Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

PLZEŇ, 2016 **DAVID BENEŠ**

Čestné prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne:_____ .....

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Luboši Šmídlovi, PhD. za poskytnutí cenných rad a vstřícný přístup během zpracování této práce.

Abstrakt

Tato práce řeší hlasové ovládání průzkumného robota. Problematika hlasového ovládání je řešena dvěma přístupy rozpoznávání mluvené řeči. První metoda je využití externího hlasového modulu, kterým lze zpracovávat hlasové příkazy a zvuky. Druhá metoda využívá více komplexního hlasového rozpoznávače SpeechCloud. SpeechCloud běží na samostatném serveru. Propojit se, se SpeechCloudem na běžícím serveru, je možné pomocí internetového připojení. Tato dvě řešení mají své pro a proti. Externí modul je možné používat s lehkou přenositelností a lze jej snadno přenastavit, ale disponuje mnoha nedostatky. Nevýhoda SpeechCloudu tkví v tom, že je potřeba připojení k internetu a dostupný server, na kterém běží. Nicméně disponuje mnoha možnostmi, jak vylepšit funkčnost a ovladatelnost. Výsledek této práce poukazuje na pohodlnější ovládání přístrojů, než pouze pomocí klávesnice či jiných fyzických zařízení.

Abstract

This work is trying to solve a problem with voice control of a robot. There are implemented two methods to solve a speech recognition. First method uses external voice module which is useful for voice recognition or some sound applications. Second method uses more complex speech recognition – SpeechCloud. This recognition engine is running on external server accessible through the internet. Both methods have pros and cons. External module is easy to use and is very portable but has a several disadvantages. On the other hand for access to SpeechCloud is important connection to the Internet. However there are several possible how to improve it. Result of this work shows how to use speech recognition as more comfortable approach than controlling a device with keyboard or any another physical controllers.

Obsah

1. Úvod....7
2. Hlasové ovládání modulem EasyVR....8
2.1. EasyVR modul....8
2.2 Komunikace....10
2.3. Trénování uživatelských hlasových příkazů....11
2.4. Ovládání robota pomocí modulu přes sériovou linku....15
2.5. Shrnutí....17
3. Hlasové ovládání SpeechCloudem....20
3.1. O SpeechCloudu....20
3.2. Náležitosti pro připojení ke SpeechCloudu....20
3.3. Správa událostí a metod....21
3.4. Tvorba gramatiky....23
3.5. Implementace do internetové stránky....27
3.6. Výhody a nevýhody ovládání pomocí SpeechCloudu....29
3.7. Shrnutí....29
4. Vlastnosti robota....30
4.1. Základní specifikace....30
4.2. Komunikační protokol....33
4.3. Senzory....37
4.4. Výkonná výpočetní část....39
4.5. Shrnutí vlastností robota....40
5. Vizuální rozhraní....42
6. Test....46
Závěr....47
Použitá literatura....48
Seznam obrázků, diagramů, kódů a tabulek....48
Seznam příloh....49

1. Úvod

Člověk od přírody využívá řeč jako hlavní způsob předávání informací. Pro každého je přirozené mluvit a tím dávat najevo naše potřeby. Přístroje ovšem standardně nedisponují sluchem, natož porozuměním řeči. Tuto oblast řeší obor zabývající se problematikou rozpoznávání mluvené řeči.

Hlasové ovládání slouží ke kontrole přístrojů a to bez nutnosti manuálního zadávání pomocí klávesnice nebo dotykového displeje. Díky tomu, že není nutné používat ruce k ovládání přístroje se může zvýšit bezpečnost obsluhy. Vzorový příklad může být GPS navigace v automobilech, kdy během diktování cílové adresy nemusíme na delší dobu pustit volant nebo přestat sledovat vozovku.

Hlasovým ovládáním lze zjednodušit i zadávání parametrů. Například pokud budeme chtít zadat cílovou destinaci pro GPS navigaci, musíme manuálně napsat adresu pomocí klávesnice. To může být velmi obtížné během jízdy. Na druhé straně, pomocí hlasového zadávání řekneme cílovou destinaci a klávesnice se nemusíme ani dotknout, což může být o poznání rychlejší a pohodlnější.

Cílem této práce je vytvoření hlasového ovládání pro průzkumného robota. Byly zvoleny dva různé přístupy, jak hlasového ovládání dosáhnout. Tyto dva přístupy mají různé způsoby zpracování výsledných dat, ale i zapojení a implementace.

2. Hlasové ovládání modulem EasyVR

2.1. EasyVR modul

Modul EasyVR[1] je multifunkční elektronický modul. Mezi jeho hlavní rysy patří rozpoznávat hlasové povely. Tyto povely je možné natrénovat nebo využít integrovaných povelů. Tyto povely jsou dostupné v různých jazycích: US angličtina, němčina, francouzština, italština, španělština, japonština. Implementovaných povelů je 26. Tyto povely se dělí do několika skupin.

		Jazyk					
Číslo skupiny	Index příkazu	US angličtina	Italština	Japonština	Němčina	Španělština	Francouzština
0	0	robot	robot	ロボット	roboter	robot	robot
1	0	action	azione	アクション	aktion	acción	action
	1	move	vai	進め	gehe	muévete	bouge
	2	turn	gira	曲がれ	wende	gira	tourne
	3	run	corri	走れ	lauf	corre	cours
	4	look	guarda	見ろ	schau	mira	regarde
	5	attack	attacca	攻撃	attacke	ataca	attaque
	6	stop	fermo	止まれ	halt	para	arrête
	7	hello	ciao	こんにちは	hallo	hola	salut
2	0	left	a sinistra	左	nach links	a la izquierda	à gauche
	1	right	a destra	右	nach rechts	a la derecha	à droite
	2	up	in alto	上	hinauf	arriba	vers le haut
	3	down	in basso	下	hinunter	abajo	vers le bas
	4	forward	avanti	前	vorwärts	adelante	en avant
	5	backward	indietro	後ろ	rückwärts	atrás	en arrière
3	0	zero	zero	ゼロ	null	cero	zéro
	1	one	uno	一	eins	uno	un
	2	two	due	二	zwei	dos	deux
	3	three	tre	三	drei	tres	trois
	4	four	quattro	四	vier	cuatro	quatre
	5	five	cinque	五	fünf	cinco	cinq
	6	six	sei	六	sechs	seis	six
	7	seven	sette	七	sieben	siete	sept
	8	eight	otto	八	acht	ocho	huit
	9	nine	nove	九	neun	nueve	neuf
	10	ten	dieci	十	zehn	diez	dix

Tabulka 2.1.1. – přehled implementovaných příkazů - přeloženo[2]

Jazykovou sadu lze v průběhu používání modulu kdykoliv změnit (posláním odpovídajícího příkazu přes sériovou linku). Zvolená testovací jazyková sada je v angličtině. Volba jazykové sady nemá žádný vliv na uživatelské příkazy.

1 Veear EasyVR 2.0 [online], dostupné z: http://www.veear.eu/products/old-products/easyvr/ [cit. 2016-05-02]
2 Veear EasyVR 2.0 User Manual 3.6.7 [online], dostupné z: http://www.veear.eu/files/easyvr_user_manual_3.6.7.pdf – strana 38 [cit. 2016-05-02]

Uživatelských příkazů může být až 32. Tyto příkazy je nutné natrénovat hlasovým vstupem (pouze hlasovým vstupem). To znamená, že největším vlivem trénování uživatelem může být prostředí ve kterém k trénování dochází. Ideální prostředí by měla být audio-komora.

Funkcionalita modulu zahrnuje i přehrávání zvukových záznamů (audio playback). Tyto záznamy mohou být jednoduché pokyny, pípání nebo odpověď na příkaz. Výstup audia je možný přímo z modulu a to připojením reproduktoru ideálně s impedancí 8Ω nebo větší (menší hodnota by mohla poškodit modul).

Maximální délka audio playbacku je ovlivněna zvolenou kompresí. Nejdelšího záznamu lze dosáhnout s maximální kompresí 8,7 minuty a nejkratšího záznamu s maximální kvalitou zvuku 38 sekund (tedy s minimální kompresí). Tato délka je společná pro všechny záznamy. V součtu všechny záznamy tedy nemohou mít délku větší, než 8,7 minuty. Maximální komprese je ideální pro jednoduché audio odpovědi (pípnutí). Pro hlasový záznam je vhodné využít nižší komprese.

Modul má další zajímavé možnosti využití mimo hlasového ovládání či audio playbacku.

- Lze ovládat 3piny na modulu (nastavením jako vstupu/výstupu).
- Generátor tónového výstupu
- Bezdrátové ovládání zvukem (jedná se o ovládání danou zvukovou frekvencí).

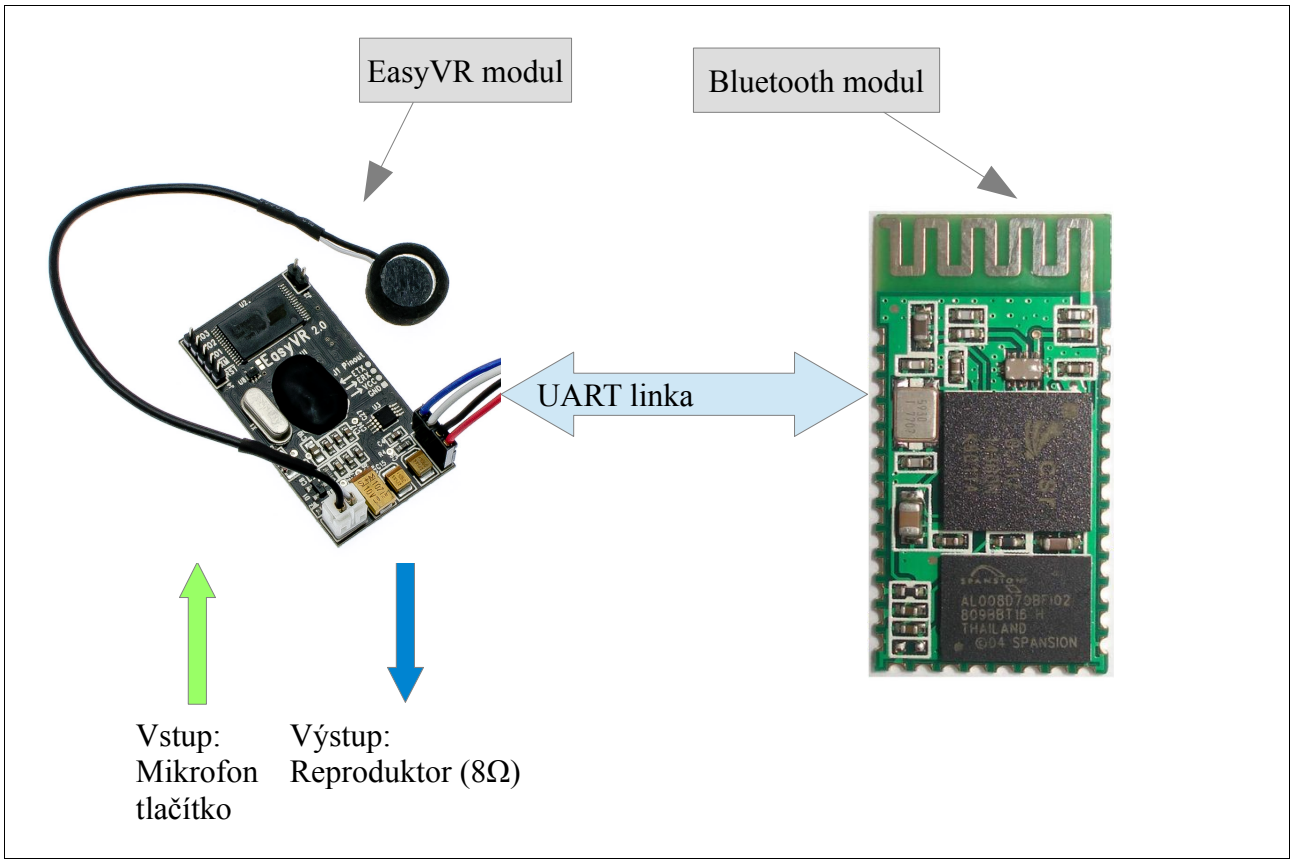
Modul komunikuje pomocí sériové linky, která je ve výchozím nastavení s parametry: 9600bps 8-N-1 [baudrate, počet bitů-parita-stop bit].

Modul využívá přepis natrénovaných příkazů do matematické podoby (která není oficiálně k dispozici). Implementované příkazy dosahují mnohem lepších kvalit rozpoznání, než lze dosáhnout pomocí uživatelsky definovaných.

Hlavní nevýhoda je, že špatně rozpozná podobně znějící příkazy. Příkladem mohou být příkazy: 'zatoč doleva' a 'zatoč doprava'. Tyto dva příkazy nelze spolehlivě využít v jedné skupině (o skupinách v dalších podkapitolách). Důvod je zjevný, tyto dva příkazy jsou totiž stejné rozsahu „zatoč do“ a liší se jen „leva“ nebo „prava“. Tedy minimálně polovina příkazu je stejná. Možností je využít jiných intonací hlasu pro každý příkaz. Toto ale může být velmi obtížné pro uživatele, který není na tento „styl“ mluvy trénován. Je to velmi nepohodlné.

Modul mnohem lépe správně vyhodnotí implementované příkazy (zkoušena pouze anglická jazyková sada z důvodu lépe zažitého přízvuku). Pravděpodobné důvody lepšího výsledku: příkazy jsou natrénované za mnohem lepších podmínek a natrénovaná data mohou být posléze vyčištěna a vylepšena softwarovou cestou. Neposledním rozdílem je fakt, že v angličtině příkazy „left“

a „right“ už od poslechu znějí rozdílně. Oproti tomu příkazy v české podobě „doleva“ a „doprava“ zní velmi podobně. Je možné využít „vlevo“ a „doprava“, ale to opět způsobuje problematiku 'znalosti' řečníka.



Obrázek 2.1.1. – zapojení modulu EasyVR[3]

2.2 Komunikace

Datová komunikace s modulem je možná pouze pomocí sériové linky. Modul využívá komunikační protokol založený na tabulkových příkazech. Každý příkaz má pevně danou strukturu. Komunikace využívá pouze tisknutelných ASCII znaků. Na každý příkaz odeslaný do modulu je odeslána odpověď. Odpověď závisí na odeslaném příkazu.

Komunikace je řízena serverem (hostujícím počítačem/mikrokontrolérem). Modul tedy nekomunikuje pokud není „požádán“. Každý bajt odeslán z modulu musí být potvrzen serverem (potvrzení je hodnota 0x20 = 0d32[4] –> mezera). Pokud modul přijme neočekávanou kombinaci příkazů nebo chybná data, odešle informace o chybě.

3 Zdroj obrázku modulu EasyVR [online] dostupné z: http://www.veear.eu/wp/wp-content/uploads/veear.eu/2013/11/EasyVR2.2_-3.jpg [cit. 2016-05-02].
Zdroj obrázku bluetooth modulu [online] dostupné z: http://wiki.pinguino.cc/images/8/84/HC-06_module_pinout.jpg [cit. 2016-05-02].

4 Zápis v podobě 0xXY odpovídá hexadeciální hodnotě a zápis v pododě 0dXY odpovídá decimální hodnotě.

Příkaz je definován jedním bajtem s hodnotou od 0x61 do 0x7A (v tisknutelných ASCII znacích 'a' ↔ 'z'), za kterým následují argumenty (může se jednat o několik bajtů nebo nemusí být žádný). Hodnoty bajtů v argumentu musí být normalizovány. Modul normalizuje hodnoty na tisknutelné ASCII znaky. Normalizace funguje přičtením hodnoty 0d65 k požadované hodnotě bajtu. Maximální „vstupní“ hodnota může být 31 a minimální hodnota -1. To znamená, že k odeslání hodnoty -1 je potřeba poslat znak '@' a k odeslání hodnoty 31 poslat znak '`'. Pro ujasnění je v dokumentaci k modulu uvedena tabulka.

ASCII	'@'	'A'	'B'	'C'	---	')'	'_'	'`'
HEX	0x40	0x41	0x42	0x43	---	0x5E	0x5F	0x60
Hodnota	-1	0	1	2	---	29	30	31

Tabulka 2.2.1. – vizualizace normalizace hodnot v argumentu[5]

Potřebné příkazy k využití modulu pro jednoduché hlasové ovládání robota jsou zobrazeny v následující tabulce:

Příkaz	Bytová podoba příkazu	Význam příkazu
STOP	CMD_BREAK: ['b' - 0x62]	Zastaví právě probíhající operace: • trénování příkazu • přehrávání audia • přerušení rozpoznávání Odpovědi modulu: • STS_SUCCESS • STS_INTERR
NASTAV_JAZYK	CMD_LANGUAGE: ['l' – 0x6C; 0-5 (volba jazyka)]	Nastaví zvolenou jazykovou sadu (implementovaných příkazů). Odpovědi modulu: • STS_SUCCESS
ROZP_PRIKAZ	CMD_RECOG_SD: ['u' – 0x75; 0-16 (volba skupiny)]	Spustí rozpoznávání uživatelsky natrénovaných příkazů. Odpovědi modulu: • STS_RESULT • STS_SIMILAR • STS_TIMEOUT • STS_ERROR
ROZP_SI_PRIKAZ	CMD_RECOG_SI: ['i' – 0x6F; 0-3 (volba skupiny)]	Spustí rozpoznávání implementovaných příkazů. Odpovědi modulu: • STS_SIMILAR • STS_TIMEOUT • STS_ERROR

5 Veear EasyVR 2.0 User Manual 3.6.7 [online], dostupné z: http://www.veear.eu/files/easyvr_user_manual_3.6.7.pdf – strana 24 [cit. 2016-05-02]

HRAJ_ZVUK	CMD_PLAY_SX: ['w' – 0x77; 0-1024 (dva byty – index na zvukovou stopu); 0-31 (hlasitost)]	Přehraje audio záznam. Index odpovídá pozici audio stopy. Odpovědi modulu: • STS_SUCCESS
PIN_STATUS	CMD_QUERY_IO: ['q' – 0x71; 1-3 (index pinu); 0-4 (pin mode – input/output)]	Nastaví nebo přečte stav zvoleného pinu. Lze nastavit výstup H/L nebo přečíst vstup se zvolenou hodnotu pull-up rezistoru. Odpovědi modulu: • STS_SUCCESS • STS_PIN

Tabulka 2.2.2. – seznam příkazů potřebné pro hlasové rozpoznávání (ovládání)

Odpovědi na jednotlivé příkazy mohou nabývat různých významů. Může se jednat o celočíselnou hodnotu udávající například index rozpoznaného příkazu nebo o celočíselnou hodnotu udávající důvod chyby:

Odpověď	Bytová podoba odpovědi	Význam odpovědi
STS_SUCCESS	['o' – 0x6F]	Vše v pořádku. Průběh úspěšný.
STS_INTERR	['i' – 0x69]	Přerušení příkazem CMD_BREAK.
STS_RESULT	['r' – 0x72; 0-31 (Číselná hodnota, která je indexem na hlasový příkaz)]	Číselná hodnota rozpoznaného hlasového vstupu. Číslo odpovídá indexu hlasového příkazu ve zvolené skupině.
STS_SIMILAR	['s' – 0x73; 0-31 (Číselná hodnota, která je indexem na hlasový příkaz)]	Rozpoznaný hlasový příkaz je podobný hlasovému příkazu na daném indexu.
STS_TIMEOUT	['t' – 0x74]	Vypršel čas, který vymezuje dobu pro hlasové rozpoznání. (Čas lze změnit nastavením)
STS_ERROR	['e' – 0x65; 0-256 (dva byty označující hodnotu chyby)]	Chyba. Hodnoty jednotlivých chyb a jejich význam: • 0x03 – Příliš rušné prostředí pro vyhodnocení vstupu • 0x04 – Hlasový projev byl příliš potichu • 0x05 – Hlasový projev byl příliš hlasitý • 0x06 – Hlasový projev byl vyřčen příliš brzy • 0x07 – Hlasový projev byl příliš komplexní (příliš mnoho slov). • 0x11 – Rozpoznání bylo neúspěšné (například vyřčeno slovo co není ve skupině) • 0x12 – Pravděpodobnost výsledku správného rozpoznání je příliš nízká. • 0x13 - Pravděpodobnost výsledku správného rozpoznání 50 na 50.

		• 0x14 – chyba příkazu uloženého v paměti. • 0x17 – chyba délky příkazu • 0x4A – neznámé číslo audio playbacku • 0x4E – neznámá komprese nebo poškozená data v audio playbacku • 0x80 – Neznámé slovo
STS_PIN	['p' – 0x70; 0-1 (vstupní hodnota H/L)]	Návratová hodnota stavu pinu. Hodnota 0 odpovídá uzemnění (0V) a hodnota 1 odpovídá napětí (VCC).

Tabulka 2.2.3. – seznam odpovědí na definované příkazy

2.3. Trénování uživatelských hlasových příkazů

Přidání uživatelských hlasových příkazů do paměti modulu je možné dvěma způsoby. První způsob lze využít, pokud je modul již implementován v nějakém zařízení a nelze se k němu připojit pomocí sériové linky. Druhý způsob, který je mnohem přehlednější díky GUI[6], je připojení modulu k PC pomocí sériové linky[7] a využitím freeware softwaru EasyVR Commander od firmy VeeaR. Tento software je dostupný na stránkách firmy http://www.veear.eu/downloads/.

