačov, pretože dovoľuje pripojenie najrôznejších zariadení a senzorov. V podstate všetky piny slúžia ako výstup alebo vstup, niektoré piny majú však špeciálny účel. Ako je vidieť na obrázku, GPIO piny nám dovoľujú ovládať funkcie [6], medzi ktoré patria: UART, SPI, I2C a HAT pomocou EEPROM.

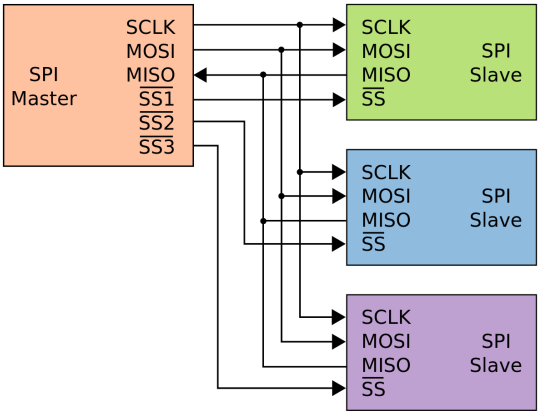
- **Napájanie** - Vyznačené oranžovou farbou sú piny 1 a 17, ktoré ponúkajú napájanie 3.3V a červenou farbou sú vyznačené piny 2 a 4, ktoré ponúkajú napájanie 5V pre externé zariadenia. Tieto zariadenia sa môžu pripojiť k Raspberry pomocou ostatných pinov.

- **UART** - Vyznačené fialovou farbou sú UART piny 8 a 10 pre posielanie dát (TxD) a pre príjem dát (RxD). UART slúži k asynchronnému sériovému prenosu, kde môže byť špecifikovaná rýchlosť aj jeho formát. Zapojenie je realizované tak, že vysielač prvého zariadenia je pripojený na prijímač druhého zariadenia a naopak. Musia byť spoločne uzemnené, a Raspberry sa tak môže pripojiť k akémukoľvek zariadeniu cez sériový interface.

- **SPI** - Farebne vyznačené modrou farbou sú piny, ktoré umožňujú master-slave komunikáciu pomocou pinu 19 MOSI (Master output , Slave Input), 21 MISO (Master Input, Slave output), 23 SCLK (Hodinový signál) a CE (Chip Enable). Komunikácia je realizovaná pomocou spoločnej zbernice kde master posiela dáta pomocou MOSI

[3] https://en.wikipedia.org/wiki/Raspberry_Pi
[4] https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

a prijíma dáta pomocou MISO. V prípade, že master ovláda viac ako jeden slave,tak výber zariadenia, s ktorým má práve komunikovať je vybraný pomocou CE pinov. Na obrázku vidíme, že Raspberry, ktoré používa tieto GPIO, môže ovládať až 2 zariadenia.



Obrázek 2.7: SPI komunikácia [5]

- **I2C** - Vyznačené ružovou farbou sú piny 3 a 5, kde sa pomocou dvoch obojsmerných vodičov dajú prenášať dáta. Konkrétne sú to SDA (datový kanál) a SCL (hodinový signál). Komunikácia funguje na základe jedného alebo viacerých mastrov a niekoľko slave zariadení. Master generuje hodinový signál na SCL a začne prenášať dáta cez SDA. Raspberry takto pomocou I2C môže ovládať napríklad rôzne senzory.
- **HAT** - Vyznačené žltou farbou sú piny 27 a 28, ktoré slúžia na pripojenie takzvaných HAT dosiek, ktoré dovoľujú Raspberry aby automaticky nakonfigurovalo piny a ovládače pripojeného zariadenia a to pomocou dvoch I2C EEPROM pinov. Každý si takto môže navrhnúť vlastnú dosku na základe istých špecifikácií.

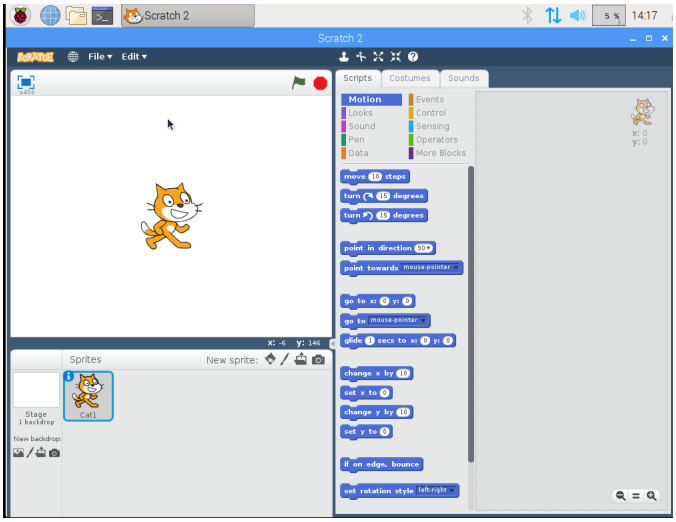
Všetky piny slúžia teda ako vstup, až na tie, ktoré slúžia na napájanie, uzemnenie alebo UART. UART piny môžu byť tiež nakonfigurované ako vstup/výstup. Aby sme s pinmi mohli pracovať a konfigurovať ich podľa našich potrieb, musíme využiť nejaký programovací jazyk ako Python alebo C.

2.2 Vývojové nástroje pre Raspberry

Jednou z výhod Raspberry a jeho systému Raspbian je, že v systéme je už predinštalované obrovské množstvo programov a programových prostredí. Napríklad jazyk Scratch.

Jazyk Scratch je určený najmä pre deti a úplných začiatočníkov. Ponúka rôzne príkazy, podmienky, cykly a mnoho iného. Tie sa potom skladajú pod seba a tým vzniká program. Scratch vie pracovať aj so zvukmi, obrázkami, kamerou a rôznymi senzormi, takže je možné v ňom naprogramovať aj dosť pokročilé veci. Scratch síce nie je výsadou Raspberry Pi, dokázal by fungovať aj v PC či na webe, no v Raspberry ide nastaviť tak, aby sa priamo

[5] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
[6] https://en.wikipedia.org/wiki/Scratch_(programming_language)



Obrázek 2.8: Scratch [6]

bootovalo do rozhrania jazyka Scratch, a preto je skvelým nástrojom na výučbu na základných školách. Niektoré z ďalších užitočných programov sú napríklad [1]:

- **BlueJ** - Veľmi jednoduché grafické vývojové prostredie pre jazyk Java. Má veľmi minimalistické a jednoduché prostredie, ktoré bolo navrhnuté hlavne na výuku. V BlueJ sa dá pristupovať k objektom, zisťovať ich hodnoty, volať nad nimi metódy a predávať im parametre. Dovoľuje tiež prístup ku všetkému hardware, čo Raspberry ponúka, pomocou otvorenej knižnice Pi4J.

- **Geany** - Vývojové prostredie, ktoré podporuje obrovské množstvo jazykov, napríklad C, Java, PHP, HTML, Python a mnoho iných. Bolo vytvorené za účelom aby nebolo závislé na GNOME alebo KDE a vyžaduje iba GTK2 knižnice. Podporuje automatické dopĺňanie, zvýrazňovanie syntaxe, snipety a zabudovaný systém na preloženie a spustenie programu.

- **Greenfoot** - Vývojové prostredie pre jazyk Java. Podobne ako BlueJ je veľmi priateľský k začiatočníkom a má jednoduché prostredie. Ponúka tiež možnosť jednoduchého prenosu do iného prostredia a publikovania kódu, či už online, alebo offline formou.

- **Mathematica** - Moderný systém na prácu hlavne v technických a vedeckých oboroch. Používaný najmä na neurálne siete, spracovanie obrazu a vizualizáciu.

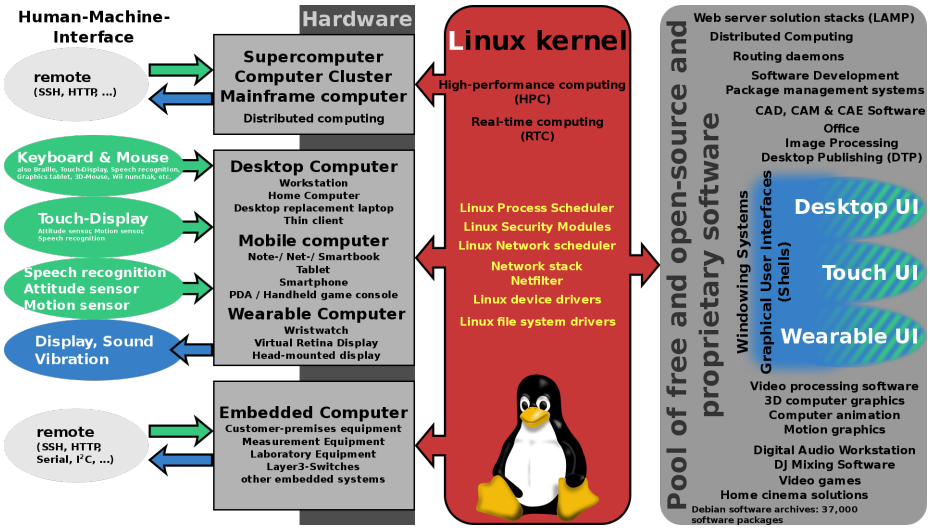
- **Ninja-IDE** - Vývojové prostredie pre Python aplikácie, podporuje však aj iné jazyky. Je extrémne jednoduché a bez zbytočných prvkov. Podporuje zmenu kódu z Python2 na Python3 a obsahuje projektového manažéra.

- **Wolfram** - Podobne ako Mathematica je veľmi silný viacúčelový jazyk, ktorý podporuje prácu s algebraickými výpočtami, funkcionálnym programovaním a logickým programovaním. Má zabudované funkcie na generovanie a prácu turingových strojov, vytváranie grafiky a zvuku, analýzu 3D modelov, operácie s maticami a riešenie diferenciálnych rovníc.

- **Squeak** - Open-source prostredie pre jazyk Smalltalk. Squeak bol použitý na vytvorenie prvej verzie programovacieho jazyka Scratch.
- **Arduino IDE** - Aplikácia napísaná v Jave, ktorá obsahuje editor, zvýrazňovanie syntaxe, konzolu, a poskytuje jednoduché preloženie a nahranie programu priamo do Arduina. Podporuje jazyky C a C++, ale je potrebné držať sa špeciálnych pravidiel. Toto prostredie je obzvlášť výhodné najmä vďaka tomu, že sa Arduino môže pripojiť cez USB a spolu s Raspberry veľmi dobre komunikuje.

2.3 Základ operačných systémov Raspberry

Linux [9] je "open source"operačný systém z rodiny Unixových operačných systémov. Linux naprogramoval začiatkom deväťdesiatych rokov Linus Torvalds. Pôvodne sa slovom "Linux"označovalo len jadro systému (kernel), ale ako sa systém stále zdokonaľoval a vyvíjal, začalo sa používať na označenie celej platformy vrátane grafických prostredí a programového vybavenia. Kernel bol pôvodne vyvinutý pre počítače s procesormi na architektúre Intel x86. V súčastnosti však podporuje viac architektúr ako akýkoľvek iný operačný systém. Linuxové jadro tvorí aj základ operačných systémov ako napríklad `Android` alebo `Chrome OS`.



Obrázek 2.9: Linux [7]

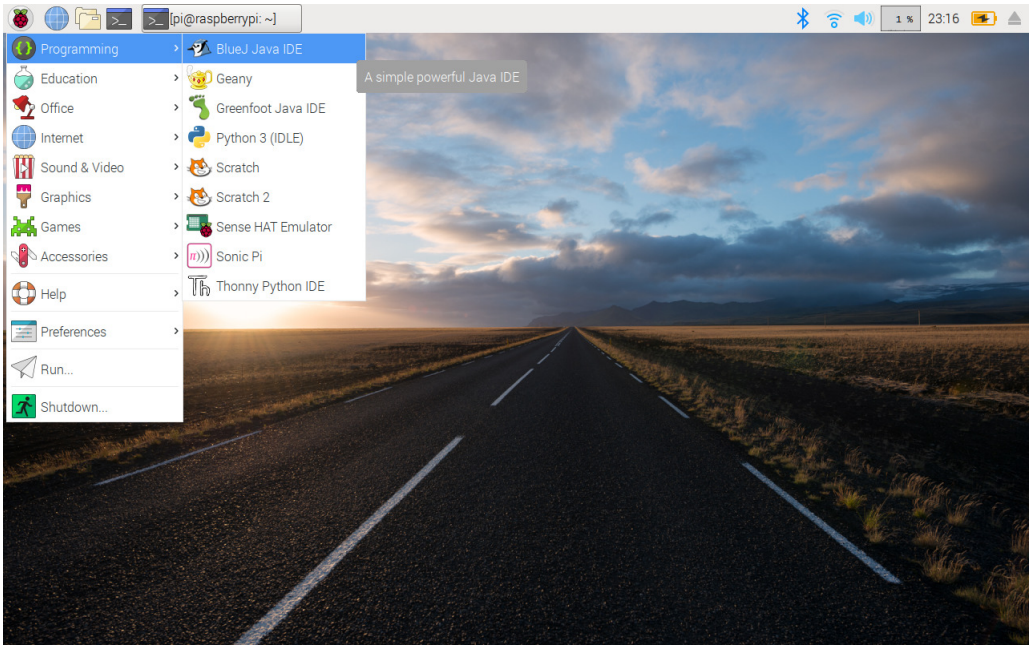
Základnou vlastnosťou Linuxu je fakt, že je "open source", to znamená, že si ktokoľvek môže zobrať kód a vytvoriť vlastný operačný systém na jeho báze. Medzi najpopulárnejšie Linuxové distribúcie patrí napríklad `Debian`, `Fedora`, `Ubuntu`. Do základných výhod Linuxu patrí napríklad:

- Stabilita,
- Bezpečnosť,
- Rýchlosť,
- Nenáročnosť na hardware,

[7] https://en.wikipedia.org/wiki/Linux

- Nezávislosť od veľkých korporácií.

