



Univerzitet Singidunum

Tehnički fakultet

IOT – EVIDENCIJA PRISUSTVA PUTEM WIFI-JA

Diplomski rad

Mentor: Prof.Dr Tanasković Marko

Student: Sekulović Milorad 201905

Beograd, 2017

Sadržaj

| | | |
|-------|---|----|
| 1 | Apstrakt | 3 |
| 1.1 | Problem | 3 |
| 1.1.1 | IoT | 3 |
| 1.2 | Motivacija za projektom | 4 |
| 1.3 | Druga rešenja | 4 |
| 1.4 | Idejno rešenje | 6 |
| 2 | Organizacija projekta..... | 8 |
| 3 | Hardverska platforma | 11 |
| 3.1 | Raspberry Pi 3 | 11 |
| 3.1.1 | Operativni sistemi | 11 |
| 3.1.2 | SoC | 14 |
| 3.1.3 | RAM..... | 21 |
| 3.1.4 | I/O and LAN | 22 |
| 3.1.5 | WiFi/Bluetooth..... | 22 |
| 3.1.6 | Napajanje..... | 24 |
| 3.1.7 | GPIO | 26 |
| 3.2 | EKRAN | 26 |
| 3.2.1 | XPT2046 | 26 |
| 4 | Softverska implementacija | 29 |
| 4.1 | Konkurentna logika izvršavanja..... | 29 |
| 4.1.1 | Threads and processes in Java..... | 29 |
| 4.1.2 | Primer razlike u implementaciji singleton obrasca | 30 |
| 4.2 | Komunikacija izmedju programskih jezika | 32 |

| | | |
|--------|-------------------------------------|----|
| 4.2.1 | ARP Tabela | 33 |
| 4.3 | Organizacija baza podataka..... | 33 |
| 4.4 | Gui (Graphical User Interface)..... | 33 |
| 4.4.1 | JavaFx | 34 |
| 4.5 | Node.js server..... | 35 |
| 4.6 | Python skripte..... | 37 |
| 4.7 | Migracije podataka..... | 37 |
| 4.7.1 | Povezivanje sa Cloud-om..... | 38 |
| 4.7.2 | Razvoj servera pristupa | 38 |
| 4.7.3 | Generisanje Excel tabela | 38 |
| 4.8 | Access Point | 39 |
| 4.8.1 | Hostapd | 39 |
| 4.9 | DHCP/DNS | 39 |
| 4.9.1 | DHCP | 39 |
| 4.9.2 | DNS..... | 40 |
| 4.9.3 | DNSMASQ | 40 |
| 4.10 | Ip forwarding | 41 |
| 4.10.1 | Tabele rutiranja | 41 |
| 4.10.2 | Implementacija..... | 42 |
| 4.11 | Automatizacija..... | 42 |
| 5 | Verifikacija | 44 |
| 6 | Zaključak | 45 |
| 7 | Literatura | 46 |

1 Apstrakt

1.1 Problem

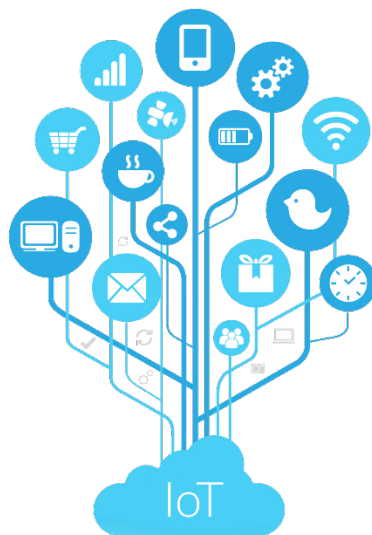
U ovom radu se rešava problem detekcije prisustva u raznim primjenama. Pod detekcijom prisustva podrazumijevaju se evidencije dolaska i odlaska, logovi prisutnosti i slično. Na ovaj način se automatizuje proces, jer je u mnogim primjenama nemoguće bez pomoći sistema evidencije zapravo vršiti evidenciju, što je uslovljeno velikom brojnošću ljudi koje treba evidentirati, kao i greškama namjernim ili nenamjernim za prijavom na evidenciju. Trenutno postoji veliki broj sistema za evidenciju zasnovani na raznim tehnologijama, ali o njima će biti riječi kasnije. Kada se govori o sistemu evidencije, postoje dvije strane, a to su strana koja evidentira prisustvo i strana koja je evidentirana. Evidencija se čuva na određenom medijumu. Ono što može biti dio sistema je i verifikacija u toku trajanja, da je subjekat evidencije zapravo prisutan. Konkretno problem evidencije se može javiti u kompanijama, na fakultetima, u saobraćaju, a sa razvojem IoT¹-a sve više i kod kuće, u privatne svrhe.

