



STUDIJSKI PROGRAM
SAVREMENE INFORMACIONE TEHNOLOGIJE

**IMPLEMENTACIJA SISTEMA ZA SIMULACIJU
INTELIGENCIJE U AUTOMATIZOVANOM DOMU**

- MASTER RAD -

Mentor

Prof. dr Mladen Veinović

Kandidat

Tair Milan
2012/410033

Beograd, 2013.

SADRŽAJ

1. APSTRAKT	3
2. UVOD	4
3. AUTOMATIZACIJA DOMA	5
4. KORISTI OD AUTOMATIZACIJE DOMA	5
5. STANDARDI ZA POVEZIVANJE UREĐAJA	6
5.1. X10 standard	6
5.2. Z-Wave standard	6
5.3. Insteon	7
5.4. Universal powerline bus	7
5.5. ZigBee standard	7
6. KONTROLNA JEDINICA SISTEMA	8
6.1. Hardverska platforma kontrolne jedinice	8
6.2. Periferije kontrolne jedinice	9
6.3. Softverska platforma kontrolne jedinice	10
6.4. Zaštita prenosa podataka u lokalnoj mreži	10
6.5. Autentifikacija klijentskih upravljačkih jedinica	10
7. SOFTVER ZA UPRAVLJANJE ELEKTRIČNIM UREĐAJIMA	11
7.1. Prihvatanje zahteva od klijenta	12
7.2. Izvršavanje naredbe korisnika	13
7.3. Pokretanje programa iz PHP skripte	16
8. SERVISNI PROGRAMI	17
8.1. Klasa Module	17
8.2. Pojedinačni moduli	17
8.3. Instalacija modula	18
8.4. Registrovanje modula	18
8.5. Aktivacija modula	19
8.6. Metod za utvrđivanje spremnosti modula za izvršavanje naredbe	19
8.7. Registar regularnih izraza modula	19
8.8. Priprema odgovora modula	20
8.9. Formatiranje odgovora modula	21
8.10. Hijerarhija klasa	22
9. KREIRANJE PROFILA KORISNIKA	23
9.1. Kreiranje profila na osnovu evidencije događaja	23
9.2. Kreiranje profila nedeljnim praćenjem ponašanja	25
10. AKTIVNO AŽURIRANJE PRIORITETA PROFILA KORISNIKA	29
11. ANALIZA GOVORNIH KOMANDI	31
12. REGULARNI IZRAZI	31
12.1. Poreklo regularnih izraza	31
12.2. Sintaksa regularnih izraza	32

13. ZNAČENJE SPECIJALNIH KARAKTERA U REGULARNIM IZRAZIMA	32
13.1. Karakter tačka .	32
13.2. Karakteri nabrojani unutar uglastih zagrada [i]	32
13.3. Opseg karaktera u uglastim zgradama [i]	33
13.4. Broj ponavljanja šablona	33
13.5. Ponavljanje nula ili više puta	34
13.6. Negacija izraza	34
13.7. Idvajanje delova na samom početku teksta	34
13.8. Izdvajanje delova na samom kraju teksta	35
13.9. Izdvajanje jednog ili više ponavljanja dela izraza	35
13.10. Izdvajanje dela izraza nula puta ili jednom	35
13.11. Operator unije ili izbora	36
13.12. Specijalni karakteri kao literali	36
13.13. Skraćenice za najčešće korišćene opsege karaktera	37
13.14. Rad sa unikodnim karakterima	38
14. IZDVAJANJE DELOVA TEKSTA POMOĆU REGULARNIH IZRAZA	39
14.1. Izdvajanje brojeva zapisanih u više formata	39
14.2. Izdvajanje rečenica	41
14.3. Izdvajanje stranih reči iz ćiriličnog teksta	42
15. TUMAČENJE GOVORNE KOMANDE	43
15.1. Modul za tumačenje komande zahteva za definicijom	43
15.2. Modul za ekstrakciju elemenata iz govorne komande	45
15.3. Identifikovanje funkcionalnih celina	45
15.4. Ekstrakcija funkcionalnih celina regularnim izrazima	46
15.5. Modul za normalizaciju tipova podataka	46
16. IZVRŠAVANJE GOVORNE KOMANDE	48
16.1. Modul za izvršavanje naredbe	48
16.2. Registar kontrolera	48
16.3. Kontroler za izvršavanje naredbe	49
17. SAOPŠTAVANJE REZULTATA	51
18. MODUL ZA NORMALIZACIJU REZULTATA	51
19. SINTEZA GOVORA	53
19.1. Sinteza govora spajanjem snimljenih delova govora	53
19.2. Sinteza govora fonetskom transkripcijom	53
19.3. Reprodukција govora	54
20. ZAKLJUČAK	55
21. LITERATURA	56

1. APSTRAKT

Rad na temu „Implementacija sistema za simulaciju inteligencije u automatizovanom domu“ napisao sam sa ciljem da predstavim praktična rešenja za automatizaciju i monitoring procesa u stanu, kući ili kancelariji, kao i za stvaranje komfornijeg ambijenta za boravak i funkcionisanje u tim sredinama.

U radu je predstavljeno korišćenje SBC računara Raspberry Pi kao osnove oko koje je izgrađena hardverska platforma sistema, dok je za izradu softverske platforme opisano korišćenje besplatnih rešenja otvorenog koda, kao što je Raspbian distribucija GNU/Linux operativnog sistema, skripting jezik PHP, ali i programski jezik Java. Za komunikaciju između sistema i pametnih uređaja u domu predstavljena je mogućnost korišćenja industrijskih standarda, ali i klasičnog TCP/IP standarda koji je u domaćinstvima najzastupljeniji i najlakši za implementaciju.

Predstavljani su izvodi izvornog koda za pojedine funkcionalnosti sistema, kao i primeri regularnih izraza koje sistem koristi za tumačenje govornih naredbi korisnika, uz detaljna objašnjenja za tumačenje regularnih izraza koji su korišćeni za izradu modula sistema.

1. ABSTRACT

The topic of this thesis is “The implementation of the intelligence simulation system for automated homes”. It is written with aim to present a practical approach and solution for automation and monitoring processes in a flat, house or office in order to provide a more comfortable living and working environment.

This thesis describes the use of Raspberry Pi SBC as the core for building the system's hardware platform. The software platform is based on free and open source solutions, such as the Raspbian distribution of GNU/Linux operating system, PHP scripting language, Java programming language etc. I have researched the possibility to use both the industrial standards and the standard TCP/IP protocol for communication between smart home devices. The TCP/IP was selected for implementation because it is most common and easiest to implement.

I have listed excerpts of the source code for certain key modules of the system, as well as a number of regular expressions used for user voice command interpretation, along with detailed explanation of regular expressions used for development of system's module for voice command interpreting.

2. Uvod

Sistem za simulaciju inteligencije u automatizovanom domu je kolekcija hardvera i softvera sastavljena na taj način da omogući automatizaciju u stanu, kući ili kancelariji korišćenjem rešenja otvorenog koda koja su besplatna i mogu se slobodno koristiti, kopirati, samostalno razvijati i menjati uz zadržavanje osnovnih principa iz licence otvorenog koda, a to je da kôd ostane otvoren, dostupan i da može da se implementira u sličnim tehničkim rešenjima i sistemima.

U ovom radu je opisan ceo sistem koji omogućava interaktivno korišćenje pametnih uređaja koji su raznim telekomunikacionim standardima povezani na centralnu jedinicu sistema. Objasnjena je primena regularnih izraza za tumačenje komandi zadatih govornim jezikom, upravljanje odlučivanjem o delegiranju izvršne funkcije modulima sistema, kontrola toka analize govorne komande i postupak sastavljanja adekvatnog odgovora sistema korisniku koji zadaje komandu.

Za hardversku platformu je korišćen RaspberryPi SBC (*single board computer*), koji odlikuju niska cena, a ujedno i odličan odnos cena-performanse. U pitanju je univerzalna računarska jedinica sa standardizovanim konektorima za periferije, ali i sa namenskim digitalnim portom za upravljanje uređajima za koje ne postoji standardni konektor za PC arhitekturu. U radu su opisani svi elementi hardverske platforme koja se koristi, kao i uopštenim načinom korišćenja komponenata jedinice.

Za softversku platformu je korišćen softver otvorenog koda. Operativni sistem koji se koristi kao osnova za sve programe koji se na RaspberryPi računaru pokreću je Debian distribucija Gnu/Linux operativnog sistema specijalno prilagođena RaspberryPi i ARM procesorskoj arhitekturi, a pod imenom Raspbian. Nad ovim operativnim sistemom je instalirano standardno Veb hosting okruženje, tj. paket softvera za pokretanje veb servisa, pod imenom LAMP, što je skraćeno od Linux Apache MySQL PHP.

Lokalni veb servis je korišćen da obezbedi mogućnost udaljenog upravljanja sistemom za simulaciju inteligencije pomoću uređaja u mreži i preko Interneta, kao što je upravljanje i monitoring preko mobilnog telefona, tablet računara itd.

Servis za kontrolu periferija na namenskom portu je pisan u Java programskom jeziku i za potrebe pokretanja Java programa instaliran je najnoviji JRE sa Java VM.

3. AUTOMATIZACIJA DOMA

Sistemi za automatizaciju doma, poznati i kao sistemi kućne automatike ili ukratko pametna kuća u najširem smislu, podrazumevaju integrisanje prvenstveno hardverske komponente, tj. električne uređaje u domaćinstvu u sistem kojim može da se upravlja sa jednog centralnog mesta, tj. kontrolne jedinice sistema.

Kontrolnoj jedinici sistema za upravljanje integrisanim uređajima, korisnik može da upućuje zahteve za uključivanje ili isključivanje uređaja, pribavljanje statusnih informacija i drugih podataka o radu uređaja kojim se upravlja. Ovi zahtevi mogu da se upućuju sistemu daljinskim upravljačima radio signalom, infracrvenim signalom ili drugim kanalom prenosa komande, mobilnim telefonom, tablet računarom, konzolom ili sličnim uređajem koji može da omogući interakciju korisnika i sistema putem posebno dizajniranog korisničkog interfejsa.

4. KORISTI OD AUTOMATIZACIJE DOMA

Automatizacijom doma se stvaraju razne mogućnosti koje imaju za cilj da povećaju komfor, kvalitet življenja ili promenu životnog stila, ispunjenje raznih potreba i navika korisnika sistema koji živi ili boravi u automatizovanom domu itd.

Neke od koristi koje se omogućava automatizacija doma su:

- Povećanje sigurnosti u domu
- Kontrola audio i video aparata
- Kontrola kućne rasvete
- Kontrola sistema za klimatizaciju prostora (hlađenje i grejanje)
- Povećanje energetske efikasnosti upravljanjem potrošnje električne energije
- Prilagođavanje životnog prostora potrebama korisnika sistema koji borave u prostoru
- Digitalni kućni pomoćnik (alarmi, podsetnici i druge funkcije)
- Integrisane komunikacije (telefon, IM programi itd.)
- Nadzor sa udaljene lokacije
- Upravljanje sa udaljene lokacije
- Briga o kućnim biljkama itd.

5. STANDARDI ZA POVEZIVANJE UREĐAJA

Električni uređaji koji se povezuju u sistem za automatizaciju doma komuniciraju sa centralnom jedinicom za kontrolu sistema. Komunikacija može da se vrši kroz više kanala: bežičnim, žičnim, optičkim, itd. Bez obzira koji vid komunikacije, odnosno medij prenosa signala se koristi, uređaj i sistem moraju da se razumeju, a to se obezbeđuje tako što svi uređaji u domu implementiraju određeni komunikacioni standard. Postoji više komunikacionih standarda i izbor standarda zavisi od količine i vrste podataka koji se prenose između kontrolne jedinice i uređaja u sistemu.

Ukoliko su svi uređaji u sistemu posebno razvijani od strane jednog proizvođača, onda postoji mogućnost da proizvođač sam definiše svoj komunikacioni standard i protokole. Ako je cilj da se napravi univerzalni sistem za automatizaciju doma, potrebno je da se poštuju komercijalno priznati standardi.

5.1. X10 STANDARD

Standard X10 je specifičan po tome što za komunikaciju između centralne jedinice i uređaja nije potrebno sprovoditi posebnu komunikacionu infrastrukturu, jer standard X10 propisuje korišćenje mreže provodnika i instalacije za prenos i distribuciju električne energije u prostorijama. Prenos signala se vrši tako što se u naponske kablove indukuju visokofrekvencijski signali u kratkim intervalima, tako da predstavljaju digitalni signal, odnosno prisustvo ili odsustvo signala predstavlja logičku jedinicu ili nulu, respektivno.

X10 protokol je star i jednostavan protokol i često nije dovoljan u savremenim sistemima koji koriste Internet i koji zahtevaju prenos složenih podataka i velike količine podataka, osim jednostavnih komandi propisanih X10 protokolom.

Ovaj standard se uglavnom više ne primenjuje u modernim, tzv. Post-X10 sistemima.

5.2. Z-WAVE STANDARD

Z-Wave standard je sličan X10 standardu u tom pogledu da koristi radio signali slabog intenziteta za prenos signala, ali za razliku od X10 ne koristi naponske kablove, već je u pitanju bežični prenos signala.

Za komunikaciju između uređaja se koriste radio talasi male snage i frekvencije oko 900MHz, što ne omogućava prenos velike količine podataka kao što je slučaj kod WiFi prenosa čije su radne frekvencije oko 2,4GHz. Brzina prenosa podataka je 9.600 bit/s ili 40kbit/s. Standard propisuje mogućnost Mančester kodiranja, tj. faznog kodiranja signala, kako bi se omogućio veći broj kanala na istoj frekvenciji.

5.3. INSTEON

Insteon je komunikacioni standard koji kao i X10 koristi postojeće instalacije za prenos signala ili bežični radio prenos. Signal kojim se prenose podaci kroz instalacije je frekvencije 131,65MHz, a kada se podaci prenose bežičnim putem, frekvencija signala je u opsegu od 902 do 924MHz.

Standardom su propisane dve vrste komandi koje se šalju i one su određene fiksne dužine: Standardna od 10 bajtova i Proširena od 24 bajta. Prenos podataka je spor i uglavnom je oko 2,880 bit/s.

5.4. UNIVERSAL POWERLINE BUS

Universal powerline bus komunikacioni standard, često samo UPB, koristi postojeće naponske instalacije za prenos signala. Ovaj standard je dosta složen i omogućava pravljenje komunikacionih uređaja raznih kompleksnosti: Od jako jednostavnih i jeftinih uređaja koji podržavaju svega nekoliko komandi, do jako složenih i skupih koji omogućavaju prenos složenih podataka i informacija.

5.5. ZIGBEE STANDARD

ZigBee standard je komunikacioni standard koji omogućava veliku fleksibilnost i dobre performanse. ZigBee je baziran na IEEE 802 standardu i prilagođen je za formiranje lokalnih bežičnih mreža za umrežavanje električnih uređaja opremljenih sa digitalnim primopredajnicima male snage.

Prednosti ZigBee komunikacionog standarda je u tome što, bez obzira na to što radi na niskim snagama, ima mogućnost prenosa signala na veće udaljenosti kroz postojeću mrežu uređaja koji se nalaze na putu signala, a u istoj su ZigBee mreži.

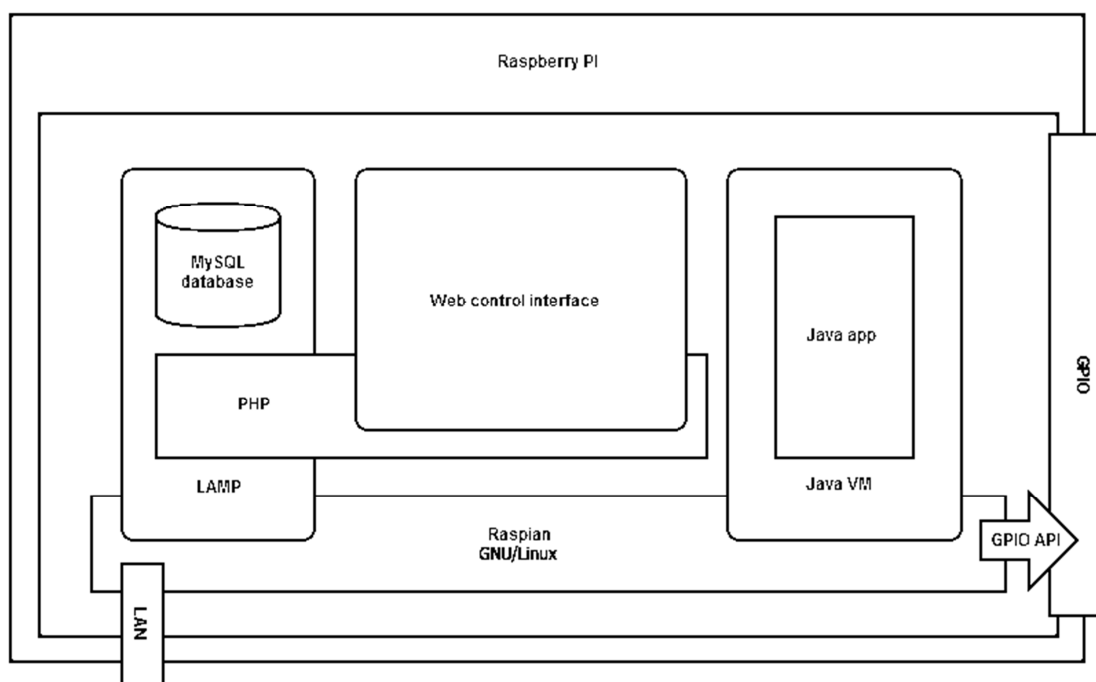
Brzina prenosa podataka je spora i reda veličine 250 kbit/s, a komunikacija je šifrovana 128 bitnim ključem. ZigBee moduli su malih dimenzija, oko 2cm sa 3cm u tipičnom pakovanju i male cene su.

