UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Prepoznava govora v primeru pametne pisarne

(Speech recognition in scope of smart offices)

Ime in priimek: Klemen Kolenc
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Jernej Vičič
Somentor: doc. dr. Branko Kavšek

Koper, september 2018

Ključna dokumentacijska informacija

Ime in PRIIMEK: Klemen KOLENC

Naslov zaključne naloge: Prepoznava govora v primeru pametne pisarne

Kraj: Koper

Leto: 2018

Število listov: 44
Število slik: 3
Število tabel: 3
Število referenc: 21
Število prilog: 2
Število strani prilog: 2

Mentor: doc. dr. Jernej Vičič

Somentor: doc. dr. Branko Kavšek

Ključne besede: Prepoznava govora, CMUSphinx, Jezikovni modeli, Avtomatizacija

Izvleček: V zaključni nalogi je opisan glasovni nadzor, oziroma opremljanje hiše/pisarne z sistemom, ki uporabniku omogoča lažje upravljanje oziroma skrajša zamudna opravila. Sistem ponuja tudi osnovno varnost, oziroma nadzor prostora kjer je nameščen. Glede na to da je to področje relativno novo in na njem prevladujejo predvsem namenski izdelki večjih korporacij (npr. Alexa, amazon Echo), je raziskana alternativa domače izdelave lahko implementirana za veliko manjšo ceno. Za problem je izbrana čimvečja funkcionalnost naprave, s čim manjšim stroškom za končnega uporabnika. Opisan je potek reševanja problema in težave pri implementaciji. V zadnjem delu je analizirana funkcionalnost sistema in cenovna primerjava z ostalimi napravami na trgu, V zaključku pa je celotna naloga povzeta in podane možnosti za nadaljevanje dela.

Key words documentation

Name and SURNAME: Klemen KOLENC

Title of the final project paper: Speech recognition in scope of smart offices

Place: Koper

Year: 2018

Number of pages: 44
Number of figures: 3
Number of tables: 3
Number of references: 21
Number of appendices: 2
Number of appendix pages: 2

Mentor: Assist. Prof. Jernej Vičič, PhD

Co-mentor: Assist. Prof. Branko Kavšek, PhD

Keywords: Speech recognition, language models, CMUSphinx, Automatization

Abstract: The final project paper describes voice recognition in the scope of equipping a room or an office with a system that enables the end user to shorten his tasks. The system should also provide basic security or monitoring of the room it's installed in. Since this field is relatively new, it is dominated by devices from bigger corporations (Alexa, amazon Echo), the research of this paper focuses on implementing a home-created alternative for a lower price. The chosen goal is maximised functionality for a price as low as possible. The process to solving the problem is described along with issues that were encountered during implementation. In the last segment an analysis of the functionality and a comparison to other devices on the market is made. The conclusion contains the summary of the entire paper, along with the possibilities to continue development.

Zahvala

Zahvaljujem se mojemu mentorju doc. dr. Jerneju Vičiču in somentorju doc. dr. Branku Kavšku za strokovno sodelovanje in vodenje pri izdelavi zaključne naloge. Zahvaljujem se tudi drugim delavcem Fakultete za Matematiko, Naravoslovje in Informacijske Tehnologije za sodelovanje in strokovno pomoč ter podporo. Zahvalil bi se tudi moji družini za podporo in spodbujanje tekom celotnega študija.

Kazalo vsebine

Kazalo tabel

Kazalo slik

Kazalo prilog

Seznam kratic

API	Application programming interface (Vmesnik za programsko interakcijo)
LAN	Local area network (Osebno ali domače omrežje)
WAN	Wide area network (Javno komunikacijsko omrežje)
VPN	Virtual private network (navidezno zasaebno omrežje)
TTS	Text to speech (pretvorba besedila v govor)
FTP	File transfer protocol (protokol za prenos datotek)
SSH	Secure shell
SD	Secure Digital
USB	Universal Serial Bus (vsestransko zaporedno vodilo)
GPIO	General Purpose Input/Output (splošno namenski vhod/izhod)
HTML	Hyper Text Markup Language (jezik za označevanje nadbesedila)
MFCC	Mel Frequency Cepstral Coefficient
LM	Language Model (jezikovni model)
HDMI	High-Definition Multimedia Interface(visoko-ločljivostni multimedijski priključek)
HMM	Hidden Markov Model (skriti Markov model)
ARM	Advanced RISC Machine
AUX	AUXiliary
PHP	PHP Hypertext Preprocessor
SQL	Structured Query Language (strukturiran povpraševalni jezik za delo s podatkovnimi bazami)
GIT	Global Information Tracker (svetovni sledilnik informacij)
CPU	Central Processing Unit (centralna procesna enota)
RAM	Random Access Memory (spomin z direktnim dostopom, delovni pomnilnik)
LED	Light-emitting diode
LCD	Liquid-crystal display (prikazovalnik z tekočimi kristali)

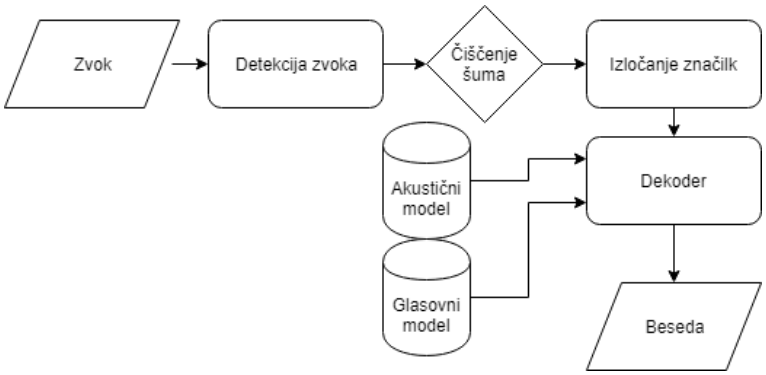
1 Uvod

Svet tehnologije vedno bolj stremi k temu, da uporabniku izboljša izkušnjo uporabe, oziroma mu delo čim bolj olajša in ne zahteva kompleksnih zunanjih vmesnikov za interakcijo. V luči prav tega se je na trgu pojavil glasovni nadzor, ki uporabniku omogoča, da z napravami upravlja brez klikanja gumbov ali uporabe drugih vnosnih naprav. Sprva so bile to le naprave, ki so bile sposobne prepoznati preproste ukaze, kot na primer "CALL HOME" ukaz, ki ga je bil sposoben prepoznati vsak povprečen telefon v 2000tih. V zadnjih letih se je trg skokovito razvil z glasovno nadzorovanimi napravami, ki funkcionirajo kot hišni pomočniki. Z napravo se je možno pogovarjati, jo spraševati preprosta vprašanja (odgovor sistem poišče preko spletnega iskalnika), ali pa mu uporabnik naroči, da naj nekomu pošlje sporočilo. Da pa naprava to zna storiti, mora vsebovati posebne algoritme in programsko opremo za prepoznavo govora. Pomembno je tudi, da taka naprava funkcionira odzivno in v realnem času, saj to drastično izboljša uporabnikovo izkušnjo kot pa, če bi mogel uporabnik čakati več deset sekund ali celo minut na odziv. Prav tako se v dotične naprave vgrajuje kvalitetne prostorske mikrofone z izničevanjem šuma, da uporabniku ni treba naprave nositi s sabo ali na sebi ampak ga le-ta "sliši" od kjerkoli v prostoru.
V nadaljevanju naloge je v večih segmentih razdelana splošna struktura programov in algoritmov za prepoznavo glasu kot so podatkovne zbirke za tvorjenje govornega korpusa, jezikovni model, slovar, ter dekoder. Predstavljeni so tudi različni principi delovanja sistemov, skupaj z načini prilagoditve sistema za željeni primer uporabe. V drugem segmentu so analizirane različne možne knjižnice z različnimi pristopi za prepoznavo govora in predstavljena implementacija prepoznave govore z izbrano knjižnico iz spleta na lastnemu sistemu. Na konkretnemu primeru je analizirana uporabnost sistema razvitega v okviru te naloge in izdelana cenovna primerjava ter primerjava po komponentah z že obstoječimi zaključenimi sistemi ki jih lahko uporabnik kupi v trgovini. Cilj naloge je za konkurenčno oziroma čim nižjo ceno implementirati sistem z podobno ali večjo funkcionalnostjo in raziskati možne razširitve sistema ter integracijo zvočnega nadzora v ostale sisteme, ki so prisotni v uporabnikovem vsakdanu.