Aj napriek všetkým výhodám, ktoré Linux ponúka, a v ktorých je lepší ako jeho konkurenti ho používajú len 2% stolových počítačov. To je najmä z dôvodu, že je systémom užívateľsky o dosť komplikovanejším, než povedzme Microsoft Windows, ktorý beží až na 87% desktopových počítačov. Tieto čísla sa však vzťahujú len na stolové počítače. Ak sa jedná o o webový server, prevláda stále Linux, na ktorom beží viac ako 60% webových serverov. Dôvodom, prečo Linux prevláda na poli webových serverov je hlavne kvôli natívnej implementácii programov ako `Apache`, `MySQL` alebo aj `PHP`. Raspbian [4] je operačný systém primárne určený pre Raspberry Pi. Je založený na báze Debianu. Napísal ho Mike Thompson a Peter Green v roku 2012. Pôvodne bol vytvorený ako nezávislý projekt, no v roku 2015 ho "Raspberry Pi Foundation"začala ponúkať ako svoj primárny operačný systém. Obsahuje viac než 35 000 predinštalovaných balíčkov a rôznych programov a je stále vo vývoji.



Obrázek 2.10: Raspbian [8]

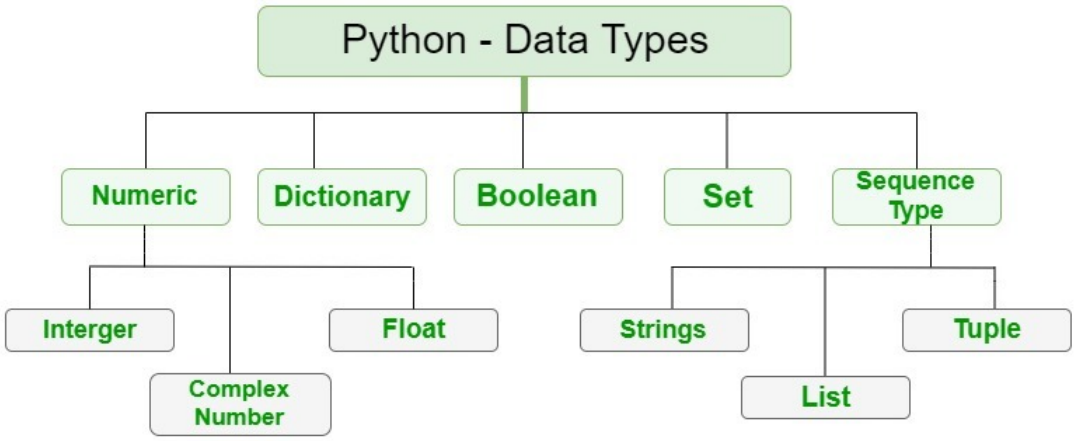
Systém je vysoko optimalizovaný pre Raspberry Pi a jeho slabší hardware. Pre svoje grafické prostredie používa `LXDE` v spojení s `Openbox`, ktorý sa stará o správu okien.

2.4 Základy jazyka Python v Raspberry

Python [8] je moderný interpretovaný programovací jazyk, ktorý vytvoril v roku 1991 Guido van Rossum. Python podporuje objektovo orientované, štrukturované a aj funkcionálne programovanie. Je to dynamicky typovaný jazyk, ktorý pracuje s vysoko úrovňovými dátovými typmi a na správu pamäte používa garbage kolektor. Aj napriek tomu, že je tak silný a flexibilný, je stále relatívne jednoduchý a dobrý pre začiatočníkov. V roku 2000 bola vydaná druhá verzia Python 2.0 a v roku 2008 tretia verzia Python 3.0, ktorá priniesla

[8] https://en.wikipedia.org/wiki/Raspberry_Pi_OS

mnoho zmien. Vďaka tomu väčšina kódu, ktorá bola napísaná na Python 2.0 sa už na novšej verzii nedala spustiť. V roku 2020 skončila oficiálna podpora Pythonu 2.0, a tak už nebudú vychádzať žiadne nové vylepšenia ani balíčky. Podporovaný je momentálne iba Python 3.5.x a vyššie verzie. Python je cross platform, a tak by väčšina kódu mala byť spustiteľná na akomkoľvek operačnom systéme, samozrejme ak obsahuje všetky potrebné balíčky. Python gramatika je veľmi jednoduchá a namiesto používania špeciálnych znakov je dosť založená na medzerách a tabulátoroch, ktoré sú veľmi dôležité pre správne štruktúrovanie programu. Jazyk je tiež veľmi jednoducho rozšíriteľný pomocou nových modulov, ktoré môžu byť napísané napríklad v C++. Jeho základná knižnica sa považuje za jednu z najväčších, pričom poskytuje prácu s grafickým rozhraním, pripojením k databáze, generovaním náhodných čísel, regulárnych výrazov a jednotkových testov. Veľmi dobrý spôsob testovania ponúka aj interaktívny režim, kde môžu byť príkazy Pythonu zadávané priamo do príkazového riadku terminálu. Vďaka tomu môže byť otestovaný kus kódu, ktorý sa následne integruje do programu.



Obrázek 2.11: Python datové typy [9]

Python podporuje všetky základné dátové typy. Premenné v Pythone však nemajú typ, ale iba hodnoty. Má stredne prísnu typovú kontrolu, takže je možné pracovať s komplexným a celým číslom bez explicitného pretypovania, nemôžeme však pretypovať reťazec na číslo. Jedným zo základných aspektov sú sekvenčné dátové typy. Sú to objekty, ktoré obsahujú iné objekty, a môžeme k nim pristupovať pomocou indexov. Špeciálnym dátovým typom je slovník, kde je na jednom indexe uložená hodnota a kľúč.

[9] https://www.geeksforgeeks.org/python-data-types/

Kapitola 3

Raspberry vo výuke

Táto kapitola obsahuje prehľad niektorých projektov [12], ktoré používajú pri implementácií Raspberry Pi. Pre Raspberry Pi existuje obrovské množstvo projektov, a tak som vybral niektoré zaujímavé alebo relevantné k tejto práci.

3.1 Kluby a akcie k problematike Raspberry

Obrovskou výhodou platformy Raspberry sú rôzne podujatia, medzi ktoré patria napríklad `"Raspberry Jams"` [11]. Ich hlavnou úlohou je zaujať a naučiť študentov niečo nové. To realizujú pomocou veľkého množstva rôznych prezentácií a súťaží. Počet týchto akcií stále rastie. Len v Anglicku sa ich v roku 2017 konalo v viac ako 160 a dokopy sa ich zúčastnilo viac ako 10 000 študentov. Študenti tvoria kluby, z ktorých najznámejšie sú `"Code Clubs"` a `"CodeDojos"`. `"Code Clubs"` bolo v roku 2017 v Anglicku viac ako 6000 a a zúčastnilo sa ich 90 000 študentov. `"CodeDojos"` bolo v Anglicku v roku 2017 približne 150 a zúčastnených bolo viac ako 4000.

Názov	**Počet klubov/akcii**	**počet zúčastnenách detí**
Raspberry Jam	160	10 000
Code Club	6 000	90 000
CodeDojo	150	4 000

Tabulka 3.1: Tabuľka klubov a akcií v Anglicku roku 2017

OpenLabTools [5] je projekt pod záštitou "University of Cambridge"[1] a nadácie Raspberry Pi, ktorého hlavným cieľom je zaistiť informačnú základňu pre prácu s rôznymi nástrojmi,ako je napríklad Raspberry Pi. Vyvíjajú rôzne moduly a komponenty pre získavanie dát z rôznych senzorov, spracovanie údajov a prácu s 3D technológiou. Všetky návody a moduly, ktoré vytvárajú sú voľne dostupné a každý s nimi môže pracovať. Všetky nástroje sú vyvíjané a pod správou študentov z univerzity. Ich aktuálne projekty sú napríklad automatizovaný mikroskop a testovanie systémov. Ďalším projektom, ktorý je priamo zameraný na prácu s Raspberry, je použitie kamerového modulu i kamery na snímanie a spracovanie obrazu pomocou SimpleCV [2]. Pracujú tiež na vývoji nástrojov pre komunikáciu Arduina a Raspberry Pi pomocou USB, GPIO pinov alebo aj I2C.

[1] Jedna z najväčších univerzít v Amerike
[2] Knižnica pre spracovanie obrazu v Pythone

Obrázek 3.1: Mikroskop, ktorý vyvíja OpenLabTools [3]

Ich projekt automatizovaného mikroskopu začal už v roku 2013, čo je iba rok po vydaní prvej generácie Raspberry. Používajú Arduino pre ovládanie mikroskopu a Raspberry slúži ako hlavný systém. Ten vykonáva požiadavky, ktoré dostáva od Arduina. Ide o jeden z prvých veľkých projektov, kde bolo využité Raspberry Pi.

3.2 Herná konzola Raspberry - RetroPie

Projekt používa Raspberry Pi na vytvorenie malej retro konzoly. Jeho základom je Raspbian a rôzne emulačné nástroje, vďaka ktorým aj slabý hardware Raspberry dokáže spustiť niektoré zo starých počítačových a iných konzolových hier. Celý systém obsahuje viac ako 800 rôznych hier. Je možnosť použiť WiFi modul k pripojeniu na sieť a hrať hry s ostatnými ľuďmi na lokálnej sieti. Záujemcovia si môžu väčšinu vecí nakonfigurovať podľa vlastných predstáv. Projekt podporuje všetky generácie a verzie Raspberry.

Obrázek 3.2: RetroPie [4]

[3] http://www.openlabtools.org/

[4] https://diyprojects.io/retrogaming-retropie-mini-console-raspberry-pi-3-3-5-hdmi-lcd-touch-screen/

Projekt bol pôvodne súčasťou väčšieho projektu `PetRockBlock`, ktorý pre Raspberry vyvíjal rôzne periférie ako ovládače a gamepady[5]. RetroPie však prerástol do väčších rozmerov, a pokračuje ako samostatný projekt. S pôvodnou organizáciou však stále spolupracujú, a všetky ovládače, ktoré vyvíja `PetRockBlock` sú kompatibilné s RetroPie. Všetky zdrojové kódy sú voľne prístupné na github pod licenciou GPL. Každý si teda môže zdrojový kód stiahnuť, ale nemôže ho bez povolenia meniť, ani inak upravovať. RetroPie sa dá zakúpiť ako vopred pripravený kit so všetkým potrebným nainštalovaným alebo sa dá jednoducho vytvoriť zo základnéj dosky Raspberry.

3.3 Spracovanie zvuku pomocou Raspberry - PiSound

PiSound je riešenie, ktoré pomáha Raspberry Pi s prácou so zvukom. PiSound je HAT doska, ktorá sa k Raspberry pripojí pomocou GPIO pinov. Je to vylepšená zvuková karta, ktorá dokáže posielať a prijímať zvukové dáta cez jack konektory, a posielať MIDI signály do zariadení, ktoré sú pripojené a kompatibilné, napríklad elektrická gitara.

Obrázek 3.3: PiSound HAT doska [6]

Doska poskytuje:

- 2 x 6mm vstup/výstup jack konektory,
- 2 x DIN-5 MIDI vstup/výstup konektory,
- Potenciometre na zvyšovanie hlasitosti,
- tlačítko pre aktiváciu manipulácie so zvukom.

