České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství

Hlasem ovládaný robot

Voice controlled robot

BAKALÁŘSKÁ PRÁCE

Vypracoval: Michal Kusý
Vedoucí práce: Ing. Josef Nový, Ph.D.
Rok: 2022

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství Akademický rok 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Michal Kusý**
Studijní program:	**Aplikace přírodních věd**
Obor:	**Aplikace softwarového inženýrství**
Název práce česky:	**Hlasem ovládaný robot**
Název práce anglicky:	**Voice controlled robot**

Pokyny pro vypracování:

1. Nastudujte problematiku rozponávání hlasu.
2. Seznamte se s platformou NVIDIA Jetson a Jetbot.
3. Navrhněte ovládací software robota využívající řízení hlasem.
4. Implementujte navržený software.
5. Implementovaný systém otestujte.

Doporučená literatura:

[1] L. Joseph, *Learning Robotics Using Python.* Packt Publishing, 2015. ISBN 1783287543.

[2] A. Kurniawan, *Getting Started with NVIDIA Jetson Nano.* PE Press, 2019.

[3] F. Chollet, *Deep learning v jazyku Python: knihovny Keras, Tensorflow.* Grada Publishing, a.s., 2019. ISBN 8024731002.

Jméno a pracoviště vedoucího práce:

Ing. Josef Nový
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta:

–

.....
vedoucí práce

Datum zadání bakalářské práce: 15. 10. 2020

Termín odevzdání bakalářské práce: 7. 7. 2021

Doba platnosti zadání je dva roky od data zadání.

.....
garant oboru

.....
vedoucí katedry

.....
děkan

V Praze dne 16.10.2020

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Děčíně dne ....

Michal Kusý

Poděkování

Děkuji Ing. Josefu Novému za zapůjčení robota a sehnání mikrofonu pro něj, aby se tato práce dala uskutečnit.

Michal Kusý

Název práce:
Hlasem ovládaný robot

Autor: Michal Kusý

Studijní program: Aplikace přírodních věd
Obor: Aplikace softwarového inženýrství
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Josef Nový, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant:

Abstrakt: Tato práce se zabývá hlasovým ovládáním pojízdného robota. Cílem práce bylo navrhnout ovládací software robota, tak aby poslouchal hlasové příkazy a pohyboval se řečeným směrem. Ovládací software robota byl vytvořen v Jupyter notebooku s pomocí neuronové sítě od Google API a bash skriptu na nahrávání zvuku z mikrofonu. Jak funguje zmíněná síť je popsáno v teoretické části práce. Ovládací software podporuje základní směrové příkazy, které jsou zmíněny v závěru. V práci je také rozebrán větší vliv okolního šumu a problémy spojené s tímto šumem.
Klíčová slova: Robot ovládaní hlas neuronová síť

Title:
Voice controlled robot

Author: Michal Kusý

Abstract: This thesis is focused on voice control of a mobile robot. The objective was to design a controller software, which is able to identify voice commands and begin moving in said direction. The controller software has been coded in Jupyter based platform using a neural network from Google API. The theoretical part describes the functioning of the neural network. The program supports basic czech movement directions as commands, some of which are listed in the conclusion. The thesis is also talking about the effect of background noise on the neural network and its problems.
Key words: Robot control voice neural network

Obsah

Úvod

Tématem této práce je robot ovládaný hlasovými příkazy. Téma jsem si vybral, jelikož mě zajímá robotika a neuronové sítě. V této práci jsem se nejdříve zaměřil na vysvětlení na jakých frekvencích operuje lidský hlas a jeho snímání. Zahrnul jsem i úvod do neuronových sítí a popis sítě RNN a RNN-T. Součástí výběru a návrhu neurové sítě bylo prozkoumání dostupných API a jejich funkcí. Ve druhé kapitole budete seznámeni s robotem Jetbot využitém u této práce a také počítačem od NVIDIA, který robota řídí. V té samé kapitole se také zmíním o knihovně Jetbot pro Python. Tato knihovna poskytuje příkazy, kterými je možné ovládat robota jednoduchými funkcemi. V další kapitole bude probrán návrh ovládacího software. Implementace nahrávání pomocí 2 alternujících vláken, aby se zabránilo ztrátě slov. Také je zahrnuto řešení spojování nahrávek z obou vláken. Využití Google API a jaký výstup jeho neuronová síť poskytuje. V práci je také zmíněno, jak jsem zpracoval výstup neuronové sítě, který poskytl Google API, i vyhodnocení řetězce slov do příkazu.

Kapitola 1

Problematika rozpoznávání hlasu

1.1 Popis vlastní problematiky rozpoznání hlasu

Lidský hlas se skládá ze zvuků vytvořených v hlasivkách. Tyto zvuky potřebujeme strojově zpracovat. Není potřeba však zpracovávat celé zvukové spektrum ale zvuky zhruba mezi 125Hz a 8kHz[16], v tomto rozmezí se totiž pohybuje lidský hlas. K nahrávání se používájí mikrofony. Existují různé typy mikrofonů s odlišnými vlastnostmi.

1.1.1 Mikrofon

Kondenzátorový mikrofon[17] snímá kmitání membrány způsobené zvukovými vlnami. Membrána je jednou z elektrod kondenzátoru elektrického obvodu mikrofonu. Kapacita kondenzátoru se mění dle pohybu membrány. Tato změna se převádí na elektrický signál. Kondenzátorové mikrofony obecně vyžadují elektrické napájení, aby mohl obvod fungovat. Při vhodné konstrukci mikrofonní vložky je také možné měnit směrové charakteristiky mikrofonu.

Kondenzátorové mikrofony jsou pokládány za nejkvalitnější a používají se často pro profesionální záznam. Také se vyrábějí pro měřicí účely.

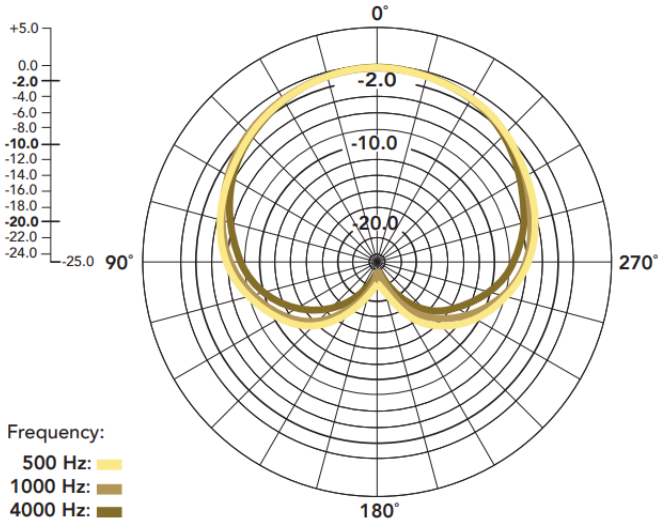
Elektretový mikrofon[17] je postaven na podobné bázi jako kondenzátorový mikrofon, akorát elektrické nápájení je nahrazeno elektretem. Elektret je feroelektrický materiál, který byl permanentně elektricky nabit nebo polarizován. Výsledkem tohoto nahrazení je jednodušší výroba a tudíž i nižší cena.

Mezi další typy patří Dynamický a Membránový mikrofon.

1.1.2 Směrové charakteristiky mikrofonů

V závislosti na konstrukci pouzdra mikrofonu může tento přijímat zvuk z různých směrů v různé intenzitě. Směrová charakteristika je frekvenčně závislá – projevuje se zpravidla u vysokých tónů, zatímco hluboké zůstávají nepoznamenány.

Všesměrová charakteristika přijímá zvuk stejně kvalitně ze všech stran a je nejjednodušší na dosažení při konstrukci. Kardioidní neboli ledvinová charakteristika potlačuje příjem zvuku „zezadu“ mikrofonu. Jde o typickou charakteristiku dynamických mikrofonů pro zpěváky, neboť potlačuje zpětnou vazbu. Superkardioidní přijímá zvuk částečně i zezadu. Hyperkardioidní přijímá ještě více zvuku zezadu mikrofonu. Osmičková neboli bidirekcionální charakteristika je taková, při které mikrofon přijímá zvuk pouze zepředu a zezadu, nikoliv však ze stran.



