Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Inteligentní domácnost s interaktivním hlasovým ovládáním

PLZEŇ, 2016 JAKUB NEDVĚD

P R O H L Á Š E N Í

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

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

V Plzni dne 6. 5. 2016

.....

vlastnoruční podpis

PODĚKOVÁNÍ

Touto cestou bych rád poděkoval vedoucímu mé diplomové práce Doc. Ing. Luďkovi Müllerovi, PhD. za cenné rady a připomínky během psaní diplomové práce.

Dále bych rád poděkoval Ing. Janu Švecovi, PhD. a Ing. Luboši Šmídlovi, PhD. za spolupráci, nápady a vřelý přístup po celou dobu psaní práce a samotného studia. Díky patří také Ing. Jaroslavu Sobotovi, PhD. za zapůjčení potřebného vybavení a pomoc při propojení mého systému.

V neposlední řadě bych rád poděkoval svým přátelům, kteří se na vyhodnocení práce podíleli, za jejich postřehy a nápady. Na závěr děkuji také svým rodičům za hodnocení mé práce a upozornění na chyby a jejich podporu během mého studia.

Anotace

Cílem diplomové práce je vytvoření inteligentní domácnosti s hlasovým ovládáním jakožto ukázkové aplikace, která bude využívat komponenty vzniklé na katedře kybernetiky a ve firmě SpeechTech. Tato aplikace bude nabízet kromě hlasového i dotykové ovládání a obě části se budou vzájemně doplňovat.

Dále proběhne zhodnocení funkčnosti aplikace a všech využívaných komponent. V případě nalezení chyb budou vytvořena doporučení pro další vývoj.

Data, získaná v rámci testování aplikace, se stanou cenným nástrojem pro budoucí evaluaci nových modulů a částí aplikace.

Klíčová slova: rozpoznávání řeči, hlasové dialogové systémy, evaluace, inteligentní domácnost, řídicí systém pro domácnost, hlasové ovládání domácnosti

Abstract

The goal of this thesis is to create an intelligent household system with voice control that should work as a demo application, which will use components created at the department of cybernetics and in the SpeechTech Company. This application will also offer the touch control which will serve as a supplement to the voice control.

Another goal is to evaluate the designed application and all the related components. In case of error detection, the thesis will present a proposal for adjustment.

The collected data will become a valuable tool for the future evaluation of the newest components.

Key words: speech recognition, voice dialogue systems, evaluation, intelligent household systems, control system for the household, voice-controlled household

Obsah

1 Úvod

Tato diplomová práce se zaměřuje na nejmodernější přístupy zpracování a syntézy řeči a autentifikaci řečníka. Kombinace těchto přístupů tvoří ucelený systém využitelný pro rozšíření a modifikaci softwaru založeného na dotykovém ovládání, ať už skrze webové stránky, tak skrze vlastní aplikace inteligentních zařízení.

V dnešní době existuje mnoho společností, které se zabývají automatizací domácností či firem. K samotnému ovládání jsou však využívány pouze TV, PC, tablety či chytré telefony. Nadstavbou těchto systémů může být hlasové ovládání, které lidem usnadní obsluhu komplexních systémů bez větší námahy a zároveň s co největší přirozeností. V minulých letech byl navržen systém na autentifikaci řečníka, který dokáže rozpoznat daného uživatele. Tímto systémem může být zabezpečeno uživatelské prostředí, jako například internetové bankovnictví, email, sociální sítě či samotný dům.

Práce kombinuje systém pro rozpoznání uživatele dle hlasového vstupu s komponenty pro rozpoznávání a syntézu řeči. Výsledkem bude aplikace, která bude hlasově zabezpečena a která zároveň poskytne jak hlasové, tak interaktivní dotykové rozhraní pro ovládání domácnosti. Vznikne tak efektivní spojení několika systémů, které dohromady vytvoří ukázkovou aplikaci pro interní potřeby katedry kybernetiky a firmy SpeechTech.

1.1 Cíle diplomové práce

Prvním cílem je prostudovat dostupné zdroje a seznámit se s problematikou tvorby hlasových dialogových systémů a inteligentní domácnosti.

Druhým cílem je získané znalosti aplikovat při návrhu vlastního systému, který by měl být vhodný pro využití v domácnosti či firmě.

Třetím cílem práce je realizovat a otestovat aplikaci dle vypracovaného návrhu s využitím softwarových komponent SpeechTech dostupných z projektu TIA (ASR, TTS). Nad rámec původního zadání bylo při realizaci a testování aplikace použito řešení katedry kybernetiky SpeechCloud a softwarová komponenta HESLO společnosti SpeechTech.

Kromě těchto cílů má práce ještě další cíl, a to získání dat, která mohou být využita pro pozdější evaluaci systému.

1.1.1 Vytvoření a testování aplikace

Aplikace musí být snadno ovladatelná a robustní – musí fungovat správně ve všech případech. Pro případ, že selže hlasová komunikace, bude možno aplikaci ovládat i pomocí dotyku. Jednoduchá přenositelnost a instalace aplikace jsou nutné pro komerční a předváděcí účely.

Po spuštění bude provedeno několik uživatelských testů a následně dotazníkové šetření. Na základě zpětné vazby od uživatelů budou vypracována doporučení pro další vývoj aplikace.

1.1.2 Otestování uživatelské autentifikace

Při konstrukci inteligentní domácnosti využijeme aplikaci, která porovnává promluvu uživatele s již dříve zaznamenanou promluvou a rozhoduje, zda se jedná o stejného uživatele či nikoli. Aplikace je na rozdíl od modulu pro rozpoznávání a syntézu řeči teprve ve vývoji a vznikla během minulého roku.

Testovat se budou délky promluv nutných k efektivnímu rozpoznání řečníka a dále míra účinnosti, s jakou systém řečníka správně identifikuje.

1.1.3 Otestování rozpoznávače

Testování modulu pro rozpoznávání řeči proběhne souběžně s testováním aplikace. Konkrétně se bude testovat jeho funkčnost v různých prostředích a s odlišnými mikrofony. Hlavním kritériem bude, zda se rozpoznává promluva, která byla řečena, či zda systém identifikoval promluvu odlišnou. Dále bude testována detekce řeči a ticha – zda nedochází ke špatné identifikaci například šumu na pozadí.

1.1.4 Získání testovacích dat

Data získaná z testování aplikace poslouží jako referenční množina, na základě které se budou porovnávat nově vytvořené verze jednotlivých komponentů systému. Na základě srovnání nové verze s původní bude možno říci, do jaké míry nová technologie vylepšuje systém.

1.2 Možnosti aplikace

Aplikace v základní verzi bude umět pouze elementární úkony, jako např. zhasnout či rozsvítit světlo, ovládat klimatizaci či žaluzie. Bezproblémové zvládnutí těchto úkonů je nezbytné pro její další vývoj. Pouze pokud bude aplikace stoprocentně funkční, mohou se k aplikaci přidávat další funkce. Pro budoucí vývoj by se aplikace dala rozšířit například o následující funkce:

- Administrační rozhraní
- Komplexní ovládání domácnosti – nastavování konkrétní teploty, intenzity světla, časování úkonů,… viz obrázek 2.2
- Připojení k internetu – zprávy, informace o počasí, sociální sítě, emaily či nakupování
- Agenda – správa kalendáře, informace o schůzkách
- Spojovatelka – vytvoření hovoru přes GSM / internet

1.3 Nedostatky

Z ekonomického hlediska není možné při vývoji využít nejmodernějších technologií, které trh nabízí. Proto jsou vývoj i samotné testování značně limitovány.

Dalším problémem je výpočetní náročnost. Aby aplikace zvládala co nejvíce úkonů, musí být umístěna na výkonném stroji, který dokáže zpracovat veškeré požadavky během setin vteřiny. Z tohoto důvodu bude využito tzv. Cloudového řešení. Na serveru, který je umístěn v budově NTIS, poběží aplikace SpeechCloud, která bude zpracovávat veškeré hlasové vstupy a výstupy.

Posledním nedostatkem je zabezpečení. Jelikož se aplikace připojuje na server, bude probíhat komunikace na vnější síti. Pro mnoho firem či domácností je takový způsob komunikace nevyhovující, jelikož kdokoli může odchytit citlivá data. Řešením může být umístění serveru do vnitřní sítě, která je odstřižena od zbytku internetu. Toto řešení se však setkává s nutností pořízení drahého technického vybavení.

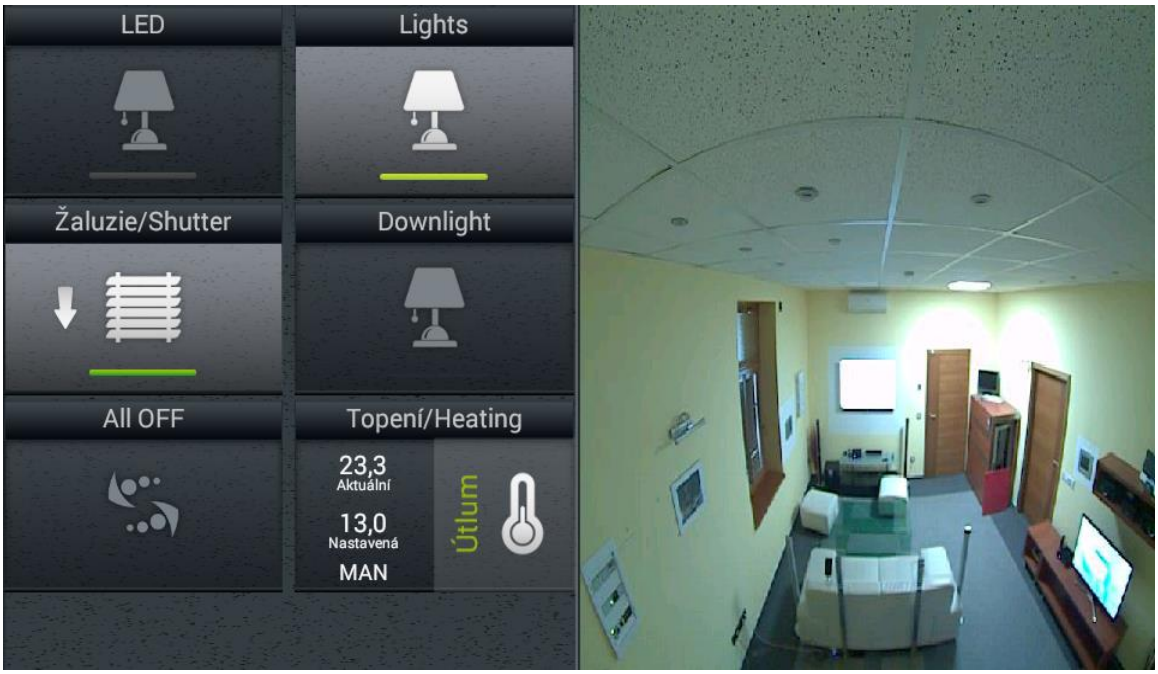
2 Inteligentní domácnost

2.1 Úvod

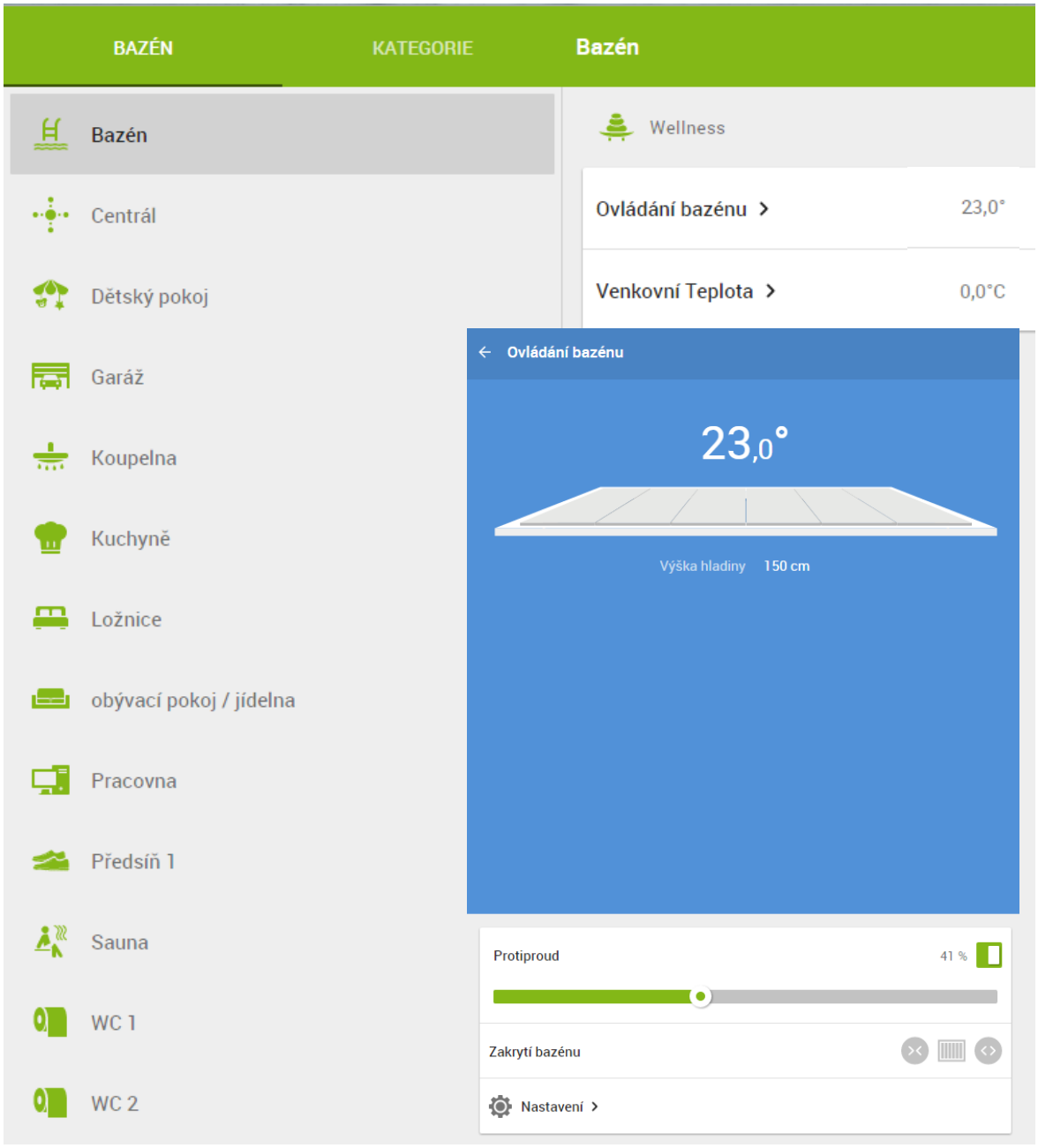
Inteligentní dům či inteligentní domácnost je fenomén moderní doby, se kterým se můžeme čím dál tím častěji setkat. Jedná se o funkční propojení zařízení v domácnosti a jejich vzdálené ovládání pomocí TV či PC a v poslední době stále častěji pomocí telefonu či tabletu. V ČR je hned několik firem, které se instalací zabývají. Některé z nich dokonce i staví novostavby se zabudovanými funkcemi dle přání majitele. Cena takovéto elektroinstalace se odvíjí od množství „inteligentních" zařízení v domácnosti, při komplexní instalaci může být i ve výši milionů korun [1].

2.2 Využití

Dnešní trh je velmi rozmanitý a nabízí tak uživatelům prakticky jakékoliv služby. Mezi základní systémy inteligentní domácnosti patří ovládání osvětlení, zabezpečení, vytápění a klimatizace. Dále je již na uživateli, jaké další funkcionality by chtěl ve svém domě mít. Na základě časté poptávky vytvořily firmy základní balíčky služeb a je na majiteli, kolik peněz je do takovéto instalace ochoten investovat. Můžeme se tak setkat například s domy, kde lze ovládat pračku, ledničku, rádio, televizi, rolety, ale i saunu a podobně. Veškeré ovládání domácnosti může být jak z pohodlí domova – interaktivní panel, tak i dálkově. Uživatel tedy ovládá (zapíná či vypíná) spotřebiče při cestě domů či z práce. Na obrázcích 2.1 a 2.2 jsou znázorněna uživatelská prostředí inteligentní domácnosti.



Obr. 2. 1 Ukázka aplikace na OS Android od firmy iNels



Obr. 2. 2 Ukázka uživatelského prostředí webové aplikace firmy Loxon

Samozřejmostí takového systému také je, že se o chod domácnosti stará sám program. Příklad fungování může být, pokud odejdeme z místnosti a nevypneme světlo či vodu – udělá to za nás. Nebo nás ráno probudí, poté roztáhne rolety, zapne topení a začne nám sám vařit vodu na kávu. Toto jsou pouhé dva příklady využití, reálné používání je samozřejmě mnohem rozmanitější. Dalším, již poněkud důležitějším příkladem,

je termoregulace domácnosti dle počtu členů, kteří se v ní nacházejí, a s tím související úspora energie.

Cíl pořízení je tedy zřejmý – usnadnění práce a zjednodušení života. Inteligentní domácnost se dá samozřejmě pořídit i jinak, než složitou elektroinstalací a zasahováním do konstrukce bytu. Stále více výrobců uvádí na trh inteligentní zařízení ať už pro zjednodušení každodenních povinností či třeba pro zabezpečení domácnosti. Příkladem může být nejnovější lednice od společnosti Samsung, která má fungovat jako mediální centrum domácnosti [2]. Lednice má velkou dotykovou obrazovku a uvnitř několik kamer, takže vidíme, co je v lednici, aniž bychom ji museli otevírat. K lednici je možno připojit několik dalších zařízení (např. kávovar či topinkovač), které se z ní dají ovládat. Při nákupu se můžeme podívat přímo do naší lednice a koupit si vše, co potřebujeme, bez strachu, že bychom na něco zapomněli. Lednice je dokonce schopna nám i napsat, co bychom měli koupit, či zboží sama objednat z internetu.

Příkladem zabezpečení může být inteligentní kamera Netamo Welcome [3]. Tato kamera má zabudovaný systém rozpoznávání obličeje a upozorní nás v případě, že do našeho domu někdo vstoupil. Rodiče se tak nemusí bát o své děti, jelikož je kamera pozná a pošle rodičům zprávu, že jsou děti doma. V případě, že do našeho domu vstoupí někdo cizí, dokáže nám ihned přehrát záznam a my tak vidíme, o koho se jedná a zda je třeba zavolat policii.

Důvody zavedení inteligentních spotřebičů či instalace elektrických rozvodů pro inteligentní domácnost můžeme shrnout několika body [4]:

- Automatizace a usnadnění všedního života
- Úspora času
- Zabezpečení
- Úspora energie a nákladů na bydlení
- Zdravější životní prostředí
- Jednoduchost
- Zvýšení komfortu
- Prevence a kontrola
- Pohoda a klid

2.3 Internet věcí

Dalším fenoménem poslední doby je tzv. internet věcí (anglicky Internet of Things). Jedná se o ucelený systém ovládání domácích spotřebičů. Na rozdíl od inteligentní domácnosti není třeba speciální elektro instalace – naprostá většina spotřebičů je vybavena Wi-Fi či Bluetooth a lze je tak připojit jednoduše do domácí sítě. Dle finančních možností si uživatelé postupně dokupují spotřebiče, které chtějí. Příkladem může být již výše zmíněná lednička či kamera. Do roku 2025 by k internetu mělo být připojeno 25 miliard přístrojů [5].

Speciální podkategorií přístrojů jsou tzv. wearable devices, tzn. nositelná zařízení. Velmi populární jsou inteligentní náramky, hodinky či brýle. Jejich aktuální nevýhodou je malá výdrž baterie a také fakt, že v naprosté většině vyžadují přítomnost inteligentního telefonu, se kterým komunikují.

Propojováním zařízení s internetem se uživatel vystavuje možnému riziku napadení zvenčí či ztráty citlivých dat. Využívání komerčních nástrojů dostupných zdarma je často podmíněno tím, že budou sbírána veškerá data, která společnost poté může využít pro další vývoj, ale také například pro cílený marketing.

2.4 Budoucnost

Dnešní svět spěje k automatizaci a je logické, že se tato oblast dostává čím dál tím častěji do našich domovů, kde trávíme nejvíce času. Nebude to dlouho trvat a lidé budou moci dostat snídani, připravené oblečení a auto je samo odveze do práce. Aktuálně nás nejvíce limituje technika – její cena a dostupnost. Pro efektivní komplexní zapojení a ovládání domácnosti potřebujeme výkonné stroje, které jsou cenově dostupné pro každého. Díky internetu a chytrým telefonům lze ovládat cokoli, co se nachází v našem domě. Přirozenou komunikací je však pro lidi řeč, a proto prvním krokem k vylepšení aktuálních systémů bude možnost hlasového ovládání. Právě problematikou hlasového ovládání inteligentní domácnosti se zabývá tato práce.

Jako příklad ovládání můžeme uvést stránky wit.ai, které se přímo zaměřují na zpracování řeči a pochopení významu. Tato služba otevírá vývojářům ze všech odvětví nové možnosti. Právě tím, že je služba zdarma, je její využití snadnou volbou. Bohužel podporuje pouze angličtinu, což je pro naše potřeby nevhodné.

3 Hlasový dialogový systém

3.1 Úvod

Hlasový dialogový systém je aplikace vytvořená expertním a statistickým způsobem. Expertní způsob se využívá pro porozumění a řízení dialogu, statistický způsob zase pro rozpoznávání a syntézu řeči.

Expertem je osoba, která vyniká v určité oblasti, má o ní dostatečné znalosti a ví, jak se v různých situacích rozhodovat. Expertní systém je založen na simulaci rozhodovací činnosti člověka, čímž lze nahradit lidskou práci. Takovýto systém na rozdíl od člověka je kdykoliv dostupný a přístupný všem, kteří ho potřebují. Díky tomu lze například nahradit firemní asistentku, jež je dostupná pouze vyššímu managementu, systémem, který může simultánně vykonávat několik úkolů pro jakéhokoliv zaměstnance. Takovýto systém umožňuje zaměstnavateli redukovat počet zaměstnanců a náklady s nimi spojené. V praxi se s podobným systémem můžeme běžně setkat např. při volání na bezplatné linky telefonních operátorů, kde při telefonní komunikaci nejdříve uživatel prochází skrze menu a až poté je přepojen reálnému zaměstnanci vyškolenému na konkrétní oblast.

Historie expertních systémů sahá do sedmdesátých let 20. století, kdy se vývoj umělé inteligence zaměřoval převážně na zpracování a reprezentaci znalostí získaných od odborníka. V současnosti se uplatňují spíše jako pomoc při rozhodování či pro rychlejší a jednodušší identifikaci. Příkladem může být znalostní systém v lékařském odvětví – po zadání určitých parametrů (např. věk, hmotnost, krevní tlak) lékař lehce zjistí, zda má pacient riziko nějaké choroby či nikoli [6].

Základem statistického přístupu je akustický procesor a lingvistický dekodér. Akustický procesor převede řečové kmity na posloupnosti vektorů příznaků (O), zatímco lingvistický dekodér překládá tyto řetězce na řetězce slov (W). Cílem je nalézt posloupnost slov $\widehat{W}$, která maximalizuje podmíněnou pravděpodobnost $P(W|O)$. Hledáme tedy nejpravděpodobnější posloupnost slov pro danou akustickou informaci (vektor příznaků).

$$\widehat{W} = arg \max_{W} P(W|O)$$

Při využití Bayesovy věty dostaneme

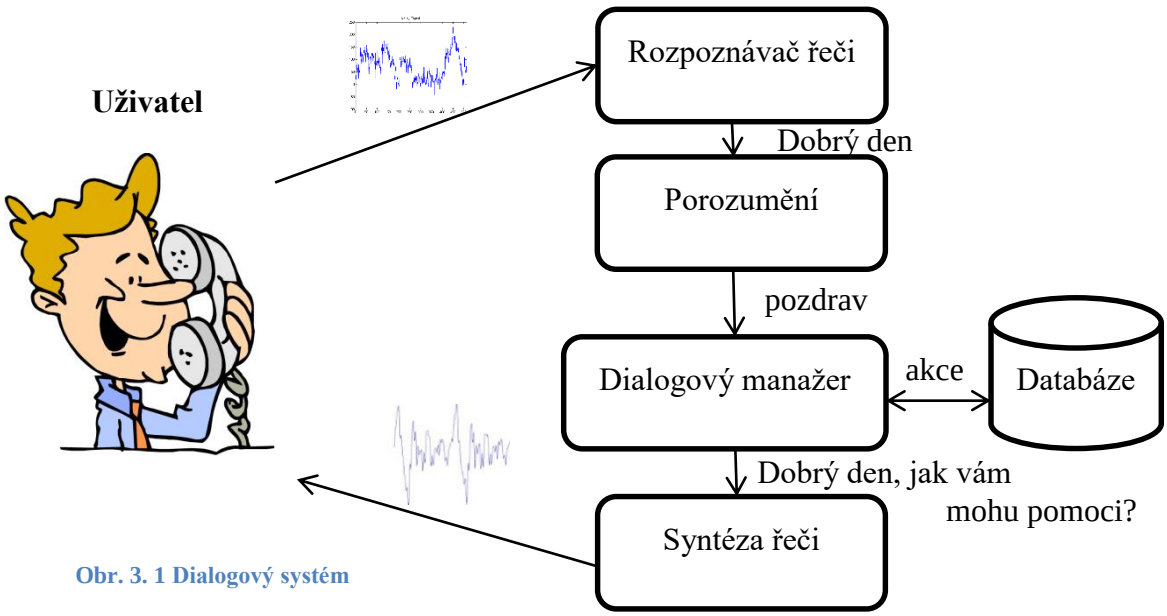
$$\widehat{W} = arg \max_{W} P(W|O) = arg \max_{W} \frac{P(W) \cdot P(O|W)}{P(O)}, kde$$

$P(O|W)$ udává pravděpodobnost, že při vyslovení posloupnosti slov bude vygenerován vektor příznaků. Grafické znázornění postupu můžeme vidět na obrázku 3.3.

Hlasové dialogové systémy umožňují člověku komunikovat s počítačem bez nutnosti přímého přístupu k PC či jinému konzolovému zařízení – veškerá komunikace tedy probíhá za pomoci hlasu. Aplikace může být spuštěna na lokálním PC či může fungovat prostřednictvím internetu na více zařízeních pomocí tzv. cloudu. V dnešní době se s těmito aplikacemi setkáváme v nejrůznějších odvětvích, jako je například řízení provozu, databázové systémy, dálkové ovládání a další. Princip je velmi jednoduchý – ulehčit člověku co nejvíce práce a dovolit komunikaci v pro něj co nejpřirozenější formě – tedy hlasem. Systémy jsou od nejjednodušších – ovládání skrze kontextové menu, až po složitější – ovládání plynulou řečí.

V současné době jsme limitováni technologií, která neumožňuje využití plného potenciálu dialogových aplikací. Místo tvorby komplexního systému, který by byl schopen komunikovat s uživatelem napříč různými odvětvími, jsme nuceni vytvářet systémy cílené jen na danou oblast (např. dialogový systém pro objednávku pizzy nebude schopen odpovědět na otázky ohledně počasí, apod.). Důvodem je, že daná konkrétní oblast je výpočetně extrémně náročná a databáze slov zabírá i několik gigabajtů místa. Prohledávání takto velkých souborů dat pak vyžaduje extrémně výkonné stroje. Můžeme jen doufat, že technologie brzy natolik pokročí, že budeme moci přijít domů z práce a pustit se do rozhovoru s naší pračkou [7].

Komunikaci zajišťuje několik modulů – řízení dialogu, rozpoznávání a syntéza řeči, porozumění mluveného jazyka a generování odezvy. Nejdůležitější součástí aplikace je dialogový manažer, který udává systému jaký modul použít a kdy. Jeho důležitost spočívá ve správném zvolení modulu, aby byla zaručena vždy patřičná reakce.



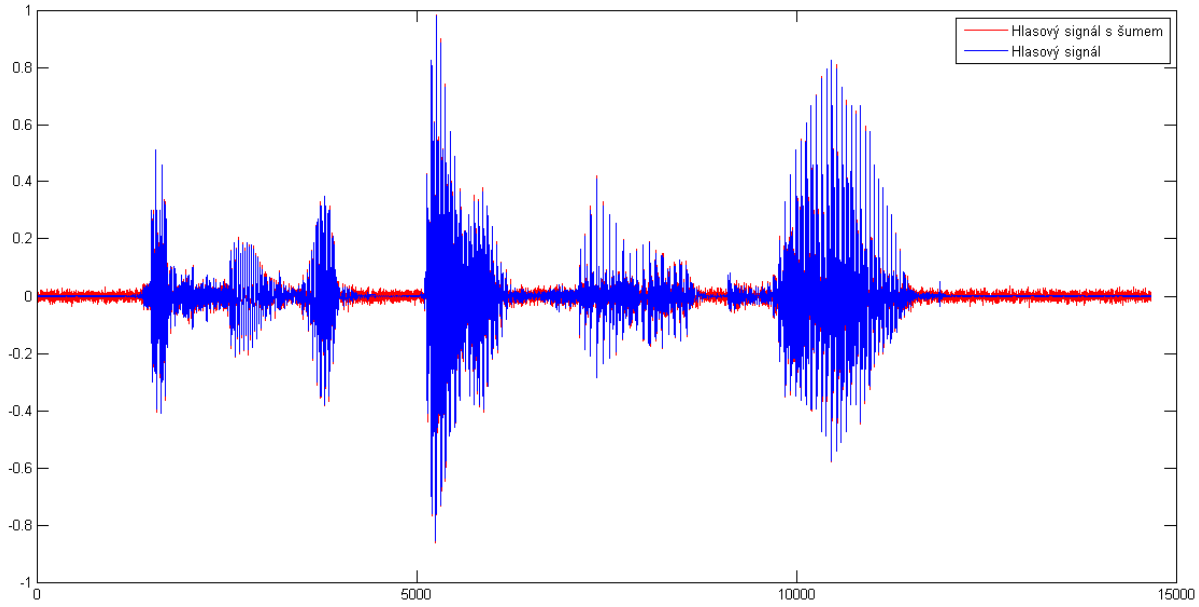
Obr. 3. 1 Dialogový systém

3.2 Rozpoznávání a porozumění řeči

3.2.1 Detekce řeči

Rozpoznání promluvy není pro člověka nijak obtížná záležitost. Náš mozek tento úkol zpracovává automaticky a většinou nemusíme této činnosti věnovat žádné úsilí. Pokud s někým hovoříme, ať už z očí do očí či pomocí komunikace na dálku (telefonní hovor, vysílačka), nebo třeba jen posloucháme (rádio, rozhlas, televize), vždy vnímáme hlas osoby, která je v předmětu našeho zájmu, a pozadí dokážeme potlačit. Pokud například televizní reportér informuje o aktuální situaci a na pozadí probíhá několik dalších rozhovorů či hraje hudba, dokážeme se zaměřit pouze na hlas reportéra a ostatní zvuky nevnímat (filtrovat).

ASR, neboli Automatic Speech Recognition (automatické rozpoznávání řeči) se zabývá právě tímto problémem. V hlasovém signálu se snaží najít frekvence, které patří danému řečníkovi a odlišit je od frekvencí pozadí (viz obrázek 3.2). Sami se s tímto postupem denně setkáváme, aniž bychom si to uvědomovali. Telefonní společnosti přenášející signál se snaží filtrací signálu vylepšit jeho kvalitu. Dokonce i moderní mobilní telefony dokáží potlačit šum okolí a vstupem do mikrofonu je tak samotný řečový signál. Po identifikaci kýženého signálu se s hlasovou nahrávkou může dále pracovat.

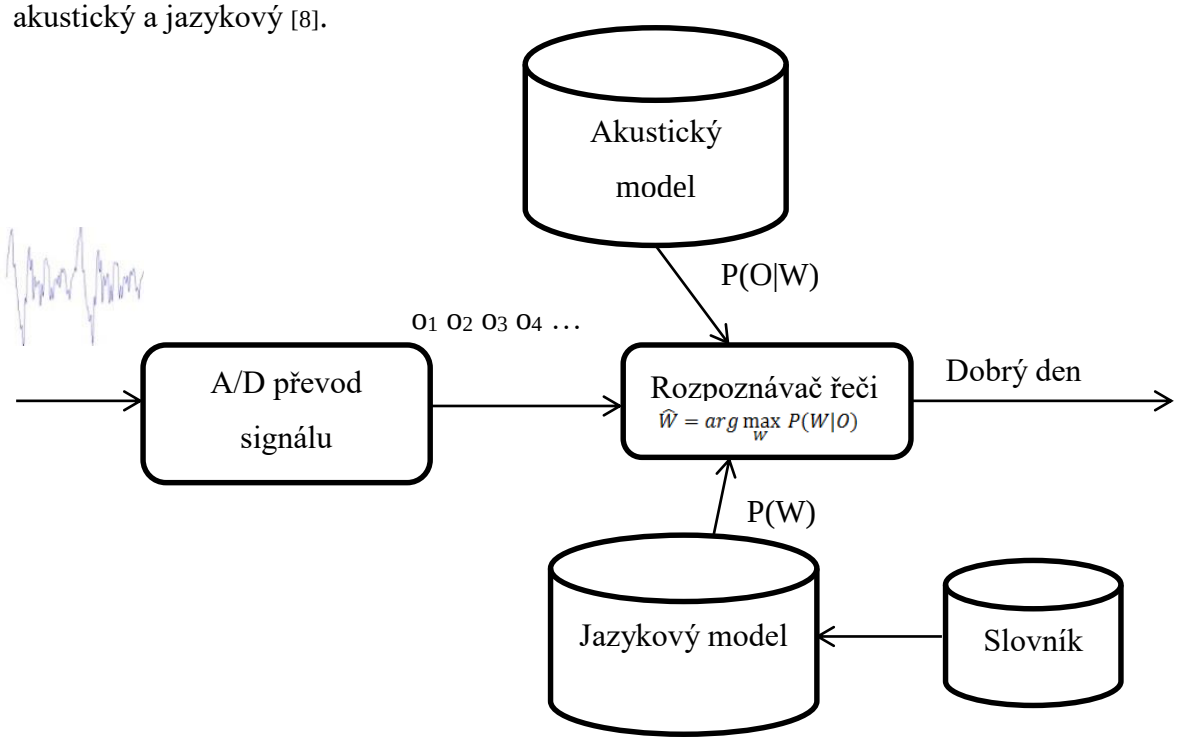


Obr. 3. 2 Signál slova „osm set padesát dva“ s šumem a bez

3.2.2 Rozpoznávání řeči

Řeč je naší přirozenou součástí. Již od dětských let se její pomocí učíme pojmenovávat své okolí a komunikovat s ostatními. Pro vzájemnou komunikaci mezi lidmi je nutné chápat význam daných slov. Abychom toho dosáhli, musíme vědět, v jakém jazyce komunikujeme, co dané slovo (alespoň přibližně) znamená, zda existuje nějaká podobnost s něčím, co již známe. Při komunikaci s počítačem musíme využít těchto znalostí. Již od vytvoření prvních počítačů se lidé snažili komunikovat s počítačem prostřednictvím řeči. Ta probíhala pomocí pouštění různých nahrávek uložených na páskách, a to i přes značné technické omezení dané doby.

V dnešní době jsou díky výkonným strojům počítače schopny rozpoznat miliony slov, identifikovat řečníka a dokonce se i přizpůsobit slovům, se kterými se dosud nesetkaly, pomocí nejrůznějších otázek či kontextu. K rozpoznávání řeči se používají dva modely – akustický a jazykový [8].



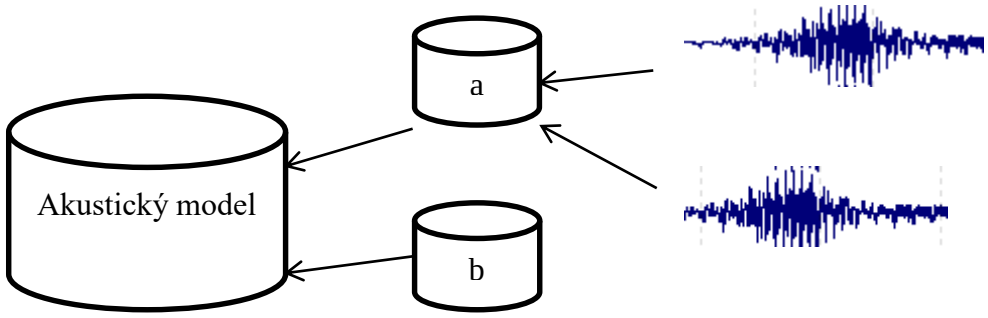
Obr. 3. 3 Model rozpoznávání řeči

3.2.2.1 Akustický model

Základní zvuková jednotka každého jazyka je foném. V českém jazyce jich máme přibližně 39. Výhodou češtiny je, že většina fonémů je totožná s písmeny abecedy na rozdíl například od angličtiny, která má odlišnou psanou formu od výslovnosti. Akustický model se skládá z řetězců příznaků jednotlivých fonémů. Základní podoba je znázorněna na obrázku 3.4. Čím více vzorů od daného fonému je k dispozici, tím přesněji bude

rozpoznávač fungovat. V praxi se používají tisíce různých fonémů pořízených od stovek řečníků, čímž se docílí vysoké přesnosti a robustnosti systému. Hlavním důvodem je, že kontext daného fonému ovlivňuje jeho výslednou podobu.

Při pořízení dostatečného množství vzorků musíme provést anotaci získaných dat, s jejichž pomocí uděláme statistický model, který je tvořen maticí příznaků daného fonému.



Obr. 3. 4 Příklad akustického modelu

3.2.2.2 Jazykový model

Každý jazyk je tvořen gramatikou. Vzhledem k omezeným možnostem (výpočetní nároky, robustnost systému) vznikají gramatiky pouze pro dané oblasti zájmu. S využitím statistického přístupu jazykový model generuje slova či řetězce slov na základě jejich nejpravděpodobnějšího výskytu v trénovacích datech.

3.2.2.3 Slovník

Nezbytnou součástí modelu je slovník, který přiřazuje fonémům správnou výslovnost s ohledem na pravidla užitého jazyka.

3.2.3 Porozumění řeči

Rozpoznání mluvené řeči je pouze prvním krokem v řešení našeho problému. Z rozpoznané promluvy dále musíme zjistit, jaký má promluva význam. Až při pochopení toho, co se po nás žádá, jsme schopni reagovat patřičnou odezvou.