6. KONTROLNA JEDINICA SISTEMA

Kontrolna jedinica sistema je hardversko-softverska komponenta koja koordinira uvezanim uređajima, upravlja njima, prati kretanje korisnika, tumači govorne komande i komande pokretima, tj. gestovima, te reaguje na njih u skladu sa zahtevom.

Kontrolna jedinica je mozak sistema. U softverskom smislu, to je složen set programa uvezanih na taj način da pruže utisak inteligentnog odlučivanja, ponašanja sistema.

Hardverska platforma na kojoj je realizovana kontrolna jedinica sistema je RaspberryPi SBC (single board computer) sa instaliranom distribucijom Debian Linux operativnog sistema prilagođenog RaspberryPi računaru i Apache Veb serverom za komunikaciju sa komponentom upravljanja.



Dijagram 1 - Kontrolna jedinica

6.1. HARDVERSKA PLATFORMA KONTROLNE JEDINICE

Hardverska platforma kontrolne jedinice je RaspberryPi integrisani računar. Integrisani računar RaspberryPi je veličine kreditne kartice i pokreće ga ARM-ov procesor ARM1176JZF-S na 700Mhz koji ima mogućnost overklokovanja na 1GHz u okviru Broadcom-ovog BCM2835 SoC čipa. Mikroračunar je bez integrisane stalne memorije u vidu hard diska, umesto kojeg se koristi SD kartica. Cena ovog računara u trenutku predstavljanja bila je 35 američkih dolara. Ta cena se do trenutka pisanja ovog rada nije menjala. Potrošnja električne energije je od 2,5 do 3,5W. Napaja se USB adapterom od 5V.

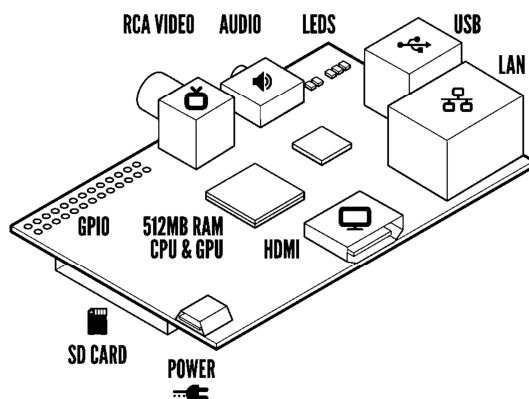
Ploča je opremljena Ethernet mrežnom karticom 10/100, USB port, audio karticom, kompozitnim RCA izlazom, kao i HDMI izlazom za digitalnu sliku.

Jedna od najbitnijih karakteristike RaspberryPi ploče za primenu u sistemima za automatizaciju doma je postojanje GPIO, UART-a i I²C komunikacionog porta.

6.2. PERIFERIJE KONTROLNE JEDINICE

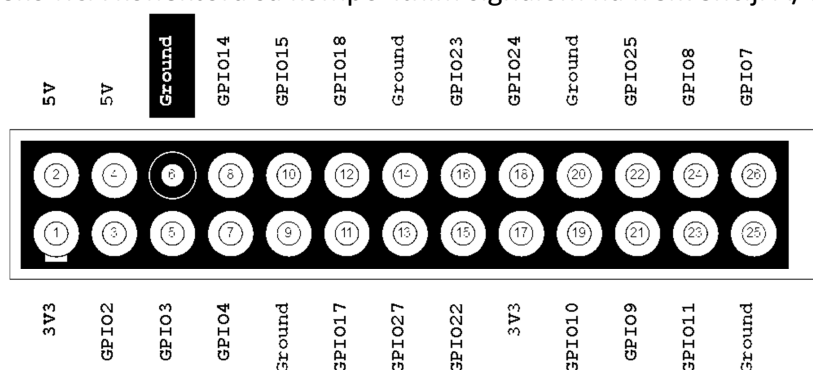
RaspberryPi računar ima GPIO (general purpose input output) port sa 8 ulazno izlaznih parova i UART pinovima za serijsku komunikaciju.

RASPBERRY PI MODEL B



Slika 1 - Ilustracija komponenata na RaspberryPi ploči

Pored ovih portova za povezivanje elektronskih uređaja posebne izrade i namene za potrebe automatizacije, postoji i USB port na koji se po potrebi može povezati tastatura za upravljanje softverom. Monitor se može povezati preko HDMI konektora ili se može povezati na televizor preko RCA konektora sa kompozitnim signalom na frekvenciji A/V kanala.



Slika 2 - Raspored pinova na GPIO portu Raspberry Pi računara

Periferijske jedinice se sa kontrolnom jedinicom sistema povezuju na dva načina.

1. Prvi način je pomoću TCP/IP protokola putem mrežne infrastrukture realizovane UTP kablovima ili bežično preko standardnih Wi-Fi konakcija ka ruteru na koji je povezana kontrolna jedinica, a koja je u istom adresnom opsegu. Kontrolna jedinica se ponaša kao server koji prihvata konekcije udaljenih pametnih uređaja i prihvata izveštaje o njihovom statusu, radu i ostalim parametrima.
2. Drugi način je direktno povezivanje na Raspberry Pi GPIO pinove. Ovaj vid veze je najlakši za implementaciju na malim udaljenostima, ali je za veće razdaljine nepogodan zbog pada napona signala, tj. nepouzdanog prenosa podataka. Ovaj vid povezivanja omogućava 17 priključaka u osnovnoj konfiguraciji, ali može da se proširi na više hiljada mesta za vezu uvođenjem adresiranja.

6.3. SOFTVERSKA PLATFORMA KONTROLNE JEDINICE

Operativni sistem kontrolne jedinice je Raspbian distribucija GNU operativnog sistema izgrađenog oko Linux Kernela. Raspbian je verzija Debian Linux distribucije posebno prilagođena za hardversku platformu RaspberryPi računara.

Komunikacija između kontrolne jedinice i korisnika se obavlja posredstvom softvera napisanog u obliku Veb servisa. Ovaj program je napisan u skriptnom jeziku PHP. Ovaj softver se izvršava na Apache Veb serveru, a za trajno skladištenje informacija koje se obrađuju koristi internu MySQL bazu podataka.

Softver za upravljanje električnim kolima i uređajima povezanim na periferije RaspberryPi računara, tj. kontrolnu jedinicu, pisan je u programskom jeziku Java, kroz JavaME SDK - Java Mobile Edition.

6.4. ZAŠTITA PRENOSA PODATAKA U LOKALNOJ MREŽI

Komunikacija između Veb servisnog softvera kontrolne jedinice i mobilne aplikacije upravljačke jedinice preko koje korisnik vrši nadzor i ručno upravljanje uređajima u automatizovanom domu se vrši bežičnim putem u lokalnoj mreži.

Šifrovanje komunikacije se vrši upotrebom SSL sertifikata na strani servera i klijenta. Privatni i javni ključevi za SSL komunikaciju su ručno definisani i ugrađeni u komponente sistema, tj. na server i na sve klijentske sisteme, tako da ne dolazi do razmene ključeva kroz mrežu u trenutku uspostavljanja komunikacije.

6.5. AUTENTIFIKACIJA KLIJENTSKIH UPRAVLJAČKIH JEDINICA

Pod autentifikacijom se podrazumeva korišćenje bilo koje tehnike koja prijemniku omogućava automatsko identifikovanje i odbacivanje poruka koje su namerno izmenjene ili su izmenjene zbog grešaka u kanalu.

Korisnik klijentske upravljačke jedinice tokom uspostavljanja konekcije ka serveru dokazuje svoj identitet kroz proces utvrđivanja identiteta koji podrazumeva prenos jedinstvenog ID broja klijentske upravljačke jedinice i PIN koda uparenog za preneti ID.

Server nakon provere i potvrde identiteta na osnovu uparivanja ID broja i PIN koda generiše autentifikacioni token koji u kodiranom obliku prosleđuje klijentskoj aplikaciji da bi sve poruke, koje se u buduću budu slale u smeru klijent-server, bile prvenstveno šifrovane određenim algoritmom, tako što će se kao ključ za šifrovanje poruka koristiti token, a sama poruka će dodatno biti šifrovana redovnim načinom koji podrazumeva prenos podatka u lokalnoj mreži SSL sertifikatom.

Autentifikacija se vrši samo jednom prilikom uspostavljanja konekcije i traje dokle god je konekcija aktivna. Po definiciji, konekcija je trajna, tj. ne prekida se. Prekid konekcije može da nastane na strani mobilne aplikacije na korisničkoj upravljačkoj jedinici ili na strani kontrolne jedinice sistema kada se prekine izvršavanje klijentske aplikacije ili veb serverskog procesa, respektivno.

7. SOFTVER ZA UPRAVLJANJE ELEKTRIČNIM UREĐAJIMA

Na kontrolnu jedinicu je preko GPIO porta povezana grupa električnih kola za upravljanje relejima za napajanje električnih uređaja visokog napona, kao i kola koja služe kao komunikacioni interfejs ka udaljenim električnim uređajima.

Ovim električnim kolima se upravlja na dva načina:

- Prekidački, što podrazumeva promenu stanja na tri načina:
 - Komanda za uključivanje uređaja
 - Komanda za isključivanje uređaja
 - Komanda za promenu trenutnog stanja uređaja (state toggle)
- Slanjem složene komande, što podrazumeva slanje niza bitova tačno određenim redosledom za svaku posebnu komandu koja se upućuje tačno adresiranom udaljenom uređaju.

Softver za slanje signala i upravljanje GPIO portom RaspberryPi računara je pisan u programskom jeziku Java, korišćenjem JavaME verzije specijalno prilagođene uređajima malih performansi i sa ograničenim resursima kao što su mobilni telefoni, ugrađeni računari itd.

Primer koda za kontrolu GPIO porta RaspberryPi računara je dat u sledećem listingu:

```
GpioController gpio = GpioFactory.getInstance(); // Inicijalizacija GPIO porta

GpioPinDigitalOutput o =
    gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "O", PinState.LOW);
o.low(); // Postavljanje vrednosti prvog pina na logičku nulu
o.high(); // Postavljanje vrednosti prvog pina na logičku jedinicu

GpioPinDigitalInput i =
    gpio.provisionDigitalInputPin(RaspiPin.GPIO_02, "I", PinPullResistance.PULL_DOWN);
boolean v = i.isHigh(); // Smeštanje logičkog stanja 2. pina u promenljivu v
```

Da bi se izlistani kôd kompajlirao, potrebno je da se u projekat uključi biblioteka Pi4J. Biblioteka Pi4J je besplatna biblioteka posebno napisana za razvoj Java aplikacija za RaspberryPi hardversku platformu.

Kompajlirani program mora da se izvrši pod root korisnikom na operativnim sistemima sa Linux kernelom da bi imao pristup manipulisanju stanjima GPIO porta, kao i da bi pristupio registru za čitanje stanja pinova ovog porta.

7.1. PRIHVATANJE ZAHTEVA OD KLIJENTA

Aplikacija koja upravlja pinovima GPIO porta može da bude napisana tako da osluškuje TCP port korišćenjem ServerSocket objekata i da po uspostavljenoj konekciji sa klijentom prihvati zahtev, a iz njega izvuče parametre u obliku parova ključ-vrednost. Analizom ovih vrednosti, aplikacija odlučuje koji pin treba da dovede na stanje logičke nule, jedinice, da zameni stanje ili da preuzme trenutno stanje.

Drugi način je da se koristi posrednik u komunikaciji, tako da klijentska aplikacija komunicira sa posvećenim veb serverom i sa skriptom pisanom u PHP skripting jeziku koja iz zaglavlja HTTP zahteva preuzima poslate vrednosti parametara u GET ili POST metodu, a zatim odlučuje koji proces treba da pokrene. Svako pokretanje procesa podrazumeva inicijalizovanje Java aplikacije koja upravlja pinovima, a kojoj se kroz shell argumente proslede potrebni parametri koji ukazuju na to kako treba da upravlja pinovima GPIO porta.

Veb servis osluškuje komunikaciju na standardnom HTTP portu 80 ako se ne koristi šifrovana komunikacija ili na portu 443 ako se koristi šifrovana komunikacija. Nakon uspostavljanja konekcije Apache veb server prosleđuje zahtev sa zaglavljem na obradu. Za obradu i pripremu odgovora je zadužena PHP skripta koja radi u Fast-CGI modu.

PHP skripta je napisana tako da prihvata sledeće parametre:

```
$action = intval($_POST['action']);  
$pin    = intval($_POST['pin']);  
$value  = intval($_POST['value']);  
$message = strval($_POST['message']);
```

Tabela značajnih vrednosti parametara:

\$action	\$pin	\$value	\$message
1	1-17	0-1	-
2	1-17	-	-
3	1-17	-	-
4	1-17	-	-
5	-	-	byte[]
6	-	1-N / 0	-

Na osnovu vrednosti parametra \$action skripta utvrđuje koju proceduru ili funkciju treba da pokrene i to:

\$action	Značenje akcije
1	Postavlja pin pod rednim brojem prosleđenom u parametru \$pin na logičku vrednost prosleđenu u parametru \$value.
2	Postavlja pin pod rednim brojem prosleđenim u parametru \$pin na logičku jedinicu, tj. setuje pin.
3	Postavlja pin pod rednim brojem prosleđenim u parametru \$pin na logičku nulu, tj. clearuje pin.
4	Očitava logičko stanje pina pod rednim brojem prosleđenim u parametru \$pin i vraća tekstualnu reprezentaciju logičke vrednosti klijentskoj aplikaciju u HTTP odgovoru. Logička jedinica je karakter '1', dok je logička nula karakter '0'.
5	Šalje niz bajtova prosleđen u parametru \$message na UART.
6	Preuzima N bajtova sa UART-a i vraća je kroz HTTP odgovor u obliku tekstualnog zapisa u ASCII kodnom šablonu. Broj bajtova za čitanje sa UART-a, tj. vrednost N je prosleđena u parametru \$value. Ako \$value ima vrednost 0, program će iščitavati komunikaciju sa UARTa sve dok ne primi NULL karakter.

7.2. IZVRŠAVANJE NAREDBE KORISNIKA

Postoje dva načina da korisnik uputi naredbu za upravljanje uređajima preko GPIO porta kontrolne jedinice. Jedan način je da korisnik zada govornu komandu koja se prepoznaje i konvertuje u HTTP zahtev, dok je drugi način da direktno, korišćenjem mobilne aplikacije na upravljačkoj jedinici, tj. mobilnom telefonu ili tablet računaru, preko Android aplikacije napisane za upravljanje kontrolnom jedinicom pošalje naredbu.

Naredba koju veb server prihvati i prosledi PHP skripti na obradu se izvršava pozivom unapred kompajliranog Java programa ili drugog programa ili skripte pisane u nekom skripting ili programskom jeziku, npr. Perl, Bash script, Python, C, C++ itd.

Pokretanje programa se vrši izvršavanjem shell komande (shell command) u konzoli putem funkcije exec ili system koja je dostupna u PHP skripting jeziku. Program se pokreće sa potrebnim argumentima koji će mu ukazati na to šta treba da uradi, tj. da li treba da dovede pin GPIP porta na stanje logičke nule ili logičke jedinice, da zameni logičko stanje iz jedinice u nulu ili obrnuto ili da očita stanje i vrati ga kao odgovor u konzoli. Ovaj odgovor će PHP skripta da prihvati i vrati u HTTP odgovoru.

Svako pokretanje programa startuje odvojen proces u memoriji koji se izvršava paralelno, tako da kontrolna jedinica i drugi moduli koji se paralelno izvršavaju na Raspbian operativnom sistemu neće biti zaustavljeni za vreme izvršavanja programa. Na ovaj način je omogućeno, da u slučaju slanja složenog zahteva za upravljanje uređajima, korisnik može da koristi sistem i zadaje druge naredbe, npr. upit o vremenskoj prognozi, zahtev za registrovanje podsetnika itd.