Nastavení modulu bylo provedeno druhou metodou z důvodu lepší kontroly modulu. Při přidání nového hlasového příkazu do zvolené skupiny je nutné tento příkaz natrénovat. Trénování probíhá pomocí mikrofonu, který je zapojen přímo do modulu. Před trénováním je vhodné nastavit funkční požadované parametry, kterými jsou vzdálenost mikrofonu (modulu) od řečníka (1), přesnost správného rozpoznání příkazu (2) a verifikace řečníka (3).

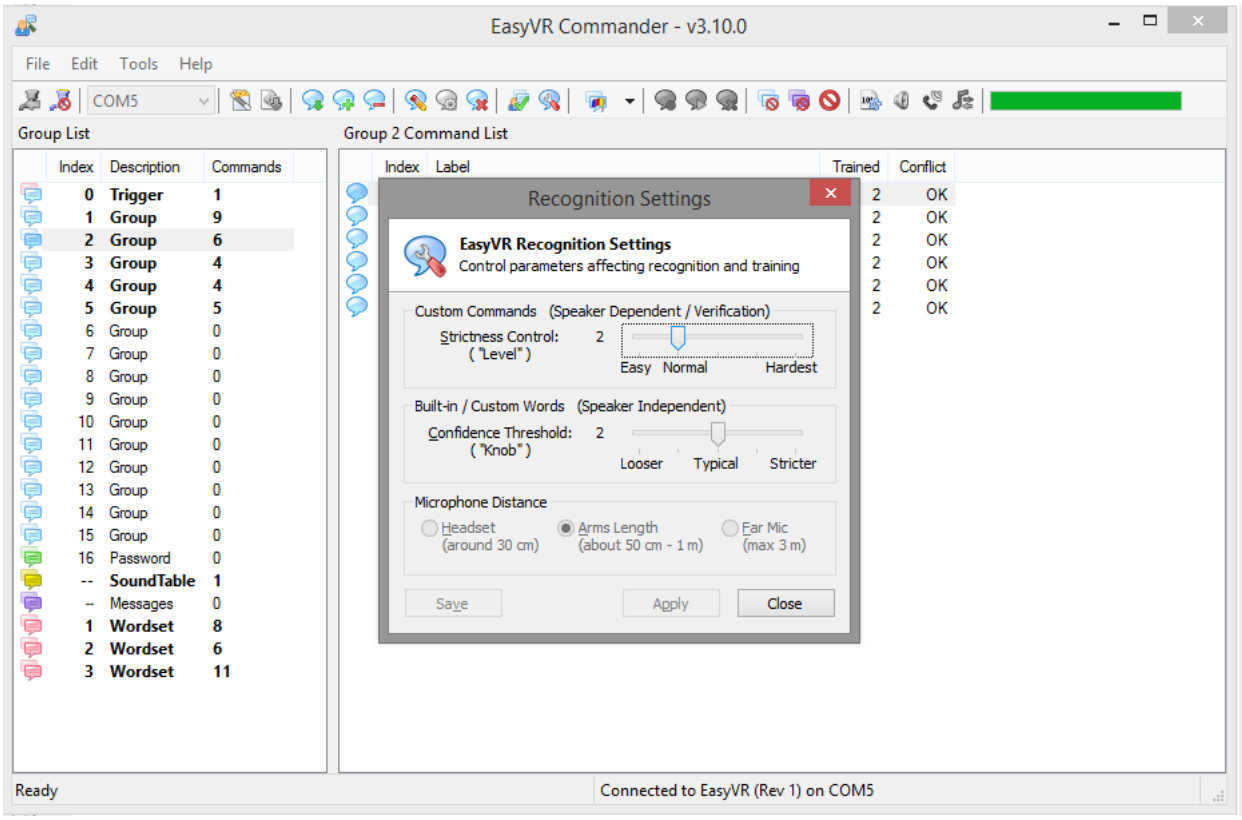
- (1) – jedná se o přibližnou vzdálenost mikrofonu od řečníka
- (2) – Rozpoznání probíhá na základě matematického porovnání hlasového vstupu. Tento parametr nám tedy v jistém smyslu udává, jak moc velké chyby se může modul dopustit při rozpoznávání příkazu.
- (3) – Verifikace řečníka je ovlivnění úrovně „ověření“ stejné osoby, jako té která daný hlasový příkaz natrénovala.

  Z testování nastavení nejlépe vycházejí výchozí hodnoty.

6 GUI – Graphical User Interface (Grafické uživatelské rozhraní).

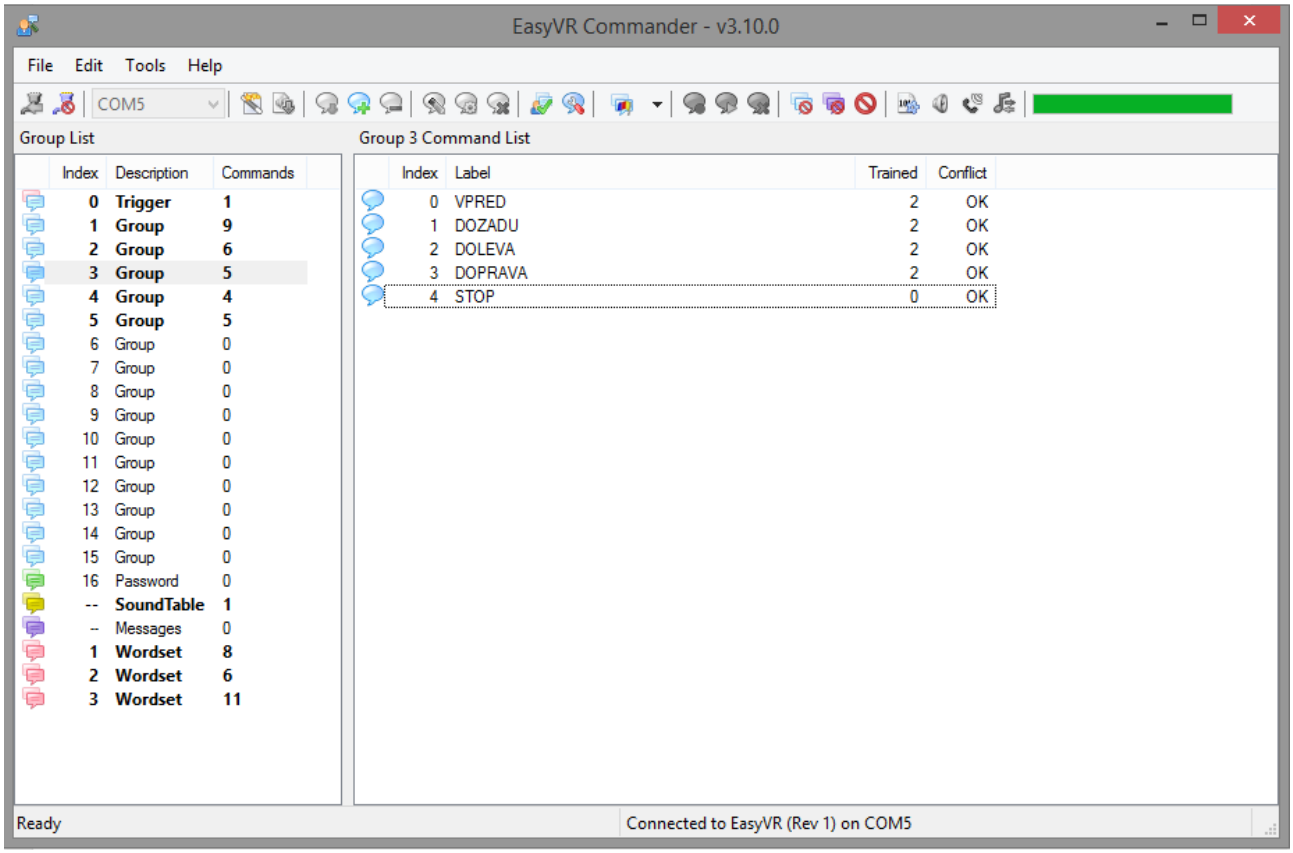
7 Připojit lze pomocí bluetooth sériové linky nebo za pomocí standardu RS232 a vhodného napěťového převodníku TTL úrovní, poslední varianta je využití jiného zařízení s převodníkem USB<>sériová linka a využít jej jako přemostění.

Program EasyVR Commander obsahuje přehled všech skupin. Každá skupina může obsahovat až 32 příkazů (ovšem v součtu všech příkazů ze všech skupin může být jen 32). Pro vytvoření nového příkazu stačí zadat vhodné pojmenování (pro lepší přehlednost je dobré názvem co nejlépe vystihnout příkaz) a poté natrénovat hlasovým vstupem. Během trénování zažádá program o hlasový vstup dvakrát. Dvojím vstupem při trénování se zlepší přesnost rozpoznávání a zamezí se chybnému projevu.



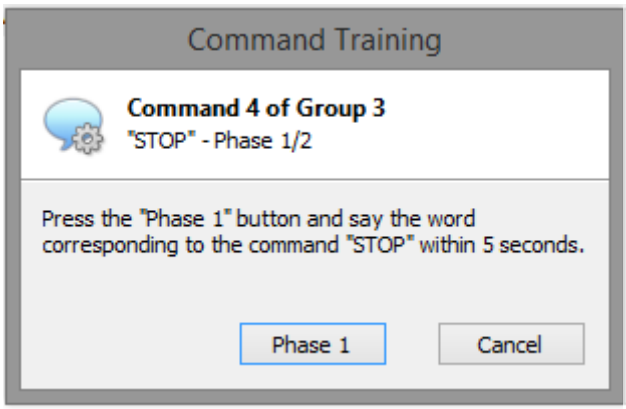
Obrázek 2.3.1. – nastavení funkčních parametrů

Následující obrázek zobrazuje obsah skupiny s indexem číslo 3 (dále jen skupina3). V této skupině jsou již natrénované příkazy: „vpřed“, „dozadu“, „doleva“, „doprava“ a příkaz „stop“, který ještě není natrénován.

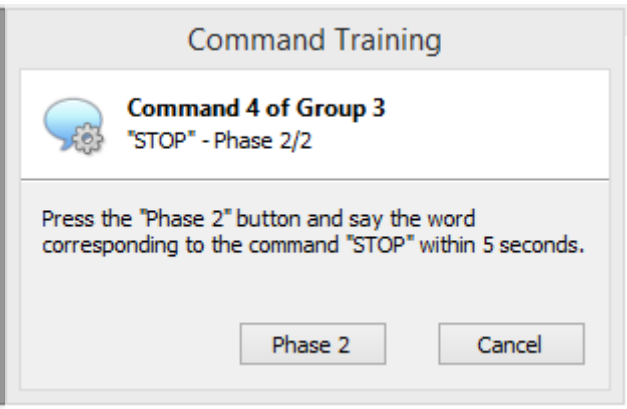


Obrázek 2.3.2. – skupina3 ukázka příkazů

Natrénování příkazu se spustí pomocí: vybráním daného příkazu a Edit – Train command (nebo vybráním příkazu a stisknutím enter).



Obrázek 2.3.3. – první fáze trénování příkazu STOP



Obrázek 2.3.4. – druhá fáze trénování příkazu STOP

Za předpokladu, že obě dvě fáze prošly trénováním bez chyby, máme přidaný nový příkaz do skupiny3. Pro ověření správné činnosti skupiny, můžeme spustit testovací režim (tools – test group) nebo pomocí (shift + enter). Po spuštění řekneme jeden z natrénovaných příkazů a program nám zobrazí námi vyřčený příkaz (pokud se nevyskytne chyba).

Pokud bychom chtěli příkaz natrénovat bez EasyVR Commanderu je nutné poslat přes sériovou linku potřebné příkazy pro přidání nového příkazu a poté stejným způsobem natrénovat.

Tento způsob je náročnější z důvodu chybějícího grafického rozhraní.

Všechny uživatelské příkazy, které jsou natrénovány v modulu a jsou připraveny k použití, jsou zapsány v následující tabulce.

Skupina	Číslo a název příkazu ve skupině	Příkaz
1	0 - VPRED	„vpřed“
	1 - COUVEJ	„couvej“
	2 - ZATOC_DOPRAVA	„zatoč doprava“
	3 - ZATOC_DOLEVA	„zatoč doleva“
	4 - KAMERU_DOLEVA	„kameru otoč doleva“
	5 - KAMERU_DOPRAVA	„kameru otoč doprava“
	6 - PODIVEJ_SE_NAHORU	„podívej se nahoru“
	7 - VYCENTRUJ_KAMERU	„vycentruj kameru“
	8 - PORID_FOTKU	„pořiď fotografii“
2	0 - JED	„jeď“
	1 - NATOC_KAMERU	„natoč kameru“
	2 - PORID_FOTKU	„pořiď fotku“
	3 - VYCENTRUJ_KAMERU	„vycentruj kameru“
	4 - OTOC_SE	„otoč se“
	5 - ZASTAV	„zastav“
3	0 - VPRED	„vpřed“
	1 - DOZADU	„dozadu“
	2 - DOLEVA	„doleva“
	3 - DOPRAVA	„doprava“
4	0 - DOLU	„dolů“
	1 - NAHORU	„nahoru“
	2 - DOLEVA	„doleva“
	3 - DOPRAVA	„doprava“

Tabulka 2.3.1. – natrénované příkazy

Pozn.: Skupina1 je velmi obtížné správně použít, protože docházelo k časté chybě během trénování z důvodu velmi podobného příkazu. Například příkazy 4-KAMERU_DOLEVA a 5-KAMERU_DOPRAVA byly často označeny jako velmi podobné příkazy a při testování docházelo k chybnému výsledku rozpoznávání. Proto bylo nutné dát rozdílný důraz na jistou slabiku v příkaze. Z tohoto důvodu je prospěšnější použít samostatné skupiny a příkazy řetězit.

Např.: Pro rozpoznání směru jízdy – Skupina2 → Skupina3.

2.4. Ovládání robota pomocí modulu přes sériovou linku

Po natrénování potřebných příkazů a otestování jejich integrity (funkčnosti), lze modul zapojit do robota. Modul je připojen k Raspberry Pi 2[8] (dále jen Rpi) pomocí bluetooth[9] modulu. Hlavní důvod je ten, že sériová linka na Rpi je již obsazena pro komunikaci s řídícím mikrokontrolérem robota. Vedlejší výhoda připojení bezdrátově je absence kabelů, což znamená pohodlnější ovládání.

Pomocí bluetooth modulu je EasyVR modul připojen k Rpi přes bluetooth rozhraní. Na výslednou funkčnost to nemá žádný vliv a v případě potřeby lze modul připojit na libovolný počítač vybavený bluetooth a přenastavit modul pomocí easyVR Commanderu.

Vzorový příklad komunikace s modulem je proces na rozpoznání čísla (od 0 do 999999999). Modul bude vyzýván k rozpoznání ze skupiny3 (implementovaných příkazů). Cyklus se přeruší, pokud dojde k „nečekané“ chybě nebo uběhne timeout (řečník již nediktuje čísla). Po každém rozpoznaném čísle je modul vyzván k přehrání zvuku pípnutí, aby bylo možné detekovat, kdy lze říci další číslo. Příklad je pro lepší ilustraci psán pseudokódem.

Rozpoznávání tedy probíhá po jednotlivých příkazech, které lze řetězit za sebou dle potřeby. Je ovšem nutné si uvědomit, že čím více příkazů bude potřeba rozpoznat, tím větších chyb se můžeme dopustit a celý segment příkazů bude náročnější na čas diktování.

Aby bylo možné kontrolovat robota pomocí EasyVR modulu, je nutné nějakým způsobem informovat server (v tomto případě Rpi), že bude probíhat promluva od řečníka. Z tohoto důvodu Rpi odesílá periodicky dotaz na stav pinu IO1 EasyVR modulu s frekvencí 2Hz. V momentě, kdy dostane odpověď v podobě stavu senzoru L – 0V, tak se spustí hlasové rozpoznání skupiny2 (uživatelsky definovaných příkazů). Na základě výsledku rozpoznání slova ze skupiny2 se aktivuje rozpoznání v odpovídající skupině, která je určena na základě slova, které bylo detekováno. Následující diagram zobrazuje rutinu pro hlasové ovládání.

8 Detailnější informace o Rpi2 [online] dostupné z: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [cit. 2016-05-02]

9 Bluetooth je otevřený standard nahrazující drátové rozhraní RS-232 ~ jedná se o bezdrátovou sériovou linku.

```
otoceni=0
pole=[]
while (otoceni<9){
        uart.send(0x69)         //Pošli požadavek na rozpoznání SI příkazu
        uart.send(3+0x41)       //Skupina2 čísla („zero“, „one“, „two“, ...)
        znak=uart.receive()     //Podle nastaveného timeoutu (do 5s) přijde odpověď
        if(znak==0x72 or znak==0x73){ //Kontrola úspěšného rozpoznání
                uart.send(0x20)//ACK byte
                pole.append(uart.receive()-0x41)
                otoceni++
                uart.send(0x77)         //Přehraj zvuk - „beep“
                uart.send(0x41)
                uart.send(0x41)
                uart.send(0x51)
        }else   //Chyba
                break
}
if(znak==0x74){//Timeout konec sekvence diktovaných čísel
        process(pole)   //Zpracuj pole, které obsahuje jednotlivá čísla
}
else if(znak==0x65){
        error=(uart.receive()-0x41)*16   //Příjem vyššího bytu
        uart.send(0x20)         //ACK byte
        error+=(uart.receive()-0x41)     //Příjem nižšího bytu
}
```

Kód 2.4.1. – vzorový příklad rozpoznávání

V případě chyby, kdy modul nerozpozná příkaz nebo dojde k timeoutu atd. neprovede se žádná reakce a Rpi skočí na začátek rutiny (chyba je oznámena trojitým pípnutím). Každá úspěšně rozpoznaná hlasová promluva nebo aktivace rozpoznávání je oznámena jedním pípnutím.

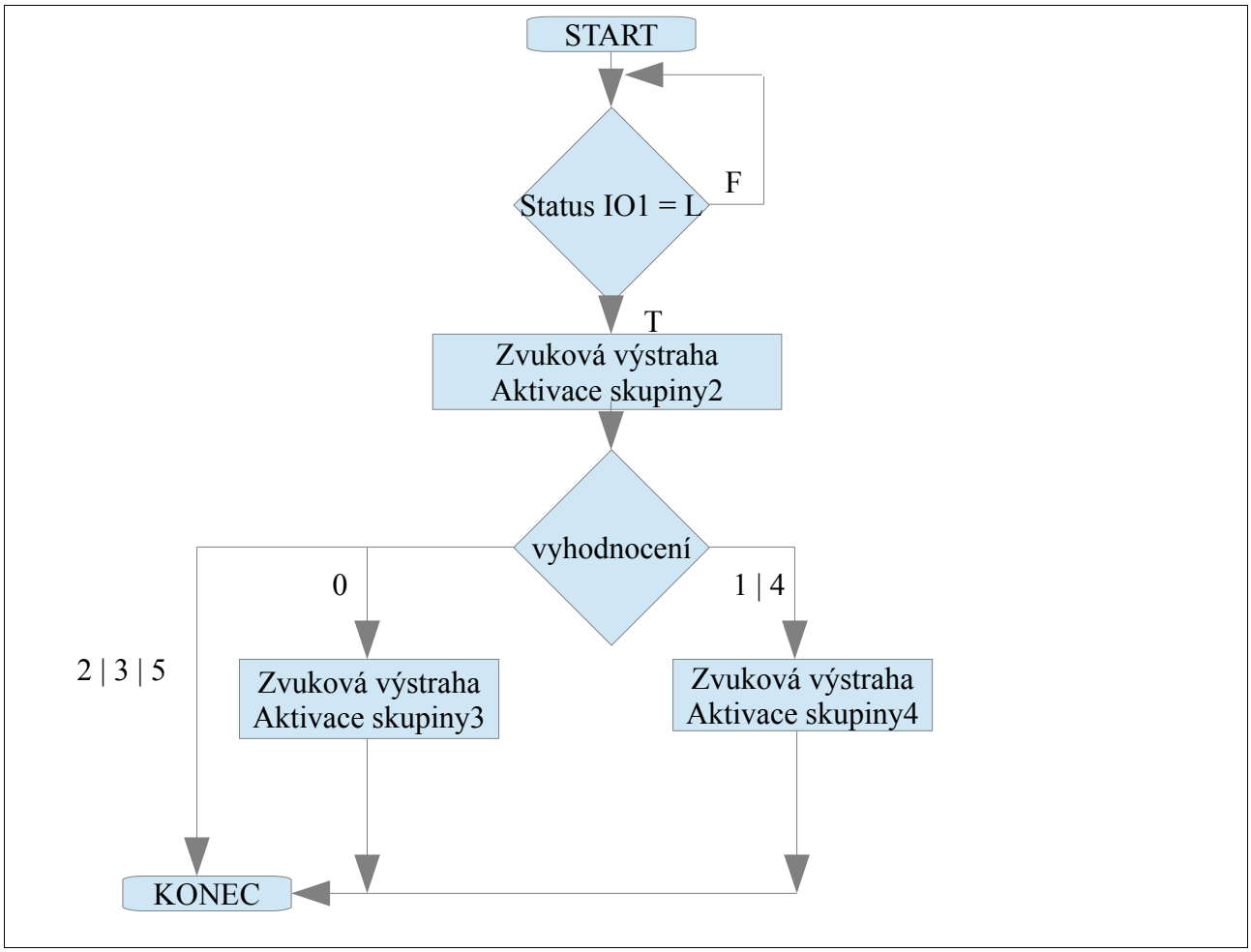


Diagram 2.4.1. – rutina zjišťování stavu modulu

2.5. Shrnutí

Hlasový modul easyVR je velmi zajímavým samostatným modulem pro hlasové ovládání, ale i pro jiný účel v oblasti zvukových aplikací. Jediným problémem je nepřesnost naučených příkazů a to hlavně z důvodu podobnosti některých příkazů. Tento modul je vhodný jako jednoduchý hlasově interaktivní přístup k automatizaci různých zařízení.