Obrázek 1.1: Znázornění kardioidní charakteristiky

1.1.3 FLAC

FLAC[22] neboli Free Lossless Audio Codec, je bezztrátový audio formát podobný MP3. Toto znamená, že audio v tomto formátu je kompresováno bez ztráty kvality. Funkcionálně je to podobné formátu ZIP, akorát FLAC je specializovaný pro audiozáznamy. FLAC je jedním z nejrychlejších a je jeden z nevíce podporovaných bezztrátových zvukových formátů. Navíc je open-source a není omezený žádnými patenty. Podporuje také značky neboli tagy a například také přebaly u hudebních skladeb.

Některé z význačných vlastností zahrnují bezztrátovost, rychlost dekódování, širokou hardware podporu, flexibilní metadata, streamovatelnost, archivovatelnost a odolnost vůči chybám. FLAC nehodlá přidat žádnou DRM ochranu proti kopírování.

1.1.4 Waveform - WAV\WAVE

Waveform Audio File Format[23] neboli WAV\WAVE je audio formát vytvořený firmou IBM a Microsoftem, pro ukládaní audio bitstreamů. V systému Windows je

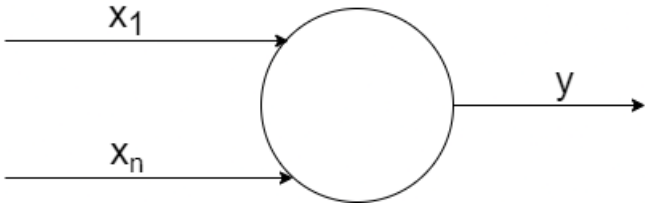
hlavním formátem pro nekompresované audio, ale WAV může obsahovat i kompresované audio. Nekompresovaná audiostopa se ukládá pod LPCM (Linear Pulse-Code Modulation) formátem, který obsahuje 2 kanály snímané na 44 100Hz s 16 bitovými snímky. Tento formát je standardní pro Audio CD.

S Windows 2000 formát WAV začal obsahovat i hlavičku. V této části jsou definovány audio data pro jednotlivé kanály, pozice reproduktorů a sjednocuje snímkovámí v souboru. Avšak nacházejí se nedostatky například v duplicitě informací v jednotlivých blocích, což naznačuje že soubory mají velkou velikost, a také absence záporných hodnot a 8-bit kódování oproti 16-bit. WAV soubory jsou také limitovány do velikosti 4GiB kvůli limitaci hlavičky velikosti souboru na 32-bit číslo bez znaménka.[23]

1.1.5 Neuronové sítě

Pro zpracování zvuku vytvořeným lidským hlasem se použije neuronová síť. Počítačová neuronová sít funguje na bázi umělého neuronu. Inspirace přisla z lidského mozku, který je složen z mnoha neuronů. Tyto neurony pomáhají mozku se rozhodovat a toto chování počítačové neuronové sítě napodobují. Základní jednotkou je perceptron, který má n vstupů a obvykle jeden výstup.[18]

Neuronová síť je vlasně tedy složení spojených perceptronů, které se kolektivně rozhodují na celkovém výsledku. Každý perceptron provede rozhodnutí a to se pak odešle do další jednotky v síti, dokud nedorazí na konec, tedy finální rozhodnutí sítě. Rozhodnutí perceptronu je vážená suma všech vstupů perceptronu. Každý perceptron má také svůj bias, neboli hodnotu přičtenou k vážené sumě. Toto číslo se pak předá většinou[20] nelineární aktivační funci. Výsledek aktivační funkce je poté výstup perceptronu.



Obrázek 1.2: Jednoduchý perceptron

U neuronové sítě počet těchto jednotek může dosahovat vysokého čísla, a také mohou být rozděleny do více vrstev než jen vstupní a výstupní. Tyto vrstvy navíc se nazývají skryté vrstvy a slouží ke zpracovávání výstupů předchozích vrstev, asice pro lepší přesnost výsledku. V některých případech jsou dokonce nutné, aby síť byla schopna vyřešit daný problém.

Většinou je také nutné data předzpracovat do určitého intervalu. Tento proces se nazývá preprocessing. U většiny sítí volíme interval $\langle -1;1 \rangle$. Díky tomuto procesu budeme mít sjednocená data což síti pomůže s rozhodováním a počítáním.

1.1.6 Učení neuronové sítě

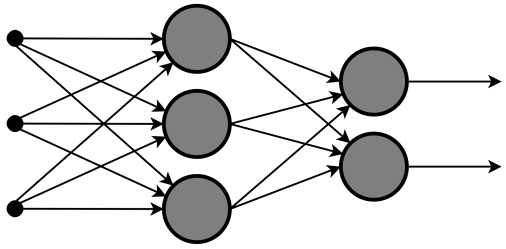
Učením se neuronová síť adaptuje přiřazené úloze pomocí vyhodnocování trénovacích dat. Při vyhodnocování se upravují váhy vstupů, aby se zlepšila přesnost sítě. Toto se provádí minimalizací chyb a učení je dokončeno, jakmile další učení významně nesníží počet chyb.

Strojové učení se běžně rozděluje do tří tříd.[18] Učení pod dohledem, učení bez dohledu a zpětnovazební učení.

Při učení pod dohledem se použije pár vstup-očekávaná odpověď. Úkolem je, aby síť odpověděla na vstup očekávanou odpovědí. Špatné odpovědi se eliminují upravou váhové funkce.

Učení bez dohledu funguje tak, že síť se snaží replikovat chování dle obdržených dat.

Zpětnovazební učení hledá nejkratší cestu nebo nejkratší počet akcí k splnění zadané úlohy. Cílem je, aby byl co nejmenší počet akcí nebo zvolit nejkratší cestu k cíli.



Obrázek 1.3: Znázornění sítě

1.1.7 Sítě typu Recurrent neural network

RNN (Recurrent neural network)[19] - Třída neuronových sítí, kde propojení mezi perceptrony vytváří graf orientovaný podle časové posloupnosti. Toto síti dovoluje se chovat časově dynamicky. Jako vstupní data se u této sítě mohou použít sekvenční data nebo data v časové posloupnosti. Tento algoritmus se používá hlavně u ordinálních nebo časově závislých problémů, jedním z příkladů je překlad vět a dalším bude zpracování mluveného slova. Tuto technologii používají populární aplikace jako například Siri, hlasové hledání a nebo i Google překladač[19].

Stejně jako běžné neuronové sítě[21] tento typ sítě využívá trénovací sadu dat. Rozdíl je v tom, že tyto sítě mají vlastní pamět, pomocí které ovlivňují svá rozhodnutí na vstupu i výstupu dat. Běžné neuronové sítě neuvažují spojitost mezi daty. Síť RNN vidí spojitost v pořadí dat přicházejících do sítě, což je velkou výhodou při překladu a zpracovávání řeci, jinak by se například u překladu překládálo slovo od slova a tento způsob u mnoha jazyků nefunguje.

Další charakteristikou této sítě je sdílení parametrů perceptrony napříč celou sítí, zatímco u tradičních sítí se parametry liší u každého perceptronu.