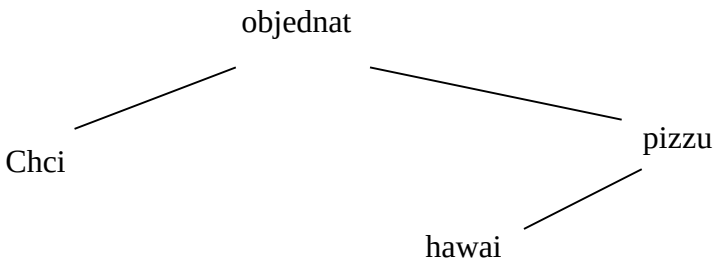
Při rozpoznávání řeči jsou nejvíce problematické cizí a hovorové výrazy a spontánní řeč. Velmi častým nepříznivým jevem jsou všelijaká přeřeknutí, nedokončená slova, restarty, šum na pozadí nebo rušení zvukového signálu. Všechny tyto prvky ovlivňují kvalitu rozpoznávané řeči a snižují pravděpodobnost správného pochopení významu. Příkladem může být objednávka pizzy:

Dobrej, já bysem chtěl to, tu vaší velkou pizzu, sal-šunkovou nebo ne víte co radši mi dejte hawai.

Úlohu porozumění řeči můžeme rozdělit na 2 části: syntaktickou analýzu a reprezentaci znalostí.

3.2.3.1 Syntaktická analýza

V angličtině se s ní můžeme setkat pod pojmem parser. Syntaktickou analýzu všichni dobře známe ze základních škol pod pojmem větný rozbor – hledáme v dané větě podmět, předmět, přísudek, atd. Při analýze data rozdělíme do přehledné datové struktury – stromu, která zachovává hierarchii a je vhodná k pozdějšímu pracování s daty.



Obr. 3. 5 Syntaktická analýza věty „Chtěl bych objednat pizzu hawai“

3.2.3.2 Reprezentace znalostí

Jedná se o soubor postupů a pravidel používaných pro zachycení znalostí a práci s nimi. V oblasti psychologie proběhlo několik výzkumů, na základě kterých byly vypracovány teorie o tom, jak člověk řeší, zpracovává a uchovává znalosti. Právě tyto znalosti byly využity pro vývoj počítačových metod.

Znalosti můžeme rozdělit na několik druhů

- Procedurální – zachycují, jak postupovat při provádění daných akcí (usuzování)
- Deklarativní – zachycují, co platí (statické pravdy)
- Meta-znalosti – znalosti o jiných typech znalostí a jak je využít
- Heuristické znalosti – znalosti odvozené selským rozumem
- Strukturální znalosti – soubory pravidel, vztahy mezi pojmy a objekty

V praxi se využívá několik metod, jak znalosti zakódovat: pravidla, rámce, O-A-V triplet (objekt, atribut, hodnota), logika a sémantické sítě. Každá tato metoda má své výhody a nevýhody [5].

V této práci bude popsán systém, který využívá určitá pravidla a rámce, tedy strukturální znalosti.

3.3 Syntéza řeči

Syntéza řeči je opakem rozpoznávání. Namísto zpracovávání analogového signálu a dekódování posloupnosti slov kódujeme posloupnost a převádíme ji do analogového signálu. K syntéze řeči slouží tzv. syntetizátory, které vytvářejí počítačově generovanou řeč. Docílit toho můžeme dvěma způsoby – spojováním již nahraných slov či simulováním charakteristik hlasového traktu.

V praxi se setkáme spíše s prvním způsobem. Hodiny nahrané řeči se zpracují a rozdělí na jednotlivé fonémy, které se pak spojují do slov či vět – viz akustický model. Pro zajištění kvality promluvy (nalezení správného fonému) se využívají difony či trifony. Jedná se o spojení dvou či tří fonémů za sebou, tím se docílí správná návaznost a řeč zní přirozeněji. Při dostatečném množství nahrávek jsme schopni počítačem generovat promluvu, která je k nerozeznání od člověka.

3.4 Konstrukce dialogu

3.4.1 Gramatika

Základním elementem dialogového systému je gramatika. Gramatika zahrnuje důležitá slova, která slouží k interakci s uživatelem – jedná se o soubor dat z dané oblasti, která omezuje množinu promluv. Jak již bylo zmíněno v úvodu, právě gramatika je důvodem, proč aktuálně nelze sestrojit systém, který by dokázal více věcí (uměl by podat informace o počasí a zároveň i diagnostikovat medicínský problém) vzhledem k paměťovým a výpočetním nárokům. Systémy jsou proto tvořeny tak, aby uměly reagovat na nejrůznější slova a promluvy z dané oblasti a dokázaly patřičně reagovat. Při vytváření je potřeba se zamyslet nad potřebami uživatele a promluvami, které by se mohly objevit. Každé rozpoznané slovo promluvy se prohledává v naší databázi a zjišťujeme, zda dané slovo něco znamená či nikoli. Nejčastěji využívaná je bezkontextová gramatika, ve které, jak již název napovídá, nezáleží na kontextu daného slova. Bezkontextová gramatika využívá přepisovacího tvaru $A \rightarrow \beta$, kde A je neterminál a β je řetězec terminálů či neterminálů. Neterminál můžeme přepsat na terminál bez ohledu na kontext [9].

Množina všech řetězců, které můžeme z gramatiky vygenerovat, je jazykem gramatiky. Příklad gramatiky je znázorněn na obrázku 3.6.

Formálně gramatiku definujeme jako

$$G = (N, \Sigma, P, S)$$

- N – konečná množina neterminálních symbolů (značíme velkými písmeny)
- Σ – konečná množina terminálních symbolů (značíme malými písmeny)
- P – konečná množina odvozovacích pravidel
- $S \in N$ – počáteční symbol

3.4.1.1 Příklad generování jazyka gramatikou

$G = (N, \Sigma, P, S)$

N = {S, A, B, C, D}

Σ = {Chtěl bych, Chci, objednat, koupit, pizzu, malou, velkou}

S = S

P:

S → A B
A → Chtěl bych
A → Chci
B → objednat C
B → koupit D
C → pizzu
D → velkou C
D → malou C

Z této gramatiky můžeme vygenerovat například následující věty

- *Chtěl bych objednat pizzu*
- *Chci koupit velkou pizzu*
- *Chci objednat pizzu*
- *Chtěl bych koupit malou pizzu*

3.4.2 Návrh gramatiky

K návrhu se využívají simulované promluvy dle konkrétních scénářů zaměřených na patřičnou oblast. Vzhledem k tomu, že ne každý využívá stejný dialekt, je třeba zajistit vzorky promluv od nejrůznějších uživatelů. Vhodné je, kromě konkrétních a striktně daných promluv, které jsou vhodné pro testování systému – zda systém správně reaguje na různé řečníky, nechat uživateli prostor pro formulaci otázek a požadavků. Z těchto požadavků se může rozšířit gramatika systému a zvýšit tak celková robustnost.

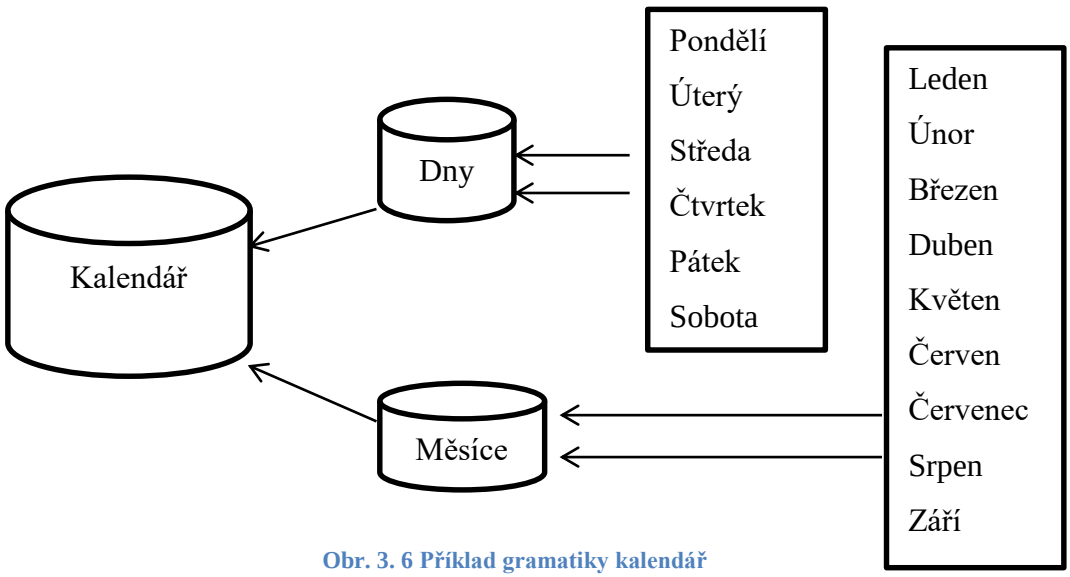
Vhodně navržený scénář má tedy značný vliv na finální podobu gramatiky a tedy i funkčnost systému. Každý člověk je jedinečný a každý člověk používá svůj charakteristický styl komunikace, s čímž je třeba počítat. Pokud z nějakého důvodu nelze pořídit dostatečný počet nahrávek, je úkolem vývojářů doplnit gramatiku o konkrétní slova.

3.4.3 Zpracování nahrávek

Pořízené nahrávky se přesně písmeno od písmene přepíší do textu a označí speciálními značkami, které blíže specifikují, o jaké slovo jde – přeřeknutí, dotaz na čas, místo, aktuální stav, atd. Na jejich základě dojde ke generování gramatiky.

3.4.4 Testování

Před vytvořením finální verze vždy předchází fáze testování, při kterém se systému pokládají nejrůznější otázky a požadavky, na základě kterých se pak vyhodnotí, zda by gramatika měla obsahovat více či méně slov.



Obr. 3. 6 Příklad gramatiky kalendář

3.4.5 ESGF gramatika

ESGF udává pravidla popisu gramatiky. Jedná se o bezkontextovou gramatiku, nezáleží nám tedy na kontextu, ale pouze na specifické promluvě. V těchto gramatikách nalezneme většinou slova, slovní spojení či části vět. V hlavičce gramatiky je uveden seznam pravidel, gramatik a jejich jména. Tělo obsahuje pravidla [10].

3.4.5.1 Příklad gramatiky

```
// hlavička
#ESGF V1.0 UTF-8 cs;      // verze, znaková sada a jazyk gramatiky
grammar pizza;            // název gramatiky

// tělo
public <pizza> =  (<velikost> | <typ>);
// pravidlo pizza rozšiřují pravidla velikost a typ
<velikost> = ( malá | střední | velká {32cm}) {velikost};
<typ> = ( hawai | salámová | šunková) {typ};
```

V praxi se využívá tzv. tagů – značek, které se předávají syntaktické analýze po rozpoznání dané promluvy. Systému se tak například namísto slova velká předá informace 32cm. Tagy se značí dvojicí závorek {}. Svislá čára | odděluje možné promluvy či výslovnosti daného slova a poskytuje tak systému různé alternativy.

3.5 Vedení dialogu – dialogový manažer

V každém časovém okamžiku se dialog nachází v konkrétním stavu. Nezáleží na tom, zda se jedná o mluvenou nebo psanou formu či zda komunikují lidé nebo stroje. Stav dialogu determinuje vždy pevně danou množinu akcí, které mohou nastat. S každou novou informací se navyšuje naše povědomí o dané úloze a jsme schopni adekvátně reagovat, v případě chybějící informace zase navrhnout vhodnou otázku, kterou si chybějící informace doplníme. Právě tímto problémem se zabývá dialogový manažer.

Stav se postupem času vyvíjí, od iniciálního až po koncový. Dialog je veden s určitým cílem, na základě kterého je nutno volit správné otázky – např. informace o klientově velikosti bot je při objednávání jídla vskutku nepodstatná. Cílem navržení dialogu je tedy uspokojit uživatelovy požadavky v co nejjednodušší a zároveň nejpřirozenější formě. Nevhodné je také časté opakování již známých informací – takovéto promluvy jsou zbytečně dlouhé a neefektivní.

Jednou z hlavních funkcí je volání přidružených aplikací na základě daného stavu – tím se zamezí zbytečnému zatěžování a zpomalení systému.

3.5.1 Rámcový systém

Stav dialogu si můžeme představit jako tabulku, kterou chceme zaplnit konkrétními informacemi. Při zjištění všech potřebných informací můžeme provést vhodnou akci.

3.5.1.1 Příklad systému pro objednávku pizzy – vhodně navržený dialog

Systém: *Dobrý den, pizzérie FAV, co si přejete?*

Uživatel: *Dobrý den, chtěl bych objednat velkou hawai.*

Stav dialogu

Jméno:	
Adresa:	
Telefonní číslo:	
Druh Pizzy:	hawai
Velikost:	32cm

Systém: *Jaké je prosím vaše jméno a adresa? Telefonní číslo můžeme použít stejné, z kterého voláte?*

Uživatel: *Jakub Nedvěd, Univerzitní 22, ano je stejné.*

Stav dialogu

Jméno:	Jakub Nedvěd
Adresa:	Univerzitní 22
Telefonní číslo:	777 777 777
Druh Pizzy:	hawai
Velikost:	32cm

Systém: *Děkujeme za vaši objednávku. Velká hawai na jméno Jakub Nedvěd, adresa Univerzitní 22, bude doručena přibližně za 40 minut.*

Uživatel: *Děkuji.*

Z ukázky je patrné, že otázky by měly následovat v logickém sledu a potvrzování dat od uživatele by mělo být podáno co nejpřirozenější formou. Pokud by systém špatně rozpoznal adresu, mohl by uživatel v posledním kroku informace změnit.

3.5.1.2 Příklad systému pro objednávku pizzy – nevhodně navržený dialog

Systém: *Dobrý den, pizzérie FAV, co si přejete?*

Uživatel: *Dobrý den, chtěl bych objednat velkou hawai.*

Stav dialogu

Jméno:	
Adresa:	
Telefonní číslo:	
Druh Pizzy:	hawai
Velikost:	32cm

Systém: *Objednávka velká hawai. Zadejte prosím vaše jméno a adresu.*

Uživatel: *Jakub Nedvěd, Univerzitní 22.*

Stav dialogu

Jméno:	Jakub Nedvěd
Adresa:	Univerzitní 22
Telefonní číslo:	
Druh Pizzy:	hawai
Velikost:	32cm

Systém: *Objednávka velká hawai na jméno Jakub Nedvěd, adresa Univerzitní 22. Telefonní číslo je stejné jako to, z kterého voláte?*

Uživatel: *ano je*

Stav dialogu

Jméno:	Jakub Nedvěd
Adresa:	Univerzitní 22
Telefonní číslo:	777 777 7777
Druh Pizzy:	hawai
Velikost:	32cm

Systém: *Děkujeme za vaši objednávku. Velká hawai na jméno Jakub Nedvěd, adresa Univerzitní 22, bude doručena přibližně za 40 minut.*

Uživatel: *Děkuji.*

V tomto příkladu je uživateli dvakrát zopakována informace, která v dialogu mohla být již vynechána. Namísto toho zbytečně prodlužuje promluvu a zaměstnává tak uživatele, který musí věnovat rozmluvě více pozornosti. Za úplně nevhodný dialog bychom mohli považovat neustále se opakující otázku či zjišťování jednotlivých informací postupně – jméno, adresa, telefon, druh pizzy, velikost.

4 Webové aplikace

Pod pojmem webové aplikace se rozumí služby, které jsou přístupné skrze internet. Software či aplikace je zprovozněná na straně serveru a klient se ke službě prostřednictvím sítě připojuje bez nutnosti mít cokoli nainstalováno (samozřejmě kromě internetového prohlížeče). Uživatelé tak mohou bez problémů obsluhovat servery, emaily, internetové obchody, ale například i výpočetní a řídicí systémy. Výhod je hned několik. Asi největší výhodou je, že pokud systém projde aktualizací a nainstalujeme nové služby, tak jakmile se klient připojí, ihned pocítí nové změny. Další výhodou je přenositelnost. Nezáleží na tom, jaký operační systém uživatel užívá, zda je to Windows, Linux či Android, na všech zařízeních aplikace funguje stejně. Nevýhodou je, že pokud je přerušen tok dat ze serveru či server selže, k aplikaci se nelze připojit [11].

Webové aplikace můžeme dělit do tří vrstev

- Prezentační – internetový prohlížeč
- Aplikační (logická) – zdrojové kódy aplikace
- Datová – databáze pro ukládání dat

Nejčastěji využívaným skriptovacím jazykem je PHP. Aplikační kód se vykonává na straně serveru a klientovi se pouze zobrazují výsledky. Při každém požadavku je tedy nutno odesílat data na server. Opakem je jazyk JavaScript, který se vykonává na straně klienta (prezentační vrstva), veškerá komunikace je tedy mnohem rychlejší a interakce uživatelsky přívětivější, jelikož se dané úkony odehrávají v reálném čase bez nutnosti odesílat data a čekat na výsledky. Kombinací těchto technologií je AJAX[1], který využívá interaktivní prvky z JavaScriptu a pomocí požadavků na pozadí získává data ze serveru, díky čemuž uživatel nemusí odesílat požadavek. Příkladem může být nápověda při psaní do vyhledavače – vyhledavač našeptává slova či slovní spojení.

Pro ukládání dat na straně serveru se nejčastěji využívají SQL databáze, ke kterým se přistupuje právě pomocí PHP. Na straně klienta lze využít díky HTML5 tzv. webové uložiště (někdy též lokální či DOM uložiště). To s sebou nese řadu výhod, například to, že uživatel při znovu otevření stránky může pokračovat ze stavu, v jakém ji opouštěl [12].

[1] AJAX je asynchronní požadavek ze strany klienta na server skrze http protokol

4.1 Komunikační porty

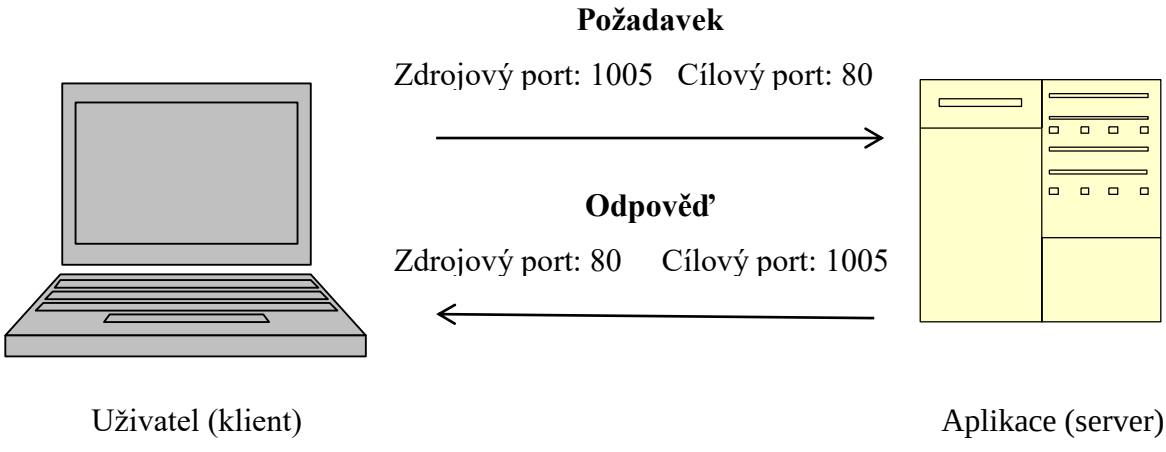
Síťové porty umožňují komunikaci mezi klientem a serverem. Číslo portu může být v rozmezí 0 až 65 535, přičemž komunikace může probíhat pomocí TCP či UDP. TCP znamená Transmission Control Protocol, což můžeme volně do češtiny přeložit jako kontrolovaný přenos. Při komunikaci skrze TCP se nejdříve naváže spojení a klient i server si spolu vyměňují důležité informace, například zda byl přenos dokončen a zda je vše v pořádku. Druhým způsobem komunikace je UDP neboli User Datagram Protokol. V češtině ho označujeme jako nespojový přenos dat. Při tomto způsobu přenosu dat již nedochází k potvrzování a může se stát, že se datový paket cestou někde ztratí nebo že do cíle dorazí v nesprávném pořadí [13].

Již z tohoto krátkého popisu je nejspíše zřejmé, že naše komunikace bude probíhat pomocí TCP, které nám zajistí spolehlivost přenosu a zachování pořadí zasílaných datových paketů. Mezi nejznámější protokoly a porty patří HTTP (80), HTTPS (443), SMTP (25), FTP (20,21), POP3 (110) či IMAP (143).

Porty se dělí do tří skupin:

- Dobře známé porty (well known ports) 0 až 1 023
- Registrované porty 1 024 až 49 151
- Dynamické a soukromé porty 49 152 až 65 535

Server rozlišuje klienta podle čísla portu, které se předá při navázání spojení. Díky tomuto způsobu komunikace nedojde k záměně odesílaných dat na jiného klienta. Na obrázku 4.1 je znázorněna zjednodušená komunikace mezi klientem a serverem při žádosti o zobrazení webové stránky.



Obr. 4. 1 Komunikace mezi klientem a serverem

4.2 HTML

HTML neboli Hyper Text Markup Language je programovací jazyk pro popis webových stránek. První verze (verze 0.9) webu byla představena v roce 1990 z CERN a jednalo se o pouhý text. V roce 1995 byla vydána verze 2.0, která s sebou přinesla interaktivní formuláře a grafiku. Poté vývoj pokračoval až do roku 1999, kdy byla vydána verze 4, která byla standardem téměř až do roku 2014, kdy vznikla verze 5. Dnes je HTML 5 standard, který podporují všechny prohlížeče a starší webové stránky se postupně předělávají do nové verze, která přináší spousty výhod.

HTML5 je založeno na několika technologiích dohromady – HTML, CSS, DOM a JavaScript. Díky novým vlastnostem již také není zapotřebí využívat externích modulů, jako např. flash. Spolu s tím také přináší zjednodušení a optimalizaci např. pomocí nových elementů pro strukturu stránky. Dříve se mohly využívat nejrůznější elementy a jejich umístění bylo libovolné. Nyní je přesně specifikováno, jaké elementy se v jaké části dokumentu mají použít, což usnadňuje orientaci ve stránce. Další velkou výhodou je ukládání dat při offline režimu na neomezenou dobu do lokálního uložiště, což je náhrada za nevhodné cookies.

4.2.1 Elementy a atributy

HTML dokument je popsán tzv. elementy (tagy), které jsou základními komponentami. Značí se špičatými závorkami a většinou jsou párové (otevírací a koncový), tzn. mezi jednotlivými elementy je vnořen určitý obsah. Koncový element je značen stejně jako počáteční, jen je na konci před závorkou uvedeno lomítko. Kromě obsahu můžeme specifikovat určité atributy, které určují konkrétní vlastnosti elementu. Tyto vlastnosti zapisujeme do prvního elementu ve tvaru atribut = hodnota [14].

Příklad: Odkaz na stránky katedry kybernetiky

```
<a href="http://kky.zcu.cz" target="_blank">Katedra Kybernetiky</a>
```

Tato syntaxe říká, že po kliknutí na text Katedra Kybernetiky se otevřou webové stránky http://kky.zcu.cz v novém okně prohlížeče, což bylo docíleno atributem target.

4.2.2 Členění stránky

HTML stránka začíná párovým elementem <html> a je rozdělena na dvě základní části – hlavička a tělo. Hlavička slouží hlavně pro definici a nastavení. Nalezneme zde prvky jako název a jazyk stránky, klíčová slova pro vyhledavače, kaskádové styly či JavaScript. Hlavičku definuje párový element <head> a pro klienta je neviditelná. Tělo dokumentu

slouží pro samotný obsah. HTML5 definuje několik strukturních elementů, které se užívají pro snadnější orientaci stránky a dbají na logické uspořádání obsahu. Jsou definovány elementy pro navigaci, obsah, postranní panel či patičku stránky. Tělo stránky se značí párovým elementem <body>.

Příklad: Odkaz na stránky katedry kybernetiky umístěný na stránkách s názvem Test

```
<html>
  <head>
    <title> Test</title>
  </head>
  <body>
     <a href="http://kky.zcu.cz" target="_blank">
          Katedra Kybernetiky</a>
   </body>
</html>
```

Hlavička dokumentu

tělo

4.3 CSS

Kaskádové styly (v angličtině Cascading Style Sheets) popisují vzhled HTML dokumentu. V HTML5 standardu je zakomponovaná verze CSS3. Pomocí atributu style, definují výslednou podobu jednotlivých elementů v HTML dokumentu. Styl formátování může být popsán několika způsoby. Popis v hlavičce stránky či v samostatném dokumentu se užívá pro formátování celého dokumentu, většinou se v něm vyskytuje několik stejných elementů, kterým chceme přiřadit dané vlastnosti. K těmto elementům pak přistupujeme pomocí jejich ID či tříd. Další možností je popsat vlastnosti přímo u elementu atributem style. V praxi se spíše využívá ukládání do externího souboru a volání do daných stránek, jelikož se tak může popsat pomocí jednoho dokumentu vzhled pro několik elementů či stránek najednou. Velikou výhodou je také to, že můžeme využít jiné styly pro odlišná zařízení. Tímto způsobem můžeme například definovat vzhled stránky pro desktopové zařízení (PC, notebook) a pro přenosné zařízení (telefon, tablet).

Elementy můžeme mezi sebou rozlišovat pomocí atributu name, ID či class a nastavovat tak unikátní podobu jednotlivým částem stránky. Spolu s JavaScriptem tvoří CSS dynamickou (proměnnou) část stránek.

Příklad: Odkaz na stránky bude zbarven červeně.

```
<a href="http://kky.zcu.cz" target="_blank"
     style="color:red;" >Katedra Kybernetiky</a>
```

4.4 JavaScript

Poslední nezbytnou součástí moderního webu je JavaScript. Tento programovací jazyk i přes svůj název nemá nic společného s jazykem Java, i když jsou si podobné. Pomocí JavaScriptu lze měnit stránky a to jak obsah (HTML), tak styl (CSS). Jelikož je součástí přímo HTML stránky, příkazy se vykonávají na straně klienta – v prohlížeči. To přináší spousty výhod a zjednodušení. Například validace formuláře tak může probíhat ihned při vložení dat a nemusí se čekat na odpověď serveru, zda jsou data vložena správně. Stejně jako kaskádové styly, script může tvořit součást kódu nebo může být uložen v externím souboru a importován do stránky.

Pomocí JavaScriptu můžeme naprogramovat různá chování, nejčastějším využitím je změna obsahu při kliknutí či najetí myši na daný objekt. Přístup k jednotlivým elementům je zprostředkován pomocí DOM, nejčastěji přes funkci getElementById či getElementByName. Volání funkce probíhá, jak již bylo zmíněno, pomocí určitých událostí (např. při kliknutí). Měnit můžeme jak samotný obsah, tak i vzhled (styl). Kód můžeme vložit do párových značek <script> do hlavičky nebo těla dokumentu či zapsat přímo k danému elementu.

Příklad: Změna barvy odkazu z červené na zelenou

```
<a href="http://kky.zcu.cz" target="_blank" style="color:red;"
    id="kky">Katedra  Kybernetiky</a>
<a onClick="document.getElementById('kky').style.color='green';
    href="#" "> Změnit  barvu</a>
```

4.4.1 JQuery

JQuery je JavaScriptová knihovna, která značně zjednodušuje programování v JavaScriptu. Díky jednoduchému kódu a snadnému volání funkcí šetří práci a čas při vývoji. V knihovně jsou již zahrnuty problémy, jako například modifikace elementů či práce s obsahem stránky až po jejím celém načtení, což je zásadní pro správné fungování kódu. Pokud bychom totiž chtěli měnit části stránky, které ještě nebyly zcela načteny, nebyla by zachována správná funkčnost. Volání probíhá pomocí značky $, která tak odlišuje kód v knihovně JQuery od zbylého kódu v JavaScriptu.

Příklad: Změna barvy odkazu pomocí JQuery

```
 <a href="http://kky.zcu.cz" target="_blank" style="color:red;"
id="kky">Katedra   Kybernetiky</a>
 <a href="#" onClick="$('#kky').css('color', 'green');">
     Změnit barvu</a>
```

Na příkladu je patrné, že oproti syntaxi v JavaScriptu je použití JQuery mnohem jednodušší a intuitivnější.

4.4.1.1 JSON

JSON je součástí JQuery, ale také dalších programovacích jazyků. Dokáže převést různé datové typy do textového řetězce a potom textový řetězec zpět na datový typ. Ve webových aplikacích se využívá pro zasílání objektů skrze WebSockets, jelikož se nic jiného než text přenášet nedá. Tato funkce velmi urychlí a usnadní komunikaci a výměnu dat se serverem.

Data = { klic: "hodnota" }  "{"klic":"hodnota"}"

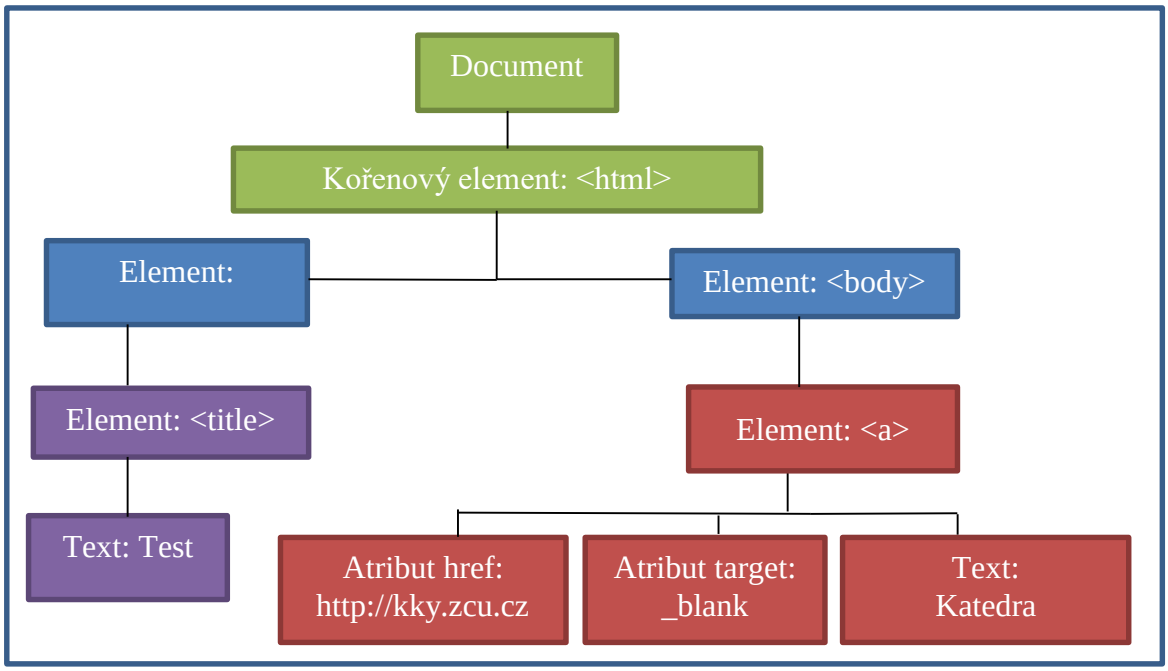
4.5 DOM

DOM neboli Document Object Model (česky též objektový model dokumentu) je aplikační rozhraní pro přístup k HTML dokumentům a objektům v nich. Definuje standard přístupu ke stránce vydaný W3C[2] konsorciem. Po načtení webové stránky je DOM sestaven a uložen do paměti prohlížeče. Můžeme tak jednoduše manipulovat s jednotlivými částmi stránky, jako například jejími elementy, atributy či se samotným textem. Kromě samotné webové stránky můžeme manipulovat i se vzhledem pomocí kaskádových stylů. Nejenže můžeme upravovat stávající, ale lze například i přidávat nové či mazat staré elementy. Pro práci s objekty se využívá JavaScript.

DOM udržuje stromovou strukturu, díky čemuž můžeme jednoduše přistupovat k jednotlivým elementům, ale třeba i k jejich nadřazeným či podřazeným jedincům. K jednotlivým objektům přistupujeme pomocí pole [15].

Příklad: Na další stránce je znázorněn na obrázku 4.2. DOM pro odkaz na katedru kybernetiky otevíraný v novém okně, který je umístěn v těle dokumentu. Názvem stránky, na které se odkaz vyskytuje, je Test (viz příklad v části 4.2 HTML).

[2] W3C, neboli World Wide Web Consorcium je mezinárodní sdružení, které definuje standardy pro World Wide Web



Obr. 4. 2 DOM

4.6 WebSockets

V úvodu této kapitoly byl zmíněn AJAX a přístup k datům v databázi na pozadí. Technologie WebSocket (dále WS), která je součástí HTML5, tento přístup dotáhla k dokonalosti, a to díky komunikaci skrze TCP pakety. Jedná se tedy o spojovou komunikaci mezi klientem a serverem, která je potvrzovaná oběma stranami – jak klientem, tak serverem. Komunikace probíhá pomocí PUSH technologie v reálném čase a probíhá pouze, pokud je potřeba (na rozdíl od nespojové komunikace, která probíhala např. v pravidelných předem definovaných intervalech a zjišťovala, zda se událo něco nového). Server s prohlížečem nejdříve naváže spojení skrze HTTP protokol (port 80, díky čemuž projde bez problémů skrze firewally) a následně si mezi sebou vyměňují datové pakety. WS je navržen předně pro implementaci ve webových prohlížečích, ale prakticky ho lze využít v jakékoliv formě komunikace klient – server, kde je žádoucí komunikace v reálném čase [16].

Na straně klienta je WS implementován v JavaScriptu, na straně serveru může být využito různých technologií, jako např. PHP či Python.

4.6.1 Připojení

Připojení je realizováno ze strany klienta pomocí příkazu

```
new WebSocket("ws://server:port")
```

Přičemž ws:// udává, že se bude jednat o komunikaci skrze nezabezpečený WebSocket (pokud bychom disponovali SSL certifikátem a chtěli tak mít šifrovanou komunikaci, prefix by byl wss). Server udává adresu zařízení, na kterém aplikace běží a port určuje číslo komunikačního portu.

4.6.2 Události

WS využívá tři události při komunikaci mezi klientem a serverem

4.6.2.1 onopen

Funkce se volá při navázání spojení, podává informaci o tom, že můžeme začít komunikovat.

4.6.2.2 onmessage

Výměna dat mezi klientem a serverem. Skrze WS lze přenášet pouze základní řetězce, což je asi jediná nevýhoda tohoto způsobu komunikace. Naštěstí existuje řešení pomocí JSON.

4.6.2.3 onclose

Oznamuje uzavření spojení.

5 Komponenty a technologie použité k tvorbě aplikace

5.1 Autentifikace uživatele

Každý člověk má specifické hlasové ústrojí. To, jak rychle mluvíme, jaké děláme pauzy, či používání intonace velmi ovlivňuje parametry promluvy. Pokud je naším úkolem rozpoznat řečníka, zajímají nás hlavně parametry související s intonací a ne s gramatikou, jelikož to, co bylo řečeno, není v tomto kontextu důležité. Úlohu rozpoznávání tedy můžeme rozdělit na identifikaci a verifikaci [17].

Verifikací rozumíme ověření, zda daná hlasová promluva patří konkrétnímu uživateli. Podle úspěšnosti rozpoznávání můžeme stanovit tzv. verifikační míru, která udává, s jakou pravděpodobností se jedná o daného uživatele nebo s jak velkou pravděpodobností by se mohlo jednat o jiného uživatele. Pokud systém vyhodnotí, že je míra příliš nízká, řečník verifikací neprošel a nejspíše se jedná o podvodníka.

Identifikace na rozdíl od verifikace nepracuje s jedním konkrétním řečníkem, ale snaží se v určité uzavřené množině osob najít tu, které patří daný hlasový záznam. V případě, že si systém není jistý, zda se opravdu jedná o danou osobu, může provést dodatečně verifikaci řečníka, při které již s jistotou může označit, zda řečník patří, či nepatří do dané množiny. Podobně jako u verifikace i zde si můžeme určit míru pravděpodobnosti, která udává, s jak velkou jistotou patří řečník do dané množiny.

Při volbách míry je vhodné vždy provést několik sérií testování, jelikož při nesprávném nastavení by se nám mohlo stát, že systém zamítá řečníky, kteří spadají do dané množiny, a naopak povolí řečníky, kteří v množině nejsou. V praxi je vhodné spíše nastavit systém přísnější, aby někdy nepovolil vstup správnému řečníkovi, ale vždy s jistotou zamezil v přístupu nesprávnému člověku. Pro uživatele sice další fáze ověření nemusí být příjemná, hlavní však je, aby se k jeho datům, např. k bankovnímu účtu, nedostal cizí člověk.

5.1.1 Rozpoznávání řečníka

Systémy rozpoznávání řečníka můžeme rozdělit na dva základní typy:

- Textově závislé – rozpoznávání probíhá na stejném textu, jako probíhalo trénování
- Textově nezávislé – dle parametrů promluvy, nezávisle na textu (fonetické jevy)

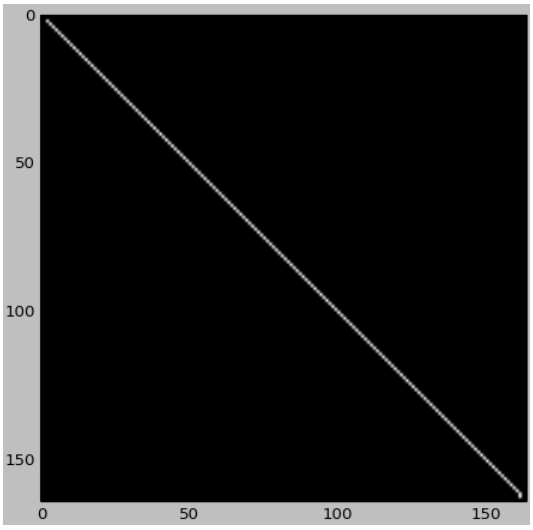
Pro textově závislé systémy se pro porovnání hlasového vstupu využívá dynamického programování. Algoritmus je velmi podobný Levenshteinově vzdálenosti, která udává odlišnost dvou řetězců. Definujeme ji jako minimální počet operací vkládání, substitucí a

mazání takových, aby oba řetězce byly totožné. Odlišnost zjistíme tak, že sestavíme tabulku, ve které porovnáváme referenční a vstupní data [18].