3. Hlasové ovládání SpeechCloudem

3.1. O SpeechCloudu

SpeechCloud je systém, který propojuje několik systémů operujících dohromady přes jedno rozhraní. Sjednocuje: hlasové rozpoznávání (ASR), syntézu řeči (TTS) a porozumění řeči (SED). Připojení na SpeechCloud je navázáno pomocí RTP a SIP protokolů. Celý systém je funkční přes internetové připojení. To znamená existenci klienta a serveru. Klient je zařízení (počítač), který má aktivní audio vstup a server je zařízení zpracovávající příchozí příkazy a audio stream.

Správu všech funkcí a kontrolu připojení k SpeechCloudu je řízeno pomocí JavaScriptu. K tomuto slouží knihovna SpeechCloud.js. Tato knihovna obsahuje několik metod a událostí, které lze a nebo je nutné využít. SpeechCloud byl vytvořen na katedře kybernetiky ZČU v Plzni ve spolupráci s firmou SpeechTech s.r.o[10].

3.2. Náležitosti pro připojení ke SpeechCloudu

SpeechCloud poskytuje stejné služby všem klientům, pokud není omezeno či určeno jinak. Každý klient by měl mít k dispozici stejné funkce. Nicméně každé zařízení, experiment nebo projekt by měl být separován od ostatních, aby nedocházelo k ovlivňování výsledků a přenastavování parametrů nechtěným zásahem. Tohoto je docíleno různými 'aplikacemi' na straně serveru. V našem případě je SpeechCloud připojován na:

- URI https://cak.zcu.cz:9443/v1/speechcloud/edu-benes.

Při otevření webové stránky dostaneme jako odpověď JSON schéma hodnot a parametrů, které jsou zapotřebí pro navázání komunikace a funkčnosti připojení. Nejdůležitější hodnoty jsou sip_* a client_*. Proměnné sip_* obsahují potřebné údaje k navázání komunikace pomocí SIP (Session Initiation Protocol – inicializační relační protokol). Tento protokol je využíván pro internetovou hlasovou komunikaci. Proměnné client_* jsou údaje pro připojení na websocket sloužící k předávání zpráv, nastavení, … .

```
{
   "client_id": ID klientského zařízení,
   "client_wss": šifrovaný WebSocket server-klient,
   "sip_password": heslo pro připojení SIP,
   "sip_uri": URI pro přístup k SIP,
   "sip_username": Přihlašovací jméno SIP,
   "sip_wss": šifrovaný WebSocket k přístupu na SIP
}
```

Problematika řešení připojení a předávání informací přes internetové komunikační protokoly není obsahem této práce. Řešen je vývojářský problém s hotovými softwarovými komponentami.

Pro účel navázání a správu připojení ke SpeechCloudu je univerzitou poskytnuta JavaScriptová knihovna SpeechCloud.js. Pro vytvoření instance SpeechCloudu je zapotřebí URI projektu. Po inicializaci a úspěšném připojení bychom měli mít k dispozici metody pro ovládání hlasového rozpoznávače a události, na které lze patřičně reagovat.

3.3. Správa událostí a metod

Jednotlivé metody a události jsou v následující tabulce. Vypsané jsou pouze ty, které byly využity pro vypracování této práce. Vstup metody nebo výstupní hodnota události jsou v JSON formátu.

Metody		
Název metody:	Vstup:	Popis:
asr_pause()	---	Pozastavení rozpoznávání. Pozastavení během promluvy se považuje jako konec příkazu.
asr_recognize()	---	Spuštění rozpoznávání. Rozpoznávání (audio stream) běží do doby, než je pozastaveno nebo když nastane chyba.
asr_process_text()	{text: String}	Zpracování textu (String) jako hlasový vstup. Vhodné pro ladění a testování. Výsledný program tuto metodu nevyužívá.
Události		
Název události:	Výstupní hodnota:	Popis:
asr_ready	---	Úspěšné připojení na SpeechCloud. Všechny potřebné protokoly jsou aktivní (SIP, WSS, RTP)
asr_signal	{speech: Bool, level: Integer}	Síla audio signálu. Hodnota speech udává, jedná-li se hlas nebo šum.
asr_result	{result: String}	Návratová hodnota výsledku hlasového rozpoznání.
sc_error	{error: String}	Návratová hodnota při chybovém stavu.
asr_set_grammar_ok	---	Úspěšné nastavení gramatiky.
asr_set_grammar_error	{error: String}	Nastavení gramatiky selhalo. Návratová hodnota obsahuje název/důvod chyby.

Tabulka 3.3.1. – seznam zpracovávaných metod a událostí

Po úspěšné inicializaci SpeechCloudu dojde k aktivaci události asr_ready. Vzhledem k tomu, že se využívá gramatika během rozpoznávání, tak se při této události odešle odkaz na gramatiku. Gramatika může být implementována přímo v programu (v html-JavaScript) nebo na jiném dostupném webovém úložišti (jednoduché stažení – bez ověření).

Metody asr_pause a asr_recognize nemohou být aktivní současně. Respektive jedna metoda vylučuje druhou. Existují dvě základní varianty, jak tyto metody využít:

1. Push-to-talk (stiskni a mluv): Jedním tlačítkem, kterým se při stisku aktivuje metoda asr_recognize a při uvolnění se aktivuje metoda asr_pause. To znamená, že pro aktivaci a úspěšné hlasové rozpoznání je nutné celou po celou dobu diktování držet tlačítko stisknuté a poté uvolnit.
2. Switch On/Off (zapni/vypni): Pro tuto metodiku je zapotřebí vytvořit přepínač, který má dvě polohy: zapnuto a vypnuto. Při přepnutí do zapnutého stavu se aktivuje metoda asr_recognize. Při přepnutí do vypnutého stavu se aktivuje metoda asr_pause.

Druhý přístup je výhodný z hlediska jednodušší manipulace, ale dosahuje větší chybovosti při rozpoznávání. Celý koncept mluvy totiž kontroluje SIP server a při detekci mluvy vrací výsledky rozpoznávání - i když se jedná o hluk nebo šum. Pokud nedojde k nalezení výplňového slova průchodem gramatikou (specifická slova, která jsou podobná ostatním slovům, ale nejsou určena pro řízení), tak se výsledek vyhodnotí.

Událost asr_ready je aktivována pouze jednou od připojení ke SpeechCloudu a to tehdy, kdy se klient připojí a odešle prvních pár audio fragmentů. Z tohoto důvodu je výhodné využít reakce na tuto událost pro inicializační nastavení ostatních komponent, příkladem může být odeslání požadavku pro načtení gramatiky.

Událost asr_signal není z funkčního hlediska nijak významná. Je to prvek, který informuje uživatele o aktivním audio přenosu. Na základě hodnoty přijatých touto události lze stanovit, zda-li funguje správně mikrofon či jiné zařízení nahrávající audio.

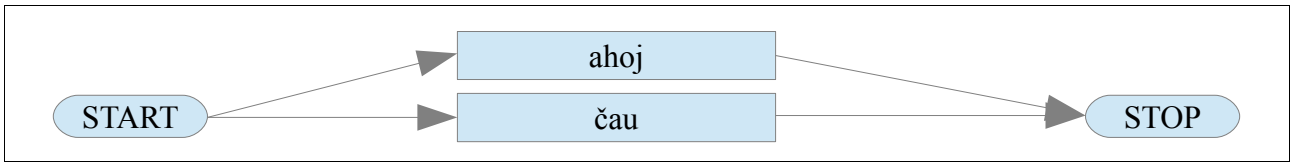
Událost asr_result vrací hodnotu v podobě Stringu. Jedná se o výsledek rozpoznání. Příkladem návratové hodnoty může být: „Jeď[GO] rovně[FORWARD] pět[#5] metrů[M]“. Slova nebo znaky v hranatých závorkách „[]“ je takzvaný tag. Tagy jsou využité k určení významu slova, aby bylo jednodušší vyhodnotit návratový String. Tag by měl být tedy unikátní pro jednu skupinu slov, které mají stejný význam například: „vpřed[FORWARD]“, „rovně[FORWARD]“, „dopředu[FORWARD]“, … .

Událost sc_error při výskytu chyby vrací informaci o daném problému. Uživatel by se neměl touto problematikou zabývat, za předpokladu, že je vše správně nakonfigurováno a nevznikne chyba při komunikaci (výpadek internetového připojení). Diagnostikou návratové informace o chybě lze stanovit chybnou část kódu.

Událost asr_set_grammar_ok a asr_set_grammar_error informují o stavu požadavku pro nastavení gramatiky. Událost asr_set_grammar_ok informuje o bezproblémovém odeslání požadavku a zavedení gramatiky. Druhá událost asr_set_grammar_error vrací chybovou hlášku, když se vyskytne problém s gramatikou (nedosažitelný cíl, chyba v gramatice, špatné kódování textu, …).

3.4. Tvorba gramatiky

Gramatika je sestavení primitiv do komplexních skupin, které je poté možné zřetězit a vytvořit rozhodovací strom, který je procházen při rozhodování, aby se určil výsledek hlasového vstupu. Primitiva jsou v tomto případě jednotlivá písmenka latinské abecedy (ABCD....) a komplexnější skupiny jsou samostatná slova.

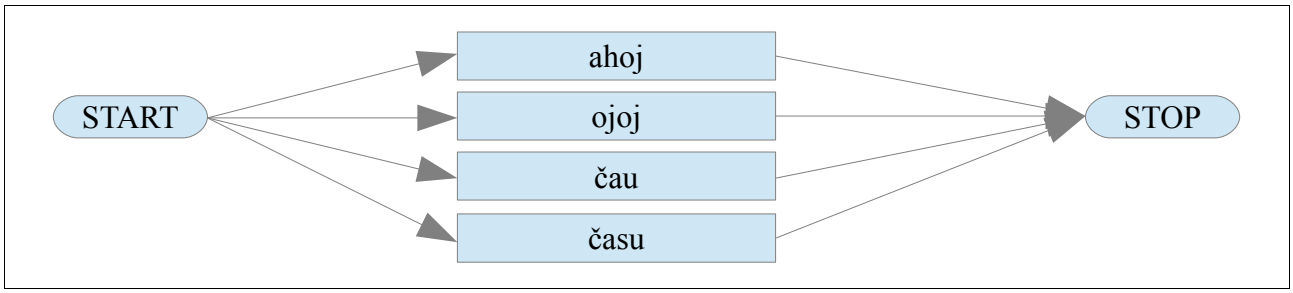


Obrázek 3.3.1. – grafické znázornění jednoduché gramatiky

Pokud budeme mít gramatiku (obrázek 3.3.1) obsahující dvě slova, například slovo „ahoj“ a „čau“, tak při rozpoznávání algoritmus vyhodnocuje pravděpodobnost vyřčení jednotlivých slov a poté asr_result (viz. 3.3) vrátí slovo s větší pravděpodobností.

Problémem může být hluk nebo šum, které stěžují správné rozpoznávání. V tomto případě můžeme dostat neodpovídající výsledek. Jiná možnost je, že je aktivované rozpoznávání, ale řečník je potichu. Přesto rozpoznávač detekuje hlas (například mluvící osoba v dálce) a na základě tohoto audio vstupu vyhodnotí výsledek. Obě dvě situace jsou nežádoucí hlavně proto, že by mohly způsobit nevyhovující reakci hlasem řízeného systému.

Obě dvě situace lze relativně jednoduše vyřešit přidáním výplňových slov, které nemají vliv na řízení (budou ignorována). Tato slova by měla být podobná slovům v gramatice (ne příliš, aby se výsledek ještě nezhoršil). V případě nutnosti lze přidat jedno/dvě slova, která jsou úplně odlišná, aby se odstranily špatné výsledky. Úpravou dostaneme novou spolehlivější gramatiku (obrázek 3.3.2).



Obrázek 3.4.2. – upravená jednoduchá gramatika

Gramatika pro řízení robota obsahuje několik uzlů, které dělí požadované cíle příkazů. Hlasem lze ovládat:

- otáčení „hlavy“, na které je umístěna kamera
- příkaz o vytvoření fotografie
- ovládat pohyb robota

Jedná-li se o příkaz pro pohyb nebo otáčení lze dodat ještě o kolik stupňů či o kolik centimetrů/metrů se má robot otočit nebo jet. V případě ovládání otáčení kamery se jedná o otáčení v úhlech pro danou osu.

Gramatika je tedy nutnou součástí, aby koncept hlasového ovládání fungoval. Každé slovo v gramatice (mimo předložky nebo spojky) pro ovládání robota je opatřeno tagem, aby bylo jednodušší zpracování výsledku a aktivace požadovaných funkcí.

Český jazyk je velmi bohatý co se týče využití různých slov nebo slovních spojení k vyjádření stejné myšlenky. Gramatika nás v tomto omezuje, protože vytváří hranice pro „nepřednastavený“ projev. Tyto hranice lze ovlivnit přidáním alternativních výrazů pro stejný příkaz, čímž se zlepší ovládání pro nepoučenou osobu.

Vytvořená gramatika pro hlasové ovládání robota je psána pro ESGF (Eris Speech Grammar Format) kompilátor. ESGF formát je člověkem snadno čitelný.

Celá textová podoba gramatiky je v příloze k této práci, jedná se o relativně dlouhý textový soubor, který je ovšem přepsán do vizuální stromové podoby (diagram 3.4.3). V gramatice jsou použita anglická slova pro lepší orientaci mezi slovy a skupinami. Tagy skupin jsou psány velkými písmeny ze stejného důvodu.

START_GRAMMAR určuje začátek gramatiky a END_GRAMMAR určuje konec gramatiky. To znamená, že při dosažení END_GRAMMAR ve stromové struktuře se odešle výsledek klientovi.

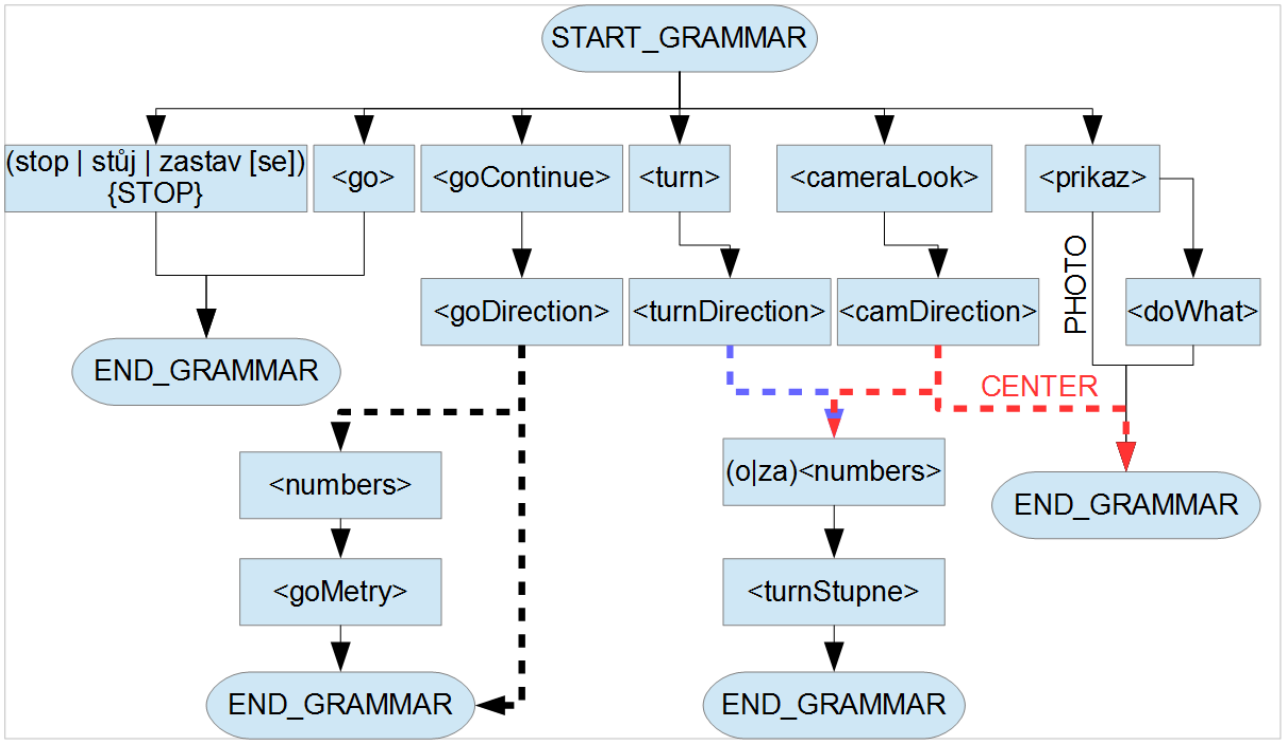


Diagram 3.4.3. – diagram vytvořené gramatiky

Jednotlivé skupiny jsou zapsané ve tvaru <název_skupiny>, pokud se jedná o slova, která nejsou ve skupině, tak jsou zapsané přímo nebo v kulatých závorkách a oddělené znakem „|“ (or): (slovo1 | slovo2 | …).

V diagramu 3.4.3 jsou použity dva typy spojnic a těmi jsou: plná čára a jemně čárkovaná čára. Plná čára má význam jistého výskytu. To znamená, že skupina spojená touto čarou je očekávána. Tečkovaná čára má význam možného výskytu, tedy slovo/skupina nemusí být vyřčeno. Tato slova jsou v gramatice zapsaná v závorkách [slovo]. Text nad čarou v diagramu udává cestu pro dané slovo nebo tag.

Jednotlivé skupiny jsou rozepsány v následující tabulce. Skupiny nebo slova označena tagem GB jsou výplňová slova, toto není zobrazeno v diagramu z důvodu lepší přehlednosti. Skupina <number> je popsána na diagramu 3.4.4. Jedná se o diktování celých číselných hodnot od 0 do 999 999 999 (od nuly do děvět set tisíc devět set devadesát devět).

Název skupiny:	Obsah skupiny:
<go>	(vpřed | kupředu){FORWARD} | (vzad | couvej){BACKWARD} | ((krat | mrak | red | brek){GB} <END_GRAMMAR>)
<goContinue>	(jeď | vydej se){GO} | ((pět | přidaj sa){GB} <END_GRAMMAR>)
<turn>	(otoč se | zatoč){TURN} | (kolotoč{GB} <END_GRAMMAR>)
<cameraLook>	((otoč | natoč) kameru | podívej se){CAMERALOOK} | ((kolotoč umeru | uvítej se){GB} <END_GRAMMAR>)
<prikaz>	(udělej | pořiď){DO} | (vyfoť){PHOTO} | ((vydělej | toť){GB} <END_GRAMMAR>)
<goDirection>	(rovně | vpřed | kupředu){FORWARD} | (vzad | zpátky){BACKWARD} | ((nařadu | rak | vdomě){GB} <END_GRAMMAR>)
<turnDirection>	(doprava | vpravo){RIGHT} | (doleva | vlevo){LEFT} | ((morava | melo){GB} <END_GRAMMAR>)
<camDirection>	<turnDirection> | (dopředu | rovně){CENTER} | (nahoru | vzhůru){UP} | (dolu | k zemi){DOWN} | ((nařadu | průrvu | kelu){GB} <END_GRAMMAR>)
<doWhat>	(fotku | fotografii) {PHOTO}
<goMetry>	(metr | metrů){M} | (centimetr | centimetrů){CM} | ((trimetr | ditr){GB} <END_GRAMMAR>)
<turnStupne>	(stupňů | stupeň){DEG} | (radiánů | radů){RAD} | ((vrutů | kmen){GB} <END_GRAMMAR>)

Tabulka 3.4.1. – tabulka jednotlivých skupin

V následující tabulce jsou odpovídající skupiny pro diagram gramatiky 3.4.4. Tato část se zabývá pouze číselnými výrazy.