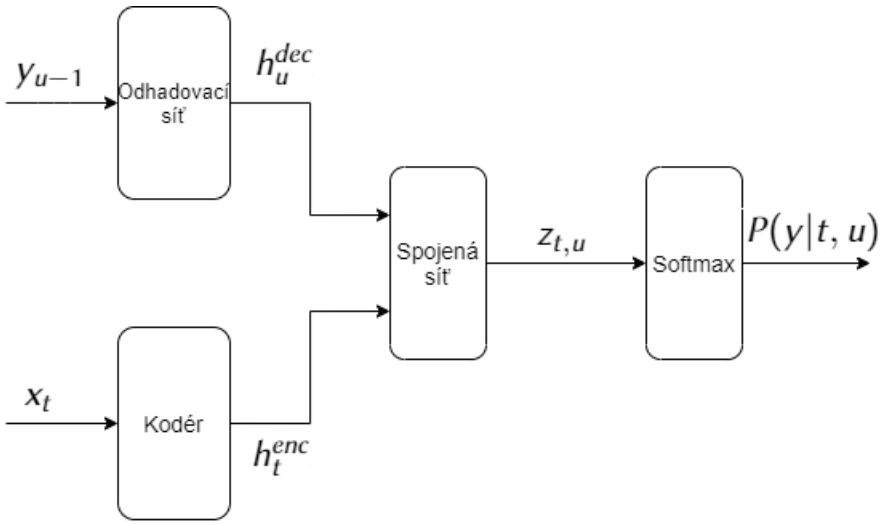
Sítě RNN trpí dvěma problémy. Těmito problémy jsou tzv. explodující a mizející váhy. Síť se neustále učí a zpětnou propagací si upravuje váhu, pomocí které se

rozhoduje. Vzhledem k tomu že váha je sdílená napříč modelem, tak se může stát že u explodující váhy bude váha nekonečně růst dokud nebudou výsledky NaN. U mizející váhy se budou váhy naopak zmenšovat.

1.1.8 Sítě typu RNN-T

Architektura sítě RNN-T navrhnutá Gravesem[1] se skládá z kodéru, odhadovací sítě a typicky jedné společné sítě. Tato síť přímo odhaduje sekvence slov bez použití externího modelu.

Síť je doplněna o prázdný symbol, aby model nevracel podslovo v každém čase $t = 1, 2, 3, ...T$, kde T je celkový počet časových okamžiků nahrávky. Vstup kodéru je dimenze signálu z log-mel filtru, který zajistí, aby v nahrávce byl pouze hlas.[?] Vstup odhadovací sítě je poslední neprázdný symbol



Obrázek 1.4: Znázornění sítě typu RNN-T

Vstup spojené sítě jsou výstupní vektory odhadovací sítě a kodéru. Spojená síť vrací vektor pravděpodobností, který pak putuje do vrstvy se aktivační funkcí softmax. Tato poslední vstva odhadne podslovo, z výsledných podslov pak síť sestaví slova, dokonce i řečenou sekvenci slov.

1.2 Příklad různých API dostupných na webu

1.2.1 CMU Sphinx

Open source API pro rozpoznání mluveného slova od Carnegie Mellon University[7]

Má několik různých verzí pro různé aplikace a programovací jazyky např. PocketSphinx napsaný v C, SphinxTrain obsahující nástroje pro trénování akustických modelů a základní knihovnu SphinxBase. CMU Sphinx nasbírala přes 20 let výzkumu

a dat. Podporuje několik jazyků, jednoduchá modifikovatelnost, možnost přidání vlastního modelu - čeština bohužel není podporována.

1.2.2 Wit.ai

Wit.ai je jazykové rozhraní pro převod vět na strukturovaná data.[15] Nabízí jednoduché vytvoření chatovacích botů a jednoduché vytvoření multiplatformních aplikací, a dokonce podporuje technologii Smart Home a chytré příslušenství. Toto rozhraní poskytuje Facebook, také podporuje mnoho jazyků pro převod psaného textu do dat. Pro převod řeči na text je podpora jazyků skromnější. Čeština není podporována, tato služba je zdarma.

1.2.3 Microsoft Azure

Microsoft nabízí vlastní API k převodu řeči na text. Nabízí tvorbu vlastních modelů spolu s jejich přednaučenými modely, flexibilní nasazení a rozpoznání mluvčího. Dále zaručuje bezpečnost dat na jejich cloudu.[12] Na stránkách Microsoftu jsou také ukázkové aplikace k této API. Podpora několika jazyků, čeština není podporována. 5 hodin zvuku na měsíc zdarma.

1.2.4 IBM Watson Speech to Text

IBM nabízí převod řeči na text přes jejich platformu Watson. Nabízí převod řeči na text v reálném čase pro 7 řečí, vysokou přesnost, upravitelnost modelu. Podporuje nahrávky až do délky 1000s. IBM zaručuje rychlý a přesný převod řeči na text i z nahrávek nižší kvality. [14] Podporují několik jazyků, čeština se mezi nimi bohužel nenachází. IBM nabízí plán na 500minut měsíčně zdarma.

1.2.5 Google API

Google nabízí svoji API pro převod řeči na text. Momentálně je podporováno 125 jazyků. Google nabízí velmi přesné rozpoznání, díky neuronovým sítím s hlubokým učením. Jednoduchou úpravu modelu pomocí uživatelského rozhraní. Google také nabízí službu se kterou se model uloží na lokální server a data se nemusí posílat na servery Google. [13] Nabízí možnost převodu na text v reálném čase, nebo přes soubor. Cenový plán Googlu je 60minut měsíčne zdarma, a poté se přejde na placený režim s cenou zhruba 0,5Kč/min viz. Obrázek 1.5

Google API má také omezení ve formátu, velikosti a délky nahrávek. Podporovány jsou nahrávky v bezztrátovém formátu *FLAC* a v *LINEAR16*. Platforma ale také podporuje ztrátová kódovaní, mezi která patří *MULAW*, *AMR*, *AMR_WB*, *OGG_OPUS*, *SPEEX_WITH_HEADER_BYTE*, *MP3* a *WEBM_OPUS*.

Feature	Standard models (all models except enhanced video and phone call)		Enhanced models (video, phone call)	
	0-60 Minutes	Over 60 Mins up to 1 Million Mins	0-60 Minutes	Over 60 Mins up to 1 Million Mins
Speech Recognition (without Data Logging - default)	Free	$0.006 / 15 seconds **	Free	$0.009 / 15 seconds **
Speech Recognition (with Data Logging opt-in)	Free	$0.004 / 15 seconds **	Free	$0.006 / 15 seconds **

Obrázek 1.5: Ceník Google API

Formáty *WAV* a *FLAC* obsahují záhlaví popisující vlastnosti audionahrávky. Díky tomuto záhlaví není potřeba ve volání API konfigurovat kódování, bitovou šířku nahrávky a jiná potřebná metadata týkající se samotného souboru. Google obecně doporučuje používat kódování *FLAC*.

Samotná nahrávka však nesmí překročit velikost 10MB a nesmí být delší jak 480 minut v případě asynchronního volání funkce *LongRunningRecognize*, nebo být delší jak minuta v případě synchronního volání funkce *Recognize*. Existuje také možnost nahrávku streamovat pomocí funkce *StreamingRecognize* a tento způsob má limit délky streamování 5 minut. Pokud by bylo třeba streamovat déle jak 5 minut, tak Google připravil návod s kódem pro nekonečný stream.

K volání rozponávacích funkcí je také možné přidat vlastní sadu frází, celkem až 5000 frází do 100 000 znaků s maximem 100 znaků pro jedno slovo. Aby Google zabráníl přetížení svých serverů, tak limitoval počet požadavků na 900 denně s maximálním zpracováním 480 hodin zvukové stopy.

1.2.6 Porovnání API

Při zvolení API pro vývoj softwaru jednoznačně vyhrálo Google API kvůli podpoře českého jazyka. Cenově je však nejméně výhodný, jak lze vidět v tabulce 1.1.

Pokud by se zvolila varianta ovládání robota pomocí anglických příkazů, byla by zvolena jedna z open source variant. Tato volba by se projevila i jako levnější alternativa za cenu možné nižší kvality datové sady a rozpoznávání oproti placeným API, které jsou poskytovány od velkých korporací.

Název API	Podpora CZ	Cena
CMU Sphinx	Ne	Zdarma
Wit.ai	Ne	Zdarma
Microsoft Azure	Ne	300minut měsíčně zdarma
IBM Watson Speech to Text	Ne	500minut měsíčně zdarma
Google API	Ano	60minut měsíčně zdarma

Tabulka 1.1: Porovnání API

Kapitola 2

Platforma NVIDIA Jetson a Jetbot

2.1 NVIDIA Jetson

NVIDIA Jetson[10] je vestavěný počítačový systém, který je optimalizován na aplikace využívající neuronové sítě. Je možné provozovat několik neuronových sítí. Aplikace využívající tyto sítě můžou klasifikovat obrázky, detekovat objekty a zpracovávat řeč.