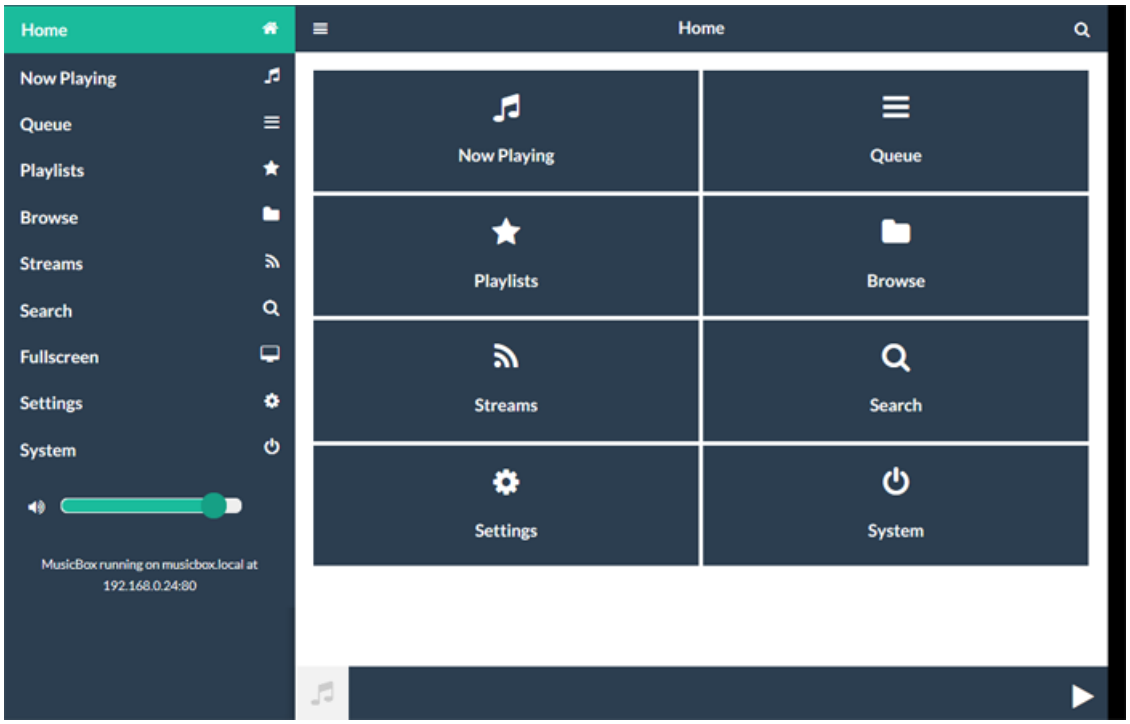
PiSound je veľmi úspešný projekt. Raspberry by tak mohlo nahradiť drahé hardvéry, ktoré sa používajú na live koncertoch alebo pri vytváraní hudby. Po pripojení hudobného nástroja dokáže vďaka MIDI konektorom nahrať zvuk zo zariadenia a upraviť ho. Pomocou plug-inu LV2, SuperCollider a SonicPi sa dá vytvoriť nahrávacia stanica, pričom sa dajú ovládať efekty pripojeného zariadenia. Inštalácia je veľmi jednoduchá, dá sa dokonca nainštalovať z USB kľúča jednoduchým stlačením tlačítka, bez použitia monitoru či klávesnice.

[5] konzolový ovládač pre hranie hier, je vidieť na obrázku 3.2
[6] https://www.raspberrypi.org/blog/making-sweet-sweet-music-pisound/

Karta sa môže pripojiť k WiFi, a tak sa môžu MIDI správy posielať priamo z mobilu alebo tabletu. Vďaka už spomínaným plug-inom človek nemusí byť ani hudobník, aby dokázal niečo vytvoriť, a tak je to skvelý nástoj aj pre začiatočníkov. Jednoduchým Python kódom sa dá vytvoriť aj internetová rádio stanica. Projekt je celkom nový a doska stojí 99 eur.

3.4 Raspberry server na prehrávanie hudby - Pi MusicBox

MusicBox je lacné a rýchle riešenie pre vytvorenie streamovacej stanice pre hudbu. Medzi takéto stanice patrí napríklad Spotify, Google Music a SoundCloud. Využívajú ho aj rôzne podcasty[7]. MusicBox v roku 2013 vytvoril Wouter van Wijk. Celý systém je postavený na `Mopidy`, čo je v čistom základe iba server, ktorý dokáže prehrávať hudbu a je napísaný v Pythone. MusicBox ho rozširuje grafickým rozhraním, ba aj menšou funkcionalitou.



Obrázek 3.4: Grafické rozhranie MusicBoxu [8]

Dokáže prehrávať aj vlastnú hudbu, ktorá bude nahraná na nejakom zariadení v sieti. Stačí si stiahnuť software na stránkach, a nainštalovať ho do Raspberry. Potom cez webový prehliadač stačí prejsť na IP adresu Raspberry, a nakonfigurovať výstupné zariadenie pre zvuk. Je tu možnosť aj prepojiť si s MusicBoxom napríklad Spotify alebo Google účet a púšťať pesničky odtiaľ. Podporuje vytváranie playlistov[9] a ukladanie konfigurácie aj po reštarte. Zdrojový kód je pod `Apache License 2.0`, takže kód je dostupný, a pre vlastné účely aj modifikovateľný a voľne šíriteľný.

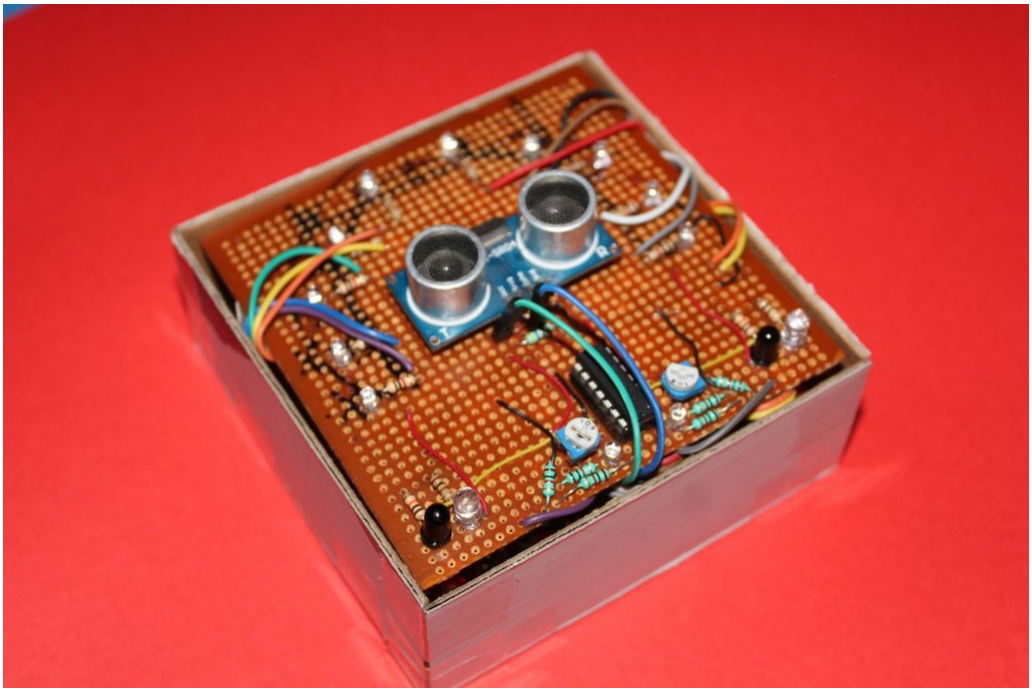
[7] podcast je živý internetový rozhovor na konkrétnu tému
[8] https://www.deviceplus.com/others/diy-projects/linking-with-spotify-via-pi-musicbox/
[9] zoznam pesničiek, ktoré majú byť prehrané

3.5 Ovládanie Raspberry pomocou pohybu - Wavepad

Tento projekt používa Raspberry Pi ako prehrávač hudby, ktorý je ovládaný pohybmi. Raspberry tak dokáže detektovať rôzne druhy pohybu a reagovať na ne zvýšením alebo znížením hlasitosti, pustením ďalšej alebo predchádzajúcej pesničky ba aj pozastavením alebo spustením prehrávania. Pre rozpoznávanie pohybu sa používajú 2 rôzny senzory.

- Ultrazvukový senzor, vďaka ktorému je možné ovládanie hlasitosti.
- 2 senzory na meranie vzdialenosti, ktoré sú použité na prepínanie pesničiek a pozastavenie alebo spustenie prehrávania.

O všetko sa stará jediný obvod, ktorý sa ďalej skladá z veľkého množstva diód a rezistorov. Senzory na meranie vzdialenosti dokážu detektovať všetky objekty v okolí bez akéhokoľvek fyzického kontaktu.



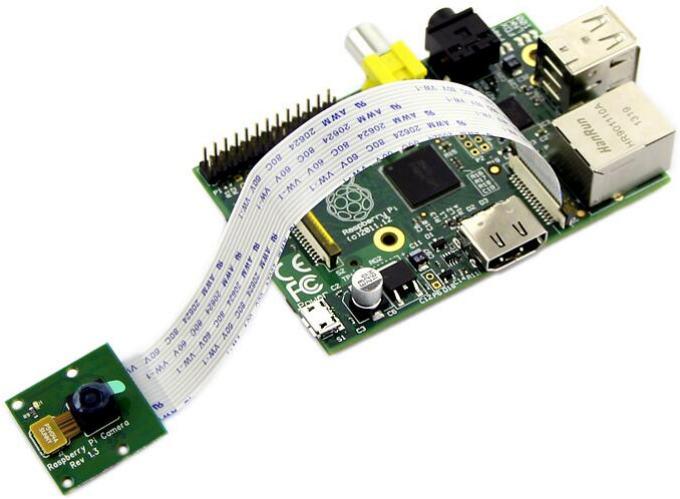
Obrázek 3.5: WavePad obvod [10]

Ako zvukový výstup je použitý malý reproduktor, ku ktorému je pripojený zosilňovač. Ten pomocou potenciometru dokáže ovládať hlasitosť výstupu. Zosilňovač však pracuje na 5V, kvôli čomu je v obvode použitý aj delič napätia. Celý obvod je pripojený k Raspberry Pi pomocou GPIO. Obvod obsahuje 10 LED diód, a tak je potrebné až 12 GPIO pinov pre celý obvod. Celý zdrojový kód programu je voľne dostupný a voľne šíriteľný, a tak si každý môže pridať alebo zmeniť funkcionalitu.

[10] https://www.instructables.com/id/Wavepad-Gesture-Controlled-Raspberry-Pi-Music-Play/

3.6 Rozpoznávanie evidenčných čísiel vozidiel na Raspberry

Projekt ANPR Raspberry Pi používa Raspberry pre indentifikáciu evidenčných čísiel vozidla. Používa strojové učenie, ktoré z fotky vozidla dokáže dekódovať poznávaciu značku a tak zistiť evidenčné číslo. K tomu použije 2 neurálne siete, najskôr k identifikácií poznávacej značky zo všetkých prístupných fotiek, a následne dekódovanie každého znaku na poznávacej značke. Systém je pripravený a dokáže si bezproblémovo poradiť s rôznymi vplyvmi prostredia ako je tma, dážď, rozmazanie a slabé rozlíšenie kamery.



Obrázek 3.6: Raspberry s kamerovým modulom [11]

Algoritmus si dokáže poradiť aj s fotkami, na ktorých ide auto obrovskou rýchlosťou, alebo nemá úplne ostrý uhol. Systém je perfektne optimalizovaný, aby aj slabší hardware Raspberry zvládol zložité spracovanie a strojové učenie. Vďaka architektúre Raspberry mohol byť systém optimalizovaný a dokonca rýchlejší, než na iných zariadeniach, ktoré podporuje. Systém podporuje len niektoré verzie Raspberry.

- Raspberry Pi 4
- Raspberry Pi 3 Model B+
- Raspberry Pi 3 Model B
- Raspberry Pi 2 Model B

Systém však nie je zadarmo, a nie sú dostupné žiadne zdrojové kódy. Pre používanie systému je potrebné získať povolenie od úradov, a platiť vysoký mesačný poplatok. Používajú ho najmä bezpečnostné zložky ako polícia. Môže byť tiež použitý na rôznych parkoviskách pri školách, a monitorovať vstup nebezpečnej osoby.

[11] https://www.elecrow.com/camera-module-with-color-cmos-qsxga-for-raspberry-pi-p-1388.html

Kapitola 4

Analýza a návrh riešení

V tejto kapitole som analyzoval súčasné riešenia, ktoré som popísal v kapitole 3, ich porovnanie, a porovnanie dostupných prostriedkov. Popísal som moje riešenia a prostriedky, ktoré som použil k ich implementácií. Na konci som zhrnul technické zadanie práce.