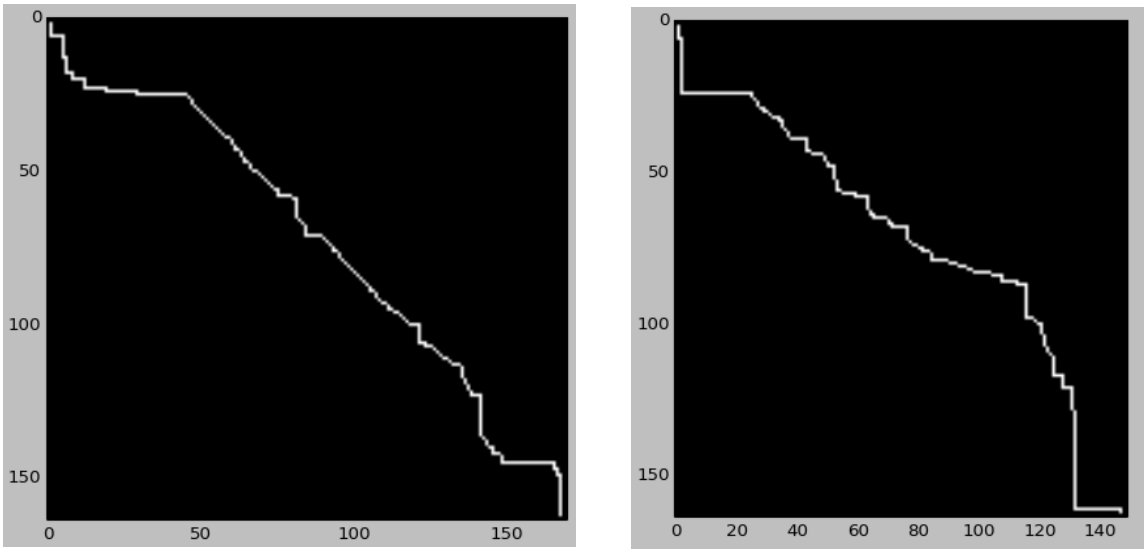
Pohyb tabulkou je následující: pokud na indexu [i,j] jsou stejné hodnoty, označíme průsečík jako identický a postupujeme diagonálně na další index. V případě, že se jedná o různé znaky, pak je nutné využít jednu z operací, kdy každá z operací má určitou cenu.

- Vkládání – vertikální pohyb tabulkou
- Mazání – horizontální pohyb tabulkou
- Substituce – diagonální pohyb tabulkou

Abychom zajistili, že celkový počet operací bude konečný a zároveň minimální, tak použijeme v každém kroku operaci, která má nejnižší cenu. Na obrázcích 5.1 a 5.2 jsou znázorněny průběhy algoritmu pro různé případy nahrávek.



Obr. 5. 1 Ideální případ – porovnání 2 identických nahrávek



Obr. 5. 2 Porovnání odlišných nahrávek stejného obsahu od stejného řečníka (vlevo) a různých řečníků (vpravo)

Z posledního grafu je velmi patrné, že kromě odlišnosti promluvy se nahrávky liší i v délce, každý řečník promluvu pronesl v jiném tempu. Tato skutečnost se výrazně projevuje na celkovém hodnocení dané nahrávky.
Testově nezávislé systémy využívají pro porovnávání modely Gausovských směsí (GMM), analýzy společných faktorů (JFA) nebo zjednodušené analýzy společných vektorů (iVectors).

Dále se budeme zabývat vyhodnocením softwarové komponenty SpeechTech provádějící textově závislou verifikaci řečníka. Program využívá kombinace porovnávání více nahrávek, díky čemuž se s vyšší pravděpodobností identifikuje daný řečník.

5.1.2 Popis

Program je konzolová aplikace, napsaná v jazyce C++. Aplikace nezpracovává data za běhu, ale využívá pořízené nahrávky, díky čemuž nejsou kladeny žádné požadavky na procesor systému. Zpracování probíhá přímo na zařízení a nevyužívá se grafické karty. Paměťové nároky také nejsou příliš velké, a proto lze program využívat i na starších zařízeních. Jedinou limitací je pak rychlost zpracování a následná odezva systému. Program by měl být nezávislý na platformě a byl otestován na zařízeních Windows 7, 10 a Debian Wheezy, Jessy.

Verze programu je ze dne 07.05.2015 a využívá VAD, který byl vytvořen ke dni 04.11.2014.

5.1.2.1 Zpracování dat

Pro práci s hlasovými nahrávkami musíme nejdříve provést matematické operace, pomocí kterých hlas převedeme do potřebného maticového formátu. K tomuto účelu slouží parametrizace dat, která každou část nahrávky aproximuje posloupností čísel. Každé číslo nám udává jiné parametry promluvy. V těchto parametrech se skrývají veškerá řečová data, jako např. co bylo řečeno či jak přesně to bylo řečeno [19].

O tyto matematické úkony se starají knihovny ACML (AMD Core Math Library) nebo BLAS (Basic Linear Algebra Subroutines).

5.1.2.2 Vstup

Vstupem programu jsou testovaná hlasová nahrávka, kterou chceme analyzovat, a referenční hlasové nahrávky, na základě kterých program vypočítá pravděpodobnost,

s jakou se jedná o daného uživatele. Počet vzorových nahrávek je libovolný, minimálně však musí být vždy jedna.

Nahrávky musí být ve stejném formátu o jednom kanále a stejné vzorkovací frekvenci.

Dalším vstupem programu je VAD (Voice Activity Detection), neboli detektor řeči. Úkolem hlasové detekce je určit, zda se jedná o vstup řečníka či o šum na pozadí nahrávky. Princip detekce je v tzv. frame dropping, neboli zahazování segmentů. Pokud program vyhodnotí, že je řečový úsek nevýznamný, nevyhodnocuje ho. Tato skutečnost je velmi důležitá, jelikož pokud například na pozadí místnosti hraje rádio, program by mohl nesprávně detekovat, že se jedná o uživatele a znehodnotit tak samotnou nahrávku.

5.1.2.3 Volání programu

password.exe VAD.bin **vstup** vzory

5.1.2.4 Výstup

P(user|score) – pravděpodobnost

Udává v procentech pravděpodobnost, s jakou se jedná o daného uživatele.

P(P(user|score)) – věrohodnost

Udává v procentech věrohodnost vypočtené pravděpodobnosti – do jaké míry je určení, že se jedná o daného uživatele, správné.

5.1.3 Návrh testu

5.1.3.1 Data

Testování proběhne na nahrávkách pořízených v rámci vývoje aplikace pro autentifikaci řečníka. Celkem byly pořízeny stovky nahrávek od mužů i žen různých věkových kategorií. Z těchto nahrávek bylo náhodně vybráno několik reprezentantů z řad mužů a žen pro získání základního přehledu ohledně funkčnosti aplikace.

Nahrávání probíhalo na různých zařízeních (mobilních či stolních telefonech) a především v různém prostředí. Z těchto důvodů se může kvalita jednotlivých nahrávek lišit a ovlivnit tak nepatrně výsledek. Pro základní představu a doporučení jsou však tato data dostačující.

Testování proběhne na čtyřech hlasových nahrávkách:

1. Hasičská vzájemná pojišťovna
2. Babička
3. Brána
4. Ano

Každá nahrávka má různou délku, která poslouží ke zjištění minimální promluvy pro úspěšnou detekci uživatele. Nahrávky jsou získány z telefonních hovorů, jejich vzorkovací frekvence je tedy 8 000 Hz.

5.1.3.2 Stavy klasifikace

Definujme několik různých stavů, do kterých se můžeme při klasifikaci dostat:

- Pozitivní – do systému je vpuštěn správný uživatel
- Falešně pozitivní – do systému je vpuštěn špatný uživatel
- Negativní – cizí uživatel nevpuštěn
- Falešně negativní – správný uživatel není vpuštěn do systému

Můžeme je zobecnit do následující tabulky:

	Uživatel rozpoznán	Uživatel nerozpoznán
Jedná se o uživatele	Pozitivní (P)	Falešně negativní (FN) chyba 1. druhu
Nejedná se o uživatele	Falešně pozitivní (FP) chyba 2. druhu	Negativní (N)

Tabulka 5. 1 Stavy klasifikace

Tabulka 5.1 nám udává možnosti klasifikace daného uživatele. Pokud je uživatel v databázi a je rozpoznán, systém funguje správně. Pokud se nejedná o správného uživatele a systém ho nerozpozná, jedná se opět o správnou funkčnost. Pokud uživatel je v systému, ale systém ho omylem označí, že se v systému nenachází, jedná se o falešně negativní výsledek. Pokud uživatel v systému není a systém ho rozpozná, jedná se o falešně pozitivní výsledek.

Naším cílem je tedy získat takové nastavení, aby v ideálním případě nedocházelo k žádnému označení výsledku jako falešně pozitivní, jelikož by pak mohl cizinec být klasifikován jako uživatel v systému. Ve statistice se tato chyba značí jako chyba druhého druhu. Pro vyhodnocení testů si nejdříve nadefinujme několik pojmů, které nám zhodnocení ulehčí [20].

5.1.3.3 Metoda F1 skóre a ROC křivka

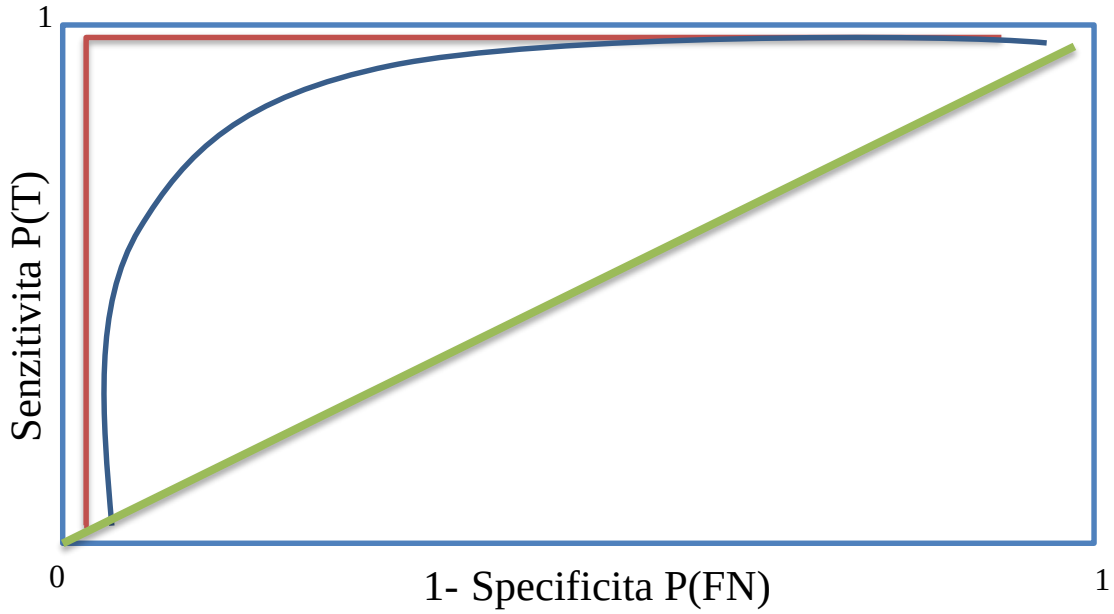
Definujme senzitivitu (citlivost) testu, která vyjadřuje úspěšnost detekce správného uživatele a specificitu, která vyjadřuje úspěšnost, se kterou zachytíme pouze nesprávného uživatele – do jaké míry zamítáme přístup.

$$senzitivita = \frac{správný\ uživatel\ rozpoznán}{(správný\ uživatel\ rozpoznán + nesprávný\ uživatel\ rozpoznán)} = \frac{P}{(P + FP)}$$

$$specificita = \frac{uživatel\ správně\ zamítnut}{(uživatel\ správně\ zamítnut + správný\ uživatel\ nerozpoznán)} = \frac{N}{(N + FN)}$$

Tyto dva vztahy můžeme graficky znázornit pomocí ROC (Receiver Operating Characteristic) křivky, která se užívá pro hodnocení a optimalizaci klasifikačních systémů. Senzitivita P(T) na svislé ose grafu udává relativní četnost pozitivních případů, tj. pravděpodobnost, že jako správný bude vyhodnocen pozitivní případ (byl ověřen správný uživatel). P(FP) udává relativní četnost falešně negativní případů, tedy že bude zamítnut uživatel, který se v systému nachází. Získáme ji jako 1 - specificita.

Nastavováním prahových hodnot se pohybujeme po křivce a hledáme kompromis mezi oběma vztahy. Kvalitu hodnotíme podle obsahu plochy pod křivkou. Na obrázku 5.3 jsou zobrazeny tři průběhy ROC křivky. Ideální stav (červeně) je takový, kdy křivka stoupá podél osy Y a teprve poté se pohybuje podél osy X. Takový test má 100% senzitivitu i specificitu. Zelená přímka udává mez, nad kterou by se křivka měla vždy pohybovat (pravděpodobnost ½ je rovna hodu mincí). Modrá křivka je nejčastější reálný průběh.



Obr. 5. 3 ROC křivka

Dále si vyčíslíme precisnost, neboli prediktivní hodnotu pozitivního testu – tj. pravděpodobnost, že jev je skutečně pozitivní, když test vyšel pozitivně. Jinými slovy s jakou úspěšností rozpoznáme správně uživatele, který se nachází v systému.

$$Precisnost = \frac{správný\ uživatel\ rozpoznán}{(správný\ uživatel\ rozpoznán + správný\ uživatel\ nerozpoznán)} = \frac{T}{(T + FN)}$$

S pomocí precisnosti můžeme číselně vypočítat míru úspěšnosti testu za pomocí F1 skóre, které je harmonickým průměrem precisnosti a citlivosti. Harmonický průměr $\widehat{Ex}$ se počítá jako převrácená hodnota aritmetického průměru Ex převrácených hodnot členů.

$$Ex = \frac{\sum_{i=1}^{n} xi}{n} \qquad \widehat{Ex} = \frac{n}{\frac{1}{\sum_{i=1}^{n} xi}}$$

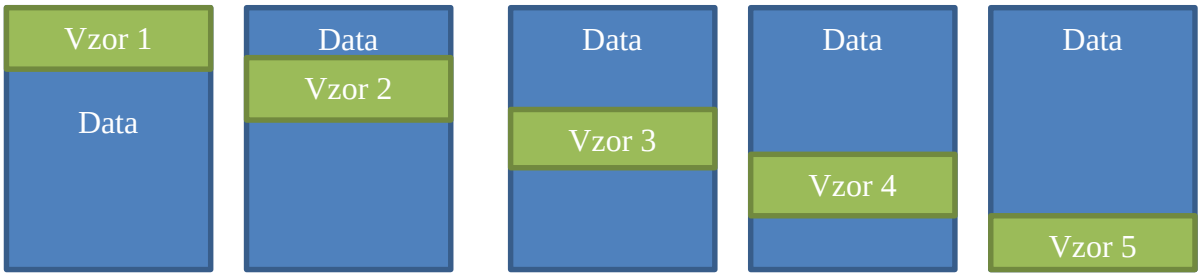
Po úpravě dostaneme vztah

$$F1 = \frac{2 \cdot T}{(2 \cdot T + FP + FN)}$$

5.1.4 Testování

5.1.4.1 Test 1

V prvním testu zjistíme, jaká je úspěšnost detekce uživatele při využití trénovacích a testovacích dat pouze od daného uživatele. Na vstupu budeme uvažovat pouze stejné nahrávky (stejný text promluvy) a od stejného uživatele, abychom získali základní přehled o funkčnosti aplikace. Testování proběhne na celkem šesti nahrávkách, se stejným poměrem pohlaví tzv. stylem každý s každým. Pro lepší představu jej znázorněme na obrázku 5.4. Na vstupu máme 2 až 7 souborů a vždy každý z nich bude použit jako vstup a ostatní jako uložené vzory, s kterými se soubor na vstupu bude srovnávat. Tento test proběhne vždy N krát pro každý počet souborů, díky čemuž získáme průměrnou hodnotu vždy pro N-1 vzorových souborů.



Obr. 5. 4 Ukázka procházení dat

Testovat budeme úspěšnost a věrohodnost rozpoznávání při daném počtu vzorových souborů. Tento způsob testování nám dá základní informaci o tom, kolik vzorových souborů je třeba k tomu, aby byl úspěšně rozpoznán správný uživatel. Data z testování jsou k dispozici v příloze 2, Test 1.

Průměrné hodnoty promluv							
	Vzorů	1	2	3	4	5	6
Hasičská vzájemná pojišťovna	pravděpodobnost	80,873	95,043	99,673	99,926	99,970	99,966
	věrohodnost	43,852	71,846	86,989	93,437	95,654	95,884
Babička	pravděpodobnost	75,708	86,818	95,700	94,147	95,116	96,478
	věrohodnost	27,330	52,884	59,821	65,862	68,771	70,265
Brána	pravděpodobnost	61,718	91,038	92,299	96,773	98,723	99,525
	věrohodnost	52,538	67,783	80,694	84,268	92,155	92,093
Ano	pravděpodobnost	55,219	77,206	81,145	85,442	89,594	90,551
	věrohodnost	48,573	69,710	80,634	84,594	87,954	88,747
Průměr	pravděpodobnost	**68,380**	**87,526**	**92,204**	**94,072**	**95,851**	**96,630**
	věrohodnost	**43,073**	**65,556**	**77,034**	**82,040**	**86,133**	**86,747**

Tabulka 5. 2 Průměrné hodnoty testu

Z prvního testu z tabulky 5.2 je patrné, že při promluvě o délce tří slov stačí tři vzorové soubory, abychom mohli říct s 92% přesností a 77% věrohodností, že se jedná o daného uživatele. Z výsledků je také patrný vliv kvality a délky nahrávek na výsledné pravděpodobnosti. Slovo babička dosahuje horších výsledků než slovo brána. Důvodem jsou nekvalitní nahrávky slova babička od uživatelů Muž 3 a Žena 3 (viz přílohu 2).

5.1.4.2 Test 2

Uvažujme nyní o případu, kdy máme uživatele, kteří mají přístup do systému. Ke každému uživateli jsou v systému přiřazeny referenční nahrávky (viz obrázek 5.5).

Obr. 5. 5 Ukázka procházení dat

Systém budeme testovat celkem 18 nahrávkami, z kterých se 8 shoduje s nahrávkami v systému, a dalších 10 nahrávek je od cizích uživatelů, kteří do systému přístup nemají. Nahrávku na vstupu budeme porovnávat postupně s každou nahrávkou v systému a vyhodnotíme, ke kterému uživateli nahrávka s největší pravděpodobností patří.

$$rozhodnutí = \arg\max Pi$$

Data z testování jsou k dispozici v příloze 2, Test 2. Považujme za správně rozpoznaného uživatele takového, který má pravděpodobnost shody alespoň 75%, pro lepší znázornění označme takové výsledky zeleně.

Vezměme v úvahu dva případy, v prvním případě (A) za správně určenou nahrávku budeme považovat takovou, která má kromě **pravděpodobnosti** alespoň **75%** současně **věrohodnost** alespoň **60%**. V tom druhém (B) nebudeme brát zřetel na věrohodnost výsledku – budeme se orientovat pouze na základě vypočítané pravděpodobnosti.

Již na první pohled na pravděpodobnosti výsledků je patrné, že tento způsob ověření asi nebude ideální. Výsledky testu můžeme zpracovat pomocí tabulky 5.1.

Vzor	Případ A				Případ B			
	P	N	FP	FN	P	N	FP	FN
hasičská vzájemná pojišťovna	2	10	0	6	5	10	0	3
babička	5	10	0	3	7	10	0	1
brána	4	10	0	4	5	10	0	3
ano	3	10	0	5	4	10	0	4

Tabulka 5. 3 Klasifikace do stavů

V tabulce 5.3 vidíme, že se data liší pouze v počtu pozitivních a falešně negativních případů, v obou případech tedy došlo k zamítnutí všech nežádoucích uživatelů. Údaje z tabulky si dále rozebereme pomocí dříve definovaných vztahů.

Senzitivita				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	1	100%	1	100%
babička	1	100%	1	100%
brána	1	100%	1	100%
ano	1	100%	1	100%

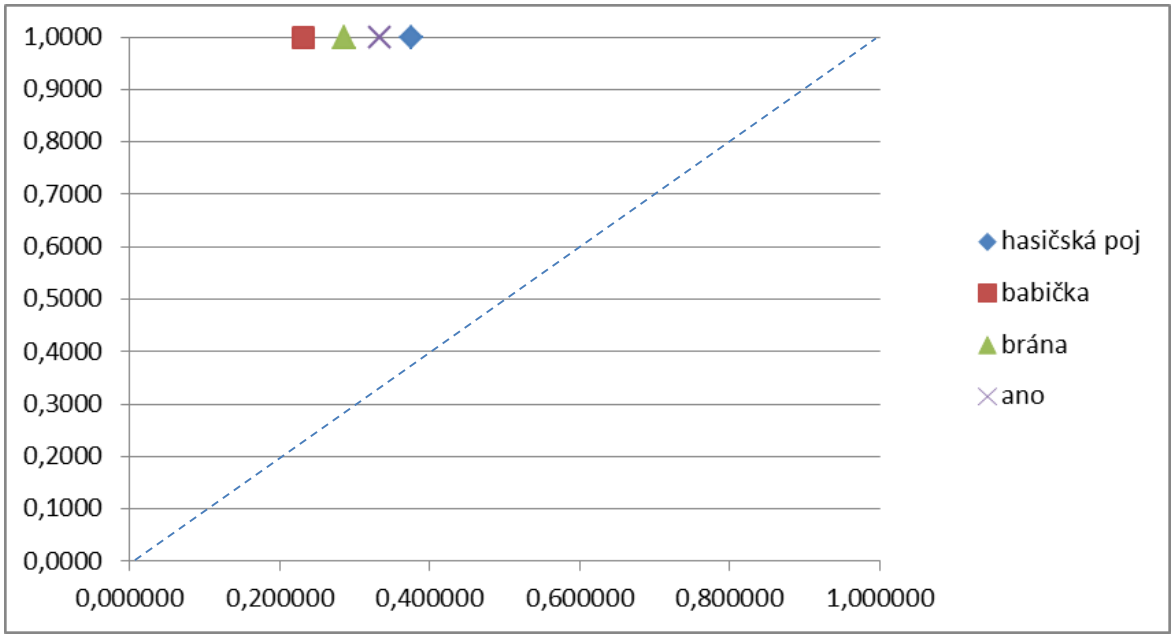
Tabulka 5. 4 Senzitivita

Tabulka 5.4 udává, že žádný uživatel, který neměl být vpuštěn, vpuštěn nebyl. Povedlo se nám tedy v obou případech zamítnout přístup všem nesprávným uživatelům, což je pro nás nejdůležitějším kritériem – do systému se nedostane žádný nežádoucí uživatel.

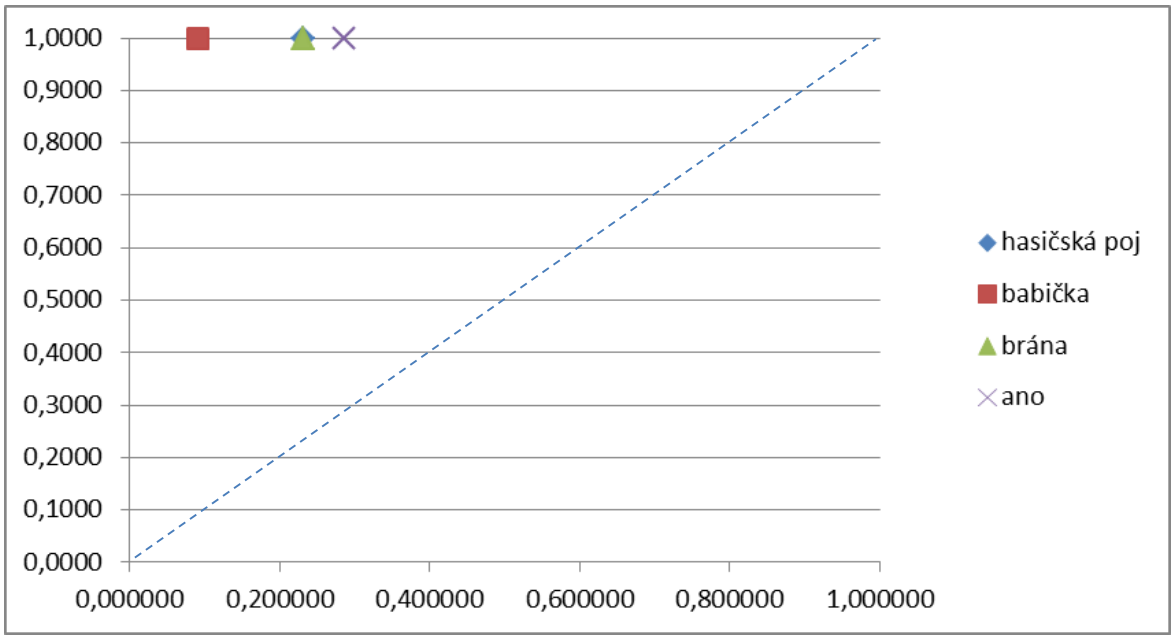
Specificita				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,62500	62,50%	0,76923	76,92%
babička	0,76923	76,92%	0,90909	90,91%
brána	0,71429	71,43%	0,76923	76,92%
ano	0,66667	66,67%	0,71429	71,43%

Tabulka 5. 5 Specificita

Tabulka 5.5 udává, do jaké míry do systému zamítáme přístup. Pokud bychom do systému nevpouštěli pouze nežádoucí uživatele, hodnoty by byly stejné jako v předchozí tabulce. Vyšší hodnoty v případě B naznačují, že nedocházelo k tak častému vyhodnocení falešně pozitivních případů (správný uživatel, který má do systému přístup, nebyl označen za cizího) jako v případě A.



Obr. 5. 6 ROC křivka pro data s věrohodností



Obr. 5. 7 ROC křivka pro data bez věrohodnosti

Z grafů na obrázcích 5.6 a 5.7 je patrné, že se body nacházejí ve správné polorovině a dokonce se blíží ideální křivce. Zobrazené hodnoty se shodují s vypočtenými pravděpodobnostmi. Porovnáme-li oba grafy, je vidět, že v případě B došlo k lepšímu nastavení – posunutí do leva.

Precisnost				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,25	25,00%	0,625	62,50%
babička	0,625	62,50%	0,875	87,50%
brána	0,5	50,00%	0,625	62,50%
ano	0,375	37,50%	0,5	50,00%

Tabulka 5. 6 Precisnost

Z tabulky 5.6 je zřejmé, že správné uživatele v případě B rozpoznáváme ve větší míře, než je tomu v případě A. Dochází tak k menšímu počtu chyb 1. druhu, které nejsou pro systém tak kritické, jako spíše nepříjemné pro uživatele, po kterých požadujeme nové ověření.

Posledním ukazatelem je F1 skóre, které nám číselně zobrazí úspěšnost testu – obdobně jako ROC křivka.

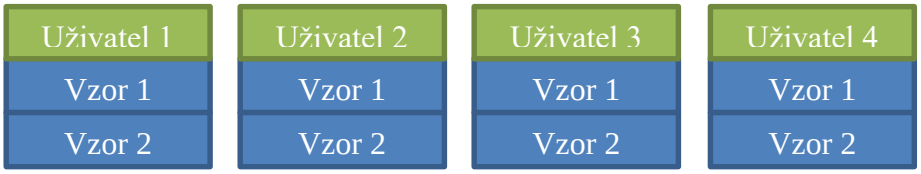
F1 skóre				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,4	40,00%	0,76923	76,92%
babička	0,76923	76,92%	0,93333	93,33%
brána	0,66667	66,67%	0,76923	76,92%
ano	0,54545	54,55%	0,66667	66,67%

Tabulka 5. 7 F1 skóre testu

Tabulka 5.7 udává, že v případě A je míra úspěšnosti velmi nízká, a tudíž systém je v praxi nepoužitelný. Z tabulky je patrné, jak moc se míra úspěšnosti zlepšila v případě B oproti případu A – při nebrání v potaz věrohodnosti výsledku. Věrohodnost je však velmi důležitá, jelikož pokud je menší než 50%, nemůžeme si být jisti, zda je vypočtená pravděpodobnost správná. Tento test nám potvrdil předpoklad, že zvolená metoda není ideální. Budeme proto muset najít jiný způsob ověření uživatele.

5.1.4.3 Test 3

V dalším testu nebudeme porovnávat vzorovou nahrávku s každou nahrávkou od uživatele zvlášť, ale se dvěma nahrávkami najednou. Tímto způsobem bychom měli získat kvalitnější odhady pravděpodobnosti i věrohodnosti, jelikož aplikace bude počítat průměr ze dvou referenčních nahrávek najednou. Grafické znázornění testování je na obrázku 5.8.



Obr. 5. 8 Ukázka procházení dat

Data z testování jsou opět k dispozici v příloze 2, Test 3. Uvažujme opět za správně rozpoznaného uživatele takového, který má pravděpodobnost shody alespoň 75%, pro lepší znázornění označme takové výsledky zeleně.

Z dat je patrné, že nyní dosahuje test mnohem lepších výsledků než v předchozím případě. Pro první případ vyhodnocení (A) budeme opět považovat za správně určenou nahrávku takovou, která má **pravděpodobnost** alespoň **75%** a **věrohodnost** alespoň **60%**. Druhý případ (B) bude opět bez věrohodnosti výsledku.

Vzor	Případ A				Případ B			
	P	N	FP	FN	P	N	FP	FN
hasičská vzájemná pojišťovna	5	10	0	3	6	10	0	2
babička	7	10	0	1	7	9	1	1
brána	5	10	0	3	6	10	0	2
ano	4	10	0	4	6	10	0	2

Tabulka 5. 8 Klasifikace do stavů

Při bližším pohledu na data v tabulce 5.8 zjistíme, že případ A je nyní totožný s případem B z minulého testu, veškeré výsledky tedy budou totožné. Nyní se v případě B vyskytl negativní případ, což znamená, že jsme do systému pustili nežádoucího uživatele a dopustili se tak chyby 2. druhu.

Senzitivita				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	1	100%	1	100%
babička	1	100%	0,875	88%
brána	1	100%	1	100%
ano	1	100%	1	100%

Tabulka 5. 9 Senzitivita

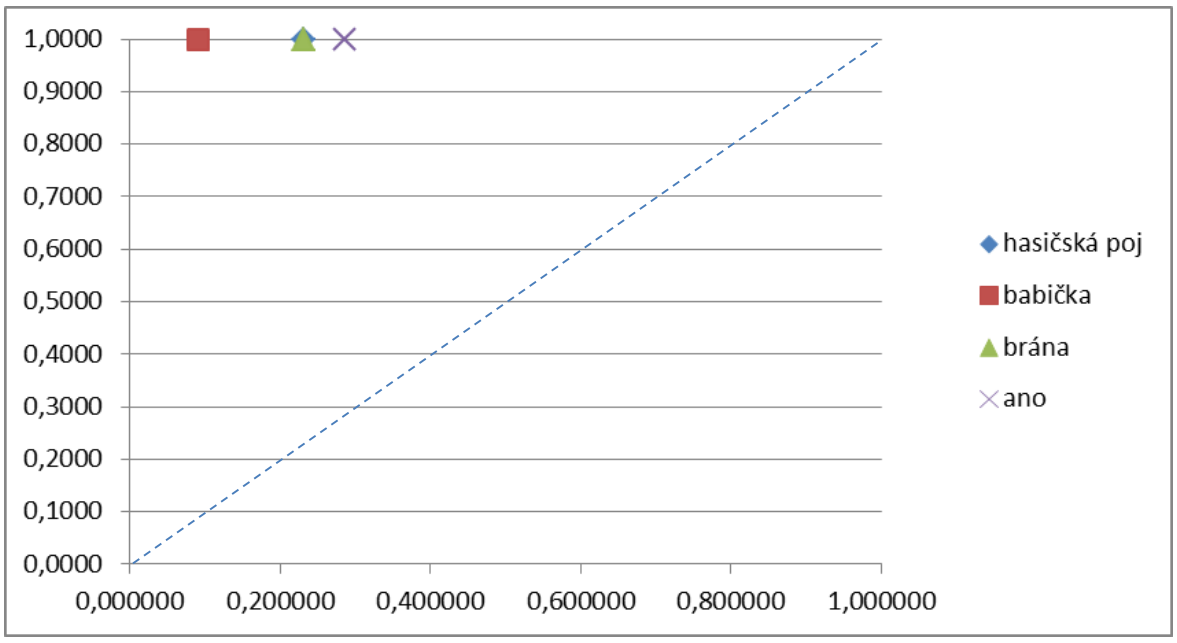
Negativní případ z tabulky 5.8 se projevil na senzitivitě znázorněné v tabulce 5.9, která nyní klesla.

Specificita				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,76923	76,92%	0,83333	83,33%
babička	0,90909	90,91%	0,90000	90,00%
brána	0,76923	76,92%	0,83333	83,33%
ano	0,71429	71,43%	0,83333	83,33%

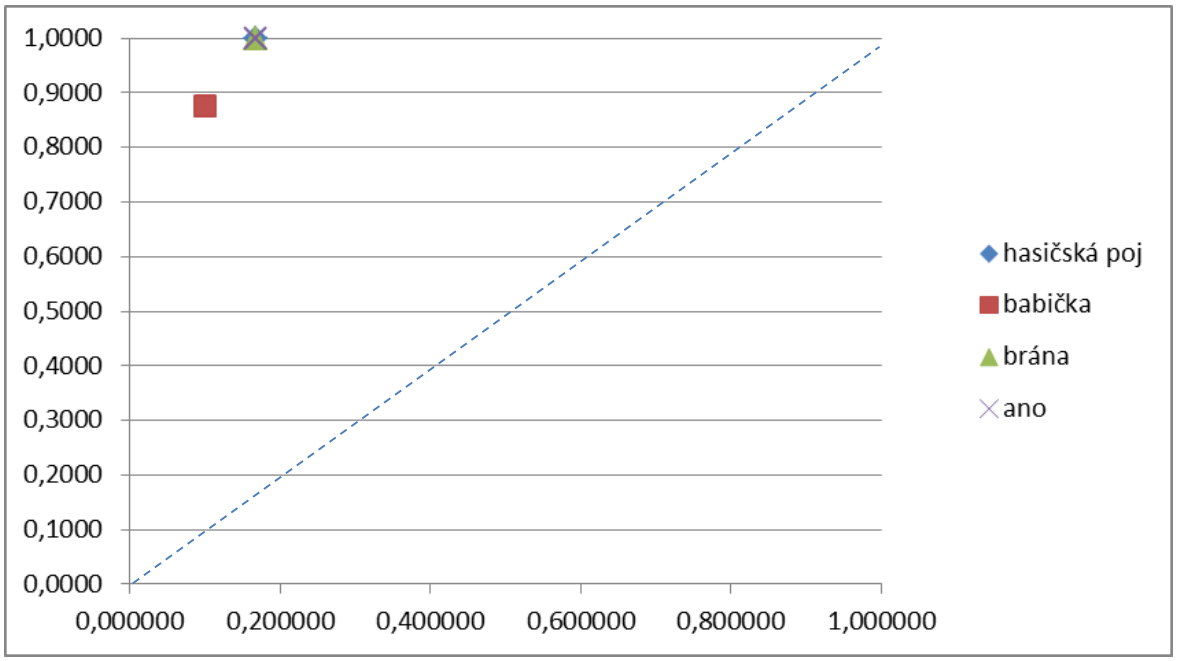
Tabulka 5. 10 Specificita

Na rozdíl od Senzitivity, Specificita z tabulky 5.10 v případě B dosahuje lepších výsledků než v případě A, systém tedy vpustil více řečníků, kteří do systému mají přístup.

Tyto skutečnosti se projeví i na ROC křivce na obrázku 5.10 posunutím hodnot doleva a v případě slova babička i negativním posunutím dolů. Na obrázku 5.9 je znázorněna ROC křivka z testu s věrohodností.



Obr. 5.9 ROC křivka pro data s věrohodností



Obr. 5.10 ROC křivka pro data bez věrohodností

Jelikož do systému bylo vpouštěno více uživatelů, stoupne tím pádem v případě B i hodnota preciznosti, znázorněná v tabulce 5.11.

Precisnost				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,625	62,50%	0,75	75,00%
babička	0,875	87,50%	0,875	87,50%
brána	0,625	62,50%	0,75	75,00%
ano	0,5	50,00%	0,75	75,00%

Tabulka 5. 11 Precisnost

Závěr testu opět zhodnotíme pomocí F1 skóre. Srovnáme-li hodnoty B a A F1 skóre v tabulce 5.12, zjistíme, že opět došlo k zlepšení jako v minulém testu, avšak s výjimkou slova babička, u kterého jsme dosáhli chyby 2. druhu, která je pro systém kritická.

F1 skóre				
Vzor	Případ A		Případ B	
hasičská vzájemná pojišťovna	0,76923	76,92%	0,85714	85,71%
babička	0,93333	93,33%	0,87500	87,50%
brána	0,76923	76,92%	0,85714	85,71%
ano	0,66667	66,67%	0,85714	85,71%

Tabulka 5. 12 F1 skóre

Na závěr můžeme ještě srovnat výsledky tohoto testu s předchozím

F1 skóre				
Vzor	**Test 2-A**	**Test 2-B**	**Test 3-A**	**Test 3-B**
hasičská vzájemná pojišťovna	40,00%	76,92%	76,92%	85,71%
babička	76,92%	93,33%	93,33%	87,50%
brána	66,67%	76,92%	76,92%	85,71%
ano	54,55%	66,67%	66,67%	85,71%

Tabulka 5. 13 F1 skóre pro všechny testy

Z výsledku z tabulky 5.13 je patrné, že věrohodnost výsledku je velmi důležitým kritériem pro klasifikaci. Přílišné snižování tohoto kritéria může mít negativní dopad na bezpečnost systému.

5.1.5 Vyhodnocení testu

Testování proběhlo na nahrávkách, které se značně lišily. Nejenže byly nahrány v jiném prostředí, ale také skrze jiná zařízení. Tento fakt má značný vliv na výsledné hodnoty pravděpodobností a věrohodností. S kvalitními nahrávkami by bylo dozajista možno dojít k značně rozlišným výsledkům. Jakožto prvotní ukazatel nám však tento test postačí.

5.1.5.1 Aplikace autentifikace uživatele v systému

Z provedených testování je patrné, že se výsledky významně liší při využití různého počtu referenčních nahrávek. Je na programátorovi aplikace, případně na uživateli, jakou míru zabezpečení chce zvolit – zda mu stačí pravděpodobnost 70% či zda musí být alespoň 90%. Dále je pro správnou klasifikaci zásadní počítání s věrohodností výsledku a následné zohlednění pro přijímání či zamítání hypotézy.

5.1.5.2 Ověření uživatele

Velmi záleží na výkonnosti stroje, na kterém aplikace běží. Pokud dokážeme za zlomek vteřiny porovnat stovky uživatelů, kteří mají 5 referenčních nahrávek, získáme s naprostou určitostí pravděpodobnosti shody a věrohodnost výsledku přes 90%. Pokud však systém nebude běžet na tak výkonném stroji, jako nejvhodnější řešení se jeví v prvním kole otestovat uživatele pomocí 2 referenčních nahrávek a následně při nízké pravděpodobnosti či věrohodnosti výsledku využít více referenčních nahrávek pro ověření, zda se opravdu jedná o daného uživatele.

5.2 SpeechCloud

Cloud v češtině znamená oblak a ve světě IT se používá pro označení sítě vzájemně propojených serverů, které poskytují různé služby. Aplikace, která na nich běží, tak může využívat určité servery například pro výpočetní úkony a jiné servery zase například pro ukládání dat. Velmi oblíbené jsou i Cloudová uložiště dat, u kterých má uživatel k dispozici data na jakémkoli zařízení (PC, mobilní telefon, tablet) – nejznámějšími jsou například Google Disk, One Drive či Dropbox.