Za potrebe pokretanja java programa napisana je shell skripta gpio.sh sa sledećim kodom:

```
#!/bin/sh
clear
java -jar gpio.jar $1 $2 $3
```

Prikazana skripta uzima tri argumenta i direktno ih prosleđuje pokrenutoj gpio.jar datoteci. Da bi shell skripta gpio.sh bila izvršna datoteka, potrebno joj je dodeliti mod x. Unix naredba za to je:

```
$ chmod +x gpio.sh
```

Primeri shell komandi kakve PHP skripta poziva da bi startovala programe sa argumentima:

```
$ ./home/milantex/master/gpio.sh 1 8 0
```

Postavlja osmi pin GPIO porta na stanje logičke nule.

```
$ ./home/milantex/master/gpio.sh 1 3 1
```

Postavlja treći pin GPIO porta na stanje logičke jedinice.

```
$ ./home/milantex/master/gpio.sh 2 6
```

Postavlja šesti pin GPIO porta na stanje logičke jedinice.

```
$ ./home/milantex/master/gpio.sh 3 2
```

Postavlja drugi pin GPIO porta na stanje logičke nule.

```
$ ./home/milantex/master/gpio.sh 4 9
```

Očitava logičko stanje na devetom pinu GPIO porta i stanje ispisuje na konzolni izlaz.

Npr. stanje je 1:

```
# 1
```

```
$ ./home/milantex/master/gpio.sh 5 "Dobar dan! Ovo ide na UART! Kako ste?"
```

Šalje tekstualnu poruku u ASCII formatu na UART port kontrolne jedinice. Neki uređaj može da bude povezan na UART i da primljenu poruku, npr. ispiše na LCD ekranu.

```
$ ./home/milantex/master/gpio.sh 6 64
```

Očitava 64 bajta (ASCII karaktera) primljenih preko UART porta i ispisuje ih na konzolni izlaz:

```
# Dobar dan i Vama... Vracam preko UARTa! Dobro sam, hvala na pita
```

Poruka je odsečena na 64. karakteru, tj. na 64. primljenom bajtu. To je zato što je kao drugi argument u pozivu programa prosleđen broj 64 što je ograničilo prijemnu poruku na samo 64 karaktera.

Ostali karakteri se čuvaju u baferu (buffer) i u sledećem iščitavanju će poruka biti nastavljena.

```
$ ./home/milantex/master/gpio.sh 6 32
```

Očitava 32 bajta (ASCII karaktera) primljenih preko UART porta i ispisuje ih na konzolni izlaz:

```
# nju! Ovde je sve u redu.×
```

Možete da primetite da je u ovom pozivu vraćena poruka od samo 24 karaktera umesto od 32 koliko je bilo zahtevano. To je zato što je u poruci poslednji karakter bio NULL karakter, tj. karakter sa ASCII vrednošću 0 (nula).

Program gpio.jar je napisan tako da prihvata najviše onoliko karaktera koliko je definisano u drugom argumentu pri pozivu programa, ali će prekinuti prijem kada detektuje NULL karakter. To je razlog zašto ovoga puta umesto odsečene poruke imamo kraću poruku.

Simbol × predstavlja NULL karakter, dok u sledećem primeru, simbol ↵ predstavlja prelazak u novi red. U pitanju je samo New Line feed, a ne i Carriage Return karakter.

```
$ ./home/milantex/master/gpio.sh 6 0
```

Očitava poruku sa UART porta sve dok ne dođe do NULL karaktera, a zatim primljenu poruku ispisuje na konzolni izlaz:

Član 1.↵

Sva ljudska bića rađaju se slobodna i jednaka u dostojanstvu i pravima. Ona su obdarena razumom i svešću i treba jedni prema drugima da postupaju u duhu bratstva.↵

↵

Član 2.↵

Svakom pripadaju sva prava i slobode proglašene u ovoj Deklaraciji bez ikakvih razlika u pogledu rase, boje, pola, jezika, veroispovesti, političkog ili drugog mišljenja, nacionalnog ili društvenog porekla, imovine, rođenja ili drugih okolnosti.↵

↵

Dalje, neće se praviti nikakva razlika na osnovu političkog, pravnog ili međunarodnog statusa zemlje ili teritorije kojoj neko lice pripada, bilo da je ona nezavisna, pod starateljstvom, ne samoupravna, ili da joj je suverenost na ma koji drugi način ograničena.↵

↵

Član 3.↵

Svako ima pravo na život, slobodu i bezbednost ličnosti.↵

↵

Član 4.↵

Niko se ne sme držati u ropstvu ili potčinjenosti: ropstvo i trgovina robljem zabranjeni su u svim oblicima.↵

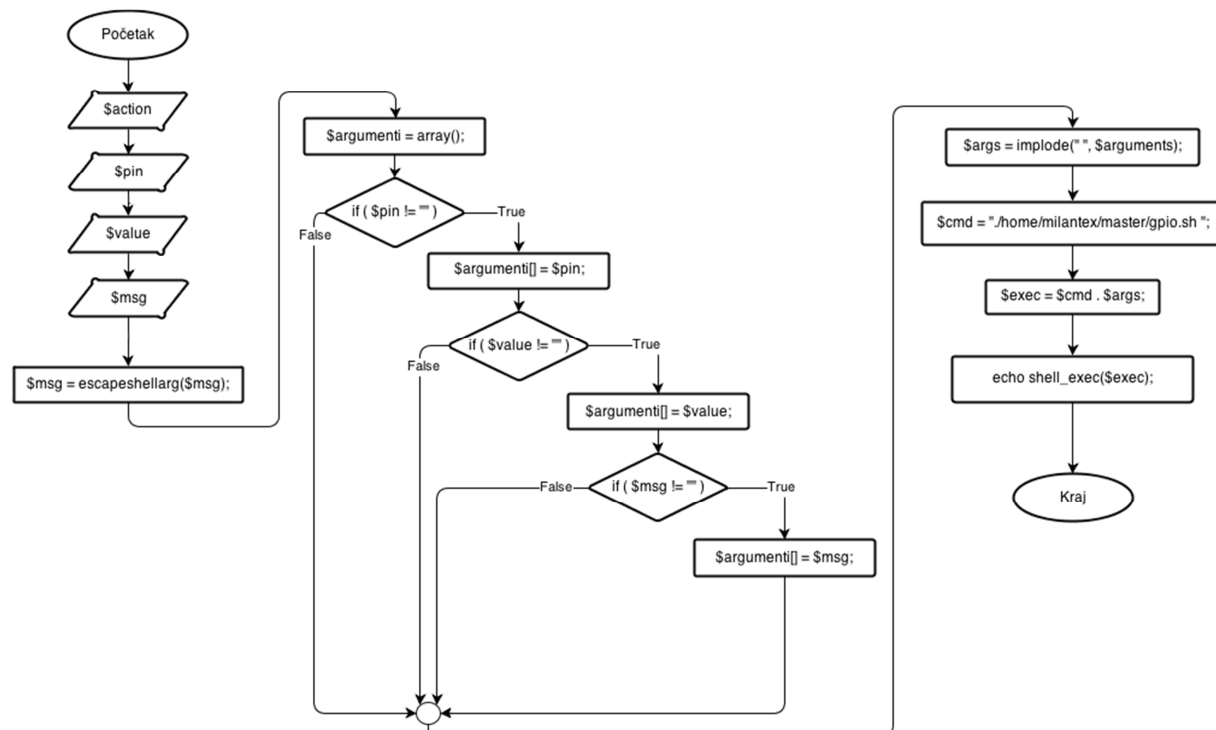
↵

Član 5.↵

Niko ne sme biti podvrgnut mučenju ili svirepom, nečovečnom ili ponižavajućem postupku ili kazni.↵×

7.3. POKRETANJE PROGRAMA IZ PHP SKRIPTE

Kao što je opisano, postoji shell komanda koja pokreće Java program sa prosleđenim argumentima. Veb servis prosleđuje zahtev PHP skripti, koja analizira zahtev i onda pokreće adekvatnu komandu na sledeći način:



Dijagram 2 - Algoritam PHP skripte za izvršavanje zahteva za upravljanje GPIO portom.

Prilikom pokretanja programa, u veb kontrolnom servisu koji startuje program, izvršava se sledeća procedura:

1. Iz zaglavlja HTTP zahteva se preuzimaju značajni parametri:
 - a. action
 - b. pin
 - c. value
 - d. msg
2. Jedini tekstualni parametar msg se obrađuje kako bi se sprečila zloupotreba
3. Kreira se niz argumenata
4. Ukoliko je postavljena vrednost za parametar pin izvršava se sledeći set koraka
 - a. Njegova vrednost se dodaje u niz argumenata
 - b. Ako je postavljena vrednost za parametar value izvršava se sledeći set koraka
 - i. Njegova vrednost se dodaje u niz argumenata
 - ii. Ako je postavljena vrednost za parametar msg izvršava se sledeći set koraka
 1. Njegova vrednost se dodaje u niz argumenata
5. Izvršava se unapred definisana shell skripta sa prosleđenim argumentima dobijenim iz niza
6. Zatvara se HTTP konekcija sa odgovorom shell skripte u telu HTTP odgovora

8. SERVISNI PROGRAMI

Pored programa za upravljanje GPIO i priključenim uređajima u sistemu, postoje servisni programi čija je namena da na osnovu ulazne naredbe dopreme adekvatan odgovor ili da pripreme sistem da u određeno vreme ili u određenim okolnostima reaguje na način koji je definisan. Neki servisni programi mogu vršiti funkciju podsetnika, alarma, sistema za obaveštavanje o važnim događajima koji nisu unapred poznati, a na osnovu interesovanja korisnika su identifikovani kao važni, ali i programi koji omogućavaju dobavljanje specifičnih informacija kao što su definicije termina, prevodi reči ili rečenica sa raznih jezika korišćenjem besplatnih on-line servisa za prevođenje itd.

Ovi programi se u sistem integrišu u vidu standardizovanih modula. Svaki modul sistema je PHP klasa koja je nasleđena iz osnovne klase Module (modul).

8.1. KLASA MODULE

Klasa Module u svom konstruktoru inicijalizuje sve potrebne resurse koje modul ima na raspolaganju, kao što su aktivna konekcija na zajedničku bazu podataka u kojoj modul može da čuva podatke potrebne za rad, ali i resurse koji obezbeđuju rad sa spoljašnjim Veb servisima, metode za obradu XML datoteka, tumačenje tekstualno zapisanih govornih naredbi sistemu itd.

8.2. POJEDINAČNI MODULI

Pojedinačni moduli su klase izvedene iz osnovne klase Module, a za svaku je potrebno definisati javne metode i po potrebi, rečnik pojmova i katalog regularnih izraza koji uparuju naredbe koje je modul spreman da izvrši.

Svaki modul može da bude praćen definicijom dodatnih klasa koje koristi osnovna klasa modula, a svaka od dodatnih klasa može da bude kompletno definisana kao nezavisna klasa ili može da bude nasleđena iz jedne od dostupnih osnovnih klasa za nasleđivanje ili za izvlačenje interfejsa ka njima.

Primer jednog izvedenog modula čija je funkcija beleženje zahteva za oglašavanje alarma, sa listingom značajnih metoda:

```
class AlarmModule extends Module {
    public function setAlarm($dateTime, $message) {
        return $this->Storage->addToArray(
            "alarm_entries", new AlarmModuleEntry($dateTime, $message)
        );
    }
    public function getAlarms() {
        return $this->Storage->getArrayItems("alarm_entries");
    }
    public function getAlarmForNow() {
        $now = date("Y-m-d H:i");
        return $this->Storage->getFromArrayWhereEntryFieldEquals(
            "alarm_entries",
            "dateTime",
            $now
        );
    }
}
```

```

public function soundAnAlarmForNow() {
    $alarm = $this->getAlarmForNow();
    if ( $alarm != null ) {
        $this->Service->Sound->NoticeSound();
        $this->Service->TTS->Speak( $alarm->message );
        $this->Storage->removeFromArrayWhereEntityFieldEquals("alarm_entries", "id", $alarm->id);
    }
}

}

class AlarmModuleEntry extends Entry {
    public $dateTime;
    public $message;

    public AlarmModuleEntry($dateTime, $message) {
        parent::Entry();
        $this->dateTime = substr($dateTime, 0, 16);
        $this->message = $message;
    }
}

```

8.3. INSTALACIJA MODULA

Da bi modul mogao da postane sastavni deo sistema za simulaciju inteligencije u automatizovanom domu, potrebno je da se instalira.

Postupak instalacije modula je sledeći:

1. Na Veb interfejsu za upravljanje sistemom, otvara se sekcija za instalaciju modula;
2. Klikne se na dugme [Browse] da se na lokalnom računaru locira ZIP arhiva sa modulom;
3. Klikne se na dugme [Upload and Install] koji pokreće proces prenošenja arhive sa modulom;
4. Nakon što je modul uspešno prenet na server i raspakovan u adekvatan direktorijum, potrebno je izvršiti registrovanje i aktivaciju modula.

8.4. REGISTROVANJE MODULA

Da bi modul mogao da se koristi u sistemu, potrebno ga je prvobitno registrovati. Registrovanje modula se vrši nakon njegove instalacije, a pre aktivacije.

Modul može da bude registrovan samo ako prođe sve provere ispravnosti. Provera ispravnosti modula se vrši tako što se za svaki instalirani modul pokreće simuliran proces izvršavanja koji instancira objekat klase modula, koja mora da nasleđuje klasu Module. Zatim se izvršavaju dijagnostički metodi koji proveravaju ispravnost modula i utvrđuju da li se struktura objekata instanciranog iz klase novog modula poklapa sa strukturom deklarisanom u manifestu modula. Ako se struktura ne poklapa, sistem ne dozvoljava registraciju modula u katalog sistema.

Ako se provera strukture završi uspešno, u katalog raspoloživih modula u sistemu se upisuje i novi modul, tako da može da se koristi kada ga korisnik aktivira i sistem pozove.

8.5. AKTIVACIJA MODULA

Svaki modul u sistemu, koji je instaliran i registrovan, mora da se aktivira kako bi mogao da se koristi. U katalogu modula u sistemu se za određeni modul, koji se aktivira ili deaktivira, postavlja određeno polje na vrednost logičke jedinice ili nule i na taj način se menja status modula.

Samo aktivni moduli iz kataloga dobijaju obaveštenja od sistema da treba da se obradi određena naredba i samo aktivnim modulima se upućuje na proveru uzorak naredbe da utvrde da li su sposobni da je protumače i izvrše.

8.6. METOD ZA UTVRĐIVANJE SPREMNOSTI MODULA ZA IZVRŠAVANJE NAREDBE

Svaki modul ima statički registar regularnih izraza koji uparjuju tekst naredbe. To znači da svaki modul koji može da izvrši neku naredbu mora da bude spreman da je protumači kada je zadata govornim jezikom i normalizovana pre slanja na proveru, a da bi mogao da utvrdi da li je spreman da je izvrši, potrebo je da je prvobitno uspešno upari sa regularnim izrazom za koji je programiran.

Za proveru je zadužen sistemski metod za utvrđivanje logičke istinitosti:

```
/* Metod za utvrđivanje spremnosti modula za izvršavanje naredbe
 * @arg1 modul_name String
 * @arg2 naredba String
 * @return spremnost Boolean
 */

function canModuleHandleTheCommand( $module_name, $command );
```

Sistem poziva ovaj metod za svaku prihvaćenu naredbu i iterativno iz registra preuzima naziv svakog registrovanog modula koji ima status da je aktivan i prosleđuje naziv metodu.

Metod preuzima registar regularnih izraza iz klase modula i proverava za svaki regularni izraz da li postoji poklapanje. Metod nikada ne instancira objekat klase modula koji u trenutku proverava, jer je registar regularnih izraza statičan niz koji je uvek dostupan operatorom Scope Resolution Operator nad imenom klase.

8.7. REGISTAR REGULARNIH IZRAZA MODULA

Kao što je već opisano, svaki modul koji može da tumači govorne komande mora da ima registar regularnih izraza kojima uparuje elemente iz naredbe. Ovaj registar se u novoj klasi koja nasleđuje Module klasu definiše kao statični neinteksirani niz nizova karaktera, tj. stringova.

```
class AlarmModule extends Module {
    public static $registarRegularnihIzraza = array(
        "Prvi regularni izrad",
        "Drugi regularni izraz",
        "Treci regularni izraz"
    );
}
```

Može da se desi da postoji više modula koji u svojim registrima imaju identične regularne izraze. Ako se to desi, sistem će korisnika prilikom registrovanja modula o tome obavestiti i pitati ga da li želi da izabere koji od dva modula će za konkretan regularni izraz imati prioritet.

Svaki regularni izraz definisan u registru regularnih izraza modula na odvojenoj lokaciji u manifestu modula ima rečima napisan opis čemu složi i šta je rezultat izvršavanja naredbe uparene kroz konkretan izraz. Na osnovu toga se korisniku ispisuju dodatna uputstva i pomoć prilikom odlučivanja kojem izrazu, tj. kojem modulu želi da dâ prednost.