The Jetson Nano module is a small AI computer that has the performance and power efficiency needed to run modern AI workloads, multiple neural networks in parallel, and process data from several high-resolution sensors simultaneously. This makes it the perfect entry-level option to add advanced AI to embedded products.[10]

Tento počítač obsahuje čtyřjádrový ARM procesor běžící na 1.43Ghz. 128 jádrové Maxwell GPU. 4GB DDR4 paměti, sériový MIPI CSI-2 port pro videokameru a PCIe gen2 slot. Jetson Nano disponuje 472 GFLOPS výpočetního výkonu na 16bitových číslech při spotřebě 5-10W.

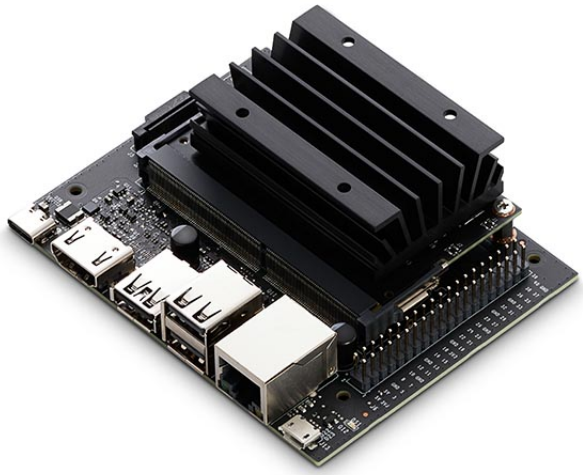
K dispozici je také levnější model Jetsonu Nano, tento model má pouze 2GB ram, méně USB 3.0 portů a jeden MIPI port na kameru.

U obou modelů se jako paměťové úložište využívá SD karta, na tuto SD kartu se instaluje image linuxového systému, který byl předpřipraven od společnosti NVIDIA.

2.2 Seznámení s NVIDIA Jetbot

NVIDIA Jetbot se skládá z NVIDIA Jetson, šasi a kol. Zakoupit se dá jako kompletní předvyrobený balíček a nebo jako DIY sada, ve které je NVIDIA Jetson a seznam materiálů a návod jak vytisknout šasi a kola.

Jetbot je v základu vybaven programy napsanými v Jupyter, které využijí jeho základní výbavu. Příkladem je třeba program, který se naučí rozpoznat cestu podle obrazu na kameře. Jeden z dalších programů dokáže rozpoznat pohyb člověka a rozpoložení těla.



Obrázek 2.1: NVIDIA Jetson

2.2.1 Nastavení sítě u robota

Jetbot se může programovat přímo na stroji nebo přes internet v prohlížeči. Pro programování v prostředí Jupyter je třeba robota pouze připojit k internetu přes kabel nebo Wi-fi. Robot pomocí tohoto síťového připojení získá adresu IP, pomocí které je možné se připojit na server s programovacím prostředím. Wi-fi lze na robotovi nastavit přes běžné uživatelské rozhraní pomocí myši, klávesnice a monitoru. Je také možné se připojit přes SSH relaci a nastavit připojení Wi-fi pomocí této vzdálené relace. U tohoto typu nastavování se nejdříve zapojí internetový kabel do robota a počká se než se na displeji robota objeví IP adresa. Alternativně lze také připojit USB- mikroUSB kabel do počítače a robota, u tohoto typu připojení má robot pevnou IP *192.168.55.1*. Se získanou IP adresou je pak možné připojení na SSH server přes vhodný software, například Putty. Po připojení a přihlášení se v konzoli zadá příkaz *nmcli device* a bude se hledat řádek *wlan0*. Přítomnost řádku s *wlan0* znamená, že robot podporuje připojení Wi-fi, a bude možné nastavit připojení. Přes příkaz *nmcli device wifi list* lze pak vypsat veškeré viditelné Wi-fi sítě v dosahu robota. Samotné připojení k Wi-fi síti pak zajistí příkaz sudo *nmcli device wifi connect <MY_WIFI_AP> password <MY_WIFI_PASSWORD>*, kde *MY_WIFI_AP* označuje SSID, neboli název Wi-fi a *MY_WIFI_PASSWORD* poté označuje heslo k této síti.

2.3 NVIDIA Jetbot použitý k realizaci

Použitý Jetbot byl zapůjčen na fakultě od vedoucího této bakalářské práce. Jedná se o model Waveshare, který má kovový podvozek se čtyřmi koly, Wi-fi antény,

Obrázek 2.2: NVIDIA Jetbot

kameru a baterii. K robotu se doukoupil kondenzátorový USB mikrofon s kardinoidní charakteristikou pro možnost nahrání hlasových příkazů.

Operační systém robota je v tomto případě Ubuntu s přídavnými systémy od společnosti NVIDIA, jako například zabudovaný Jupyter server pro sepsání a vykonání kódu, pomocí kterého lze robota ovládat.

2.4 Knihovna Jetbot v Pythonu

Jedná se o open-source knihovnu od společnosti NVIDIA pro Jetbota. Obsahuje příkazy pro zjednodušení kódu ovládání robota, jeho kol a kamery. Knihovna dostává časté aktualizace od komunity uživatelů a také od společnosti samotné. Součástí knihovny jsou příkladové programy, ukazující základní pohyb, rozpoznání objektů a využití kamery.

Kapitola 3

Návrh ovládacího software

3.1 Použitý jazyk - Python

Python je interpretovaný, objektově orientovaný, vysokoúrovňový jazyk s dynamickou sémantikou. Python má vysokoúrovňové datové struktury s kombinací dynamiky Pythonu. Je velmi lákavý pro rychlý vývoj a jednoduchý debug. Je jednoduchý na naučení, díky jeho syntaxi a čitelnosti. Součástí Pythonu je také balíček Pip, pomocí kterého se dají jednoduše stahovat knihovny.[11]

3.2 Prostředí Jupyter notebook

Jupyter je neziskový open-source projekt, který vznikl z IPythonu. Jupyter se snaží podporovat interaktivní data a vědecké výpočty napříč všemi programovacími jazyky. [24]

Jupyter notebook je webová aplikace, která umožnuje vytvářet a sdílet dokumenty se spustitelným kódem přímo z této aplikace, umožňuje zobrazit obrázky, grafy, výpočty a doprovodný text, který může vysvětlovat postup, nebo funkci daného kódu.