4.1 Analýza existujúcich riešení

Všetky opísané riešenia využili Raspberry trochu iným spôsobom a myslím, že všetky projekty použili Raspberry správne. Nič však nie je dokonalé, a každý systém má svoje obmedzenia a chyby. Pri niektorých projektoch to bola vysoká cena, ktorá by mohla ľudí odradiť. Iné projekty zase neboli príliš výnimočné, podobných alebo dokonca rovnakých riešení boli na internete desiatky. Každý systém má svoje chyby. Pri dlhom vývoji a horšej optimalizácií pri niektorých projektoch už nemusí všetko, čo ponúkajú, fungovať správne. Takým prípadom je napríklad 3.2.

Názov projektu	Dostupnosť zdrojového kódu	Využitie GPIO	Využitie WiFi modulu	cena v eurách
OpenLabTools mikroskop	Áno	Áno	Áno	zdrojový kód zdarma
RetroPie konzola	Áno	Môže, ale nemusí	Môže, ale nemusí	100-200
PiSound-HAT doska	Áno	Áno	Môže, ale nemusí	100-300
Pi MusicBox server	Áno	Nie	Áno	Zdarma
Wavepad-pohybové ovládanie	Áno	Áno	Nie	10-20
openALPR-rozpoznávanie ŠPZ	Nie	Nie	Nie	10-90 mesačne

Tabulka 4.1: Zhrnutie popísaných riešení

Všetky spomenuté riešenia majú svoje výhody i nevýhody. Ak je riešenie veľmi dobré, väčšinou býva nadmieru drahé pre obyčajných užívateľov. Ak je však riešenie zdarma, väčšinou nebýva veľmi prepracované a optimalizované. PiSound je riešenie, ktoré má najlepšie ohlasy a hodnotenia. Plne využíva všetky výhody, ktoré na trhu ponúka iba Raspberry

Pi a jemu podobný hardware. Z tohoto je teda jasné, že Raspberry má oproti konkurencií obrovskú výhodu vo:

- Veľkej podpore projektov od Raspberry Pi Foundation,
- Obrovskom množstve periférií, ktoré sa dajú k Raspberry pripojiť,
- Značnom množstve zdrojových kódov je open-source,
- Relatívne nízkej cene.

Úloh so spracovaním zvuku je na trhu obzvlášť málo. Ak tu nejaké riešenie je, je zvyčajne enormne drahé, alebo nedostupné. Spracovanie zvuku a signálov je stále veľmi nepreskúmaná oblasť informatiky, a preto je po takýchto riešeniach obrovský dopyt.

4.2 Porovnanie dostupného hardware

Raspberry má všetku funkcionalitu ako klasický stolový počítač, no rozdiel je vo výkone. Aj keď najnovšie modely Raspberry ponúkajú solídny procesor a operačnú pamäť, GPU je stále veľmi slabý najmä pre veľkosť čipu a problémom s chladením. To ale nie je hlavným účelom Raspberry, kvôli užívateľsky menej prístupnému systému nikdy nenahradí počítače.

Procesor (CPU)	1.4 GHz 64-bit quad-core ARM Cortex-A53
Video (GPU)	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz)
	OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS)
	MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)
Paměť (SDRAM)	1 GB (sdílená s GPU)
USB 2.0 porty	4 (přes zabudovaný pětiportový USB hub; jeden USB port vnitřně propojen s ethernet portem)
Video vstup	15-pinový MIPI konektor kamerového rozhraní (CSI)
Video výstup	HDMI (rev 1.3 & 1.4), 14 HDMI rozlišení od 640×350 do 1920×1200 plus různé PAL a NTSC standardy, kompozitní video (PAL a NTSC) via 3.5 mm TRRS jack sdílený s výstupem zvuku, MIPI konektor rozhraní displeje (DSI)
Zvukový vstup	Přes I2C rozhraní
Zvukový výstup	Analogový (přes 3.5mm jack), digitální (přes HDMI), I2S
Interní paměť	MicroSDHC, USB Boot Mode
Integrovaná síť	Gigabitový Ethernet skrze USB 2.0 + WiFi 802.11ac, Bluetooth 4.2 BLE a možnost PoE napájení skrze externí HAT modul
Nízkoúrovňové periferie	17× GPIO plus stejné funkce a HAT ID sběrnice
Jmenovitý výkon	300 mA (1,5 W) v průměru na volnoběh
	1,34 A (6,7 W) při maximálním zatížení (připojený monitor, klávesnice, myš a WiFi)

950kč

Procesor	
Řada procesoru	Intel Pentium
Model procesoru	Intel Pentium G5420 Coffee Lake
Frekvence procesoru	3,8 GHz (3 800 MHz)
Operacni pamet	
Intel Optane	Ne
Velikost operační paměti RAM	4 GB
Frekvence paměti	2 666 MHz
Typ paměti	DDR4
Výstupy	
USB 2.0	4 ×
USB 3.2 Gen 1 (USB 3.0)	6 ×
Grafická	VGA D-SUB, HDMI, DisplayPort
Další	LAN, COM port, Výstup na sluchátka / reproduktor
Optická mechanika	DVD

8400kč

Obrázek 4.1: porovnanie ceny obyčajného stolového PC a Raspberry

Na trhu je tiež obrovské množstvo hardvérov, ktoré sa snažia Raspberry konkurovať. Niektoré z nich sú úplne nové riešenia od známych značiek, medzi ktoré patrí napríklad `Asus` alebo `Arduino`. Iné sa len snažia okopírovať Raspberry za nižšiu cenu a kvalitu.

Názov	Procesor	GPU	Pamäť	cena v eurách
Onion Omega2Plus	580 MHz MIPS	nemá	128 MB	10
BBC micro:bit	ARM Cortex-M0	nemá	256KB Flash ROM, 16KB RAM	12
Rock64 Media Board	Rockchip RK3328 (4x Cortex-A53@ 1.5GHz)	Mali-450 MP2	1-4GB DDR3L, prázdny slot eMMC	22
Arduino Mega 2560	ATmega2560	nemá	256KB Flash ROM	30
Le Potato	Amlogic S905X (4X Cortex-A53 @ 1.5GHz)	Mali-450 MP2	1-2 GB DDR3 RAM, 8-64GB eMMC	33
Rock Pi 4 Model B	Rockchip RK3399	Mali T860MP4	1-4GB LPDDR4 @3200Mb/s	45
Banana Pi M64	Allwinner A64 (4x Cortex-A53 @ 1.2GHz)	Mali-400 MP2	2GB DDR3 RAM, 8-64GB eMMX	55
Asus Tinker Board S	Rockchip RK3288 (4x Cortex-A17 @ 1.8GHz)	Mali-T760	2GB LPDDR RAM	80
LattePanda	Intel Cherry Trail Z8350 Quad core 1.8GHz	Intel HD Graphics @200-500 Mhz	2GB DDR3L	105
Minnowboard Turbot Dual Ethernet	Inteal Atom E38 Series	Gen 7	2GB DDR3L 1067MT/s	200
BeagleBoard-X15	TI AM5728 2x1.5-GHz ARM Cortex-A15	Dual Core SGX544, 532 MHZ	2GB DDR3L RAM	210

Tabulka 4.2: Prehľad konkurenčného hardware

Aj napriek tomu, že dostupných hardvérov je obrovské množstvo, je Raspberry stále najpoužívanejšie. Je to tým, že na trhu sa objavilo ako prvé, a za roky vývoja si získalo obrovskú komunitnú základňu. Má tiež najlepšiu podporu pre začiatočníkov, ktorí s takýmto hardvérom začínajú, a cena Raspberry je jedna z najnižších z uvedených.

4.3 Rozvedenie cieľa

Na základe rozboru existujúcich riešení som sa rozhodol vytvoriť úlohy, ktoré by mali byť vhodné ako pre začiatočníkov, tak aj pre pokročilých. Sú úplne zdarma, s minimálnym využitým externých knižníc a prostriedkov.

- Prvou úlohou, ktorú som sa rozhodol spracovať, je framework pre jednoduchú prácu so zvukom. Jeho hlavným cieľom je vytvoriť prostredie, kde si ktokoľvek môže napísať vlastný filter. Môj program sa postará o vstupy a výstupy, ako aj o správne spracovanie a formátovanie dát. Pre jadro celého programu som sa rozhodol použiť programovací jazyk Python 3 a to z dôvodu, že poskytuje veľké množstvo knižníc pre jednoduchú prácu so zvukom i manipuláciu s dátami. Pre samotnú filtráciu bude použitý jazyk C kvôli jeho výpočetnej rýchlosti a správy pamäte. Filter si však bude môcť užívateľ napísať aj v Pythone.
- Druhou úlohou je domáci server s menším informačným systémom, ktorý bude získavať dáta od DHT11 senzoru, ktorý meria teplotu a vlhkosť. Cieľom tejto úlohy je vytvoriť jednoducho rozšíriteľný domáci server s informačným systémom, a ukázať výhody GPIO pinov. Na vytvorenie serveru som použil Appache z dôvodu, že je to jedno z najvýkonnejších riešení pre webový server. PHP používam hlavne na spojenie s databázou a to najmä z dôvodu, že je to jazyk, ktorý sa primárne používa na tvorbu webov. Použil som tiež technológie ako bootstrap, ktorý mi uľahčil prácu s návrhom rozhrania. Celý informačný systém stojí na frameworku nette.

Úlohy, ktoré som navrhol, sú určené na výuku a rovnako aj na demonštráciu Raspberry Pi. Zameriavajú sa na silné stránky a výhody, ktoré Raspberry ponúka, a v ktorých je lepšie ako iné platformy.

4.4 Technické zadanie

Na základe popisu z 4.3 som sa rozhodol vytvoriť úlohy podložené nasledujúcimi špecifikáciami. Úloha pre spracovanie zvuku má nasledujúce špecifikácie:

- Vstupom môžu byť wav súbory alebo zvuk z mikrofónu,
- Vstupné dáta musia byť v 16 bitovom Litlle-endian formáte, a môžu mať 1 alebo 2 kanály,
- Možné nastavenie vstupnej frekvencie a mikrofónu,
- Zvukové vstupy podporujú aj USB mikrofón,
- Užívateľské funkcie budú mať dostupný aktuálny a starý buffer spolu s inicializačnými údajmi,
- Užívateľské funkcie môžu byť napísané v jazyku C alebo v Pythone.

Vstupné dáta v užívateľských funkciách budú uložené vo formáte double, a to kvôli lepšiemu spracovaniu, a aby užívateľ mohol písať aj zložitejšie filtre. Pred zápisom výstupných dát do súboru budú však tieto hodnoty zaokrúhlené a pretypované na "int". To z dôvodu, že program pracuje iba s 16 bitovými hodnotami, pre uloženie double hodnoty by bolo potrebných až 32 bitov.

Úloha s meteostanicou má nasledujúce špecifikácie:

- Jednoducho rozšíriteľný informačný systém zobrazujúci dáta zo senzoru,
- Raspberry môže byť pripojené k lokálnej sieti,

- Obsahuje skript pre získavanie aktuálnych dát zo senzoru,
- Jednoduché zapojenie.
- Možnosť zapojenia nových senzorov

Úloha slúži primárne ako demonštračná. Užívateľ nemusí do ničoho zasahovať a nemusí písať žiadny kód. Úloha má ukázať prácu s GPIO a ďalšie možnosti využitia architektúry Raspberry.

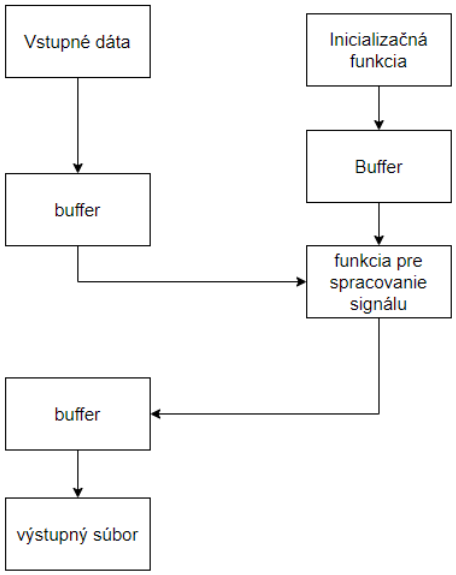
Kapitola 5

Implementácia úloh

V tejto kapitole som vysvetlil ako som postupoval pri práci na jednotlivých úlohách. Vysvetlil som kľúčové prvky programov a ako som sa vysporiadal s problémami, na ktoré som narazil pri riešení. Obsahom sú aj vývojové diagramy, popis úloh ktoré som implementoval a na konci kapitoly podrobne vysvetlené testovanie celého systému a jeho jednotlivých prvkov.