1.1 Zgodovina

Prepoznava govora je bila v fazi razvoja že od leta 1950. Prvi sistem sposoben osnovne prepoznave številk govorjenih z enim glasom, je bil ustvarjen v Bell Laboratories ("Audrey" System) leta 1952. V letu 1970 pa so se raziskave na področju prepoznave glasu začele pospešeno izvajati, tudi ob pomoči programa ministrstva za obrambo (DoD) DARPA Speech Understanding Research. Kot plod teh raziskav štejemo sistem "Harpy", ki je bil sposoben prepoznati približno tisoč besed, kar je za primerjavo s prejšnjimi sistemi ogromno. Posebnost sistema Harpy je bilo drevo končnih stanj, ki je vsebovalo možne stavkovne strukture. Na tej točki v razvoju je razvidna povezava razvoja iskalnih dreves in ostalih algoritmov iskanja, z vzporednim razvojem prepoznave govora. To je bilo bolj izrazito dokazano v letih 1980, kjer so se pojavili sistemi sposobni prepoznave več tisoč besed, zavoljo uporabe statistične metode poznane kot skriti Markov model [21]. Tak model je računal možnosti, da je glas, ki ni bil enolično prepoznan v sistemu, beseda. V prihodnjem desetletju so se z razvojem močnejših osebnih računalnikov sistemi prepoznave govora odprli tudi na javni trg. V letih 2000 je prepoznava dosegla mejo stagnacije pri približno 80% natančnosti. Novost so čez nekaj let prinesli googlovi sistemi, ki niso le uporabljali govorčevega vnosa kot element prepoznave, temveč so se glasovni zapisi uporabnikov hranili na strežnikih in so funkcionirali kot podlaga za učenje in izboljšavo modela. S tem načinom dela se sistem postopoma sam izboljšuje in se uči hitreje in bolj učinkovito z naraščanjem števila uporabnikov, ki ga uporabljajo. Za primerjavo - googlov trenutni sistem je sposoben prepoznati več kot 200 miljard besed, naučenih iz iskanj uporabnikov. [13]

2 Principi delovanja posameznih segmentov sistema



Slika 1: Princip delovanja celotne prepoznave človeškega glasu oziroma konkretnega ukaza je lahko prikazan s posplošenim diagramom, ki bo v nadaljevanju razdelan.

2.1 Zajemanje govora (end-point detection)

V tem segmentu je predstavljen strojni vidik prepoznave govora. Program mora čim bolj natančno prepoznati začetek in konec besede, kar lahko izvede s preverjanjem in izločanjem moči, frekvence, ter dob za prehajanje čez ničelno mejo (zero crossing rate). [10] V tem segmentu je potrebno posebno pozornost usmeriti tudi na besede ki se spreminjajo v moči (na koncu stavka, ko moč glasu govorca govora ni enaka kot moč ob začetku izgovorjave) in na besede ki se končajo s šumom oziroma nosnikom. V zadnjem primeru (nosni glas) lahko predstavlja težavo iskanje natančnega konca izgovorjene besede, kar v nadaljevanju programu oteži razvozlavanje besede, in kot posledica razumevanje pomena. Za olajšanje dela lahko uporabimo tri osnovne operacije nad glasovnimi posnetki- rezanje zgornjih frekvenc, rezanje spodnjih frekvenc, in izolacija posameznih glasov glede na frekvenco.

2.2 Izločanje značilk (feature extraction)

Izločanje značilk je naslednji korak pri prepoznavanju govora. V tem koraku se z uporabo različnih algoritmov za analizo frekvenc in anomalij v digitalnem zapisu govora razloči posamezne značilnosti izgovorjene bedsede, oziroma glasove. Eden od algoritmov za tako analizo je Mel-Scale Filter Bank spektrogram, ki vsakemu glasu v besedi določi moč in barvo glasu, ter tako določi izgovorjen glas kar se da natančno. Predstavljen je v nadaljevanju.

2.2.1 MFCC izločanje posameznih značilk (Mel Frequency Cepstral Coefficient)

MFCC je algoritem ki sta ga v letu 1980 predstavila Davis in Mermelste. Deluje v nekaj posplošenih korakih:

- Razdeljevanje signala v manjše okvirje
- Kalkuliranje periodogramskega približka za vsak okvir
- Apliciranje filtrov na spektrum moči, seštevanje moči vsakega filtra
- Logaritmiranje energij filtrov
- Izvedba diskretne kosinusne transformacije nad logaritmom energij filtrov
- Ohranjanje sredinskih koeficientov, zavračanje ostalih

[17]

2.3 Govorni korpus

Korpus je podatkovna baza branih besedil in izgovorjenih besed, ki lahko služi kot osnova za kreacijo akustičnih modelov. Pri kreaciji se ponavadi poslužuje več korpusov, da je naučen akustični model čim bolj natančen. Ponavadi so korpusi deljeni v dve skupine: brani govor in prost dialog med dvema osebama. Prvi vsebuje sezname knjižno pravilnih besed, ki jih beremo napravi, sekvence števil, brane zgodbe oziroma zvočne knjige in odlomke iz člankov. Ti primeri vsebujejo predvsem knjižno slovnico brano s strani govorca z lepo izgovorjavo in pravilno naglašenimi besedami.
Druga skupina pa je za učenje statistično težja, vendar mnogo bolj donosna, saj vsebuje tudi medmete, šume, napačno naglašene in narečne besede ter različno hitrost oziroma barvo glasu, kar je predvsem razvidno v posnetkih dialogov z večimi osebami. S tem podamo modelu več informacij o možnih strukturah govora. Pozitivna stran takšnih

glasovnih datotek je, da se iz takega korpusa naučen model lažje in hitreje odziva v primeru vsakodnevnega govora, saj vsebuje tudi narečne besede in drugačno naglaševanje. Slednje se lahko pojavi tudi v komuniciranju z napravo oziroma v konkretnem primeru sistema, ki bo implementiran v sklopu te naloge. Tak model tudi lažje izloči dodatne zvoke in brezpredmetne dele komunikacije - medmete, šume in izolirane glasove, ki bi v nasprotnem primeru lahko zmotile model naučen iz striktno knjižno branih besedil.

2.4 Jezikovni korpus

Jezikovni korpus je velik strukturiran izbor besedil, člankov, leposlovij in zapisanih govorov jezikovnega območja, ki so bila elektronsko procesirana in klasificirana. Jezikovni korpus je uporabljen za izvajanje statističnih in lingvističnih analiz jezika ter pravil, ki sodijo v izbrano jezikovno področje. Glede na število besed v korpusu se poveča tudi število podatkov in značilnosti jezika, saj več besed pomeni večjo podlago za študijo pojavitev, ter potrjevanje hipotez za izbrani jezik. Takšni korpusi lahko obsegajo od nekaj tisoč do več miljard besed, na primer Slovenski korpus Gigafida [1], ki vsebuje skoraj 1,2 miljarde besed v izbranih zbirkah avtentičnih Slovenskih besedil in omogoča napredno iskanje jezikovnih podatkov Slovenskega jezika. [1]

2.5 Slovar

Slovar je programska datoteka, ki vsebuje statistiko in informacije o glasovih, ki tvorijo besedo. Vsaki možni besedi iz uporabljene slovnice je pripisana fonemska sestava (iz končnega števila zvočnikov-fonemov). Za konkreten primer lahko uporabimo besedo "hiša", ki jo zapišemo kot "HH I SH A". Vsaka beseda vsebuje tudi statistično analizo uporabe oziroma pogostosti pojavljanja v vsakodnevnem govoru ter posledično v danem ukazu, ki smo ga posredovali. Tej strukturi se reče skriti Markov Model (HMM- hidden Markov Model), ki je sestavljen za vsak fonem posebej.