1.1.1 IoT

To je mreža fizičkih stvari, ugrađene elektronike, senzora, aktuatora i softvera, koji mogu da razmjenjuju podatke među sobom i sa spoljnim uređajima. IoT-GSI² je definisala IoT kao “Globalnu infrastrukturu informatičkog društva koja omogućava napredne usluge (fizičkim i virtualnim) umrežavanjem stvari, pritom se zasnivajući na postojećim i interoperabilnim informacionim i komunikacionim tehnologijama u razvoju. U tu svrhu termin – stvar predstavlja predmet fizičkog svijeta informatija ili riječ, koji je moguće identifikovati i koji može biti integrisan u komunikacionim mrežama” Sami termin IoT je predložen od strane Kevina Eštona 1999. Godine. Sva istraživanja pokazuju trend rasta samih uređaja i smatra se da će do 2020-e biti oko 26 milijardi uređaja. Ovakav tempo razvoja je i jedan od razloga za postanak IP protokola verzije šest, s’ obzirom da postojeći način adresiranja putem IP protokola verzije četiri ima mogućnost da adresira 4,3 miliona uređaja.

¹ Internet Of Things

² Global Standards Initiative on Internet of Things



Ilustracija 1: IoT

1.2 Motivacija za projektom

Sama motivacija za radom na ovakvom projektu se javila prilikom saradnje sa profesorima sa fakulteta i konstantnim radom na IoT projektima u okviru aktivnosti na fakultetu. Uz razvoj drugih projekata, javila se i ideja evidencije prisutnosti studenata na predavanjima, usred potreba profesora da smanje vrijeme potrebno za evidenciju prisutnih studenata, kako na predavanjima, tako i na polaganjima ispita i kolokvijuma. Ovo je dalo osnovnu motivaciju za rešenjem konkretnog problema, koji bi omogućio olakšanje rada i uštedu vremena.

1.3 Druga rešenja

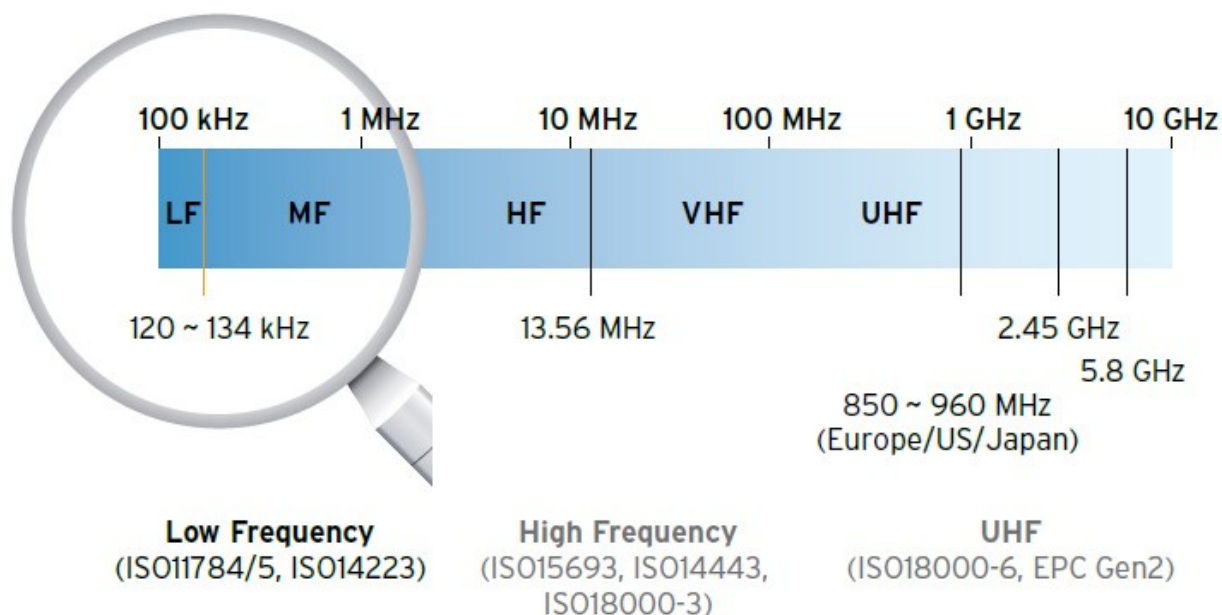
Kao rešenja koja se već nalaze na tržištu i koriste, to su kontrola pristupa i evidencije putem beskontaktnih kartica, otiska prsta, šifrom, infra crvenim senzorima, laserskim senzorima, kamerama i sl.