Ako se desi da prilikom izvršavanja neke naredbe sistem za jednu naredbu utvrdi da postoji više od jednog modula koji ima regularni izraz sposoban da izvrši zadatu naredbu, sistem beleži naredbu i regularne izraze u modulima koji su uspeali da je upare i omogućava korisniku da kroz administracioni panel za konkretan slučaj definiše izuzetak.

Po ugledu na prvobitno opisan slučaj kada postoje dva identična regularna izraza, korisnik biva obavešten o načinu rada jednog i drugog ili više od dva modula i izraza koji su uparili konkretnu zabeleženu naredbu i nakon toga može da odluči kojem modulu za definisani slučaj želi ubuduće da dâ prednost u izvršavanju naredbe.

Svaki definisani izuzetak kojim korisnik daje prednosti izrazu ili modulu za konkretan slučaj može u bilo kojem trenutku da se promeni kroz administracioni panel.

8.8. PRIPREMA ODGOVORA MODULA

Nakon utvrđivanja adekvatnog modula na osnovu uparivanja komande kroz registar regularnih izraza, sistemski proces instancira objekat klase utvrđenog modula i pokreće inicijalni metod `start()` koji kao argument uzima niz karaktera koji predstavlja tekstualno zapisanu govornu naredbu, prethodno normalizovanu od strane sistemskog modula za normalizaciju naredbe.

Metod `start()` utvrđuje koji regularni izraz iz njegovog registra regularnih izraza je uspešno upario naredbu i na osnovu toga odlučuje kojem privatnom metodu da delegira izvršavanje naredbe.

Delegirani metod mora da bude spreman da uspešno analizira naredbu, utvrdi šta treba da obavi i pokrene proceduru pripremanja odgovora.

Pripremanje odgovora može da bude preuzimanje tekstualnog sadržaja sa Internet baziranog servisa, a na osnovu ključne reči ili fraze sadržane u naredbi, ali i informacija dobijena iz drugog izvora. Nakon što je predmet odgovora pribavljen sa adekvatnog izvora, tj. senzora, Internet servisa itd. metod kojem je izvršavanje delegirano treba da prilagodi odgovor za iščitavanje korisniku.

8.9. FORMATIRANJE ODGOVORA MODULA

Delegirani metod modula kojem je prepušteno da pripremi odgovor korisniku na zadatu naredbu ima obavezu da prilagodi povratnu informaciju u vidu odgovora koji bi bio saopšten korisniku u govornoj formi, tj. da bude prečišćen od nepotrebnih karaktera koji ne mogu da se čitaju, a da oni koji su značajni budu zamenjeni adekvatnim izrazom, rečju ili opisom koji ukazuje na njihovo postojanje u odgovoru, ako sa sobom nose neku informaciju od značaja za korisnika.

Na primer, ako je naredba korisnika glasila:

```
Are there any messages for Milan?
```

Modul kojem je delegirano izvršavanje naredbe je:

```
MessageModule
```

Informacije do kojih je `start()` metod modula došao je da postoje poruke u bazi za korisnika sa imenom Milan i te poruke su:

- Post a blog entry today.
Watch the new Game of Thrones episode.

Samo iščitavanje ovih poruka ne bi bio u potpunosti prirodan odgovor, već je potrebno dodatno obraditi i proširiti isti kako bi delovao prirodnije.

Primer proširenog odgovora:

```
There are two messages for you.  
The first is [pause] Post a blog entry today.  
The second is [pause] Watch the new Game of Thrones episode.  
Those are all messages I have for you.
```

Naknadna obrada ove proširene poruke nije potrebna, jer sistemski modul za normalizaciju odgovora može naknadno da uredi odgovor i ukloni eventualne nedostatke, ali i da prilagodi poruku za iščitavanje.

Primer naknadnog prilagođavanja koje obavlja sistemski modul za normalizaciju je zamena pomoćne naredbe `[pause]` sa pauzom u trajanju od jedne sekunde tokom iščitavanja odgovora korisniku.

8.10. HIJERARHIJA KLASA

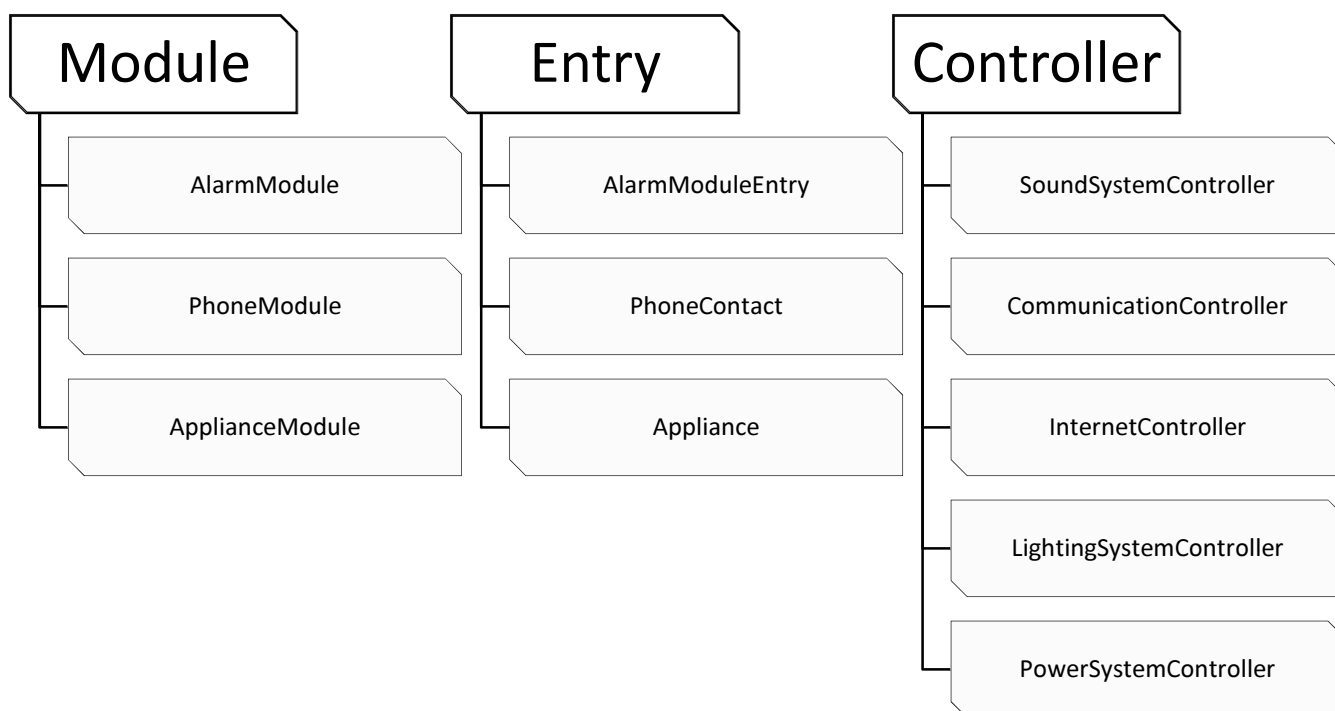
Klasna hijerarhija se ogleda u postojanju tri osnovne klase iz koje su izvedene sve ostale klase sistema, klase modula, klase dodataka za upravljanje specijalizovanim modulima itd.

Osnovna klasa u sistemu za simulaciju inteligencije u automatizovanom domu je klasa **Module**. Karakteristike klase Module su te da je ona apstraktna klasa i da bi se instancirao objekat nekog modula, mora prvo da se izvede nova klasa koja nasleđuje Module.

Druga klasa je klasa **Entry**. Entry je takođe apstraktna klasa koja se koristi kao medijator između dva ili više objekta klase izvedenih iz klase Module, a za potrebe prenosa složenih struktura podataka. Prilikom izvođenja nove klase iz klase Entry, neophodno je dodati članove podatke koji se prenose u objektu instanciranom od nove izvedene i proširene klase.

Treća klasa je klasa **Controller**. Klasa kontroler je takođe apstraktna klasa sa nekoliko virtualnih metoda koje je obavezno definisati u novim izvedenim klasama. U pitanju su komandni metodi koje sistem poziva prilikom registracije i aktivacije kontrolera, kao i metod za izvršavanje delegirane funkcije nekog modula. Definisani metodi koji kao argumente uzimaju objekte klase izvedenih od klase Module moraju za argument da navedu konkretan naziv, tj. tip izvedene klase koju su sposobni da izvrše.

Iz ova tri osnovna tipa podatka se izvede tri nezavisna stabla klasa, kao što je ilustrovano u sledećem primeru:



9. KREIRANJE PROFILA KORISNIKA

Sistem za simulaciju inteligencije u automatizovanom domu prati navike korisnika. Navike se prate na dnevnom, nedeljnom i mesečnom nivou. Za pojedine module i na duži vremenski period.

Pored navika pri korišćenju funkcija modula, sistem može da prati i kretanje korisnika van doma na dnevnom nivou u cilju utvrđivanja radnog vremena i uobičajenog vremena kada se korisnik kreće van doma i kada se vraća. Razlog zbog kojeg se pravi profil kretanja je da bi sistem mogao da pripremi dom za dolazak korisnika.

Jedna od primena kreiranog profila je utvrđivanje trenutka kada će korisnik d se vrati, kako bi sistem mogao na vreme da uključi bojler za grejanje vode, tako da se do povratka korisnika ona dovoljno zagreje da korisnik može da se istušira posle posla i slično.

9.1. KREIRANJE PROFILA NA OSNOVU EVIDENCIJE DOGAĐAJA

Prvi vid profila korisnika je u pogledu njegovih navika u samom domu, tj. o načinu na koji koristi određene uređaje, kao što je stanje i intenzitet osvetljenja u prostorijama, jačina zvuka pri slušanju muzike, omiljeni TV program u određeno vreme u toku dana, provera vremenske prognoze i druge funkcije koje sistem za automatizaciju može da nudi korisniku kao servis ili funkcija upravljanja. Profil se kreira u dužem vremenskom periodu. Kreiranje profila se vrši tako što se beleži korišćenje uređaja, servisa i funkcija sistema svakog dana.

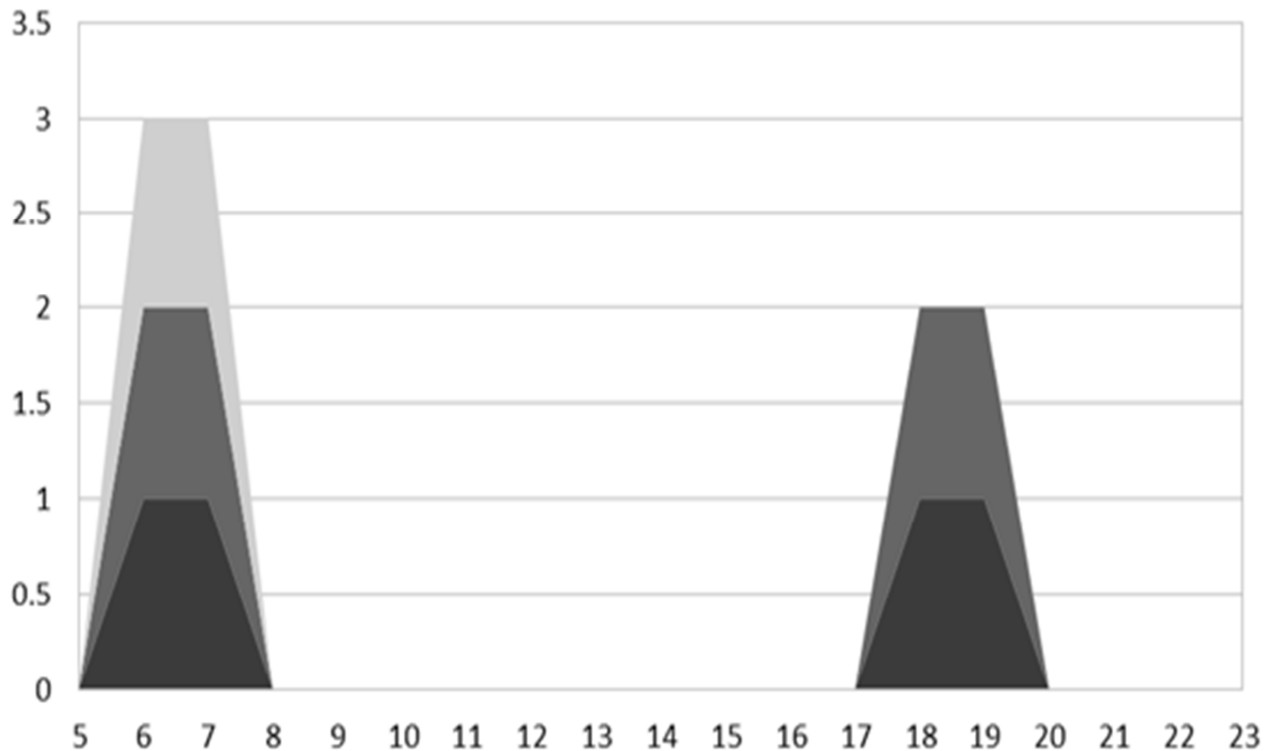
Primer:

Korisnik koji se prati koristi sistem na sledeći način u periodu od tri dana uzastopno:

Vreme	Dan 1	Dan 2	Dan 3	Profil
05 - 06				100% Uključi bojler (30 min pre)
06 - 07	Uključi bojler	Uključi bojler	Uključi bojler	100% Isključi bojler (30 min pre)
07 - 08	Isključi bojler Uključi TV	Isključi bojler Uključi TV	Isključi bojler Uključi muziku Isključi muziku	66% Uključi TV 33% Uključi muziku 33% Isključi muziku (prosečno trajanje)
08 - 09	Isključi TV Ugasi svetla	Isključi TV Ugasi svetla	Ugasi svetla	66% Isključi TV (prosečno trajanje) 100% Ugasi svetla
09 - 10				
...				
18 - 19				66% Uključi bojler (30 min pre)
19 - 20	Uključi bojler	Uključi bojler		66% Isključi bojler (30 min pre)
20 - 21	Isključi bojler Uključi TV	Isključi bojler		33% Uključi TV
21 - 22				
22 - 23	Isključi TV			33% Isključi TV

Ovaj vid kreiranja profila je pogodan za aktivnosti koje se ne ponavljaju svakodnevno, tj. za one koji se dese jednom nedeljno ili jednom u dužem periodu. Npr. slušanje muzike.

Drugi vid kreiranje profila je na nedeljnom nivou, za svaki dan u nedelji. Za ovaj vid kreiranja profila korišćenja sistema je karakteristično da se verovatnoća vršenja neke aktivnosti meri nedeljno.



Slika 3 - Grafik profila upravljanja radom uređaja na osnovu dnevnog praćenja

Iz grafika se vidi da je ovaj vid kreiranja profila u nekim situacijama optimalan, jer profil jasno oslikava način na koji korisnik koristi uređaj.

Problem bi nastao tek kada bi se kreiranje profila vršilo na ovaj način u periodu dužem od sedam dana, tako da uključi vikend, tj. neradne dane kada je način korišćenja drugačiji.

9.2. KREIRANJE PROFILA NEDELJNIM PRAĆENJEM PONAŠANJA

U slučaju kada korišćenje uređaja nije svaki dan identično, koristi se metod za kreiranje profila praćenjem ponašanja korisnika u dužem vremenskom periodu.

Primer:

Korisnik koji se prati koristi sistem na sledeći način u periodu od mesec dana:

Dan u mesecu	Uređaj A - Svetlo	Uređaj B - Računar	Analiza zavisnosti za par A -> B
1	9:15 Uključen	9:19 Uključen	Uključen A -> 4 min. -> Uključi B
2	7:15 Uključen	7:16 Uključen	Uključen A -> 1 min. -> Uključi B
3	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
4	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
5	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
6	11:13 Uključen	12:20 Uključen	Uključen A -> 67 min. -> Uključen B
7	12:00 Uključen	11:50 Uključen	Nije ispunjeno A -> B
8	9:15 Uključen	9:16 Uključen	Uključen A -> 1 min. -> Uključi B
9	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
10	7:15 Uključen	7:16 Uključen	Uključen A -> 1 min. -> Uključi B
11	7:15 Uključen	7:16 Uključen	Uključen A -> 1 min. -> Uključi B
12	7:15 Uključen	7:17 Uključen	Uključen A -> 2 min. -> Uključi B
13	13:12 Uključen	10:47 Uključen	Nije ispunjeno A -> B
14	12:00 Uključen	17:12 Uključen	Uključen A -> 312 min. -> Uključen B
15	9:15 Uključen	9:15 Uključen	Uključen A -> 0 min. -> Uključi B
16	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
17	7:15 Uključen	7:16 Uključen	Uključen A -> 1 min. -> Uključi B
18	7:15 Uključen	7:20 Uključen	Uključen A -> 5 min. -> Uključi B
19	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
20	06:30 Uključen		Nije ispunjeno A -> B
21		18:00 Uključen	Nije ispunjeno A -> B
22	9:15 Uključen	9:20 Uključen	Uključen A -> 5 min. -> Uključi B
23	7:15 Uključen	7:16 Uključen	Uključen A -> 1 min. -> Uključi B
24	7:15 Uključen	7:20 Uključen	Uključen A -> 5 min. -> Uključi B
25	7:15 Uključen	7:17 Uključen	Uključen A -> 2 min. -> Uključi B

26	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B
27	11:12 Uključen	11:14 Uključen	Uključen A -> 2 min. -> Uključen B
28	13:03 Uključen	12:46 Uključen	Nije ispunjeno A -> B
29	9:15 Uključen	9:15 Uključen	Uključen A -> 0 min. -> Uključi B
30	7:15 Uključen	7:15 Uključen	Uključen A -> 0 min. -> Uključi B

Iz date tabele sa prikupljenim podacima o načinu korišćenja uređaja A i uređaja B, tj. osvetljenja u spavaćoj sobi i računara koji se nalazi u istoj sobi, dobijamo sledeću tabelu na nedeljnom nivou.