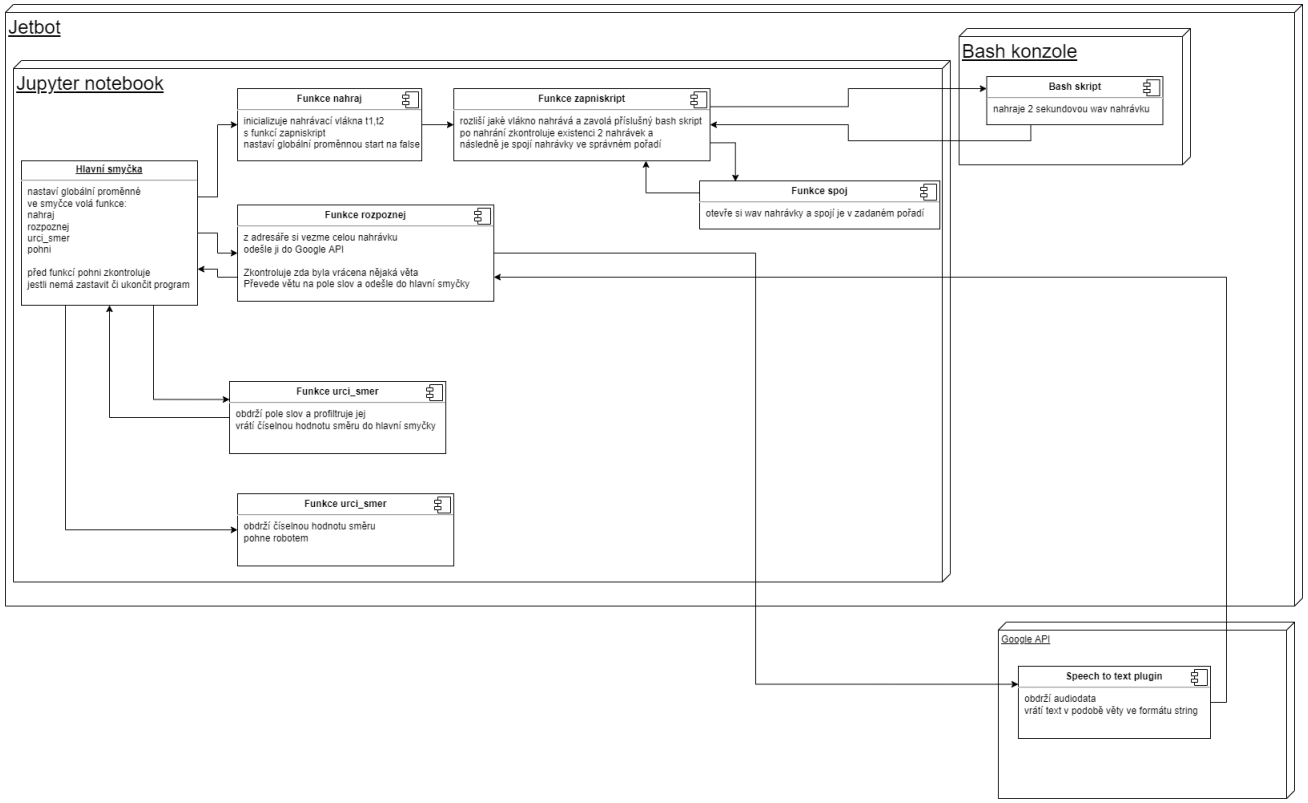
3.3 Popis ovládacího software

Program jsem si rozdělil na několik částí. Nahrávací část, kde se ve 2 proudech nahrává zvuk. Nejprve nahrává první proud po dobu několika vteřin, poté začne nahrávat proud druhý, a následně se vrací k prvnímu proudu. Tento postup zajistí, že robot bude vždy poslouchat. Jakmile jakýkoli proud nahraje svoji stopu, tak se poslední dvě stopy spojí a odešlou se do další části programu.

V druhé části programu se přijatý audio soubor odešle ke zpracování do rozpoznávacího software, který vrátí textovou formou slova, která byla v nahrávce řečena. Přijatá slova se pak odešlou k dalšímu zpracování uvnitř robota. Software profiltruje

slova na příkazy, které program zná a umí je zpracovat a pošle je pohybové funkci. Pohybová funkce tento vstup zpracuje a sdělí robotovi kam se pohybovat, či jestli se má zastavit.

Různé komponenty programu jsou znázorněny na diagramu 3.1.

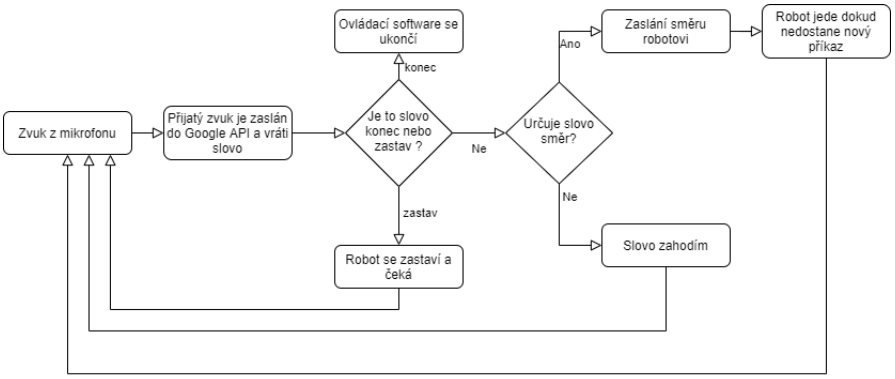


Obrázek 3.1: Diagram programu

3.3.1 Diagram ovládacího software

Ovládací software by se měl skládat alespoň ze čtyř částí, aby mohl fungovat tak jak je požadováno.

Potřebuje nahrávací smyčku, která zaznamená příkazy. Ideálně je zaznamená bez šumu a bez okolního hluku. Poté bude potřeba funkce převádějící hlasová data na text, tuto část ve většině případů zařizuje neuronová síť. Třetí část by měla zpracovat text obdržený z minulé části a převede ho na příkaz, který se zašle do poslední části programu. Poslední část programu bude řídit jízdu robota podle příkazu z obdrženého třetí částí. Běh celého programu je znázorněn na diagramu 3.2.



Obrázek 3.2: Diagram programu

Kapitola 4

Možné implementace

Ovládací software lze implementovat s lokální neuronovou síťí, která by zajistila i fungování bez nutnosti internetu. V tomto případě by bylo nutno síť natrénovat aby reagovala na určitá slova, ideálně od většího počtu různých mluvčích, a tím se zajistí rozpoznání klíčových slov. U většího počtu slov se může jednat klidně o stovky nahrávek slov a tím pádem i delší dobu trénování sítě a více zabraného místa na disku.

Druhá možnost bylo využít přednatrénované API s velmi vysokým počtem trénovacích nahrávek, tudíž i vysokou spolehlivostí, pokud lze zajistit nahrávku dostatečné kvality. Pokud nahrávka bude obsahovat okolní šum, tak je možné že síť tomuto jevu nebude přizpůsobená oproti lokální variantě a budou vznikat chyby. Výhodou je ovšem velmi rychlé vyhodnocení v datacentrech externích API, které jsou optimalizovány na tyto rozpoznávací operace.

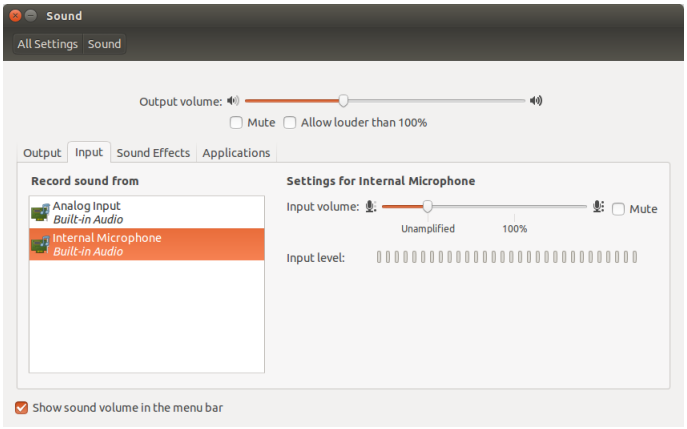
4.1 Popis realizace

K robotovi bylo nutné nejdříve připojit mikrofon, aby bylo čím nahrávat zvuk. Mikrofon bylo třeba nastavit na správnou úroveň citlivosti snímání, aby negeneroval šum (Obrázek 4.1). V případě vysokého šumu by pak neuronová síť v další části programu nemohla rozponat řečená slova.

4.1.1 Vývoj programu

Program jsem začal vyvíjet v Google Colab, kde byl sestaven testovací program samotného Google API. Tento program se skládal ze skriptu na nahrávání zvuku, uložení nahrávky na disk Google a odeslání do Google API. V Google Colab bylo nutné povolit přístup k prostředkům prohlížeče a povolit nahrávání mikrofonem pomocí javascriptu.

Po testování různých frází jsem vývoj přesunul spolu s testováním na Jetbota s vývojovým prostředím Jupyter notebook. Při prvním testu jsem zjistil, že nefungoval



Obrázek 4.1: Příklad nastavení mikrofonu při testování

dříve napsaný nahrávací skript. Při hledání řešení tohoto problému jsem testoval ovládání robota s pevně stanovenými frázemi. U nastavení směru pohybu robota bylo třeba stanovit sílu motoru, nebo zda-li robot správně reaguje na stanovené fráze. Po dokončení testování jsem se vrátil k implementaci nahrávání, kde jsem zvolil nahrávání přes bash soubor spuštený z vývojového prostředí. Na další testováni jsem si připravil čistou instalaci Ubuntu na notebooku pro možnost testování mikrofonu před nasazením a testem na robotovi. Na notebook bylo třeba nainstalovat Jupyter a knihovny potřebné pro běh programu. Server Jupyteru jsem posléze spustil z příkazového řádku ve složce s programem pomocí příkazu *jupyter notebook*.