Zavisno od toga da li je bitno konkretno da se zna ko je i kada bio u određenoj prostoriji ili dijelu na koji je instaliran sistem, što zavisi i od projektovanja samog sistema, kao najpouzdaniji vid je kontrola pristupa i prisustva otiskom prsta. U praksi se međutim koristi najviše za kontrolu pristupa, ali ne i za praćenje vremena zbog potrebe da se postavi i na ulazu i na izlazu, što zahtijeva dodatne i bespotrebne troškove, takođe ukoliko postoji veliki broj ljudi koji treba da uđe u prostoriju za kratak period, to predstavlja problem.

Verifikacija putem šifre, u zavisnosti od broja korisnika zahtijeva veliki broj šifara koje bi bile dodijeljene svakom korisniku, a postoji i mogućnost da jedna osoba ukuca više puta istu šifru i prijavi se za ostale korisnike. Pristup putem kartica i tagova različite frekvencije u radijskom spektru je najčešće korišćen vid kontrole pristupa, kao i detekcije prisutnosti.

Kartice i tagovi se u pogledu frekvencije dijele na Nisko Frekventne, Visoko Frekventne i Ultra Visoko Frekventne, i oni se kreću u rangu oko 100kHz, 10MHz i 1GHz respektivno. Kartice mogu biti aktivne i pasivne, u zavisnosti od toga da li koriste neki izvor energije za napajanje ili ne. Svaki frekventni opseg ima svoje prednosti i mane, a postoje i hibridne kartice koje spajaju više prednosti ukoliko je potrebno. Razlike se ogledaju u količini podataka koje mogu da sačuvaju, blizini na kojoj mogu biti očitane, cijeni

proizvodnje.



Ilustracija 2: Frekventni opsezi kartica koje rade na radio talasima

Mane ovog sistema za konkretan problem je potreba za konstantnom izradom kartica za studente, kao i mogućnost da se jedna osoba otkuca različitim karticama.

Sistemi koji uključuju video nadzor se zasnivaju na prepoznavanju lica i zapravo su u praksi se pokazali kao mnogo komplikovaniji za implementaciju i pouzdano korišćenje. Za konkretan slučaj je rađeno testiranje s' obzirom da je ovaj sistem bio

isplativ sa korišćenjem postojećeg sistema video nadzora kao i Open-Source³ biblioteka za detekciju i prepoznavanje lica. U praksi je moguće porediti lice sa slikama u postojećoj bazi, ali bi takođe bilo potrebno napraviti mnogo više od jedne slike da bi sistem pouzdano radio, a slike bi morale imati razne izraze na licu, što nije uvijek moguće realizovati, s' obzirom da bi i pristanak svih osoba bio potreban.

Ostali sistemi uglavnom se koriste za detekciju prisustva bez osvrta na identitet, a to su sistemi lasera i infra crvenih senzora koji samo imaju uvid o stanju da li i koliko osoba ima u dijelu na koji su senzori ugrađeni.

1.4 Idejno rešenje

Kao idejno rešenje u ovom slučaju podrazumijeva se da svaka osoba ima mobilni uređaj koji može uspostaviti WiFi konekciju, a u suprotnom ručno se evidentira putem softvera koji bi se nalazio na uređaju koji je prenosiv. Sami uređaj bi imao ekran osjetljiv na dodir, mogućnost priključivanja na internet putem Ethernet-a, kao i mogućnost dijeljenja veze putem WiFi-ja. Interfejs na uređaju bi nakon njegovog uključivanja imao podešavanja detekcije prisustva kao i dugme za pokretanje, nakon čega bi se mobilni uređaji povezivali na WiFi. Omogućiti korisnicima korišćenje interneta je opciono i prilikom pokretanja se podešava. Prilikom prve konekcije bilo bi potrebno pristupiti web stranici koja se nalazi na samom uređaju i registrovati ime i broj indeksa, nakon čega bi uređaj čuvao podatke i registrovao tog korisnika sa svakom konekcijom. Konkretno to znači da korisnik koji se registruje na tom uređaju, ukoliko je čekirana opcija automatske konekcije na WiFi, ne mora da brine o tome da li je njegovo prisustvo evidentirano, dok korisnik koji prati prisustva ima generisanu tabelu sa brojevima indeksa, imenima, kao i vremenu koje je korisnik proveo na predavanju. Dalje, moguće je čuvati ove podatke na Cloud⁴-u, kao i proširiti funkcionalnosti da uređaj komunicira sa drugim sklopovima i dijeli podatke, sačinjava razne statistike. Prednost ovakvih aplikacija i jeste na prvom mjestu mogućnost daljeg proširenja i prilagođavanja. Kao platforma za potrebe ovog

³ Open-Source: odnosi se na softver koji ima svoj izvorni kod dostupan pod "open-source" licencama, i njihov se kod može mijenjati, prilagođavati i poboljšavati. U zavisnosti od licence moguće je i izmenjeni kod komercijalno distribuirati.