2.6 Akustični model

Akustični model je narejen iz prej predstavljenih zvočnih korpusov, katerim je dodana še relacija med posameznim fonemom v slovarju in frekvenco glasu, oziroma sama povezava med besedo in glasovi na katerih se model uči. Kot pomoč je akustičnemu modelu posredovana tudi slovnica, v kateri so začrtane vse sprejemljive besede za prepoznavo. Vsebuje samostojne besede, besedne zveze, fraze in celotne stavke. Slovnico uporabimo

[1] Gigafida `http://www.gigafida.net/`

kadar želimo omejiti svoj ukaz, oziroma naučiti model samo za sprejemanje danih besednih zvez. Za konkreten primer se lahko vzame avtomatsko klicanje z glasovnim nadzorom: "KLIČI DOMOV", kar pa je lahko parafrazirano tudi v obliki 'POKLIČI DOMA' in vseh možnih kombinacijah besed, ki so lahko izgovorjene z namenom posredovanja istega ukaza napravi. Če je uporabljena katerakoli druga beseda oziroma kombinacija, ki ni začrtana v akustičnemu modelu, npr. "DOMA POKLIČI"bo naučen jezikovni model javil napako, da prepoznana besedna zveza ni bila najdena v trenutni slovnici. Z uporabo slovnice lahko zožimo svoj izbor ukazov za interpretacijo, saj je en ukaz lahko izveden z različnimi možnimni kombinacijami členov. Za primer telefonskega klica lahko uporabnik isti ukaz izvede z besedo KLIČI, POKLIČI ali ZAČNI. Besedo ki je izgovorjena za tem, oziroma naslednja beseda, ki jo sistem prepozna je interpretirana kot opcijska spremenljivka oziroma objekt nad katerim je določen ukaz izveden. Za primer opisane datoteke je na voljo priloga, v kateri je razvidna struktura slovarja in slovnice uporabljene v konkretni implementaciji.

2.7 Jezikovni model

Jezikovni model[2] je datoteka, ki pomaga enkoderju pri prepoznavi zvoka. Vsebuje seznam besed in za vsako besedo izračunano verjetnost, da je bila izgovorjena oziroma, da se je pojavila v besedilu. Jezikovni model tako omeji nabor besed, ki jih mora dekoder pretehtati na posamezni točki med prepoznavo govora. To pripomore k boljši natančnosti samega naučenega modela in tudi k hitrejšemu delovanju prepoznave govora, kar je predvsem občutno, če je prepoznava izvajana v realnem času- npr. Google assistant. Na opisan način naučeni jezikovni modeli lahko funkcioniorajo na dva načina:

2.7.1 Prvi način - absolutni

je preprostejši absolutni način, v katerem dekoder izbere nekaj besed, ki imajo v tistem trenutku največjo možnost, da so bile izgovorjene. To ne vzame veliko časa, saj algoritem samo preveri vrednosti, jih sortira po možnosti pojavitve in vzame x najvišje uvrščenih. Na dolgi rok to pomeni besedišče oz. slovnico združeno v strukturo, ki je lahko interpretirana kot graf možnosti pojavitve, ki ga nato model uporablja.

2.7.2 Drugi način - verjetnostni

pa je bolj zahteven in časovno potraten, vendar lahko na dolgi rok natančnejši- probabilističen oziroma verjetnostni način. Ta način vključi v algoritem še verjetnost besed

[2]Primer jezikovnega modela za slovenski jezik `http://centerslo.si/wp-content/uploads/2015/10/21-jakopin.pdf`

naslednic, ki lahko sledijo izbrani besedi ter izračuna skupno verjetnost na podlagi katere se nato odloča. Na podlagi tega so poznani tako imenovani n-gramski modeli, kjer n - 1 pomeni število predhodnih besed. Tak model izračuna možnost pojavitve besede glede na njene predhodnike (starši, pra-starši in prejšnji) in njihovih lastnih možnosti, da se pojavijo. [12] V primeru, da je takemu modelu dodana še fleksibilnost oziroma zmožnost uporabe tudi samo besede z enim staršem (bigram) ali samo ene besede (unigram), je dobljen model, ki je sposoben prepoznati vsako besedno zvezo, ne glede na dolžino. V takem primeru je potrebno modelu seveda posredovati dobro definiran akustični model in slovnico. Statistični jezikovni model je lahko najbolje uporabljen pri prepoznavi prosto govorjenih besedil ali pogovorih, v katerih je nemogoče vnaprej strukturirati vse besedne zveze, ki bojo upoštevane, ker bi bilo le-teh preveč za absolutno učenje.

Trigramski model

Pri prepoznavi govora je največkrat uporabljen trigramski model (n = 3), kjer za oceno možnosti pojavitve besede uporabimo kontekstno zgodovino prejšnjih treh besed. Najpogosteje je izbran, ker predstavlja dobro ravnovesje med kompleksnostjo algoritma in robustnostjo oziroma grobim predvidevanjem besed. Algoritem trigramskega modela za računanje verjetnosti pojavitve besede izračuna še vrednost starša in pra-starša oz. predhodnih dveh besed (n - 1 = 2), in računa verjetnost pojavitve vseh treh kot celote. Konkreten primer delovanja:

$$P(jaz, sem, bil, danes, doma) = P(jaz| < s >, < s >) * P(sem| < s >, jaz) * P(bil|jaz, sem) * P(danes|sem, bil) * P(doma|bil, danes) * P(< /s > |danes, doma) \tag{2.1}$$

V tem primeru "s" in "/s" označujeta začetek in konec stavka, ker moramo v izračun vzeti tudi robne pogoje zaradi pravilnega izračuna verjetnosti. [8]

2.8 Dekoder

Dekoder je temeljni del programa v katerem se vrši prepoznava govora. Združuje akustični in jezikovni model ter slovnico, na podlagi česa se izvršuje prepoznava besed. Primer delovanja delovanja: dekoder "posluša" posamezne spremembe v frekvencah oziroma glasovih, ki jih je uporabnik izgovoril in so bili obdelani z izločevanjem značilk. Za tem z uporabo nabora najdenih zvokov po statistiki iz akustičnega modela dekoder išče vse možne kombinacije možnih fonemov, ter izbere fonem, ki ima največjo možnost, da je bil izgovorjen. [2] Ko najde kombinacijo do takrat izgovorjenih fonemov si jo

označi (na primer za vsako celo izgovorjeno besedo) ter posluša naprej. V kar se da realnem času prepoznave si dekoder tvori drevo možnosti za izgovorjene besede, dokler uporabnik ne preneha z govorom ali dekoder ne pride do premora med besedami. Nato dekoder primerja označene (slišane) besede s slovnico, ki mu je bila posredovana in najde najbolj primerno besedo oziroma besedno zvezo z največ ujemanji. Če taka beseda obstaja, jo nato vrne kot prepoznano.

Z dobro naučenim jezikovnim modelom je mogoče ujemanje glasov dobro umeriti med ugibanje sistema (kadar fonemi niso razločni bo sistem vzel najbolj podobne, ali pa celo vse možnosti v primeru, da en fonem iz besede sploh ni prisoten zaradi napake v komunikaciji) in kompleksnostjo (veliko možnosti za izgovorjene besede). V tem primeru govorimo o robustnem jezikovnem modelu. [12] Tak primer delovanja v realnem času je lepo viden v googlovem asistentu, ki je prisoten v večini modernejših pametnih telefonov, vendar za popolno delovanje potrebuje dostop do spleta.

3 Analiza obstoječe programske opreme

Definicija Jedro sistema je predvsem vmesnik za prepoznavo in interpretacijo govora. Na spletu obstaja že več knjižnic in okolij (API), ki uporabniku omogočijo integracijo v lasten zgrajen sistem in kreacijo lastnih skript za izvajanje na ukaz. V nadaljevanju je pripravljena kratka predstavitev funkcionalnosti vsakega programskega orodja in analiza za konkretno uporabo na našem izbranem primeru ter razlogi za uporabo najprimernejšega.

Kriteriji V vsaki testirani knjižnici je predvsem iskana dobra integracija z konkretnim že izdelanim vezjem Raspberry Pi- to prihrani čas implementacije in morebitno kompliciranо konfiguracijo. Program je seveda moral biti odprtokoden, kajti cilj te naloge je med drugim tudi čim nižja cena, kar pa bi plačljiva programska oprema lahko znatno otežila.

Delitev Knjijžnice in orodja so deljene na več skupin glede na način delovanja. V glavnini se delijo na plačljive in odprtokodne programe, v podrobnem pa so razdeljene glede na podatke ki jih potrebujejo. Za primer, CMUsphinx je povsem samostojno orodje, ki je konfigurabilno za delovanje brez povezave na mrežo, v nasprotnem primeru pa je Google Speech API, ki za svoje delovanje potrebuje konstantno povezavo na googlov strežnik na medmrežju.