Uređeni podaci po danima u nedelji analiziraju se na osnovu učestalosti ponavljanja slučaja iz nedelje u nedelju, pri čemu se izuzeci i velika odstupanja eliminišu. Tabela u međukoraku izgleda ovako:

Dan	Nedelja 1	Nedelja 2	Nedelja 3	Nedelja 4	Nedelja 5	Prosek između događaja
1	4 min.	1 min.	0 min.	5 min.	0 min.	2,00 min.
2	1 min.	0 min.	0 min.	1 min.	0 min.	0,40 min.
3	0 min.	1 min.	1 min.	5 min.		1,75 min.
4	0 min.	1 min.	5 min.	2 min.		2,00 min.
5	0 min.	2 min.	0 min.	0 min.		0,50 min.
6	67 min.	-	-	2 min.		-
7	-	312 min.	-	-		-

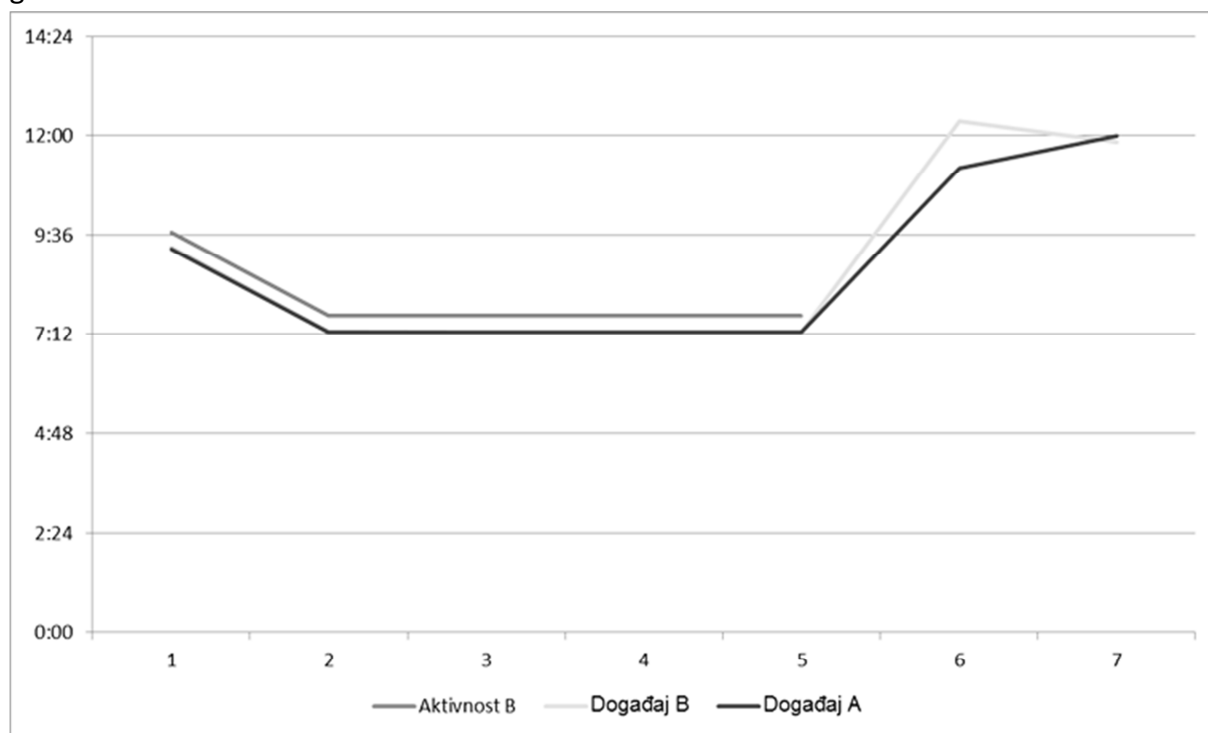
Analiza uključuje određivanje prosečne vrednosti čekanja između događaja A i događaja B, tj. između promene stanja uređaja A i uređaja B, uz obavezno otklanjanje grešaka i izuzetaka, tj. gde je odstupanje veće od 10%.

Nedelja	Događaj	Prag za aktivnost	Aktivnost
1	Uključen A	2,00 min.	Uključi B
2	Uključen A	0,40 min.	Uključi B
3	Uključen A	1,75 min.	Uključi B
4	Uključen A	2,00 min.	Uključi B
5	Uključen A	0,50 min.	Uključi B
6	Uključen A	-	
7	Uključen A	-	

Nakon što je nedeljni profil formiran, potrebno je utvrditi minimalno vreme koje je proteklo od događaja A do aktivnosti B, tj. od uključivanja uređaja A do uključivanja uređaja B. Nedeljni profil je potrebno ažurirati tako da se svuda podesi na iznos minimalnog vremena.

Nedelja	Događaj	Prag za aktivnost	Aktivnost
1	Uključen A	0,40 min. = 24 sek.	Uključi B
2	Uključen A	0,40 min. = 24 sek.	Uključi B
3	Uključen A	0,40 min. = 24 sek.	Uključi B
4	Uključen A	0,40 min. = 24 sek.	Uključi B
5	Uključen A	0,40 min. = 24 sek.	Uključi B
6	Uključen A	-	
7	Uključen A	-	

Ovako kreiran profil može da se testira primerom korišćenja ilustrovanim sledećim grafikonom:



Slika 4 - Testiranje kreiranog profila na snimljenom uzorku

Grafik je generisan na osnovu realnih podataka iz tabele koja predstavlja samo prvu nedelju u mesecu tokom kojeg je vršena analiza i kreiran profil. Tabela sadrži i informaciju o vremenu posle kojeg sistem automatski sprovodi aktivnost B od kako se evidentira događaj A. Događaj B je originalni događaj koji je korisnik ručno aktivirao neko vreme nakon događaja A u tabeli i u grafiku je prikazan samo u svrhu ilustracije.

Dan	Događaj A	Događaj B	Prag	Aktivnost B
1	9:15	9:19	0:24	9:39
2	7:15	7:16	0:24	7:39
3	7:15	7:15	0:24	7:39
4	7:15	7:15	0:24	7:39
5	7:15	7:15	0:24	7:39
6	11:13	12:20		
7	12:00	11:50		

Slučaj 1

Korisnik prvog dana u nedelji u 09:16:11 uključi uređaj A. Uključivanje uređaja A izaziva okidanje Događaja A. Sistem analizira spisak svih aktivnosti koje treba sprovesti kada se dogodi Događaj A. Jedna od aktivnosti je Aktivnost B koja za prvi dan u nedelji ima prag od 24 sekunde. Sistem će automatski izvršiti Aktivnost B 24 sekunde nakon Događaja A, tj. u 09:16:35.

Slučaj 2

Korisnik sedmog dana u nedelji u 13:11:02 uključi uređaj A. Uključivanje uređaja A izaziva okidanje Događaja A. Sistem analizira spisak aktivnosti koje treba sprovesti kada se dogodi Događaj A. Pošto za sedmi dan ne postoji u registru aktivnosti ni jedan zapis, tj. ne postoji informacija o aktivnosti koja treba da se izvrši, sistem ne reaguje na Događaj A.

Iz tabele mogu da se zaključe preostalih nekoliko slučajeva korišćenja.

10. AKTIVNO AŽURIRANJE PRIORITETA PROFILA KORISNIKA

Kako se profili kreiraju i osvežavaju neprekidno, njihov prioritet se menja. Svaki profil ima atribut „prioritet“ koji je zabeležen u obliku realnog broja, a u bazi podataka kao tip Decimal (10,9). Za prioritet se koristi samo desetina pozitivne strane mogućeg opsega za definisano polje u bazi i to brojevi od 0,000000000 do 1,000000000.

Vrednost prioriteta kreiranog profila korisnika koja iznosi 0,000000000 znači da je u pitanju profil koji sistem ne treba nikada da koristi, dok vrednost 1,000000000 znači da sistem uvek za konkretan uređaj treba da se upravlja prema slučajevima korišćenja iz ovog profila.

U realnom sistemu se skoro nikada ne javljaju ove granične vrednosti prioriteta profila, već se prioritet neprekidno menja i kreće se u pomenutom opsegu.

Funkcija za izračunavanje nove vrednosti prioriteta profila uzima u obzir nekoliko parametara:

1. Profil, označeno sa P
2. Datum i vreme, označeno sa DT
3. ID broj uređaja čije se stanje promenilo, označeno sa ID
4. Novo stanje u koje je uređaj doveden, označeno sa S

Na osnovu ulaznih parametara se iz sistema dopremaju dodatni zavisni parametri:

1. Niz slučajeva u profilu, označeno sa θ , gde $\theta \in P$
2. Broj slučajeva u nizu θ , označeno sa $|\theta|$, gde je $|\theta| > 0$
3. Dan u nedelji, na osnovu datuma i vremena, označava se sa δ
4. Sat u danu, na osnovu datuma i vremena, označava se sa H
5. Minut u danu, na osnovu datuma i vremena, označava se sa M
6. Niz slučajeva profila koji su najbliži satu H i minutu M u toku dana δ u profilu P, označavaju se sa Θ , gde je:

$$\Theta = \{ \tau \mid \tau \in \theta, \Delta h \rightarrow 0 \wedge \Delta m \rightarrow 0 \wedge \text{dan}(\tau) = \delta \wedge S(\tau) = S \},$$

Pri čemu su:

- τ slučaj;
- $\Delta h = H - H(\tau)$, je razlika sata u danu i sata za slučaj τ ;
- $\Delta m = M - M(\tau)$, je razlika minuta u danu i minuta za slučaj τ ;
- $\text{dan}(\tau)$ je dan u nedelji za slučaj τ ;
- Novnastalo stanje slučaja je isto kao novonastalo stanje u događaju koji je okinuo ažuriranje vrednosti prioriteta profila.

7. Slučaj koji je najbliži satu i minuti promene stanja, označava se sa τ' ,
gde je:

$$\tau' = \{ \tau \mid \tau \in \Theta, \min(\Delta h) \wedge \min(\Delta m) \};$$

Element niza Θ čija su odstupanja sata i minuta u odnosu na sat i minut događaja najmanja.

Kada je utvrđen najbliži slučaj u postojećem profilu, potrebno je utvrditi odstupanje karakteristika tog slučaja u odnosu na osobine događaja koji je okinuo ažuriranje vrednosti prioriteta profila.

Prioritet Π se računa na sledeći način:

$$\Pi = \Pi + \frac{\mu - s}{\mu^2}$$

Gde je $s = 60 \cdot \Delta h + \Delta m$

Vrednost μ je najduže trajanje između starog i novog stanja u slučajevima skupa Θ .

Korekcija vrednosti prioriteta Π za profil u kojem je pronađen odgovarajući slučaj τ' je reda veličine nekoliko promila i može da se kreće u plus ili u minus.

Nakon izračunavanja novog prioriteta, vrši se njegovo ograničavanje na sledeći način:

$$\Pi = \begin{cases} 1, & \Pi > 1 \\ 0, & \Pi < 0 \end{cases}$$

Konačna dobijena vrednost prioriteta Π se kreće u definisanim granicama opsega od 0 do 1, s tim da se u bazi podataka prilikom čuvanja vrednosti vrši neizbežno odsecanje svih decimala iza desete decimale. Ovaj nivo preciznosti je izuzetno zadovoljavajući za prosečan sistem i ne očekuje se loše sortiranje profila pri odlučivanju koji treba da bude aktivan.

11. ANALIZA GOVORNIH KOMANDI

Sistem za simulaciju inteligencije u automatizovanom domu za analizu govornih komandi koristi regularne izraze. Pomoću njih se tumače naredbe zadate govornim jezikom.

Sam sistem čine moduli za konverziju govora u tekst, analizu teksta, transformaciju govorne komande u niz karaktera pogodan za uparivanje regularnim izrazima, kao i za samu konverziju koja se vrši posredstvom programa za automatsko pretvaranje govora u tekst koji je u sistem uveden kao black-box modul.

12. REGULARNI IZRAZI

Regularni izraz (regular expression) je niz karaktera sa tačno određenom sintaksom koja omogućava sparivanje delova teksta, izdvajanje pojedinih delova teksta na osnovu definisanog pravila ili šablona, kao i kompleksne izraze za zamenu pojedinih delova teksta na osnovu izraza kojim su delovi koji se menjaju upareni.

Regularni izrazi se primenjuju u raznim aplikacijama i rešenjima koja rade sa tekstovima, kao što su editori izvornog koda programa, gde se regularni izrazi koriste za identifikaciju pojedinih delova koda za potrebe promene stila kojim su prikazani u editoru. Koriste se za analizu teksta na Internetu od strane botova, tj. programa koji analiziraju sadržaj i kontekst, kako bi identifikovali ključne reči za pretragu ili prikaz reklamnih poruka.

12.1. POREKLO REGULARNIH IZRAZA

Osnovni razlozi zbog kojih su nastali regularni izrazi su i dalje najdominantniji razlozi zašto se oni razvijaju i u koje svrhe se koriste, a to je da u editorima teksta identifikuju određene šablone kako bi se oni izdvojili i stilizovali.

U početku su editori teksta radili tako što su opcije stilizovanog prikaza teksta na osnovu sadržaja bili pisane u vidu pod-programa koji su se kretali kroz ceo tekst i tražili određene ključne reči ili markere, a koje su u prolazu menjali drugim kodom i na taj način omogućavali promenu prikaza koda.

Kasnije su tekstualni editori počeli da koriste relativne šablone za identifikaciju delova teksta, a vremenom se uz podršku teorije automata i teorije formalnih jezika formirao formalni jezik sa sintaksom koja je dotadašnje regularne skupove zamenila regularnim izrazima.

Prvi programi koji su koristili ovaj novi vid nalaženja šablona bili su programi na UNIX operativnim sistemima. Notacija je postala skoro standardna za većinu editora teksta, poput ed i grep editora, a sa malim izmenama se proširila na veliku lepezu programa.

U početku kao biblioteka, a kasnije kao sastavni deo, regularni izrazi se integrišu u mnoge programske i skripting jezike i na taj način dodatno olakšavaju pisanje programa koji na mnogo efikasniji način, pisanjem relativno jednostavnih izraza, mogu da omogućе izdvajanje delova teksta za potrebe sortiranja, analize, postavljanje markera, nalaženje i izvlačenje podataka određenih tipova ili formata itd.

12.2. SINTAKSA REGULARNIH IZRAZA

Postoji više tipova regularnih izraza. Programski jezik Perl ima svoj tip koji je specifičan na određeni način, dok većina programskih jezika, operativnih sistema i programa koristi POSIX standard koji je kao standard prihvaćen od strane IEEE. Akronim je od Portable Operating System Interface, gde se slovo X odnosi na UNIX.

Sintaksa regularnih izraza na unix-olikim operativnim sistemima je danas zastarela po POSIX definicijama, ali se i dalje koristi zbog kompatibilnosti unazad.

Kod tumačenja regularnih izraza u jednom uzorku izraza sa većinom karaktera se postupa kao sa literalom, tj. sparuju se samo sa sobom. Na primer, kada se u izrazu nađe slovo n, onda će svako slovo n, bilo gde u tekstu koji se analizira biti izdvojeno u rezultatu.

13. ZNAČENJE SPECIJALNIH KARAKTERA U REGULARNIM IZRAZIMA

Određeni karakteri u regularnim izrazima imaju posebno značenje u izdvajanju i sparivanju delova teksta.

Primeri koje ću navoditi u ovom poglavlju će biti podeljeni u tri dela:

- Tekst - Predstavlja tekst nad kojim se sprovodi izdvajanje regularnim izrazom.
- Izraz - Predstavlja izraz koji se primenjuje nad tekstom za izdvajanje delova.
- Izdvojeno - Predstavlja tekst nad kojim je primenjen izraz. Izdvojeni delovi su podvučeni.

13.1. KARAKTER TAČKA .

Tačka . u izdvajanju karaktera sparuje bilo koji karakter samo jednom, tj. izdvaja jedan jedini karakter, bez obzira koji je.

Tekst: Rekao je da je bilo belo.