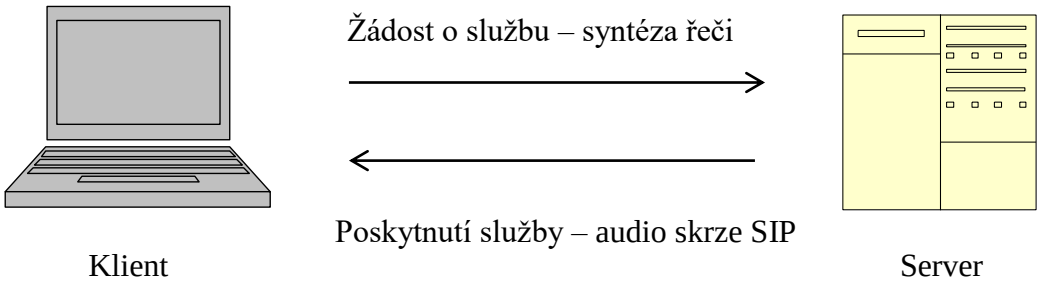
Uživatel si tedy, aniž by disponoval drahým technickým zařízením, které zvládne výpočetně složité úkony, místo toho pronajme server, který vykoná všechnu složitou práci ve zlomku sekundy. Jedna aplikace běžící na takovém serveru tak může poskytovat služby desítkám či stovkám klientů. Tento způsob umožňuje vývojářům poskytovat kvalitní a rychlé služby, které jsou zároveň cenově dostupné. Vývojář pouze pronajímá své aplikace, díky čemuž se nemusí bát o své know how a zákazník má zase pravidelně aktualizovanou a udržovanou službu, čili nemusí mít ani potřebné znalosti o dané problematice [21].

Nevýhodou, která již byla zmíněna v úvodu této práce, je, že servery se fyzicky nacházejí na jiném místě a komunikace tak probíhá skrz internet. V případě výpadku spojení tak nelze garantovat funkčnost poskytované cloudové aplikace. Další nevýhodou je zpracování citlivých dat třetí osobou či nemožnost konfigurace serverů dle potřeb individuálního klienta. Způsob komunikace je znázorněn na obrázku 5.11.

5.2.1 Předávání informací

SpeechCloud je aplikační rozhraní, které poskytuje služby rozpoznávání a syntézy řeči. Aplikace vznikla na katedře kybernetiky s využitím knihoven a technologií od firmy SpeechTech.

Komunikace mezi klientem a serverem probíhá skrze WebSocket protokol za pomocí vzájemných JSON zpráv. Audio stream je vytvořen pomocí SIP protokolu, který využívá RTP (Real-time Transport protocol) UDP pakety. Klient vznese požadavek pomocí určité metody, kterou server zpracuje a následně klientovi odpoví patřičnou událostí. Tento způsob komunikace je znázorněn na obrázku 5.11.



Obr. 5.11 Komunikace mezi klientem a serverem

SpeechCloud běží na serveru o následující konfiguraci:

IBM System x3550 M4, 2x12 jader CPU E5-2620 v2 @ 2.10GHz, 128GB paměti

Díky výkonnosti serveru lze zpracovat jakýkoliv požadavek v řádech setin sekundy a vykonávat tak efektivně dané úkony.

5.2.2 Rozpoznávání řeči

V úvodu u Hlasového dialogového systému byl popsán princip, na kterém funguje rozpoznávání řeči.

$$\widehat{W} = arg\max_{W} P(W|O) = arg\max_{W} \frac{P(W) \cdot P(O|W)}{P(O)}, kde$$

$P(O|W)$ je akustický model a $P(W)$ jazykový model a $P(O)$ parametrizované řetězce příznaků.

5.2.2.1 Akustický model

Aktuální akustický model, který je v aplikaci využit, nese kódové označení TEL05b-2k0-16G, verze 7977 ke dni 11.2.2015. Model vznikl z telefonních nahrávek, čtených promluv a datového korpusu SpeechDat. Je tvořen tří-stavovými levopravými HMM modely s 2 000 stavy, přičemž 1 stav má 16 gausiánů. Jedná se o telefonní nahrávky, které byly parametrizovány pomocí PLP parametrizace s 12 koeficienty a frekvencí 8 000 Hz, která je dostatečná, jelikož telefonní hovory mají frekvenční rozsah 300 Hz – 3 400 kHz. Vzorkování tak splňuje Shannonův teorém (vzorkovací frekvence je minimálně dvojnásobná než největší frekvence obsažená v signálu). Následně byla provedena normalizace na střední hodnotu. Zvuk je monofonní, neboli jednokanálový. Jednokanálové zvuky jsou takové, které nevyvolávají prostorový dojem zvuku.

5.2.2.2 Jazykový model

Gramatiku si definuje uživatel sám a serveru ji následně pouze předává. Díky tomuto způsobu může být neustále upravována množina všech možných promluv a lze tak systém naučit nová slova.

Výhodou je, že gramatiky lze za běhu aplikace i měnit, což zvyšuje pravděpodobnost správně rozpoznané promluvy.

5.2.3 Syntéza řeči

O syntézu řeči se stará cloudový server SpeechTech TTS. Pro aplikaci byl nastaven hlas pod označením Alena 2.10. Tento hlas je úplně prvním hlasem, který vznikl, je vytvořen z 5 000 nahrávek, které mají bez pauz celkem 9,73 hodin. Hlas je tvořen spojováním difonů do sebe, což umožňuje plynulou a přirozenou návaznost, jedná se o tzv. metodu unit selection (výběr jednotek). Verze hlasu je FULL.33.

5.2.4 Připojení

SpeechCloud.js je javascriptová knihovna, která se stará o veškeré náležitosti – připojení k serveru, stream audia, metody a události.

Aktuální verze knihovny je k datu 19.4.2016, 12:07:01 GMT.

Prvním krokem je inicializovat připojení k serveru pomocí WebSocketu. Informace, které zasíláme, jsou URI adresa s parametry pro připojení a název HTML5 bloku, který obstarává přehrávání audia. URI adresa obsahuje následující parametry

- **client_id**: ID klienta – slouží pro logování
- **client_wss**: websocket server
- **sip_uri**: URI adresa SIP serveru
- **sip_username**: uživatelské jméno pro SIP
- **sip_password**: heslo pro SIP
- **sip_wss**: URI adresa serveru pro komunikaci skrze WebSocket

Blok se v HTML jmenuje audioout.

```
sc = new SpeechCloud({
  uri: https://hydra4.kky.zcu.cz:9443/v1/speechcloud/edu-nedved,
  tts: "#audioout"
});
```

Propojení SIP a WebSocketu (sc) se provede zavoláním funkce *sc.init()*.

Následně proběhne registrace na SIP ústřednu a spuštění protokolu „SIP over WebSocket“, která obstarává komunikaci s webovým prohlížečem. Veškeré úkony spojené s přenosem hlasu obstarává JavaScriptová knihovna SIP.js.
Po úspěšné inicializaci dojde k otevření WebSocket protokolu a navázání spojení, při kterém spolu již mohou klient a server vzájemně komunikovat.

5.2.5 Ovládání aplikace

5.2.5.1 Metody

Metodou se rozumí požadavek klienta na server o danou službu. V aplikaci se využívají 3 metody:

asr_recognize

Zapne rozpoznávání řeči a audio se začne přenášet z webového prohlížeče na server.

asr_pause

Vypne rozpoznávač řeči.

tts_synthesize

Provede syntézu požadovaného textu, který se zašle jako parametr funkce.

5.2.5.2 Události

Události jsou odpověďmi serveru na požadovanou metodu.

asr_ready

Rozpoznávač řeči je připraven přijímat streamované audio.

asr_set_grammar_ok

Byla nastavena gramatika a aplikace je tak připravena rozpoznávat příchozí audio.

asr_signal

Poskytuje informace o signálu, který na server putuje. Poskytuje informaci, zda se jedná či nejedná o řeč a jak velkou úroveň energie má příchozí signál. Při slabém či příliš silném signálu je velmi pravděpodobné, že bude text nesprávně rozpoznán.

asr_result

Vrací výsledek rozpoznávání, pomocí JSON ho můžeme převést na objekt s následujícími parametry:

- word_1best: rozpoznaná promluva
- type: typ události, kterou server vrací
- result: rozpoznaná promluva
- partial_result: informace o tom, zda se jedná o celou promluvu

Pro naši konkrétní aplikaci, která využívá ESGF gramatiku jsou hodnoty word1_best a result totožné.

tts_done

Informace o tom, že celý požadovaný text byl syntetizován. Praktické využití této funkce může být například v případě, že chceme provést danou akci až po vyřčení promluvy.

5.2.6 Logování

SpeechCloud veškeré úkony loguje. Při připojení na server vytvoří jedinečné session ID, pomocí kterého si lze zobrazit záznam komunikace s klientem. Výhodou je ukládání audio nahrávek, ke kterým pak máme později přístup. Tato skutečnost bude kritická pro pozdější testování, při kterém tak lépe budeme moci zjistit, zda byla případná chyba v gramatice, rozpoznávači či samotné aplikaci.

6 Návrh aplikace

Většina hlasových dialogových systémů se spoléhá pouze na hlasový vstup. Uživatel tak nikdy neví, co se právě odehrává v samotné aplikaci. Namísto toho je mu znám pouze prezentovaný hlasový výstup. Právě tato skutečnost je obrovskou slabinou hlasových systémů, jelikož si nikdy nemůžeme být stoprocentně jisti, zda jsme rozpoznali požadavek správně. Dostávat se pak z nežádoucího stavu může být někdy velmi složité, hlavně v případech, kdy je vstup pochopen několikrát po sobě špatně, uživatel ztrácí s aplikací trpělivost a není pak schopen řádně formulovat své požadavky. V takovéto situaci pak většinou dochází k restartování promluvy a uživatel pak znovu formuluje svůj problém.

Čím více funkcí poté systém má, tím je větší pravděpodobnost, že se dostaneme do nežádoucího stavu. Právě toto byl problém například některých hlasových dialogových systémů vytvořených na katedře kybernetiky (s využitím komponent SpeechTech) v minulých letech. Příliš mnoho funkcí a absence grafického rozhraní vedlo k tomu, že původně navrhovaná aplikace byla v realitě nepoužitelná.

Cílem práce je vytvoření funkční demo aplikace, která dokáže aktivně reagovat na uživatelský vstup a umožňuje alternativní způsob ovládání kromě hlasu pro případ, že se uživatel dostane do nechtěného stavu. Pokud aplikace rozpozná pouze část promluvy a bude pro provedení daného úkonu potřebovat dodatečné informace, dá to uživateli najevo nejen hlasem, ale i změnou nabídky na obrazovce.

Před samotným ovládáním proběhne ověření uživatele na základě referenčních nahrávek uložených v systému. Takovýto způsob ověřování by mohl nahradit ruční zadávání kódu k alarmu či odemykání dveří. Stejně jako pro samotný dialogový systém je důležité, aby i ověřování uživatele nabízelo alternativní řešení v případě, že by autentifikace selhala a uživatel by nebyl vpuštěn.

6.1 Funkce a vlastnosti

Aplikace nabídne čtyři základní funkce. Půjde vždy pouze o zapnutí a vypnutí či otevření a uzavření.

6.1.1 Funkce aplikace

- Ovládání osvětlení
- Ovládání klimatizace
- Ovládání žaluzií
- Ovládání dveří
- Ovládání audia

Každá funkce kromě audia bude mít přiřazené konkrétní místnosti, v kterých můžeme vykonávat požadované úkony. Veškeré úkony týkající se rozpoznání uživatelovy akce se budou vyhodnocovat přímo ve webovém prohlížeči. V momentě, kdy si bude aplikace jistá, jakou akci uživatel chce zvolit, kontaktuje server, který danou funkci zavolá. Webový prohlížeč tak bude komunikovat se dvěma servery – SpeechCloud a server běžící na lokálním zařízení. Na lokálním serveru dále poběží aplikace pro autentifikaci uživatele. Tímto způsobem docílíme zabezpečení aplikace, jelikož pokud nebude uživatel na serveru ověřen, nebudou mu ani zpřístupněny žádné funkce. Pokud bychom nechali ovládání na straně webového prohlížeče, zdatný uživatel by mohl pomocí JavaScriptu obejít bezpečnostní protokoly a dostat se tak přes přihlášení uživatele. Takovýto krok by nebyl příliš obtížný, jelikož JavaScript je viditelný ve zdrojovém kódu stránky.

6.1.2 Stav aplikace

Aplikace se v každém okamžiku nachází v určitém stavu. Jak již bylo zmíněno výše, máme dvě různé akce – vypnout a zapnout. Dále máme 5 různých funkcí, které můžeme ovládat a pro každou funkci je možnost ovládat několik místností. Definujme tak stav (**stav**) aplikace

```
věc: 'světlo' / ' klimatizace' / 'žaluzie' / 'dveře' / 'audio',
místnost: 'kuchyně' / 'obývák' / 'ložnice' / 'garáž' /
              'zadní vchod' / 'hlavní vchod',
akce: 'on' / 'off'
```

Dále definujme objekt (**domacnost)**, který uchovává aktuální informace o všech funkcích a místnostech

```
'svetlo' : {
                    "kuchyne":"off",
                    "obyvak":"off",
                    "loznice":"off",
                },
'klimatizace' : {
                    "kuchyne":"off",
                    "obyvak":"off",
                    "loznice":"off",
                },
'zaluzie' : {
                    "kuchyne":"off",
                    "obyvak":"off",
                    "loznice":"off",
                },
'dvere' : {
                    "hlavni":"off",
                    "garaz":"off",
                    "zadni":"off",
                },
'audio': "off"
```

Při rozpoznání uživatelovy akce aktualizujeme stav objektu a zapneme / vypneme tak daný spotřebič v dané místnosti.

Zvolený objekt a funkce v něm jsou pouze pro ukázkové účely, a proto nemusí logicky odpovídat funkcím v reálném životě – např. světlo by mohlo být i pro další místnost – garáž. Objekt s funkcemi je tvořen tak, aby se bez větších úprav kódu daly přidávat či ubírat další funkcionality aplikace.

Aplikace je schopna si udržovat svůj stav i při odpojení uživatele, což je vhodné v případě, že by webový prohlížeč ztratil komunikaci se serverem. Pro větší bezpečnost je při každém odpojení zaslána aktuální konfigurace ze serveru a celé grafické rozhraní je tak vždy restartováno. To zamezí jakýmkoli úpravám kódu na straně klienta.

6.2 Server na lokálním zařízení

Jak již bylo zmíněno výše, pro větší bezpečnost se provádí ověření uživatele na straně serveru a ne na straně webového prohlížeče – klienta. Dalším důvodem pro vytvoření serveru byl fakt, že autentifikace uživatele je konzolová aplikace a je tudíž zapotřebí propojit webový prohlížeč se systémem, na kterém aplikaci bude možno spustit.

Dalším podnětem pro vytvoření serveru byl fakt, že zařízení, na kterém poběží webový prohlížeč, bude nejspíše zařízení typu tablet, které nebude moci dále komunikovat s elektrickou sítí uvnitř domu. O tuto komunikaci se bude tedy muset starat server, který bude ovládat jednotlivé spotřebiče.

Server, který se postará o výše zmíněné úkony, byl naprogramován v jazyce Python.

6.2.1 Tornado Framework

Tornado Framework je webový server, který skrze WebSocket protokol umožňuje komunikaci s webovým prohlížečem. Jedná se o open source program, který lze stáhnout z oficiálních stránek[3].

6.2.1.1 Instalace

Instalace probíhá stejně jako s ostatními knihovnami určenými pro Python. Stačí rozbalit archiv a spustit v dané složce příkazový řádek s právy administrátora. Následně příkazem

```
python setup.py install
```

nainstalujeme potřebné knihovny a můžeme již spustit vlastní server.

[3] Dostupné na stránkách http://www.tornadoweb.org/

6.2.1.2 Inicializace

Server bude komunikovat skrze TCP port 8888 a bude spuštěn na adrese zařízení 127.0.0.1, neboli localhostu.

Inicializace WebSocket serveru

```
application = tornado.web.Application([(r'/ws', WSHandler),])
```

Jako parametr se nastavuje třída WSHandler, která zpracovává veškeré úkony mezi serverem a klientem.

Nastavení komunikačního portu a adresy serveru

```
application.listen(8888, address='localhost')
```

Spuštění serveru

```
tornado.ioloop.IOLoop.instance().start()
```

6.2.2 Raspberry PI

Raspberry PI je miniaturní počítač přibližně o velikosti tabletu, který se může pyšnit poměrně dobrým výkonem. Počítač si poradí s různými operačními systémy a lze tak kromě Linuxu instalovat například i Windows. Počítač disponuje HDMI na připojení grafického rozhraní, výstupem a USB a vstupy na klávesnici či myš. Jednotlivé verze se dále velmi liší, některé mají např. i VGA výstup či síťovou kartu. Zařízení lze tak použít kromě ovládání určitých zařízení také k samotným výpočetním výkonům [22].

6.2.2.1 Domácí automatizace

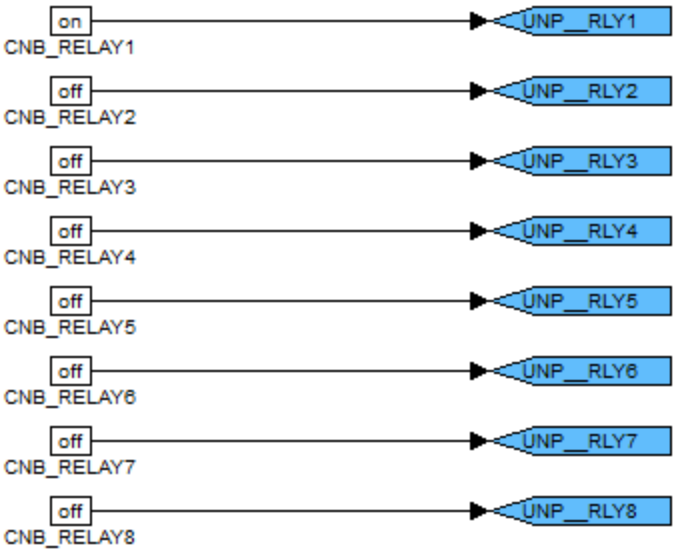
Nás bude zajímat konkrétní verze Raspberry PI s názvem UniPi, které je znázorněno na obrázku 6.1. Zařízení disponuje kromě výše zmíněných rozhraní také vstupy, které umožňují napojení elektrické sítě na základní desku pomocí konektorů. Toto rozhraní otevírá dveře aplikaci v domácí automatizaci, a to v plném rozsahu (viz část Inteligentní domácnost), a to hlavně proto, že kromě zapínání a vypínání deska obsahuje například i teplotní čidla.

Samotné řízení obstarává řídicí systém REX, na jehož vývoji se podílela katedra kybernetiky. Komunikace se zařízením probíhá pomocí HTTP protokolu skrz síťový kabel. Požadavky obdržené od klienta ohledně ovládání konkrétního spotřebiče se tak zasílají na zařízení UniPi, které daný úkon provede [23].

Obr. 6. 1 Raspberry UniPi ze stránky unipy.technology

6.2.2.2 Ovládání

Systém přijímá požadavky pomocí protokolu POST ve formě JSON. Požadavek ve formě vypnout / zapnout je zasílán na adresu, která odkazuje na konkrétní relé. Deska dokáže obsluhovat celkem 8 spotřebičů, jednotlivá relé jsou znázorněna na obrázku 6.2, na kterém jsou všechny kromě prvního vypnuty.



Obr. 6. 2 Relé na UniPi

Pokud chceme například vypnout první relé, požadavek bude zaslán na následující adresu,

```
http://192.168.1.100:8008/api/tasks/blink_task/CNB_RELAY1:YCN
```

kde YCN představuje proměnnou, která nabývá hodnot on / off. 192.16.8.1.100 je adresa vnitřní sítě, která byla vytvořena mezi PC a deskou UniPi. Komunikace probíhá na portu 8008.

6.2.3 Třídy a Funkce

O veškeré úkony serveru se starají celkem tři třídy:

- **WSHandler** – WebSocket komunikace
- **OvereniUzivatele** – autentifikace uživatele
- **Dum** – ovládání domu

6.2.3.1 WSHandler

Jak již bylo popsáno v části o Webových aplikacích, WebSocket komunikuje pomocí tří hlavních událostí – navázání spojení, přijetí zprávy a ukončení spojení. Vzhledem k vyšší bezpečnosti Tornado obsahuje funkci check_origin, která může zabránit přístupu z nechtěné adresy. Lze tak nastavit určité povolené domény, ze kterých lze na server přistupovat. Jelikož se ale jedná pouze o demo aplikaci, která navíc běží na lokálním zařízení, nemusíme nikterak omezovat přístup a funkce tak bude vždy vracet hodnotu True.

```
def check_origin(self, origin):
   return True
```

Většina funkcí si předává parametr self, který obsahuje instanci třídy, pro kterou je metoda volána, a lze tak jednoduše pracovat s proměnnými z jiných tříd a funkcí.

Funkce open

Po úspěšném připojení zařízení vypíše na serverové straně informaci o připojení a také odešle stejnou zprávu zpátky prohlížeči. Pokud nedojde k přijetí zprávy, webová stránka se pokusí znovu načíst, jelikož došlo k chybné prvotní inicializaci. Dále se ve funkci inicializují zbylé dvě třídy.

```
def open(self):
     print 'Uživatel připojen.'
     self.d = Dum();
     self.oV = OvereniUzivatele();
```

Funkce on_message

Funkce zpracovává přijatou zprávu ze strany klienta. Jedná se o hlavní funkci programu, která dále obsluhuje další metody a funkce a zajišťuje tak správnou funkčnost systému.

```
def on_message(self, message)
```

Samotný parametr message již obsahuje obdrženou zprávu ze strany klienta a lze tak jednoduše zpracovat přijatý text.

Ověření uživatele

Přihlášení (ověření) uživatele může proběhnout dvěma způsoby – rozpoznáním řečníka či rozpoznáním hesla. V prvním případě klient odešle serveru zprávu, ve které dává pokyn, aby si stáhl audio soubor. Pokud je uživatel rozpoznán, zašle server aktuální konfiguraci (nastavení spotřebičů). V případě, že uživatel není rozpoznán, zašle negativní zprávu.

```
if (message[:5]=="verif"):
    self.oV.ziskejWav(message[6:])
    vys = self.oV.cyklus("users/file.wav");
    if (vys=="ne"): self.write_message("ne")
    else:
         pripojen = True
         self.write_message("ok" + json.dumps(self.d.domacnost))
```

Pokud se nedaří ověřit dle hlasu nebo pokud uživatel preferuje ověření pomocí hesla, server prohledá uložená hesla. V případě, že se heslo shoduje či neshoduje, postupuje aplikace stejně jako v předchozím případě.

```
elif (message[:5]=="heslo"):
    vys = self.oV.hledejHeslo(message[6:])
    if (vys=="ne"): self.write_message("ne")
    else:
         pripojen = True
         self.write_message("ok" + json.dumps(self.d.domacnost))
```

Pokud uživatel aktualizoval stránku, ale neodhlásil se, server aktualizuje konfiguraci a zašle mu ji zpět.

```
if message=="conf":
    self.write_message("ok" + json.dumps(self.d.domacnost))
```

Ovládání

Po přihlášení z klienta na server putují pokyny pro zapnutí či vypnutí daného spotřebiče v daném místě.

Vypnutí všech spotřebičů

Pokud chceme vypnout všechny spotřebiče, server načte počáteční konfiguraci, při které je vše vypnuto, a zašle ji zpět na klienta stejným způsobem, jako když se klient přihlašuje. O této skutečnosti informuje i zprávou na straně serveru.

```
if (message == "all_off"):
  print "Vypínám všechna zařízení"
  self.d.domacnost = self.d.konfigurace()
  self.write_message("ok" + json.dumps(self.d.konfigurace()))
```

Odhlášení ze systému

V případě, že uživatel chce vypnout spotřebiče a odhlásit se, nastaví se defaultní konfigurace na všech spotřebičích a uloží se do souboru. Následně se přeruší spojení s klientem, na kterém se nastaví přihlašovací obrazovka.

```
elif (message=="out"):
    self.d.domacnost = self.d.konfigurace()
    pickle.dump( self.d.domacnost, open( "stav.p", "wb" ) )
    pripojen = False
    self.write_message("nepripojen")
    self.close()
```

Další komunikace mezi klientem a serverem je již jen požadavek na vypnutí či zapnutí konkrétního spotřebiče. Jelikož se jedná o pouhé tři informace, není zapotřebí využívat JSON a text stačí rozdělit pomocí bílých znaků. Je třeba dávat pozor, zda se nejedná o ovládání audia, které nemá definovánu místnost. Server opět informuje pomocí výstupu o konané akci a výsledek následně zašle zpět na klienta. Také se zavolá funkce, která se postará o danou akci.

```
data = re.split(" ",message[2:])
if data[0] == "audio":
    self.d.domacnost[data[0]] = data[1]
else:
    self.d.domacnost[data[0]][data[1]] = data[2]
    self.d.vysliPrikaz(data)
self.write_message('done')
```

Například pokud bychom chtěli vypnout světlo v kuchyni, server obdrží zprávu **„svetlo kuchyne on“**

Po nařezání skrze bílé znaky získáme

```
data[0] = svetlo
data[1] = kuchyne
data[2] = on
```

V objektu pak budeme nastavovat danou hodnotu

```
self.d.domacnost[svetlo][kuchyne] = on
```

Funkce on_close

Při odpojení klienta od serveru uložíme aktuální instanci do proměnné **stav.p** pomocí modulu pickle, který umožňuje ukládat objekty.

```
def on_close(self):
   print 'Přípojení ukončeno\n'
   pickle.dump( self.d.domacnost, open( "stav.p", "wb" ) )
```

6.2.3.2 OvereniUzivatele

Třída je volána při hlasovém přihlašování uživatele.

Inicializace

Prvotní inicializace přiřadí k uživatelům pořízené audio nahrávky a jako alternativu určené číselné heslo. Nastaví také jako pracovní složku hlavní složku, ve které je aplikace uložena pro snadnější přístup k jednotlivým souborům. V inicializaci lze snadno přidávat či ubírat uživatele.

```
def __init__(self):
    os.chdir("cesta")
    self.uziv = {}
    self.uziv['usr1'] = [ ["wav1","wav2"],"heslo"];
    self.uziv['usr2'] = [ ["wav1","wav2"],"heslo"];
```

Funkce ziskejWav

Funkce je volána při přihlašování uživatele. Audio nahrávka je získána z logu aplikace, pomocí ID, které je předáno z klienta na server pomocí WebSocketu. V logu se vyhledá část „audio records“, ve které jsou nahrávky rozděleny na dva typy – ASR a TTS. TTS je syntetizovaná promluva, která byla jako výstup aplikace, ASR je pak záznam uživatelovy promluvy. Nové záznamy jsou přidávány na konec seznamu, proto vždy aktuální nahrávku získáme tak, že budeme prohledávat pořízené nahrávky od konce, což je díky JSON formátu velmi zjednodušeno. Při nalezení konkrétní nahrávky soubor stáhneme pomocí knihovny urllib a následně uložíme do složky k uživatelským nahrávkám, pod názvem file16.wav.

```
def ziskejWav(self, uri):
  url = "https://cak.zcu.cz:9443/v1/session/"+uri+"?format=json"
  response = urllib.urlopen(url)
  data = json.loads(response.read())
  for i in range(len(data['sessions'][0]['audio_records']),-1,-1):
    try:
      if data['sessions'][0]['audio_records'][i]['type'] =='asr':
          print data['sessions'][0]['audio_records'][i]['uri']
          break
      except:
          continue
  wav = data['sessions'][0]['audio_records'][i]['uri']
  urllib.urlretrieve(wav,'users/file16.wav')
```

Část logu

```
"sessions": [
  {
  "audio_records": [
   {
    "filename": "log/20160411-174822-1.wav",
    "tstamp": "2016-04-11T17:48:22",
    "type": "asr",
    "uri":
     "https://cak.zcu.cz:9443/v1/session/bhzKuR7NoDh4Eusss7aJYD/
         audio_records/570be36ae90fbf00622c4d1c"
  },
  {
    "filename": "tts_prompt_1.wav",
    "transcript": "Zadejte pros\u00edm va\u0161e heslo",
    "tstamp": "2016-04-11T17:48:28",
    "tts_stop_tstamp": null,
    "type": "tts",
    "uri":
     "https://cak.zcu.cz:9443/v1/session/bhzKuR7NoDh4Eusss7aJYD/
         audio_records/570be36ee90fbf00622c4d20"
  }
]
```

Funkce hledejUzivatele

Po stažení nahrávky je soubor předán další funkci, která projde všechny uživatele nacházející se v systému a v případě shody navrátí pozitivní výsledek. Pokud není uživatel nalezen, je to nejspíše zapříčiněno špatnou nahrávkou (nesprávná promluva, nekvalitní záznam,…) a nahrávání tak probíhá znovu. Aplikace opět vypisuje průběžné výsledky na straně serveru.

```
def hledejUzivatele(self,vstup):

   for uz in self.uziv:
      if self.overen == True: break
      reference = "";
      for wav in self.uziv[uz][0]:
          reference = reference + " " + wav
      output = self.autentifikace(vstup, reference);
      print uz +" : ", output
      if (output[0] > 75 and output[1] > 60):
         return "ok"
   return "ne"
```

Funkce hledejHeslo

Velmi podobně jako předchozí funkce funguje i funkce hledejHeslo. Namísto porovnávání nahrávek však porovnává uživatelem zadané heslo ve formuláři a uložené heslo v systému.

```
def hledejHeslo(self,heslo):
    for key in self.uziv:
          if self.uziv[key][1] == heslo:
          return "ok"
    return "ne"
```

Funkce autentifikace

Funkce je volána z funkce hledejUzivatele. Program se volá stejným způsobem, jakým to bylo popsáno v části Komponenty a technologie. Výstup aplikace se zpracuje za pomocí regulérních výrazů. Nejdříve vstup rozdělíme a dále v něm hledáme všechna čísla v exponenciálním tvaru, které pomocí funkce **format** můžeme převést na číselný tvar typu float (např. 1.25 e^{-003} převedeme na 0.00125).

```
def autentifikace(self, vstup, user):
    import subprocess
    out = list()
    args = "password.exe VAD-EN.bin " + vstup + " " + user
    output = subprocess.Popen(args,
           stdout=subprocess.PIPE).communicate()[0]
    output = re.split("\r\n",output)
    out1 = float(re.findall("[0-9]+\.[0-9]+e-[0-9]+",
           output[1])[0]) * 100
    out2 = float(re.findall("[0-9]+\.[0-9]+e-[0-9]+",
           output[2])[0]) * 100
    out.append(float(format(out1,'f')))
    out.append(float(format(out2,'f')))
    return out
```

6.2.3.3 Dum

Třída ovládající samotnou domácnost.

Inicializace

Po připojení uživatele získává aktuální konfiguraci ze souboru, pokud by došlo k chybě a soubor by se nepodařilo nahrát, načte místo toho základní konfiguraci.

```
def __init__(self):
   try:
      self.domacnost = pickle.load(open( "stav.p", "rb" ) );
   except:
      self.domacnost = self.konfigurace()
```

Funkce konfigurace

Nahrává základní konfiguraci domu. Funkce je volána, pokud není dostupný soubor s již uloženou konfigurací či v případě, že přijde příkaz na vypnutí celého systému.

```
def konfigurace(self):
  conf = {
     'svetlo' :
          {"kuchyne":"off", "obyvak":"off", "loznice":"off" },
     'klimatizace' :
          { "kuchyne":"off", "obyvak":"off", "loznice":"off },
     'zaluzie' :
          { "kuchyne":"off", "obyvak":"off", "loznice":"off },
     'dvere' :
          {"hlavni":"off", "garaz":"off", "zadni":"off },
     'audio': "off"
     }
  return conf
```

Funkce vysliPrikaz

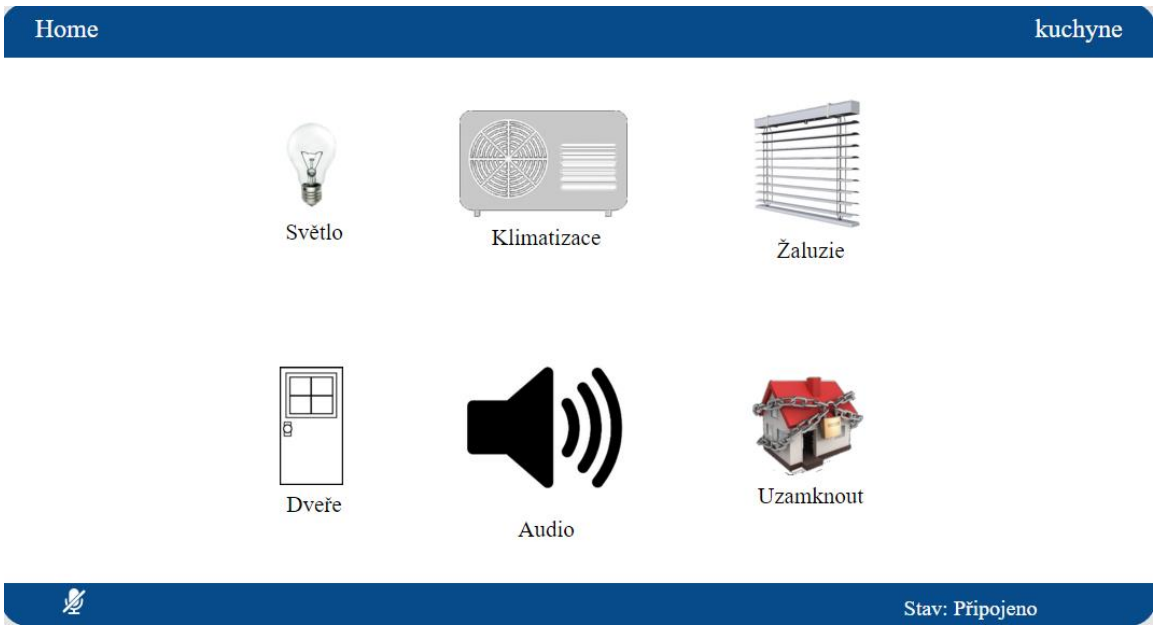
Funkce komunikuje s UniPi skrze HTTP pomocí POST zpráv. Funkce je volána z hlavní metody při přijmu zprávy. Aplikace na druhé straně čeká na parametr *v*, který zašleme pomocí JSON, aby zprávu správně přečetla, je třeba nastavit její hlavičku (headers).

```
def vysliPrikaz(self,data):
  if data[0] == "svetlo":
    if data[1] == "kuchyne": rele = "CNB_RELAY1"
    elif data[1] == "obyvak": rele = "CNB_RELAY2"
    else: rele = "CNB_RELAY3"
  elif data[0] == "klimatizace":
    if data[1] == "kuchyne": rele = "CNB_RELAY4"
    elif data[1] == "obyvak": rele = "CNB_RELAY5"
    else: rele = "CNB_RELAY6"
  url =
   'http://192.168.1.100:8008/api/tasks/blink_task/'+rele+':YCN'
  if data[2] == "on": data = { 'v':'true' }
  else: data = { 'v':'false' }
  headers = {'Content-type': 'application/json',
             'Accept': 'application/json'}
  r = requests.post(url, data=json.dumps(data),headers=headers)
```

6.3 GUI

Veškerý obsah stránky je popsán pouze v jednom dokumentu – index.html. Aplikace dále využívá další 3 dokumenty – styl.css, ve kterém jsou uloženy grafické prvky, gui.js, který se stará převážně o JavaScriptové funkce vyvolané skrze kliknutí, a speech.js, který obsluhuje převážně JavaScriptové funkce pomocí hlasového vstupu. Aplikace také využívá knihovnu jQuery verze 1.10.1.min a již dříve zmíněnou knihovnu SpeechCloud.js.

Grafické rozhraní je znázorněno na obrázku 6.3. Obrázky byly staženy ze serveru Pixabay[4].



Obr. 6. 3 GUI aplikace

6.3.1 HTML

6.3.1.1 Hlavička dokumentu

Hlavička definuje kódování HTML dokumentu, dále název stránky, který se zobrazí v prohlížeči, a odkazy na dokumenty nutné pro funkčnost aplikace

```
<head>
    <title>Inteligentní domácnost s hlasovým ovládáním</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="styl.css" />
    <script src="jquery-1.10.1.min.js"></script>
    <script src="https://cak.zcu.cz:9444/speechcloud.js"></script>
    <script src="speech.js"></script>
    <script src="gui.js"></script>
</head>
```

6.3.1.2 Tělo dokumentu

Tělo dokumentu je základní část každé webové aplikace. Pro co nejrychlejší přechody mezi stránkami byl zvolen způsob zobrazování jednotlivých elementů na stránce pomocí JavaScriptu a CSS stylů. Na jednom místě tak definujeme veškerý obsah, uživateli se však zobrazí pouze část z něj. Až zavoláním funkce a změnou stylu dosáhneme toho, že část

[4] Pixabay.com poskytuje k libovolnému užití (pod licencí Creative Commons CC0) obrázky a fotografie ve vysoké kvalitě

stránky se uživateli skryje a jiná zase zobrazí. Veškeré změny probíhají přímo ve webovém prohlížeči bez potřeby načítání dalších stránek či dat.

Audio

Element audio slouží k přehrávání mediálního obsahu. Běžně je využíván pro přehrávání písniček přímo na webových stránkách, v naší aplikaci slouží taktéž jako výstup syntetizéru, kterému se informace o jeho ID názvu předá při inicializaci spojení.

```
<audio id="audioout"></audio>
```

Obsah

Veškerý obsah stránky je definován elementem,

```
<div id="content"> veškerý obsah </div>
```

ve kterém se dále definuje záhlaví (navigace), obsah a zápatí (stavový řádek) aplikace. V aplikaci nastavujeme kritické velikosti v procentech, díky čemuž se nám zobrazí na všech zařízeních stránky stejně – jedná se o tzv. responzivní design.

CSS

```
#content {
   margin: auto;
   width:85%; height:80%;
   display:block; clear:both;
   overflow: auto; position:relative;
   justify-content: center;
   align-items: center;
}
```

Menu

V menu na levé straně obsahu, se uživateli zobrazuje aktuální stav, v jakém se aplikace nachází – například pokud chce ovládat světlo, bude v menu v sekci Home / svetlo (viz obrázek 6.4). Home funguje i jako tlačítko a při kliknutí na něj se dostaneme do hlavní nabídky. V pravé části se ukazuje místnost, ve které je umístěna dotyková obrazovka. Defaultní hodnota byla nastavena kuchyně (zařízení je tedy zabudováno v kuchyni).