3.1 Google Speech API

Google Speech API [1] temelji na pošiljanju posnetkov na googlove strežnike na katerih tečejo modeli nevronskih mrež za prepoznavo glasu. Je orodje z najširšo prepoznavo jezikov (110 različnih variant). Glavna prednost tega je, da posamezen model dobiva velike količine vhodnih podatkov, kar mu omogoča hitro izboljševanje in veliko natančnost glede na čas uporabe. Centralizirana strežniška osnova pomeni tudi hitro delovanje in zmožnost prepoznave v realnem času zaradi rabe močnih strežniških se-

[1] GOOGLE SPEECH API `https://cloud.google.com/speech-to-text/`

stavov in strojne opreme, ki se z strojno opremo uporabljeno v sklopu naloge niti ne more primerjati. Prednost strežniške osnove je tudi delovanje na večih napravah, glede na to, da uporabnikova naprava sama postane sredstvo za posredovanje vhodnih audio podatkov in vračanje interpretacij iz strežnika. [9]

3.2 Microsoft Bing Speech API

Podobno kot Google Speech API tudi Bing Speech API[2] temelji na strežniški osnovi z zelo podobno funkcionalnostjo in uporabo. Glede na podobnost med funkcionalnostmi googlove in microsoftove knjižnice je bila primernejša googlova, zaradi širšega nabora podprtih jezikov. Microsoftovo orodje namreč temelji bolj na uporabi okolij c in .net, ki pa za vezje uporabljeno v konkretnem primeru ne pride v poštev. [14]

3.3 Dialogflow

Dialogflow[3] je orodje pripravljeno za zelo enostavno implementacijo pogovora med uporabnikom in napravo. Vsebuje preprost vmesnik, ki omogoča uporabniku enostavno nastavljanje jezikovnega modela in ukaznih fraz preko grafičnega vmesnika. Podpira zelo širok nabor jezikov, od jezika java do jezika Python ter vsebuje dobro dokumentacijo, kar pomeni enostavnejše delo in implementacijo na konkretnem uporabljenem vezju, predstavljenem v nadaljevanju. [7]

3.4 Speechmatics

Speechmatics[4] je plačljiv program, ki ponuja več različnih spletnih servisov, ki temeljijo na prepoznavi glasu. Ponuja servise s prepoznavo glasu v samostojnem okolju, realnem času, oblaku ter prepoznavo vnaprej posnetega govora. [4]

3.5 Vocapia Speech To Text API

Vocapia[5] je plačljivo orodje namenjeno predvsem za sisteme Linux. Temelji na pošiljanju zvokovnih datotek na njihov namenski strežnik, na katerem se prepoznava vrši v realnem času. [20]

[2]BING SPEECH API `https://azure.microsoft.com/en-us/services/cognitive-services/speech/`

[3]DIALOGFLOW `https://dialogflow.com/`

[4]SPEECHMATICS `https://www.speechmatics.com/`

[5]VOCAPIA `https://www.vocapia.com/`

3.6 CMU Sphinx

CMU Sphinx[6] je odprtokoden nabor orodja za prepoznavo govora, preprost in lahek sistem, ki ga zaradi majhnosti lahko uporabimo tudi na prenosnih napravah oz. mobilnih telefonih. Deluje brez povezave v splet, kar pomeni da se sama prepoznava govora vrši lokalno, na napravi. Model je potrebno naučiti iz narejenih struktur opisanih v prvem delu naloge, ali pa ga uvoziti prednaučenega. [6]

3.7 UWP speech recognition

UWP Speech Recognition [7] je Microsoftovo odprtokodno orodje za prepoznavo glasu. Deluje z vnaprej definirano slovnico; API je mišljen za integracijo z Visual Basic in .NET okoljem, za dopolnjevanje funkcije besedila v govor (TTS- Text To Speech) [19]

3.8 Kaldi

Kaldi [8] je odprtokodno orodje za prepoznavo glasu, zasnovano v jeziku c++. Omogoča napredno učenje modelov in konfiguracijo za vsako potrebo, vendar zaradi velike zahtevnosti oz. prevelikega spektra uporabe ni primerno za konkreten projekt. [16]

3.9 Simon

Simon[9] je odprtokoden program za prepoznavo govora, narejen na osnovi CMU Sphinx in Julius, zasnovan za uporabo preko preprostega grafičnega vmesnika. Poganjati ga je mogoče na Windows ali Linux sistemih, obstajajo pa tudi navodila za integracijo na vezje raspberry pi. [18]

3.10 Julius

Julius[10] je API za prepoznavo govora zasnovano na jeziku c. Zmožen je prepoznave govora v realnem času ter uporabe velikih slovnic. Orodje je zelo močno, vendar manj primerno za konkreten primer glede na to, da ni iskana prepoznava velikega števila

[6] CMUSPHINX `https://cmusphinx.github.io/`
[7] UWP `https://docs.microsoft.com/en-us/windows/uwp/design/input/speech-recognition`
[8] KALDI `kaldi-asr.org/`
[9] SIMON `https://simon.kde.org/`
[10] JULIUS `https://github.com/julius-speech/julius`

ukazov v realnem času. Povečana zahtevnost lahko pomeni tudi težave pri poganjanju, glede na zmogljivost uporabljenega vezja. [11]

4 Implementacija- strojna oprema

Za uspešno implementacijo zadane naloge je potrebno izbrati tudi primerno strojno opremo na kateri bo tekla prej predstavljena programska oprema. Glavni del sistema predstavlja vezje Raspberry PI 1 model B+, ki je glede na strojne specifikacije primerno za poganjanje večine prej predstavljenih programov. Na osnovno vezje so preko vmesnika USB ali preko GPIO priklopljene še ostale komponente, ki služijo v večini kot vhodno/izhodne naprave in naprave za komunikacijo z ostalimi sistemi.

4.1 Začrtane specifikacije jedra sistema - Raspberry Pi 1 model B+

Na temu vezju je načrtovano izvajanje glavne sistemske skripte, procesa za posodabljanje stanja varnostnega sistema na spletni strani ter procesa za prepoznavo zvoka. Glede na potrebe vsakega od omenjenih procesov bo na omenjenem vezju moralo teči še nekaj drugih pomožnih procesov, na primer konkretno baza, ki bo hranila podatke o trenutnem stanju senzorjev in strežnik za dostop do spletne strani sistema. Cena vezja in primerjava je opisana v razdelku Rezultati, specifikacije izbranega vezja pa so naslednje:[1]

- 700 Megaherčen Cortex-A7 ARM procesor
- 512MB pomnilnika
- 16GB spominska kartica; služi kot glavni pomnilnik in zagonski disk sistema
- Ethernet priklop- služi za komunikacijo z ostalimi računalniki na mreži, ter prikazovanje podatkov na spletni strani
- 26 GPIO- splošni vhodno izhodni pini; uporabljeni bodo predvsem za povezavo senzorjev za varnostni sistem ki bo poleg prepoznave glasu in ukazov integriran na sistemu

[1]RASPBERRY PI 1 MODEL B+ `https://www.adafruit.com/product/1914`

- 3.5 audio izhod; uporabljen bo za povezavo z zunanjim zvočnikom, priklopljenim preko AUX kabla. S tem so sistemu omogočene povratne zvočne informacije uporabniku

- USB zvočna kartica; naknadno dodana zvočna kartica, ki bo služila za priklop mikrofona na vezje. Za branje podatkov iz vhoda in uporabo mikrofona prek ukazne vrstice je potrebno namestiti še gonilnik

4.2 Ostala strojna oprema

- 3,5mm zunanji mikrofon; priklopljen v sistem kot glavni vhod zvočnih ukazov. Uporabljen je nizkocenovni bližinski mikrofon, po potrebi bi lahko bil zamenjan s prostorskim mikrofonom z večjim dometom, vendar je za učenje modela bolj primeren čistejši zvok s čim manj anomalijami, ki se pri prostorskih mikrofonih lažje pojavijo. [5]

- Bluetooth kartica, USB priklop(opcijsko); v primeru potrebe po implementaciji komunikacije s telefonom oziroma drugo zunanjo napravo. Preko bluetootha bi se lahko izvajala skripta za sinhronizacijo opomnikov in telefonskih obvestil, koledarja ter budilk.

- Termo senzor DS18B20[2]; povezan na sistem preko GPIO pinov, služi za dodatno informacijo nad temperaturo v prostoru, ki bo prikazana na spletnem vmesniku.

- Raspberry Pi 2 NoIR kamera[3]; zajeta slika iz kamere se bo shranila vsakih 10 sekund in zadnjih nekaj slik bo prikazanih na spletnem vmesniku za direkten ogled iz drugih naprav. V primeru, da bo sprožen tudi senzor, se slika ne bo prepisala. NoIR infrardeče tipalo pomeni boljši zajem slik v slabših svetlobnih pogojih, ampak slabšo barvno paleto pri dobrih pogojih.