Izraz: b.lo

Izdvojeno: Rekao je da je bilo belo.

13.2. KARAKTERI NABROJANI UNUTAR UGLASTIH ZAGRADA []

Karakter koji se nađu između uglastih zagrada [] izdvajaju iz teksta jedan od karaktera koji je nabrojan u njima.

Tekst: Jednu sodu, jednu vodu i jednu rodu.

Izraz: [sv]odu

Izdvojeno: Jednu sodu, jednu vodu i jednu rodu.

13.3. OPSEG KARAKTERA U UGLASTIM ZAGRADAMA [i]

Kada se unutar uglastih zagrada nađe opseg karaktera, definisan kao prvi karakter (niže kodne vrednosti), praćen karakterom - (minus) i zatim poslednji karakterom opsega (više kodne vrednosti), to označava opseg od prvog do poslednjeg karaktera u tom opsegu, po kodnoj vrednosti karaktera.

Tekst: Samo heksadecimalno zapisani brojevi: 0E AE 44 5R E4 8U AG 32 12
Izraz: [0-9A-F]{2}
Izdvojeno: Samo heksadecimalno zapisani brojevi: 0E AE 44 5R E4 8U AG 32 12

Regularni izraz: "[0-9A-F]{2}" će izdvojiti sve delove teksta, gde se nalaze dva karaktera iz dva naznačena opsega. Prvi opseg su karakteri od 0-9, tj. cifre, a drugi opseg su karakteri od A-F, tj. dodatni karakteri koji predstavljaju cifre u heksadecimalnom zapisu. Bilo koja dva karaktera iz tog opsega će biti izdvojena.

Jedini slučaj kada se karakter - (minus) smatra doslovno, tj. kao karakter -, a ne kao specijalni karakter za definisanje opsega, je kada se unutar uglastih zagrada nalazi na prvom ili poslednjem mestu.

Tekst: Moj broj telefona je 2416-804.
Izraz: [0-9-]*
Izdvojeno: Moj broj telefona je 2416-804.

13.4. BROJ PONAVLJANJA ŠABLONA

Kada hoćemo da se određeni deo izraza odnosi na više uzastopnih delova, tj. da imamo šablon koji se ponavlja, koristimo vitičaste zagrade { i } između kojih postavljamo uslov koji određuje koliko puta navedeni šablom sme da se ponovi. Unutar vitičaste zagrade možemo da navedemo jedan ili dva argumenta. Način korišćenja sa jednim argumentom je predstavljen u prvom primeru za opseg karaktera u uglastim zgradama.

Tekst: Telefon: 2-416-804, 2416-804, 24-16-804, 241-608-4, 2416804.
Izraz: ([0-9-]{1,3}){2,3}
Izdvojeno: Telefon: 2-416-804, 2416-804, 24-16-804, 241-608-4, 2416804.

Prvi izraz koji se sparuje je uokviren kao celina unutar zagrada (i) i on izdvaja delove teksta koji se sastoje od jednog, dva ili tri karaktera. Ti karakteri mogu da budu cifre od 0 do 9 ili karakter - (minus). Ostatak izraza koji je izvan zagrada pronalazi i izdvaja ako se prethodni blok ponavlja dva ili tri puta za redom.

13.5. PONAVLJANJE NULA ILI VIŠE PUTA

Za razliku od vitičastih zagrada kada naglašavamo da se šablon definisan delom izraza ponavlnja određeni broj puta korišćenjem vitičastih zagrada sa jednim ili dva argumenta, specijalni karakter asterisk * sparuje nula ili više kopija delova izdvojenih izrazom iza kojeg se nalazi.

Tekst: Tekst, pa 6107373, a onda opet 75473712 i zatim 80310012.

Izraz: [0-9]*

Izdvojeno: Tekst, pa 6107373, a onda opet 75473712 i zatim 80310012.

Asterisk naglašava da se obeleže delovi koji se ponavljaju nula ili više puta, dokle god karakteri ili nizovi karaktera koji slede odgovaraju delu regularnog izraza koji je ispred *.

13.6. NEGACIJA IZRAZA

Ako želimo da izdvojimo delove teksta u kojima se ne nalazi određeni karakter, tada koristimo karakter ^ tako što ga stavljamo ispred karaktera koji ne želimo da postoji u izdvojenom delu teksta. Specijalni simbol ^ sparuje jedan karakter koji nije sadržan unutar uglastih zagrada.

Tekst: Svi karakteri koji nisu slovo e se nalaze u rezultatu.

Izraz: [^e]*

Izdvojeno: Svi karakteri koji nisu slovo e se nalaze u rezultatu.

Tekst: Svi karakteri koji nisu samoglasnik su izdvojeni.

Izraz: [^[aeiou]]*

Izdvojeno: Svi karakteri koji nisu samoglasnik su izdvojeni.

13.7. IDVAJANJE DELOVA NA SAMOM POČETKU TEKSTA

Posebna funkcionalnost karaktera ^ je da ako se nalazi na početku linije tada izrazom naglašavamo da je ostatak izraza koji sledi primenjen samo na karaktere od samog početka teksta, a ne na slučaj kada izraz odgovara nekom delu teksta dalje od početka.

Tekst: Prva reč samo prvi put. Prva reč je sada drugi put. Prva.

Izraz: ^Prva

Izdvojeno: Prva reč samo prvi put. Prva reč je sada drugi put. Prva.

Primer kada nije naglašeno da se izraz koji sledi odnosi na delove teksta pri njegovom početku.

Tekst: Prva reč samo prvi put. Prva reč je sada drugi put. Prva.

Izraz: Prva

Izdvojeno: Prva reč samo prvi put. Prva reč je sada drugi put. Prva.

13.8. IZDVAJANJE DELOVA NA SAMOM KRAJU TEKSTA

Kao što postoji specijalna namena karaktera ^ da upari delove teksta koji odgovaraju izrazu koji iza njega sledi samo ako se to odnosi na deo teksta na njegovom samom početku, tako postoji karakter koji omogućava da se ista funkcionalnost primeni, ali da se izraz odnosi na deo teksta na njegovom samom kraju. Taj karakter je simbol za dolar \$.

On sparuje kraj teksta ako se ne radi u višelinijском načinu rada ili kraj bilo koje linije, kad je primenjen u višelinijском načinu rada.

Tekst: Izdvoj poslednje reci svakog reda.
Ako tekst ima vise redova.
Uvek izdvoj samo poslednju reč u redu.

Izraz: \s[a-z.]*\$

Izdvojeno: Izdvoj poslednje reči svakog reda.
Ako tekst ima vise redova.
Uvek izdvoj samo poslednju reč u redu.

13.9. IZDVAJANJE JEDNOG ILI VIŠE PONAVLJANJA DELA IZRAZA

Izraz nakon kojeg sledi plus + izdvaja jednu ili više kopija dela teksta koji odgovara izrazu posle kojeg je plus.

Tekst: Sušena banana na tacni.

Izraz: (na)+

Izdvojeno: Sušena banana na tacni.

13.10. IZDVAJANJE DELA IZRAZA NULA PUTA ILI JEDNOM

Izraz nakon kojeg sledi uitnik ? izdvaja deo teksta koji odgovara izrazu posle kojeg je upitnik nula puta ili jednom, tj. može da ne postoji u nizu ili ako postoji, biće uparen samo jednom.

Tekst: 100 200 450 120 500 00 340 020 800 5657800

Izraz: [1-9]?00

Izdvojeno: 100 200 450 120 500 00 340 020 800 5657800

Navedeni izraz izdvaja delove teksta gde se nalazi karakter koji je cifra od 1 do 9 (ne uključuje cifru 0) nula ili jednom, a da iza njega slede dva puta cifre 0, kao literali.

13.11. OPERATOR UNIJE ILI IZBORA

Sparuje levu ili desnu stranu izraza oko simbola cev (pipe) |. Sa leve i desne strane operatora unije može da se nalazi prost ili složeni regularni izraz.

Tekst: Operi ruke pre jela. Operi ruke posle jela.

Izraz: Operi ruke (pre | posle) jela

Izdvojeno: Operi ruke pre jela. Operi ruke posle jela.

Tekst: List je zelen

Papir je zelen

List je suv

Papir je suv

List je beo

Papir je savijen

List je izvrljan

Papir je izvrljan

Izraz: (List je (mali | zelen | suv | žut)) | (Papir je (beo | savijen | izvrljan))

Izdvojeno: List je zelen

Papir je zelen

List je suv

Papir je suv

List je beo

Papir je savijen

List je izvrljan

Papir je izvrljan

13.12. SPECIJALNI KARAKTERI KAO LITERALI

Kada imamo potrebu da koristimo specijalne karaktere u izrazima u svojstvu literala, tada moramo ispred njih da napišemo komandni znak za ekranizaciju specijalnog karaktera. Komandni znak za ekranizaciju je \ (*backshelsh*), a ovaj postupak se zove *escape*-ovanje.

Tekst: Cena telefona je \$156.75

Izraz: \\$[0-9.]*

Izdvojeno: Cena telefona je \$156.75

13.13. SKRAĆENICE ZA NAJČEŠĆE KORIŠĆENE OPSEGE KARAKTERA

U većini implementacija regularnih izraza postoje specijalne skraćenice za najčešće korišćene opsege karaktera. One mogu da se koriste bilo gde u izrazu ravnopravno kao i definisani opseg na način opisan ranije u radu.

\d Upariće svaki karakter koji je cifra, tj. karakter u opsegu od 0 do 9.

\D Upariće svaki karakter koji nije cifra, tj. karakter koji nije u opsegu od 0 do 9.

\s Upariće beline, tj. karaktere koji stvaraju prazninu u tekstu (razmak, tabualtor itd.)

\S Upariće sve što nije belina, tj. karaktere koji se vide u tekstu i u štampi.

\w Upariće sve karaktere u opsezima od 0 do 9, A - Z i a - z. Sve alfanumerike.

\W Upariće sve karaktere koji nisu u opsezima u kojima su alfanumerici.

\b Upariće svaki karakter koji se nalazi na granici dela teksta koji predstavlja reč.

\B Upariće svaki karakter koji se ne nalazi na početku ili na kraju reči.

Tekst: Cena telefona je \$156.75

Izraz: \\$[\d.]*

Izdvojeno: Cena telefona je \$156.75

Tekst: Sve samo ne 194524 brojeve da 736293 izdvojis.

Izraz: [\D]*

Izdvojeno: Sve samo ne 194524 brojeve da 736293 izdvojis.

Tekst: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Izraz: \s

Izdvojeno: Ova_recenica_ima_razmake_izmedju_reci._Kako_drugacije_da_ima_reci?

Tekst: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Izraz: \S*

Izdvojeno: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Tekst: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Izraz: \w*

Izdvojeno: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Tekst: Ova recenica ima razmake izmedju reci. Kako drugacije da ima reci?

Izraz: \W*

Izdvojeno: Ova_recenica_ima_razmake_izmedju_reci._Kako_drugacije_da_ima_reci?

Tekst: Banana na stolu a jabuka na polici.

Izraz: \bna\b

Izdvojeno: Banana na stolu a jabuka na polici.

Tekst: Banana na stolu a jabuka na polici.

Izraz: \Bna\B

Izdvojeno: Banana na stolu a jabuka na polici.

13.14. RAD SA UNIKODNIM KARAKTERIMA

Kada su nastali, regularni izrazi su bili dizajnirani tako da omoguće rad sa tekstom čiji su karakteri pisani po ASCII kodnoj knjizi. Kasnije su proširivane mogućnosti regularnih izraza tako da omoguće rad sa unikodnim karakterima, tj. karakterima enkodiranim UTF-8 kodnim rasporedom, te je sada moguće raditi sa ćiriličnim karakterima, proširenom latinicom, grčkim pismom, kineskim, koreanskim, japanskim, burmanskim karakterima itd.

Većina skraćenica koje su gore pomenute u opsege karaktera uključuju karaktere za više pisama i jezika, ali to nije uvek slučaj.

Tekst: Latinica i 194524 brojevi su 736293 izdvojeni.
Моргу и ћирилицом да се 746538 пишу бројеви.
二千三百二十三 ili 1 2 3 4 5 4 .

Izraz: [\d]*

Izdvojeno: Latinica i 194524 brojevi su 736293 izdvojeni.
Моргу и ћирилицом да се 746538 пишу бројеви.
二千三百二十三 ili 1 2 3 4 5 4 .

Iz ovog primera može da se vidi da određeni karakteri u nekim pismima, kao što je slučaj sa japanskim pismom u datom primeru, nisu obeleženi kao cifre i ako jesu u pitanju cifre.

二千三百二十三 predstavlja zapis broja 2323, dok je 1 2 3 4 5 4 zapis broja 123454.

Još jedan primer koji ilustruje izdvajanje karaktera koji se koriste u rečima je:

Tekst: Slova koja mogu da se koriste, a latinicna su.
Слова која могу да се користе, а ћирилична су.
かなと漢字を使用することができます。

Izraz: [\w]*

Izdvojeno: Slova koja mogu da se koriste, a latinicna su.
Слова која могу да се користе, а ћирилична су.
かなと漢字を使用することができます。

Iz ovog primera može da se vidi da komandni karakter \w koji se odnosi na sve karaktere koji se koriste u rečenicama, diskriminiše sva ostala pisma osim latinice, što je još jedno od ograničenja korišćenja predefinisanih komandnih karaktera kao skraćenica za definisanje izrata.

14. IZDVAJANJE DELOVA TEKSTA POMOĆU REGULARNIH IZRAZA

Kada postoji potreba da se iz veće količine teksta izdvoje delovi koji odgovaraju određenom šablonu, najlakše je koristiti regularne izraze. U ovom delu rada ću navesti primere izdvajanja određenih tipova podataka u vidu ilustracije načina primene regularnih izraza.

14.1. IZDVAJANJE BROJEVA ZAPISANIH U VIŠE FORMATA

Brojevi mogu da budu zapisani na više načina:

- 1 Kao niz cifara
- 2 Kao niz cifara sa decimalnim zarezom
- 3 Kao niz cifara sa decimalnim zarezom i tačkom za grupisanje hiljada
- 4 Kao niz cifara sa decimalnim zarezom prećen malim ili velikim slovom e (e ili E), praćen simbolom minus ili plus, iza kojeg mora da sledi neprekidan niz cifara

Uzmimo na primer sledeći tekst:

Izmerene vrednosti:

134.190,75

134.187,98

134.189,88

134.191,02

134.190,76

Srednja vrednost je:

134.190,08

Ukupno izvršeno merenja:

5

Devijacija uzorka:

1,24948

Standardna greška:

5,58787e-1

Za izdvajanje svih brojeva iz navedenog teksta može da se koristi sledeći regularni izraz:

$([\backslash d, .]+e[-+]?[\backslash d]*)|([\backslash d, .]+)$

Izraz će izdvojiti sledeće delove teksta:

Izmerene vrednosti:

134.190,75

134.187,98

134.189,88

134.191,02

134.190,76

Srednja vrednost je:

134.190,08

Ukupno izvršeno merenja:

5

Devijacija uzorka:

1,24948

Standardna greška:

5,58787e-1

Prvi deo izraza, sa leve strane operatora unije ($([\backslash d,.] + e[-+]?[\backslash d]^*)$) će izdvojiti broj zapisan u obliku 5,58787e-1, dok će izraz sa desne strane operatora unije ($([\backslash d,.] +)$) izdvojiti sve brojeve koji sadrže pored cifara i karaktere tačka i zarez.

Ovako napisan regularni izraz će izdvojiti i brojeve napisane anglosaksonskim formatiranjem decimalnog zareza, tj. u tom slučaju decimalne tačke i zareza za grupisanje hiljada.