Název skupiny:	Obsah skupiny:
<cisla_nula_devatenact>	((jedna | jeden){#1} | dva{#2} | tři{#3} | čtyři{#4} | … devatenáct {#19})

<cisla_dvacet_devadesat>	(dvacet{#20} | třicet {#30} | … devadesát{#90})
<cisla_sto_devetset>	(sto {#100} | dvěstě {#200} | … devětset {#900})
<cisla_tisic>	(tisíc | tisíce){#1000}

Tabulka 3.4.2. – hodnoty skupin čísel

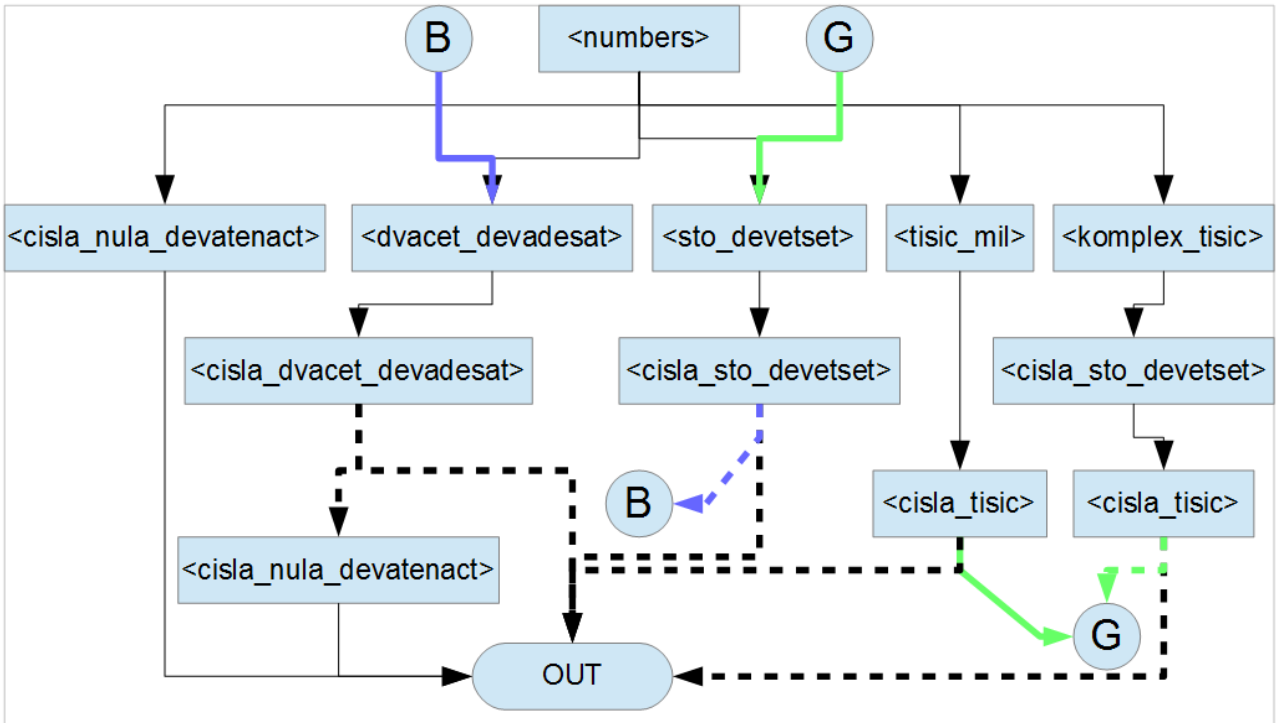


Diagram 3.4.4. – diagram znázorňující číselné výrazy

Rozpoznávač tedy vrací výsledek, který je ovlivněný sestavenou gramatikou. Obslužný program, který tento výsledek přijme, ho vyhodnotí a provede odpovídající proces. Tato reakce v našem případě odpovídá nečinnosti (chybné rozpoznání – tag [GB]) nebo pohybem či funkcí.

3.5. Implementace do internetové stránky

Pro kontrolu robota je vytvořeno jednoduché webové rozhraní, které obsahuje hlasové ovládání a kontrolní prvky. Implementace je jednoduše vyřešena. Po načtení internetové stránky dojde k inicializaci instance SpeechCloudu, která je poté využívána pro správu připojení na SIP ústřednu SpeechCloudu.

Veškeré potřebné kódování pro správu funkčnosti SpeechCloudu je obsažené v jednom JavaScriptu, který je importován jako externí modul do stránky.

Po načtení stránky je tento modul importován a je vytvořena instance SpeechCloudu. Pokud nedojde k chybě při inicializaci, spustí se událost asr_ready. Jako důsledek této události se odešle odkaz na gramatiku, která je uložena na veřejně přístupném webovém úložišti. Dosavadně využívaná adresa http://benesda.4fan.cz/YBPrikazy.txt.

Odesláním požadavku na načtení gramatiky se očekává odpověď, zda-li byl import gramatiky úspěšný. Pokud nedojde k žádné chybě v importování gramatiky, aktivuje se událost asr_set_grammar_ok, která aktivuje přepínač (Switch On/Off – viz 3.3.) pro hlasové ovládání. Pokud dojde k chybě, tlačítko zůstane deaktivováno. Tímto se zamezí uživateli v nepovolených aktivitách.

Pokud je aktivována metoda asr_recognize, tak se periodicky aktivuje i událost asr_signal, která udává informaci o audio vstupu. Aktivováním této události se změní progress bar udávající intenzitu zvuku s rozlišením, zda-li se jedná o hlas nebo šum.

Poslední důležitá událost je asr_result. Tato událost má návratovou hodnotu v podobě výsledku rozpoznávání. Důsledek aktivace této události je spuštění funkce, která odešle pomocí websocketu údaje o výsledku robotovi, který daný výsledek zpracuje a adekvátně na něj reaguje.

Všechny funkce nebo operace se mohou setkat s chybou, které jsou příslušně vypsány do konzole. Pokud tedy dojde k chybě, můžeme tuto chybu diagnostikovat a nalézt.

Zpracování výsledku zpracované v Rpi python scriptu funguje na velmi jednoduchém principu. Vyhledávají se pouze tagy ve výsledku. Na pořadí tagů nezáleží a to z důvodu, že význam příkazu je daný a pořadí je známo, protože je zachováno gramatikou.

Přijatý výsledný String je rozdělen přes mezery do pole. Před vyhodnocením je zkontrolováno celé pole, vyskytuje-li se v něm hodnota '[GB]'. Pokud je tato hodnota nalezena, je celý výsledek klasifikovaný jako chybně rozpoznaný hlasový vstup a dále se již nezpracovává. V opačném případě se aplikují postupně zřetězená pravidla, které vyhledávají tagy v poli podle priority.

Nejdříve jsou hledány tagy s nejvyšší prioritou, mezi které patří: '[STOP]', '[GO]', … . Poté, vzhledem ke zjištěnému tagu s vyšší prioritou, se hledají další tagy. Takto se postupuje až k nejnižší prioritě. Jedná se tedy o jednoduchý rozhodovací strom.

Veškeré zdrojové kódy a potřebné soubory pro spuštění jsou v příloze k této práci.

3.6. Výhody a nevýhody ovládání pomocí SpeechCloudu

Výhody a nevýhody hlasového ovládání je nutné brát s ohledem k tomu, že se jedná o řízení robota. Pokud by se jednalo o ovládání stacionárního objektu nebo počítače, může se kritika změnit v závislosti na využití.

Výhody:

- Možnost sestavit komplexní ovládání v závislosti na použité gramatice.
- Robota je možné ovládat z jakéhokoliv místa, pokud je dostupný přes bezdrátové připojení (ve speciálním případě pomocí Ethernetu).
- Složitější funkce robota lze ovládat rychleji a více přirozeně (pro člověka), než při zadávání pomocí klávesnice.

Nevýhody:

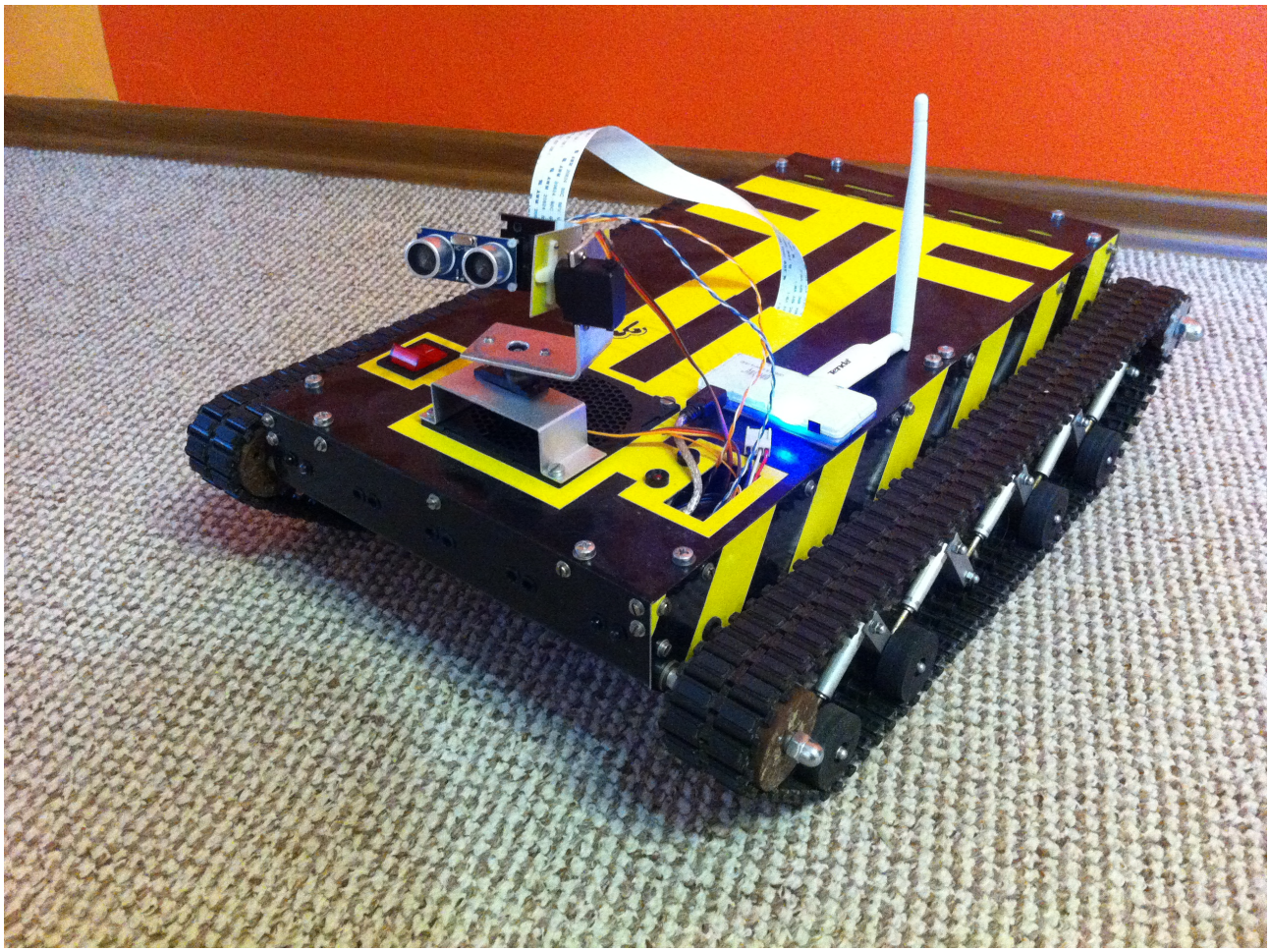
- S vyšší složitostí ovládání je větší šance na chybné rozpoznání.
- Za předpokladu navigace robota ve hlučném prostředí je tento přístup ovládání nevhodný z důvodu rušení.
- Pokud se při navigaci dopustí rozpoznávač chyby v kritickém okamžiku (moment nad propastí) může tato chyba způsobit značnou komplikaci.
- Při pomalém připojení k internetu je pravděpodobná nefunkčnost rozpoznávání. Je nutné připojení na internet nebo na síť přes kterou je SpeechCloud přístupný (univerzitní síť).

3.7. Shrnutí

Ovládání pomocí SpeechCloudu je možné využít v širokém spektru možností. Věta či slova, které jsou rozpoznávána, musí být určitým způsobem přítomna v gramatice. Pokud nejsou v gramatice, jako by neexistovala. V jistém smyslu nejsme limitování složitostí gramatiky, ale je nutné brát na zřetel, že čím složitější gramatika je, tím náročnější bude vyhodnocování.

Ovládání lze sestavit tak, aby bylo efektivnější, než běžný projev člověka. To znamená, že je ignorována gramatika českého jazyka a využijí se pouze slova, která jsou podstatná pro daný příkaz. Příkladem může být: „Jeď opatrně dopředu padesát metrů a pak zastav“ a jednoduchá verze „rovně padesát metrů stop“. Samozřejmě, pokud chceme mít ovládání přístupnější široké veřejnosti, je nutné gramatiku vytvořit tak, aby ji mohl využít i nepoučený člověk.

4. Vlastnosti robota



Obrázek 4.0.0. – ilustrativní fotografie robota

4.1. Základní specifikace

Podvozek robota je sestaven převážně z materiálu hliníkových slitin (dural), který dosahuje dostatečné pevnosti. Hlavní kostra je rám z duralových profilů na kterém je připevněna hliníková karosérie s ochranným lakem. Jedná se o konstrukci s pásovým pohonem a deseti pojezdovými koly (dvě kola na jednom rameni) na každé straně. Každé rameno má samostatné odpružení, což zajišťuje značnou stabilitu při jízdě různým terénem.

Robot je schopen pohybu pomocí dvou elektromotorů, které jsou vybaveny převodovkou. Na druhém převodovém kole je dekódovací kolečko, které slouží k určení rychlosti. Robot je vybaven kamerou ke snímání obrazu. Tato kamera nemá zatím žádné pokročilé využití, ale jedná se o důležitou budoucí komponentu například pro rozpoznávání objektů atd. Celý robot je napájen li-ion baterií s nominálními hodnotami 7,4v-8Ah → cca 56Wh.

Robot je dále připraven pro montáž různých senzorů kterými jsou: ultrazvukový dálkoměr pro stanovení vzdálenosti k objektu, infračervené senzory pro detekci blízkých předmětů (max. 10 cm) a 3D kompas.

Elektromotory jsou řízeny pomocí PWM (pulsně šířkovou modulací) s frekvencí 312,5Hz a rozlišením 200 na puls. Rozlišení je nicméně redukováno v závislosti na napětí baterie a to tak, aby byly elektromotory řízeny maximálně 5 volty. Šířka pulsu se nastavuje procentuální hodnotou výkonu, kde je jedno procento vypočítáno pomocí vztahu:

$$procento = \frac{1024}{napětíBaterie * 2 * 100} * 200 = \frac{512}{napětíBaterie} * 2 = \frac{1024}{napětíBaterie}$$, kde napětíBaterie je napětí baterie vyhodnocené pomocí AD převodníku

Maximální hodnota je tedy 100% ~ 100. Pokud má baterie napětí 7V, tak AD převodník mikrokontroléru naměří hodnotu 716 (celkové napětí baterie je děleno dvěma). Procento výkonu motorů odpovídá hodnotě 1,43. Při nastavení výstupního výkonu na hodnotu 50% bude na výstupu PWM naměřena střední hodnota napětí 2,485V podle vztahu:

$$výkon = \frac{procento * nastavení}{200} * napětíBaterieV$$, kde napětíBaterieV je měřitelné napětí, nastavení je požadovaný výkon a procento je jedno procento výkonu.

Tato normalizace napětí motorů má dva hlavní důvody:

1) Tímto se omezí „přetěžování“ elektromotorů příliš vysokým proudem.
2) Zjednoduší se regulace výkonu elektromotorů. Důvod je ten, že napětí baterie se v průběhu běhu robota mění. Mění se napětí i v závislosti procházejícího proudu podle Ohmova zákona a vztahu pro úbytek napětí na vodiči: $U_d = I * R$, kde U_d je úbytek napětí na vodiči, I je proud procházející vodičem, R je odpor vodiče. Tedy elektromotor má jiné vlastnosti (točivý moment, otáčky, spotřeba, …) v závislosti na napájecím napětí. Normalizací zůstanou tyto vlastnosti stejné.

Robot má kameru, která je umístěna na „hlavě“ s dvěma osami pohybu – vertikální a horizontální otáčení. Díky tomuto je možno kameru natočit bez pohybu podvozku. Každá osa je řízena jedním servomotorem. Každé servo je řízeno frekvencí 50Hz s pulsem o délce 1.0-2.0 ms (5% - 10%). Pro natočení kamery do výchozí polohy je délka řídícího pulsu serva 1.5ms. Natočení na pozici 0° odpovídá délka řídícího pulsu 1ms. Natočením na úhel 180° je délka pulsu 2ms. Serva mají softwarové omezení otočení v rozsahu 160° z důvodu možného poškození vnitřní konstrukce.

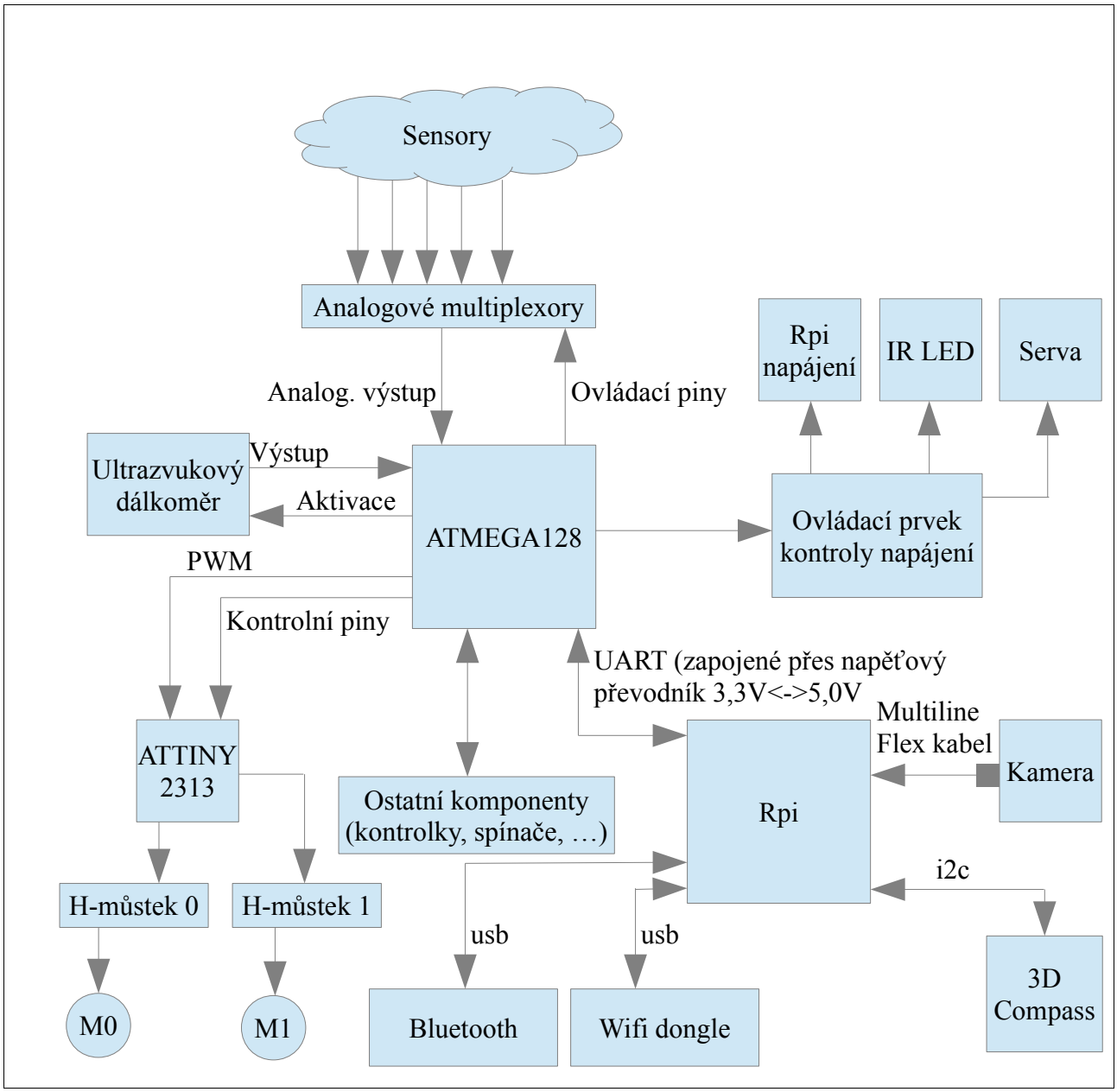


Diagram 4.2.1. – jednoduché blokové schéma propojení důležitých komponent robota.

Šipky udávají směr datového toku nebo napájení. Například ultrazvukový dálkoměr vyžaduje krátký impuls na vstupním portu pro aktivaci měření a jako výstup využívá výstupní port, který je přiveden na externí přerušení na MCU.

Každá komponenta je napájena příslušným napětím. USB využívá 4 žilový kabel (2 napájecí a 2 datové), i2c používá tři (interně jenom dva – SDA, SCL), uart využívá tři (interně jenom dva – RXD, TXD).

Robot je připraven pro vybavení množstvím IR senzorů, které slouží k detekci hran nebo překážek před/za nebo pod robotem. Slouží k zabránění sjezdu robota „z útesu“ nebo k ochraně

proti nechtěným nárazům do překážek (například do dveří). Dále by měl být vybaven i senzory na bocích, které by bylo možné využít k těsnému sledování stěny.

K řízení komponent, jako například vyhodnocování stavů senzorů, řízení H-můstků a tedy i motorů, řízení servomotorů a další, slouží mikrokontrolér (dále jen MCU) ATMEGA128. MCU zajišťuje komunikaci s okolím pomocí sériové linky, více v kapitole 5.2. Elektromotory a servomotory řídí pomocí PWM na dvou odlišných časovačích.

Využitím integrovaného AD převodníku se vyhodnocují dvě kategorie senzorů:

1) „Životně“ důležité hodnoty: napětí baterie, proudu, teploty baterií a teploty H-můstků.
2) Specifické senzory: toto jsou vyhodnocené stavy senzorů přiblížení: infračervené senzory.

„Životně“ důležité hodnoty jsou podstatné pro ochranu komponent. Robot je napájen li-ion baterií, která vyžaduje provozní kontrolu. Li-ion baterie by se neměla proudově přetěžovat, z tohoto důvodu mimo mechanicky přerušitelné pojistky, kontrolují proud měřením úbytku napětí na vodiči se známým odporem a pomocí operačního zesilovače je toto napětí zesíleno do měřitelného spektra s dostatečným rozlišením (AD převodník má rozlišení 10-bit na Vcc (5V)).