Obr. 6. 4 Menu aplikace

```
<div id="navigace">
  <ul>
      <li> <a href="#" onClick="zmenMenu('menu','polozky');">
                Home</a></li>
      <li id="nav"> </li>
  </ul>
  <a href="#"><span style="float:right;margin-right:3%;"
          id="def-mistnost">  </span> </a>
</div>
```

Abychom dali najevo, že se jedná o interaktivní obsah, při najetí myši zbarvíme text na zeleno. Dále definujeme velikost písma a odsazení textu. Díky vlastnosti border-radius vytvoříme pro rozhraní zakulacené rohy, které mají přívětivější vzhled, než kdybychom měli pouhý obdélník.

CSS

```
#navigace {
   background-color:#084B8A;
   color:#ffffff;
   border-top-left-radius:25px;
   border-top-right-radius:25px;
   padding-left:2%;
   padding-top:1%;
   font-size: 24px;height:40px;
}

#navigace ul {
    list-style-type: none;
    position:relative;
    margin: 0; padding: 0;
    display:none;
}

#navigace li {
    float: left;padding-left:1%;
}

#navigace li a:hover {
    color: green;
}

#navigace a {
    color:#ffffff;
    text-decoration:none;
}

#navigace a:hover {
    color: green;
}
```

Po navigaci definujeme již samotný obsah, který se nachází na bílém pozadí. Veškerý obsah je zarovnán do středu a má omezení velikosti, aby nedošlo k přetékání prvků do sousedních.

CSS

```
#obsah {
    background-color:#ffffff;
    position:relative; height:70%;
    text-align:center; vertical-align: middle;
    font-size: 21px;
    display:block;  clear:both;
    overflow: auto;
    justify-content: center;
    align-items: center;
}
```

Přihlášení

Přihlášení má 2 kroky, které jsou znázorněny na obrázku 6.5. V prvním se z režimu offline přepne do režimu připraveného, v kterém je možnost zadat heslo, které se následně zašle na server. Pomocí třídy hide schováváme, z důvodu jednodušší manipulace, nežádoucí elementy. S elementy dále pracujeme hlavně díky jejich ID.

Obr. 6. 5 Přihlášení do aplikace

```
<div id="login" style="display:none;" class="hide">
    <a href='#'><img src='images/login.png' width='40%'>
        <br />Přihlášení</a>
</div>
<div id="login2" style="display:none;" class="hide">
    <input type="text" id="login5" value="heslo">
        <br /> <a href="#" id="log">Odeslat</a>
</div>
```

Hlavní nabídka

Hlavní nabídka má ID položky a obsahuje veškeré funkce, které aplikace nabízí. Nabídka je jednoduchá a kromě textového popisu nabízí i ilustraci (viz obrázek 6.6). Při kliknutí na danou položku je volána funkce, která se postará o zobrazení další nabídky na základě informací, které jsou při volání funkci předávány. Například při kliknutí na světlo je volána funkce

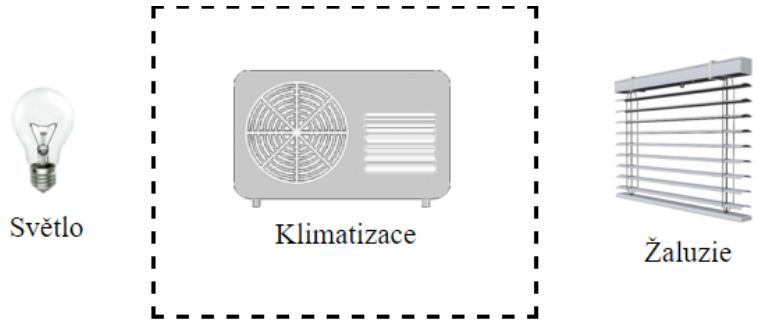
```
zmenMenu('polozky', 'svetlo');
```

na základě čehož se následně skryje nabídka položky a zobrazí nabídka pro světlo. Zobrazení všech prvků je provedeno pomocí tabulky, která se postará o jednotný styl menu. Tabulku označuje párový element <table>, řádek se značí <tr> a sloupec <td>. U obou se jedná o párové elementy.

Obr. 6. 6 Obsah aplikace

```
<div id="polozky" style="display:none;">
 <table>
  <tr>
    <td onclick="zmenMenu('polozky', 'svetlo');">
     <img src="images/svetlo.png" <br /> Světlo </td>
    <td onclick="zmenMenu('polozky', 'klimatizace');">
     <img src="images/klimatizace.jpg"> <br/>Klimatizace</td>
    <td onclick="zmenMenu('polozky', 'zaluzie');">
     <img src="images/zaluzie.jpg" > <br /> Žaluzie</td>
  </tr> <tr>
    <td onclick="zmenMenu('polozky', 'dvere');">
     <img src="images/dvere.png" > <br /> Dveře</td>
    <td onclick="zmenMenu('polozky', 'audio');">
     <img src="images/audio.jpg" > <br /> Audio</td>
    <td onclick="zmenMenu('polozky', 'zamknout');">
     <img src="images/alarm.png" > <br /> Uzamknout</td>
  </tr>
 </table>
</div>
```

Vzhledem k tomu, že jako odkaz funguje celý prvek tabulky, je třeba uživatele o tom patřičně informovat. Při najetí myší nad daný prvek se objeví čárkovaný rámeček, který je znázorněn na obrázku 6.7. Dále nastavíme velikost obrázků, aby se všechny správně zobrazovaly.



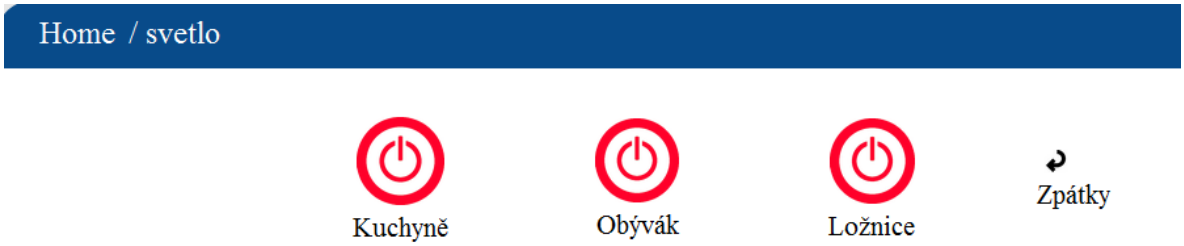
Obr. 6. 7 Zobrazení rámečku po najetí na element

CSS

```
#obsah table {
      margin:0 auto;
      width: 60%;
}

#obsah table td {
      position:relative;
      padding:5%;
      border: dashed;
      border-color: #ffffff;
}

#obsah table td:hover {
      border: dashed;
}

#obsah table img, #table td
img {
      width:60%;
}
```

Položky světlo, klimatizace a žaluzie

Jednotlivé položky v práci není potřeba vypisovat, jelikož jsou všechny stejně konstruované – jsou definovány ve stejných místnostech. Jejich podobu si demonstrujeme na světle. Jediná odlišnost těchto položek je u volané funkce, kde je jako první parametr právě ovládaný spotřebič (světlo, klimatizace, žaluzie). Každý spotřebič má v dané místnosti dva stavy – vypnuto či zapnuto. Stav definujeme pro jednotlivé místnosti – kuchyně, obývák, ložnice. Další položkou nabídky je návrat zpátky na hlavní menu (viz obrázek 6.8).

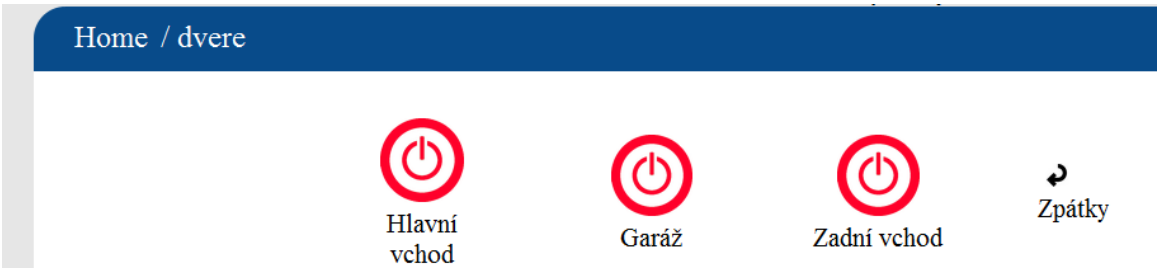


Obr. 6. 8 Nabídka světlo

```
<div id="svetlo" style="display:none;" class="hide">
      <table>
            <tr>
               <td onclick="zmenHodnotuGUI('svetlo','kuchyne');">
                  <img src="images/svetlo.png" id="svetlo-kuchyne">
                  <br /> Kuchyně </td>
               <td onclick="zmenHodnotuGUI('svetlo','obyvak');">
                  <img src="images/svetlo.png" id="svetlo-obyvak">
                  <br /> Obývák</td>
               <td onclick="zmenHodnotuGUI('svetlo','loznice');">
                  <img src="images/svetlo.png" id="svetlo-loznice">
                  <br /> Ložnice</td>
               <td onClick="zmenMenu('menu','polozky');">
                  <img src="images/back.png"> <br /> Zpátky</td>
            </tr>
      </table>
</div>
```

Položka dveře

Dveře mají podobnou nabídku jako předchozí 3 položky, liší se však místnostmi, které zahrnují (viz obrázek 6.9).



Obr. 6. 9 Nabídka dveře

```
<div id="dvere" style="display:none;" class="hide">
   <table> <tr>
          <td onclick="zmenHodnotuGUI('dvere','hlavni');">
               <img src="images/dvere.png" id="dvere-hlavni">
               <br /> Hlavní vchod </td>
          <td onclick="zmenHodnotuGUI('dvere','garaz');">
               <img src="images/dvere.png" id="dvere-garaz">
               <br /> Garáž</td>
          <td onclick="zmenHodnotuGUI('dvere','zadni');">
               <img src="images/dvere.png" id="dvere-zadni">
               <br /> Zadní vchod</td>
          <td onClick="zmenMenu('menu','polozky');">
               <img src="images/back.png">
               <br /> Zpátky</td>
    </tr> </table>
</div>
```

Položka audio

Na rozdíl od předchozích nabídek se ovládání hudby značně liší. Hudba se nedefinuje pro danou místnost, ale pro celou domácnost. Z toho důvodu je v nabídce pouze vypnout a zapnout. Při zapnutí se navíc v nabídce objeví přehrávač hudby deklarovaný přímo v HTML5, kterému můžeme i nastavit počáteční hlasitost, aby hudba nehrála příliš nahlas. Přehrávač je znázorněn na obrázku 6.10. Hudba hraje, dokud se aplikaci nepředá pokyn k vypnutí, máme tak během přehrávání hudby na pozadí přístup k celému systému. Jako zdroj bylo použito univerzitní rádio Bomba.

Obr. 6.10 Nabídka audio

```
<div id="audio" style="display:none;" class="hide">
 <table>
  <tr>
    <td onclick="zmenHodnotuGUI('audio','kuchyne');">
     <img src="images/audio.jpg" id="radio"> <br /> Radio </td>
    <td> </td>
    <td onClick="zmenMenu('menu','polozky');">
          <img src="images/back.png"> <br /> Zpátky</td>
  </tr> <tr>
   <td colspan="3" id="radiob" style="display:none;">
     <audio id="bomba" controls>
       <source src="http://147.228.209.18:8000/;"
          type="audio/mp3" />
     </audio>
          <script>
               var aud = document.getElementById("bomba");
               aud.volume=0.1;
          </script> </td>
  </tr>
 </table>
</div>
```

Položka uzamknout

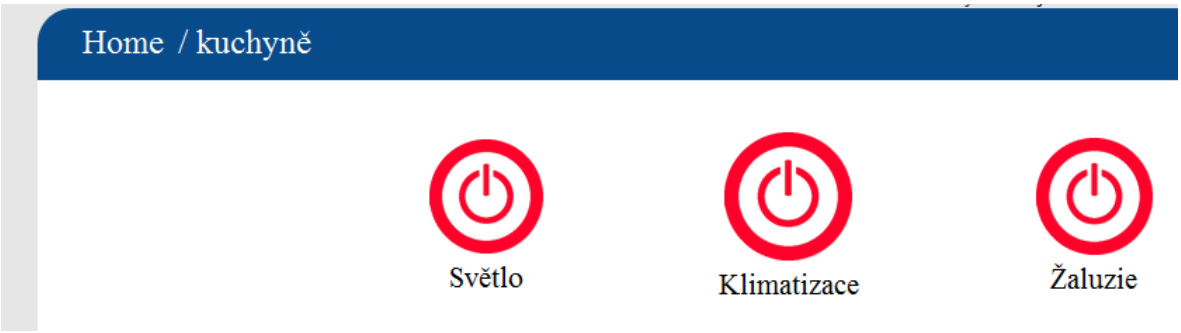
Poslední možností v nabídce je uzamknutí. Uživatel má možnost buďto vypnout všechna zařízení či vypnout všechna zařízení a odhlásit se ze systému – vhodné například pokud člověk odchází z domu, v případě napojení na alarm by domácnost mohla být zakódována. Grafické znázornění najdeme na obrázku 6.11.

Obr. 6.11 Nabídka uzamknout

```
<div id="zamknout" style="display:none;" class="hide">
 <table>
    <tr>
      <td onclick="turnOff('all');"><img src="images/off.png" />
          <br /> Vše vypnout </td>
      <td onclick="turnOff('logout');">
          <img src="images/alarm.png" /><br /> Vše vypnout a
               uzamknout</td>
      <td onClick="zmenMenu('menu','polozky');">
          <img src="images/back.png"> <br /> Zpátky</td>
    </tr>
  </table> </div>
```

Stav v místnosti

Stejně jako tlačítko Home i tlačítko kuchyne skrývá šikovnou funkci. Po stisknutí se nám zobrazí aktuální stav spotřebičů v místnosti – světlo, klimatizace a žaluzie. Jedná se o podobnou funkci jako v případě ovládání světla s rozdílem, že se prohodily spotřebiče s místnostmi. Znázornění je ukázáno na obrázku 6.12.



Obr. 6.12 Nabídka kuchyně

```
<div id="kuchyn" style="display:none;" class="hide">
   <table>
     <tr>
       <td onclick="zmenHodnotuGUI('svetlo','kuchyne');">
          <img src="images/svetlo.png" id="svetlo-kuchyne"><br />
               Světlo </td>
       <td onclick="zmenHodnotuGUI('klimatizace','kuchyne');">
          <img  src="images/klimatizace.jpg" id="klimatizace-
          kuchyne"> <br /> Klimatizace</td>
       <td onclick="zmenHodnotuGUI('zaluzie','kuchyne');">
          <img  src="images/zaluzie.jpg" id="zaluzie-kuchyne">
          <br /> Žaluzie</td>
     </tr>
     <tr>
       <td></td>
       <td onClick="zmenMenu('menu','polozky');">
          <img  src="images/back.png"> <br /> Zpátky</td>
       <td></td>
     </tr>
   </table>
</div>
```

6.3.1.3 Stavový řádek

Poslední částí stránky je informační panel, který dává uživateli zpětnou vazbu o tom, v jakém stavu se systém nachází. Definovány jsou:

- Nepřipojeno – uživatel není přihlášen
- Připojeno – proběhlo přihlášení uživatele
- Rozpoznávač zapnut – aplikace rozpoznává vstup
- Rozpoznávač vypnut – aplikace nerozpoznává vstup

Pokud je rozpoznávač zapnut, uživatel dále dostává zpětnou vazbu o síle signálu pomocí barevného kolečka, které signalizuje daný stav. Definované stavy jsou

- Prázdné kolečko – žádný vstup či příliš slabý signál
- Zelené kolečko – aplikace rozpoznává řeč
- Červené kolečko – uživatel mluví příliš nahlas

Uživatel má také možnost vypnout či zapnout rozpoznávač. To provede kliknutím na ikonku mikrofonu. Ukázka stavového řádku je na obrázku 6.13.

Obr. 6.13 Stavový řádek

```
<div id="bottom">
    <table style="position:relative; width:90%;">
       <tr>
         <td id="recognize" style="display:none;"><a href="#">
               <img id="recog" src="images/rec_off.png" /></a>
          </td>
         <td align="right">   Stav:
               <span id="volume">Nepřipojeno</span>
               <span id="vol" style="font-size:150%;"> </span>
         </td>
       </tr>
    </table>
</div>
```

Stejně jako u navigace i nyní definujeme border-radius, kterým zaoblíme rohy. Dále definujeme velikost elementu a odsazení, abychom dosáhli uživatelsky přívětivého prostředí.

CSS

```
#bottom {
   position:relative;
   background-color:#084B8A;
   color:#ffffff;
   border-bottom-left-radius:25px;
   border-bottom-right-radius:25px;
   padding-left:5%;
   font-size: 20px;
   height:8%;
 }
```

6.3.2 Funkce

Jak již bylo zmíněno, interaktivitu grafickému prostředí přidávají funkce v JavaScriptu. Pro dotykové prostředí jsou definovány odlišné funkce než pro hlasové ovládání, avšak některé funkce jsou totožné. Aby oba soubory mohly sdílet stejné proměnné, je třeba je nadefinovat na začátku souboru. Samotnou deklaraci dané proměnné provedeme následně v konkrétním souboru.

```
var def_mist = "kuchyne"; // defaultní místnost
var pripojeno;            // informace přihlášení
var ws                  // websocket server na lokálním zařízení
var hlas = "false"        // přihlášení hlasem či kliknutím
stav = { akce : '', mistnost : '', vec: '' }   // stav aplikace
```

6.3.2.1 Funkce prihlasUzivatele

Funkce je volána při načtení stránky či po přihlášení uživatele. Funkce nejdříve nastaví hodnoty u všech spotřebičů dle jejich aktuálního stavu (zapnuto / vypnuto). Vzhledem k tomu, že audio není definováno stejným způsobem jako ostatní parametry, je třeba s ním pracovat zvlášť. Funkce dále upraví grafické prostředí – zobrazí navigaci a přidá defaultní místnost. Zobrazí základní přehled všech spotřebičů a upraví informace ve stavovém řádku aplikace. Informaci o připojení uživatele uloží do lokálního uložiště, díky čemuž může být znovu načtena stránka a data se neztratí.

```
function prihlasUzivatele(app) {
    for (var vec in app) {
        if (vec =="audio") {
           $("#radio").attr("src","images/"+app.audio+".png");
           if (app.audio=="on")
               $('#frame').attr('src',
               'http://player.abradio.cz/player/27/805/');
           else $('#frame').attr('src','');
        } else {
           for (var mistnost in app[vec]) {
               $("img#"+vec+"-"+mistnost).attr("src",
                     "images/"+app[vec][mistnost]+".png");
              }
        }
    }
    $("#navigace ul").show();
    $("#def-mistnost").text(def_mist);
    $("#polozky").show();
    $('#recog').attr('src','images/rec_off.png');
    document.getElementById("volume").innerHTML = "Připojeno";
    pripojeno="true";
    localStorage.setItem("pripojeno","true");
    zmenMenu('menu','polozky');
 }
```

6.3.2.2 Funkce zmenHodnotuGUI

Funkce je volána při kliknutí na položku v aplikaci. Využívá 2 parametry – místnost a spotřebič. Na základě těchto dvou údajů již může vykonat danou akci – nejdříve se podívá na aktuální nastavení a poté stav nastaví na opačný. Při změně nastavení zašle příkaz na server, který ho následně vykoná. Informace opět uložíme do lokálního uložiště pro případ výpadku.

Kromě stavu audia musí aplikace zvlášť vyřešit také samotnou implementaci hudby, která se pouští za pomocí příkazu play() či v opačném případě zastavuje pomocí příkazu pause().

```
function zmenHodnotuGUI(co,kde) {
   if (stav.vec=="audio") {
     var b = document.getElementById("bomba");
     if (app.audio=="on") {
          b.pause()
          app.audio="off"
          $("#radiob").hide()
     }
     else {
          b.play()
          app.audio="on"
          $("#radiob").show()
     }
     $("#radio").attr("src","images/"+app.audio+".png");
      sendMessage("z audio "+app.audio)
    } else  {
     stav.vec = co;
     stav.mistnost = kde;
     if (app[co][kde]=="on") {
          app[co][kde] = "off";
          sendMessage("z "+stav.vec+" "+stav.mistnost+" off")
     } else {
          app[co][kde]="on";
          sendMessage("z "+stav.vec+" "+stav.mistnost+" on")
     }
     $("img#"+[co]+"-
          "+[kde]).attr("src","images/"+app[co][kde]+".png");
   }
   localStorage.setItem('app', JSON.stringify(app));
   stav = {  akce : '', mistnost : '', vec: ''  }
}
```

6.3.2.3 Funkce turnOff

Funkce je volána převážně z nabídky Uzamknout systém. Jediná výjimka je při parametru „login“, který je aplikaci předán v případě, že ze serveru přijde zpráva o tom, že uživatel není přihlášen. Tímto způsobem je zabezpečeno uživatelské rozhraní proti neautorizovanému vstupu. Při parametru „shutdown“ server vypne všechna zařízení a

odpojí uživatele. Při parametru „all“ server zašle novou defaultní konfiguraci, ve které jsou všechna zařízení vypnuta.

```
function turnOff(volba) {
     if (volba=="login") {
                localStorage.clear()
                window.location="index.html";
     } else if (volba=="shutdown") {
          sendMessage("out")
     } else if (volba=="all") {
          sendMessage("all")
     }
 }
```

6.3.2.4 Funkce zmenMenu

Funkce pracuje s obsahem stránky tím, že zobrazuje či schovává patřičné části. Díky změnám prostředí stránka zachovává patřičnou interaktivitu, ale i vlastnosti. Pokud tedy pustíme hudbu, můžeme bez starostí ovládat další funkčnosti a hudba bude stále hrát. V prohlížeči je totiž stále definována, pouze není zobrazena. Funkce má dva parametry – první udává, jakou část chceme skrýt a druhý, jakou chceme zobrazit. Aplikace také přidává informaci o aktuálním umístění do navigace za tlačítko Home.

```
function zmenMenu(skryt, zobraz) {
     $(".hide").hide();
      if (pripojeno=="false") {
                skryt="menu";
                zobraz="login";
     }
     if (skryt=="menu") {
          $(".hide").hide();
          stav.vec = '';
          stav.akce='';
          stav.mistnost=''
     } else if (zobraz=="kuchyn") {
          $("#polozky").hide();
          $(".hide").hide();
     } else {
          $("#"+ skryt).hide();
          stav.vec = zobraz;
     }
     $("#"+ zobraz).show();
     if (zobraz =="kuchyn") $("#nav").html("/ <a href='#'>
          kuchyně</a>")
     else if (zobraz !="polozky") $("#nav").html("/<a href='#'>"
          +zobraz +"</a>")
     else $("#nav").text('')
  }
```

6.3.2.5 Funkce sendMessage a waitForSocketConnection

Funkce sendMessage je hlavní funkce, která zodpovídá za komunikaci ze strany klienta na server. Využívá přitom funkci waitForSocketConnection, která zajistí spojení se serverem v případě momentální nedostupnosti. Funkce je důležitá hlavně při začátku navazování komunikace, kdy klient kontaktuje server a požaduje nastavení.

```
function sendMessage(msg) {
        waitForSocketConnection(ws, function() {
              ws.send(msg);
        });

};

function waitForSocketConnection(socket, callback){
      setTimeout(
          function(){
              if (socket.readyState === 1) {
                  if(callback !== undefined){
                      callback();
                  }
                  return;
              } else {
                  waitForSocketConnection(socket,callback);
              }
          }, 5);
};
```

Všechny výše zmíněné funkce se volají až po načtení celého dokumentu, což je zajištěno funkcí

```
$(document).ready(function ()
```

Při načtení dokumentu nejdříve zjistíme, jestli již máme uloženou v lokálním uložišti informaci o tom, zda jsme přihlášeni či nikoli. Pokud nejsme, zobrazíme uživateli přihlašovací stránku, pokud jsme, zobrazíme mu hlavní nabídku

```
pripojeno = localStorage.getItem("pripojeno");
if (typeof pripojeno === 'undefined' || pripojeno === null ||
   pripojeno =="false") {
        pripojeno = "false";
        $('#login').show()
} else {
      sendMessage("conf")
}
```

6.3.2.6 onClick události

Definujeme celkem 3 události, které se volají po kliknutí na daný element. U těchto událostí nelze využít JavaScript většinou proto, že se nejedná o původní obsah, který chceme měnit, ale o obsah již skriptem změněný. Události obstarává jQeury, který pracuje s DOM mnohem lépe než JavaScript a umí takové případy obejít.

První událost je volána při kliknutí na tlačítko Odeslat a postará se o zaslání hesla na server.

```
 $("#log").click(function(evt) {
         sendMessage($('#login5').val())
 });
```

Další funkce se stará o zobrazení kuchyně při kliknutí na defaultní místnost v pravém horním rohu aplikace.

```
$("#def-mistnost").click(function(evt) {
         zmenMenu('polozky','kuchyn')
 });
```

Poslední funkce zapíná a vypíná rozpoznávač řeči a spolu s ním i mění ikonku ukazující aktuální stav. Pro nastavení využívá proměnné isRecognizing, kterou poskytuje metoda volaná přes SpeechCloud.

```
$("#recognize").click(function(evt) {
            if (sc.isRecognizing == true) {
              sc.asr_pause();
              $("img#recog").attr('src','images/rec_off.png')
              $("#volume").text("Rozpoznávač vypnut");
              $("#vol").text("")
            } else {
              $("img#recog").attr('src','images/rec_on.png')
              sc.asr_recognize();
              document.getElementById("volume").innerHTML =
                      "Rozpoznávač zapnut";
            }

 });
```

6.3.2.7 Komunikace skrze WebSocket s lokálním serverem

Připojením na server deklarujeme proměnnou ws. Jak již z části Server na lokálním zařízení víme, připojujeme se na adresu lokálního zařízení pod portem 8888.

```
ws = new WebSocket("ws://localhost:8888");
```

Událost onmessage

Událost přijímá zprávy ze strany serveru a následně volá potřebné funkce. Podává také uživateli zpětnou vazbu o tom, co se na serveru právě děje a to hlavně při ověřování uživatele. V případě špatně zadaného hesla poskytne kromě hlasové i textovou informaci, kterou zobrazí nad přihlašovacím formulářem. Při obdržení konfigurace přepne gramatiku a zavolá funkci prihlasUzivatele. Pokud se uživatel snaží obejít zabezpečení a zašle příkaz bez přihlášení, aplikace ho odhlásí pomocí funkce turnOff.

```
ws.onmessage = function(evt) {
    if (evt.data == "verif") {
       sc.tts_synthesize({text: "Ověřuji uživatele"});
    }
    if (evt.data == "nepripojen") {
       turnOff('login')
    }
    if (evt.data == "ne") {
       sc.tts_synthesize({text: "Uživatel nenalezen."});
       sc.tts_synthesize({text: "Zadejte prosím vaše heslo"});
       $("#login2").prepend("Špatně zadané heslo<br />")
    }
    if (evt.data.slice(0,2) =="ok") {
       if (hlas=="true")
          sc.tts_synthesize({text: "Uživatel ověřen"});
       hlas = "false"
       if (pripojeno!="true") sc.asr_set_grammar_uri({
          grammar_type: "esgf",
          grammar_uri:asr_grammar_uri2})
          app = JSON.parse(evt.data.slice(2))
          prihlasUzivatele(app)
    }
}
```

Událost onclose

Pokud je server odpojen, zobrazí informaci o tom, že je potřeba připojit server, jinak nelze aplikaci ovládat.

```
ws.onclose = function(evt) {
   $("#obsah").html("<br /><br />Server odpojen <br /><br />
         Zapněte server a <a href='index.html'>aktualizujte</a>
         stránku<br /><br />");
}
```

6.4 Hlasový Dialogový systém

Po dokončení grafické podoby aplikace bylo zapotřebí začít pracovat na hlasovém ovládání, které by dokázalo plnit veškeré požadavky a funkčně se projevovalo podobně jako ovládání pomocí dotyku. I přesto, že systém není obsáhlý, je zapotřebí věnovat mnoho úsilí do tvorby této části, která je pro celý systém kritická. Vzhledem k tomu, že při

rozpoznávání řeči se může ledacos přihodit, bylo zapotřebí definovat a vyřešit všechny možné problémy, které mohou nastat.

Pokud systém špatně rozpozná promluvu, je zapotřebí zvolit správné otázky a zjistit, jaký je uživatelův požadavek. Jelikož aplikace ovládá různé místnosti a spotřebiče, je zapotřebí zajistit, abychom nenaplnili stav požadavkem, který není konzistentní s možností domácnosti. Nesmí se tak například stát, že bychom získali následující stav:

věc: 'světlo'
místnost: 'garáž'
akce: 'on'

Jelikož světlo je definováno pouze pro místnosti kuchyně, ložnice a obývák.

Dále je třeba počítat s tím, že můžeme špatně rozpoznat uživatelův požadavek a vykonat tak nežádoucí akci. Proto je nutné definovat akci pro zamítnutí a vrácení poslední provedené akce zpět a následné vykonání původně požadované akce. Z tohoto důvodu je důležité, aby aplikace ihned reagovala změnou obsahu při rozpoznání dané promluvy.

Jedním z největších problémů je skutečnost, že systémy pro rozpoznávání vzdálené řeči jsou teprve v raném vývoji a neexistují tak modely pro rozpoznávání této řeči. Model pro rozpoznávání řeči je natrénován na telefonních datech, při kterých má uživatel mikrofon přímo u úst. Je tedy vysoce pravděpodobné, že při ovládání aplikace např. skrze hlasitý odposlech na telefonu nebudeme mít příliš dobré výsledky. Pro reálné využití bude tedy nejspíše třeba implementovat do systému kvalitní mikrofon, který dokáže tlumit zvuky na pozadí a uživatelovy požadavky přenáší čistě. Tuto skutečnost bychom si měli ověřit pomocí uživatelských testů.

6.4.1 Gramatika

Základem hlasového dialogového systému je gramatika. Každý člověk má jedinečný styl mluvy. Je proto velmi těžké obsáhnout všechny možné způsoby požadavků. Právě testování systému od různých uživatelů ukáže, o kolik slov je třeba rozšířit stávající gramatiku či naopak jaká slova odstranit pro správnou funkčnost. Může se stát, že příliš mnoho slov v gramatice ve výsledku škodí při rozpoznávání, jelikož pravděpodobnostní modely mohou vybrat nesprávné slovo. Při tvorbě byly rozděleny požadavky tak, aby se již při promluvě nemohlo stát, že by aplikace rozpoznala nežádoucí příkaz, například rozsvítit dveře v kuchyni. Toho bylo docíleno tak, že pro každý spotřebič byla vytvořena vlastní pravidla.

Aplikace využívá dvě gramatiky – jednu pro přihlášení uživatele a probuzení, druhou pro ovládání aplikace.

Gramatika pro přihlášení je jednoduchá a obsahuje pouze jeden příkaz

```
#ESGF V1.0 UTF-8 cs;
grammar gram;
public  <gramatika> = <BEGIN_GRAMMAR> <VIRTUAL> (pripojit|prihlasit) {prip}
                <END_GRAMMAR>;
```

Druhá gramatika je značně složitější a obsahuje řadu pravidel

```
#ESGF V1.0 UTF-8 cs;
grammar gramatika;
public <gramatika> = <BEGIN_GRAMMAR> <VIRTUAL> ( <prikazy> | <potvrzeni>
| <spotrebice> | <mistnost> | <zamitnuti>+ ) <END_GRAMMAR>;
```

Znaménko + u pravidla zamítnutí značí, že se může vyskytovat jedenkrát a více.

6.4.1.1 Pravidlo příkazy

Jedná se o základní příkazy, které uživatel může říci, konkrétně o reakce na otázky: Co pro vás mohu udělat nebo co chcete udělat. Dále pravidlo definuje příkaz pro vypnutí všech zařízení a odhlášení ze systému.

```
<prikazy> = ( (menu| ([hlavni] nabidka)| (nic nechci) | nic ) {nabidka} |
( (vse vypnout)|(vypnout vse)|(vypnout system) ) {vypnout} | (odhlasit [se]) {odhlas});
```

6.4.1.2 Pravidlo potvrzení

V případě, že aplikace nezná akci, zeptá se, zda chce uživatel přístroj vypnout či zapnout Jedná se tedy o odpověď na položenou otázku. Také funguje jako odvolání nesprávného požadavku.

```
<potvrzeni> = ( (ano|jo|(ano chci)) {ano} | (ne|nechci) {ne} );
```

6.4.1.3 Pravidlo spotřebiče

Toto pravidlo rozšiřují další pravidla, která definují veškerou gramatiku pro daný spotřebič. Pro co největší srozumitelnost a snadnou manipulaci je definováno více pravidel, která se navzájem doplňují.

```
<spotrebice> = <nabidka> | <dvere> | <svetlo> | <klimatizace> | <zaluzie> | <audio>
```

Pravidlo generuje například následující gramatiku:

- světlo, rozsvítit světlo, rozsvítit světlo v ložnici, světlo v ložnici, rozsvítit, rozsvítit v ložnici
- otevřít, garáž, otevřít garáž

Pravidlo nabídka

Definuje pouze daný spotřebič – nutné pro případ, kdy aplikace zná akci i místnost a dotazuje se na chybějící znalost.

```
<nabidka> = ( (svetlo|svetla) {v_svetlo} | (klimatizace|klimatizaci) {v_klimatizace} |
        zaluzie {v_zaluzie} | (radio|hudbu|audio) {v_audio} );
```

Další pravidla

Zbylá pravidla definují ostatní požadavky. Pravidla klimatizace a žaluzie jsou totožná s pravidlem světlo, až na jiný spotřebič.

```
<dvere> = ( ( (<dvere_on> | <dvere_off>) [<dvere_co>] ) | ( [(<dvere_on> |
<dvere_off>)] <dvere_co> ) );

<dvere_on> = (( otevri | otevrit | odemknout | odemkni ) {turn_on});
<dvere_off> = (( zavri | zavrit | zamknout | zamkni) {turn_off});
<dvere_co> = ( ( ( ( (hlavni | predni) {m_hlavni} ) | zadni {m_zadni} ) ( (vchod | dvere)
{v_dvere} ) ) | ( (vrata | garaz) {m_garaz}) | dvere {v_dvere} );


<svetlo> = ( ( (<svetlo_on> | <svetlo_off>) (svetlo | svetla) {v_svetlo} [<mistnost>] ) | (
(svetlo | svetla) {v_svetlo} [<mistnost>] ) | ( (<svetlo_on> | <svetlo_off>) {v_svetlo}
[<mistnost>] ) );

<svetlo_on> = ((zapnout | zapni | rozsvitit | rozsvit | rosvitit | rosvit) {turn_on});
<svetlo_off> = ((vypnout | vypni | zhasnout | zhasni) {turn_off});


<audio> = ( (<trn_on> | <trn_off>) (radio | hudbu | audio) {v_audio} [<mistnost>] ) | (
(radio | hudbu | audio) {v_audio} [<mistnost>] ) | ( (<trn_on> |<trn_off>) [<mistnost>])

<trn_on> = ( (zapnout | zapni | pust | pustit) {turn_on} );
<trn_off> = ((vypnout | vypni | zastav | zastavit) {turn_off});
```

6.4.1.4 Pravidlo místnost

Jak již název napovídá, pravidlo místnost obsahuje místnosti, avšak ne všechny, pouze kuchyni, ložnici a obývák. Dveře, hlavní a zadní vchod jsou definovány v pravidle dvere_co. Pravidla jsou rozdělena právě z důvodu, že tyto místnosti jsou definovány u spotřebičů světlo, klimatizace a žaluzie Pokud bychom zde měli i dveře, mohlo by nastat, že se promluva špatně rozpozná a z rozpoznávače získáme nekonzistentní požadavek.

```
<mistnost> = ( [v] ( (kuchyn | kuchyne | kuchyni) {m_kuchyne} | (obyvak | obyvaku)
{m_obyvak} | (loznice | loznici) {m_loznice} ) );
```

6.4.1.5 Pravidlo zamítnutí

Posledním pravidlem je zamítnutí, které definuje promluvy související s odmítnutím provedené akce. Pravidlo využívá již výše definovaná pravidla a nové pravidlo výplň, které obsahuje výplňová slov typu „chtěl jsem, já chtěl, já nechtěl,…“

Pravidlo generuje například následující gramatiku:

- ne já nechtěl světlo, já jsem chtěl žaluzie
- nechtěl jsem zapnout klimatizaci, chtěl jsem vypnout světlo
- já nechtěl kuchyni, ale ložnici

<zamitnuti> = (<vypln> (<spotrebice> | <nabidka> | <mistnost>));

Pravidlo výplň

<vypln> = (((ne | nechci) | ((to | ja) jsem (chtel | nechtel | chtela | nechtela)) | ([ja] (chtel | nechtel | chtela | nechtela) [jsem])) {ne} | (ale | ((chtel | chtela) jsem) | ((rad | rada) bych) | (ja bych) | prosim | (mohl | mohla | muzes)));

6.4.2 Funkce

Jak již bylo zmíněno v části GUI, aby oba soubory mohly sdílet stejné proměnné, je třeba je nadefinovat na začátku souboru. Tento script využívá i funkce z minulého souboru, konkrétně se jedná o funkci zmenMenu.

```
// gramatiky
var asr_grammar_uri =
    "http://students.kky.zcu.cz/nedvedj/gram.esgf"
var asr_grammar_uri2 =
    "http://students.kky.zcu.cz/nedvedj/gramatika.esgf

var sc;                     // instance SpeechCloud
var def_mist = "kuchyne";   // defaultní místnost
var potvrzeni;              // čekáme na potvrzení
var hist;                   // historie stavu
var sessionID;              // ID pro získání hesla
var otazka = "ne"           // ptáme se uživatele
```

6.4.2.1 Funkce zmenHodnotu

Funkce má velmi podobný název jako funkce pro ovládání dotykem a má i stejné výstupy, liší se však stylem ovládání.