Primer teksta prevedenog na engleski jezik:

```
Measurements:
    134,190.75
    134,187.98
    134,189.88
    134,191.02
    134,190.76
Average:
    134,190.08
Number of measurements:
    5
Standard deviation:
    1.24948
Standard error:
    5.58787e-1
```

Isti regularni izraz se primenjuje za izdvajanje brojeva:

```
([\backslash d,.] + e[-+]?[\backslash d]^*) | ([\backslash d,.] + )
```

Izraz će izdvojiti sledeće delove teksta:

```
Measurements:
    134,190.75
    134,187.98
    134,189.88
    134,191.02
    134,190.76
Average:
    134,190.08
Number of measurements:
    5
Standard deviation:
    1.24948
Standard error:
    5.58787e-1
```

14.2. IZDVAJANJE REČENICA

Kada postoji potreba da se dugačak tekst na srpskom jeziku podeli u rečenice za potrebe kontekstualne analize ili neke druge obrade mogu se koristiti regularni izrazi za tokenizaciju teksta:

Uzmimo na primer sledeći tekst:

Sva ljudska bića rađaju se slobodna i jednaka u dostojanstvu i pravima. Ona su obdarena razumom i svešću i treba jedni prema drugima da postupaju u duhu bratstva. Svakom pripadaju sva prava i slobode proglašene u ovoj Deklaraciji bez ikakvih razlika u pogledu rase, boje, pola, jezika, veroispovesti, političkog ili drugog mišljenja, nacionalnog ili društvenog porekla, imovine, rođenja ili drugih okolnosti. Dalje, neće se praviti nikakva razlika na osnovu političkog, pravnog ili međunarodnog statusa zemlje ili teritorije kojoj neko lice pripada, bilo da je ona nezavisna, pod starateljstvom, nesamoupravna, ili da joj je suverenost na ma koji drugi način ograničena. Svako ima pravo na život, slobodu i bezbednost ličnosti. Niko se ne sme držati u ropstvu ili potčinjenosti: ropstvo i trgovina robljem zabranjeni su u svim oblicima. Niko se ne sme podvrgnuti mučenju ili svirepom, nečovečnom ili ponižavajućem postupku ili kažnjavanju. Svako ima pravo da svuda bude priznat kao pravni subjekt. X

Izraz napisan tako da deli u tokene ovaj tekst je: `[\w šđććžšĐČĆŽ, :]+\\.`

Nakon tokenizacije, tekst bi bio podeljen u delova koje odgovaraju rečenicama i to:

- 1 Sva ljudska bića rađaju se slobodna i jednaka u dostojanstvu i pravima.
- 2 Ona su obdarena razumom i svešću i treba jedni prema drugima da postupaju u duhu bratstva.
- 3 Svakom pripadaju sva prava i slobode proglašene u ovoj Deklaraciji bez ikakvih razlika u pogledu rase, boje, pola, jezika, veroispovesti, političkog ili drugog mišljenja, nacionalnog ili društvenog porekla, imovine, rođenja ili drugih okolnosti.
- 4 Dalje, neće se praviti nikakva razlika na osnovu političkog, pravnog ili međunarodnog statusa zemlje ili teritorije kojoj neko lice pripada, bilo da je ona nezavisna, pod starateljstvom, nesamoupravna, ili da joj je suverenost na ma koji drugi način ograničena.
- 5 Svako ima pravo na život, slobodu i bezbednost ličnosti.
- 6 Niko se ne sme držati u ropstvu ili potčinjenosti: ropstvo i trgovina robljem zabranjeni su u svim oblicima.
- 7 Niko se ne sme podvrgnuti mučenju ili svirepom, nečovečnom ili ponižavajućem postupku ili kažnjavanju.
- 8 Svako ima pravo da svuda bude priznat kao pravni subjekt.

14.3. IZDVAJANJE STRANIH REČI IZ ĆIRILIČNOG TEKSTA

Kada je potrebno iz nekog dugačkog teksta izdvojiti sve reči pisane drugim pismom, može se koristiti regularni izraz u te svrhe.

Uzmimo na primer sledeći tekst:

Регуларни изрази (regular expressions) је израз у виду низа карактера са тачно одређеном синтаксом која омогућава спаривање делова текста, издвајање појединих делова текста на основу дефинисаног правила или шаблона, као и комплексне изразе за замену појединих делова текста другима на основу изрази којим су делови који се мењају упарени. X

Regularni izraz koji izdvaja reči koje nisu pisane ćirilicom može da bude napisan ovako:

$$[\wedge[A-\text{ш}, \backslash.\backslash)\backslash(\text{ћћљљјцѣѣљљјц}\backslash d\backslash-]*]^{*}$$

Delovi teksta koji će biti izdvojeni primenom navedenog regularnog izraza nad tekstom su:

Регуларни изрази (regular expressions) је израз у виду низа карактера са тачно одређеном синтаксом која омогућава спаривање делова текста, издвајање појединих делова текста на основу дефинисаног правила или шаблона, као и комплексне изразе за замену појединих делова текста другима на основу изрази којим су делови који се мењају упарени. X

Iz datog primera se može videti da regularni izrazi kogu da se koriste za isključivanje određene grupe karaktera, s tim da je evidentno još jedno ograničenje, a to je da opseg karaktera [А-щ] ne pokriva sva ćirilična slova koja se koriste u srpskom jeziku, već je potrebno dodatno navesti za isključivanje sledeće karaktera: ђ, ѣ, љ, њ, ј, џ, Ђ, Ѓ, Ђ, Љ, Ј, Џ.

15. TUMAČENJE GOVORNE KOMANDE

Komanda se u obliku teksta doprema programu koji je regularnim izrazima tumači putem eksternog programa za pretvaranje govora u tekst (text to speech), koji se u ovom radu koristi u svojstvu crne kutije (black box), tj. način rada samog TTS programa se ne analizira i nije potrebno poznavanje načina rada, već se samo koristi njegova funkcija pretvaranja, koja kao rezultat vraća niz karaktera. Taj vraćeni niz karaktera se provlači kroz module programa, od kojih je svaki napisan tako da tumači jednu komandu koja može da bude zadata na nekoliko načina.

15.1. MODUL ZA TUMAČENJE KOMANDE ZAHTEVA ZA DEFINICIJOM

Jedan od modula sistema za simulaciju inteligencije koji koristi regularne izraze je modul koji omogućava korisniku da zada komandu za dopremanje definicije ili objašnjenja pojma. Regularni izrazi se koriste za identifikaciju komande i za izdvajanje pojma čija se definicija zahteva.

15.1.1. Analiza govorne komande

Potrebno je analizirati moguće načine zadavanja komande. Zbog karakteristika TTS programa koji se koristi u implementaciji, a koje su loše u programskom modu za prepoznavanje govora na srpskom jeziku, u izradi sistema se koristi mod za engleski jezik.

Mogući načini zadavanja naredbe su:

- Define this.
- What is this?
- What's this?
- Definition for this.

Svaka od četiri navedene rečenice zahteva od sistema da dopremi definiciju ili objašnjenje za pojam this. Regularni izraz mora da bude napisan tako da identifikuje komandu i da izdvoji pojam.

Primer rečenica dobijenih od TTS programa:

```
Define a metre  
Definition for radio wave  
What's an acronym  
What is a zip code
```

Regularni izraz za identifikovanje komande: $^(defin(e|ition\ for)|\ what\ ?[i']s)$

Rezultat primene napisanog izraza nad spiskom rečenica dobijenih od TTS programa je:

```
Define a metre  
Definition for radio wave  
What's an acronym  
What is a zip code
```

Nakon što je uspešno identifikovana komanda, ostatak teksta koji sledi iza izdvojenog dela se pomoću sledećeg regularnog izraza može izdvojiti kao termin za koji se traži definicija ili objašnjenje.

Potrebno je naknadno normalizovati izdvojeni deo teksta, pre nego što se on kao argument prosledi funkciju koja pribavlja definiciju.

Regularni izraz za izdvajanje teksta koji sledi: `[\w\d ]*`

Regularni izraz će izdvojiti delove teksta:

- a metre
- radio wave
- an acronym
- a zip code

Normalizacija argument podrazumeva uklanjanje članova a, an, the i eventualno pretvaranje reči iz množine u jedninu, što se radi pomoću rečnika.

Nakon normalizacije, argumenti koji se prosleđuju funkciji za dobavljanje definicija su:

- metre
- radio wave
- acronym
- zip code

15.1.2. Funkcija za dobavljanje definicije

Jedan od načina za dobavljanje definicije podrazumeva korišćenje javno dostupnih servisa i enciklopedija znanja kao što su Wikipedia ili Dictionary.Reference.com.

Ako se koristi Dictionary.Reference.com za dobavljanje definicije za traženi termin, rezultat je HTML stranica iz koje se izdvajaju sekcije označene klasom body u kojoj se nalazi tekst definicije. Primer rezultata pribavljanja definicije za termin metre sa linka za dobavljanje definicija putem ovog servisa koji se nalazi na nadictionary.reference.com/browse/metre je:

— n

1. a metric unit of length equal to approximately 1.094 yards
2. the basic SI unit of length; the length of the path travelled by light in free space during a time interval of 1/299 792 458 of a second. In 1983 this definition replaced the previous one based on krypton-86, which in turn had replaced the definition based on the platinum-iridium metre bar kept in Paris. X

Naknadnom normalizacijom dobijenog teksta nakon pribavljanja sa servisa, tekst može da se prilagodi i pripremi za iščitavanje, tj. pretvaranje u govor korišćenjem programa za sintetizaciju govora koji se, kao i TTS program, u ovom sistemu koristi kao crna kutija, bez dodatne analize načina rada STT programa.

Normalizacija može da podrazumeva zamenu najčešćih delova teksta kao što je deo — n koji u definiciji naznačava da je u pitanju imenica, pa je taj deo moguće zameniti rečenicom “this is a noun”.

Drugi element normalizacije podrazumeva zamenu nabiranja originalnim tekstom koji je izdvojen regularnim izrazom iz komande praćen adekvatnim oblikom glagola to be, tako da pripremljeni tekst za iščitavanje izgleda ovako:

This is a noun.

A meter is a metric unit of length equal to approximately 1.094 yards

A meter is the basic SI unit of length; the length of the path travelled by light in free space during a time interval of 1/299 792 458 of a second. In 1983 this definition replaced the previous one based on krypton-86, which in turn had replaced the definition based on the platinum-iridium metre bar kept in Paris.×

15.2. MODUL ZA EKSTRAKCIJU ELEMENATA IZ GOVORNE KOMANDE

Modul za ekstrakciju elemenata iz govorne komande je deo sistema za simulaciju inteligencije, koji iz teksta dobijenog od STT programa treba da analizom, deljenjem i identifikovanjem funkcionalnih celina markira i izvuče delove teksta sa specijalnim značenjem, dodeli im tip, komandnu vrednost ili funkciju na osnovu kojeg će program za sprovođenje, tj. izvršavanje komande da se upravlja.

15.3. IDENTIFIKOVANJE FUNKCIONALNIH CELINA

Postoji više tipova govornih komandi. Svaka se sastoji iz instrukcija i argumenata. Jedna naredba može da bude inicirana sa više različitih instrukcija. Svaka instrukcija kojom može da se inicira isti proces je sinonim osnovne instrukcije.

Za program koji upravlja izvršavanjem komande, pokazatelj koji utvrđuje šta će se izvršiti je osnovna instrukcija. Kada program detektuje sinonime, on ih preimenuje u osnovnu instrukciju.

Argumenti naredbe od kojih se komanda sastoji su reči koje se normalizacijom dovode u takav oblik i stanje da mogu da se koriste kao parametri funkcija na koju ukazuje svaka osnovna instrukcija sadržana u naredbi.

Tipovi argumenata su različiti, a mogu biti između ostalih:

- reč ili grupa reči
- broj ili grupa brojeva
- datum ili vreme, apsolutno vreme
- vremenski interval, relativno vreme
- registrovano ime
- itd.

15.4. EKSTRAKCIJA FUNCIONALNIH CELINA REGULARNIM IZRAZIMA

Za identifikaciju i izdvajanje celina iz naredbi koriste se regularni izrazi, a opisnim šablonom je određeno da li je izvučeni deo teksta reč, broj, datum itd.

Za ekstrakciju brojeva, koristi se izraz predstavljen u prethodnom poglavlju, gde su navedeni primeri regularnih izraza za različite tipove podataka.

Izraz za izdvajanje brojeva će u tekstu identifikovati sve brojeve zapisane u bilo kom formatu. To je u principu suvišno, obzirom da STT programi uvek brojeve daju u tekstualnom obliku.

Primer:

Ukoliko korisnik aplikacije izgovori sledeću rečenicu:

```
Calculate 1020 + 79.
```

STT program će ovu izgovorenu rečenicu programu za analizu i ekstrakciju elemenata proslediti u sledećem obliku:

```
Calculate one thousand and twenty plus seventy nine.
```

Prema tome, pre nego što tekst u ovom obliku bude prenet programu za ekstrakciju, on mora da bude normalizovan, tj. da određeni tipovi podataka budu prevedeni u zapis koji je odgovarajući za analizu i identifikovanje argumenata naredbi.

15.5. MODUL ZA NORMALIZACIJU TIPOVA PODATAKA

Podaci poput brojeva i datuma kakvi se u govornom jeziku javljaju do STT programa dolaze u obliku toka podataka koji predstavlja zvučni signal, koji se nakon konverzije pretvara u tekst u obliku u kojem se direktno prevode reči onako kako su izgovorene.

To znači da ako korisnik izgovori datum, on će u tekstu biti zapisan rečima. Takođe, datum će biti zapisan onako kako je izgovoren, a ne u standardizovanom formatu.

Primer: Korisnik sistemu za simulaciju inteligencije zadaje sledeću govornu komandu:

```
Remind me to call Ann on September 9th 2013 at 11:45.
```

Dati iskaz će iz STT programa izaći u obliku:

Remind me to call Ann on September ninth two thousand thirteen at eleven fourty five. X

Modul za normalizaciju treba da uračuna više mogućih izgovora datuma, brojeva i ostalih tipova podataka i da ih prepíše u standardizovanom obliku. Pomenuti primer nakon normalizacije treba da bude upisan u sledećem obliku:

Remind me to call Ann on 2013-09-09 at 11:45.

Način na koji modul za normalizaciju tipova podataka konvertuje tipove podataka iz zapisa izvedenog direktno iz govornog jezika u standardizovan format je unapred utvrđenim postupkom za konverziju.

Postoji veliki broj algoritama za identifikovanje brojeva, datuma i ostalih tipova podataka u tekstu i za njihovo prepisivanje u standardizovane oblike.

U ovom radu ću koristiti najjednostavniji oblik za direktno prevođenje, koji radi tako što pronalazi reči koje predstavljaju cifre ili brojeve 10, 100, 1000 i ostale stepene broja 10, a zatim množenjem od najvećeg stepena sa susednim brojem ili cifrom dobija grupu brojeva u nizu, koja prostim sabiranjem svih dobijenih brojeva u nizu u jedan akumulator, konačno za cilj ima formiranje jednog celog broja prevedenog iz tekstualnog zapisa rečima u tekstualni zapis ciframa.

Primer: Sledeća govorna komanda koja sadrži brojeve se normalizuje:

Calculate one thousand and twenty plus seventy nine.

Prevođenje stepena broja deset rezultira sledećim oblikom:

Calculate one 1000 and twenty plus seventy nine.

Prevođenje brojeva iz rečnika rezultira sledećim oblikom:

Calculate 1 1000 and 20 plus 70 9.

Identifikovanje govornih veznika i njihovo ignorisanje:

Calculate 1 1000 20 plus 70 9.

Potrebno je identifikovati nizove brojeva i grupisati ih za množenje sa susednim brojem pored stepena broja 10:

Calculate 1 1000 20 plus 70 9.

Nakon množenja susednih brojeva u nizovima:

Calculate 1000 20 plus 70 9.

Nakon sabiranja nizova brojeva:

Calculate 1020 plus 79.

Dobijeni tekst se prenosi modulu za izvršavanje naredbe koja će brojeve 1020 i 79 identifikovati kao argumente, reč plus kao operator, tj. specijalni argument za određivanje funkcije. Naredba se identifikuje na osnovu reči Calculate.

16. IZVRŠAVANJE GOVORNE KOMANDE

Osim tumačenja govornih komandi upućenih od strane korisnika, sistem takođe poseduje deo zadužen za izvršavanje tih komandi ili naredbi od kojih su sastavljene.

Komande se izvršavaju na više načina. Najčešći način je interpretersko tumačenje naredbi reč po reč. Drugi način je pretragom rečnika pojmova i naredbi od kojih komanda može da bude sastavljena, a zatim izvršavanje adekvatne funkcije na koju ukazuje zapis u rečniku. Deo sistema sa izvršnom funkcijom je modul za izvršavanje naredbe.

16.1. MODUL ZA IZVRŠAVANJE NAREDBE

Svaka naredba se nakon normalizacije prosleđuje na izvršavanje u modul za izvršavanje naredbe. Ovaj modul čita registar kontrolera da odluči kojem kontroleru da delegira izvršenje. Kontroler je zadužen za izvršavanje same funkcije koja se naredbom zahteva. Funkcija ovog modula je da iz registra pročita definicije u obliku regularnih izraza sa kojima upoređuje dobijeni tekst naredbe. Ako neki od registrovanih izraza uspe da utvrdi da je kontroler, za koji je konkretan izraz vezan, sposoban da izvrši naredbu, modul će na dalju obradu proslediti tekst naredbe registrovanom modulu. Modul tada preuzima dužnost izvršavanja naredbe.

16.2. REGISTAR KONTROLERA

Sistem za simulaciju inteligencije, da bi mogao da izvrši veliki broj različitih komandi, mora da bude pisan modularno. Pristup koji je u ovoj implementaciji odabran je da se koriste kontroleri koji izvršavaju naredbe koje su nakon normalizacije prosledjene modulu za izvršavanje naredbi. Svaki od ovih kontrolera mora da bude registrovan kako bi sistem za simulaciju inteligencije delegirao izvršenje naredbe jednom od njih.