Další omezení baterie je minimální napětí, kterého může baterie dosáhnout. Proto je napětí baterie měřeno přímo na svorkách, aby byl výsledek co nejméně ovlivněn. Pokud dojde k přílišnému snížení napětí, tak se omezí funkce robota kterými jsou: pohon elektromotorů, servomotorů, atd.

Kontrolou teploty baterií a H-můstků se zamezí jejich možnému přehřívání. „Životně“ důležité hodnoty jsou striktně periodicky kontrolovány s periodou 62,5ms ~ 16Hz. Na tyto hodnoty je možné poslat dotaz přes sériovou linku dále v kapitole 4.2.

4.2. Komunikační protokol

Robot je připojen přes asynchronní sériovou linku. Tato sériová linka je ve výchozím nastavení na hodnotách [9600-2-O] [baud rate – stop bits – parity]. Jediný parametr, který se dá změnit i za běhu je pouze baud rate. Proto je nutné nastavit parametry sériové linky na obou stranách tak, aby byly v souladu.

Rychlost komunikace (baud rate) je vhodné udržet na dolní hranici maximální možné rychlosti. Pokud by byla rychlost příliš vysoká, tak bude náchylnější k rušení (například od motorů). Pokud bude příliš nízká, tak nebude dostatečná rychlost obsluhy příchozích/odchozích dat. Tedy

pokud bude potřeba rychlost 11538 bps, tak je vhodné nastavit baud rate na 19200 bps. Vytížení sériové linky není pevně dané, záleží na periodě dotazů například na stavy senzorů.

Komunikace probíhá pomocí jednoduchého komunikačního protokolu. Všechna data jsou odesílána v bitové podobě, to znamená, že hodnoty jsou v rozmezí 0-255. Rámec komunikace má 4 segmenty. První segment „s0“ o délce jednoho bajtu slouží k určení délky rámce, druhý segment „s1“ o délce jednoho bajtu k určení příkazu, třetí segment „s2“ jsou data s velikostí 0-“s0“ (maximální velikost s0 by měla být 127), poslední segment „s3“ s velikostí jednoho bajtu je kontrolní bajt.

s0 – délka fragmentu	s1 – číslo příkazu	s2 - data	s3 – kontrolní byte

Tabulka 4.2.1. – schéma rámce

Velikosti segmentů:

Index segmentu:	Velikost segmentu:
s0	1
s1	1
s2	0<s2<(s0-3) (0<>124)
s3	1

Tabulka 4.2.2. – velikosti segmentů rámce

Kontrolní bajt je XOR funkce všech hodnot ve fragmentu. Tedy vzorový rámec příkladu příkazu jízdy vpřed řízení pomocí PID regulátoru:

s0	s1	s2-0	s2-1	s2-2	s3
6	5	25	25	136	139

Tabulka 4.2.3. – vzorový rámec

První segment rámce s0 má hodnotu 6 to znamená, že délka rámce je rovna 6. Druhý segment s1 má hodnotu 5 této hodnotě odpovídá příkaz řízení motorů PID regulátorem. Třetí segment s2 obsahují potřebná data k provedení příkazu. Čtvrtý segment s3 je kontrolní bajt, který se vypočítá s3=s0 ^ s1 ^ s2-0 ^ s2-1 ^ s2-2 ^ s3 = 139.

Kontrolní bajt slouží k ověření správnosti rámce. Pokud se odesílá větší množství rámců ihned za sebou, může dojít k pomalé obsluze a tedy i vynechání jednoho i více bajtů. Pokud k této situaci dojde, tak na straně příjemce nebude odpovídat přijatý kontrolní bajt kontrolnímu bajtu odeslaného. Tímto je zajištěna celistvost dat a minimalizace chyby v komunikaci.

Komunikace funguje na dotazovém principu. Na danou instrukci odpoví datovým rámcem nebo provedením funkce robota. To znamená, že na příkaz číslo 1 dostane klient odpověď v podobě

datového rámce obsahující informace ze senzorů (napětí baterie, teploty komponent, …). V jiném případě robot neodesílá žádná data po sériové lince, ale pouze poslouchá do dalšího požadavku.

K řízení robota lze využít několik příkazů, kterými lze nastavit nebo ovládat funkce robota. Lze nastavit pohyb robota resp. rychlost jednotlivých motorů. Otáčet „hlavou“ na které je připevněna kamera. Přenastavit konstanty PID regulátoru a další možnosti viz. následující tabulka.

Funkce příkazu:	Hodnota instrukce příkazu:	Segment data:	Odpověď:
Požadavek hodnot stavů hlavních senzorů. To jsou senzory napětí baterie a teploty.	1	---	ano
Požadavek hodnot stavů 8 senzorů od daného indexu	2	[index(1)](1)	ano
Dotaz na rychlost otáček motorů	3	---	ano
Ovládání výkonu motorů bez využití PID regulátoru	4	[výkonA(1),výkonB(1),směr(1)](3)	ne
Ovládání výkonu motorů s využitím PID regulátoru	5	[výkonA(1),výkonB(1),směr(1)](3)	ne
Ovládání výkonu motorů s využitím PID regulátoru s určeným dojezdem	6	[výkonA(1),výkonB(1),směr(1),dojezdA(4), dojezdB(4)](11)	ne
Nastavení jednotlivých konstant PID regulátoru	7	[Kp(2),Ki(2),Kd(2)](6)	ne
Natočení kamery v horizontální ose	8	[úhel(1)](1)	ne
Kontrola předem definovaného pohybu bez PID regulátoru	9	[příkaz(1),výkon(1)](2)	ne
Natočení kamery ve vertikální ose	10	[úhel(1)](1)	ne
Požadavek hodnoty ultrazvukového senzoru	11	---	ano
Ovládání výkonu motorů bez využití PID regulátoru s určeným dojezdem	12	[výkonA(1),výkonB(1),směr(1),dojezdA(4), dojezdB(4)](11)	ne

Požadavek na stav změny hodnot senzorů.	18	---	ano

Tabulka 4.2.4. – příkazy

Vysvětlivka k tabulce 4.2.4. Hodnoty ve sloupci segment dat je uveden segment s2 (potřebná data příkazu). Hodnoty jsou uvedeny v závorkách […] skládají se z názvu proměnné, která je následována hodnotou v jednoduchých závorkách obsahující délku proměnné v bajtech.

Následující tabulka obsahuje doplňující informace k příkazům v tabulce 4.2.4.

<table>
<tr><td>Hodnota instrukce:</td><td>Doplňující údaje:</td></tr>
<tr><td>1</td><td>Odpověď – [napětíBat1(2),napětíBat2(2),proud(2),teplotaBat1(2), teplotaBat2(2),rezerva(2),teplotaHB1(2),teplotaHB2(2)](16)</td></tr>
<tr><td>2</td><td>Odpověď – [sz0(2),sz1(2),sz2(2),sz3(2),sz4(2),sz5(2),sz6(2),sz7(2)](16)</td></tr>
<tr><td>3</td><td>Odpověď – [rychlostMot1(2),rychlostMot2(2)](4)</td></tr>
<tr><td>4 | 5 | 6 | 12</td><td>Výkon je hodnota, která odpovídá procentuálnímu výkonu motoru při napájení 5V. Pokud je motor řízen PID regulátorem, tak je řízen na počet přerušení od otáčkoměru za ¼ sekundy.
Směr je jeden bajt, který obsahuje informaci o směru obou motorů.
<table>
<tr><td>Motor</td><td>Vpřed</td><td>Vzad</td><td>Nečinný</td><td>Brzda</td></tr>
<tr><td>MotorA</td><td>0x80</td><td>0x40</td><td>0x20</td><td>0x10</td></tr>
<tr><td>MotorB</td><td>0x08</td><td>0x04</td><td>0x02</td><td>0x01</td></tr>
</table></td></tr>
<tr><td>6 | 12</td><td>DojezdX – počet přerušení od dekódovacího kolečka v převodovce. Jedna otočka trakčního kolečka odpovídá 139,82 přerušení (~11,1cm).</td></tr>
<tr><td>7</td><td>Každá hodnota je po přijetí interně vydělena hodnotou 1000. To znamená, že při odeslání hodnot [255,255,0,0,0,0] bude v konstantě Kp hodnota 65,535.
Proměnná Kp – proporcionální konstanta, Ki – integrační konstanta, Kd – derivační konstanta</td></tr>
<tr><td>8 | 10</td><td>Proměnná úhel může mít maximálně hodnotu 250. Hodnota 125 odpovídá středu serva, hodnota 0 odpovídá otočení 0° a hodnota 250 odpovídá otočení 160°. Jeden stupeň odpovídá hodnotě 1,5625 v celých číslech hodnotě 2.</td></tr>
<tr><td>9</td><td>Proměnná výkon odpovídá procentuálnímu napětí napájení motoru.
Proměnná příkaz samostatného pohybu:
<table>
<tr><td>Hodnota:</td><td>Pohyb:</td></tr>
<tr><td>0</td><td>Nečinný</td></tr>
<tr><td>1</td><td>Vpřed</td></tr>
<tr><td>2</td><td>Vzad</td></tr>
<tr><td>3</td><td>Levý vpřed, pravý stop</td></tr>
<tr><td>4</td><td>Pravý vpřed, levý stop</td></tr>
<tr><td>5</td><td>Pravý vpřed, levý vzad</td></tr>
<tr><td>6</td><td>Levý vzad, pravý vpřed</td></tr>
</table></td></tr>
</table>

11	Odpověď – [vzdálenost (proporcionálně odpovídá 5cm)(1)](1)
18	Odpověď – [8 bajtů, každý bit tohoto bajtu označuje změnu hodnoty senzoru na daném indexu o +-30% od posledního volání příkazu 18](8) Pozn. Hodnota 1 označuje, že došlo ke změně. Hodnota 0 označuje, že nedošlo ke změně. Řazení: nejvýznamnější bit nejvýznamnějšího bajtu nese informaci o senzoru 0.

Tabulka 4.2.5. – Doplňující informace

4.3. Senzory

Vyhodnocení stavů senzorů pomocí AD převodníku je děleno na dvě skupiny, z důvodu důležitosti i různého způsobu měření. „Životně“ důležité hodnoty jsou vyhodnoceny jednou za periodu. To znamená, že se změří napětí prvního článku baterie a při dalším vyhodnocení se změří celkové napětí baterie a dalším vyhodnocením se změří proud a tak dále.

Specifické senzory jsou infračervené tranzistory[11]. Stavy těchto senzorů jsou vyhodnocovány na základě dvou vzorků. Jeden vzorek je vyhodnocení stavu senzoru bez osvětlení IR LED[12] diody. Dostaneme tedy hodnotu „osvětlení“ prostředí. Druhý vzorek je vyhodnocení senzoru s osvětlením IR diody. Výstup senzoru je tedy vyhodnocen na základě rozdílu těchto dvou hodnot. Tento přístup je aplikován z důvodu proměnlivých podmínek. Tím je myšleno, že kdyby se vyhodnocovaly pouze hodnoty osvíceného prostředí pomocí IR LED, docházelo by k velkému rušení okolním osvětlením například od Slunce.

Následující tabulka udává hodnoty stavů senzorů s jejich limity. Specifické senzory nemají limity z důvodu možného externího přenastavení:

Název senzoru	Index senzoru	Min hodnota	Max hodnota	Omezení při překročení hodnot max/min
Napětí baterie	0	675	neomezeno	Omezení výkonných funkcí
Napětí jednoho prvního článku baterie	1	675	neomezeno	Omezení výkonných funkcí
Proud	2	neomezeno	450	Omezení výkonných funkcí
Teplota prvního článku baterie	3	377 (45°C)	800(-2.5°C)	Omezení výkonných funkcí
Teplota druhého článku baterie	4	377 (45°C)	800(-2.5°C)	Omezení výkoných funkcí

11 Tranzistor citlivý na infračervené světlo (IR). Mění svoji propustnost na základě dopadajícího množství IR záření na přechod tranzistoru.

12 IR LED dioda je dioda vyzařující záření v infračerveném spektru.

Teplota H-můstku 1	6	63(60°C)	500(-2°C)	Vypnutí motorů
Teplota H-můstku 2	7	63(60°C)	500(-2°C)	Vypnutí motorů
IR senzory	8-63	Nestanoveno	Nestanoveno	Nestanoveno nebo zastavit

Tabulka 4.3.1. – tabulka senzorů

IR senzory potřebují pro správnou funkci dva vzorky. Následující tabulka ilustruje výslednou hodnotu stavu senzoru. Při větším vystavení IR transistoru infračerveným spektrem světla dojde ke zvýšení napětí na výstupu senzoru.

Senzor	Hodnota – vypnutá IR LED - s^{off}	Hodnota – zapnutá IR LED - s^{on}	Výsledek stanoven $s^{on} - s^{off}$
IR senzor	137	433	296 – detekována překážka
IR senzor	137	145	8 – šum?
IR senzor	137	137	0 – nic, nebo chyba

Tabulka 4.3.2. – tabulka hodnot IR senzorů

Další senzor je ultrazvukový dálkoměr, který slouží k měření vzdálenosti od objektu. Tento senzor je umístěn na stejné pozici jako kamera, takže je možné měřit vzdálenost k objektu, který je v záběru kamery.

Jedná se o měření pomocí zvukové vlny. Za pokojových podmínek odpovídá jedna ms vzdálenosti 34,3 centimetru. Jedná se o lineární závislost, čím dále je objekt, tím větší čas bude zvuková vlna potřebovat na cestu k objektu a zpět do senzoru. Senzor je nastaven na rozlišení 5 cm, což odpovídá času 0,292ms. Při frekvenci MCU 16Mhz je to 4672 cyklů. Maximální vzdálenost objektu je stanovena na 2m, což odpovídá času 11,68ms a pro procesorový čas 186880 cyklů.

Perioda vzorkování ultrazvukového senzoru je 4Hz. Jedná se o kompromis mezi rušením od zbytkových zvukových vln z dřívějšího vzorku a vytížením MCU. Vzdálenost objektu od senzoru je určena pomocí vztahu:

$vzdálenost = value * 5$, kde value je hodnota změřena časovačem MCU

Je nutné brát v potaz omezení ultrazvukového dálkoměru. Některé objekty jsou těžko detekovatelné, jako například natažené prostěradlo. Může také docházet k chybně stanovené vzdálenosti. K chybě může dojít například z důvodu detekce vzdáleného předmětu při odrazu od jiného bližšího. To je způsobeno rozptylem zvuku. Jako příklad lze uvést detekci postavy ve vzdálenosti 1,5 metru. Před postavou bude krabice, která není úplně v přímce nasměrování senzoru, ale z důvodu rozptylu zvukové vlny dojde k odrazu od této překážky.

4.4. Výkonná výpočetní část

Robot je řízen pomocí mini počítače Raspberry Pi 2 (dále jen Rpi). Tento mini počítač disponuje dostatečným výpočetním výkonem k obsluze kamery a síťové komunikace přes Wi-Fi usb adaptér (maximální vytížení při běžícím obslužném scriptu je přibližně 35% CPU a 24% RAM). Na Rpi běží operační systém Raspbian Jessie. K softwarové výbavě je dále nainstalován apache, který je nastaven na distribuci webového rozhraní v https režimu poslouchající na portu 25555.

Řídící software, který zajišťuje komunikaci s MCU a vyhodnocování výsledků hlasového ovládání, je programovaný v jazyce python. Potřebné moduly, které nejsou v základu nainstalované ve standardní distribuci pythonu 2.7.x jsou:

- pyserial (pro komunikaci pomocí sériové linky).
- python-tornado (slouží pro nastavení komunikace pomocí websocketu s ssl zabezpečením).
- opencv (tento modul je použit pro zpracování obrazu z kamery).
- PIL (modul pro práci s obrázky).

Rpi je napájeno z interní baterie robota. Napětí je stabilizováno pomocí LDO stabilizátoru na 5V. Stejný zdroj je použit k posílení USB hubu. Rpi není napájeno, pokud nejsou „životní“ atributy v rámci definovaných mezí. To znamená, že baterie nemá příliš nízké napětí, není odebírán příliš vysoký proud a nebo baterie nemá příliš nízkou/vysokou teplotu. Důvod je ten, že Rpi odebírá relativně vysoký proud i v režimu neaktivity = idle režim (přibližně 480 mA se zapojenou kamerou a Wi-Fi adaptérem). Pokud by bylo napětí baterie příliš nízké, tak by aktivita Rpi mohla způsobit příliš rychlý pokles napětí a mohlo by dojít k poškození baterie.

MCU je připojeno na sériovou linku Rpi. Sériová linka je přímo na desce Rpi a má čísla pinů: 8 a 10. Rychlost sériové komunikace je nastavena na 19200bps | 2 stop bity | odd parita (sudá parita). Vzhledem k tomu, že Rpi má jiné TTL úrovně signálu (3,3V), než MCU(5,0V), je výhodné použít převodník napěťových úrovní, aby se zamezilo poškození hardwaru.

Veškerá externí USB zařízení jsou připojena do externího USB hubu, který je samostatně napájen. Samostatné napájení (nebo-li posílení zdrojem) je použito z důvodu, aby bylo možné napájet zařízení větším proudem, než 0,5A. USB hubem je ošetřena možnost nedostatečného napájení USB zařízení a tím i jeho následnou nefunkčnost a nestabilitu Rpi. Podle dokumentace k Rpi lze napájet přímo z USB zařízení do 0,5A.

Další dodatečný software je připojením Rpi na soukromou VPN. VPN je použito hlavně z důvodu pravděpodobného umístění robota za NATem (nemá veřejnou IP adresu). Pomocí VPN je možné se na robota připojit odkudkoliv na světě (pokud nejsou na dané síti blokované potřebné porty nebo služby).

Další důvod je lehčí nalezení lokální IP adresy Rpi v rozsáhlé Wi-Fi síti. Příkladem může být připojení do školní sítě eduroam: robot je připojen, ale není známa jeho IP adresa. Připojením se na robota přes VPN je možné zjistit jeho IP adresu, aby se dalo připojit na webové rozhraní. Je možné otevřít webové rozhraní přes VPN připojení, ale tento přístup má svá úskalí a to hlavně v podobě vyšší odezvy. Další možností je odeslání broadcastového požadavku a v arp tabulce nalézt zařízení s odpovídající MAC adresou Wi-Fi adaptéru robota (velmi nepřehledné a časově náročné).

4.5. Shrnutí vlastností robota

- Nízkoúrovňové funkce robota jsou řízeny MCU. Tyto funkce jsou:
  - řízení elektromotorů
  - vyhodnocování stavů senzorů (hodnot)
  - bezpečnostní kontrola baterií (teplota, napětí, proud)
  - řízení servomotorů
  - ovládání napájení všech proudově náročných součástí (IR LED senzorů, snímač otáček, elektromotory, Rpi, …)
  - pohyb na základě zpětné vazby od otáčkoměru (ujetí určité vzdálenosti jednotlivých elektromotorů – otáčení ve stupních, měření vzdálenost, …)
- Vyšší funkce jsou řízeny Rpi:
  - připojení k bezdrátové sítě Wi-Fi
  - záznam obrazu z kamery
  - komunikace s MCU pomocí uart linky (vybaven napěťovým převodníkem úrovní)
  - distribuce webového rozhraní (https-html) na portu 25555
  - periodické (0,5s) odesílání „hello“ rámce pro reset watchdogu MCU[13]
  - běh python scriptu, který komunikuje pomocí websocketu s webovým rozhraním
    - zpracování výsledku hlasového vstupu
    - odesílání informačních dat
    - a další funkce
- Řízení pohybu:
  - robot je řízen dvěma elektromotory s vlastními převodovkami

13 Watchdog je samostatný časovač, který slouží pro reset MCU při zacyklení. V tomto případě dojde k resetování a tedy i zastavení robota pokud přestane fungovat Rpi nebo se vypne řídící script bez zastavení pohybu (bezpečnost).

- Každý z pásů robota je hnán jedním elektromotorem. Oba pásy mají odpružená pojezdová kola.

- Napájení:
  - akupack je sestaven z li-ion článků s celkovou nominální hodnotou 7,4V – 8Ah
  - akupack je nabíjen na 8,2V – tzn. přibližně 85% kapacity
  - Stabilizace napětí pro jednotlivé komponenty:
    - Rpi je napájeno samostatným stabilizátorem napětí +5V
    - oddělený stabilizátor napětí +5V pro:
      - MCU (ATMEGA128, ATTINY2313)
      - analogové multiplexory
      - další funkční komponenty
    - servomotory a jiné výkonné části jsou napájeny pomocí PWM regulaci napětí pomocí výkonové tranzistorů.

5. Vizuální rozhraní

Vizuální rozhraní je webová aplikace, která je distribuována na IP adresách Rpi pomocí serverové aplikace Apache[14]. Přístup k webovému rozhraní lze získat pomocí webového prohlížeče a to zadáním IP adresy Rpi s portem 25555.

Toto vizuální rozhraní využívá dva websockety. Jeden slouží k oboustranné komunikaci ve které je využit formát JSON. Přes tento websocket jsou posílány veškeré příkazy nebo informace z/do robota. Tento websocket je přístupný na stejné IP adrese s dodatkem „/chat:8080“.

Druhý websocket je použit pro odesílání binárních dat z robota. Tento websocket je použit pouze pro „stahování“ fotografií z robota. Tento websocket je přístupný na stejné IP adrese s dodatkem „/chat:8081“. Stavy obou websocketů jsou zapsány do formuláře Websocket logger.