4.1.2 Příprava bash souboru

Bash obsahuje příkaz *arecord*. Tento příkaz spustí nahrávání výchozím systémovým mikrofonem a pomocí parametrů je možné upravit jeho chování. V základní podobě tento příkaz nahrává v 8-bit formátu a se vzorkovací frekvencí 8000Hz. Při testování příkazu byl formát změněn na S16_LE a frekvenci 44 100 Hz. Tyto parametry se osvědčili vyšší kvalitou nahrávky a lepším rozpoznáním v Google API. Pomocí parametrů jsem stanovil formát nahrávky na příponu *wav* formátu Waveform. Parametr *-c 2* nastavuje nahrávaní na Stereo místo základního jednokanálového Mono a při testování zvýšil kvalitu nahrávky. Délka nahrávky byla stanovena na 2 sekundy a ukládala se pod názvem *nahravka.wav*.

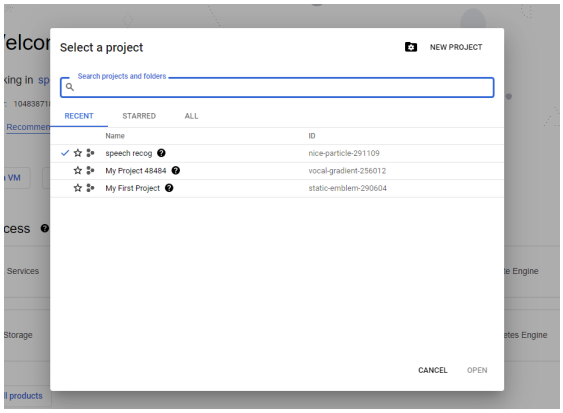
Při testování této metody se projevil problém usekávání slov a zpožděného nového spuštění nahrávání kvůli sekvenčnímu zpracování programu. Řešením tohoto problému byla paralelizace dvou nahrávacích proudů a jejich následné spojení do jedné nahrávky. V příkazu bash skriptu se změnil pouze název souboru, aby bylo poznat, jestli nahrál první či druhý proud. Do skriptu bylo třeba přidat záhlaví pro systém, aby poznal, že se jedná o shell skript pomocí *#!/bin/sh*. Za zahlávím pokračuje kód skriptu: *arecord -t wav -d 2 -f S16_LE -c2 -r44100 "nahravka1"*. Aby skript bylo možné programově spustit, bylo třeba nastavit práva pomocí příkazu *chmod +x "název skriptu"*.

Po otestování funkčnosti skriptu jsem sepsal funkci, která bude ve vlastním vlákně

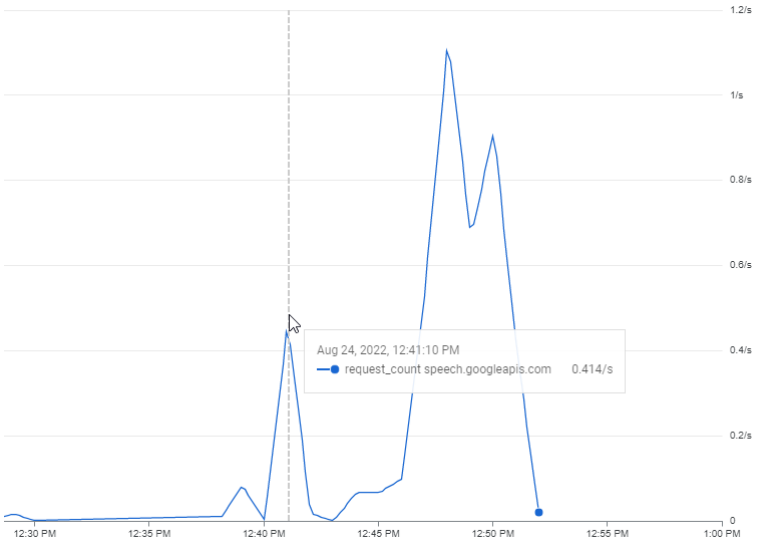
volat nahrávání každé 2 sekundy, a po dokončení nahrávání spojí nahrávky z obou vláken ve správném pořadí. Také bylo třeba zahrnout kontrolu, jestli program neukončil běh. Spojená nahrávka se pak odeslala do Google API.

4.1.3 Nastavení Google API

Rovněž zapotřebí nastavit Google API. K používání je nutností Google účet. Nejdříve bylo třeba založit projekt v Google Cloud, abych mohl vytvořit autorizační klíč.



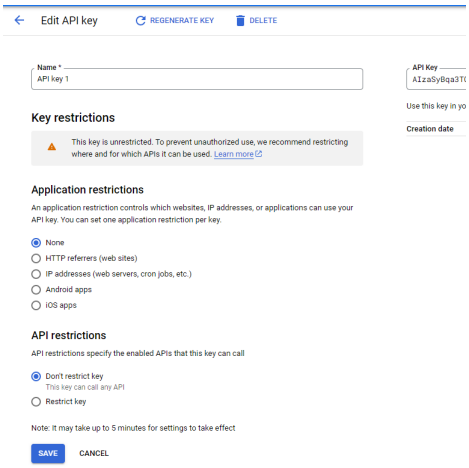
Obrázek 4.2: Vytvoření projektu



Obrázek 4.3: Tabulka volání API

Pomocí projektů se dají rozlišovat jednotlivé aplikace, které mají vestavěný reporting systém, pomocí něhož se dá používání služeb Google sledovat a debugovat. Na obrázku 4.3 máme znázorněn graf využití volání Google API. Dodatečná informace u myši zobrazuje, kolikrát za vteřinu se volala API.

Pomocí této cloudové konzole lze nastavit funkci OAuth, a nebo také různé služby API nebo i SDK, jako například: SDK pro mapy Google a pro Android/iOS, Text-to-Speech, Speech-to-Text, různé služby do Youtube, strojový překlad a mnoho jiných. Po povolení API potřebné pro tuto práci bylo nutné vygenerovat autorizační klíč v nastavení projektu. U tohoto klíče je zapotřebí nastavit práva přístupu (viz obrázek 4.4), aby nedocházelo k neoprávněnému použití klíče. Tento klíč se pak použije v příkazu uvnitř kódu pro autorizaci.



Obrázek 4.4: Nastavení přístupového klíče do Google API

4.1.4 Hlavní smyčka

Program jsem si rozdělil na několik elementárních funkcí, které jsem poté implementoval.

Program začíná vytvořením počítadla, které počítá počet oznámení programu do prostředí Jupyter a pokračuje nekonečnou smyčkou, ve které začíná běžet hlavní část programu. Funkce *try* kontroluje, zda-li se neobjevila v programu neočekávaná chyba. Pokud rozpoznávací fuknce nepoznala řečená slova, nebo žádná neslyšela, program nahlásí, že nic nezaregistroval.

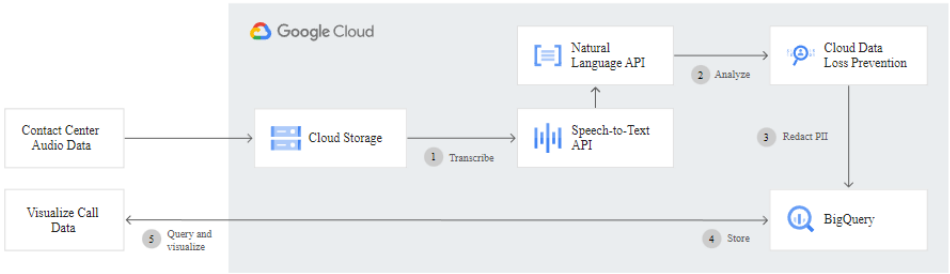
V první sekci programu se počítá počet výstupů do prostředi Jupyter, a pokud počítadlo dosáhne stanovené hodnoty (momentálně 6), tak program vstoupí do sekce funkce, která vyčistí výstup vypsaný robotem. Také jsou stanoveny globální proměnné určující, zda-li se program nyní zapnul, nebo jestli je program ve stavu ukončení.