5.1 Ciel frameworku

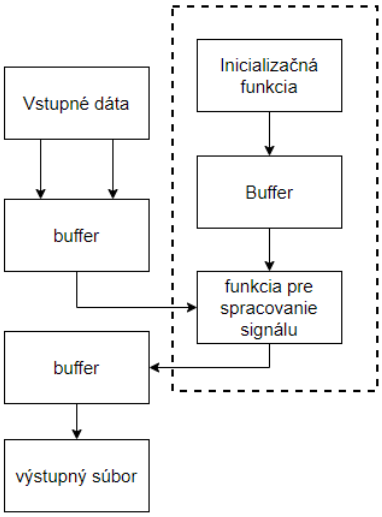
Na diagrame 5.2 je znázornený príklad použitia programu, kde užívateľ dokáže pomocou pripravených funkcií manipulovať so vstupnými dátami. Použiť sa dá rôzne. Povedzme, že máme súbor, ktorý obsahuje šum, alebo iné zvuky, ktoré chceme odstrániť. Užívateľ si tak môže napísať filter, ktorý tento šum odstraňuje a frameworku zadá vstupný súbor, prípadne iné parametre. Výstup sa jednoducho zapíše do výstupného súboru.



Obrázek 5.1: Základ použitia

Študent môže použiť inicializačnú funkciu, kde si môže napísať spracovanie vstupných parametrov, a tieto spracované údaje bude mať dostupné vo funkcií pre spracovanie signálu. Okrem týchto dát bude mať dostupné aj vstupné dáta. Veľkosť vstupných dát závisí od vzorkovacej frekvencie, ktorá určuje koľko vzoriek dát dostane za sekundu.

5.2 Príklad použitia



Obrázek 5.2: Príklad použitia

Užívateľ má vo funkcií dostupné aktuálne dáta získané zo vstupu a dáta zo starého bufferu. Buffre fungujú ako ping-pong algoritmus 5.3, takže funkcia pre spracovanie signálu stále pracuje. Takto spracované dáta sa uložia do výstupného súboru. Užívateľ má na výber, čiže filtre môže napísať v jazyku C alebo Pythone. V adresárovej štruktúre sú pripravené súbory, stačí v argumentoch programu zvoliť jazyk prepínačom `-c` pre jazyk C, alebo `-p` pre Python.

Povedzme, že chceme filtrovať vstupné dáta pomocou low-pass filtru, ktorý je popísaný rovnicou: $y(n) = x(n) + x(n-1)$. Je to ideálny filter, pri ktorom použijeme jazyk C.

```
/**
 * funkcia pre spracovanie vstupných zvukových dát
 * @param init hodnoty z inicializačnej funkcie alebo z dodaného súboru
 * @param init_len dĺžka poľa init
 * @param old_len dĺžka poľa v ktorých sú uložené staré dáta
 * @param in_len dĺžka pola v ktorom sú uložené aktuálne dáta
 * @param out pola výstupných hodnot, sem prídu upravené(filtrované) vszupné hodnoty
 * @param old_data dáta zo starého bufferu
 * @param vstup dáta z aktuálneho bufferu
 * @param tmp nesie dôležité metadata, vo funkcií nepoužívať
 */
void process(double *init,int init_len ,int old_len, int in_len, double *out, double *old_data, double *vstup,double *tmp)
{

        if(old_len > 0){
                out[0] = vstup[0] + old_data[old_len];
        }

|
        for (int i = 1; i < in_len; i++)
        {
                out[i] = vstup[i]+vstup[i-1];
        }
```

Obrázek 5.3: príklad ideálneho low-pass filtru

Po dokončení funkcie je potrebné preložiť ju pomocou prekladača gcc, a to s nasledujúcimi argumentami:

```
gcc -o nazov_funkcie.so -shared -fPIC -O2 nazov_funkcie.c
```

Obrázek 5.4: preklad funkcií v jazyku C

Po dokončení filtru stačí jednoducho spustiť framework so vstupnými dátami, napríklad zo súboru.

```
python3 Bachelor.py -b 35100 -v file -n input.wav -t stereo -f 10.0 11.2 -c
```

Obrázek 5.5: spustenie frameworku

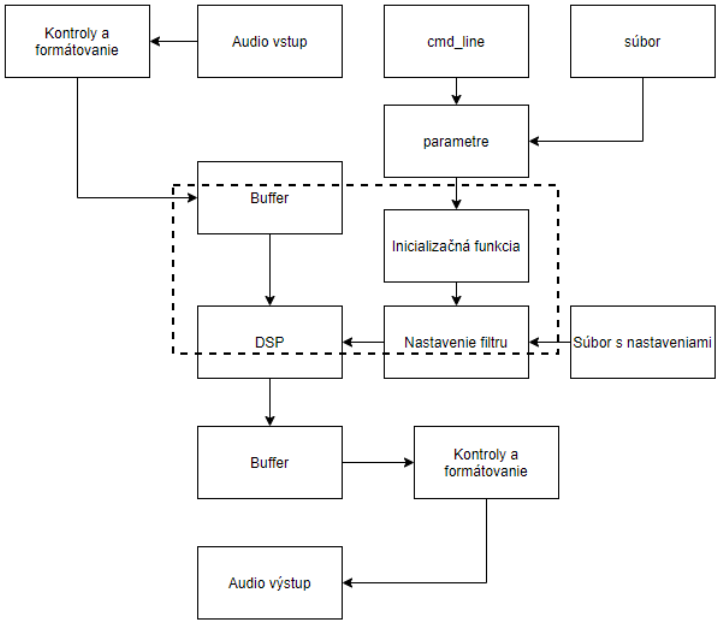
Užívateľ teda dostane funkcie, v ktorých si môže vytvoriť inicializačnú funkciu pre pomocné dáta a funkciu, kde môže spracovávať vstupné dáta v aktuálnom čase. Na diagrame 5.6 Je znázornená časť, ktorú spracováva užívateľ.

- -b: nastavuje počet framov, ktoré majú byť načítané do bufferu. Pri čítaní dát z mikrofónu odporúčam minimálne polovicu vzorkovacej frekvencie, aby sa dáta stihli spracovať.
- -v: výber, odkiaľ sa majú získavať vstupné dáta.
- -t: pri zvolení mikrofónu výber počtu zvukových kanálov.
- -n: pri zvolení súboru názov súboru z ktorého sa majú získavať dáta.
- -c: použitie jazyku C pre spracovanie užívateľských funkcii.
- -p: použitie Pythonu pre spracovanie užívateľských funkcii.
- -d: v prípade, že užívateľ nepotrebuje použiť inicializačnú funkciu, argument bude obsahovať názov súboru, v ktorom má uložené potrebné dáta.
- -f: V prípade, že užívateľ potrebuje použiť inicializačnú funkciu, zadá súbor, v ktorom sú uložené dáta pre funkciu. Prípade môže zadať hodnoty priamo do tohoto argumentu.
- -e: Pri použití mikrofónu, nastavenie frekvencie.
- -r: Pri použití mikrofónu, index zariadenia, ktoré sa má použiť na zaznamenávanie zvuku.

Užívateľ má obrovské množstvo parametrov, s ktorými môže program spustiť a uľahčiť si tak prácu.

5.3 Popis riešenia frameworku pre prácu so zvukom

Základom tejto úlohy je spracovanie dát zo vstupu a následné správne naformátovanie na výstup.



Obrázek 5.6: podrobný diagram logického návrhu programu

Spracovanie argumentov

Keďže program dovoľuje užívateľovi veľké množstvo nastavení priamo v argumentoch programu, tak som sa rozhodol, že si funkciu na spracovanie argumentov napíšem sám. Keďže niektoré argumenty sú povinné, a niektoré sú povinné len v prípade, že bol už zadaný iný argument, tak som sa rozhodol, že argumenty budem vkladať do poľa v konkrétnom poradí a argumenty, ktoré spolu súvisia kontrolovať už pri vkladaní. Niektoré argumenty môžu byť zadané v podobe rady čísel, alebo v podobe súboru, preto sú kontroly argumentov veľmi dôležité k tomu ako bude fungovať zvyšok programu.

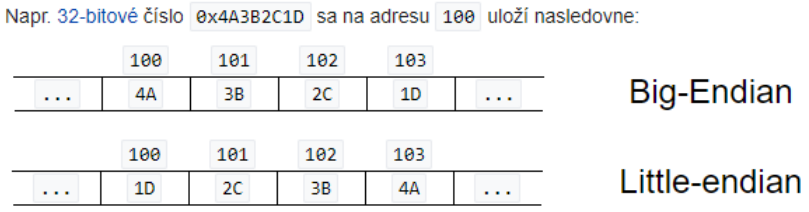
Konfigurácia vstupu a výstupu

Vstupom programu môže byť súbor formátu "wav", alebo vstup z mikrofónu v reálnom čase. V oboch prípadoch musí byť zvuk v 16 bitovom Little-endian formáte, a môže byť mono alebo stereo. Endianita je veľmi podstatná pre správne fungovanie celého programu. Keďže vstupom môžu byť 2 úplne rozdielne zdroje dát, najskôr sa z argumentov programu zistí, ktorý z nich to je. V prípade mikrofónu je použitá knižnica "pyaudio", v ktorej sa nakonfigurujú veci ako formát, počet kanálov a zariadenie, ktoré má byť brané ako zvukový vstup. Ako výstup sa potom vytvorí "wav"súbor, ktorý sa nakonfiguruje presne podľa rovnakých parametrov ako dostal vstup. Ak bude vstupom súbor, tak sa jednoducho vstupný súbor otvorí len pre čítanie, a s presne rovnakými parametrami sa otvorí aj výstupný súbor len pre zápis.

Endianita

Podstatou endianity je poradie, v akom sa bajty ukladajú pri vyjadrení rôznych čísel. Poznáme tak Big-endian, kde sa na pamäťové miesto s najnižšou adresou uloží najvýznamnejší bajt, a za ten sa ukladajú ostatné bajty až po najmenej významný. Little-endian funguje

presne naopak, na pamäťové miesto s najnižšou adresou uloží najmenej významný bajt a za ten uloží zvyšok. Majme teda nejaké vstupné zvukové 16 bitové dáta, napríklad FF08. Pri architektúre Little-endian sa dáta uložia ako 08 FF, a pri architektúre Big-endian ako FF 08, čo vyjadruje 2 úplne rozdielne čísla.



Obrázek 5.7: endianita

Inicializačná funkcia

Programu môže byť predaná sada parametrov pre inicializačnú funkciu a to priamo cez argumenty programu, alebo pomocou súboru, kde sú uložené. Tieto parametre budú konštantné po celú dobu programu, a pre ich zmenu bude potrebné znovu spustiť program s novými hodnotami. Parametre sa môžu predpracovávať, a to pomocou inicializačnej funkcie, ktorej poskytnem parametre, ktoré budú uložené v poli double hodnôt, a veľkosť tohoto poľa. Tu si bude môcť užívateľ z týchto parametrov vypočítať pomocou funkcie, ktorú si sám naprogramuje hodnoty, ktoré budú neskôr predané funkcii so samotným filtrom. Užívateľ však už môže mať tieto hodnoty spočítané v nejakom súbore, a tak je tu možnosť vôbec inicializačnú funkciu nepoužiť a programu rovno predať hodnoty, ktoré si sám spočítal.

Paralelný beh programu

Kvôli čo najväčšej rýchlosti a efektivite celý program beží na troch vláknach. Na jednom vlákne beží funkcia, ktorá sa stará o čítanie dát zo vstupu a ukladanie do bufferov. Na ďalšom vlákne beží predávanie dát do DSP funkcie, samotné vykonanie funkcie a následné uloženie do súboru. Posledné vlákno je to najdôležitejšie, na ňom totiž beží celý hlavný program, kde sú uložené všetky hodnoty a premenné. Prebieha na ňom inicializácia a nastavenie vstupov a výstupov. Poskytuje ostatným funkciám prístup k hodnotám, s ktorými pracujú ostatné vlákna, a čaká na vstup od užívateľa na ukončenie programu. Niektoré hodnoty sú kvôli jednoduchšiemu predávaniu nastavené ako globálne, aby každé vlákno malo aktuálnu hodnotu. Mohlo by tak dôjsť k nekonzistencií, ktorá by mohla byť vážnym problémom. Vďaka správnej synchronizácií je ale použitie globálnych premenných úplne bezpečné.

```
thread1 = threading.Thread(target = read, args=(frames_to_read,args[1]))
thread2 = threading.Thread(target = write, args=[chans])
thread1.start()
thread2.start()
```

Obrázek 5.8: práca s vláknami

Pre prácu s vláknami je použitá knižnica `threading`, ktorá umožňuje jednoduché spustenie niektorých funkcií na inom vlákne so zadanými argumentami.