Funkce je volána, pokud máme zaplněný celý stav, který zároveň využívá jako vstupní parametr. Cílem je pouze změna hodnoty – zda je či není nastavena, se nekontroluje, jelikož kontrola proběhla již dříve. Informace se opět ukládají do lokálního úložiště pro případ výpadku.

```
function zmenHodnotu(cmd) {
 if (cmd.vec=="audio") {
      var b = document.getElementById("bomba");
     app[cmd.vec] = cmd.akce;
      $("#radio").attr("src","images/"+app[cmd.vec]+".png");
      if (cmd.akce=="on") {
           b.play()
           $("#radiob").show()
     } else {
           b.pause()
           $("#radiob").hide()
     }
 }  else {
     app[cmd.vec][cmd.mistnost] = cmd.akce;
     $("img#"+[cmd.vec]+"-"+[cmd.mistnost]).attr("src",
           "images/"+app[cmd.vec][cmd.mistnost]+".png");
 }
 if (potvrzeni!="ne" && otazka !="ano") {
           hist = stav;
           stav = {  akce : '', mistnost : '', vec: ''  }
           zmenMenu('menu', 'polozky')
 }
 localStorage.setItem('app', JSON.stringify(app));
}
```

6.4.2.2 Funkce zkontrolujVýjimky

Funkce zajišťuje, aby nedocházelo k nekonzistentním požadavkům. Díky pravidlům v gramatice, by rozpoznávačem neměly projít žádné nepovolené promluvy, avšak při doplňování a úpravách stavů při zamítnutí by k takovému případu dojít mohlo. Gramatikou tedy sice neprojde například promluva: „rozsvítit světlo v garáži“, pokud bychom však měli v historii uložený stav

věc: 'světlo'	*místnost: 'kuchyně'*	*akce: 'on'*

A poté vznesli požadavek „nechtěl jsem světlo, ale garáž“, nový stav by byl

věc: 'světlo'	*místnost: 'garáž'*	*akce: 'on'*

Před provedením akce je tedy třeba zkontrolovat aktuální stav.

```
function zkontrolujVyjimky(stav) {
 if (typeof hist !="undefined" && hist.mistnost==stav.mistnost &&
    hist.mistnost=="garaz" && stav.vec!="dvere") stav.mistnost=""
 if (stav.vec=="audio" && typeof hist !="undefined" &&
    hist.vec=="audio" &&  stav.mistnost!="") stav.vec=""
 if (stav.vec=="audio") stav.mistnost='';
 if (stav.mistnost=="garaz" && stav.vec=="") stav.vec="dvere";
 if (typeof hist !="undefined" && hist.vec =="dvere" &&
    hist.vec ==stav.vec && (stav.mistnost =="kuchyne" ||
     stav.mistnost =="loznice" || stav.mistnost =="obyvak"))    {
              stav.vec=""
 }
 if (stav.vec =="dvere" && (stav.mistnost =="kuchyne" ||
    stav.mistnost =="loznice" || stav.mistnost =="obyvak")) {
         stav.mistnost=""
 }
 if (stav.vec!="audio" && stav.vec!="dvere") {
    if (stav.mistnost=="hlavni" || stav.mistnost=="zadni" ||
         stav.mistnost=="garaz") stav.mistnost=""
 }
}
```

6.4.2.3 Funkce generujPromluvu

Jednou z nejdůležitějších funkcí je právě funkce pro generování promluvy. Na základě zaplnění stavu aplikace zvolí promluvu k syntéze a následně ji prezentuje uživateli. Při rozpoznávání se můžeme dostat do několika stavů, ve kterých nám chybí určitá informace. Právě zjištění této informace je pro vykonání akce důležité. Abychom informaci získali, je třeba zvolit správnou otázku, která je konkrétní. Pokud by se aplikace při chybějící položce stavu ptala stále stejně, například: „upřesněte prosím svůj požadavek“, uživatel by nevěděl, co aplikace již rozpoznala a co ne, a opakoval by stále dokola svoji promluvu, která například ani nemusí být definována gramatikou.

Vzhledem k obsáhlosti kódu bude ukázka pouze pro světlo. Funkce generuje otázky uvedené v tabulce 6.1

Věc	Místnost	Akce	Otázka
ne	ne	ne	Co pro vás mohu udělat?
ne	ano	ne	Co chcete v místnosti udělat?
ano	ano	ne	Světlo v místnosti je zapnuto/vypnuto chcete ho zapnout/vypnout?
ne	ano	ano	Co chcete v místnosti zapnout/vypnout?
ano	ne	ano	V jaké místnosti chcete zapnout/vypnout světlo?
ano	ano	ano	Vypínám světlo v místnosti / Světlo v místnosti je již vypnuto
ano	ne	ne	Jaký je váš požadavek ohledně světla?
ne	ne	ano	Co chcete zapnout / vypnout?

Tabulka 6. 1 Generovatelné otázky

Důležitým krokem je také nahradit kódové označení reálnými slovy.

```
function generujPromluvu(stav) {
  mistnosti = [
        ["zadni","zadní vchod"],
        ["garaz","garáž"],
        ["hlavni","hlavní vchod"],
        ["obyvak","obýváku"],
        ["loznice","ložnici"],
        ["kuchyne","kuchyni"],
        ["svetlo","světla"],
        ["audio","hudby"],
        ["zaluzie","žaluzií"],
        ["dvere","dveří"]
    ]

tx = ""
```

Víme věc, místnost i akci

```
if ( (stav.akce != "" && stav.vec!="" && stav.mistnost!="") ||
  stav.akce != "" && stav.vec=="audio") {
    if (stav.akce=="off") {
          if (stav.vec=="svetlo") {
             if (app[stav.vec][stav.mistnost] == stav.akce)
                tx = "Světlo v    "+stav.mistnost+" je již vypnuto"
             else tx = "Vypínám světlo v "+stav.mistnost;
          }
    } else {
          if (stav.vec=="svetlo") {
             if (app[stav.vec][stav.mistnost] == stav.akce)
                tx = "Světlo v "+stav.mistnost+" je již zapnuto"
             else tx = "Zapínám světlo v "+stav.mistnost;
          }
    }
    zmenHodnotu(stav);
}
```

Víme věc, místnost, ale nevíme akci

```
else if (stav.vec !="") {
    if (stav.mistnost!="") {
       if (app[stav.vec][stav.mistnost]=="on" ||
          app[stav.vec]=="on") {
          if (stav.vec=="svetlo")
                tx = "Světlo v "+stav.mistnost+" je zapnuto.
                     Chcete ho vypnout?"
     } else {
          if (stav.vec=="svetlo")
                tx = "Světlo v "+stav.mistnost+" je vypnuto.
                     Chcete ho zapnout?"
     }
```

Víme věc, nevíme místnost a akci

```
else tx = "Jaký je váš požadavek ohledně " + stav.vec +"?"
    zmenMenu('polozky', stav.vec)
}
```

Víme místnost a akci, nevíme věc

```
else if (stav.vec =="" && stav.akce !="" && stav.mistnost!="") {
    if (stav.akce=="off") tx = "vypnout";
    else tx = "zapnout";
    tx = "Co chcete v " + stav.mistnost + tx + "?"
}
```

Víme věc i akci, nevíme místnost

```
else if (stav.akce !="" && stav.vec!="") {
    if (stav.akce=="off") tx = "vypnout";
    else tx = "zapnout";
    tx = "V jaké místnosti chcete "+tx+" " + stav.vec + "?"
}
```

Víme akci, nevíme místnost a věc

```
else if (stav.vec =="" && stav.akce !="") {
    if (stav.akce=="off") tx = "vypnout";
    else tx = "zapnout";
    tx = "Co chcete " + tx +"?"
}
```

Víme místnost, nevíme akci a věc

```
else if (stav.mistnost!="") {
    tx = "Co chcete v " + stav.mistnost +" udělat?"
    if (stav.mistnost=="kuchyne") zmenMenu('polozky','kuchyn')
}
```

Substituce z pole náhrady (háčky a čárky)

```
for (i = 0; i<mistnosti.length;i++) {
     tx = tx.replace(mistnosti[i][0],mistnosti[i][1])
}
return tx
```

Stejně jako v předchozím případě funkce voláme až po načtení celého dokumentu

```
$(document).ready(function ()
```

6.4.2.4 Komunikace skrze WebSocket se SpeechCloudem

V části Komponenty a technologie jsme již popsali, jak probíhá připojení na SpeechCloud.

```
sc = new SpeechCloud({
  uri: https://hydra4.kky.zcu.cz:9443/v1/speechcloud/edu-nedved,
  tts: „#audioout"
 });
```

Událost ws_session

Při navázání komunikace získáme ze SpeechCloudu session ID, na základě kterého následně můžeme přistoupit k pořízeným logům a audio nahrávkám. ID zašleme na soubor save.php, ve kterém je uložen PHP script. Ten ho uloží do souboru, abychom případně mohli rekonstruovat problematickou promluvu.

```
 sc.on('_ws_session', function (data) {
   sessionID = data.id;
   $("#frame").attr("src","save.php?id="+data.id);
 });
```

Na rozdíl od JavaScriptu probíhá vyhodnocení PHP kódu na straně serveru, což nám umožňuje upravovat soubory na něm uložené. Kromě ID do souboru zapíšeme i aktuální čas přístupu na stránky, které budeme porovnávat s časem odeslání formuláře, a tím identifikujeme daného uživatele.

```
<?php
 $myfile = fopen("logs.txt", "a") or die("Unable to open file!");
 fwrite($myfile, "\n".date('Y-m-d H:i:s')."|https://cak.zcu.cz:
      9443/v1/session/".$_REQUEST['id']."?format=json.html");
 fclose($myfile);
?>
```

Událost asr_signal

Na základě informací o hodnotě signálu poskytujeme uživateli zpětnou vazbu pomocí barevných koleček ve stavovém řádku. Využijeme k tomu ASCII znaků ○ (●) a ● (○).

```
sc.on('asr_signal', function (msg) {
   if (msg.speech) $("#volume").text("Rozpoznávám");
   else $("#volume").text("Slabý signál ");
   $("#vol").html("&#9679;");
   if (msg.level > 6.2) {
           $("#vol").css("color", "#ff0000");
   } else if (msg.speech) {
           $("#vol").css("color", "#00ff00");
   } else {
           $("#vol").html("&#9675;");
           $("#vol").css("color", "#000000");
   };
});
```

Událost start_session

Při navázání spojení SpeechCloud informujeme o tom, jakou gramatiku chceme využívat. Gramatiku nastavujeme dle toho, zda uživatel je či není přihlášen.

```
sc.on('sc_start_session', function (msg) {
    if (pripojeno != "true") {
       sc.asr_set_grammar_uri({grammar_type: "esgf",
               grammar_uri:asr_grammar_uri});
    }
    else {
       sc.asr_set_grammar_uri({grammar_type: "esgf",
               grammar_uri:asr_grammar_uri2});
    }
});
```

Událost asr_result

Nejdůležitější částí scriptu je právě událost asr_result, která vrací rozpoznanou promluvu. Z této funkce jsou volány všechny výše popsané funkce. Po vyhodnocení řeči z nahrávky pozastavíme rozpoznávač, díky čemuž se nahrané audio uloží.

Dokud probíhá přihlašování uživatele, tak s promluvou nijak nepracujeme a čekáme na zprávu ze serveru.

```
sc.on('asr_result', function (msg) {
 if (msg.partial_result) {
    return;
 };
 sc.asr_pause();
 if (hlas=="true") {
    sc.tts_synthesize({text: "Probíhá ověřování uživatele"});
    sendMessage("verif "+sessionID)
 }
```

Přijatou promluvu zpracujeme pomocí regulérních výrazů, za pomocí nichž vyselektujeme tagy z gramatiky, se kterými následně dále pracujeme. Pokud nerozpoznáme žádné tagy, vyzveme uživatele k zopakování promluvy.

```
    if (data!="") {
        var reg = /[[a-z_]+]/gi;
        var prikazy = data.match(reg);
        var txt="";
        if (prikazy=="") {
            txt = "Zopakujte prosím váš požadavek";
         }
    }
```

Při rozpoznání požadavku přihlásit přepneme přihlašovací obrazovku a nastavíme druhou gramatiku, uživatele pak vyzveme k zadání hesla.

```
else if (prikazy=="[prip]") {
   sc.asr_set_grammar_uri({grammar_type: "esgf",
                    grammar_uri:asr_grammar_uri2})
   $("#odhlas").show();
   $("#def-mistnost").text(def_mist);
   $("#login").hide()
   $("#login2").show()
   sc.tts_synthesize({text: "Zadejte prosím vaše heslo"});
```

Při rozpoznání jednoho z příkazů akci ihned vykonáme, jelikož se jedná o činnosti, které nepracují se stavem. Nepotřebujeme tak zpracovávat další tagy.

```
    if (prikazy[i] == "[nabidka]") {
        sc.tts_synthesize({text: "Co pro vás mohu udělat?"});
        zmenMenu('menu', 'polozky')
        break
   }
   else if (prikazy[i] == "[vypnout]") {
        sc.tts_synthesize({text: "Vypínám všechna zařízení"});
        turnOff('all')
        break
   }
   else if (prikazy[i] == "[odhlas]") {
        sc.tts_synthesize({text: "Vypínám systém a aktivuji
               zabezpečení"});
        turnOff('shutdown')
        break
   }
```

V případě, že jsme se dostali až sem, zpracujeme v cyklu celou promluvu a za pomoci regulérních výrazů zkusíme vyplnit celý stav. Pro potřeby zamítnutí je třeba také pracovat s proměnnou „potvrzeni“.

```
 else if (prikazy[i] == "[ano]") potvrzeni = "ano";
 else if (prikazy[i] == "[ne]") potvrzeni = "ne";
 else {
    reg = /_[a-z]+/gi;
    tmp = prikazy[i].match(reg)[0].substr(1);
    if (prikazy[i].slice(1,5) == "turn") stav.akce=tmp;
    if (prikazy[i].slice(1,2) == "v") stav.vec=tmp;
    if (prikazy[i].slice(1,2) == "m") stav.mistnost=tmp;
 }
```

Pokud jsme věděli místnost i spotřebič a zjišťovali jsme, jaký cíl má uživatel, zpracujeme odpověď dle jeho reakce – aktivujeme spotřebič či se vrátíme do hlavní nabídky.

```
if (potvrzeni == "ano" && stav.mistnost!="" &&   stav.vec!="") {
     if (app[stav.vec][stav.mistnost] == "off") stav.akce="on";
     else stav.akce = "off";
     sendMessage("z "+stav.vec+" "+stav.mistnost+ " " +
          stav.akce)
     zmenHodnotu(stav);
     txt = "Hotovo";
}
else if (potvrzeni == "ne" && (otazka=="ano" ||
  typeof hist.akce==='undefined') ) {
      sc.tts_synthesize({text: "Co pro vás mohu udělat?"});
     zmenMenu('menu', 'polozky')
     potvrzeni = ""
     stav = {akce : '', mistnost : '', vec: ''}
}
```

Pokud uživatel zamítl předchozí akci, např. byla špatně rozpoznaná místnost, aplikace vrátí spotřebič na původní hodnotu. Uživatel mohl zároveň vznést nový požadavek, proto je třeba uchovat si původní cíl v proměnné hist_a, s kterou můžeme nadále pracovat.

```
else if (potvrzeni == "ne") {
  hist_a = hist.akce;
  if (otazka!="skip") {
  hist_a = hist.akce;
  if (hist.akce=="on") hist.akce="off"
  else hist.akce="on"
  hist = zkontrolujVyjimky(hist)
  if (hist.mistnost=="" && hist.vec=="audio") {
      zmenHodnotu(hist)
      sendMessage("z "+hist.vec+" " + hist.akce)
      sc.tts_synthesize({text: "Upravuji požadavek"});
  }
  else if (hist.mistnost=="")
      txt = "Nerozuměla jsem vašemu požadavku."
  else {
      zmenHodnotu(hist)
      sendMessage("z "+hist.vec+" "+hist.mistnost+ " " +
          hist.akce)
      sc.tts_synthesize({text: "Upravuji  požadavek"});
   }
}
```

Pokud uživatel pronesl spolu se zamítnutím nový požadavek, například chtěl místo světla klimatizaci, máme aktuálně vyplněnu pouze jednu položku stavu. Vzhledem k tomu, že ale uživatel nevyřčením jiné místnosti či akce s minulým požadavkem nepřímo souhlasil, doplníme tyto informace z minulého stavu. Posledním krokem je kontrola výjimek.

```
if (stav.akce=="") stav.akce = hist_a
if (stav.vec=="") stav.vec = hist.vec
if (stav.mistnost=="") stav.mistnost = hist.mistnost
stav = zkontrolujVyjimky(stav)
```

Jestliže máme zaplněn celý stav, můžeme zaslat požadavek na server. V případě, že nám nějaké informace chybí, zavoláme funkci generujPromluvu a předáme jí aktuální stav. Ta na jeho základě generuje promluvu, díky které zjistí od uživatele potřebné informace.

```
if ((stav.akce != "" && stav.vec!="" && stav.mistnost!="") ||
 (stav.akce != "" && stav.vec=="audio") ) {
  if (stav.vec=="audio" && app.audio != stav.akce)
          sendMessage("z "+stav.vec+" "+stav.akce)
  else if (stav.vec!="audio" && app[stav.vec][stav.mistnost] !=
   stav.akce)
    sendMessage("z "+stav.vec+" "+stav.mistnost+ " " + stav.akce)
  if (txt=="") txt = generujPromluvu(stav)
  sc.tts_synthesize({text: txt});
}
```

Událost asr_set_grammar_ok

Při prvotním načítání stránky a přihlášení probíhá změna gramatiky. Aby byl uživatel informován o tom, zda může či nemůže mluvit, změníme po příchodu události ikonku mikrofonu a zapneme rozpoznávač.

```
sc.on('asr_set_grammar_ok', function (msg) {
  $("img#recog").attr('src','images/rec_on.png')
  sc.asr_recognize();
});
```

Událost tts_done

U události asr_result jsme pozastavili rozpoznávání pro potřeby uložení nahrávky. Jakmile uživateli podáme reakci, rozpoznávač opět zapneme.

```
sc.on('tts_done', function () {
   sc.asr_recognize();
   $("img#recog").attr('src','images/rec_on.png')
});
```

7 Testování

O tom, zda aplikace splňuje všechny náležitosti, se přesvědčíme pomocí testování. Nejdříve provedeme testování funkčnosti, ve kterém otestujeme aplikaci pomocí nejrůznějších příkazů a budeme sledovat její chování. Testování bude prováděno s externím mikrofonem pro co nejkvalitnější přenos zvuku. Dále aplikaci otestujeme z různých zařízení s odlišnými mikrofony, abychom vyhodnotili vliv mikrofonu na kvalitu rozpoznávání řeči a schopnosti plnit dané úkony.

7.1 Testování funkčnosti

Aplikace by měla být schopna splnit zadaný úkol z jakéhokoli výchozího stavu (samozřejmě kromě stavu, kdy uživatel není přihlášen či je server vypnut). Princip pro práci se světlem, klimatizací a žaluziemi je pro všechny tři spotřebiče stejný, proto můžeme snížit možné přechody mezi stavy a příkazy ze tří pouze na jeden. Pokud aplikace bez problémů obstojí v testování, bylo vše naprogramováno v pořádku a aplikaci může otestovat širší skupina uživatelů.

Označme

- Zeleně výsledky, které jsou v pořádku, a nenastala větší chyba
- Oranžově výsledky, při kterých nastala komplikace, ale aplikace vykonala daný cíl
- Červeně výsledky, při kterých jsme se nedostali do cíle, či cesta trvala příliš dlouho

Test 1

Lokace	světlo		
Výchozí stav	**Akce**	**Místnost**	**Věc**
			svetlo
Požadovaná akce	Rozsvítit světlo v ložnici		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	Rozsvítit v ložnici		
Rozpoznaná promluva	rozsvitit loznici		
Stav po zpracování	**Akce**	**Místnost**	**Věc**
	on	loznice	svetlo
Výstup aplikace	Zapínám světlo v ložnici		
Vykonaná akce	světlo zapnuto		
Chyba	nerozpoznáno v		
Řešení	Netřeba, v gramatice je vloženo jako nepovinné – nemusí se rozpoznat		

Tabulka 7. 1 Test 1

V prvním testu vycházíme z nabídky světlo a pokoušíme se zapnout světlo v ložnici. Vyslovená promluva nebyla rozpoznána zcela správně. Jedná se však pouze o předložku v,

která je v gramatice uvedena jako volitelná, a proto ji rozpoznávač nemusí klasifikovat jako vyslovené slovo, zvláště pokud uživatel požadavek nepronese příliš zřetelně.

Test 2

Lokace	Kuchyně		
Výchozí stav	**Akce**	**Místnost**	**Věc**
		kuchyne	
Požadovaná akce	Vytáhnout žaluzie		
Stav daného spotřebiče	zataženo		
Vyslovená promluva	vytáhnout žaluzie		
Rozpoznaná promluva	vytahnout zaluzie		
Stav po zpracování	**Akce**	**Místnost**	**Věc**
	on	kuchyne	zaluzie
Výstup aplikace	Vytahuji žaluzie v kuchyni		
Vykonaná akce	Žaluzie vytaženy		

Tabulka 7. 2 Test 2

Ve druhém testu jsme se z nabídky kuchyně pokusili vytáhnout žaluzie. Příkaz by byl proveden totožně, i kdybychom se nenacházeli v nabídce kuchyně, jelikož by byla do chybějícího stavu doplněna defaultní místnost. Během testu nedošlo k žádným komplikacím a požadavek byl splněn – žaluzie byly vytaženy.

Test 3

Lokace	Hlavní nabídka		
Výchozí stav	**Akce**	**Místnost**	**Věc**
Požadovaná akce	Rozsvítit světlo v kuchyni		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	rozsvítit světlo		
Rozpoznaná promluva	rozsvitit svetlo		
Stav po zpracování	**Akce**	**Místnost**	**Věc**
	on	kuchyne	svetlo
Výstup aplikace	Zapínám světlo v kuchyni		
Vykonaná akce	Rozsvíceno světlo v kuchyni		

Tabulka 7. 3 Test 3

Ve třetím testu jsme se z hlavní nabídky pokusili rozsvítit světlo. Aplikace příkaz vykonala a rozpoznala bez problémů.

Test 4

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Změnit původní požadavek na klimatizaci		
Stav daného spotřebiče	Světlo zapnuto, klimatizace vypnuta		
Vyslovená promluva	já chtěl klimatizaci		
Rozpoznaná promluva	ja chtel klimatizaci		
Stav po zpracování	Akce	Místnost	Věc
	off	kuchyne	svetlo
	on	kuchyne	klimatizace
Výstup aplikace	Upravuji požadavek. Zapínám klimatizaci v kuchyni		
Vykonaná akce	Vypnuto světlo v kuchyni, zapnuta klimatizace v kuchyni		

Tabulka 7. 4 Test 4

Ve čtvrtém testu jsme změnili požadavek z testu 3, chtěli jsme místo světla zapnout klimatizaci – můžeme si test představit tak, že nám aplikace rozuměla špatně a my již v testu 3 chtěli ovládat klimatizaci. Aplikace náš požadavek zpracovala správně, původní akci změnila zpět a zapnula klimatizaci v dané místnosti. Zamítnutí a změna akce tedy fungují v pořádku.

Test 5

Lokace	hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Změnit požadavek - místnost na ložnici		
Stav daného spotřebiče	Klimatizace v kuchyni zapnuta, v ložnici vypnuta		
Vyslovená promluva	Já chtěl ložnici		
Rozpoznaná promluva	Ja chtel loznici		
Stav po zpracování	Akce	Místnost	Věc
	off	kuchyne	klimatizace
	on	loznice	klimatizace
Výstup aplikace	Upravuji požadavek. Zapínám klimatizaci v ložnici		
Vykonaná akce	Vypnuta klimatizace v kuchyni, zapnuta v ložnici		

Tabulka 7. 5 Test 5

V dalším testu se opět pokusíme změnit požadavek a to ohledně místnosti. Můžeme si představit, že aplikace nerozuměla správně naší místnosti, a proto do příkazu dosadila defaultně nastavenou – kuchyni. Po informování o zapnutí jsme však příkaz zamítli a uvedli vše na správnou míru. Systém tedy vypnul klimatizaci v kuchyni a zapnul ji místo toho v ložnici. Změna místnosti proběhla v naprostém pořádku.

Test 6

Lokace	hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Změnit požadavek – vypnout klimatizaci, vytáhnout žaluzie		
Stav daného spotřebiče	Klimatizace v ložnici zapnuta, žaluzie v obýváku zataženy		
Vyslovená promluva	Já jsem chtěl žaluzie v obýváku		
Rozpoznaná promluva	Ja jsem chtel zaluzie v obyvaku		
Stav po zpracování	Akce	Místnost	Věc
	off	loznice	klimatizace
	on	obyvak	zaluzie
Výstup aplikace	Upravuji požadavek. Vytahuji žaluzie v obýváku		
Vykonaná akce	Vypnuta klimatizace v ložnice, vytaženy žaluzie v obýváku		

Tabulka 7. 6 Test 6

V testu jsme zkombinovali předchozí testy dohromady. Kromě spotřebiče jsme požadovali změnit i místnost. Aplikace požadavek rozpoznala správně a příkaz byl ihned vykonán.

Test 7

Lokace	žaluzie		
Výchozí stav	Akce	Místnost	Věc
			zaluzie
Požadovaná akce	otevřít garáž		
Stav daného spotřebiče	zavřeno		
Vyslovená promluva	Otevři garáž		
Rozpoznaná promluva	Otevri garaz		
Stav po zpracování	Akce	Místnost	Věc
	on	garaz	dvere
Výstup aplikace	Otevírám garáž		
Vykonaná akce	Garáž otevřena		

Tabulka 7. 7 Test 7

V tomto testu vycházíme ze stavu žaluzie – můžeme si to představit tak, že jsme zatáhli dotykem žaluzie v obýváku a jako další příkaz jsme chtěli otevřít dveře od garáže. Aplikace požadavek opět zpracovala bez problémů a garáž byla otevřena.

Test 8

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Změnit požadavek - chtít otevřít hlavní vchod		
Stav daného spotřebiče	Garáž otevřena, hlavní vchod uzavřen		
Vyslovená promluva	Eh, já jsem nechtěl garáž, já chtěl hlavní vchod		
Rozpoznaná promluva	To jsem nechtel garaz, ja jsem chtel hlavni vchod		
Stav po zpracování	Akce	Místnost	Věc
	off	garaz	dvere
	on	hlavni	dvere
Výstup aplikace	Upravuji požadavek. Otevírám hlavní vchod		
Vykonaná akce	Zavřena garáž, otevřen hlavní vchod		
Chyba	Rozpoznána jiná slova		
Řešení	Netřeba, jedná se o výplňová slova		

Tabulka 7. 8 Test 8

Opět pokračujeme z předchozího testu. Původně jsme otevřeli garáž, ale spletli jsme si požadavek. Ve skutečnosti jsme totiž chtěli otevřít hlavní vchod (kde například čeká návštěva). Původní příkaz jsme tedy změnili. Aplikace několik slov rozpoznala nepřesně, naštěstí se jedná pouze o výplňová slova, která na rozpoznání požadované akce nemají vliv. Test dopadl opět v pořádku, garáž byla zavřena a hlavní vchod otevřen.

Test 9

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Otevřít hlavní vchod		
Stav daného spotřebiče	otevřen		
Vyslovená promluva	Otevřít hlavní vchod		
Rozpoznaná promluva	Otevrit hlavni vchod		
Stav po zpracování	Akce	Místnost	Věc
	on	hlavni	dvere
Výstup aplikace	Hlavní vchod je již otevřen		
Vykonaná akce	nic		

Tabulka 7. 9 Test 9

V testu 8 jsme změnili požadavek na otevření vchodu, dejme tomu, že jsme zpětné vazbě od aplikace nerozuměli či ji neslyšeli. Zopakujeme tedy svůj požadavek na otevření hlavního vchodu. Jelikož je již otevřen, dostala se nám pouze správná informace o tom, že požadovaná akce nelze vykonat.

Test 10

Lokace	kuchyně		
Výchozí stav	Akce	Místnost	Věc
		kuchyne	
Požadovaná akce	Informace o klimatizaci		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	klimatizace		
Rozpoznaná promluva	klimatizace		
Stav po zpracování	Akce	Místnost	Věc
		kuchyne	klimatizace
Výstup aplikace	Klimatizace v kuchyni je vypnuta. Chcete ji zapnout?		
Vykonaná akce	Informace o kuchyni		
Požadovaná akce	Potvrdit zapnutí		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	ano		
Rozpoznaná promluva	ano		
Stav po zpracování	Akce	Místnost	Věc
	on	kuchyne	klimatizace
Výstup aplikace	Hotovo		
Vykonaná akce	Zapnuta klimatizace v kuchyni		

Tabulka 7. 10 Test 10

V kuchyni jsme se zeptali na stav klimatizace, která byla vypnuta. Systém nám tuto skutečnost sdělil a poté se zeptal, zda ji chceme zapnout. Po odsouhlasení došlo k zapnutí spotřebiče.

Test 11

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Pustit hudbu		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	Pustit hudbu		
Rozpoznaná promluva	Pustit hudbu		
Stav po zpracování	Akce	Místnost	Věc
	on		audio
Výstup aplikace	Zapínám hudbu		
Vykonaná akce	Hudba puštěna		

Tabulka 7. 11 Test 11

Jednalo se o jednoduchý požadavek stejně jako u prvních testů, který byl v pořádku splněn.

Test 12

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Vypnout všechny spotřebiče		
Stav daného spotřebiče	Klimatizace v kuchyni zapnuta, hudba zapnuta		
Vyslovená promluva	Vše vypni		
Rozpoznaná promluva	prosim zaluzie		
Stav po zpracování	Akce	Místnost	Věc
			zaluzie
Výstup aplikace	Jaký je váš požadavek ohledně žaluzií?		
Vykonaná akce	Otázka na akci		
Vyslovená promluva	Vše vypnout		
Rozpoznaná promluva	Vse vypnout		
Stav po zpracování	Akce	Místnost	Věc
Výstup aplikace	Vypínám všechna zařízení		
Vykonaná akce	Vše vypnuto		

Tabulka 7. 12 Test 12

Aplikace nerozuměla správně požadavku. V gramatice je vyslovená promluva definována, avšak problém nastává v tom, že příkaz vypnout je i u spotřebičů. Při vyslovení slova v infinitivu se již požadavek rozpoznal správně a akce byla vykonána. Skutečnost tohoto problému nabízí otázku ohledně gramatiky – zda konstruovat pravidla jiným způsobem nebo zaměnit klíčová slova pro vypnutí systému. Abychom měli lepší přehled o rozpoznávání tohoto požadavku, byly provedeny následující testovací promluvy.

Vyslovená promluva	**Rozpoznaná promluva**
všechno vypni	chtel jsem vypni
všechno vypni	chtel vypni
vše vypni	prosim vypni
všechno vypnout	vše vypni

Příkazy pro vypnutí nejsou rozpoznávány ideálně a je třeba upravit gramatiku. Vzhledem k tomu, že mnoho podobných slov je v gramatice definováno v jiném kontextu, jeví se jako vhodný příkaz pro vypnutí zařízení nechat pouze slovní spojení „**vypnout systém**“ a všechny ostatní příkazy z gramatiky odstranit.

Test 13

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Otevřít hlavní vchod		
Stav daného spotřebiče	zavřen		
Vyslovená promluva	otevři hlavní vchod		
Rozpoznaná promluva	odemkni hlavni vchod		
Stav po zpracování	Akce	Místnost	Věc
	on	hlavni	dvere
Výstup aplikace	Otevírám hlavní vchod		
Vykonaná akce	Hlavní vchod otevřen		
Chyba	Rozpoznáno odemkni místo otevřít		
Řešení	Odemkni způsobí stejnou akci, čili není třeba nic opravovat		

Tabulka 7. 13 Test 13

V dalším testu jsme opět zkusili jednoduchý příkaz a to otevřít hlavní vchod. Příkaz byl vykonán v pořádku, ovšem i v tomto případě došlo k záměně slov při rozpoznávání. Namísto slova otevři, bylo rozpoznáno slovo odemkni. Naštěstí se jedná o alternativu a výsledek je tak při obou promluvách totožný.

Test 14

Lokace	Hlavní nabídka		
Výchozí stav	Akce	Místnost	Věc
Požadovaná akce	Změnit požadavek – chtít zapnout audio		
Stav daného spotřebiče	Hlavní vchod otevřen, audio vypnuto		
Vyslovená promluva	Já jsem nechtěl dveře, já chtěl pustit hudbu		
Rozpoznaná promluva	ja jsem nechtel dvere prosim hudbu		
Stav po zpracování	Akce	Místnost	Věc
	off	hlavni	dvere
	on		audio
Výstup aplikace	Upravuji požadavek. Zapínám hudbu		
Vykonaná akce	Zavřeny hlavní dveře, puštěna hudba		
Chyba	Špatně rozpoznaný požadavek		
Řešení	Upravit gramatiku, aby nešlo zaměnit prosím s pustit		

Tabulka 7. 14 Test 14

Dále jsme zkusili změnit požadavek na puštění hudby, abychom zjistili, zda i přechody mezi těmito dvěma spotřebiči fungují v pořádku. Příkaz byl rozpoznán a vykonán jako ve všech ostatních případech správně, jediný problém nastal opět v gramatice, ve které došlo k záměně slov. I přesto však vše proběhlo bez problémů.

Vhodným krokem je upravit gramatiku, aby nenabízela možnost slova prosím.

Test 15

Lokace	Audio		
Výchozí stav	Akce	Místnost	Věc
			audio
Požadovaná akce	Zjistit co chceme v kuchyni		
Stav daného spotřebiče	Světlo, klimatizace i žaluzie vypnuty		
Vyslovená promluva	kuchyně		
Rozpoznaná promluva	kuchyne		
Stav po zpracování	Akce	Místnost	Věc
		kuchyne	
Výstup aplikace	Co chcete v kuchyni udělat?		
Vykonaná akce	Žádost o dodatečné informace		
Požadovaná akce	Zjistit co chceme v kuchyni zapnout		
Stav daného spotřebiče	Světlo, klimatizace i žaluzie vypnuty		
Vyslovená promluva	zapnout		
Rozpoznaná promluva	svetlo		
Stav po zpracování	Akce	Místnost	Věc
	on	kuchyne	
Výstup aplikace	Co chcete v kuchyni zapnout?		
Vykonaná akce	Dotázání na spotřebič		
Požadovaná akce	Zapnout světlo v kuchyni		
Stav daného spotřebiče	vypnuto		
Vyslovená promluva	světlo		
Rozpoznaná promluva	svetlo		
Stav po zpracování	Akce	Místnost	Věc
	on	kuchyne	svetlo
Výstup aplikace	Zapínám světlo v kuchyni		
Vykonaná akce	Světlo zapnuto		

Tabulka 7. 15 Test 15

V závěrečném testu jsme vyzkoušeli, jak si aplikace poradí s doplňováním stavů a změnou místnosti, když původní požadavek vzešel ze sekce audio, kde místnost není definována. Aplikace si poradila v pořádku a postupnými otázkami zjistila požadovanou akci.

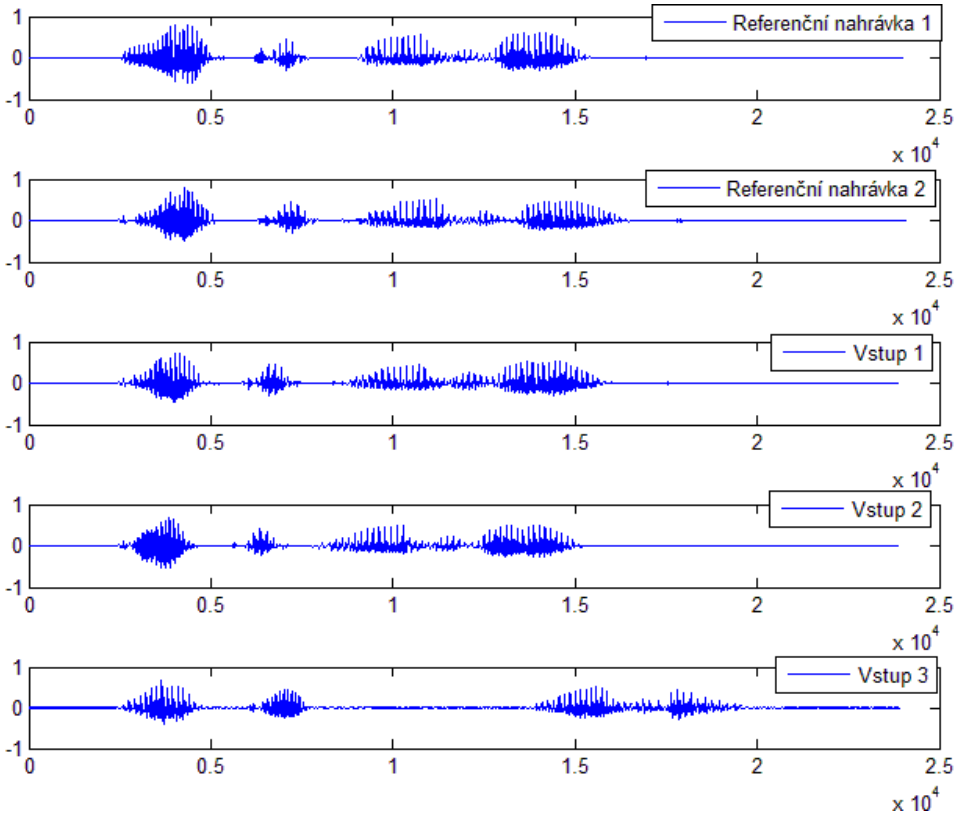
Testy ukázaly, že pokud uživatel bude pokládat logické požadavky a takovým způsobem, aby jim aplikace rozuměla (byly definovány v gramatice), aplikace vždy dojde do požadovaného cíle a to i v případě, kdy se rozpozná špatný cíl, jelikož lze jedním příkazem původní akci vrátit zpátky a zároveň vykonat akci novou.

7.2 Aplikace autentifikace uživatele v inteligentní domácnosti

V části autentifikace uživatele jsme na několika nahrávkách vyhodnotili chování systému na základě pravděpodobností shody referenčních nahrávek s nahrávkou na vstupu. Po zakomponování této funkce do aplikace je vhodné vyhodnotit chování, zda odpovídá původním poznatkům. Pro potřeby aplikace budeme brát uživatele za správně rozpoznaného při 80% pravděpodobnosti shody a 70% věrohodnosti výsledku.

Nahrávky pořízené skrze SpeechCloud mají vzorkovací frekvenci 16 000 Hz. Po otestování několika nahrávek vyšlo najevo, že velikost vzorkovací frekvence je zásadní pro správnou funkčnost.

Pro testy využijeme celkem 5 nahrávek, první dvě budou referenční, se kterými budeme porovnávat jednotlivé vstupy. Signály nahrávek jsou znázorněny na obrázku 7.1.



Obr. 7. 1 Srovnání všech nahrávek

Nejdříve provedeme testování pro nahrávky se vzorkovací frekvencí 16 000 Hz. V tabulce 7.16 jsou uvedeny pravděpodobnosti a věrohodnosti pro jednotlivé vstupy.

Nahrávka	Pravděpodobnost	Věrohodnost
Vstup 1	94,87%	29,62 %
Vstup 2	99,91%	91,41%
Vstup 3	0,005%	97,97%

Tabulka 7. 16 Autentifikace uživatele pro 2 referenční nahrávky

Nyní zkusme převzorkovat nahrávky na poloviční frekvenci, tedy 8 000 Hz. Výsledky jsou znázorněny v tabulce 7.17.