V další sekci je nahrávací funkce realizovaná pomocí multiprocessingu. Multiprocessing umožní programu pracovat ve více vláknech, která mohou běžet současně nezávisle na stádiu programu. Pomocí této technologie mohu spouštět externí soubor zajišťující nahrávání přes Linuxové bash příkazy. Toto nahrávání je postaveno na příkazu *arecord*, pomocí kterého volám nativně mikrofon bez dodatečných knihoven, které by mohli ovlivnit kvalitu a rychlost zpracování nahrávání. Sestavený příkaz nahrává po dobu 2 sekund z připojeného mikrofonu, a po této době se nahrávka uloží.

```
global t1,t2,start,konec

i = 0
start = True
konec = False

try:
    while True:
        i+=1
        if i%6 == 0:
            clear_output()      #vyčistí výstup v notebooku
            i-=6                #po každých 6 cyklech
        nahraj()                #nahraje audiozáznam
        pole = rozpoznej()      #záznam se odešle do Google API
        sm = urci_smer(pole)    #z Google API se vrátí pole slov, které se nyní zpracuje
        if sm<-10:              #kontrola ukončení smyčky
            break               #
        #pohni(sm)               #změna směru pobyhu robota
except Exception as e:
    print ("\n\n\033[91m\033[1m"+repr(e)+"\033[0m\033[0m")

#ukončení vláken
konec = True

#zastavení robota
#robot.stop()

#program počká než vlákno dokončí poslední cyklus
t1.join()
t2.join()
```

Obrázek 4.5: Hlavný smyčka programu

Po 2 sekundách se zapne druhý proud nahrávání, který vytváří nahrávku v době kdy se zpracovává první nahrávka. Program dále kontroluje, jestli existuje nahrávka z obou těchto proudů, aby mohli být spojeny do jedné nahrávky ve správném pořadí. Spojování jednotlivých nahrávek zajisťuje Python knihovna wave, která umí zpracovávat soubory typu Waveform.



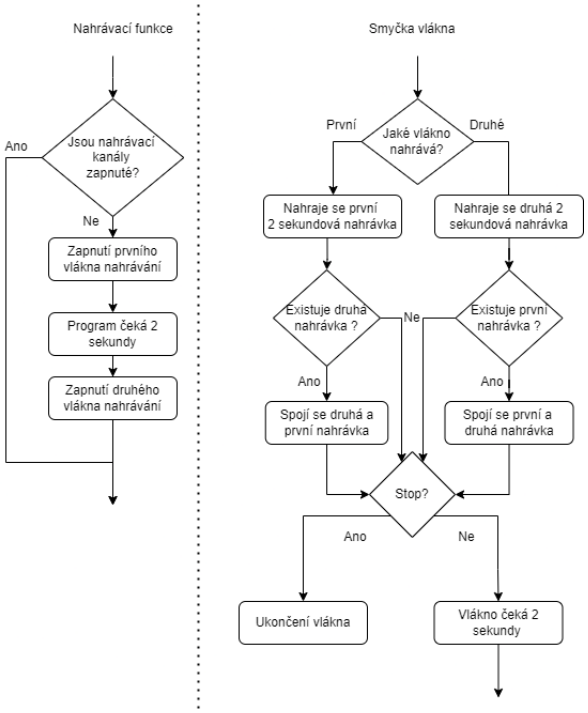
Obrázek 4.6: proces zpracování požadavku v Google API[13]

Tato sloučená nahrávka se pak odešle do rozpoznávací funkce. Rozpoznávací funkce zašle požadavek na Google API s autorizačním klíčem, který se vygeneruje na google stránkách. Tento klíč se musí nastavit kvůli autorizaci požadavku a případné fakturaci, pokud by počet požadavků překročil rámec bezplatné verze. Po přijetí požadavku se v Google API zpracuje odeslaná nahrávka neuronovou sítí RNN-T. Po zpracování nahrávky neuronová síť vrátí větu, která byla v nahrávce řečena a tuto větu obdrží program zpět od Google API. Tato věta je odeslána do funkce, která větu rozdělí na pole slov. Toto pole je pak dále zpracováno filtrující funkcí,

která vybere předefinovaná směrová slova a převede je na číslo určující směr, kterým robot pojede, nebo jestli se zastaví.

4.1.5 Testování a eliminace potkaných problémů

Během vývoje a testování programu byli objeveny různé nedostatky prvotních návrhů. Jedním z problémů bylo usekávání slov v nahrávce způsobené koncem nahrávání, aby se vytvořil soubor nahrávky. Tento styl nahrávání slov byl v jednom proudu problémový, protože běžel pár vteřin, a poté byl program na nějakou chvíli hluchý také se stávalo, že bylo useknuto příkazové slovo. Tento problém se opravil implementací dvouproudového nahrávání, které nahrává v jednom proudu a mezitím druhý proud ukládá nahrávku. Tento přístup zajistí, že robot nebude hluchý v jakémkoli stavu programu. Toto řešení požadovalo implementaci multiprocessingu a spouštění proudů ve dvou vláknech.



Obrázek 4.7: Proces nahrávání pomocí vláken

Spolu s implementací vláken do programu se objevil problém s jejich chováním po ukončení hlavní smyčky. V testovacích verzích se vlákna spustili, ale nevěděli, jestli je program ukončen. Toto způsobilo, že i po ukončení programu se generovali nové nahrávky. Při opětovném zapnutí programu se pak vytvořila další nová vlákna, která také vytvářela nahrávky a způsobovalo to problémy s nahráváním, a také toto neustálé přepisování rovněž není vhodné pro životnost disku. Tento problém byl původně řešen vypnutím celého procesu Pythonu. Toto řešení nebylo ideální, jelikož bylo třeba definovat všechny funkce a proměnné v programu znovu. Jelikož vlákna sdílí všechny globální proměnné, tak jsem zvolil možnost smyčky řízené globální proměnnou *konec* typu Boolean, která vlákno ukončí po ukončení hlavní smyčky.

Nakonec si nechám programem potvrdit konec vlákna pomocí zavolání funkce *join*, která je definována v objektu vlákna.

Dalším problémem byla kvalita a hlasitost nahrávané řeči, program jsem testoval na dvou různých mikrofonech, bohužel jeden z nich nebyl vhodný k využití vzhledem k mobilní povaze robota. Hlavním problémem byl šum. Při kalibraci citlivosti mikrofonu bylo třeba dbát na sílu šumu na jednotlivých úrovních nastavení, také bylo třeba hlídat hlasitost nahrávky, aby nebyla příliš tichá. Pokud by hlas v nahrávce byl příliš tichý, síť by měla problém rozpoznat, jaká slova byla řečena. Při zvyšování výstupní hlastosti se také objevoval šum. U kvalitnějších mikrofonů nebylo potřeba výrazné nastavování, zatímco u levnějšího mikrofonu byl přítomen šum způsobený elektrickým polem robota. Při kontrole dokumentace Google API bylo uvedeno doporučení nahrávku neupravovat, jelikož úpravy snižují přesnost rozpoznání[13].

4.1.6 Mikrofony použité k testování

MC-1UN0 mini

MC-1UN0[22] mini je malý mikrofon, který se dá zapojit pomocí USB konektoru do počítače. Tento mikrofon je kondenzátorový s kardinoidní charakteristikou což znamená, že mikrofon sníma nejsilněji z místa, kde jsou otvory k okruhu mikrofonu. Jeho frekvenční rozsah zahrnuje frekvence mezi 100Hz a 16KHz. Má impedanci 2.2KOhm s citlivostí -47dB (± 4dB).