Načítanie dát

O čítanie dát zo vstupu sa kvôli efektívnemu a rýchlemu spracovaniu stará funkcia, ktorá beží na samostatnom vlákne. Funkcia berie ako vstup počet framov [1], ktorý sa načíta v jednom cykle. Tento počet si užívateľ môže špecifikovať, má však isté špecifikácie, ktoré boli vysvetlené v 5.1. Ďalším argumentom je typ vstupných dát, od ktorého záleží, aká knižnica sa pre získanie dát použije. V prípade vstupného súboru sa na získanie dát použije knižnica `wave`, ktorá uloží dáta do bufferu[2] ako reťazec bajtov. V prípade mikrofónu sa použije vytvorený audio stream pomocou knižnice `pyaudio`, a rovnako vráti reťazec bajtov.

```
if inputType == 2:
    frames2 = inp.read(frames_to_read)
    if len(frames2) > 0:
        old_data2 = frames2
        tmp = 4
if inputType == 1:
    frames2 = inp.readframes(int(frames_to_read))
    old_data2 = frames2
    tmp = 4
```

Obrázek 5.9: načítanie dát

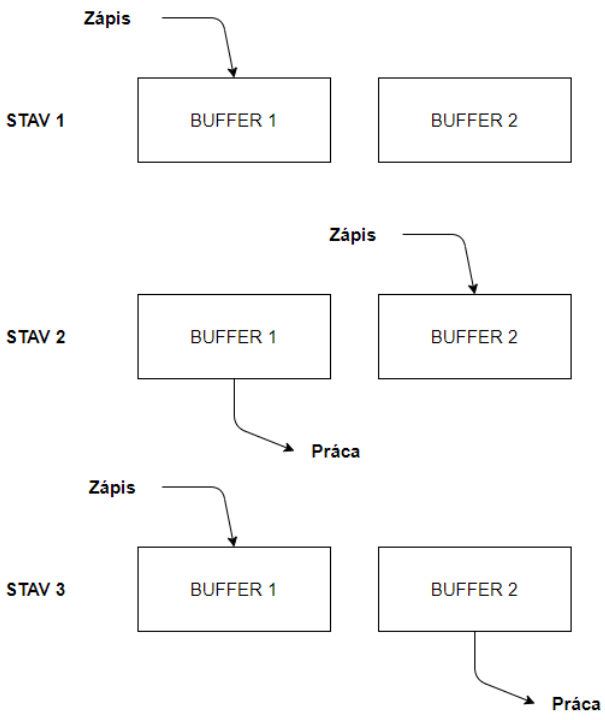
Pre výpočty však občas bude treba aj buffer s predchádzajúcimi vzorkami, a tak sa tiež ukladajú. Vlákna pre načítanie dát a pre prácu a ukladanie s dátami pracujú ako ping-pong algoritmus 5.3, a preto je potrebné používať vždy 2 (vlastne 4) buffre naraz. Pri čítaní dát zo súboru sa funkcia automaticky zastaví a dá užívateľovi vedieť pomocou výpisu, aby mohol zastaviť aj zvyšné vlákna a ukončiť tak program.

Ping-pong algoritmus

Základom tohoto algoritmu je použitie 2 buffrov, kde vždy jeden pracuje, a do druhého sa ukladajú dáta. V prípade aktuálneho programu to je ideálne využitie, kde sa do jedného bufferu načítajú vstupné dáta a okamžite sa posielajú na spracovanie do DSP funkcie. Zatiaľ čo sa čaká na spracovanie dát, do druhého bufferu sa už načítajú ďalšie dáta. Toto riešenie so sebou nesie pár výhod, ale zároveň aj pár nevýhod. Základnou výhodou je rýchlosť a efektivita, ktorú program ponúka, keďže beží na viacerých vláknach. Zároveň vždy vykonáva výpočty, a je tak veľmi efektívny a niekoľkonásobne rýchlejší.

Hlavnou nevýhodou je pretečenie vstupných dát pri čítaní z mikrofónu. Knižnica, ktorá je použitá na čítanie dát má svoj vlastný interný buffer, kde sa ukladajú dáta keď sa zrovna neukladajú do premennej. Tento buffer má však obmedzenú veľkosť, a ak majú vstupné dáta veľkú vzorkovaciu frekvenciu, program nemusí stíhať a buffer pretečie. To spôsobí okamžitý pád programu, dá sa tomu ale zabrániť menšou vstupnou vzorkovacou frekvenciou, alebo väčším buffrom pre čítanie vstupných dát. Ďalšou nevýhodou je použitie viacerých vlákien a vyššie vyťaženie procesoru, čo pri slabšom hardware, ako je Raspberry, nemusí byť optimálne, ak by bežal ešte nejaký iný program alebo proces. Poslednou menšou nevýhodou je možná desynchronizácia vlákien pri práci s globálnymi premennými.

[1] frame je najmenšia časť zvukového súboru, jej veľkosť závisí od počtu kanálov a formáte dát
[2] kus pamäte, kde sa dočasne ukladajú vstupné alebo výstupné dáta



Obrázek 5.10: princip ping-pong algoritmu

Užívateľské funkcie

Inicializačnú funkciu aj samotnú funkciu na spracovanie signálu si užívateľ môže napísať v jazyku C alebo Pythone. To hlavne z dôvodu, že každý jazyk ponúka niečo iné. Jazyk C je rýchlejší, ak ide o výpočty a prácu s pamäťou, a preto sa dokonale hodí do situácie, kde je rýchlosť výpočtu kľúčová. Na rozdiel od toho je Python jednoduchý, pracuje sa v ňom lepšie, a predávanie hodnôt je jednoduchšie. Zároveň zaberie menej pamäte, pretože aj hlavný program je písaný v Pythone. Komunikácia Pythonu a jazyka C prebieha hlavne pomocou modulov, čo je úplne bežná vec. Pre toto riešenie to však nebolo ono, keďže moduly sa používajú hlavne ako knižnice, kde celý kód už funguje, a Python len volá funkcie. V týchto moduloch sa pracuje so špeciálnymi objektami, a vyžadujú inicializáciu a wrapper [3]. Užívateľ by tak nemusel presne vedieť, ako s tým pracovať, a hlavný program by nemusel prijať dáta, ktoré dostane od funkcii. Použitá je preto knižnica `ctypes`, ktorá importuje do Pythonu dátové typy, ktoré používa jazyk C. Stačí tak len kód z jazyka C preložiť cez "gcc"do súboru s príponou `.so`.

Najskôr sa do premennej uloží cesta k preloženému súboru. Táto cesta sa následne overí a ak takýto súbor existuje, tak sa funkcia, ktorú zo súboru chceme, uloží do Python objektu. Môžeme mu následne nastaviť dátový typ, ktorý funkcia vráti, a kdekoľvek v kóde túto funkciu zavolať.

Pri volaní funkcie sa musia argumenty pretypovať na dátové typy, s ktorými pracuje jazyk C. Na druhom riadku v obrázku 5.12 je vytvorené pole `double` hodnôt o veľkosti `n`, a je naplnené inicializačnými hodnotami z listu. Na riadku 3 je vytvorená Python premenná

[3] používa sa pri zložitejších funkciách a vykonáva úlohy ako: rezervovanie zdrojov, kontrolovanie podmienok ...

```
lib_path = '/home/student/BAchelor/basic_function_linux.so'

try:
    basic_function_lib = CDLL(lib_path)
except:
    print('OS %s not recognized' % (sys.platform))

python_init = basic_function_lib.init
python_init.restype = int
```

Obrázek 5.11: inicializácia funkcie jazyka C

```
n = len(vstup)
c_arr_in = (c_double * n)(*vstup)
lenght = c_int(0)
pi = pointer(lenght)

init_values = python_init(c_int(n),pi,c_arr_in)

c_pointer_len = lenght.value
```

Obrázek 5.12: volanie funkcie jazyka C

s hodnotou 0, na riadku 4 je na ňu vytvorený ukazateľ, ktorý je predaný funkcii jazyka C, a ten s ním pracuje ako s normálnym ukazateľom. Po ukončení funkcie sa získa hodnota, ktorá leží na mieste kam ukazuje vytvorený ukazateľ.

```
#include <stdlib.h>
#include <stdio.h>

int* init(int n,int *len ,double *array_in)
{
    int lenght = 10
    double *test = (double *)malloc(lenght * sizeof (double));
    return test;
}
```

Obrázek 5.13: funkcia jazyka C

V užívateľskom programe potom môže užívateľ pracovať bez akýchkoľvek obmedzení. V inicializačnej funkcii si alokuje potrebné miesto a Pythonu vráti hodnotu, ktorá reprezentuje miesto v pamäti. Táto hodnota bude ďalej predaná DSP funkcii ako ukazateľ.
Pri použití Pythonu sú obe funkcie importované, žiadne hodnoty sa nemusia pretypovať a vďaka tomu, že je Python tak vysokoúrovňový jazyk, všetky funkcie berú a vracajú o dosť menej argumentov.

DSP spracovanie

Predtým, než môžu byť dáta poslané užívateľskej funkcii na spracovanie, sa musia pretypovať z reťazca bajtov na pole hodnôt, ktoré bajty reprezentujú. K tomuto je použitá knižnica `struct`.

K tomu, aby boli dáta pretypované správne, je potrebné vedieť ich formát, endianitu a dĺžku. Z tohoto dôvodu program počíta s 16 bitovými little endian hodnotami. Ich dĺžka sa dá jednoducho vypočítať, keďže sú dáta uložené ako string bajtov, ich dĺžka sa dá vypočítať ako `dĺžka_bufferu/2`. Dáta sa môžu uložiť do listu ako celočíselné hodnoty. Aktuálne a staré dáta prídu v takejto podobe do užívateľskej funkcie. V závislosti od toho, či užívateľ používa funkcie jazyka C alebo Pythonu, sa pokračuje v spracovaní ďalej. Pri použití Pythonu sa všetky potrebné dáta pošlú funkcii, a tá vráti upravené dáta, ktoré sa zapíšu

```
unpstr = '<{0}h'.format(data_len)
actual_data = list(struct.unpack(unpstr, data))
```

Obrázek 5.14: pretypovanie

do výstupného súboru. Pri použití funkcie jazyka C sa hodnoty musia pretypovať do C-čkových dátových typov. Počet vstupných argumentov bude pri funkciách v C väčší, pretože sa budú musieť predávať aj veľkosti polí. Užívateľská funkcia pre spracovanie signálu má implementované 3 funkcie, ktoré uľahčia prácu s dátami. Sú to funkcie na získanie dát len z pravého alebo ľavého zvukového kanálu, a funkcia na spojenie týchto dát. Užívateľ tak môže pri stereo zvuku upraviť len jeden konkrétny kanál.

```
void returnRight(double data[],int len,double vystup[]){

    int j = 0;
    for (int i = 1; i < len; i += 2, j++)
    {
        vystup[j] = data[i];
    }

}

void joinChannels(double data_left[],double data_right[],int len,double vystup[]){

    int j = 0;
    for (int i = 0; i < len; i += 2, j++)
    {
        vystup[i] = data_left[j];
        vystup[i+1] = data_right[j];
    }
}
```

Obrázek 5.15: práca so stereo zvukom

Dáta sú totiž v stereo súbore uložené ako 16 bitové dvojice, kde prvých (ľavých) 16 bitov je ľavý zvukový kanál, a druhých (pravých) 16 bitov je pravý zvukový kanál. Užívateľ bude mať tieto funkcie pripravené vždy, použiť ich však môže len ak sú vstupné dáta stereo. Pri použití na mono vstup nastane neočakávané chovanie a výsledné dáta budu deformované.

Zápis dát

Po ukončení funkcie pre spracovanie dát v jazyku C sa dáta pretypujú a uložia do listu. Tieto dáta sa musia pretypovať znova na vhodný výstupný formát. K tomu sa použije knižnica struct, a všetky hodnoty v získanom liste sa preformátujú na string bajtov.

```
wr_data = b""

for x in range(len(end_data)):
    wr_data+=(struct.pack("<h",int(end_data[x])))
```

Obrázek 5.16: pretypovanie na bajty

Tieto dáta sa pomocou knižnice wave uložia do výstupného súboru.