- Gumbi in stikala; uporabljeni kot imitacija senzorjev na vratih in oknih. Služijo za sprožanje ostalih funkcij sistema, vsak pritisk pa se zabeleži v podatkovno bazo in je prikazan na spletni strani v realnem času.

- Opcijsko: namenski strežnik; če bi se pokazala potreba po dodatni procesorski moči, bi se lahko del programov za izvajanje preselilo na namenski strežnik, čemur pa se bom poskusil izogniti zaradi občutnega dviga pri ceni projekta oziroma ceni za vzdrževanje sistema.

[2]DS18B20 `https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf`
[3]PI CAMERA `https://www.raspberrypi.org/documentation/hardware/camera/`

4.3 Dodatna uporabljena programska oprema

Sledijo opisi programske opreme, ki jo je potrebno namestiti za pravilno delovanje samega sistema in omogočanje funkcij, ki si jih želimo implementirati. Zaradi tega je ločena od analize orodij za prepoznavo govora, ker je bližje strojni opremi in funkcionira kot osnova projekta.

- LAMP[4] strežnik- linux (v konkretnem primeru Raspbian, ki je distribucija Linux operacijskega sistema), Apache, mySQL, PHP. Služi za osnovno postavitev spletnega strežnika, ki bo dostopen iz ostalih naprav na isti mreži. Prav tako lahko je v ta segment dodana podpora za javascript ter HTML5, na čemer lahko zgradimo lep in funkcionalen vmesnik.
- Python[5]- uporabljen za izvajanje glavnega cikla programa in večine stranskih skript. Za programski jezik python sem se odločil, ker je glede na strukturo in preprostost najbolj primeren jezik za hitro implementacijo in spreminjanje funkcionalnosti ter je podprt na konkretnih vezjih uporabljenih v okviru naloge.
- PuTTY[6] in WinSCP[7]- programska oprema predvsem potrebna za namestitev in nastavljanje ter prenos potrebnih datotek na uporabljeno vezje.
- Arduino Studio[8] - uporabljen predvsem zaradi funkcije Serial Monitor; potreben za komunikacijo z morebitnim vezjem nižjega strojnega nivoja. Uporabljen je bil za preizkusno programiranje brezžičnega sprejemnika, ki bi bil lahko v končni implementaciji nameščen na releju v hišnem omrežju in bi deloval kot brezžično stikalo, nadzorovano z skripto v jedru sistema. Ta segment ni bil dokončan, saj sega izven obsega naloge, tako da je bil izveden samo preizkus funkcionalnosti.

4.4 Programske specifikacije

Programske specifikacije za končan sistem so bile začrtane glede na pomembnost njihovih prisotnosti v projektu, da bi bil ta lahko označen kot uspešen. Najbolj pomembne funkcije pomenijo samo jedro sistema, zraven le-tega pa so bile dodane ostale funkcije kot opcijske dodatne funkcionalnosti, ki niso kritične za delovanje sistema, vendar je

[4]LAMP `https://www.liquidweb.com/kb/what-is-a-lamp-stack/`
[5]PYTHON `https://www.python.org/`
[6]PUTTY `https://www.putty.org/`
[7]WINSCP `https://winscp.net/eng/index.php`
[8]ARDUINO STUDIO `https://www.arduino.cc/en/Main/Software`

zaželjena njihova implementacija v čim večjem številu in predstavljajo dodano vrednost k uspešnosti projekta. V seznamu, ki sledi, so razvrščene od najbolj do najmanj pomembne.

- Jedro sistema
  - Delujoč jezikovni model; želja je, da bi model naučil sam, vendar če bo to pretežavno bo uporabljen model dobljen iz spletnega vira.
  - Zajem glasu in odzivanje na preproste ukaze
  - Delujoč video varnostni sistem s časovnim zajemom slik in preprostim spletnim vmesnikom
- Opcijske funkcije sistema
  - Odzivanje na "wake word"- beseda za bujenje sistema
  - Možnost predvajanja audio vsebin na ukaz
  - Prikaz vnaprej določene vsebine na spletno stran
  - Beleženje senzorjev za aktivnost po prostoru, prikaz informacij na spletnem vmesniku
  - Časovna zvočna opozorila glede na prednastavljene opomnike na spletnem vmesniku
  - Lepši HTML5 Vmesnik z možnostjo prijave uporabnika
  - Odgovarjanje uporabniku preko TTS (text to speech- besedilo v govor) http://espeak.sourceforge.net/
  - Funkcionalen temperaturni senzor
  - Povezava s telefonom za sinhronizacijo opomnikov

4.5 Potek gradnje sistema

Po tem, ko je bila na podlagi ocen izbrana strojna oprema in orodje za prepoznavo zvoka, je sledil pričetek gradnje sistema. Na formatirano SD kartico je bila naložena zadnja verzija operacijskega sistema Raspbian, ki bo služil kot okolje v katerem se bodo izvajali ostali deli sistema. Vsebuje preprost grafični vmesnik za osnovno konfiguracijo, podpira SSH in FTP protokole direktno po namestitvi, za delo s programoma puTTY in WinSCP neposredno iz razvojnega okolja (osebna delovna postaja).
Na delovni postaji je bilo nato urejeno in nameščeno Python okolje za programiranje

glavnega dela programa. Za potrebe izdelave spletnega vmesnika je bil v okolje Raspbian nameščen tudi omenjen LAMP stack in orodje za dodajanje javascript skripte (na slednji temelji posodabljanje vmesnika sistema za nadzor prostora).
Iz GIT repozitorija posredovanega na spletnih straneh izbranega orodja CMUSphinx je bila prenešena zadnja verzija paketov CMUSphinx[9] in Pocketsphinx[10]. Razlog za uporabo obeh paketov je njihova medsebojna interakcija, oziroma delitev v več delov zaradi različnih možnosti namestitve. Segment Sphinxbase vsebuje osnovne knjižnice potrebne za prepoznavo zvoka in kot samostojna še ni pripravljena za uporabo. Potrebna je še namestitev dodatnega dela Pocketsphinx, ki predstavlja vmesnik za zagon prepoznave in za svoje delovanje potrebuje prej nameščene knjižnice. Na mestu paketa Pocketsphinx bi se lahko namestilo tudi paket Sphinx oziroma paket Sphinx - Full, ki pa se od izbranega Pocketsphinx razlikuje po procesorski zahtevnosti. Paket Pocketsphinx je namreč optimiziran za poganjanje na manjših integriranih vezjih ter mobilnih napravah, zaradi česar je bil tudi izbran za konkretno implementacijo v nalogi.
Po namestitvi obeh paketov na sistem je sledil preizkus delovanja in odpravljanje morebitnih napak, kar je bilo izvedeno z uporabo preprostega ukaza *pocketsphinx_continuous -inmic yes*. Ob posredovanju tega ukaza se v konzoli izpiše inicializacija knjižnic Sphinx, nato pa vrstica *Listening....* to pomeni, da je sistem aktiven, in da analizira podatke dobljene preko standardnega nastavljenega vhoda (v tem primeru mikrofon). Ker je ob namestitvi v paketno datoteko vključen tudi prednaučen model, se za prepoznavo glasu uporabi slednji. Tako je možno preveriti, če sistem deluje, vendar je vključen model nenatančen in splošen, kar pomeni, da obstajajo zelo majhne možnosti da bo pravilno prepoznal besedo, ki jo bomo izgovorili.

4.5.1 Učenje modela

Kot en izmed ciljev zaključne naloge je bilo tudi naučiti lasten jezikovni model, za uporabo v konkretnemu sistemu, vendar je to zaradi več razlogov postalo nepotrebno oziroma potratno. Za učenje lastnega modela bi bilo namreč potrebno imeti vsaj 5 ur čistih glasovnih posnetkov večih govorcev, ali pa eno uro v primeru enega uporabnika. [6] Glede na namembnost sistema za uporabo s strani čim več uporabnikov je bila želja uporabiti model, ki bi lahko temu ustrezal. Konkretno, potrebovali bi model, ki je dovolj natančen takoj po namestitvi sistema v prostor in deluje za vse uporabnike prostora brez dodatnega učenja. Ključni vzrok, da je bilo učenje lastnega modela označeno kot neprimerno in neučinkovito za trenutno implementacijo je nepoznavanje orodja za učenje, konkretno poznavanje nastavitev in parametrov pri učenju lastnega

[9]CMUSPHINX `git://github.com/cmusphinx/sphinxbase.git`
[10]POCKETSPHINX `git://github.com/cmusphinx/pocketsphinx.git`

modela. Glede na to, da z tem orodjem nisem bil seznanjen, bi to vzelo neprimerljivo več časa kot uporaba že naučenega modela z še boljšo natančnostjo. Tako je bilo učenje lastnega modela opuščeno.