Nahrávka	Pravděpodobnost	Věrohodnost
Vstup 1	99,99%	97,99%
Vstup 2	99,97%	97,63%
Vstup 3	99,99%	97,49%

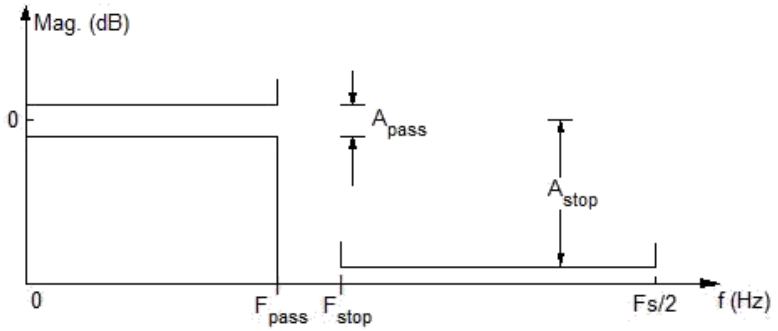
Tabulka 7. 17 Autentifikace uživatele pro 3 referenční nahrávky

Z testů je patrné, že pokud v nahrávkách necháme vyšší frekvence než 8 000 Hz, vyhodnocení bude ovlivněno nežádoucími frekvencemi a uživatel nebude správně rozpoznán.

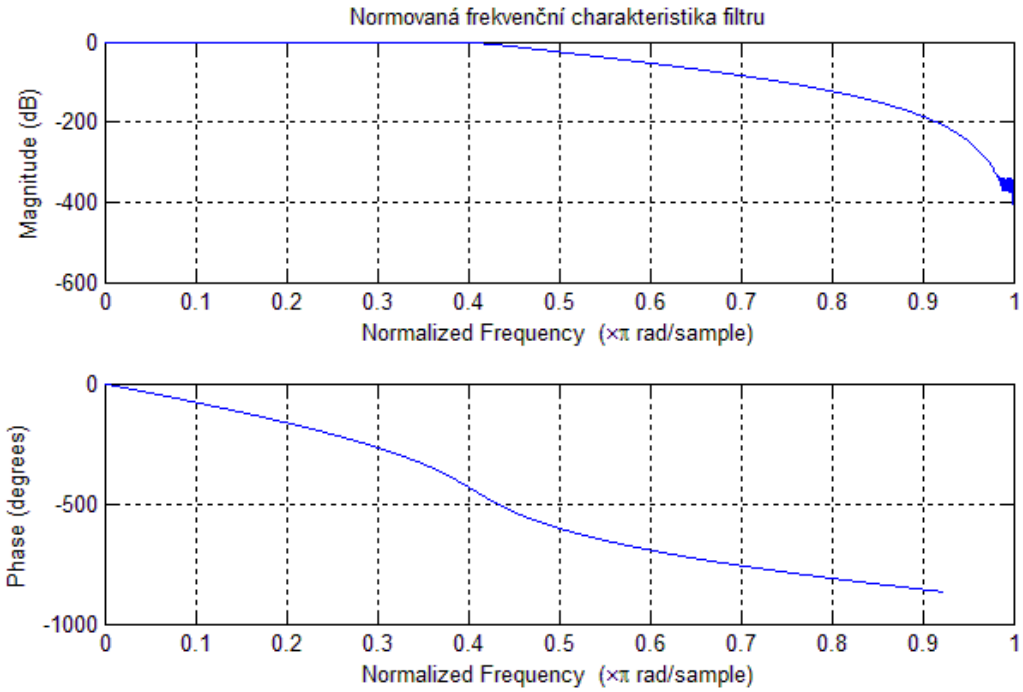
Převzorkování na frekvenci 8 000 Hz docílíme pomocí dvou kroků. Nejdříve provedeme filtraci signálu pomocí IIR filtru. IIR filtr aproximuje frekvenční charakteristiku podílem polynomů, čímž poskytuje velmi dobrou aproximaci signálu při nízkém řádu. Jelikož chceme odfiltrovat vyšší frekvence, využijeme filtr typu dolní propust, díky kterému ze signálu odstraníme frekvence vyšší než 4 000 Hz. Frekvence lidské řeči tuto hodnotu nepřekračuje. Abychom potlačili nežádoucí frekvence, navrhneme Butterworthův filtr. Při návrhu využijeme následující parametry:

- F_{pass} - mezní frekvence propustného pásma = 3 000 Hz
- F_{stop} - maximální přípustná tolerance = 5 000 Hz
- A_{pass} - mezní frekvence nepropustného pásma = 0.6 dB
- A_{stop} - tolerance v nepropustném pásmu = 60 dB

První dva parametry udávají úsek, na kterém chceme dosáhnout útlumu. Čím menší interval, tím rychleji začne filtr potlačovat frekvence. Druhé dva parametry určují, jak rychle k útlumu dojde. Čím vyšší hodnoty nastavíme, tím strmější bude přechod. Při požadavku rychlejší filtrace na krátkém úseku roste řád filtru, což zvyšuje výpočetní nároky. Vizualizaci návrhu můžeme vidět na obrázku 7.2. Při aktuálním nastavení jsme vytvořili filtr řádu 10, který dokáže dostatečně potlačit nežádoucí frekvence.

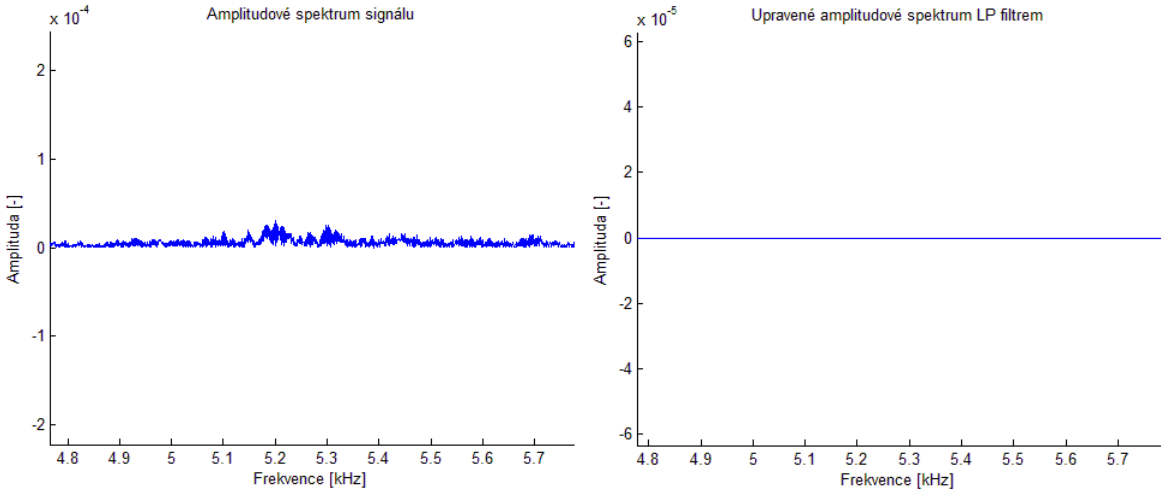


Obr. 7. 2 Návrh IIR filtru



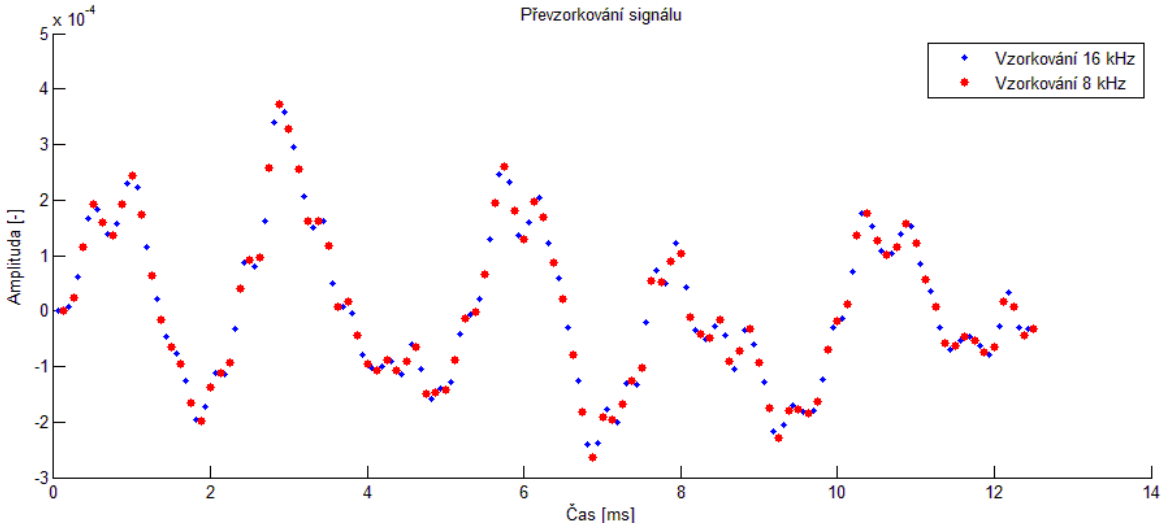
Obr. 7. 3 Frekvenční charakteristika filtru

Na obrázku 7.3 je zobrazena frekvenční charakteristika námi navrženého filtru. Z obrázku je patrné, že k filtraci dochází již přibližně v bodě 0.41, což odpovídá frekvenci 3 280 Hz. Výsledek filtrace můžeme vidět na obrázku 7.4.



Obr. 7. 4 Srovnání frekvenčních spekter před a po filtraci

Dalším krokem je převzorkování signálu, které provedeme pomocí lineární interpolace. Ta proloží dva sousední body přímkou, na které v polovině definujeme náš požadovaný bod. Převzorkovaný signál má poloviční frekvenci, čili k rekonstrukci postačí 2x méně hodnot. Grafické znázornění signálu je na obrázku 7.5.



Obr. 7. 5 Srovnání původního a převzorkovaného signálu

Funkce buttord spočítá koeficienty pro Butterworthův filtr (řád a frekvenci). Vstupní frekvenci je třeba zadat v normovaném formátu (rozsah 0 až 1), je tedy třeba frekvence převést

$$f_n = \frac{2 \cdot f}{f_s}$$

```
%načtení dat
[x, Fs] = wavread('file16.wav');

% Filtrace signálu
[N,omegaN] = buttord(0.375,0.625,0.6,60)
[z,p] = butter(N,omegaN,'low');
y = filter(z,p,x);

Y = 2*abs(fft(y)/ length(y));
Y = Y(1:length(Y)/2);
df = Fs/ length(y);

% Interpolace na frekvenci 8 kHz
Fp = 8e03;
Tp = 1/Fp;
Ts = 1/Fs;
delkaB = floor((Fp/Fs)* length(y));

for i = 1:delkaB-1
   pomerF = floor(i*(Fs/Fp));
   t1 = Ts*pomerF;
   t2 = Ts*(pomerF+1);
   k = (y((pomerF+1))-y(pomerF))/(t2-t1);
   b(i) = (((i*Tp)-t1)*k)+y(pomerF);
end

wavwrite(b,Fp,'file.wav');
```

Kód pro převzorkování nahrávky je napsán v programu Matlab. Pomocí funkce mcc překonvertujeme funkci do jazyka C a vytvoříme tak spustitelný program Sample.exe. Ten budeme volat v Pythonu z funkce ziskejWav, jakmile obdržíme nahrávku ze serveru SpeechCloudu. Program voláme stejným způsobem, jako jsme volali program pro autentifikaci uživatele.

```
args = "Sampling.exe users/file16.wav users/file.wav"
output = subprocess.Popen(args).communicate()[0]
```

7.3 Uživatelské testování

Vzhledem k tomu, že je systém v prvotní fázi vývoje, je potřeba zjistit aktuální uživatelskou spokojenost a dle získaných poznatků aplikaci přizpůsobit. Díky logování každého testu budeme mít přístupné jednotlivé nahrávky, pomocí kterých můžeme zopakovat konkrétní test a zjistit tak přesně, jaká chyba nastala a zda se nám ji podařilo úspěšně odstranit.

7.3.1 Návrh testování

Funkčnost systému byla již ověřena v kapitole 7.1. Kromě uživatelské spokojenosti se také zaměříme na funkčnost aplikace v závislosti na zařízení a prohlížeči. Tyto poznatky budou velmi důležité pro další vývoj a doporučení k užívání. Pro aplikaci byl vytvořen základní uživatelský manuál (viz přílohu 1), který by měl uživatelům dostatečně vysvětlit, jak s aplikací zacházet. Po provedení několika jednoduchých úkonů budou uživatelé vyzváni k vyplnění formuláře, pomocí kterého nám sdělí své postřehy a dojmy. Vyhodnocení formuláře bude výstupem z uživatelského testování.

Při zpracování výsledků formuláře bude třeba nahlédnout do uloženého logu a dohledat chybu, která nastala, abychom mohli posoudit relevantnost daného výstupu. Pro co nejjednodušší procházení logu můžeme využít jednoduchého scriptu, který zobrazí na jednom místě jak časové informace (kdy byl log pořízen), které se mohou srovnat s časovými informacemi odeslání formuláře, tak samotný záznam (log) ze SpeechCloudu. Pomocí tlačítek se následně můžeme pohybovat mezi dalšími uloženými logy. Session ID získáme ze souboru logs.txt, do kterého se ukládaly pomocí PHP scriptu. Pomocí jQuery a funkce load můžeme načíst obsah logu do HTML stránky, viz obrázek 7.6.

```
Poslední log | Novější | Starší
Časová značka:2016-05-02 16:48:04
Otevřít v novém okně
```

```
{
 "$schema":https://cak.zcu.cz:9443/v1/schema/session/response/0",
 "sessions": [
   {
   "audio_records": [
   {
    "filename": "log/20160502-144809-1.wav",
    "tstamp": "2016-05-02T14:48:09",
    "type": "asr",
    "uri":"https://cak.zcu.cz:9443/v1/session/tMvcQgffjeczmPbTm5bB3f
            /audio_records/572768b0949d8c006017f539"
   }
],
```

Obr. 7. 6 Aplikace pro procházení uložených relací

HTML

```
<a href="#" onclick="nacti('pos')">Poslední log</a>  |
<a href="#" onclick="nacti('new')">Novější</a>  |
<a href="#" onclick="nacti('old')">Starší</a>  <br /> <br />
<div id="info"> </div>
<hr />
<div id="zobraz">  </div>
```

JavaScript

```
var i = 1;
var lines;
function nacti(co) {
  if (co == "pos") i=1
  else if (co=="new") i = i-1
  else i = i+1
  if (typeof lines[lines.length-i] === "undefined") {
    $("#zobraz").html("<h1>Nic zde není</h1>")
    $("#info").html("<br />")
  } else {
    zk = lines[lines.length-i].split('|')
    $("#zobraz").load(zk[1])
    $("#info").html("Časová značka:"+ zk[0]+"<br />
         <a href='"+zk[1]+"' target='_blank'>
              Otevřít v novém okně</a>")
  }
}
jQuery.get('logs.txt', function(data) {
    lines = data.split('\n');
    zk = lines[lines.length-1].split('|')
    $("#zobraz").load(zk[1])
    $("#info").html("Časová značka:"+ zk[0]+"<br />
         <a href='"+zk[1]+"' target='_blank'>
              Otevřít v novém okně</a>")
});
```

7.3.2 Výsledky testu

Testování se zúčastnilo celkem 34 lidí z několika věkových skupin, přičemž průměrný věk byl mezi 20 a 25 lety. Žen v testu bylo 13 a mužů 21. Nejčastějším zařízením, na kterém byla aplikace spuštěna, byl počítač (27 lidí), zbylých 6 lidí použilo smartphone.

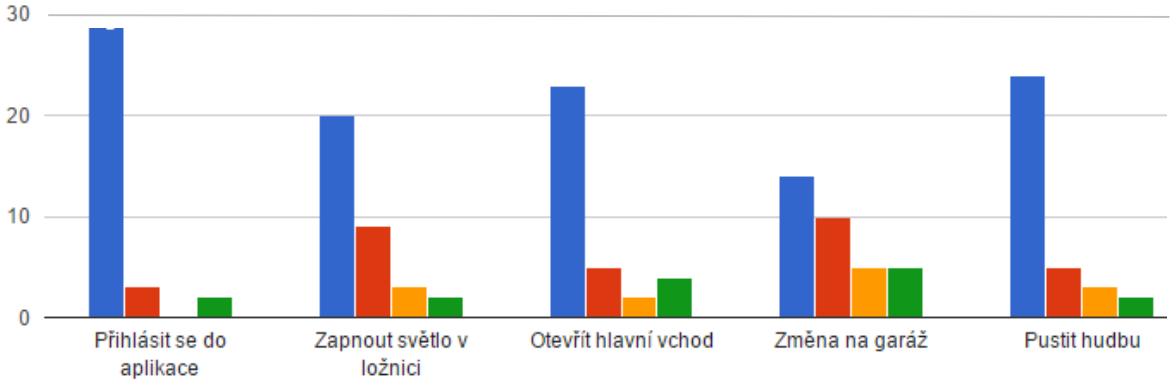
Z hodnocení je značně patrné, že uživatelé se smartphonem měli větší problémy s rozpoznáváním hlasu vzhledem k citlivějšímu mikrofonu na těchto zařízeních.

Jak napovídá obrázek 7.7, návod na ovládání aplikace byl pro všechny uživatele dostačující a vše z něj pochopili. Jediné, co v návodu postrádali, byla gramatika pro odhlášení.

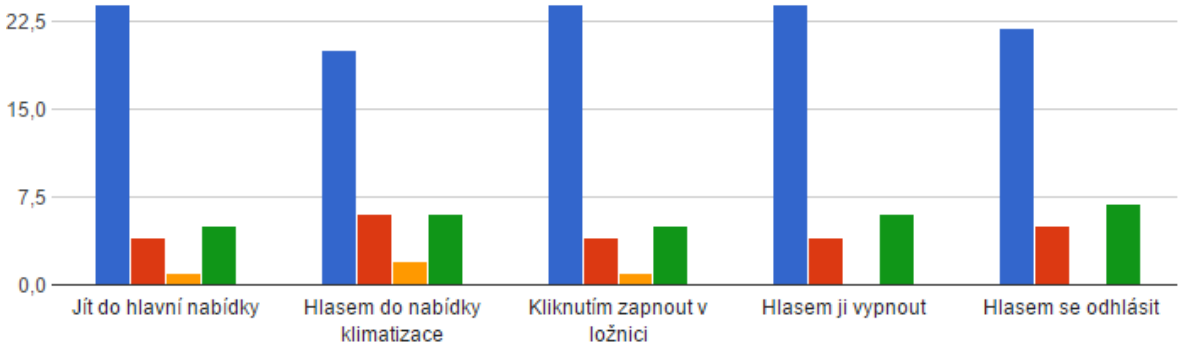
Obr. 7. 7 Hodnocení uživatelského manuálu

Při testování uživatelé měli vykonat celkem 10 příkazů, na kterých zhodnotili funkčnost aplikace. Testování bylo rozděleno na dvě části, v první (viz obrázek 7.8) se testovalo pouze hlasové ovládání, v druhé (viz obrázek 7.9) byla kombinace hlasového a dotykového ovládání. Barvy v grafech znázorňují následující hodnocení

- Modrá - aplikace mi bez problémů rozuměla, úkol byl hned splněn
- Červená - aplikace něčemu nerozuměla, ale bez problémů jsem se dostal/a do cíle
- Oranžová - aplikace téměř ničemu nerozuměla, bylo to velmi obtížné
- Zelená - úkol se mi nepodařilo splnit



Obr. 7. 8 Hodnocení hlasového ovládání



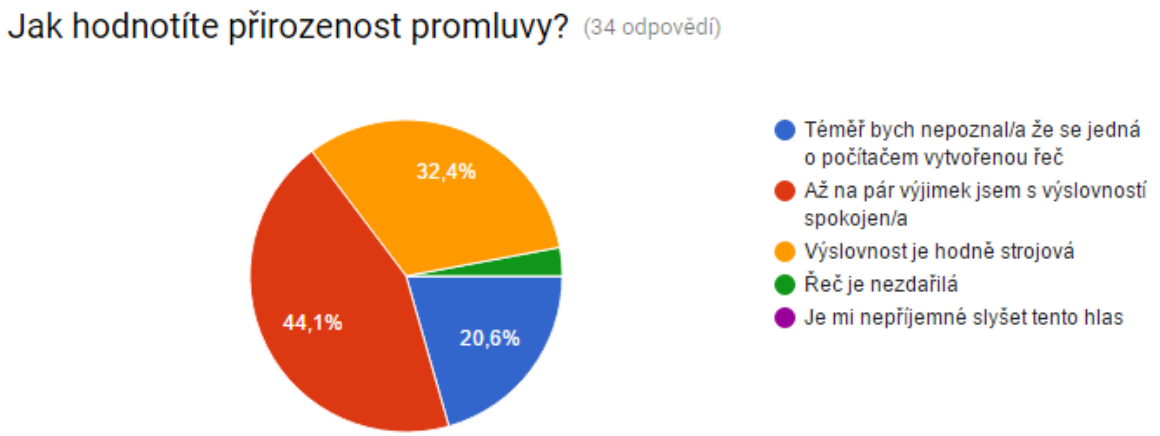
Obr. 7. 9 Hodnocení kombinovaného ovládání

Ze získaných dat je zarážející počet lidí, kteří se nedokázali bez problémů dostat do hlavní nabídky. Jedná se celkem o 10 uživatelů, kteří vůbec nedokázali splnit úkol „kliknutí na tlačítko“, které bylo popsáno v přiloženém návodu. Návod přitom všichni uživatelé dle předchozí odpovědi pochopili. Vyšší počet lidí, kterým se nepodařilo kliknutím klimatizaci zapnout, může být způsoben faktem, že se do klimatizace nemuseli vůbec dostat, na základě čehož nemohli klimatizaci ani hlasem vypnout.

Obr. 7. 10 Celkové shrnutí ovládání

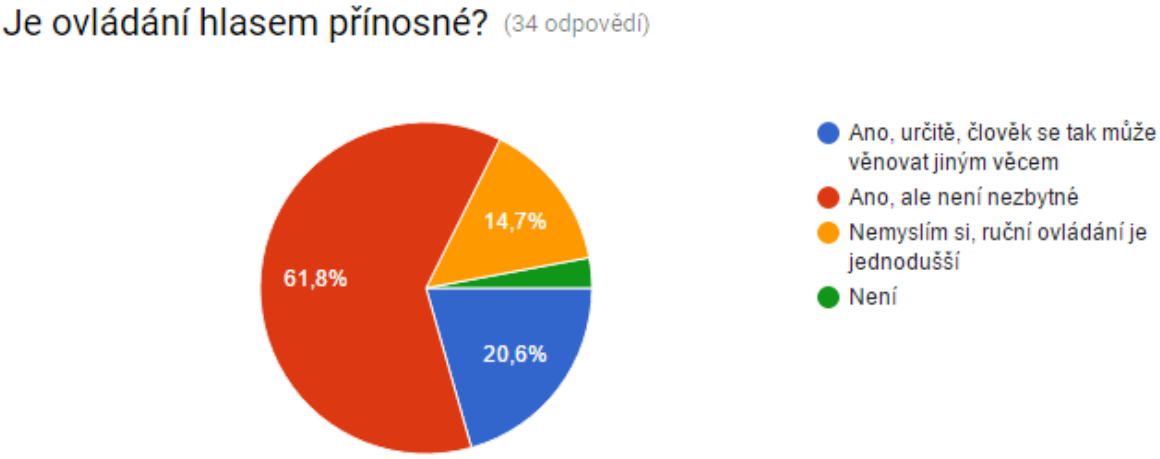
Na obrázku 7.8 můžeme vidět celkové hodnocení reakcí na požadavky. Celkem dvěma lidem se aplikace nepodařila spustit. Toto mohlo být zapříčiněno několika způsoby: uživatelé nepochopili správně návod, uživatelé neměli povolený mikrofon a nešel tak do aplikace žádný vstup, a posledním důvodem je, že uživatelé se neřídili přesně pokyny. Tyto dvě negativní odpovědi bychom si pomyslně mohli odečíst z grafů na obrázku 7.8 a 7.9 a také z celkového hodnocení, ke kterému se dostaneme za chvíli. Bohužel již nemůžeme soudit, jaké možnosti vybrali v ostatních odpovědích.

Modul syntézy řeči uživatelé hodnotili vesměs kladně, 7 lidí dokonce tvrdí, že je řeč totožná s řečí člověka. Hodnocení nalezneme na obrázku 7.11.



Obr. 7. 11 Hodnocení syntézy řeči

V další otázce byli uživatelé tázáni na přínos ovládání pomocí řeči. Nejednalo se pouze o naší konkrétní aplikaci, ale otázka byla mířena obecně. Naprostá většina uživatelů hodnotí ovládání pomocí řeči pozitivně, nemyslí si však, že by bylo nutností. Shrnutí je zobrazeno na obrázku 7.12.



Obr. 7 12 Přínos hlasového ovládání

Nyní se již dostáváme k celkovému zhodnocení aplikace. Uživatelé hodnotili propracovanost a ovladatelnost aplikace na stupnici od jedné do deseti, přičemž deset bylo nejvíce. Aplikace získala celkem **7.34** bodů, což je velmi dobré hodnocení na to, že se jedná teprve o prototyp.

Poslední část hodnocení tvoří komentáře, ve kterých uživatelé mohli poskytnout zpětnou vazbu na jednotlivé části systému. Nejvíce byl zmiňován šum (dýchání, větrák, hudba a hluk na pozadí) a jeho vyhodnocování jako řeči. Aplikace tak samovolně ovládala některé části domu, i když uživatel do přístroje zrovna nemluvil. Někteří uživatelé si myslí, že aplikace velmi usnadní práci a mohou snadno ovládat domácnost z jednoho místa. Jiní takovému způsobu ovládání zase vytýkají, že lidé zpohodlní a zleniví.

8 Závěr

Tato práce se zabývá moderními přístupy ovládání domácnosti. V průběhu psaní práce jsem se seznámil s úlohou tvorby dialogových systémů a inteligentních domácností. Následně na základě těchto poznatků a s pomocí softwarových komponent z katedry kybernetiky a firmy SpeechTech jsem navrhl a realizoval vlastní systém, který byl nakonec otestován širokou veřejností.

V kapitole 2 byl popsán aktuální stav vývoje aplikací pro inteligentní domácnosti a diskutován fakt, že možnost hlasového ovládání v těchto aplikacích stále chybí a to i přesto, že se jedná o uživatelsky nejpřívětivější způsob komunikace.

8.1 Úloha tvorby hlasových dialogových systémů

V kapitole 3 byla popsána problematika týkající se hlasových dialogových systémů. Vzhledem k tomu, že v dnešní době stále ještě nejsme schopni stoprocentně vyhodnotit spontánní řeč (potřeba obrovského množství dat a expertních znalostí), nejsme tudíž schopni ani reagovat na dané požadavky. Také je důležité myslet na poskytování zpětné vazby uživateli. Bohužel opakování celé promluvy je zdlouhavé a v případě, že chybí mnoho informací, je tento proces pro uživatele obtěžující místo přínosný. Z tohoto důvodu je vhodné využít dalšího způsobu přístupu k informacím – a tou může být vizuální podoba.

8.2 Návrh systému pro použití v domácnosti

Vzhledem k potřebě poskytování vizuálních informací uživateli a faktu, že inteligentní domácnosti využívají k ovládání moderní technologie, jako jsou tablety či smartphony (viz kapitolu 2), byl pro naprogramování aplikace zvolen jazyk HTML5. Podoba a možnosti moderních webových aplikací byly sepsány v kapitole 4. Díky programovacímu jazyku JavaScript se stává aplikace interaktivní a ihned reaguje na veškeré úkony. Právě tato skutečnost poskytuje uživateli příjemné grafické prostředí a zároveň zpětnou vazbu, jelikož vidí, v jaké části aplikace, a tedy i stavu dialogu, se nachází.

8.2.1 Vypracování základního popisu komponent

V kapitole 5 byly popsány komponenty, které vznikly na katedře kybernetiky a ve firmě SpeechTech. Popis komponenty pro autentifikaci uživatele je konkrétnější a více informativní, jelikož jsem se osobně mohl podílet na vývoji aplikace a tato problematika mi tak byla bližší.

SpeechCloud je nové aplikační rozhraní, které zastřešuje komponenty ASR a TTS a zpřístupňuje je pomocí moderních webových služeb – WebSockety. Vzhledem k tomu, že sepsání dokumentace nebylo předmětem zadání a probíhá mimo diplomovou práci, není v práci popis uveden na stejné detailní úrovni, jako tomu bylo v případě popisu rozhraní komponenty autentifikace uživatele. Základní informace o funkčnosti, aktuálních modulech a jejich verzích a o tom, jak se k aplikaci připojit, však v práci nechybí.

Veškeré podklady získané z kapitoly 5 poslouží k vytvoření komplexní dokumentace, která bude zahrnovat další služby a aplikace, které byly v průběhu posledních let vytvořeny.

8.2.2 Otestování uživatelské autentifikace

Součástí dokumentace uživatelské autentifikace bylo i testování funkčnosti. Pro vyhodnocení byla zvolena metoda F1 skóre a grafické znázornění pomocí ROC křivky. Z testu bylo zjištěno, že je třeba brát v potaz kromě pravděpodobnosti shody i věrohodnost výsledku. Nastavení těchto hodnot závisí na potřebách aplikace a jejím zabezpečení. Jako vhodná konstrukce hesla pro přístup do systému se jeví jedno až dvě slova celkem o pěti až dvanácti slabikách.

8.2.3 Návrh a realizace aplikace

V kapitole 6 byla navržena aplikace, která za pomocí HTML5 zajišťuje komunikaci s několika zařízeními a poskytuje uživateli zpětnou vazbu. O veškeré požadavky ohledně rozpoznávání a syntézy řeči se stará server SpeechCloudu. Rozpoznané požadavky následně putují na server, který je vytvořen skrze Tornado Framework. Tento server se stará o vykonávání již konkrétních požadavků zaslaných z webové aplikace. Kromě zpracování požadavků na ovládání domácnosti se také stará o ověření uživatele. Požadavky týkající se ovládání domácnosti server dále zasílá do zařízení Raspberry PI, konkrétně desky UniPi, která je se serverem propojena pomocí ethernetového kabelu a stará se o fyzické provedení daného úkonu.

8.3 Vyhodnocení systému

8.3.1 Otestování systému

Navržený systém byl následně v kapitole 7 otestován. Nejprve proběhla série testů, která prověřila navrženou funkčnost z kapitoly 6. Prvotní testy neodhalily žádný zásadní problém a aplikace tak byla zaslána širší skupině lidí, kteří provedli testování a zhodnotili navržený systém.

Při opětovném testování systému autentifikace řečníka byl zjištěn problém vzorkování signálu a jeho vliv na výsledné pravděpodobnosti. Z tohoto důvodu byla do systému přidána funkce na převzorkování signálu na 8 kHz pomocí filtrace a interpolace signálu.

Poslední částí testování bylo vyhodnocení aplikace širokou škálou uživatelů. Aplikace získala hodnocení **7.34/10**, což je velmi hezké hodnocení a ukazuje na zájem o podobné aplikace ze strany spotřebitelů.

8.3.2 Otestování rozpoznávače

Kromě samotných funkcí byl otestován také rozpoznávač řeči, který při využití externího mikrofonu a klíčových slov z gramatiky zafungoval správně. Problém nastal při použití integrovaných mikrofonů, které nejsou schopny odfiltrovat šum z pozadí. Vzhledem k benevolentně zvolenému prahu pravděpodobnosti akustického modelu (viz kapitolu 3.1) se následně i naprosto odlišná slova rozpoznávají jako slova pocházející z aplikaci předané gramatiky.

8.3.3 Testovací data

Veškerá získaná data (nahrávky, výsledky rozpoznávání) budou podrobně zpracována (anotace, pojmenování a evidence) a uložena na pozdější využití. Na základě doporučení pro úpravu daných modulů se tato data následně využijí na provedení identických testů, při kterých se bude moci porovnat původní aplikace s aktualizovanou. Tato data tedy budou sloužit jako základní testovací množina pro sledování pokroku vývoje a evaluaci nových knihoven. Data také poslouží k rozšíření gramatiky aplikace.

8.4 Doporučení pro další vývoj

Aby nedocházelo k nechtěnému rozpoznávání spontánní mluvy na pozadí, je třeba definovat, kdy má aplikace rozpoznávat a kdy ne. Vhodné by bylo aktivovat aplikaci klíčovým slovem, které zapne rozpoznávač řeči. Pokud je systém delší dobu nečinný, mohl by se automaticky přepnout opět do režimu naslouchání, ve kterém by čekal na aktivaci.

SpeechCloud je velmi užitečný nástroj, který vývojáři usnadňuje práci, avšak bylo by vhodné mít přístup k základním nastavením, které jsou pro aplikaci velmi důležité. Aktuálně lze pouze nastavovat gramatika, bohužel se však nedá nijak nastavit práh pro zamítnutí dané promluvy, který by určil, do jaké míry se jedná či nejedná o dané slovo z gramatiky. Aplikace tak rozpoznává libovolnou promluvu jako slova, která jsou definována gramatikou.

Seznam literatury

[1] AMX – Inteligentní dům. *Insight Home: Technologie pro inteligentní dům, inteligentní domy a digitální domácnost*. [online]. 03.2011 [cit. 2016-01-26]. Dostupné z: http://www.insighthome.eu

[2] Samsung's 4-Door Flex Refrigerator's 21.5-inch touchscreen has it all. *Digital Trends*. [online]. 05.01.2016 [cit. 2016-01-26]. Dostupné z: http://digitaltrends.com

[3] Welcome Smart Home Camera With Face Recognition. *Netatmo*. [online]. 2016 [cit. 2016-01-26]. Dostupné z: https://www.netatmo.com

[4] iNELS BUS System. *iNELS.cz*. [online]. 2016 [cit. 2016-01-26]. Dostupné z: http://inels.cz

[5] Internet věcí: Co musíte vědět už nyní. *BusinessIT*. [online] 01.2015 [cit. 2016-02-24]. Dostupné z: http://www.businessit.cz

[6] Kepka, Jiří; Psutka, Josef. *Expertní systémy: Umělá inteligence*. 1.vyd. Plzeň: ZČU, 1994. ISBN 80-7082-135-3

[7] Psutka, Josef. *Komunikace s počítačem mluvenou řečí*. Praha: Academia, 1995. ISBN 80-200-0203-0

[8] Nedvěd, Jakub. *Evaluace hlasových dialogových systémů*. Plzeň, 2013. Bakalářská práce (Bc.). Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Jan Švec.

[9] Habiballa Hashim. *Regulární a bezkontextové jazyky II* [online]. Ostrava 2005 [cit. 2016-01-28]. Dostupné z: http://osu.cz

[10] Speech Recognition Grammar Specification Version 1.0. *W3C*. [online]. Publikováno 16.03.2004 [cit. 2016-01-29]. Dostupné z: http://www.w3.org

[11] .Webové aplikace. *Jak na internet*. [online]. 2014 [cit. 2016-04-28]. Dostupné z: http://www.jaknainternet.cz

[12] Lokální uložiště. *Dive into HTML5*. [online]. 2014 [cit. 2016-04-28]. Dostupné z: http://kniha.html5.cz

[13] Service Name and Transport Protocol Port Number Registry. *Internet Assigned Numbers Authority*. [online]. 22.04.2016 [cit. 2016-04-28]. Dostupné z: http://www.iana.org

[14] Struktura dokumentu. *Jak psát web, návod na HTML stránky*. [online]. 04.2016 [cit. 2016-04-28]. Dostupné z: http://www.jakpsatweb.cz

[15] JavaScript HTML DOM. *W3Schools Online Web Tutorials*. [online]. 2016 [cit. 2016-04-28]. Dostupné z: http://www.w3schools.com

[16] Web Sockets. *HTML Standard*. [online]. 27.04.2016 [cit. 2016-04-28]. Dostupné z: https://html.spec.whatwg.org

[17] Psutka, Josef et al. *Mluvíme s počítačem česky*. Vyd. 1. Praha: Academia, 2006. 746 s. Česká matice technická; roč. 111, č. spisu 502. ISBN 80-200-1309-1.

[18] Furui, S. *Speaker Recognition*. [online] Scholarpedia, 3(4):3715, ISSN 1941-6016 [cit. 2016-04-28]. Dostupné online: http://scholarpedia.org

[19] Kunešová, Marie. *Optimalizace algoritmů pro zpracování řečového signálu*. Plzeň, 2011. Bakalářská práce (Bc.). Západočeská univerzita v Plzni, Fakulta aplikovaných věd. Vedoucí práce Ing. Jan Vaněk, Ph.D.

[20] Sattar, Abdul, eds. *AI 2006 advances in artificial intelligence: 19th Australian Joint Conference on Artificial Intelligence*, Hobart, Australia, December 4-8, 2006:proceedings. Berlin: Springer, 2006. ISBN 9783540497882.

[21] Rylich, Jan. *Cloudové služby: data i počítače v oblacích. Ikaros* [online]. 2012, ročník 16, číslo 9 [cit. 2016-04-28]. urn:nbn:cz:ik-13965. ISSN 1212-5075. Dostupné z: http://ikaros.cz

[22] Vyzkoušeli jsme mikropočítač Raspberry Pi. *Živě.cz*. [online]. 12.09.2012 [cit. 2016-04-28]. Dostupné z: http://www.zive.cz

[23] Co je to řídicí systém REX. *Řídicí systém REX*. [online]. 2016 [cit. 2016-04-28]. Dostupné z: https://www.rexcontrols.cz

Seznam obrázků

Seznam tabulek

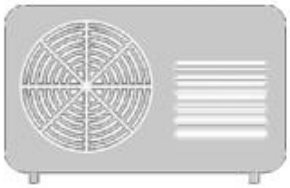
Seznam příloh

Inteligentní domácnost

Právě se vám dostala do ruky ukázková aplikace, která vznikla v rámci Diplomové práce na katedře Kybernetiky na ZČU.

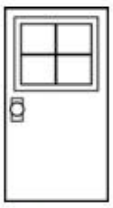
Jedná se o uživatelské rozhraní k ovládání domácnosti. Domácnost lze ovládat dvěma způsoby – hlasem či dotykem. Oba způsoby se vzájemně doplňují.

Světlo



Klimatizace

Žaluzie



Dveře

Audio

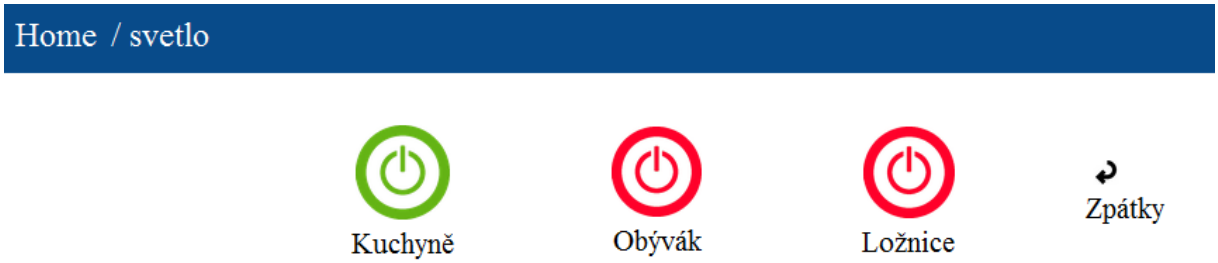
Uzamknout

Jak to funguje?