Ukončenie programu

Ukončiť program musí užívateľ a to vstupom z klávesnice. Po ukončení sa pošle signál všetkým vláknam, aby dokončili aktuálny cyklus a okamžite sa ukončili. Keď sa všetky vlákna ukončia, program pokračuje, uvoľní všetko miesto a zatvorí zvukový kanál. Pri použití užívateľských funkcii v jazyku C sa zavolá deinicializačná funkcia, ktorá uvoľní všetko miesto v pamäti.

```
x = input("Stlačte 'e' pre ukončenie a potvrdte pomocou 'enter':")
if x == "e":
    exit = 1
    thread1.join()
    thread2.join()

    if args[1] == 2:
        inp.stop_stream()
        inp.close()
        audio.terminate()

    if c_pointer != 0:
        deInit(c_pointer)
```

Obrázek 5.17: ukončenie programu

Pomocné skripty a štruktúra

Raspberry je veľmi špeciálny hardware, a práca s ním nie je jednoduchá. Pripojené periférie nemusia byť správne nakonfigurované, a Raspberry ich vôbec nemusí podporovať. Môžu byť obrovské problémy s ovládačmi a zariadenia nemusia pracovať tak, ako na iných zariadeniach, alebo podľa špecifikácie. K programu sú pridané aj nejaké pomocné skripty a príkazy, ktoré pomôžu správne nakonfigurovať zvukovú kartu a mikrofón.

```
import pyaudio
p = pyaudio.PyAudio()
for i in range(p.get_device_count()):
  dev = p.get_device_info_by_index(i)
  print((i,dev['name'],dev['maxInputChannels']))
```

Obrázek 5.18: Výpis zariadení

Skript 5.18 vypíše všetky zvukové vstupy, ktoré sú k Raspberry pripojené. Dokáže zistiť ich názov a počet podporovaných kanálov.

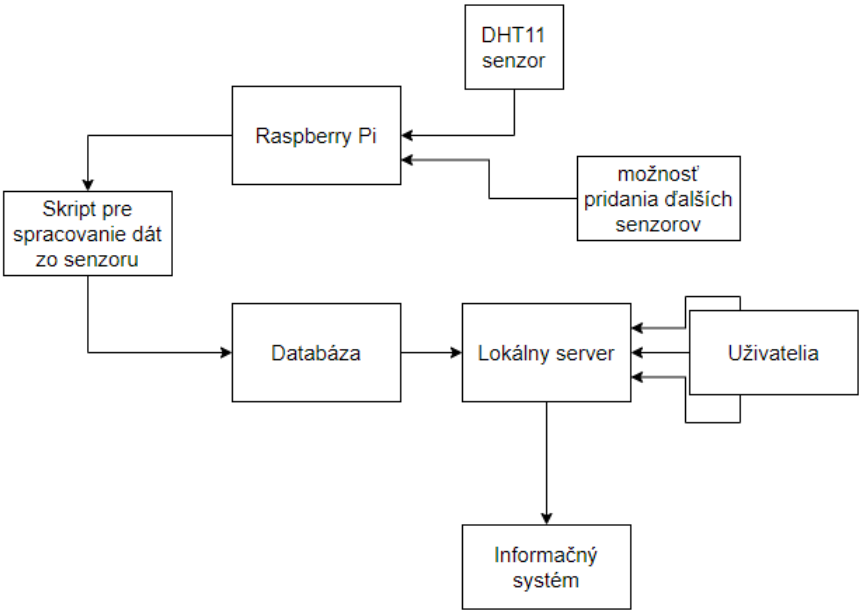
```
pi@raspberrypi:~ $ arecord -f dat -r 30000 -D hw:1,0 -d 5 -c 1 test11.wav
```

Obrázek 5.19: test mikrofónu

Pomocou príkazu 5.19 sa dokáže otestovať mikrofón, a prípadne zistiť akú vzorkovaciu frekvenciu podporuje. Keď sa do argumentu `-r` zadá frekvencia, ktorú nepodporuje, tak sa automaticky nastaví na najbližšiu, ktorú podporuje.

5.4 Popis riešenia systému s meteo stanicou

V tejto úlohe je postup prípravy Raspberry Pi a implementácia skriptu, ktorý získa informácie od DHT11 senzoru cez GPIO piny. Tieto informácie následne ukladá do databáze, ktorá bude prepojená s lokálnym serverom, na ktorom beží informačný systém, a ten zobrazuje dáta.



Obrázek 5.20: Diagram funkcionality

Úloha má demonštrovať výhody GPIO pinov, ktoré Raspberry poskytuje, a pripraviť funkčný základ informačného systému, ktorý sa môže ďalej rozšíriť. Užívateľ bude schopný vidieť v informačnom systéme aktuálne dáta zo senzoru zoradené v tabuľke podľa času. Bude mať aj možnosť si upraviť svoj účet a pridať nové funkcie.

Konfigurácia serveru

Pre konfiguráciu serveru je použitý Apache server kvôli stabilite a bezproblémového fungovania na Linuxovom systéme. Apache sa dá stiahnuť a nainštalovať priamo na oficiálnych stránkach, alebo pomocou príkazu 5.21.

```
pi@raspberrypi:~ $ sudo apt install apache2 -y
```

Obrázek 5.21: Stiahnutie balíčku apache2

Po úspešnej inštalácií je potrebné upraviť konfiguračný súbor `/etc/apache2/sites-enabled/000-def`
Súbor je potrebné si otvoriť ako správca a do kódu pridať povolenie `AllowOverride All`, čím sa povolí používanie všetkých typov príkazov v súbore `.htaccess`.

Htaccess je jeden z najstarších a najsilnejších konfiguračných súborov, kde sú nastavenia a správa prístupu pre HTTP protokol. Povolí konfigurovať server a toto povolenie je veľmi dôležité pre bezproblémový beh nette frameworku, na ktorom bude stáť informačný systém. Nakoniec je potrebné reštartovať službu apache, po čom je server pripravený a funkčný.

```
<Directory /var/www/html>
        AllowOverride All
</Directory>
```

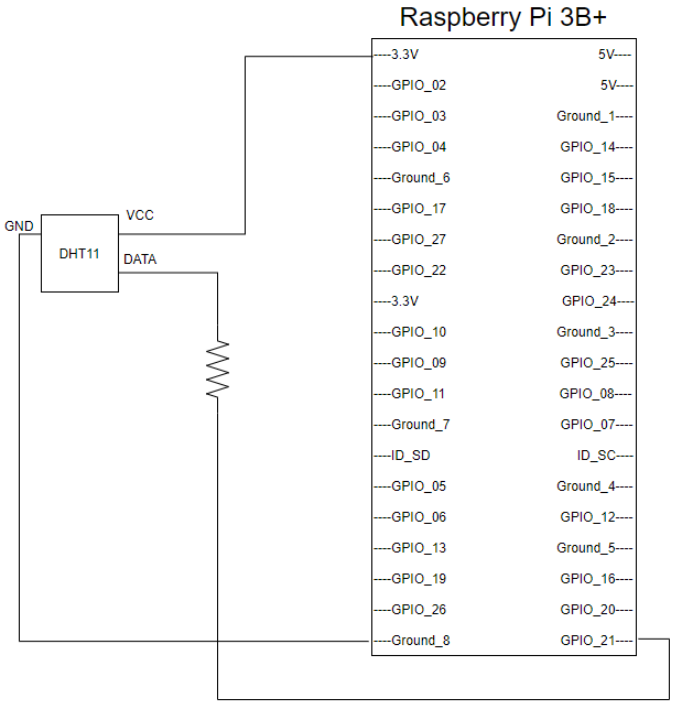
Obrázek 5.22: konfiguračný súbor 000-default

```
pi@raspberrypi:~ $ sudo service apache2 reload
```

Obrázek 5.23: Reštart služby

Meranie teploty a vlhkosti

Pre meranie teploty a vlhkosti je použitý DHT11 senzor. Senzor meria vlhkosť od 20% do 50% s presnosťou na 5%. Teplotu meria v rozmedzí 0-50 °C s odchýlkou 2 °C. Je k Raspberry pripojený pomocou GPIO pinov podľa diagramu 5.24.



Obrázek 5.24: Diagram zapojenia DHT11 senzoru

Po správnom zapojení senzoru je potrebné získať dáta, ktoré senzor zaznamená. K tomu je použitý Python a knižnica s názvom `Adafruit_DHT`. Adafruit je organizácia, ktorá ponúka obrovské množstvo hardware a knižnice na prácu s ním.

Práca s knižnicou je veľmi jednoduchá a intuitívna. Užívateľ si môže jednoducho pripojiť vlastné senzory, alebo iný hardware cez GPIO piny, a ovládať ho. Najskôr sa do premennej uloží typ senzoru a k akému dátovému pinu je pripojený. Následne do premenných na teplotu a vlhkosť budú uložené hodnoty získané zo senzoru. Tento skript môže bežať na pozadí, a každú pol hodinu zaznamená aktuálnu teplotu a vlhkosť, a uloží ju do databáze.

```
sensor = Adafruit_DHT.DHT11

gpio = 21

vlhkost, teplota = Adafruit_DHT.read_retry(sensor,gpio)
```

Obrázek 5.25: získanie dát zo senzoru

Databáza

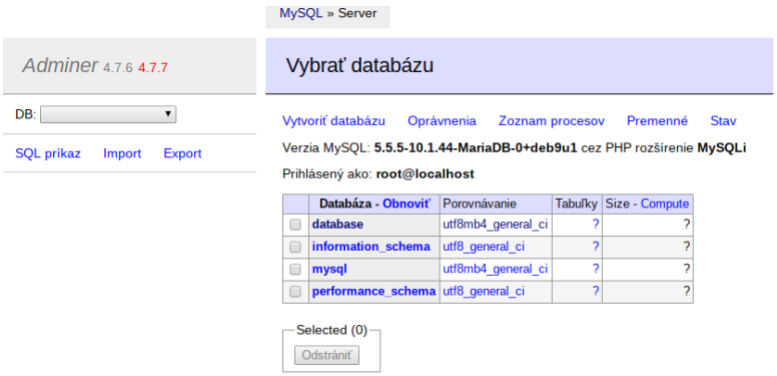
Pre vytvorenie databáze je použitá `MariaDB`. Je veľmi podobná MySQL, čo je asi najpoužívanejšie riešenie pre databázu. Obe vytvoril ten istý človek, `Michael Widenius`, ale MySQL patrí pod značku Oracle.

Ako prvú vec je potrebné databázu stiahnuť a nainštalovať na Raspberry, a to pomocou príkazu:

```
pi@raspberrypi:~ $ sudo apt-get install mariadb-server
```

Obrázek 5.26: inštalácia databáze

Počas inštalácie sa databáza sama nakonfiguruje. Teraz je možné vytvoriť novú databázu a tabuľky v nej. Pre prístup k databáze je použitý nástroj `adminer`. Je to jediný PHP súbor, ktorý sa jednoducho vloží do zložky s webom, a pristúpiť k nemu sa dá webovým rozhraním na adrese `localhost/adminer.php`. Má veľmi jednoduché a intuitívne grafické rozhranie, kde sa dá vytvoriť nová databáza a tabuľky v nej. Dajú sa tu vytvoriť aj otázky, pohľady, obmedzenia a aj odkazy na tabuľky pomocou cudzích kľúčov.



Obrázek 5.27: rozhranie admineru

Po vytvorení databáze je ďalším krokom uložiť do nej dáta získané zo senzoru. Použitý je rovnaký Python skript, ktorý získava dáta z DHT11 senzoru. Je potrebné však pridať knižnicu na prácu s databázou, a tou je `mysql.connector`. Najskôr je potrebné vytvoriť databázový objekt v Pythone s prístupom k vytvorenej databáze. Následne vytvoriť kurzor a vložiť do tabuľky namerané hodnoty.

Informačný systém a zobrazenie výsledkov

Informačný systém je posledná časť úlohy. Použitý je open-source PHP framework Nette. Nette pracuje podľa návrhového vzoru MVC, ktorý celú aplikáciu rozdeľuje do 3 vrstiev.