4.6 Programska zasnova

4.6.1 Slovar in slovnica

Kot posledica odločitve o uporabi prednaučenega modela je bilo potrebno spremeniti potek dela: Naložen model iz spletnega vira je bilo potrebno omejiti s slovnico in slovarjem, da bi bil najbolj efektiven za konkreten primer uporabe. Kot je predstavljeno v teoretičnem delu naloge, je bilo potrebno začrtati vse kombinacije besed, ki bi jih uporabnik lahko potreboval za učinkovito rabo sistema, analizirati vse možne načine in scenarije uporabe in strukturo zapisati.
To je storjeno z izdelavo lastne datoteke .jsgf[11], ki predstavlja slovnico in vsebuje vse možne relacije med besedami, ki so razvrščene po podrazredih (za primer glej prilogo A). Format .jsgf omogoča opisovanje slovnice v načinu ki je združljiv z večino ostalih platform in sistemov. Druga datoteka, ki jo je potrebno izdelati hkrati kot slovnico, je slovar, ki vsebuje fonemski zapis vsake besede, ki je uporabljena v slovnici. Vsako besedo zapišemo v svojo vrstico in zraven dopišemo fonetični zapis izgovorjave (na primer *OFFICE AO F IH S*) ter datoteke .dic, ki predstavlja fonetičen zapis uporabljenih besed oziroma slovar. Ker je bil uporabljen model prenešen s spleta učen na angleškem jeziku, je bilo potrebno tudi jezik ukazov omejiti na angleščino za večjo natančnostjo modela. Po preizkusu zapisov slovenskih besed s fonemi iz angleških besed je bil razviden padec v natančnosti in govorec je mogel besedo izgovoriti nenaravno, da je bila ta prepoznana.

4.6.2 Python skripta

Program temelji na stalnem kroženju zanke v programu, ki se izvaja kadar je naprava v načinu pripravljenosti. Naprava stalno posluša za ukaz oziroma ključno besedo pred ukazom preko vhodne naprave (mikrofona). Kadar izgovorimo ključno besedo se zajem nadaljuje, dokler govorec ne pride do premora. Datoteka se zapiše v medpomnilnik in referenca je nato posredovana dekoderju, ki začne vršiti prepoznavo glasu. Želja je prepoznavo poganjati na posebni niti, zato, da varnostni sistem v času, ko je prepoznava aktivna, ne prekine svojega delovanja, ampak nemoteno deluje naprej. Ko je preko zgoraj omenjenih datotek slovarja in slovnice prepoznava izvršena, glavna skripta dobi

[11]FORMAT .JSGF `https://www.w3.org/TR/2000/NOTE-jsgf-20000605/`

podatek o besedah, ki smo jih izgovorili. Na podlagi izgovorjenih besed se vrši preprosta primerjava in izbira segmenta kode oziroma zunanjega skripta.
Hkrati se na drugi napravi na časovni interval zajame slika iz kamere in enkrat na sekundo preverijo podatki iz senzorjev na vratih. V ozadju teče strežnik, ki je drugim napravam dostopen preko IP naslova. Podatki iz senzorjev in kamere se zapisujejo v podatkovno bazo (slika se zapiše samo z referenco na datoteko v pomnilniku, kar nam prihrani prostor in kompleksnost podatkovne baze), HTML/javascript skripta pa nato podatke prikazuje in spletno stran avtomatsko osvežuje.
V primeru implementacije sistema na eni napravi zaradi cenovnih razlogov, bi obe skripti potekali na enemu sistemu. To bi bilo storjeno z uporabo knjižnice *thread* in ukaza *start_new_thread(skripta_za_ nadzor).*

4.7 Psevdokoda glavnih Python zank

Algoritem 1: Poenostavljen primer glavne zanke sistema. V zanki, ki se neprestano izvaja znotraj programa, se konstantno vrši prepoznava glasu (poenostavljeno z ukazom v zunanjo knjižnico - vrstica 3), nato pa se vršijo različni ukazi glede na prepoznano besedo. To omogoča tudi enostavno dodajanje in spreminjanje funkcij glede na prepoznan ukaz.

```
1  //inicializacija spremenljivk in objektov kot npr. light in weather
2  while True do
3      command = pocketSphinxListener.getCommand()
4      if command.startswith(activation word): then
5          command.replace("the","") //odstrani vezno besedo
6          if command.length == 3 then
7              action = command(0)
8              powerState = command(1)
9              object = command(2)
10         else
11         if command.length = 2 then
12             action = command(0)
13             object = command(1)
14             powerState = null
15         else
16         if object = "light" && powerState != light.currentPowerState then
17             light.switch(!light.currentPowerState)
18         else
19             return
20         if action = "show" && object == weather then
21             weather.show()
22         else
23             return
24         //tukaj so vstavljene še vse druge možnosti, ki so bile skrite zaradi
             dolžine.
25     else
26         go back to the beginning of loop
```

Algoritem 2: Poenostavljen primer zanke nadzornega sistema. Prav tako kot glavni del sistema se tudi tukaj stalno izvaja zanka, v kateri sistem zajame sliko in preveri stanje senzorjev, oziroma če je bila na kateremu koli senzorju sprememba. Ta zanka se od zanke glavnega sistema razlikuje v zakasnitvi, ki je lahko nastavljena na poljubno število sekund. Zanka glavnega sistema pa se izvaja konstantno kakor hitro je mogoče, zaradi čim boljše uporabniške izkušnje in kontinuitete v delovanju.

```
1  //inicializacija spremenljivk in objektov kot npr. sensors in camera
2  while True do
3      camera.capture('image.png')
4      for each sensor in sensors do
5          if sensor.state != sensor.laststate then
6              log.writechange(sensor)
7              sensor.state = !sensor.laststate
8          else
9      updateHtmlData()
10     time.sleep(refreshSpeed)
```

4.7.1 Zunanje skripte

V posebnih datotekah so skripte, katere neposredno kliče procedura za prepoznavo govora, glede na ukaz, ki ga je prepoznala. Na tak način je pridobljena večjo modularnost med skriptami, saj jih je preprosteje dodajati ali modificirati, kot pa če bi bile le-te zapisane kot funkcije v eni datoteki. Skripte delujejo neodvisno od glavne zanke. Ko se klicana skripta izvede, program ponovno preide v stanje pripravljenosti in čaka na izgovorjeno ključno besedo za začetek prepoznave govora. Glede na to, da se glavna zanka neprestano izvaja, mora vsebovati "fail-safe"elemente, ki preprečujejo, da bi se celoten sistem podrl ob napaki v enem od manjših delov.

4.8 Težave

Pri implementaciji programa se je pojavila težava z uporabo integriranega vezja Raspberry Pi. Težava se je pojavila pri sami prepoznavi govora, saj je proces obdelave sprejetega glasu zahteven in potrebuje precej zajetno procesorsko moč. Vezje, ki je bilo uporabljueno od samega začetka projekta (Raspberry Pi 1), je za prepoznavo ukaza potrebovalo tudi do 20 sekund, tudi v načinu izvajanja samo jedra sistema - prepoznave govora (brez hkratnega izvajanja dodatnih skript na istem vezju). To je občutno znižalo uporabnost sistema in uporabnikovo izkušnjo do točke, kjer je bilo potrebno najti rešitev za optimalno delovanje sistema kot celote. Težava je bila rešljiva na več načinov:

- Selitev dela prepoznave govora na namenski strežnik z večjo procesno močjo; to bi pomenilo uporabo več dodatne infrastrukture, ki bi tudi občutno povečala ceno projekta, vendar bi predstavljala največjo procesno moč zaradi uporabe namenske strojne opreme.
- Opustitev uporabe manjšega zunanjega vezja in vršenje prepoznave neposredno na uporabnikovem računalniku; to bi pomenilo obremenjevanje uporabnikovega računalnika z dodatno programsko opremo in bi bilo odvisno od strojne moči uporabnikovega računalnika. Kljub temu bi bila to cenovno najbolj optimalna rešitev.
- Nakup in nadgradnja zunanjega vezja na vezje z večjo procesno močjo; v tem primeru ohranimo infrastrukturo sistema za dokaj minimalno povečanje stroška implementacije.