Řeč je přenášena na server, kde probíhá vyhodnocení a rozpoznání řečené promluvy. Server poté zašle promluvu webové stránce a zobrazí ji v boxu Rozpoznaná promluva.

Pokud tedy řekneme například „rozsviť světlo v kuchyni“, server nám vrátí následující text

Promluva v hranatých závorkách – např. [turn_on] slouží pro aplikaci běžící na webové stránce, která na základě rozpoznaných příkazů provede určitou akci – v tomto případě například rozsvítí světlo v kuchyni.



Ovládání

Aplikace uživateli poskytuje hlasovou i grafickou odezvu, pokud se například rozpozná promluva špatně a aplikace bude rozumět pouze, že chcete světlo, položí doplňující otázku, ve které se dotáže na váš konkrétní cíl. Pro aplikaci jsou zásadní 3 informace

- akce – vypnout, zapnout
- místnost – kuchyně, obývák, ložnice, garáž, přední / zadní vchod
- věc – světlo, klimatizace, žaluzie, dveře, hudba

Pokud ovládáme světlo, žaluzie či klimatizaci a neřekneme místnost, aplikace bere jako defaultní hodnotu kuchyni (=aktuální místnost, v které se nacházíme).

Pokud je v místnosti příliš rušno, je možné, že aplikace bude rozpoznávat i hluk z pozadí. Pokud se tak stane, můžeme tlačítkem ovládat, kdy má aplikace rozpoznávat promluvu a kdy nikoli.

Aplikace při načítání využívá přeškrtnutý mikrofon jako signalizaci, že je třeba počkat (vždy při načítání stránky a během hlasového výstupu). Jakmile se mikrofon změní na nepřeškrtnutý, aplikace je připravena a můžete ji ovládat. Rychlost zpracování záleží na vašem zařízení, na kterém aplikaci zkoušíte – může to být otázka vteřiny, ale také klidně i deseti.

Gramatika

Aplikace bohužel není člověk a zná tak pouze základní slova. Abychom mohli něco ovládat, musíme říct příkaz, který aplikace pozná. Pokud aplikace slovo nezná, nerozpozná se (viz rozpoznaná promluva), ale místo toho rozpozná slovo, které se dle daných parametrů s vámi vyřčeným slovem nejvíce shoduje.

Aplikace by měla znát následující slova

Akce – vypnout, zapnout, vypni, zapni, rozsvítit, rozsvit, zhasnout, zhasni

Místnost – kuchyně , obývák, ložnice, garáž, vrata, přední / hlavní / zadní vchod/dveře

Věc – světlo, klimatizace, žaluzie, dveře, hudba, audio, rádio

Zamítnutí – ne, nechci

Potvrzení – ano, jo

Přihlášení – přihlásit (se), připojit

Ukázková promluva

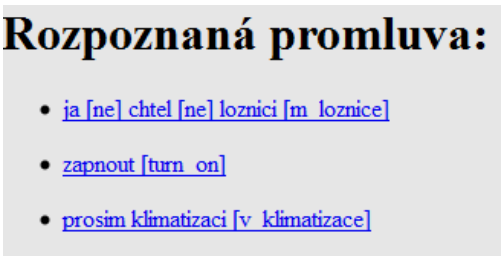
V této ukázce aplikace rozuměla všemu, co jsme řekli, žádné slovo nebylo špatně rozpoznáno

- **Uživatel**: rozsvit světlo v kuchyni
- **Aplikace**: zapínám světlo v kuchyni
- **Uživatel**: zapni klimatizaci
- **Aplikace**: zapínám klimatizaci v kuchyni
- **Uživatel**: já chtěl v ložnici
- **Aplikace**: zapínám klimatizaci v ložnici

Ukázková promluva 2

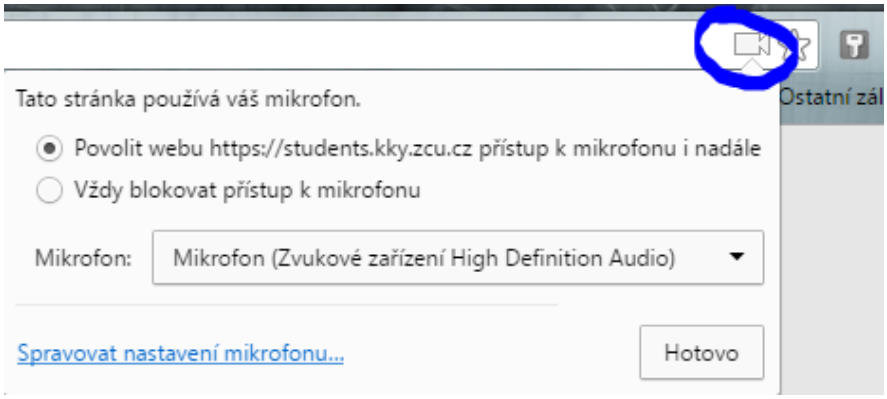
Nyní aplikace rozpoznala špatně některá slova, cílového stavu jsme však stejně dosáhli.

- **Uživatel**: zapni klimatizaci
- **Aplikace**: jaký je váš požadavek ohledně klimatizace?
- **Uživatel**: zapnout
- **Aplikace**: zapínám klimatizaci v kuchyni
- **Uživatel**: já chtěl v ložnici
- **Aplikace**: zapínám klimatizaci v ložnici

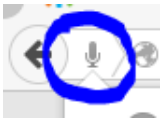


Testování

Prvním krokem je povolení přístupu aplikace k mikrofonu. Po otevření webové stránky https://students.kky.zcu.cz/nedvedj/index.html by se vám měl objevit následující dotaz.



Ikonka (v modrém kroužku) se může v různých prohlížečích lišit. V Mozille Firefox to je například mikrofon, který je na druhý straně před samotnou adresou stránky.



Pokud se dotaz neobjeví, klikněte na ikonku a vyberte mikrofon, který budete s aplikací sdílet. Aplikaci lze ovládat pouze pokud máte funkční mikrofon.

Nejdříve se pokuste s aplikací seznámit, zadat pár příkazů či ovládat aplikaci pomocí dotyku. Až budete s aplikací seznámeni, proveďte prosím následující test. Vyhodnocení testu proveďte pomocí Google formuláře.

A. Hlasové ovládání

1. Přihlásit se do aplikace
2. Zapnout světlo v ložnici
3. Otevřít hlavní vchod
4. Zamítnout, chtít garáž
5. Pustit hudbu

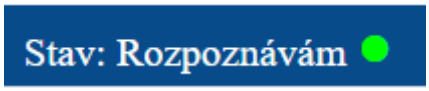
Nyní klikněte na Uzamknout a poté na ikonu Vše vypnout

B. Kombinované ovládání

1. Přihlásit se kliknutím
2. Kliknout na světlo
3. Hlasem rozsvítit obývák
4. Chtít hlasem místo světla klimatizaci
5. Hlasem se odhlásit

Mikrofon

Pokud test provádíte bez headsetu (sluchátka s mikrofonem), váš mikrofon bude zřejmě příliš citlivý a aplikace bude rozpoznávat šumy na pozadí jako promluvu. Zda aplikace rozpoznává, si můžete ověřit pomocí aktuálního stavu.



Výstup aplikace pak může být následovný

- prosim zastav [turn_off] nechci [ne] audio [v_audio] mohla pustit [turn_on] klimatizaci [v_klimatizace] nechci [ne] zastav [turn_off] nechci [ne] garaz [m_garaz] mohla rozsvitit [turn_on]

V tomto případě prosím zapínejte a vypínejte rozpoznávání pomocí tlačítek

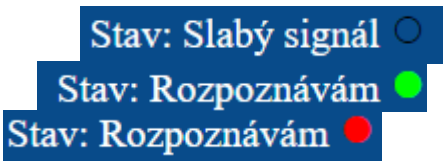
Stavy

informační panel dává zpětnou vazbu o tom, v jakém stavu se systém nachází. Je definováno několik stavů:

- Nepřipojeno – uživatel není přihlášen
- Připojeno – proběhlo přihlášení uživatele
- Rozpoznávač zapnut – aplikace rozpoznává vstup
- Rozpoznávač vypnut – aplikace nerozpoznává vstup

Pokud je rozpoznávač zapnut, uživatel dále dostává zpětnou vazbu o síle signálu pomocí barevného kolečka, které signalizuje daný stav. Definované stavy jsou

- Prázdné kolečko – žádný vstup či příliš slabý signál
- Zelené kolečko – aplikace rozpoznává řeč
- Červené kolečko – uživatel mluví příliš nahlas



Pokud svítí jiné kolečko než zelené, je vysoce pravděpodobné, že řeč nebude správně rozpoznána.

Test 1, kapitola 5.1.4.1

Test pro 6 vzorových souborů							
Hasičská vzájemná pojišťovna							
	Vzorů	**1**	**2**	**3**	**4**	**5**	**6**
Muž 1	**pravděpodobnost**	93,560	99,862	99,967	99,989	99,995	99,967
	věrohodnost	25,280	89,568	95,249	96,975	97,325	97,560
Muž 2	**pravděpodobnost**	70,776	81,700	99,575	99,889	99,956	99,974
	věrohodnost	26,740	59,262	79,300	91,188	94,727	95,914
Muž 3	**pravděpodobnost**	59,893	97,740	99,820	99,794	99,913	99,944
	věrohodnost	59,890	49,730	88,117	87,528	92,083	93,696
Žena 1	**pravděpodobnost**	97,223	99,691	99,916	99,975	99,982	99,989
	věrohodnost	43,741	82,741	92,049	95,681	96,158	96,747
Žena 2	**pravděpodobnost**	99,740	99,988	99,994	99,998	99,999	99,997
	věrohodnost	83,998	96,551	97,187	97,758	97,865	97,646
Žena 3	**pravděpodobnost**	64,044	91,278	98,763	99,909	99,977	99,926
	věrohodnost	23,463	53,225	70,031	91,493	95,764	93,738
Průměr	**pravděpodobnost**	**80,873**	**95,043**	**99,673**	**99,926**	**99,970**	**99,966**
	věrohodnost	**43,852**	**71,846**	**86,989**	**93,437**	**95,654**	**95,884**
Babička							
	Vzorů	**1**	**2**	**3**	**4**	**5**	**6**
Muž 1	**pravděpodobnost**	99,796	99,972	99,993	99,998	99,999	99,999
	věrohodnost	86,234	95,354	97,085	97,738	97,782	97,871
Muž 2	**pravděpodobnost**	32,899	98,562	99,967	96,843	99,417	99,937
	věrohodnost	64,827	69,391	95,009	96,080	87,063	94,484
Muž 3	**pravděpodobnost**	29,605	52,795	92,343	96,986	92,608	97,465
	věrohodnost	97,533	55,827	62,880	73,734	67,382	73,941
Žena 1	**pravděpodobnost**	99,517	99,970	99,994	99,998	99,996	99,997
	věrohodnost	76,794	95,141	97,200	97,616	97,439	97,588
Žena 2	**pravděpodobnost**	98,795	99,966	99,980	99,991	99,995	99,995
	věrohodnost	61,427	95,003	96,044	96,869	97,316	97,333
Žena 3	**pravděpodobnost**	46,708	50,403	57,41	58,747	90,866	67,898
	věrohodnost	89,44	31,942	49,896	43,389	34,084	54,87
Průměr	**pravděpodobnost**	**67,887**	**83,611**	**91,614**	**92,094**	**97,147**	**94,215**
	věrohodnost	**79,376**	**73,776**	**83,019**	**84,238**	**80,178**	**86,015**
Brána							
	Vzorů	**1**	**2**	**3**	**4**	**5**	**6**
Muž 1	**pravděpodobnost**	54,144	90,561	99,698	99,975	99,984	99,990
	věrohodnost	18,190	72,633	88,216	95,967	96,501	96,923
Muž 2	**pravděpodobnost**	25,769	85,249	62,065	82,099	92,537	97,270
	věrohodnost	93,165	32,059	47,072	49,992	75,905	72,127
Muž 3	**pravděpodobnost**	73,251	98,652	99,757	99,909	99,964	99,968
	věrohodnost	37,442	65,834	86,060	92,374	95,281	95,490
Žena 1	**pravděpodobnost**	99,838	72,348	92,428	98,967	99,908	99,974
	věrohodnost	87,956	65,125	76,245	82,683	93,649	96,179
Žena 2	**pravděpodobnost**	24,442	99,724	99,988	99,992	99,998	99,998
	věrohodnost	54,679	87,207	96,815	97,147	97,661	97,736
Žena 3	**pravděpodobnost**	92,866	99,692	99,861	99,694	99,949	99,951
	věrohodnost	23,795	83,842	89,756	87,446	93,935	94,103

Příloha 2 Výsledky testování autentifikace uživatele

Průměr	**pravděpodobnost**	**61,718**	**91,038**	**92,299**	**96,773**	**98,723**	**99,525**
	věrohodnost	**52,538**	**67,783**	**80,694**	**84,268**	**92,155**	**92,093**
	Ano						
	Vzorů	**1**	**2**	**3**	**4**	**5**	**6**
Muž 1	**pravděpodobnost**	87,323	98,489	99,251	99,770	99,857	99,962
	věrohodnost	12,940	60,842	72,918	88,166	91,683	95,497
Muž 2	**pravděpodobnost**	67,427	62,151	83,225	87,290	86,329	91,365
	věrohodnost	18,201	62,242	41,727	59,549	61,555	60,111
Muž 3	**pravděpodobnost**	98,035	99,828	99,833	99,951	99,982	99,966
	věrohodnost	51,010	88,456	89,427	94,084	96,186	95,320
Žena 1	**pravděpodobnost**	19,270	91,595	99,786	99,988	99,994	99,990
	věrohodnost	40,438	71,073	87,011	96,699	97,203	96,912
Žena 2	**pravděpodobnost**	94,552	99,488	99,861	99,967	99,694	99,860
	věrohodnost	34,002	77,437	91,063	95,510	89,040	92,658
Žena 3	**pravděpodobnost**	56,834	66,412	88,233	81,888	85,563	89,338
	věrohodnost	27,170	45,536	46,598	54,882	69,898	71,458
Průměr	**pravděpodobnost**	**70,574**	**86,327**	**95,032**	**94,809**	**95,236**	**96,747**
	věrohodnost	**30,627**	**67,598**	**71,457**	**81,482**	**84,261**	**85,326**

Test 2, kapitola 5.1.4.2

Test identifikace uživatele									
Hasičská vzájemná pojišťovna									
[%]	Uživ	Muž 1	Muž 1	Muž 2	Muž 2	Žena 1	Žena 1	Žena 2	Žena 2
M1	ppst	88,577	47,366	0,147	0,131	0,002	0,007	0,003	0,004
	věr	14,421	0,041	95,227	95,529	98,014	97,950	98,000	97,989
M1	ppst	94,471	96,145	0,122	0,084	0,004	0,004	0,005	0,003
	věr	28,029	35,840	95,707	96,458	97,987	97,994	97,978	98,001
M2	ppst	0,130	0,039	1,021	0,642	0,001	0,001	0,000	0,000
	věr	95,554	97,341	80,620	86,489	98,032	98,030	98,036	98,037
M2	ppst	0,062	0,085	0,637	1,951	0,001	0,001	0,001	0,001
	věr	96,888	96,437	86,577	68,513	98,028	98,028	98,033	98,035
Ž1	ppst	0,003	0,022	0,002	0,003	49,358	48,863	0,033	0,018
	věr	98,011	97,677	98,024	98,010	0,002	0,008	97,461	97,748
Ž1	ppst	0,010	0,008	0,003	0,003	83,130	86,675	0,019	0,017
	věr	97,888	97,934	98,009	98,012	8,666	12,007	97,729	97,778
Ž2	ppst	0,004	0,005	0,001	0,000	0,024	0,024	99,933	99,896
	věr	97,997	97,974	98,034	98,038	97,636	97,640	92,722	90,680
Ž2	ppst	0,004	0,003	0,001	0,000	0,022	0,012	98,893	97,514
	věr	97,993	98,011	98,030	98,038	97,674	97,869	62,951	45,708
Usr 1	ppst	0,011	0,008	0,002	0,001	0,008	0,005	0,036	0,075
	věr	97,872	97,937	98,024	98,031	97,928	97,973	97,400	96,643
Usr 2	ppst	0,002	0,005	0,000	0,001	0,006	0,047	0,313	0,139
	věr	98,016	97,976	98,036	98,032	97,963	97,196	92,127	95,387
Usr 3	ppst	0,003	0,003	0,001	0,001	0,054	0,099	0,303	0,038
	věr	98,008	98,002	98,028	98,026	97,044	96,163	92,308	97,378
Usr 4	ppst	0,001	0,002	0,002	0,002	0,417	0,022	1,864	0,362
	věr	98,027	98,018	98,024	98,017	90,282	97,676	69,528	91,260
Usr 5	ppst	0,007	0,006	0,000	0,002	0,065	0,170	0,078	0,063
	věr	97,953	97,965	98,036	98,024	96,824	94,790	96,585	96,870
Usr 6	ppst	0,001	0,002	0,002	0,002	0,017	0,033	0,023	0,029
	věr	98,034	98,023	98,016	98,016	97,778	97,475	97,652	97,554
Usr 7	ppst	0,001	0,006	0,001	0,001	0,011	0,000	0,006	0,003
	věr	98,031	97,964	98,025	98,032	97,872	98,037	97,963	98,003
Usr 8	ppst	0,029	0,117	0,028	0,065	0,001	0,005	0,002	0,001
	věr	97,536	95,804	97,568	96,826	98,027	97,982	98,023	98,026
Usr 9	ppst	0,224	0,194	0,025	0,012	0,002	0,003	0,004	0,001
	věr	93,773	94,326	97,620	97,855	98,016	98,009	97,995	98,025
Usr 10	ppst	0,009	0,004	0,001	0,000	0,093	0,042	0,184	0,041
	věr	97,910	97,994	98,035	98,037	96,281	97,282	94,521	97,301

Příloha 2 Výsledky testování autentifikace uživatele

Babička									
[%]	Uživ	Muž 1	Muž 1	Muž 2	Muž 2	Žena 1	Žena 1	Žena 2	Žena 2
M1	ppst	99,252	98,446	0,070	13,982	0,004	0,005	0,001	0,001
	věr	70,079	56,037	96,735	15,381	97,991	97,976	98,028	98,032
M1	ppst	99,988	99,688	0,702	21,968	0,003	0,001	0,001	0,001
	věr	96,598	82,100	85,510	6,906	98,002	98,030	98,032	98,025
M2	ppst	2,153	0,378	99,879	95,636	0,003	0,001	0,001	0,000
	věr	66,245	90,974	89,863	33,101	98,011	98,031	98,030	98,036
M2	ppst	0,114	0,010	0,798	3,468	0,002	0,003	0,000	0,000
	věr	95,866	97,895	84,003	53,841	98,015	98,009	98,037	98,039
Ž1	ppst	0,001	0,002	0,000	0,001	97,294	99,318	0,044	0,270
	věr	98,025	98,016	98,036	98,028	43,797	71,608	97,248	92,922
Ž1	ppst	0,001	0,003	0,000	0,001	99,519	82,178	0,019	0,069
	věr	98,029	98,001	98,037	98,032	76,833	7,953	97,738	96,758
Ž2	ppst	0,007	0,005	0,002	0,002	1,126	3,373	56,869	89,094
	věr	97,952	97,977	98,025	98,018	79,097	54,621	0,269	15,183
Ž2	ppst	0,002	0,000	0,001	0,005	1,072	0,053	97,746	93,726
	věr	98,024	98,036	98,031	97,979	79,875	97,080	47,912	25,428
Usr 1	ppst	0,003	0,001	0,003	0,008	0,015	0,010	0,041	0,222
	věr	98,007	98,028	98,009	97,927	97,815	97,899	97,300	93,799
Usr 2	ppst	0,005	0,002	0,001	0,001	0,073	0,008	0,089	0,118
	věr	97,981	98,025	98,031	98,025	96,669	97,930	96,362	95,789
Usr 3	ppst	0,003	0,005	0,002	0,001	4,102	0,030	0,039	0,059
	věr	98,008	97,976	98,024	98,034	49,043	97,519	97,355	96,944
Usr 4	ppst	0,001	0,001	0,001	0,001	48,835	9,066	8,108	7,167
	věr	98,035	98,035	98,030	98,025	0,008	26,087	29,161	32,666
Usr 5	ppst	0,010	0,001	0,000	0,002	0,473	0,020	0,030	0,008
	věr	97,893	98,028	98,038	98,023	89,317	97,715	97,521	97,929
Usr 6	ppst	0,010	0,003	0,002	0,001	0,020	0,007	0,000	0,004
	věr	97,893	98,000	98,023	98,028	97,715	97,941	98,036	97,986
Usr 7	ppst	0,386	0,031	0,001	0,010	0,000	0,001	0,003	0,004
	věr	90,825	97,506	98,030	97,899	98,038	98,030	98,011	97,996
Usr 8	ppst	1,700	0,157	0,007	5,729	0,001	0,001	0,000	0,000
	věr	79,906	95,028	97,953	39,209	98,033	98,025	98,037	98,039
Usr 9	ppst	0,037	0,105	0,009	0,040	0,007	0,004	0,000	0,002
	věr	97,379	96,036	97,910	97,334	97,952	97,987	98,036	98,014
Usr 10	ppst	0,000	0,003	0,000	0,002	0,041	3,995	0,005	0,508
	věr	98,036	98,010	98,036	98,024	97,308	49,807	97,983	88,723

Příloha 2 Výsledky testování autentifikace uživatele

Brána									
[%]	Uživ	Muž 1	Muž 1	Muž 2	Muž 2	Žena 1	Žena 1	Žena 2	Žena 2
M1	ppst	90,087	99,461	0,011	0,012	0,004	0,007	0,002	0,001
	věr	16,802	75,208	97,879	97,868	97,990	97,951	98,012	98,028
M1	ppst	95,647	99,082	0,013	0,005	0,006	0,006	0,003	0,002
	věr	33,159	66,477	97,847	97,973	97,968	97,958	98,010	98,024
M2	ppst	0,005	0,004	0,159	4,660	0,001	0,001	0,001	0,001
	věr	97,971	97,990	94,993	45,313	98,035	98,033	98,030	98,028
M2	ppst	0,005	0,005	0,111	1,825	0,002	0,002	0,006	0,011
	věr	97,982	97,984	95,929	69,997	98,021	98,013	97,966	97,882
Ž1	ppst	0,001	0,002	0,007	0,004	0,701	92,176	0,000	0,003
	věr	98,033	98,023	97,949	97,994	85,533	21,094	98,036	98,002
Ž1	ppst	0,001	0,002	0,010	0,008	1,209	73,657	0,002	0,017
	věr	98,028	98,016	97,901	97,924	77,937	3,602	98,015	97,769
Ž2	ppst	0,007	0,002	0,001	0,001	0,002	0,001	99,813	99,763
	věr	97,942	98,021	98,027	98,026	98,023	98,031	86,885	84,839
Ž2	ppst	0,007	0,001	0,001	0,006	0,003	0,001	99,047	94,884
	věr	97,950	98,031	98,034	97,959	98,012	98,033	65,784	29,667
Usr 1	ppst	0,011	0,000	0,006	0,019	0,004	0,004	0,008	0,034
	věr	97,874	98,036	97,959	97,740	97,991	97,998	97,927	97,456
Usr 2	ppst	0,002	0,005	0,021	0,015	0,616	0,030	0,704	0,561
	věr	98,022	97,978	97,698	97,809	86,908	97,520	85,481	87,829
Usr 3	ppst	0,002	0,001	0,001	0,036	0,182	0,007	0,123	0,033
	věr	98,021	98,034	98,035	97,409	94,557	97,944	95,696	97,467
Usr 4	ppst	0,003	0,014	0,038	0,037	0,183	0,182	0,071	0,247
	věr	98,006	97,834	97,362	97,394	94,539	94,555	96,716	93,344
Usr 5	ppst	0,000	0,002	0,002	0,005	0,015	0,088	0,012	0,014
	věr	98,037	98,015	98,024	97,981	97,808	96,375	97,867	97,823
Usr 6	ppst	0,003	0,002	0,008	0,006	0,042	0,020	0,476	0,025
	věr	98,003	98,024	97,935	97,966	97,283	97,720	89,266	97,620
Usr 7	ppst	0,315	0,101	0,013	0,002	0,002	0,001	0,002	0,002
	věr	92,098	96,130	97,839	98,023	98,013	98,032	98,025	98,023
Usr 8	ppst	0,000	0,002	0,000	0,009	0,000	0,000	0,000	0,000
	věr	98,038	98,020	98,038	97,917	98,039	98,038	98,039	98,039
Usr 9	ppst	0,010	0,027	0,218	0,808	0,003	0,009	0,030	0,031
	věr	97,894	97,576	93,875	83,842	98,002	97,916	97,534	97,505
Usr 10	ppst	0,000	0,000	0,008	0,028	0,008	0,003	0,001	0,001
	věr	98,037	98,038	97,936	97,570	97,927	98,012	98,029	98,031

Příloha 2 Výsledky testování autentifikace uživatele

Ano									
[%]	Uživ	Muž 1	Muž 1	Muž 2	Muž 2	Žena 1	Žena 1	Žena 2	Žena 2
M1	ppst	24,849	2,42	0,008	0,007	0,000	0,001	0,013	0,001
	věr	5,133	65,303	97,923	97,943	98,039	98,027	97,838	98,033
M1	ppst	96,448	8,776	0,031	0,006	0,001	0,003	0,003	0,003
	věr	37,667	26,973	97,505	97,959	98,033	98,011	98,005	98,012
M2	ppst	0,004	0,007	36,123	1,267	0,002	0,000	0,000	0,000
	věr	97,998	97,945	1,257	77,123	98,020	98,038	98,038	98,037
M2	ppst	0,008	0,006	0,994	29,228	0,003	0,001	0,001	0,005
	věr	97,930	97,960	81,013	3,167	98,010	98,030	98,029	97,975
Ž1	ppst	0,000	0,000	0,001	0,001	9,066	68,676	0,000	0,008
	věr	98,037	98,038	98,027	98,035	26,086	2,113	98,038	97,929
Ž1	ppst	0,000	0,000	0,000	0,000	39,833	99,695	0,008	0,018
	věr	98,038	98,039	98,037	98,039	0,644	82,347	97,921	97,748
Ž2	ppst	0,001	0,002	0,000	0,001	0,024	0,012	49,595	99,859
	věr	98,027	98,020	98,038	98,035	97,649	97,864	0,001	88,923
Ž2	ppst	0,001	0,001	0,000	0,001	0,009	0,001	6,405	99,920
	věr	98,029	98,030	98,039	98,026	97,912	98,033	35,929	92,003
Usr 1	ppst	0,004	0,000	0,000	0,000	0,002	0,005	0,016	0,011
	věr	97,990	98,038	98,039	98,036	98,015	97,975	97,790	97,887
Usr 2	ppst	0,001	0,001	0,001	0,001	0,010	0,004	0,637	0,345
	věr	98,029	98,031	98,030	98,034	97,901	97,998	86,575	91,559
Usr 3	ppst	0,004	0,005	0,013	0,070	0,002	0,003	0,008	0,026
	věr	97,993	97,978	97,849	96,729	98,023	98,009	97,931	97,598
Usr 4	ppst	0,001	0,008	0,002	0,007	0,074	0,687	0,017	0,236
	věr	98,030	97,937	98,020	97,948	96,653	85,759	97,780	93,537
Usr 5	ppst	0,008	0,004	0,000	0,003	0,011	0,067	0,012	1,027
	věr	97,933	97,992	98,038	98,002	97,886	96,797	97,868	80,529
Usr 6	ppst	0,012	0,001	0,001	0,001	0,008	0,000	0,005	0,010
	věr	97,856	98,035	98,028	98,033	97,932	98,038	97,976	97,900
Usr 7	ppst	0,133	0,256	0,001	0,009	0,002	0,002	0,000	0,005
	věr	95,494	93,168	98,031	97,912	98,024	98,017	98,038	97,974
Usr 8	ppst	0,023	0,012	0,004	0,002	0,001	0,000	0,000	0,000
	věr	97,656	97,862	97,997	98,013	98,033	98,038	98,037	98,039
Usr 9	ppst	0,003	0,003	0,004	0,002	0,000	0,000	0,001	0,000
	věr	98,001	98,010	97,997	98,014	98,037	98,039	98,035	98,036
Usr 10	ppst	0,003	0,001	0,004	0,001	0,001	0,016	0,357	0,689
	věr	98,011	98,033	97,998	98,028	98,026	97,783	91,344	85,725

Test 3, kapitola 5.1.4.3

Test pro identifikaci uživatele					
Hasičská vzájemná pojišťovna					
[%]	Uživatel	Muž 1	Muž 2	Žena 1	Žena 2
Muž 1	pravděpodobnost	99,091	4,68	0,030	0,028
	věrohodnost	66,667	45,184	97,525	97,560
Muž 1	pravděpodobnost	99,877	2,304	0,044	0,017
	věrohodnost	89,756	64,616	97,253	97,770
Muž 2	pravděpodobnost	1,227	22,551	0,008	0,004
	věrohodnost	77,677	6,509	97,928	97,999
Muž 2	pravděpodobnost	0,966	32,774	0,008	0,005
	věrohodnost	81,433	2,039	97,932	97,980
Žena 1	pravděpodobnost	0,052	0,013	97,000	0,274
	věrohodnost	97,088	97,849	41,465	92,835
Žena 1	pravděpodobnost	0,050	0,014	99,798	0,136
	věrohodnost	97,128	97,823	86,248	95,444
Žena 2	pravděpodobnost	0,019	0,003	0,239	99,994
	věrohodnost	97,727	98,002	93,492	97,219
Žena 2	pravděpodobnost	0,019	0,005	0,206	99,949
	věrohodnost	97,742	97,970	94,109	93,690
Usr 1	pravděpodobnost	0,048	0,008	0,048	1,492
	věrohodnost	97,179	97,922	97,171	74,121
Usr 2	pravděpodobnost	0,014	0,006	0,292	6,019
	věrohodnost	97,826	97,955	92,511	37,752
Usr 3	pravděpodobnost	0,016	0,012	1,943	2,726
	věrohodnost	97,792	97,863	68,606	60,376
Usr 4	pravděpodobnost	0,011	0,011	2,466	17,187
	věrohodnost	97,872	97,885	62,933	11,134
Usr 5	pravděpodobnost	0,049	0,015	3,941	1,448
	věrohodnost	97,153	97,801	50,201	74,702
Usr 6	pravděpodobnost	0,005	0,024	0,196	0,239
	věrohodnost	97,982	97,637	94,296	93,485
Usr 7	pravděpodobnost	0,009	0,008	0,022	0,019
	věrohodnost	97,913	97,934	97,687	97,732
Usr 8	pravděpodobnost	0,410	0,615	0,020	0,013
	věrohodnost	90,408	86,923	97,712	97,847
Usr 9	pravděpodobnost	5,6	0,248	0,013	0,015
	věrohodnost	42,880	93,326	97,840	97,810
Usr 10	pravděpodobnost	0,032	0,002	0,854	2,015
	věrohodnost	97,482	98,013	83,125	67,780

Příloha 2 Výsledky testování autentifikace uživatele

Babička					
[%]	Uživatel	Muž 1	Muž 2	Žena 1	Žena 2
Muž 1	pravděpodobnost	99,978	62,289	0,029	0,008
	věrohodnost	95,683	0,875	97,548	97,935
Muž 1	pravděpodobnost	99,998	89,344	0,015	0,008
	věrohodnost	97,685	15,570	97,815	97,935
Muž 2	pravděpodobnost	50,743	99,964	0,008	0,005
	věrohodnost	0,003	94,652	97,931	97,980
Muž 2	pravděpodobnost	0,317	36,511	0,017	0,002
	věrohodnost	92,070	1,181	97,779	98,015
Žena 1	pravděpodobnost	0,008	0,005	99,960	4,664
	věrohodnost	97,923	97,981	94,436	45,290
Žena 1	pravděpodobnost	0,013	0,007	99,965	0,656
	věrohodnost	97,850	97,942	94,750	86,267
Žena 2	pravděpodobnost	0,036	0,019	61,463	99,762
	věrohodnost	97,413	97,740	0,758	84,829
Žena 2	pravděpodobnost	0,005	0,029	4,589	99,940
	věrohodnost	97,973	97,538	45,761	93,120
Usr 1	pravděpodobnost	0,014	0,035	0,094	2,404
	věrohodnost	97,825	97,427	96,267	63,575
Usr 2	pravděpodobnost	0,029	0,007	0,308	2,056
	věrohodnost	97,548	97,948	92,220	67,326
Usr 3	pravděpodobnost	0,023	0,011	7,822	1,696
	věrohodnost	97,663	97,878	30,171	71,555
Usr 4	pravděpodobnost	0,004	0,008	96,579	85,648
	věrohodnost	97,993	97,932	38,510	10,908
Usr 5	pravděpodobnost	0,020	0,008	3,081	0,308
	věrohodnost	97,723	97,934	57,111	92,231
Usr 6	pravděpodobnost	0,026	0,006	0,072	0,021
	věrohodnost	97,611	97,960	96,696	97,699
Usr 7	pravděpodobnost	2,879	0,059	0,004	0,047
	věrohodnost	58,937	96,947	97,988	97,192
Usr 8	pravděpodobnost	11,261	22,819	0,011	0,002
	věrohodnost	20,454	6,334	97,876	98,022
Usr 9	pravděpodobnost	0,787	0,223	0,028	0,007
	věrohodnost	84,168	93,788	97,573	97,942
Usr 10	pravděpodobnost	0,006	0,016	31,495	3,773
	věrohodnost	97,962	97,789	2,406	51,453

Příloha 2 Výsledky testování autentifikace uživatele

		Brána			
[%]	Uživatel	Muž 1	Muž 2	Žena 1	Žena 2
Muž 1	pravděpodobnost	99,978	62,289	0,029	0,008
	věrohodnost	95,683	0,875	97,548	97,935
Muž 1	pravděpodobnost	99,998	89,344	0,015	0,008
	věrohodnost	97,685	15,570	97,815	97,935
Muž 2	pravděpodobnost	50,743	99,964	0,008	0,005
	věrohodnost	0,003	94,652	97,931	97,980
Muž 2	pravděpodobnost	0,317	36,511	0,017	0,002
	věrohodnost	92,070	1,181	97,779	98,015
Žena 1	pravděpodobnost	0,008	0,005	99,960	4,664
	věrohodnost	97,923	97,981	94,436	45,290
Žena 1	pravděpodobnost	0,013	0,007	99,965	0,656
	věrohodnost	97,850	97,942	94,750	86,267
Žena 2	pravděpodobnost	0,036	0,019	61,463	99,762
	věrohodnost	97,413	97,740	0,758	84,829
Žena 2	pravděpodobnost	0,005	0,029	4,589	99,940
	věrohodnost	97,973	97,538	45,761	93,120
Usr 1	pravděpodobnost	0,014	0,035	0,094	2,404
	věrohodnost	97,825	97,427	96,267	63,575
Usr 2	pravděpodobnost	0,029	0,007	0,308	2,056
	věrohodnost	97,548	97,948	92,220	67,326
Usr 3	pravděpodobnost	0,023	0,011	7,822	1,696
	věrohodnost	97,663	97,878	30,171	71,555
Usr 4	pravděpodobnost	0,004	0,008	96,579	85,648
	věrohodnost	97,993	97,932	38,510	10,908
Usr 5	pravděpodobnost	0,020	0,008	3,081	0,308
	věrohodnost	97,723	97,934	57,111	92,231
Usr 6	pravděpodobnost	0,026	0,006	0,072	0,021
	věrohodnost	97,611	97,960	96,696	97,699
Usr 7	pravděpodobnost	2,879	0,059	0,004	0,047
	věrohodnost	58,937	96,947	97,988	97,192
Usr 8	pravděpodobnost	11,261	22,819	0,011	0,002
	věrohodnost	20,454	6,334	97,876	98,022
Usr 9	pravděpodobnost	0,787	0,223	0,028	0,007
	věrohodnost	84,168	93,788	97,573	97,942
Usr 10	pravděpodobnost	0,006	0,016	31,495	3,773
	věrohodnost	97,962	97,789	2,406	51,453

Příloha 2 Výsledky testování autentifikace uživatele

Ano					
[%]	Uživatel	Muž 1	Muž 2	Žena 1	Žena 2
Muž 1	pravděpodobnost	94,640	0,025	0,002	0,028
	věrohodnost	28,681	97,623	98,017	97,572
Muž 1	pravděpodobnost	99,340	0,055	0,007	0,010
	věrohodnost	72,127	97,034	97,942	97,901
Muž 2	pravděpodobnost	0,030	71,407	0,003	0,001
	věrohodnost	97,524	2,861	98,005	98,026
Muž 2	pravděpodobnost	0,080	65,940	0,005	0,011
	věrohodnost	96,533	1,503	97,972	97,876
Žena 1	pravděpodobnost	0,001	0,004	96,026	0,009
	věrohodnost	98,034	97,987	35,162	97,915
Žena 1	pravděpodobnost	0,001	0,001	99,949	0,156
	věrohodnost	98,035	98,033	93,695	95,061
Žena 2	pravděpodobnost	0,006	0,002	0,112	99,935
	věrohodnost	97,968	98,013	95,910	92,817
Žena 2	pravděpodobnost	0,005	0,002	0,011	99,960
	věrohodnost	97,972	98,012	97,887	94,390
Usr 1	pravděpodobnost	0,017	0,001	0,013	0,109
	věrohodnost	97,772	98,029	97,837	95,956
Usr 2	pravděpodobnost	0,004	0,003	0,063	6,291
	věrohodnost	97,991	98,006	96,874	36,455
Usr 3	pravděpodobnost	0,019	0,721	0,008	0,118
	věrohodnost	97,730	85,214	97,928	95,782
Usr 4	pravděpodobnost	0,019	0,019	7,850	1,255
	věrohodnost	97,741	97,737	30,700	77,296
Usr 5	pravděpodobnost	0,022	0,004	0,612	1,027
	věrohodnost	97,680	97,988	86,986	80,529
Usr 6	pravděpodobnost	0,018	0,007	0,022	0,045
	věrohodnost	97,760	97,949	97,675	97,228
Usr 7	pravděpodobnost	5,300	0,020	0,017	0,007
	věrohodnost	43,056	97,720	97,780	97,949
Usr 8	pravděpodobnost	0,094	0,012	0,004	0,001
	věrohodnost	96,260	97,865	97,995	98,034
Usr 9	pravděpodobnost	0,009	0,017	0,001	0,004
	věrohodnost	97,916	97,764	98,029	97,994
Usr 10	pravděpodobnost	0,009	0,008	0,020	8,800
	věrohodnost	97,908	97,921	97,710	26,895