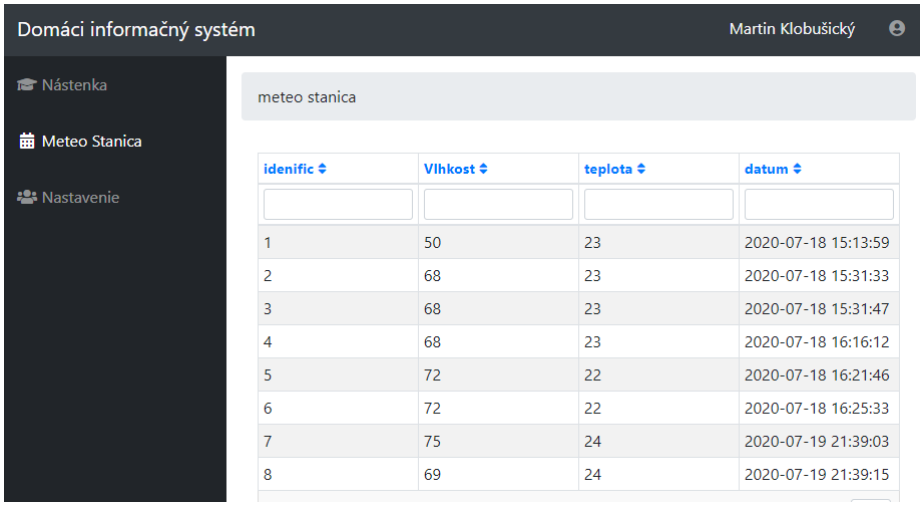
```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="123456",
  database="database"
)

mycursor = mydb.cursor()

if humidity is not None and temperature is not None:
    mycursor.execute("INSERT INTO dht11(vlhkost,teplota,datum) VALUES ({0},{1},NOW())".format(humidity, temperature))
    mydb.commit()
else:
    print('Failed to get reading. Try again!')
```

Obrázek 5.28: skript pre uloženie dát do databáze

Prvou je dátový model, ten sa stará o to, ako a kde sú dáta uložené. Druhou je aplikačná logika, ktorá sa stará o volanie a používanie dát z dátového modelu, je to srdce celej aplikácie. Tretia vrstva je užívateľské rozhranie, to sú všetky veci, ktoré má užívateľ k dispozícií. Nette používa šablónovací systém Latte, ten sa stará o vykresľovanie. Pre predávanie dát medzi dátovou vrstvou a užívateľským rozhraním Nette používa takzvaný Presenter.



Obrázek 5.29: rozhranie informačného systému

Užívateľ má tak pripravené rozhranie pre prihlásenie do systému. Po prihlásení má k dispozícií základné nastavenia účtu, kde sa dá zmeniť meno, heslo a email. Okrem toho má prístup k dátam, ktoré zaznamenáva senzor. Rozhranie je jednoduché a minimalistické z dôvodu jednoduchého rozšírenia.

5.5 Testovanie

Testovanie [10] je veľmi dôležitá časť vývoja prakticky čohokoľvek. Testovanie by malo tvoriť aspoň 30% vývoja aplikácie, a malo by v ideálnom prípade odhaliť všetky chyby a neočakávané prípady, ktoré vyvíjaný software môže mať. Pomocou testovania by sa mali odhaliť aj neoptimalizované časti, ktoré sa dajú zefektívniť. V neposlednom rade by sa malo vďaka spätnej väzbe od užívateľov zlepšiť užívateľské rozhranie. Testovať by sa teda mala:

- Funkčnosť programu,

- Užívateľské rozhranie,
- Optimalizácia,
- Stabilita.

Samozrejme, nič nie je dokonalé a tak ani testy neodhalia úplne všetky chyby, ale čím lepšie a dlhšie testovanie, tým viac chýb sa odhalí a opraví. Sú rôzne druhy testov, niektoré testy sa robia automaticky už počas vývoja, a iné až po tom, ako je vytvorený celý systém. Keď je systém hotový, tak sa poskytne skupine užívateľov na testovanie. Takéto testovanie môže trvať niekoľko mesiacov, než sa systém začne používať v reálnych podmienkach. V rámci bakalárskej práce som ale musel rozsah testovania zredukovať, systém som testoval v dvoch fázach.

Unit testing

Pri oboch úlohách sa systém rozdelil na menšie časti, ktoré boli implementované samostatne. Pri úlohe so zvukom to bolo:

- Spracovanie argumentov,
- Konfigurácia vstupu a získanie vstupných dát,
- Pretypovanie a kontroly,
- Príprava užívateľských funkcií a predanie potrebných dát,
- Pretypovanie a zápis výstupných dát do súboru,
- Uvoľnenie pamäte a ukončenie programu.

Ako prvé bolo implementované a testované spracovanie argumentov. Následne boli testované rôzne knižnice na získanie vstupných dát, či už z mikrofónu, alebo zo súboru. Podstatnú časť testovania tvorilo prepojenie Pythonu a jazyku C.

Pri úlohe s meteo stanicou sa systém rozdelil na:

- Nastavenie serveru,
- Príprava databáze,
- Implementácia informačného systému,
- Zapojenie a získanie dát zo senzoru,
- Uloženie dát do databáze.

Pri týchto testoch sa po implementácií jednotlivých častí každá časť otestovala samostatne. Následne sa nasadila do zvyšku systému a testovala sa kompatibilita a funkčnosť celého systému.

Keď bol celý systém hotový, tak sa otestoval ako celok. Boli testované rôzne vstupy a rôzne typy spracovania dát. Pri úlohe so zvukom boli testované jednoduché filtre a práca s každým kanálom zvlášť. Boli testované užívateľské funkcie programované v jazyku C i Pythone. Obe fungovali bezchybne, jazyk C však podával výsledky o niekoľko percent rýchlejšie. Pri testovaní boli odhalené problémy, ktoré má Raspberry s pripojením externej zvukovej karty, ktorá nebola pôvodom od Raspberry.

Testovanie celého systému

Pre testovanie použiteľnosti celého systému som zvolil jednoduchú metódu, pri ktorej som si vybral jedného spolužiaka, ktorý má prehľad v problematike, ku ktorej sa vzťahujú úlohy, a jedného rodinného príslušníka, ktorý má len základné znalosti.

Pri testovaní úlohy pre prácu so zvukom dostali užívatelia pripravenú celú štruktúru programu s plne nakonfigurovaným hardware pre použite na mojom Raspberry. Vzhľadom na nenáročnosť testovania nie sú pokryté úplne všetky možnosti, a tak možno neodhalili všetky chyby. Cieľom testovania je zistiť, ako na zadané úlohy bude reagovať študent, ktorý sa v nich vyzná a niekto, kto má len slabé základy. Na základe pozorovania by sa následne systém mal upraviť. Pri úlohách som užívateľa sledoval a odpovedal som na prípadné otázky.

- Dodať programu dáta pomocou súboru.
- Dodať programu dáta pomocou mikrofónu.
- Predať inicializačnej funkcií údaje.
- Predať programu už inicializované vlastné dáta.
- Implementovať inicializačnú funkciu.
- Implementovať jednoduchý low-pass filter.
- Odstrániť zvukové dáta z jedného kanálu.
- Pre implementáciu spracovania použiť oba podporované jazyky.
- Ukončenie programu.

Študent nemal žiadny problém dodať programu dáta rôznym spôsobom, pochopil význam inicializačnej funkcie a aj spôsoby predávania dát. Pri implementácií funkcií v Pythone taktiež nemal žiadny väčší problém. Úlohy dokázal splniť, potreboval však vedieť vzorec low-pass filtru. Pri implementácií funkcií v jazyku C bol menší problém, no úlohu zvládol.
Rodinný príslušník dokázal pochopiť, čo má program robiť a ako funguje, vzhľadom na obmedzené znalosti však nepoužil inicializačnú funkciu. Po chvíli skúšania a sledovania rôznych výsledkov dokázal splniť úlohy v Pythone, v jazyku C už však nie.

Z týchto pozorovaní som zistil, že pre ľudí je jednoduchšie použitie Pythonu. Po rozhovore s testovanými subjektmi som zistil, že je to pre nich o dosť prehľadnejšie a pohodlnejšie. Vďaka tomu som sa rozhodol do užívateľskej funkcie v jazyku C pridať podrobný komentár

ako funkcia funguje.

Pri úlohe s meteostanicou som použil rovnaké testovacie subjekty. Úloha je hlavne demonštračná, a tak úlohy neboli veľmi náročné. Išlo mi hlavne o test informačného systému. Užívatelia dostali podobné podmienky ako pri prvej úlohe, a ich úlohou bolo:

- Zapojiť DHT11 senzor podľa schémy,
- Spustiť skript pre získavanie dát,
- Prihlásiť sa v IS,
- Zobraziť dáta namerané senzorom,
- Upraviť svoj účet,
- Prepojiť senzor do iného dátového pinu, a upraviť skript na získanie dát.

Obaja užívatelia dokázali splniť zadané úlohy takmer bez problémov. Ich spätnou väzbou boli hlavne menšie úpravy informačného systému. Obaja užívatelia potvrdili, že by zvládli prípadné rozšírenie informačného systému o ďalšie prvky. Týmto som testovanie uzavrel ako úspešné.

Kapitola 6

Záver

Cieľom práce bolo vytvoriť sadu úloh, ktoré by začiatočníkom bližšie priblížili Raspberry Pi, a ukázali im výhody a nevýhody použitia. Tento cieľ sa mi podarilo úspešne splniť, a vytvoril som 2 úlohy, ktoré demonštrujú možnosti Raspberry Pi vo výuke.

Prvou úlohou bolo naštudovanie literatúry o Raspberry Pi, túto úlohu som splnil a poznatky sú zhrnuté v kapitole 2. Ďalej som navrhol a popísal spôsob implementácie modelových úloh, toto popisujem v kapitole 4. Následne som implementoval a podrobne popísal modelové úlohy, ktoré sú zhrnuté v kapitole 5. Nakoniec som zosumarizoval vlastnosti a možnosti pokračovania práce, tie sú uvedené tu, v závere.

Úlohy sú určené na edukatívne účely. V prvej úlohe môže užívateľ upravovať vstupné 16-bitové zvukové dáta. Program dokáže pracovať so vstupom z mikrofónu, alebo zo súboru, a podporuje až 2 zvukové kanály. Užívateľské funkcie môžu byť napísané v jazyku C alebo Pythone. Druhou úlohou je domáci informačný systém, ktorý zobrazuje dáta zo senzoru na meranie teploty a vlhkosti. Systém je pripravený na rozšírenie.

Pri implementácií som zistil, že Raspberry Pi má problém pracovať s niektorými perifériami. Pri jeho používaní by som teda radšej využíval iba hardware, ktorý sa dá zakúpiť špeciálne pre Raspberry. Vzhľadom na to nefungovali správne ani niektoré knižnice, hlavne pri práci s mikrofónom, kde bol problém so vzorkovacou frekvenciou. Pre meteostanicu by som zvolil iné meracie čidlo, ktoré by bolo presnejšie. Vďaka všetkým ťažkostiam a prekážkam som sa toho dozvedel veľa o Raspberry a o tom, ako funguje spracovanie zvuku a ukladanie zvukových dát.

V budúcnosti by som rád vylepšil úlohu so zvukom, kde by sa všetky potrebné nastavenia dodávali pomocou konfiguračného súboru. Program by tiež mohol podporovať viac formátov alebo zvukových kanálov. Server môže byť jednoducho rozšíriteľný o ďalšie funkcie, napríklad ovládanie svetiel cez wi-fi, chytrá domácnosť, rozvrh alebo iné. Prípadne by sa mohli dopracovať ďalšie úlohy, ako spracovanie obrazu, alebo riadenie robotických zariadení.

Literatura

[1] *Raspberry Pi Documentation* [online]. Raspberry Pi Foundation, United Kingdom, 2012 [cit. 2019-10-02]. Dostupné z: https://www.raspberrypi.org/documentation/.

[2] *Raspberry Pi Compute Module 3+* [online]. Raspberry Pi Trading Ltd, United States, 2019 [cit. 2019-07-23]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf.

[3] Halfacree, G. *Raspberry Pi Beginner's Guide* [online]. Raspberry Pi Trading Ltd, United States, 2018 [cit. 2019-07-23]. Dostupné z: https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf.

[4] Harrington, W. *Learning Raspbian*. Packt Publishing, United Kingdom, 2015. ISBN 978-1784392192.

[5] Hart Davis, G. *Deploying Raspberry Pi in the Classroom*. Apress, United States, 2016. ISBN 978-1-4842-2303-1.

[6] Loyse, G. *Raspberry-pi Documentation* [online]. Raspberry Pi Trading Ltd, United States, 2017 [cit. 2019-07-23]. Dostupné z: https://readthedocs.org/projects/raspberry-pi-intro/downloads/pdf/latest/.

[7] McManus, S. *Raspberry Pi For Dummies*. For Dummies, United States, 2017. ISBN 978-1-119-41200-7.

[8] Monk, S. *Programming the Raspberry Pi: Getting Started with Python*. McGraw-Hill Companies, United States, 2012. ISBN 978-0-07-180784-5.

[9] Negus, C. *Linux Bible*. 8. vyd. Wiley Publishing, United States, 2012. ISBN 978-1-118-21854-9.

[10] Patton, R. *Software Testing*. Sams, United States, 2001. ISBN 978-0672327988.

[11] Quinlan, O. *Embedding Picademy learning in schools* [online]. Raspberry Pi Foundation, United Kingdom, 2017 [cit. 2019-07-23]. Dostupné z: https://www.raspberrypi.org/app/uploads/2018/08/Embedding-Picademy-learning-in-schools.pdf.

[12] Robinson, A. *Raspberry Pi Hardware Projects*. Wiley Publishing, United States, 2013. ISBN 978-1-118-58894-9.