Težava je bila odpravljena z zamenjavo vezja Raspberry Pi 1 model B z vezjem Raspberry Pi 3 Model B, kar je pomenilo več kot zadostno povečanje v procesni moči

in izvajanje programa je lahko potekalo nemoteno. V tabeli 1 je tudi razvidna razlika v specifikacijah obeh vezij. K boljšemu izvajanju prepoznave predvsem pripomore večje število jeder v arhitekturi novejšega modela ter frekvenca na kateri teče CPU.
Glede na to, da pa je vezje Raspberry Pi 1, ki je bilo zamenjano, še vedno dokaj zmogljivo za preproste naloge in zaželen nabor operacij, se je pojavila možnost dodatnega olajšanja vezja na katerem se poganja prepoznava glasu. Na vezje Raspberry Pi 1 je bil postopoma premeščen drugi del sistema - nadzor prostora. Na ta način povsem ločimo oba dela sistema na dve namenski vezji, kar pomeni največji možen izkoristek in visoko funkcionalnost sistema, vendar na drugi strani prinese večji strošek pri gradnji sistema. Glede na to, da se je možnost uporabe obeh vezij pojavila, je bila izbrana kot optimalna. V primeru gradnje sistema za čim nižjo ceno pa bi se oba dela izvajala na enem, močnejšem vezju.

Tabela 1: Primerjava vezij Raspberry Pi

Model	**Raspberry Pi model 1 B+**	**Raspberry Pi model 3 B**
Arhitektura	ARMv6Z (32-bit)	ARMv8-A (64/32-bit)
Število jeder	1	4
Processor	700 MHz Single Core ARM1176JZF-S	1.2 GHz Quad Core ARM Cortex-A53
RAM	512MB	1 GB
Cena	20 €[a]	35 €

[a]Vezja Raspberry Pi 1 ni več mogoče kupiti iz originalne trgovine, ker je bil zamenjan z kasnejšimi različicami. Možno ga je kupiti rabljenega preko spletnih preprodajaln, kjer se cena giblje okoli 20 €, začetna cena naprave pa je bila 30 €. [15]

[15]

5 Rezultati

5.1 Uporabnost sistema

Sistem, ki je bil razvit v sklopu te naloge, je lahko označen kot zadovoljivo funkcionalen, da pokrije vse sistemske zahtevke, ki so bili postavljeni v samem začetku naloge. Zmožen je prepoznati osnoven nabor ukazov ter na podlagi posredovanih parametrov v ukazu prikazati želeno informacijo ali opraviti preprosto operacijo (prižig električne naprave) Sistem je po osebni oceni dovolj uporaben za vsakodnevno rabo in ne predstavlja preveč zapletenih elementov, ki bi oteževale uporabo. Vezje zavzame dovolj majhen prostor, da je lahko na mizi zraven osebnega računalnika ali na delovnem mestu ter ne moti delovnega procesa.
Edino področje kjer uporaba domačega sistema lahko povzroča težavo je zagon sistema, ki poteka preko SSH protokola oziroma preko ukazne vrstice. Ta problem bi bil lahko rešljiv z uporabo spletnega vmesnika, ali gumbov na sami napravi ter LED diod za prikaz trenutnega stanja sistema. Uporabljen bi bil lahko tudi majhen LCD prikazovalnik, nameščen neposredno na napravo. Glede na stanje razvoja sistema sistem tudi fizično oziroma estetsko ni vrhunski. Potrebno bi bilo uporabiti vsaj ohišje za glavni del, da se izognemo neželenim kratkim stikom med elementi vezja in olajšamo prenos ter namestitev.

Slika 2: Slika vezja Raspberry Pi, brez povezanega napajalnika. Vidna je priklopljena USB zvočna kartica ter uporabljen mikrofon klasičnih slušalk.

```
Poslusanje...
Prepoznan ukaz:
office turn on the light
Poslusanje...
Prepoznan ukaz:
office turn off the light
Poslusanje...
Prepoznan ukaz:
office turn on the fan
connected to remote switch
relay open
done communicating with relay
Poslusanje...
Prepoznan ukaz:
thankyou
you're welcome :)
Poslusanje...
^CNasvidenje.
pi@raspberrypi3:~/diplomatest $
```

Slika 3: Slika izhoda sistema preko konzole PuTTY. Razvidna je prepoznava glasu in odgovarjanje s prednastavljenimi besedami ter komunikacija z zunanjim brezžičnim stikalom (kot primer delovanja)

5.2 Cena

strošek celotnega sistema, ki je opisan zgoraj je zapisan v naslednji tabeli. Razvidno je, da je trenutni sistem "bare minimum" v smislu izbire komponent, saj je bil eden izmed glavnih ciljev projekta tudi ohraniti čim nižjo ceno. Najbolj se uporaba cenejših kom-

ponent pozna pri dometu mikrofona, saj bi prostorski mikrofon pomenil veliko večji strošek v primerjavi z uporabljenim - klasičen mikrofon iz naglavnih slušalk. Taka nadgradnja bi predstavljala tudi največji dvig funkcionalnosti sistema, saj bi lahko uporabnik z napravo upravljal od mnogo dlje stran. Izvedenih je bilo nekaj testov dometa mikrofona in primerjava z razumevanjem izgovorjenega ukaza na različnih oddaljenostih od mikrofona. V kakršnem stanju je sistem z trenutnimi komponentami, je mikrofon zanesljiv na največjo razdaljo približno enega metra oddaljenosti govorca, ob predpostavki da je prostor brez prisotnosti večjih zunanjih motenj (zvok z ulice, ventilator).

Tabela 2: Skupen strošek sistema

Komponenta	**Raspberry Pi model 1 B+**	**Raspberry Pi model 3 B**	**Mikrofon in USB zvočna kartica**	**modul PiCamera v2**	**Spominska kartica**
Cena	20€	30€	15€	10€	20€
Kritičnost za sistem	Nizka*	Visoka;	Visoka	Visoka	Visoka

*del, ki se izvaja na vezju, bi lahko v celoti prevzelo vezje Raspberry Pi model 3.

V ceno niso vključeni dodatni drobni stroški kot so na primer napajalnik za vezje, priklopi za stikala, mrežni kabli in LED luči (uporabljene zgolj za ponazoritev delovanja stikala). V nadaljevanju, dodaten del stroškov bi predstavljali releji in brezžični moduli, ki bi jih bilo potrebno namestiti na hišno omrežje za vklapljanje in izklapljanje naprav. Ta strošek ni bil vključen v tabelo, glede na to, da bi tudi v primeru nakupa že končanega sistema podobnega našemu (Google Home, Amazon Echo) bilo potrebno storiti isto in to predstavlja dodaten strošek v obeh primerih. V sledeči tabeli je primerjan strošek za nakup že narejenega sistema s stroškom implementacije iz te naloge. Po podatkih cene ostalih naprav iz istega področja je razvidno, da je cenejši samo Amazon Echo Dot, ostale naprave pa so dražje od domače uresničitve podobne funkcionalnosti.

Tabela 3: Primerjava cen sistemov

Komponenta	**Domači sistem**	**Amazon Echo 2nd generation**	**Google Home**	**Amazon Echo Dot**
Cena	pribl. 70€	84€	110€	50€

[3]

5.3 Uspešnost naloge

Na podlagi zaključkov iz vidika uporabnosti in vidika cene je razvidno, da lahko označimo implementacijo kot uspešno, vendar z svojimi negativnimi stvarmi. Težavo predvsem predstavlja domet naprave ter delo potrebno za implementacijo v prostor. Zaključeni kupljivi sistemi so v veliki večini že pripravljeni za uporabo out-of-the-box, sistem implementiran v okviru te naloge pa bi potreboval še konfiguracijo in nameščanje glede na uporabnikove potrebe. Vsekakor lahko pridemo do zaključka, da je na sistemu možno še vedliko nadgradenj in lahko služi kot odlična osnova za razvoj domačih projektov.