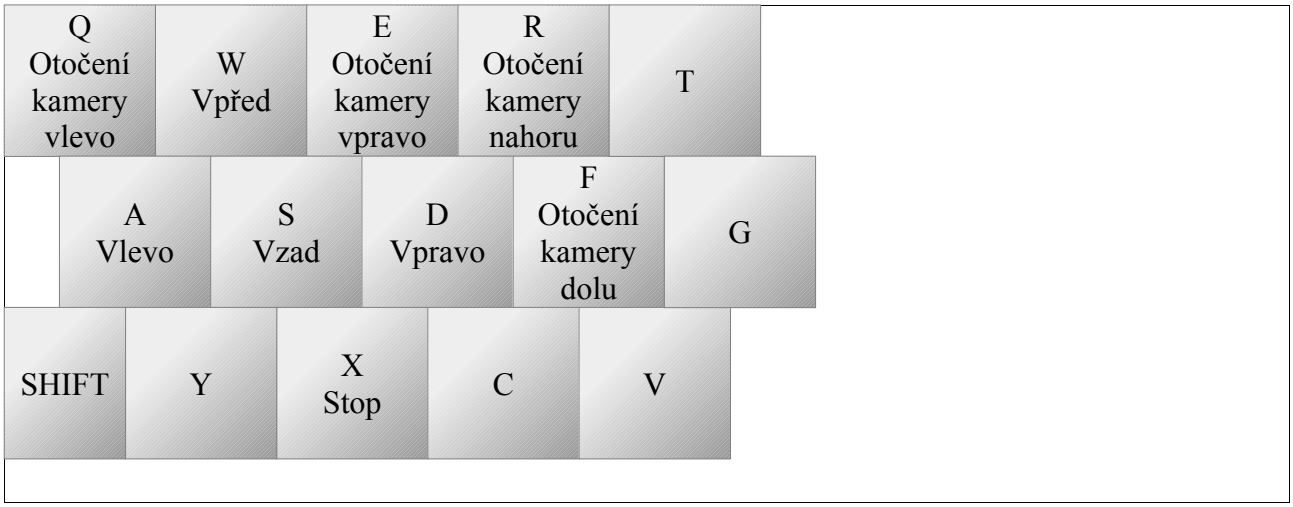
Websocket logger zaznamenává chyby nebo informace o stavů připojení na server. V běžných případech se vypíše připojeno nebo nepřipojeno. Pokud dojde k chybě připojení je to s největší pravděpodobností způsobeno tím, že na Rpi není puštěn příslušný script, který spravuje komunikaci. Tento script se musí spouštět manuálně. Není aktivní při start-up (při spuštění) z důvodu bezpečnosti, ale i z důvodu jednoduššího ladění.

Vizuální rozhraní má tedy z informačního hlediska: aktuální pohled z kamery, údaje ze senzorů (Stav baterie, odebíraný proud, teploty, …). Dalším neméně důležitým prvek je logger, do kterého se vypisují výsledky hlasového rozpoznávání.

Ovládacích prvků je větší množství:

- ovládání kamery pomocí aktivních prvků:
    - změna barevného modelu přijímaného „videa“
    - změna rozlišení kamery
    - požadavek pro vytvoření fotografie. Tato fotografie se uloží na lokální úložiště v Rpi (stejná složka jako složka se scriptem) s názvem v podobě aktuálního času.
    - a další prvky

14 Apache Server je aplikace s otevřeným kódem. Jedná se o aplikaci, která poskytuje přístup k webovým stránkám, které jsou uloženy na serverovém zařízení.
Více informací [online] dostupné z: http://www.apache.org/ [cit. 2016-05-02]



Obrázek 5.1.1. – vizualizace ovládácích kláves robota

- přímé ovládání pohybu podvozku pomocí tlačítek **w|a|s|d|q|e|r|f|x** (a jejich velké varianty **W|A|...**):
  - **w | a | s | d** je použito pro pohyb podvozku a **x** je pro „nouzové“ zastavení:
  - q | e | r | f je použito pro otáčení kamery:
- Hlavní řízení je směrováno na hlasového ovládání. Výsledek hlasového vstupu je zapsán do formuláře Speech recognition logger. Tento výsledek je odeslán pomocí websocketu do Rpi, kde dojde k následnému zpracování.
- Poslední ovládací prvek slouží k vypnutí python programu v Rpi. Pro ošetření nechtěného stisku tlačítko vyvolá alert typu ano/ne.

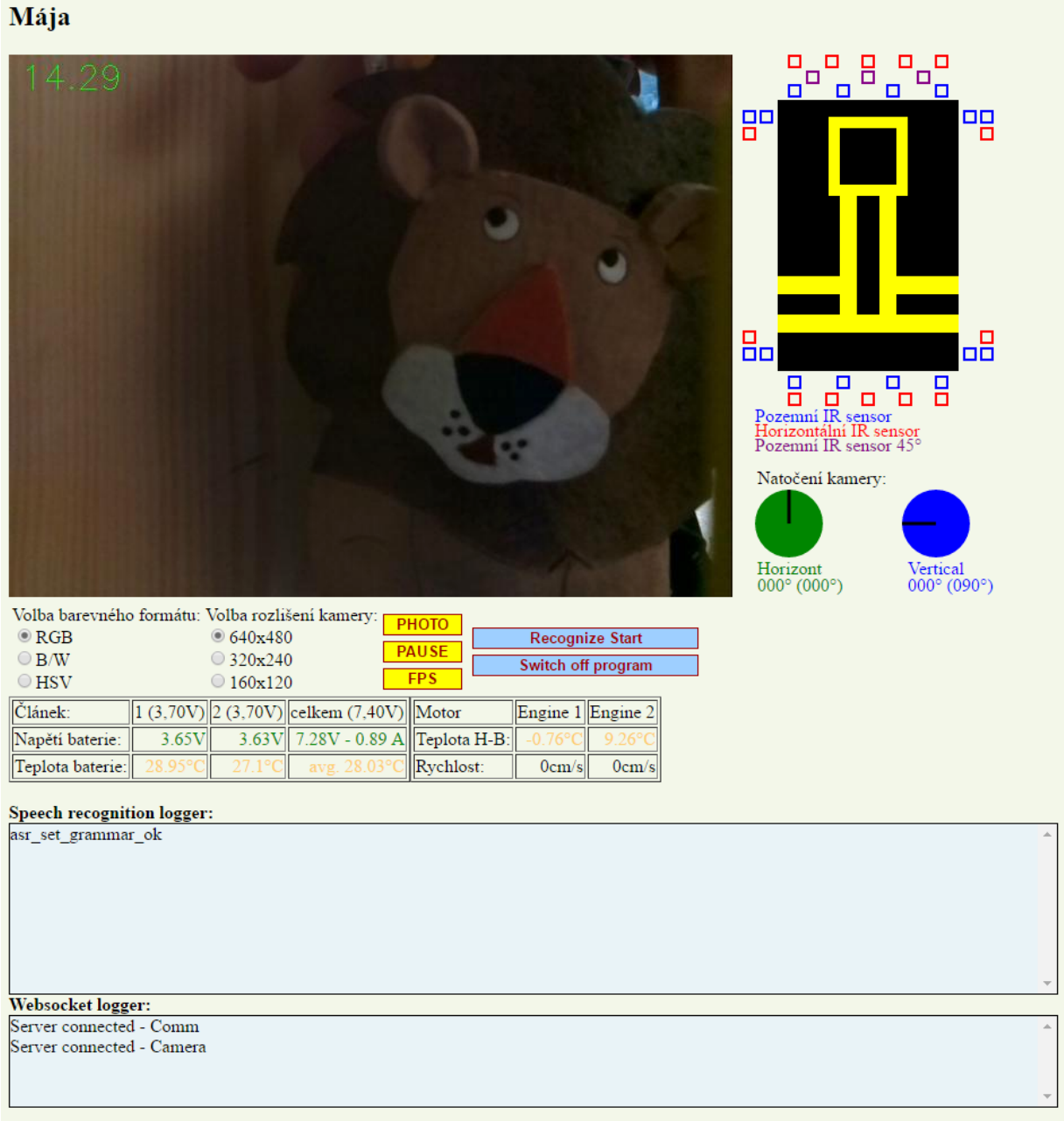
Po přijetí výsledku hlasového vstupu je výsledek odeslán přes websocket do Rpi. Každý příkaz nebo stav senzorů má jiný index (rozhodovací hodnota) v JSON schématu. Na základě tohoto indexu se aplikuje vyhodnocení podle dané procedury. Následující tabulka znázorňuje veškeré příkazy odesílané a přijímané webovým rozhraním.

Název indexu	Tx/Rx	Další prvky	Funkce
„camera“	Tx	„CameraFormat“: Fce 1	Fce 1: • color:[0,1,2] – nastavení barevného modelu (0-rgb, 1-b/w, 2-hsv) • resolution:[0,1,2] – nastavení rozlišení kamery (0-qqVGA, 1-qVGA, 2-VGA) • photo:[1] – puls pro příkaz pořízení fotografie • pause:[0,1] – nastavení pozastavení „videa“ (0-pauza, 1-běh) • quit:[rezervováno] – vypnutí procesu kamery

			(nenastavovat) • fps:[0,1] – zobrazení fps ve snímku (0-bez fps, 1-s fps)
„direct“	Tx	„input“ : tlacitka Fce 1	Fce 1: • „w“ : [0|1] • „a“ : [0|1] • „s“ : [0|1] • „d“ : [0|1] - přednastavené pole, které mění své hodnoty při stisku klávesy ve webovém rozhraní. Tyto hodnoty lze například použít pro přímé řízení podvozku
„directKamera“	Tx	„input“ : tlacitka Fce 1	Fce 1: • „q“ : [0|1] • „e“ : [0|1] • „r“ : [0|1] • „f“ : [0|1] - přednastavené pole, které mění své hodnoty při stisku klávesy ve webovém rozhraní. Tyto hodnoty lze například použít pro přímé otáčení kamery
„speech“	Tx	„Recognition“: Fce 1	Fce 1: • data : string – výsledek hlasového rozpoznání v podobě datového typu string
„baterie“	Rx	„vBat(1|2)“: Fce 1 „vBatSum“: Fce 2 „proud“: Fce 3 „tempBat(1|2)“: Fce 4 „tempHB(1|2)“: Fce 5	Fce 1: • Napětí baterie (jednotlivé články). Jednotka [V]. Fce 2: • Celkové napětí baterie (svorkové napětí). Jednotka [V]. Fce 3: • Naměřený proudový odběr. Jednotka [A]. Fce 4: • Teplota baterie (samostatné články). Jednotka [°C]. Fce 5: • Teploty H-můstků. Jednotka [°C].
„motor“	Rx	„engine(1|2)“: Fce 1	Fce 1: • Rychlost elektromotoru (1 nebo 2). Jednotka [cm/s].
„CLOSE“	Tx	---	Slouží jako ukončovací příkaz k zastavení programu na serveru (Python script na Rpi).
„PID“	Tx	„value“: Fce 1	Fce 1: • k1:[0-65535] – Hodnota k nastavení proporcionální složky PID regulátoru. • k2:[0-65535] – Hodnota k nastavení integrační složky PID regulátoru • k3:[0-65535] – Hodnota k nastavení derivační

			složky PID regulátoru - Všechny odeslané hodnoty jsou po přijetí vyděleny 1000. Reálné nastavení hodnot konstant je tedy: 0-65,535.

Tabulka 5.1.1. – tabulka hodnot pro správu komunikace



Obrázek 5.1.2. – vzhled webového rozhraní

Webové rozhraní slouží jako prostředí pro ovládání robota. Toto rozhraní by mělo být přístupné pomocí libovolného zařízení disponující potřebnou softwarovou výbavou a připojením na síť, přes kterou se lze připojit na IP adresu Rpi.

Vzhled rozhraní je velmi jednoduchý. Je pravděpodobný jeho pozdější vývoj (vizuální změny nebo změna ovládacích prvků) v rámci vylepšení přístupu či čitelnosti různých údajů robota.

6. Test

Vyhodnocením efektivity zadávání je možné otestovat rychlost zadávání/diktování a následné provedení jednotlivých příkazů. Pro tento druh testu byl stanoven jednoduchý experiment průjezdu 'bludištěm'. Byly vyzkoušeny 4 různé kombinace ovládání:

- Zadávání pomocí klávesnice
- Ovládání pomocí SpeechCloudu
- Ovládání pomocí modulu EasyVR
- Ovládání pomocí klávesnice a SpeechCloudu

Bludiště se skládá z následujících fází:

1. Jízda vpřed 1 metr nebo do 'zastavení' o zeď.
2. Vytvoření fotografie číslo 1.
3. Otočení kamery doleva.
4. Vytvoření fotografie číslo 2.
5. Otočení kamery doprava (možné během následujícího kroku).
6. Otočení podvozku doleva o 90°.
7. Couvnutí o 0,5 metru.
8. Vytvoření fotografie číslo 3.
9. Otočení kamery vpřed (vycentrování).

Následující tabulka zobrazuje naměřené výsledky (jedná se o průměr hodnot ze tří měření).

Metoda:	Čas:	Počet stisků klávesy:	Pozice dle času:
Klávesnice	(34,8±2,1)s	26	1
SpeechCloud	(46,9±7,5)s	2	3
EasyVR	(120,4±6,1)s	11 (stisků spínače)	4
Klávesnice + SpeechCloud	(37,92±8,51)s	8	2

Tabulka 6.1.1. – tabulka hodnot z testovacího průjezdu bludištěm

Ovládání pomocí klávesnice vychází z časového hlediska nejlépe. Nicméně hlasové ovládání je méně náročnější na zásahy uživatelem.

Závěr

Nasazením hlasového ovládání pro účel navigace a kontroly robota lze docílit jednoduššího ovládání, než použitím klávesnice nebo joysticku. Zadávání příkazů pomocí klávesnice je přesnější a jistější metodou z důvodu absolutního významu hodnot a příkazů. Pro komplexnější řízení však nastává problém s větším a méně přehledným uživatelským rozhraním.

Pokud je použito hlasové ovládání, stačí nám zobrazit pouze zpětná vazba od senzorů a ostatních komponent a není potřeba implementovat ovladače pro nastavení rychlostí, vzdáleností a dalších parametrů. Tyto ovládací prvky mohou snížit efektivitu a přehlednost celkového ovládacího rozhraní.

Použitá literatura

- PSUTKA, Josef. Mluvíme s počítačem česky. Praha: Academia, 2006. Česká matice technická (Academia). ISBN 80-200-1309-1.
- Veear EasyVR 2.0 User Manual 3.6.7 [online], dostupné z: http://www.veear.eu/files/easyvr_user_manual_3.6.7.pdf [cit. 2016-05-02].
- Eris Esgf Compiler [online], dostupné z: http://voice.zcu.cz/VoiceXML/download/doc/esgf.pdf [cit. 2016-05-02].
- Atmel ATMEGA128 Rev. 2467X-AVR-06/11 [online], dostupné z: http://www.atmel.com/images/doc2467.pdf [cit. 2016-05-02].
- Raspberry Pi products and manuals [online] dostupné z: https://www.raspberrypi.org/ [cit. 2016-05-02].

Seznam obrázků, diagramů, kódů a tabulek

Tabulka 2.1.1. – přehled implementovaných příkazů.....7

Tabulka 2.2.1. – vizualizace normalizace hodnot v argumentu.....10

Tabulka 2.2.2. – seznam příkazů potřebné pro hlasové rozpoznávání (ovládání).....10

Tabulka 2.2.3. – Seznam odpovědí na definované příkazy.....11

Tabulka 2.3.1. – Natrénované příkazy.....15

Tabulka 3.3.1. – Seznam zpracovávaných metod a událostí.....20

Tabulka 3.4.1. – tabulka jednotlivých skupin.....25

Tabulka 3.4.2. – hodnoty skupin čísel.....25

Tabulka 4.2.1. – schéma rámce.....33

Tabulka 4.2.2. – velikosti segmentů rámce.....33

Tabulka 4.2.3. – vzorový rámec.....33

Tabulka 4.2.4. – příkazy.....34

Tabulka 4.2.5. – doplňující informace.....35

Tabulka 4.3.1. – tabulka senzorů.....36

Tabulka 4.3.2. – tabulka hodnot IR senzorů.....37

Tabulka 5.1.1. – tabulka hodnot pro správu komunikace.....42

Tabulka 6.1.1. – tabulka hodnot z testovacího průjezdu bludištěm.....45

Obrázek 2.1.1. – Zapojení modulu EasyVR.....9

Obrázek 2.3.1. – nastavení funkčních parametrů.....13

Obrázek 2.3.2. – skupina3 ukázka příkazů.....14

Obrázek 2.3.3. – první fáze trénování příkazu STOP.....14

Obrázek 2.3.4. – druhá fáze trénování příkazu STOP.....14
Obrázek 3.3.1. – grafické znázornění jednoduché gramatiky.....22
Obrázek 3.4.2. – upravená jednoduchá gramatika.....23
Obrázek 4.0.0. – ilustrativní fotografie robota.....29
Obrázek 5.1.1. – vizualizace ovládácích kláves robota.....42
Obrázek 5.1.2. – vzhled webového rozhraní.....44

Kód 2.4.1. – vzorový příklad rozpoznávání.....17

Diagram 2.4.1. – rutina zjišťování stavu modulu.....18
Diagram 3.4.3. – diagram vytvořené gramatiky.....24
Diagram 3.4.4. – diagram znázorňující číselné výrazy.....26
Diagram 4.2.1. – jednoduché blokové schéma propojení důležitých komponent robota.....31

Seznam příloh

Přílohy v textové podobě:

Gramatika ve formátu ESGF [YBPrikazy.txt].....I
Pomocný modul pro HTML rozhraní [control.js].....VI
HTML stránka webového rozhraní [index.html].....VIII
Python script vyhodnocující výsledky z SpeechCloudu [server.py].....XVII
Řídicí python script pro komunikuci s EasyVR modulem [EasyVR.py].....XXV

Gramatika ve formátu ESGF [YBPrikazy.txt]

```
#ESGF V1.0;
grammar YBPrikazy;
public <YBPrikazy> = <BEGIN_GRAMMAR>
        <VIRTUAL>
        (
                (stop|stůj|zastav [se]){STOP} |
                (<go><VIRTUAL><END_GRAMMAR>) |
                (<goContinue><VIRTUAL><goDirection><VIRTUAL>[(<END_GRAMMAR>|
<numbers>)<VIRTUAL><goMetry>]) |
                (<turn><VIRTUAL>[<turnDirection><VIRTUAL>[(<END_GRAMMAR>|
<turnNumber>)<VIRTUAL><turnStupne>]]) |
                (<prikaz><VIRTUAL><doWhat>) |
                (<cameralook><VIRTUAL><cameraTurnDirection><VIRTUAL>[(<END_GRAMMAR>|
<turnNumber>)<VIRTUAL><turnStupne>]) |
                <robotGarbage>
        )
        <VIRTUAL>
        <END_GRAMMAR>;
<cameralook>=(
        ((otoč|natoč) kameru|podívej se) {CAMERALOOK} |
        ((uvítej se){GB} <END_GRAMMAR>)
);
<prikaz>=(
        (udělej|pořiď){DO} |
        (vyfoť){PHOTO} |
        ((hoď|vyřiď){GB} <END_GRAMMAR>)
);
<doWhat>=(
        (fotku|fotografii){PHOTO}
);
<robotGarbage>=(
        (ropot|morot|lolote|bot|rot|brok|mlok|orot|tock) {GB}
        <END_GRAMMAR>
);
<go> = (
        (vpřed|kupředu){FORWARD} |
        (vzad|couvej){BACKWARD} |
        ((krat | mrak | red | brek){GB} <END_GRAMMAR>)
);
```

```
<goContinue>=(
        (jeď|vydej se){GO} |
        ((pět){GB} <END_GRAMMAR>)
);
<goDirection>=(
        (rovně|vpřed|kupředu){FORWARD} |
        (vzad|zpátky){BACKWARD} |
        ((nařadu|rak|vdomě){GB} <END_GRAMMAR>)
);
<goMetry> = (
        (metr | metrů) {M} |
        (centimetr | centimetrů) {CM}
);
<turn>=(
        (otoč se|zatoč){TURN} |
        (kolotoč {GB} <END_GRAMMAR>)
);
<turnDirection>=(
        (doprava|vpravo){RIGHT} |
        (doleva|vlevo){LEFT} |
        ((morava|olovo){GB} <END_GRAMMAR>)
);
<cameraTurnDirection>=(
        <turnDirection> |
        (nahoru|vzhůru){UP} |
        (dolu|k zemi){DOWN} |
        ((dopředu|rovně){CENTER}<END_GRAMMAR>)
);
<turnNumber>=(
        (o|za) <numbers>
);
<turnStupne>=(
        (stupnů|stupen) {DEG} |
        ((vrtů) {GB} <END_GRAMMAR>)
);
<cisla_nula_devet>=(
        (jedna {#1} |
        dva {#2} |
        tři {#3} |
        čtyři {#4} |
```

```
        pět {#5} |
        šest {#6} |
        sedm {#7} |
        osm {#8} |
        devět {#9}) |
        ((vetři|poro) {GB} <END_GRAMMAR>)
);
<cisla_nula_devatenact>=(
        (jedna {#1} |
        dva {#2} |
        tři {#3} |
        čtyři {#4} |
        pět {#5} |
        šest {#6} |
        sedm {#7} |
        osm {#8} |
        devět {#9} |
        deset {#10} |
        jedenáct {#11} |
        dvanáct {#12} |
        třináct {#13} |
        čtrnáct {#14} |
        patnáct {#15} |
        šestnáct {#16} |
        sedmnáct {#17} |
        osmnáct {#18} |
        devatenáct {#19})
        ((vetři|poro|nasázet) {GB} <END_GRAMMAR>)
);
<cisla_dvacet_devadesat>=(
        (dvacet {#20} |
        třicet {#30} |
        čtyřicet {#40} |
        padesát {#50} |
        šedesát {#60} |
        sedmdesát {#70} |
        osmdesát {#80} |
        devadesát {#90})
        ((naramaset) {GB} <END_GRAMMAR>)
);
```

```
<cisla_sto_devetset>=(
        (sto {#100} |
        dvěstě {#200} |
        třista {#300} |
        čtyřista {#400} |
        pětset {#500} |
        šestset {#600} |
        sedmset {#700} |
        osmset {#800} |
        devětset {#900})
        ((okolomeset) {GB} <END_GRAMMAR>)
);
<cisla_tisic>=(
        (tisic|tisíce) {#1000}
);
<dvacet_devadesatdevet>=(
        <cisla_dvacet_devadesat>
        <VIRTUAL>
        [<cisla_nula_devet>]
);
<sto_devetsetdevadesatdevet>=(
        <cisla_sto_devetset>
        <VIRTUAL>
        [<dvacet_devadesatdevet>]
);
<tisic_devetsetdevadesatdevettisic>=(
        <cisla_tisic>
        <VIRTUAL>
        [<sto_devetsetdevadesatdevet>]
);
<komplex_tisice>=(
        <sto_devetsetdevadesatdevet>
        <VIRTUAL>
        <cisla_tisic>
        <VIRTUAL>
        [<sto_devetsetdevadesatdevet>]
);
<numbers>=(
        (<cisla_nula_devatenact>|
        <dvacet_devadesatdevet>|
```

```
        <sto_devetsetdevadesatdevet>|
        <tisic_devetsetdevadesatdevettisic>|
        <komplex_tisice>)
);
```