Registar je način da se sistem obavesti o tome kojim kontrolerima može da delegira izvršenje, ali i da mu pomogne da odluči koji kontroler je najadekvatniji za izvršenje naredbe. Registar je realizovan u formi rečnika, gde postoje parovi ključ-vrednost, gde je ključ u paru regularni izraz sa kojim modul za izvršenje naredbe uparuje naredbu, a vrednost je ime kontrolera kojem će sistem delegirati izvršenje naredbe ako regularni izraz, koji čini ključ para uspešno upari sve elemente naredbe i na taj način potvrdi da je njegov kontroler napisan na taj način da može da protumači naredbu i da zna da je izvrši.

Ne mogu da postoje dva identična ključa, tj. dva identična regularna izraza, ali je moguće da više različitih izraza ukazuju na isti kontroler.

Prema tome, ključ u registru kontrolera je jedinstven, dok vrednost nije. Jedino ograničenje koje se odnosi na vrednost je to da kontroler na koji ključ ukazuje mora da postoji, tj. da je njegova izvršna datoteka prisutna u direktorijumu izvršnih datoteka sistema.

Primer zapisa u registru:

Key: `^Calculate\s(\d+)\s(times|minus|divided\sby|multiplied\sby|plus)\s(\d)+.*$`
Value: `simple_calculator_controller`

16.3. KONTROLER ZA IZVRŠAVANJE NAREDBE

Kontroler je izvršna jedinica koja je napisana kao nezavisna celina, tj. nezavisan program koji čini podsistem unutar sistema za simulaciju inteligencije.

Svaki kontroler u sistemu ima zajedničku osobinu, a to je da na isti način preuzima ulazni niz karaktera koji predstavlja komandu korisnika sistemu, koja je prošla sve korake normalizacije i adaptacije i da nakon izvršavanja adekvatne funkcije na standardizovan način, koji je zajednički svim kontrolerima vrati odgovor sistemu koji će dalje biti usmeren na obradu, prilagođavanje za izgovor i zatim sintezu govornog signala koji će biti upućen kao odgovor korisniku.

Svaki kontroler mora da bude sposoban da tumači barem jedan set naredbi koje odgovaraju tačno definisanom regularnom izrazu.

Kontroler može da bude sposoban da tumači i interpretira i više od jedne vrste naredbe, gde je svaka opisana posebnim regularnim izrazom. Svaki od ovih regularnih izraza su ključevi u registru kontrolera sistema.

Primer kontrolera

Ovde je naveden primer kontrolera sa njegovim načinom funkcionisanja. Svaki kontroler, pored imena i spiska regularnih izraza (numerisanih od 1 do N po prioritetu uparivanja) sadrži i specifične regularne izraze za svaku od funkcija koje obavlja.

Primer zapisa u zaglavlju kontrolera:

Ime kontrolera u registru:

`simple_calculator_controller`

Regularni izrazi koji uparuju komande koje može da izvrši:

[1]: `^Calculate\s(\d+)\s(times|minus|divided\sby|multiplied\sby|plus)\s(\d+)\. *$`

Simple Calculator kontroler ima samo jedan regularni izraz koji uparuje naredbu koju je sposoban da izvrši, a nakon što mu je zadatak da je izvrši delegiran, on je deli u tri celine:

- 1 Prvi operand
- 2 Operator
- 3 Drugi operand

Ekstrakcija operandada se vrši pomoću regularnog izraza: **(\d+)**, dok se ekstrakcija operatora vrši pomoću regularnog izraza: **(times|minus|divided by|multiplied by|plus)**.

Uzmimo da je naredba prosleđena na izvršavanje, nakon normalizacije, sledećeg sadržaja:

`Calculate 1020 plus 79.`

Uparivanje delova naredbe sa regularnim izrazom za izdvajanje operandada daje sledeće rezultate:

```
1 1020
2 79
```

Uparivanje delova naredbe sa regularnim izrazom za izdvajanje operatora daje sledeći rezultat:

```
1 plus
```

Kontroler omogućava četiri funkcije, od kojih je svaka direktno vezana sa uparenim operatorom ili sa više njih i to na sledeći način:

- Za operator **plus** je vezana funkcija sabiranja operandada.
- Za operator **minus** je vezana funkcija oduzimanja drugog operandada od prvog.
- Za operatore **times** i **multiplied by** je vezana funkcija množenja operandada.
- Za operator **divided by** je vezana funkcija deljenja prvog operandada drugim.

Nakon odabira adekvatne funkcije koju treba pokrenuti, kontroler joj kao parametre prosleđuje uparene operande, a kao rezultat dobija povratnu vrednost u obliku niza karaktera koje predstavljaju decimalni zapis rezultata adekvatne aritmetičke operacije uz adekvatnu prateću poruku korisniku.

U slučaju datog primera, rezultat izvršavanja funkcije sabiranja će biti:

```
Result of 1020 + 79 = 1099.
```

Kontroler kojem je delegirano izvršavanje naredbe rezultat vraća u pripremljenom obliku sistemu koji dalje rezultat izvršavanja naredbe prosleđuje modulu za normalizaciju rezultata koji će preimenovati delove odgovora.

17. SAOPŠTAVANJE REZULTATA

Nakon izvršavanja govorne komande i naredbe u njoj, a zatim i dobijanja rezultata od modula za izvršavanje, koji je putem kontrolera kojem je izvršavanje bilo delegirano sistemu isti i prosledio, potrebno je rezultat saopštiti korisniku.

Deo sistema za saopštavanje rezultata je podeljen u nekoliko procesnih celina, od kojih svaka priprema rezultat dobijen u delu za izvršenje i sprovodi ga do spoljnog programa za sintezu govora.

18. MODUL ZA NORMALIZACIJU REZULTATA

Modul za normalizaciju rezultata ima zadatak da dobijeni odgovor prilagodi za izgovor. Osnove preimenovanja delova se sastoje u preslovljavanju ili u prepisivanju podataka iz simbola u reči, tj. u one koje direktno odgovaraju govornom jeziku.

Prevođenje se vrši na dva načina:

- 1 Korišćenjem rečnika
- 2 Korišćenjem algoritama za konverziju

U postupku normalizacije se izvršava prvo prevođenje korišćenjem rečnika, a zatim se sprovodi prevođenje korišćenjem algoritma, koji će prevesti sve što nije bilo moguće prevesti pomoću rečnika.

Normalizacija rezultata korišćenjem rečnika

Ovaj način da se odgovor sistema korisniku pripremi za izgovor podrazumeva, da simboli i karakteri čije čitanje ne može biti obavljeno direktno od strane programa za sintezu govora, budu prepisani rečima, tako da izgovor bude moguć.

Primer rečnika:

+	plus	)	closed parenthesis	€	euros
-	minus	&	and	#	hash
_	underscore	%	percent	=	equals
(	open parenthesis	\$	dollars	°C	degrees Celsius
				°F	degrees Fahrenheit

Primer rezultata pre i nakon sprovođenja kroz proces normalizacije korišćenjem rečnika:

Pre normalizacije: Result of 1020 + 79 = 1099.

Posle normalizacije: Result of 1020 plus 79 equals 1099.

Normalizacija rezultata korišćenjem algoritma

Ovaj način podrazumeva da se odgovor sistema korisniku prepíše tako da se podaci zapisani u nekom obliku koji nije adekvatan za prosleđivanje programu za sintezu govora prepíšu na način koji jeste. Trenutna implementacija dela modula za normalizaciju odgovora zaduženog za prevođenje korišćenjem algoritma poseduje samo deo kojim se brojevi zapisani ciframa u decimalnom obliku prevode u reči. Postupak podrazumeva da se decimalni zapis broja ciframa razloži na celine koje predstavljaju jedinice, desetice, stotine, hiljade i ostale stepene baze zapisa, a da se zatim brojevi rečnikom prevedu u reči.

Primer postupka prevođenja brojeva iz prethodnog primera je ilustrovan sledećim koracima:

Result of 1020 plus 79 equals 1099.

Deljenje na tokene prema stepenu baze decimalnog zapisa osim onih koji imaju posebno čitanje, kao što su u engleskom jeziku brojevi ispod 100:

1 thousand 0 hundred 9 10 9

Zatim se eliminišu delovi pored kojih se nalaze brojevi koji u govoru rezultuju izostavljanje izgovaranja dela broja, kao što je slučaj sa delom u primeru koji se odnosi na stotine:

1 thousand ~~0 hundred~~ 9 10 9

Zatim se asimiliraju delovi koji imaju poseban izgovor, tj. koji su izuzeci:

1 thousand ninety 9

Sledeći korak je da se prepíšu brojevi koji ostaju pored stepena broja deset ili koji su samostalni, tj. cifre jedinice:

one thousand ninety nine

Nakon normalizacije broj je za izgovor pripremljen u obliku: one thousand ninety nine.

Primer primene identičnog postupka na broju 1020.

1	1 thousand 0 hundred 2 10 0
2	1 thousand 2 10
3	1 thousand twenty
4	one thousand twenty

Primer primene identičnog postupka na broju 79.

1	7 10 9
2	7 10 9
3	seventy 9
4	seevnty nine

Rezultat nomalizacije odgovora sistema korisniku prirpempljenog za sintezu u govor je:

Result of one thousand twenty plus seventy nine equals one thousand ninety nine.

19. SINTEZA GOVORA

Sinteza govora je proces kojim se korišćenjem računara veštački kreira ljudski govor. Sintetizatori govora mogu da budu ugrađeni u druge programe u obliku biblioteke ili mogu da se koriste kao nezavisni programi. U implementaciji koju opisuje ovaj rad koristi se eksterni program eSpeak - Speech Synthesizer koji je program otvorenog koda i besplatan je.

Programi za sintezu govora mogu da budu ugrađeni u softveski paket, ali mogu da se ugrađuju i u sam hardver, upisani u čip zadužen za konvertovanje teksta poslatog magistralom na port kontrolera u zvučni signal koji se vodi na pojačavač i zatim na priključak koji vodi na zvučnike.

Postoje dva osnovna postupka za sintezu govora.

19.1. SINTEZA GOVORA SPAJANJEM SNIMLJENIH DELOVA GOVORA

Ovaj postupak podrazumeva da se tekst pretvara u govor tako što se delovi teksta dobijeni deljenjem istog na glasove, dvoglas ili cele slogove, formira nadovezivanjem unapred snimljenih zvučnih zapisa konkretnog glasa, dvoglasa ili sloga koji se u datom trenutku iščitava i izgovara. U slučaju jezika koji nemaju pravilno glasovno čitanje, kao što je engleski jezik, često je potrebno snimati svaku reč pojedinačno i celu snimljenu reč nadovezivati u izgovor tamo gde se u tekstu pominje.

19.2. SINTEZA GOVORA FONETSKOM TRANSKRIPCIJOM

Ovaj postupak može da se koristi za bilo koji jezik. On je znatno sporiji od sinteze spajanjem snimljenih delova govora, ali je po kvalitetu sintetizovanog govora znatno kvalitetniji.

Da bi se izvršila sinteza govora ovim postupkom prvo je potrebno izvršiti fonetsku transkripciju reči. Za jezike kod kojih je čitanje fonetsko, vrši se direktno preslovljavanje uz neophodnu proveru izuzetaka koje svaki jezik ima, dok se za jezike koji nemaju fonetsko čitanje svog pisma, vrši preslovljavanje prema zapisu u rečniku jezika u IPA (International Phonetic Alphabet) jezičku formu, tako da svaki karakter teksta, tj. svaki simbol uvek ima jedan glas ili da grupa simbola uvek čini identično izgovoren dvoglas ili slog.

Uzmimo na primer rečenicu:

This is a document. X

Rečenica na engleskom jeziku ne može direktno da se pročita zbog specifičnosti engleskog jezika da nema pravilan izgovor koji može uvek da se utvrdi na osnovu kombinacije slova korišćenih u formiranju reči, već se svaka reč izgovara na sebi svojstven način. Potrebno je izvršiti prepisivanje rečenice iz engleskog alfabeta u International Phonetic Alphabet, tj. IPA. Prevedena rečenica bi mala sledeći zapis u skladu sa IPA:

'ðɪs ɪz ə 'dɒkjəmənt.

Sledeći korak je da se rečenica zapisana u IPA tokenizuje u manje delove za koje može da se izvrši sinteza glasa koji svaki token predstavlja. Vodi se računa da se tihi galsovi, tj. glasovi koji se na nekim mestima uopšte ne čuju u izgovoru, spoje sa glasom koji u izgovoru delimično menjaju glas u odnosu na izgovor kakav bi on bio da je sam ili u rečenici stoji pored drugog glasa. U pitanju je prosta korekcija IPA zapisa da bi izgovor poštovao sve poznate izuzetke koji posotje u govoru.

'ð ɪ s ɪ z ə 'd α k j ə m ə n t.

Ovako podeljena rečenica sada može da bude izgovorena glas po glas.

Glasovi se ne snimaju da bi se reprodukovali snimci, već se po matematičkoj formuli generiše složeni signal, koji se sastoji od više sinusoida različitih talasnih dužina, čijim kombinovanjem u jedan signal i reprodukcijom računar stvara izgovor koji je prirodniji od prostog nadovezivanja snimljenih glasova, koji često bivaju isprekidani i nepovezani.

Formiranjem jedinstvenog signala koji je neprekidan i čije komponente se ne prekidaju naglo, već postepeno iščeznu, program za sintezu zvuka uspeva da obezbedi prirodniji veštački generisan izgovor.

Kako su jednačine za sintezu komponentata signala uglavnom proste trigonometrijske funkcije čiji parametri definišu talasnu dužinu, amplitudu i promenu amplitude i talasne dužine sa vremenom, moguće je formirati setove parametara za svaki glas koji bi odgovarali različitim glasovima.

Na ovaj način svaki set parametara za funkcije za generisanje komponentata pojedinačnih glasova može da definiše potpuno jedinstven glas, akcenat, dijalekt i da obezbedi veću lepezu prilagodljivosti programa za sintezu govora potrebama korisnika.

19.3. REPRODUKCIJA GOVORA

Kada je signal sintetizovan sledi njegovo reprodukovanje. Pošto je signal generisan u obliku audio zapisa on se sistemskim pozivom šalje na zvučnu karticu i reprodukuje, bez potrebe za dodatnom obradom, konverzijom i enkodiranjem određenim kodekom itd.

Reprodukcija zvuka se izvršava u nezavisnom procesu, tako da rad aplikacije nije pauziran dok se ne završi reprodukcija zvučnog signala u celosti, već aplikacija može paralelno da nasatvi da obavlja sve započete procese i aktivnosti.

20. ZAKLJUČAK

Sistem za simulaciju inteligencije u automatizovanom domu je moguće realizovati korišćenjem besplatnog softvera otvorenog koda koji se izvršava na hardverskoj platformi u rangu SBC računara kao što je Raspberry Pi, a na operativnom sistemu iz GNU/Linux familije i to na distribuciji Raspbian, koja je adaptirana Debian distribucija za Raspberry Pi. Softverska komponenta sistema je u potpunosti izrađena pomoću rešenja otvorenog koda.

Za skladištenje podataka se koristi besplatan Database Management System MySQL, dok je veći deo izvornog koda sistema napisan u skripting jeziku PHP, besplatnom skripting jeziku, a manji deo u programskom jeziku Java.

Demonstrirano je da je moguće korišćenje industrijskih standarda koji se koriste za komunikaciju sistema i uređaja u domaćinstvu i dat je opširan opis algoritama za sintezu profila za automatizaciju kojima se korisniku povećava komfor života i rada u sredini gde je implementiran sistem.

Pokazano je da je razvoj hardverske i softverske platforme sistema za simulaciju inteligencije u automatizovanom domu, kao i adekvatnih algoritama za rešavanje problema analize rada uređaja, monitoringa njihovog stanja i automatskom upravljanja drugim hardverskim sklopovima i softverskim paketima u domu, moguć uz male troškove izrade.

21. LITERATURA

- [1] *Intelligent Instrumentation: Principles and Applications*, CRC Press, 1st ed., Manabendra Bhuyan, 2010.
- [2] *Home Automation & Wiring*, The McGraw-Hill Companies Inc., 1st ed., James Gerhart, 1999.
- [3] *Practical Raspberry Pi*, Apress Media, Brendan Horan, 2003.
- [4] *Raspberry Pi Home Automation with Arduino*, Packt Publishing, Andrew K Dennis, 2013.
- [5] *Introducing Regular Expressions*, O'Reilly Media, 1st ed., Michael Fitzgerald, 2012.
- [6] *Regular Expressions Cookbook*, O'Reilly Media, 2nd ed., Jan Goyvaerts, 2012.
- [7] *An Introduction to Formal Languages and Automata*, 3rd ed., Jones and Bartlett Publishers inc., Peter Linz, 2000.
- [8] *Java Network Programming*, O'Reilly Media, 3rd ed., Elliotte Rusty Harold, 2004.
- [9] *PHP Objects - Patterns and Practice*, Apress Media, 3rd ed., Matt Zandstra, 2010.
- [10] *Text-to-Speech Synthesis*, Cambridge University Press, 1st ed., Paul Taylor, 2009.