6 Zaključek

6.1 Razširljivost

Glede na obsežnost same naloge in rabo večih različnih segmentov sistema je na voljo veliko možnosti za izboljšave in nadgradnje. Veliko možnosti zaradi same velikosti ni bilo možno zajeti v sistem, ker bi implementacija le-teh pomenila večanje samega stroška sistema, ali pa bi občutno povečala čas porabljen za izdelavo. Kot možne razširitve obstoječega sistema lahko navedemo:

- Možnost oddaljenega dostopa do sistema: Preko zaščitenega protokola bi uporabnik lahko dostopal do nadzorne plošče sistema, si ogledal zadnje spremembe na senzorjih oz. spremembah v prostoru, preveril trenutno stanje preko slikovnega gradiva in nastavljal parametre sistema. Za implementacijo te nadgradnje bi bilo potrebno nastaviti oddaljen dostop oz. VPN dostop do notranje mreže, na kateri bi bil strežnik oziroma spletni vmesnik sistema. To bi lahko pomenilo dodatno varnostno luknjo glede na to, da odpiramo notranjo mrežo (LAN) v zunanjo (WAN).

- Nadgranja sistema za nadzor z implementacijo e-obveščanja: V primeru, da bi uporabnik vklopil to funkcijo, bi sistem pošiljal e-sporočila o spremembah na senzorjih v prostoru na prednastavljen naslov. Sporočilu bi bila dodana tudi slika prostora oziroma objekta v tistem trenutku. Uporabniku bi to lahko pomenilo dodaten nivo varnosti, saj bi bil v primeru neželenega vstopa v prostor obveščen. Za implementacijo te nadgradnje bi bilo potrebno konfigurirati e-poštni strežnik,

- Uporabniku prijazen vmesnik z nastavljivimi parametri: Ta segment bi predvsem prišel prav v delu sistema za nadzor prostora. uporabnik bi skozi omenjen vmesnik imel možnost nastaviti število nameščenih senzorjev v prostoru ali objektu, parametre zajema slike ter parametre sistema za obveščanje. Za implementacijo te nadgradnje bi bilo potrebno postaviti spletno stran z avtentikacijo (glede na to, da za trenutno stanje sistema ne moremo označiti da je varen), ki bi se izvajala na strežniku. Potrebna bi bila zadostna back-end komunikacija z izvajajočo Python skripto, da bi uporabnik lahko nastavil vse parametre med samim delovanjem sistema.

- Odzivanje na uporabnikove ukaze z glasom: Da bi uporabnik občutil pristnejšo izkušnjo uporabe bi bilo možno v sistem dodati segment text-to-speech, ki bi uporabniku posredoval zvočne informacije preko računalniško branega govora. Za ta namen bi lahko bil uporabljen Googlov Text-To-Speech API, v kodi programa pa bi bilo potrebno predprogramirati odzive sistema.

- Napredni nadzor nad osebnim računalnikom; Z uporabo primerne komunikacije in skrbjo za varnost bi bilo mogoče sistem povezati z osebnim računalnikom, za razširitev aspekta govornega nadzora. Uporabnik bi lahko glasovno zaklenil računalnik, odpiral programe in nadziral sistem preko osebnega oziroma delovnega računalnika. Potrebno bi bilo le posvetiti posebno pozornost varnosti, saj se z takim posegom lahko odpre neželena varnostna luknja do osebnega računalnika.

V vsakem primeru je razvidno, da je tudi na podlagi rezultatov tak sistem upravičen za implementacijo, vendar je le-ta neprimerno bolj zahtevna kot preprost nakup katere koli druge naprave na trgu. Implementiran sistem pade v enak oziroma malce nižji cenovni razred, vendar kljub temu pridobimo na funkcionalnosti in razširljivosti, saj lahko vsak posameznik z nekaj programerskega znanja nadgradi sistem po svojih željah in potrebah. Po trenutni funkcionalnosti je mogoče sklepati, da bi za implementacijo sistema v večjih poslovnih stavbah ali podjetjih bili primorani bolje razdelati vsaj sam vmesnik, velika verjetnost pa obstaja, da bi bilo za kaj takega potrebno predelati celoten sistem.

6.2 Osebno mnenje

Kot ustvarjalec v tej nalogi opisanega sistema imam namen na projektu še delati naprej, saj v njem vidim uporabno orodje za vsakodnevno uporabo. Zanima me tudi kje je meja vezja Raspberry Pi in kaj vse je z trenutno konfiguracijo mogoče implementirati v obstoječ sistem. Namen imam preizkusiti tudi operacijski sistem IoT core na omenjenem vezju, za primerjavo funkcionalnosti, ki jo nudi proti sistemu Raspbian in za raziskavo delovanja okolja.

7 Literatura in viri

[1] Amebis. Gigafida - O korpusu, 2018. *(Citirano na strani 5.)*

[2] Anyy. What is an Acoustic Model? - voxforge.org, 2013. *(Citirano na strani 7.)*

[3] Brian Heater. Comparing Alexa, Google Assistant, Cortana and Siri smart speakers — TechCrunch, 2017. *(Citirano na strani 26.)*

[4] Cantab Research Ltd (Speechmatics). Products Pricing - Speechmatics, 2018. *(Citirano na strani 10.)*

[5] CMU Sphinx. CMU Sphinx Project, 2009. *(Citirano na strani 14.)*

[6] CMUSphinx. CMUSphinx Tutorial For Developers – CMUSphinx Open Source Speech Recognition, 2017. *(Citirano na straneh 11 in 17.)*

[7] DialogFlow. Overview — Dialogflow, 2017. *(Citirano na strani 10.)*

[8] J. Fry. BIGRAMS AND TRIGRAMS. Technical report, 2011. *(Citirano na strani 7.)*

[9] Google. Cloud Speech-to-Text - Speech Recognition — Cloud Speech-to-Text API — Google Cloud, 2018. *(Citirano na strani 10.)*

[10] J. Howard and S. Ruder. Fine-tuned Language Models for Text Classification. *arXiv*, 2018. *(Citirano na strani 3.)*

[11] N. I. o. T. Julius project team. Open-Source Large Vocabulary CSR Engine Julius, 2014. *(Citirano na strani 12.)*

[12] Kmaclean. What is a Grammar? - voxforge.org, 2010. *(Citirano na straneh 7 in 8.)*

[13] Melanie Pinola. History of voice recognition: from Audrey to Siri — IT Business, 2011. *(Citirano na strani 2.)*

[14] Microsoft. Microsoft Speech Service — Microsoft Docs, 2018. *(Citirano na strani 10.)*

[15] ModMyPi. ModMyPi — Raspberry Pi Comparison Table, 2017. *(Citirano na strani 23.)*

[16] K. Povey, Daniel and Ghoshal, Arnab and Boulianne, Gilles and Burget, Lukas and Glembek, Ondrej and Goel, Nagendra and Hannemann, Mirko and Motlicek, Petr and Qian, Yanmin and Schwarz, Petr and Silovsky, Jan and Stemmer, Georg and Vesely. The Kaldi Speech Recognition Toolkit, 2011. *(Citirano na strani 11.)*

[17] M. Sahidullah and G. Saha. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. *Speech Communication*, 54(4):543–565, may 2012. *(Citirano na strani 4.)*

[18] Simon. About Simon — Simon, 2017. *(Citirano na strani 11.)*

[19] UWP. Speech recognition - UWP app developer — Microsoft Docs, 2018. *(Citirano na strani 11.)*

[20] Vocapia Research SAS. Speech to Text API — Vocapia, 2018. *(Citirano na strani 10.)*

[21] A. Waibel and K.-F. Lee. Readings in speech recognition. In *Readings in speech recognition*. 1990. *(Citirano na strani 2.)*

Priloge

A Slovnica

```
#JSGF V1.0;
grammar commands;

public <command> = <sprozitelj> <akcija> THE <objekt> |
                       LOUDER              |
                       QUIETER             |
                       THANKYOU         |
                       THANK YOU           |
                       BYE-BYE             |
                       BYE                 |
                       NEXT             ;

<akcija> = TURN ON |
      TURN OFF |
      SHOW    ;

<sprozitelj> = OFFICE |
        HOUSE   ;

<objekt> = LIGHT |
        FAN     |
      WEATHER    ;
```

B Slovar

HOUSE HH AW S
OFFICE AO F IH S
VOLUME V AA L Y UW M
WEATHER W EH DH ER
UP AH P
NEXT N EH K S T
PREVIOUS P R IY V IY AH S
QUIETER K W AY AH T ER
HI HH AY
BYE B AY
BYE-BYE B AY B AY
LOUDER L AW D ER
THANK TH AE NG K
THANK'S TH AE NG K S
THANKYOU TH AE NG K Y UW
YOU Y UW
DOWN D AW N
ON AA N
ON(2) AO N
OFF AO F
LIGHT L AY T
FAN F AE N
TURN T ER N
THE DH AH
SHOW SH OW