Pomocný modul pro HTML rozhraní [control.js]

```
$(document).ready(function () {
    var asr_grammar_uri = "http://benesda.4fan.cz/YBPrikazy.txt";
    //var asr_grammar_uri = "http://home.zcu.cz/~benesda/YBPrikazy.txt";
    var SPEECHCLOUD_URI = "https://cak.zcu.cz:9443/v1/speechcloud/edu-benes";
    var sc_options = {
        uri: SPEECHCLOUD_URI,
        tts: '#audioout'
    };
    var speechCloud = null;
    var sc_ready;
    var recognizing = false;
    function init_sc() {
        speechCloud = new SpeechCloud(sc_options);
        speechCloud.on("asr_ready", function () {
            sc_ready = true;
            speechCloud.asr_set_grammar_uri({grammar_type: "esgf", grammar_uri:asr_grammar_uri});
        });
        speechCloud.on("asr_result", function (e) {
            console.log("Recognized", e);
            $('#results').prepend(e.result+'<br/>');
            posliRozpoznani(e.result);
        });
        speechCloud.on("slu_entities", function (e) {
            console.log("SLU:", e);
        });
        speechCloud.on("asr_signal", function (e) {
            if (e.speech) {
                document.getElementById('signal').style.backgroundColor="green";
                if(e.level<10.0)
                    document.getElementById('signal').style.width=(Math.round(10*e.level))+"%";
            else
                document.getElementById('signal').style.width=(Math.round(100))+"%";
            }else{
                document.getElementById('signal').style.backgroundColor="blue";
                if(e.level<10.0)
                    document.getElementById('signal').style.width=(Math.round(10*e.level))+"%";
                else
                    document.getElementById('signal').style.width=(Math.round(100))+"%";
            }
        });
        speechCloud.on('sc_error', function (msg) {
            console.log(msg);
            $('#results').prepend('error: '+msg+'<br/>');
        });
        speechCloud.on('asr_set_grammar_ok', function(){
            console.log('asr_set_grammar_ok');
            document.getElementById('recognition').innerHTML="Recognize Start";
            document.getElementById('recognition').className="controlButtons";
            document.getElementById('recognition').disabled = false;
            $('#results').prepend('asr_set_grammar_ok <br/>');
        });
        speechCloud.on('asr_set_grammar_error', function(msg) {
            console.log('asr_set_grammar_error',msg);
        });
        speechCloud.init();
    }
    init_sc();
```

```
    $('#recognition').click(function (){
        if (!recognizing){
            speechCloud.asr_recognize();
            recognizing = true;
            document.getElementById('recognition').className="controlButtonsRecognize";
            document.getElementById('recognition').innerHTML="Recognize Stop";
        }else{
            speechCloud.asr_pause();
            recognizing = false;
            document.getElementById('recognition').className="controlButtons";
            document.getElementById('recognition').innerHTML="Recognize Start";
      }
    })
});
```

HTML stránka webového rozhraní [index.html]

```
<!DOCTYPE html>
<html lang="cs">
<style>
    .controlButtons {
        color: #900;
        background: #99CCFF;
        font-weight: bold;
        border: 1px solid #900;
        width: 200px;
        margin-top: 5px;
        margin-left: 5px;
    }
    .Dis {
        color: black;
        background: gray;
        font-weight: bold;
        border: 1px solid #900;
        width: 200px;
        margin-top: 5px;
        margin-left: 5px;
    }
    .controlButtonsRecognize {
        color: #900;
        background: #5CD71F;
        font-weight: bold;
        border: 1px solid #900;
        width: 200px;
        margin-top: 5px;
        margin-left: 5px;
    }
    .controlButtons:hover {
        color: #FFF;
        background: #900;
        cursor: pointer;
    }
    .controlButtonsRecognize:hover {
        color: #FFF;
        background: #900;
        cursor: pointer;
    }
    .cameraControl {
        color: #900;
        background: #FF0;
        font-weight: bold;
        border: 1px solid #900;
        width: 70px;
        margin-top: 5px;
    }
    .cameraControl:hover {
        color: #FFF;
        background: #900;
        cursor: pointer;
    }
    .napetiBaterie{
        text-align: right;
        color: green;
    }
```

```
    .teplotaBaterie{
        text-align: right;
        color: #FFCC66;
    }
    .rychlostMotor{
        text-align: right;
        color: black;
    }
    #results {
        border-style: solid;
        border-width: 1px;
        background-color: #E8F2F6;
    }
    #msgs {
        border-style: solid;
        border-width: 1px;
        background-color: #E8F2F6;
    }
</style>
<head>
    <meta charset="utf-8"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=2">
    <title>Mája</title>
    <script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
    <script src="https://cak.zcu.cz:9444/speechcloud.js" type="text/javascript"></script>
    <script src="control.js"></script>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="application/javascript">
        var ws;
        var pausing=1;
        var fpsing=0;
        var ipAdd=document.location.hostname;
        var pripojen=[0,0];
        var tlacitka={"rizeni":{"w":0,"a":0,"s":0,"d":0,"x":0},"kamera":{"q":0,"e":0,"r":0,"f":0}};
        var map = {};
        for(var i=1; i<255; i++)
            map[String.fromCharCode(i)]=false;
        var nastaveni={
            "index":"camera",
            "cameraFormat":
                {
                    "color":1,
                    "resolution":1,
                    "photo":0,
                    "pause":0,
                    "quit":0,
                    "fps":1
                },

        };
        function posliRozpoznani(text){
            wsComm.send(JSON.stringify(
                {"index":"speech",
                    "Recognition":text,}
                ));
        }
        function buttons(znak){
            if(znak=="W" || znak=="S" || znak=="A" || znak=="D" || znak=="X"){
                if(map["W"])
```

```
                tlacitka["rizeni"]["w"]=1;
            else
                tlacitka["rizeni"]["w"]=0;
            if(map["S"])
                tlacitka["rizeni"]["s"]=1;
            else
                tlacitka["rizeni"]["s"]=0;
            if(map["A"])
                tlacitka["rizeni"]["a"]=1;
            else
                tlacitka["rizeni"]["a"]=0;
            if(map["D"])
                tlacitka["rizeni"]["d"]=1;
            else
                tlacitka["rizeni"]["d"]=0;
            if(map["X"])
                tlacitka["rizeni"]["x"]=1;
            else
                tlacitka["rizeni"]["x"]=0;
            wsComm.send(JSON.stringify({"index":"direct","input":tlacitka["rizeni"]}));
        }
        if(znak=="Q" || znak=="E" || znak=="R" || znak=="F"){
            if(map["Q"])
                tlacitka["kamera"]["q"]=1;
            else
                tlacitka["kamera"]["q"]=0;
            if(map["E"])
                tlacitka["kamera"]["e"]=1;
            else
                tlacitka["kamera"]["e"]=0;
            if(map["R"])
                tlacitka["kamera"]["r"]=1;
            else
                tlacitka["kamera"]["r"]=0;
            if(map["F"])
                tlacitka["kamera"]["f"]=1;
            else
                tlacitka["kamera"]["f"]=0;
            wsComm.send(JSON.stringify({"index":"directKamera","input":tlacitka["kamera"]}));
        }
    }
    onkeydown = onkeyup = function(e){
        e = e || event;
        map[String.fromCharCode(e.keyCode)] = e.type == 'keydown';
        buttons(String.fromCharCode(e.keyCode));
    }

    function evaluation(data){
        text=JSON.parse(data);
        switch(text["index"]){
            case "baterie":
                document.getElementById("voltageBattery0").innerHTML=text["vBat0"]+"V";
                document.getElementById("voltageBattery1").innerHTML=text["vBat1"]+"V";
                document.getElementById("voltageBattery2").innerHTML=text["vBatSum"]+"V -
"+text["proud"]+" A";
                document.getElementById("tempBattery0").innerHTML=text["tempBat1"]+"°C";
                document.getElementById("tempBattery1").innerHTML=text["tempBat2"]+"°C";
                document.getElementById("tempBatteryC").innerHTML="avg.
"+Math.round((text["tempBat1"]+text["tempBat2"])*100/2)/100+"°C";
                document.getElementById("tempHB1").innerHTML=text["tempHB1"]+"°C";
```

```
                document.getElementById("tempHB2").innerHTML=text["tempHB2"]+"°C";
            break;
            case "motor":
                document.getElementById("speedEngine1").innerHTML=text["engine1"]+"cm/s";
                document.getElementById("speedEngine2").innerHTML=text["engine2"]+"cm/s";
            break;
            case "servos":
                document.getElementById("camVerticalAngleLine").setAttribute("transform", "rotate("+(125-
text["angle2"])+",50,415)");
                document.getElementById("camHorizontalAngleLine").setAttribute("transform", "rotate("+
(text["angle1"]-125)+",180,415)");
                document.getElementById("camVerticalAngleText").innerHTML=(125-text["angle2"])+"°
("+(125-text["angle2"])+"°)";
                document.getElementById("camHorizontalAngleText").innerHTML=(-125+text["angle1"])+"°
("+(text["angle1"]-35)+"°)";
            break;
        }
    }
    function readSingleFile(evt) {
        var f = evt;
        var r = new FileReader();
        r.onload = function(e) {
            evaluation(r.result);
        }
        r.readAsText(f)
    }

    function init() {
        wsCamera = new WebSocket("wss://"+ipAdd+":8080/chat");
        wsCamera.binaryType="arraybuffer";
        wsComm = new WebSocket("wss://"+ipAdd+":8081/chat");
        wsCamera.onopen = function(){
            msgs.innerHTML = "Server connected - Camera<br>"+msgs.innerHTML;
            pripojen[0]=1;
        };
        wsCamera.onmessage = function(e){
            var urlCreator = window.URL || window.webkitURL;
            var imageUrl=urlCreator.createObjectURL(new Blob([e.data],{type:"text/plain"}));
            document.getElementById("image").src=imageUrl;
        };
        wsCamera.onclose = function(){
            msgs.innerHTML = "Server disconnected - Camera<br>"+msgs.innerHTML;
            if(pripojen[0])
                wsCamera = new WebSocket("wss://"+ipAdd+":8080/chat");
        };
        wsComm.onopen = function(){
            msgs.innerHTML = "Server connected - Comm<br>"+msgs.innerHTML;
            pripojen[1]=1;
        };
        wsComm.onmessage = function(e){
            evaluation(e.data);
        };
        wsComm.onclose = function(){
            msgs.innerHTML = "Server disconnected - Comm<br>"+msgs.innerHTML;
            if(pripojen[1])
                wsComm = new WebSocket("wss://"+ipAdd+":8081/chat");
        };
    }
    function sendNastaveni(){
        wsComm.send(JSON.stringify(nastaveni));
```

```
            nastaveni["cameraFormat"]["photo"]=0;
        }
        function radioButtons1(){
            var temp = document.getElementsByName("COLOR");
            for(var i=0; i<temp.length; i++){
                if(temp[i].checked){
                    nastaveni["cameraFormat"]["color"]=i;
                    break;
                }
            }
            sendNastaveni();
        }
        function radioButtons2(){
            var temp = document.getElementsByName("RESOLUTION");
            for(var i=0; i<temp.length; i++){
                if(temp[i].checked){
                nastaveni["cameraFormat"]["resolution"]=i;
                break;
                }
            }
            sendNastaveni();
        }
        function turnoff(){
            if(confirm('Are you sure? You will have to turn on whole program manualy!'))
                wsComm.send(JSON.stringify({"index":"CLOSE"}));
        }
        function sendkonst(){
            var k1=document.getElementById("Kp").value;
            var k2=document.getElementById("Ki").value;
            var k3=document.getElementById("Kd").value;
            k1=parseInt(k1)||0;
            k2=parseInt(k2)||0;
            k3=parseInt(k3)||0;
            console.log(k1,k2,k3);
            wsComm.send(JSON.stringify({"index":"PID","value":[k1,k2,k3]}));
        }
    </script>
</head>
<body onload="init();" style="background-color:#F2F6E8; ">
    <h2>Mája</h2>
    <div id="controls" style="float:none">
        <div style="float:left">
            <img id="image" name="image" style="height:480px; width:640px"/><br>
            <table>
                <tr><td>
                    Volba barevného formátu:
                    <form onchange="radioButtons1()">
                    <input type="radio" name="COLOR" value="0">RGB<br>
                    <input type="radio" name="COLOR" value="1" checked>B/W<br>
                    <input type="radio" name="COLOR" value="2">HSV<br>
                    </form></td>
                <td>
                    Volba rozlišení kamery:
                    <form onchange="radioButtons2()">
                        <input type="radio" name="RESOLUTION" value="0">640x480<br>
                        <input type="radio" name="RESOLUTION" value="1" checked>320x240<br>
                        <input type="radio" name="RESOLUTION" value="2">160x120<br>
                    </form></td>
                <td>
                    <input class="cameraControl" type="button" value="PHOTO"
```

```
onclick='nastaveni["cameraFormat"]["photo"]=1;sendNastaveni();' /><br>
                    <input class="cameraControl" type="button" value="PAUSE"
onclick='nastaveni["cameraFormat"]["pause"]=pausing;pausing=(pausing+1)%2;sendNastaveni();' /><br>
                    <input class="cameraControl" type="button" value="FPS"
onclick='nastaveni["cameraFormat"]["fps"]=fpsing;fpsing=(fpsing+1)%2;sendNastaveni();' /><br>
                </td>
                <td>
                    <button class="Dis" id="recognition" disabled="true">Recognize non-function</button><br>
                    <input class="controlButtons" type="button" onclick="turnoff();" value="Switch off program"
/><br>
                </td>
                </tr>
            </table>
        </div>
        <div>
            <svg style="height:480px; width:240px">
                <rect x="40" y="40" width="160" height="240" style="fill:black;stroke-width:0;stroke:none"/>
                <rect x="90" y="60" width="60" height="60" style="fill:black;stroke-width:10;stroke:yellow"/>
                <rect x="95" y="120" width="15" height="110" style="fill:yellow;stroke-width:0;stroke:none"/>
                <rect x="130" y="120" width="15" height="110" style="fill:yellow;stroke-width:0;stroke:none"/>
                <rect x="40" y="230" width="160" height="16" style="fill:yellow;stroke-width:0;stroke:none"/>
                <rect x="40" y="196" width="55" height="16" style="fill:yellow;stroke-width:0;stroke:none"/>
                <rect x="145" y="196" width="55" height="16" style="fill:yellow;stroke-width:0;stroke:none"/>
                <rect id="whfloorsz1" x="10" y="50" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz2" x="25" y="50" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz3" x="205" y="50" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz4" x="220" y="50" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz5" x="10" y="260" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz6" x="25" y="260" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz7" x="205" y="260" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="whfloorsz8" x="220" y="260" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="frontsz1" x="50" y="1" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="frontsz2" x="83" y="1" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="frontsz3" x="115" y="1" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="frontsz4" x="148" y="1" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="frontsz5" x="180" y="1" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="backsz1" x="50" y="300" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="backsz2" x="83" y="300" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="backsz3" x="115" y="300" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="backsz4" x="148" y="300" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="backsz5" x="180" y="300" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
```

```
                <rect id="frontanglesz1" x="66" y="15" width="10" height="10" style="fill:none;stroke-
width:2;stroke:purple"/>
                <rect id="frontanglesz2" x="115" y="15" width="10" height="10" style="fill:none;stroke-
width:2;stroke:purple"/>
                <rect id="frontanglesz3" x="164" y="15" width="10" height="10" style="fill:none;stroke-
width:2;stroke:purple"/>
                <rect id="frontfloorsz1" x="50" y="27" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="frontfloorsz2" x="93" y="27" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="frontfloorsz3" x="137" y="27" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="frontfloorsz4" x="180" y="27" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="backfloorsz1" x="50" y="285" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="backfloorsz2" x="93" y="285" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="backfloorsz3" x="137" y="285" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="backfloorsz4" x="180" y="285" width="10" height="10" style="fill:none;stroke-
width:2;stroke:blue"/>
                <rect id="wallsz1" x="10" y="65" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="wallsz2" x="220" y="65" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="wallsz3" x="10" y="245" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>
                <rect id="wallsz4" x="220" y="245" width="10" height="10" style="fill:none;stroke-
width:2;stroke:red"/>

                <text x="20" y="325" fill="blue" >Pozemní IR sensor</text>
                <text x="20" y="338" fill="red" >Horizontální IR sensor</text>
                <text x="20" y="351" fill="purple" >Pozemní IR sensor 45°</text>

                <text x="22" y="380">Natočení kamery:</text>
                <circle cx="50" cy="415" r="30" style="fill:green;stroke:none"/>
                <line id="camVerticalAngleLine" x1="50" y1="415" x2="50" y2="385"
style="fill:none;stroke:black;stroke-width:3"/>
                <circle cx="180" cy="415" r="30" style="fill:blue;stroke:none"/>
                <line id="camHorizontalAngleLine" x1="180" y1="415" x2="150" y2="415"
style="fill:none;stroke:black;stroke-width:3"/>
                <text x="22" y="460" fill="green">Horizont</text>
                <text x="155" y="460" fill="blue">Vertical</text>
                <text id="camVerticalAngleText" x="22" y="475" fill="green">000° (000°)</text>
                <text id="camHorizontalAngleText" x="155" y="475" fill="blue">000° (090°)</text>
            </svg>
        </div>
    </div><br>
    <div id="sensors" style="width:640px">
        <table border align="left">
            <tr>
                <td>
                    Článek:
                </td>
                <td>
                    1 (3,70V)
                </td>
                <td>
                    2 (3,70V)
```

```
            </td>
            <td>
                celkem (7,40V)
            </td>
        </tr>
        <tr>
            <td>
                Napětí baterie:
            </td>
            <td class="napetiBaterie" id="voltageBattery0">
                0,00V
            </td>
            <td class="napetiBaterie" id="voltageBattery1">
                0,00V
            </td>
            <td class="napetiBaterie" id="voltageBattery2">
                0,00V - 0,00A
            </td>
        </tr>
        <tr>
            <td>
                Teplota baterie:
            </td>
            <td class="teplotaBaterie" id="tempBattery0">
                0°C
            </td>
            <td class="teplotaBaterie" id="tempBattery1">
                0°C
            </td>
            <td class="teplotaBaterie" id="tempBatteryC">
                avg.: 0°C
            </td>
        </tr>
    </table>
    <table border>
        <tr>
            <td>
                Motor
            </td>
            <td>
                Engine 1
            </td>
            <td>
                Engine 2
            </td>
        </tr>
¨       <tr>
            <td>
                Teplota H-B:
            </td>
            <td class="teplotaBaterie" id="tempHB1">
                0°C
            </td>
            <td class="teplotaBaterie" id="tempHB2">
                0°C
            </td>
        </tr>
        <tr>
            <td>
                Rychlost:
```

```
                </td>
                <td class="rychlostMotor" id="speedEngine1">
                    000 cm/s
                </td>
                <td class="rychlostMotor" id="speedEngine2">
                    000 cm/s
                </td>
            </tr>
        </table>
    </div>
    <div style="width:200px">
        <div id="signal" style="width:0%; background-color:blue">
            <span>|</span>
        </div>
    </div>
    <b>Speech recognition logger:</b>
    <div id="results" style="overflow-y: scroll; height:150px;"><br></div>
    <b>Websocket logger:</b>
    <div id="msgs" style="overflow-y: scroll; height:80px;"><br></div>
</body>
</html>
```