U tohoto mikrofonu se projevoval šum způsobený elektrickým polem robota. Podobný šum se projevoval i při testování na stolním počítači. U mikrofonu bylo potřeba zmenšit citlivost tak, aby šum nebyl výrazný a nedělal problémy s identifikací slov. I přes tyto změny šum stále ovlivňoval výsledky sítě. Při pokusu o odstranění šumu v profesionálním nahrávacím programu, zůstala zvuková stopa velmi tichá.

Při podařené identifikaci začal mikrofon snímat zvuk motoru a kol robota, takže bylo nutné mluvit více nahlas, jinak by byl zvuk jízdy robota hlasitější než hlasové příkazy.

Audio-Technica ATGM2

Audio-Technica ATGM2[23] je nasazovací mikrofon s lepivým úchytkem na sluchátko. Jeho hlavní využitím je volání a streamování. Mikrofon je flexibilní a dá se různě ohýbat, umožnuje dosáhnutí optimální pozice. S hyperkardiodní charakteristikou je zajištěna minimalizace okolního hluku, a také je přibalená ochranná vrstva pro omezení zvuků dýchání. Kabel mikrofonu je 3m dlouhý a obsahuje tlačítko na vypnutí mikrofonu. K počítači se připojuje 3.5mm jack konektorem, k připojení do robota byla využita externí ADC zvuková karta připojená přes USB. Citlivost tohoto mikrofonu je -37dB s impedancí 2,1KOhm.

Používání tohoto mikrofonu je určeno na malou vzdálenost díky jeho konstrukci. Proto není velmi vhodný pro toto využití jako snímač hlasu na robotovi, uživatel

Obrázek 4.8: MC-1UN0 mini

by musel chodit s mikrofonem v ruce za robotem pro zaručení ovladatelnosti a rozpoznání řečených slov.

Obrázek 4.9: Audio-Technica ATGM2

Závěr

Implementace programu umožnila realizovat veškeré požadované funkce, které může uživatel od daného robota požadovat. Při testování se objevili problémy hlavně u vývoje a testování zvukových částí programu, především nahrávání kde bylo potřeba správně specifikovat a nastavit nahrávací zařízení, a to hlavně kvůli kontrole šumu, jelikož Google API je velmi citlivé na šum a není doporučeno používat šumové filtry podle jejich dokumentace, kvůli přesnosti detekce mluvených slov. Předpokládám však, že robot bude moci operovat i v prostředí se zvýšeným šumem a jiným okolním hlukem. Robot reaguje na základní směrové příkazy jako "dopředu","dozadu", "doleva", "doprava"a také na příkazy "zastavit"a "stát"při kterých se přestane pohybovat. Rovněž umí rozpoznat příkaz "konec", kde se zastaví smyčka progamu a program se vypne. Pro ideální implementaci z hlediska šumu je vhodné použít mikrofon s konfigurovatelnou funkcí gain a rozsahem snímaní 360°.

Literatura

[1] GRAVES, Alex. *Sequence Transduction with Recurrent Neural Networks* [online]. Department of Computer Science, University of Toronto, Canada: University of Toronto, Canada, 2012 [cit. 2021-9-6]. Dostupné z: https://arxiv.org/pdf/1211.3711.pdf

[2] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. Přeložil Rudolf PECINOVSKÝ. Praha: Grada Publishing, 2019. Knihovna programátora (Grada). ISBN 8024731002.

[3] JOSEPH, Lentin. *Learning Robotics Using Python*. Birmingham: Packt Publishing, 2015. ISBN 978-1783287536.

[4] KURNIAWAN, Agus. *Getting Started with NVIDIA Jetson Nano*. PE Press, 2019. ISBN 9780359978212.

[5] *JetBot* [online]. Santa Clara, California, United States: NVIDIA, 2021 [cit. 2021-9-6]. Dostupné z: jetbot.org

[6] *Jetson FAQ* [online]. Santa Clara, California, United States: NVIDIA, 2021 [cit. 2021-9-6]. Dostupné z: https://developer.nvidia.com/embedded/faq

[7] CMU Sphinx. *CMU Sphinx* [online]. Carnegie Mellon University: -, 2021 [cit. 2021-9-6]. Dostupné z: https://cmusphinx.github.io/

[8] Convolutional Neural Networks (CNNs / ConvNets). *CS231n Convolutional Neural Networks for Visual Recognition* [online]. [cit. 2021-3-11]. Dostupné z: https://cs231n.github.io/convolutional-networks/

[9] *Language model fusion for streaming end to end speech recognition* [online]. Mountain View, California, United States: Google, 2021 [cit. 2021-9-6]. Dostupné z: https://arxiv.org/pdf/2104.04487.pdf

[10] *NVIDIA Jetson* [online]. Santa Clara, California, United States: -, 2021 [cit. 2021-9-6]. Dostupné z: https://www.nvidia.com/cs-cz/autonomous-machines/jetson-store/

[11] *Python* [online]. Python, 2021 [cit. 2021-9-6]. Dostupné z: python.org

[12] *Speech to Text: A Speech service feature that accurately transcribes spoken audio to text* [online]. Redmond, Washington, United States: Microsoft, 2021 [cit.

2021-9-6]. Dostupné z: https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/#documentation

[13] *Speech-to-Text* [online]. Mountain View, California, United States: Google, 2021 [cit. 2021-9-6]. Dostupné z: https://cloud.google.com/speech-to-text

[14] *Watson Speech to Text* [online]. Armonk, New York, United States: IBM, 2021 [cit. 2021-9-6]. Dostupné z: https://www.ibm.com/cz-en/cloud/watson-speech-to-text

[15] *WIT.AI* [online]. Wit.ai, 2020 [cit. 2021-9-6]. Dostupné z: https://wit.ai/

[16] Lidský hlas. *Wikipedia* [online]. San Francisco, USA: nadace Wikimedia, 2022, 13. 4. 2022 [cit. 2022-06-18]. Dostupné z: https://cs.wikipedia.org/wiki/Hlas

[17] Mikrofon. *Wikipedia* [online]. San Francisco, USA: nadace Wikimedia, 2022, 13. 4. 2022 [cit. 2022-06-18]. Dostupné z: https://en.wikipedia.org/wiki/Microphone

[18] Neuronové sítě. *Wikipedia* [online]. San Francisco, USA: nadace Wikimedia, 2022, 13. 4. 2022 [cit. 2022-06-18]. Dostupné z: https://en.wikipedia.org/wiki/Artificial_neural_network

[19] Recurrent neural networks. *IBM* [online]. Armonk, New York, United States: IBM Cloud Education, 2020, 14.9.2020 [cit. 2022-07-02]. Dostupné z: https://www.ibm.com/cloud/learn/recurrent-neural-networks

[20] Mc-1un0-mini. *NEVEN.CZ* [online]. BRNO: NEVEN.CZ, 2022, 2.7.2022 [cit. 2022-07-02]. Dostupné z: https://www.neven.cz/kategorie/pocitace-a-kancelar/mikrofony/mc-1un0-mini-usb-pc-mikrofon/#cerna/

[21] ATGM2. *Audio-Technica* [online]. Machida, Tokyo, Japan: Audio-Technica, 2022, 2.7.2022 [cit. 2022-07-02]. Dostupné z: https://www.audio-technica.com/en-us/atgm2

[22] FLAC - introduction. *FLAC* [online]. Xiph.Org: Xiph.Org Foundation, 2022 [cit. 2022-07-28]. Dostupné z: https://xiph.org/flac/features.html

[23] Wav. *Wikipedia* [online]. San Francisco, USA: nadace Wikimedia, 2022, 28. 7. 2022 [cit. 2022-07-28]. Dostupné z: https://en.wikipedia.org/wiki/WAV

[24] Jupyter. *Jupyter* [online]. USA, 2022, 13. 9. 2022 [cit. 2022-09-13]. Dostupné z: https://jupyter.org/

Obsah CD

Kusy_BP.pdf obsahuje tuto práci, record1.sh a record2.sh jsou nahrávací skripty pro nahrávací část programu a vygenerují soubory nahravka, nahravka1 a nahravka2. Ovládací software robota je uložen v souboru Voice_recog.ipynb