Python script zpracovávající výsledky z SpeechCloudu [server.py]

```
import tornado.ioloop
import tornado.web
import tornado.websocket
import threading, time, multiprocessing, math, sys, StringIO, json, base64, serial, re, struct, random
import numpy as np
import cv2 as cv
import RPi.GPIO as GPIO
from PIL import Image
from datetime import datetime
from picamera.array import PiRGBArray
from picamera import PiCamera

clients={"com":0,"cam":0}
fronta=multiprocessing.Queue()
ser=serial.Serial("/dev/ttyAMA0",9600)
ser.parity=serial.PARITY_ODD
ser.stopbits=serial.STOPBITS_TWO
print ser
while not ser.open:
    time.sleep(0.1)
ser.write(bytearray([3,0,3]))
smery=[1,1]

class IndexHandler(tornado.web.RequestHandler):
    def get(request):
        request.render("index.html")

class WebSocketHandler2(tornado.websocket.WebSocketHandler):
    def open(self):
        print("Websocket number 1 opened")
        clients["com"]=self
    def on_message(self, message):
        fronta.put(message)
    def on_close(self):
        print("Websocket number 1 closed")
        clients["com"]=0
    def check_origin(self,origin):
        return True

class WebSocketHandler1(tornado.websocket.WebSocketHandler):
    def open(self):
        print("Websocket number 2 opened")
        clients["cam"]=self
    def on_message(self, message):
        pass
    def on_close(self):
        print("Websocket number 2 closed")
        clients["cam"]=0
    def check_origin(self,origin):
        return True

def posli(data):
    tosend=len(data)+2
    crc=tosend
    dats=[tosend,]
    for i in data:
        crc=crc^i
```

```
    dats.append(i)
    dats.append(crc)
    print dats
    ser.write(bytearray(dats))

def camera(com,info):
    setup=[1,1,0,0,1,0] #color, fps, pause, photo, format, quit
    cam=PiCamera()
    cam.resolution=(640,480)
    cam.framerate=10
    cam.vflip=True
    rawCapture=PiRGBArray(cam,size=(640,480))
    output=StringIO.StringIO()
    cas=time.time()
    velikost=1
    for frame in cam.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        tt=time.time()
        casT=time.time()
        frame=frame.array
        rawCapture.truncate(0)
        if(info.poll()):
            setupT=info.recv()
            if("PHOTO"==setupT):
                setup[3]=1
            else:
                setup=setupT
                if(setup[5]):
                    break
        if(setup[2]):
            time.sleep(0.2)
        else:
            if setup[3]:
                datum=datetime.now()
                cv.imwrite(datum.strftime("%Y_%m_%d_%H_%M_%S"+".jpg"),frame)
                print "Photo taken: "+datum.strftime("%Y/%m/%d %H:%M:%S"+".jpg")
                setup[3]=0
            if setup[0]==0:
                frame=cv.cvtColor(frame,cv.COLOR_RGB2BGR)
            elif setup[0]==1:
                frame=cv.cvtColor(frame,cv.COLOR_RGB2GRAY)
            elif setup[0]==2:
                frame=cv.cvtColor(frame,cv.COLOR_RGB2HSV)
            if setup[4]==0:
                velikost=1
                pass
            elif setup[4]==1:
                velikost=0.5
                frame=cv.resize(frame,(320,240))
            elif setup[4]==2:
                velikost=0.25
                frame=cv.resize(frame,(160,120))
            if setup[1]:
                cv.putText(frame, str(round(1/cas,2)), (10,int(30*velikost)), cv.FONT_HERSHEY_SIMPLEX,
velikost, (0,255,0))
            img=Image.fromarray(frame)
            output.seek(0)
            img.save(output,"JPEG")
            try:
                com.write_message(output.getvalue(),binary=True)
            except:
```

```
                print "Websocket CAM is closed"
                time.sleep(2)
            cas=time.time()-casT
            waitTime=0.070-cas
            if waitTime>0:
                time.sleep(waitTime)
            cas=cas+waitTime
    cam.close()
    print "Exiting CAM process"

def start():
    tornado.ioloop.IOLoop.instance().start()

def vyhodnot(data):
    if "[GB]" not in data:
        matchs = re.findall(r"\[#\d+\]",data,)
        numbers=[]
        hodnota=0
        if len(matchs)>=1:
            for i in matchs:
                numbers.append(int(i[2:len(i)-1]))
            cisla=[]
            od=0
            pokracuj=True
            posledni=od
            if len(numbers)==1:
                cisla.append(numbers[0])
                pokracuj=False
            while pokracuj:
                if od==0 and numbers[od]==1000:
                    temp=numbers[od]
                elif od==0 and numbers[od]!=1000:
                    temp=numbers[od]
                elif od>0 and numbers[od]!=1000:
                    temp=numbers[od]
                if temp>0:
                    for i in range(od,len(numbers)-1):
                        if numbers[i]>numbers[i+1]:
                            temp=temp+numbers[i+1]
                        else:
                            od=i+1
                            break
                    if posledni==od:
                        pokracuj=False
                    posledni=od
                    cisla.append(temp)
                    temp=0
                    if od==0:
                        break
                else:
                    od=od+1
            cislo=cisla[0]
            if len(cisla)>1:
                cislo=cislo*1000+cisla[1]
            hodnota=cislo
        if "[FORWARD]" in data:
            smery=[1,1]
            if hodnota==0:
                return bytearray([5,rych,rych,0b10001000])
            else:
```

```
            hodnota=int(round(cmtops*hodnota))
        if "[CM]" in data:
            hodnota=hodnota*1
        elif "[M]" in data:
            hodnota=hodnota*100
        return bytearray([6,rych,rych,0b10001000,
            (hodnota>>24)&0xff,(hodnota>>16)&0xff,
            (hodnota>>8)&0xff,(hodnota)&0xff,
            (hodnota>>24)&0xff,(hodnota>>16)&0xff,
            (hodnota>>8)&0xff,(hodnota)&0xff])
    elif "[BACKWARD]" in data:
        smery=[-1,-1]
        if hodnota==0:
            return bytearray([5,rych,rych,0b01000100])
        else:
            hodnota=int(round(cmtops*hodnota))
            if "[CM]" in data:
                hodnota=hodnota*1
            elif "[M]" in data:
                hodnota=hodnota*100
            return bytearray([6,rych,rych,0b01000100,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff])
    if "[TURN]" in data:
        if "[RIGHT]" in data:
            smery=[1,1]
            if hodnota==0:
                hodnota=int(round(degtopsCorner*90))
            else:
                hodnota=int(round(degtopsCorner*(hodnota%360)))
            return bytearray([6,rych,0,0b10000010,0,0,0,0,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff])
        elif "[LEFT]" in data:
            smery=[1,1]
            if hodnota==0:
                hodnota=int(round(degtopsCorner*90))
            else:
                hodnota=int(round(degtopsCorner*(hodnota%360)))
            return bytearray([6,0,rych,0b00101000,0,0,0,0,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff])
        else:
            otocka=0b00101000
            smery=[1,1]
            if random.random()>0.5:
                otocka=0b10000010
                smery=[1,1]
                hodnota=int(round(180*degtopsCorner))
            return bytearray([6,rych,rych,otocka,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff,
                (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                (hodnota>>8)&0xff,(hodnota)&0xff])
    elif "[STOP]" in data:
        return bytearray([5,0,0,0b00100010])
    elif "[PHOTO]" in data:
        return "PHOTO"
```

```
        elif "[CAMERALOOK]" in data:
            if hodnota==0:
                if "[RIGHT]" in data:
                    return bytearray([8,0])
                if "[LEFT]" in data:
                    return bytearray([8,250])
                if "[DOWN]" in data:
                    return bytearray([10,0])
                if "[UP]" in data:
                    return bytearray([10,250])
                if "[CENTER]" in data:
                    return bytearray([8,125])
            else:
                hodnota=int(round(hodnota*1.722))
                if ("[RIGHT]" in data or "[UP]" in data):
                    hodnota=125+hodnota
                if ("[LEFT]" in data or "[DOWN]" in data:
                    hodnota=-hodnota+125
                if hodnota>250:
                    hodnota=250
                if hodnota<0:
                    hodnota=0
                if ("[DOWN]" in data or "[UP]" in data):
                    return bytearray([10,hodnota])
                if ("[LEFT]" in data or "[RIGHT]" in data):
                    return bytearray([8,hodnota])

def process(data):
    if(data[1]==1):
        napeti1=data[2]*256+data[3]
        napeti2=data[4]*256+data[5]
        proud=(data[6]*256+data[7])*5.0/1024.0
        tempBat1=data[8]*256+data[9]
        tempBat2=data[10]*256+data[11]
        reserva=data[12]*256+data[13]
        tempHB1=data[14]*256+data[15]
        tempHB2=data[16]*256+data[17]
        napeti2=napeti2*4.97/1024
        napeti1=napeti1*9.8/1024-napeti2
        napetiCelek=napeti1+napeti2
        proud=proud/(0.037*9)
        tempHB1=73.80904-0.147303*tempHB1
        tempHB2=73.80904-0.147303*tempHB2
        tempBat1=107.0507-0.1420*tempBat1
        tempBat2=107.0507-0.1420*tempBat2
        dat={"index":"baterie",
            "vBat0":round(napeti1,2),"vBat1":round(napeti2,2),"vBatSum":round(napetiCelek,2),
            "proud":round(proud,2),"tempBat1":round(tempBat1,2),"tempBat2":round(tempBat2,2),
            "tempHB1":round(tempHB1,2),"tempHB2":round(tempHB2,2)}
        try:
            clients["com"].write_message(json.dumps(dat))
        except:
            pass
    elif(data[1]==3):
        rychlost1=smery[0]*(data[2])*4/cmtops
        rychlost2=smery[1]*(data[3])*4/cmtops
        dat={"index":"motor",
            "engine1":int(round(rychlost1)),
            "engine2":int(round(rychlost2))}
        try:
```

```
            clients["com"].write_message(json.dumps(dat))
        except:
            pass

def batTest(data,switchOff):
    while True:
        if switchOff.poll():
            break
        try:
            time.sleep(0.25)        #time 0.25
            ser.write(data[0])
            ser.write(data[1])
            posli(0)
            time.sleep(0.25)        #time 0.5
            ser.write(data[2])
            ser.write(data[1])
            time.sleep(0.25)        #time 0.75
            ser.write(data[1])
            posli(0)
            time.sleep(0.25)        #time 1
            ser.write(data[1])
        except:
            pass

def uart(link):
    batteryTest=bytearray([3,1,2])
    engineTest=bytearray([3,3,0])
    lengthTest=bytearray([3,11,8])
    datMain,datDef=multiprocessing.Pipe()
    threading.Thread(target=batTest,args=([batteryTest,engineTest,lengthTest],datDef)).start()
    while True:
        temp=""
        prijem=[]
        if link.poll():
            temp=link.recv()
        if temp=="CLOSE":
            datMain.send("CLOSE")
            break
        else:
            delka=ord(ser.read(1))
            if delka>0 and delka<128:
                prijem.append(delka)
                koler=ser.read(delka-1)
                for i in koler:
                    prijem.append(ord(i))
                CRC=prijem[-1]
                for i in range(len(prijem)-1):
                    CRC^=prijem[i]
                if CRC==0:
                    process(prijem)
                else:
                    print "Error CRC: ",prijem
            else:
                ser.read()

app1=[(r"/chat",WebSocketHandler1),(r"/",IndexHandler)]
app2=[(r"/chat",WebSocketHandler2),(r"/",IndexHandler)]
app1 = tornado.web.Application(app1)
app2 = tornado.web.Application(app2)
app1.listen(8080,ssl_options={"certfile":"apache.crt","keyfile":"apache.key"})
```

```
app2.listen(8081,ssl_options={"certfile":"apache.crt","keyfile":"apache.key"})

infoGet, infoSend=multiprocessing.Pipe()
uartMain, uartDef=multiprocessing.Pipe()
threading.Thread(target=start).start()

while(clients["cam"]==0 and clients["com"]==0):
    time.sleep(0.1)
multiprocessing.Process(target=camera,args=(clients["cam"],infoGet)).start()
threading.Thread(target=uart,args=(uartDef,)).start()

cont=True
uhel=125
uhel2=125
posl=0
cmtops=12.59694 #139.826 = 111mm
degtops=3.1055    #887.4999mm obvod (otocka na miste), 1deg =2.4653mm=3.1055ps
degtopsCorner=cmtops*2          #jeden motor brzdi
rych=60
tempCAM=clients["cam"]
uhelT=125
uhelT2=125
while cont:
    time.sleep(0.1)
    if(tempCAM != clients["cam"] and clients["cam"]!=0):
        tempCAM = clients["cam"]
        infoSend.send([0,0,0,0,0,1])
        time.sleep(1)
        multiprocessing.Process(target=camera,args=(clients["cam"],infoGet)).start()
    while(not fronta.empty()):
        temp = json.loads(fronta.get())
        if temp["index"]=="PID":
            posli([7,temp["value"][0]/256,temp["value"][0]%256,temp["value"][1]/256,temp["value"]
[1]%256,temp["value"][2]/256,temp["value"][2]%256])
        if temp["index"]=="direct":
            inp=temp["input"]
            if(inp["w"]):
                smery=[1,1]
                posli([5,rych,rych,0b10001000])
            elif(inp["s"]):
                posli([5,rych,rych,0b01000100])
                smery=[-1,-1]
            elif(inp["a"]):
                posli([5,0,rych,0b00101000])
                smery=[1,1]
            elif(inp["d"]):
                smery=[1,1]
                posli([5,rych,0,0b10000010])
            elif(inp["x"]):
                posli([5,0,0,0b00100010])
        if temp["index"]=="directKamera":
            if(temp["input"]["e"]):
                if(uhel>=5):
                    uhel=uhel-5
            if(temp["input"]["q"]):
                if(uhel<=245):
                    uhel=uhel+5
            if(temp["input"]["r"]):
                if(uhel2<=245):
                    uhel2=uhel2+5
```

```
            if(temp["input"]["f"]):
                if(uhel2>=5):
                    uhel2=uhel2-5
            posli([8,uhel])
            posli([10,uhel2])
        elif temp["index"]=="speech":
            navratovaHodnota=vyhodnot(temp["Recognition"])
            if navratovaHodnota=="PHOTO":
                infoSend.send("PHOTO")
            elif navratovaHodnota:
                posli(navratovaHodnota)
        elif temp["index"]=="CLOSE":
            tornado.ioloop.IOLoop.instance().stop()
            cont=False
            infoSend.send([0,0,0,0,0,1])
        elif temp["index"]=="camera":
            dosud=[temp["cameraFormat"]["color"],temp["cameraFormat"]["fps"],temp["cameraFormat"]
["pause"],temp["cameraFormat"]["photo"],temp["cameraFormat"]["resolution"],temp["cameraFormat"]["quit"]]
            infoSend.send(dosud)
        if uhelT != uhel or uhelT2!=uhelT:
            uhelT=uhel
            uhelT2=uhel2
            clients["com"].write_message(json.dumps({"index":"servos","angle1":uhel2,"angle2":uhel}))
infoSend.send([0,0,0,0,0,1])
tornado.ioloop.IOLoop.instance().stop()
uartMain.send("CLOSE")

time.sleep(2)
try:
    ser.close()
except:
    pass
```

Řídicí python script pro komunikuci s EasyVR modulem [EasyVR.py]

```
import threading, time, sys, StringIO, serial
import numpy as np
import cv2 as cv
import RPi.GPIO as GPIO
from PIL import Image
from datetime import datetime
from picamera.array import PiRGBArray
from picamera import PiCamera

ser=serial.Serial("/dev/ttyAMA0",19200)
ser.parity=serial.PARITY_ODD
ser.stopbits=serial.STOPBITS_TWO
print ser
while not ser.open:
   time.sleep(0.1)
ser2=serial.Serial("COM5",9600,timeout=5)
print ser2
while not ser2.open:
    time.sleep(0.1)

errors={0x03:"too noisy", 0x04:"spoke too soft", 0x05:"spoke too loud", 0x06:"spoke too soon",
0x07:"too many segments/too complex", 0x11:"recognition failed", 0x12:"recognition result doubful",
0x13:"recognition result maybe", 0x14:"invalid SD/SV command stored in memory", 0x17:"bad pattern
durations",
0x4A:"bad release number in speech file", 0x4E:"bad data in speech file or invalid compression",
0x80:"recognized word is not in vocabulary"}

def posli(data):
   tosend=len(data)+2
   crc=tosend
   dats=[tosend,]
   for i in data:
      crc=crc^i
      dats.append(i)
   dats.append(crc)
   print dats
   ser.write(bytearray(dats))
```

```
def camera():
    cam=PiCamera()
    cam.resolution=(640,480)
    cam.framerate=10
    cam.vflip=True
     datum=datetime.now()
     cam.capture(datum.strftime("%Y_%m_%d_%H_%M_%S"+".jpg"))
     print "Photo taken: "+datum.strftime("%Y/%m/%d %H:%M:%S"+".jpg")
     cam.close()
def receive():
     temp=ser2.read(1)
     ser2.write(' ')
     if(temp=='x'):     #Version
          temp=ser2.read(1)
          ser2.write(' ')
          return ["V",'Version :'+temp]
     if(temp=='o'):     #Success
          temp=ser2.read(1)
          ser2.write(' ')
          return ["S",'Operation success'+temp]
     if(temp=='e'):     #Error
          temp=(ord(ser2.read(1))-65)*16
          ser2.write(' ')
          temp=temp+(ord(ser2.read(1))-65)
          ser2.write(' ')
          return ["E",'Error: '+errors[temp]]
     if(temp=='p'):     #PIN Status
          temp=ord(ser2.read(1))-65
          ser2.write(' ')
          return ["P",temp]
     if(temp=='r'):#Result
          temp=ord(ser2.read(1))-65
          ser2.write(' ')
          return ["R",temp]
     if(temp=='s'):     #Similar
          temp=ord(ser2.read(1))-65
          ser2.write(' ')
          return ["S",temp]
     if(temp=='t'):#Timeout
          return ["T",0]
```

```
    if(temp=='v'):     #Invalid
        return ["V",0]
def setup():
    ser2.write('o')
    ser2.write('B')
    print receive()[1]
    ser2.write('y')
    ser2.write(chr(ord('A')+10))
    print receive()[1]
def checkPIN():
    ser2.write('q')
    ser2.write('B')
    ser2.write('D')
    if receive()[1]==1:
        return False
    else:
        return True
def checkFirmware():
    ser2.write('x')
    print receive()[1]
def playAudio():
    ser2.write('w')
    ser2.write('A')
    ser2.write('A')
    ser2.write('P')
    print receive()[1]
def recogSD(index):
    ser2.write('d')
    ser2.write(chr(ord('A')+index))
    temp=receive()
    print temp
    return temp
def trippleBEEP():
    playAudio()
    playAudio()
setup()
checkFirmware()
uhel=125
uhel2=125
rych=60
```

```
cmtops=12.59694 #139.826 = 111mm
degtops=3.1055     #887.4999mm obvod (otocka na miste), 1deg =2.4653mm=3.1055ps
degtopsCorner=cmtops*2          #jeden motor brzdi
while True:
    time.sleep(0.5)
    if(checkPIN()):
        playAudio()
        temp=recogSD(2)
        if(temp==["R",0]):
            print "Jed"
            playAudio()
            temp=recogSD(3)
            if(temp==["R",0]):
                print "vpred"
                posli([5,rych,rych,0b10001000])
            elif(temp==["R",1]):
                print "vzad"
                posli([5,rych,rych,0b01000100])
            elif(temp==["R",2]):
                print "doleva"
                hodnota=int(round(degtopsCorner*90))
                posli([6,0,rych,0b00101000,0,0,0,0,
                    (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                    (hodnota>>8)&0xff,(hodnota)&0xff])
            elif(temp==["R",3]):
                print "doprava"
                hodnota=int(round(degtopsCorner*90))
                posli([6,rych,0,0b10000010,0,0,0,0,
                    (hodnota>>24)&0xff,(hodnota>>16)&0xff,
                    (hodnota>>8)&0xff,(hodnota)&0xff])
            else:
                print temp[1]
                trippleBEEP()
        elif(temp==["R",1]):
            print "Otoc kameru"
            playAudio()
            temp=recogSD(4)
            if(temp==["R",0]):
                print "dolu"
                uhel2=0
```

```
    elif(temp==["R",1]):
        print "nahoru"
        uhel2=250
    elif(temp==["R",2]):
        print "doleva"
        uhel=0
    elif(temp==["R",3]):
        print "doprava"
        uhel=250
    else:
        print temp[1]
        trippleBEEP()
    posli([8,uhel])
    posli([10,uhel2])
elif(temp==["R",2]):
    print "Udelej fotku"
    camera()
elif(temp==["R",3]):
    print "Vycentruj kameru"
    uhel=125
    uhel2=125
    posli([8,uhel])
    posli([10,uhel2])
elif(temp==["R",4]):
    print "Otoc se"
    playAudio()
    temp=recogSD(4)
    if(temp==["R",2]):
        print "doleva"
        posli([5,0,30,0b00101000])
    elif(temp==["R",3]):
        print "doprava"
        posli([5,30,0,0b10000010])
    else:
        print temp[1]
        trippleBEEP()
elif(temp==["R",5]):
    print "Zastav se"
    posli([5,0,0,0b00100010])
else:
```

```
            print temp
            trippleBEEP()
    print "Neaktivni"
ser2.close()
ser.close()
```