⁴ Cloud: Ideja cloud tehnologije zasniva se na tome da svi podaci koji su neophodni korisniku (bile to aplikacije, dokumenti, hardver, ili nešto drugo) budu dostupni u svakom trenutku, naravno uz preduslov da je prethodno uspostavljena internet veza. Dakle, cloud na neki način predstavlja uslugu dostavljanja servisa umesto samog proizvoda.

projekta odabran je Raspberry Pi, kompjuter koji sadrži sve komponente potrebne za funkcionisanje na jednoj ploči. Sam kompjuter će biti proširen sa ekranom osjetljivim na dodir, dijagonale 3,5 inča i imati zaštitnu kutiju od akrila, providne termo-otporne plastike. Ideja je da uređaj bude prenosiv i da uključivanjem u struju pokrene sistem sa gore pomenutim interfejsom za kontrolu i podešavanje sistema za detekciju prisustva.

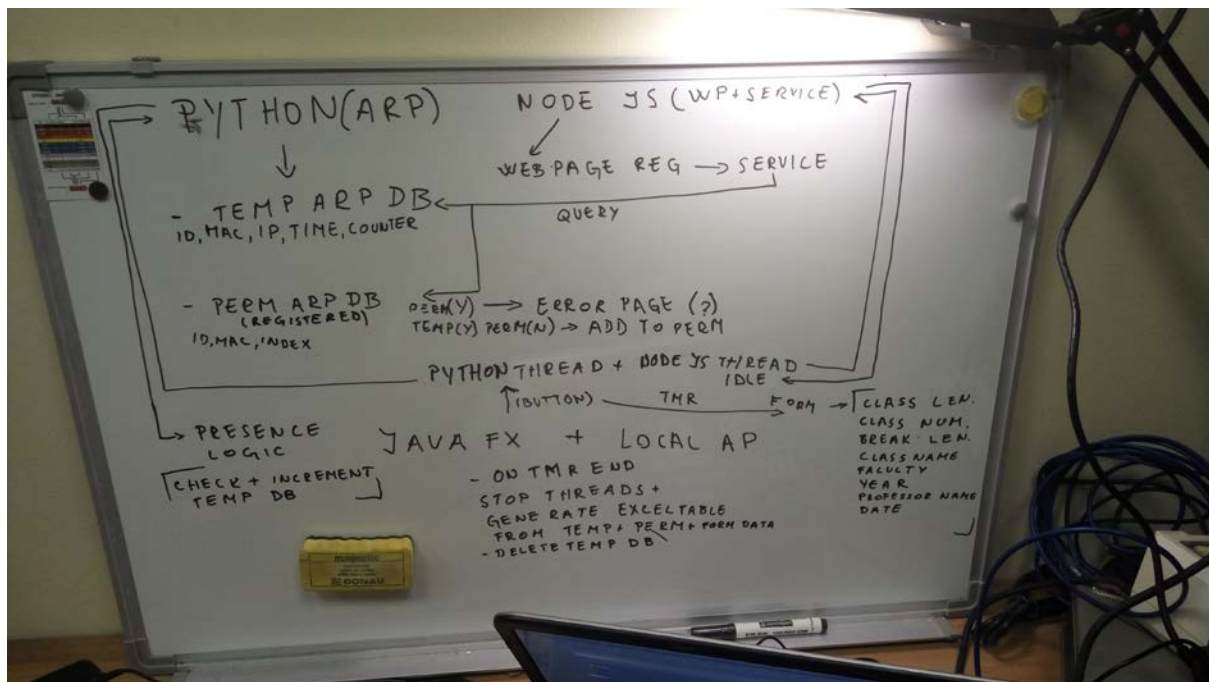
2 Organizacija projekta

Projekat je organizovan tako da sadrži dva dijela za realizaciju, i to hardverski i softverski dio. Hardverski dio je baziran na Raspberry Pi platformi, na kojoj je potrebno podesiti ispravno funkcionisanje Linux operativnog sistema. Kako je trenutna platforma verzije 3 isporučena sa svim potrebnim djelovima za projekat ugrađenim u sam uređaj, nije potrebna instalacija nekih dodatnih drajvera i samog povezivanja hardvera. Kako ovaj projekat na trenutnoj platformi nije komercijalno isplativ, a i sam uređaj ima mnogo više funkcionalnosti nego što je potrebno, moguća je i buduća promjena platforme. Hardverski zahtjevi na osnovu kojih bi se mogla izvršiti selekcija neke druge platforme je:

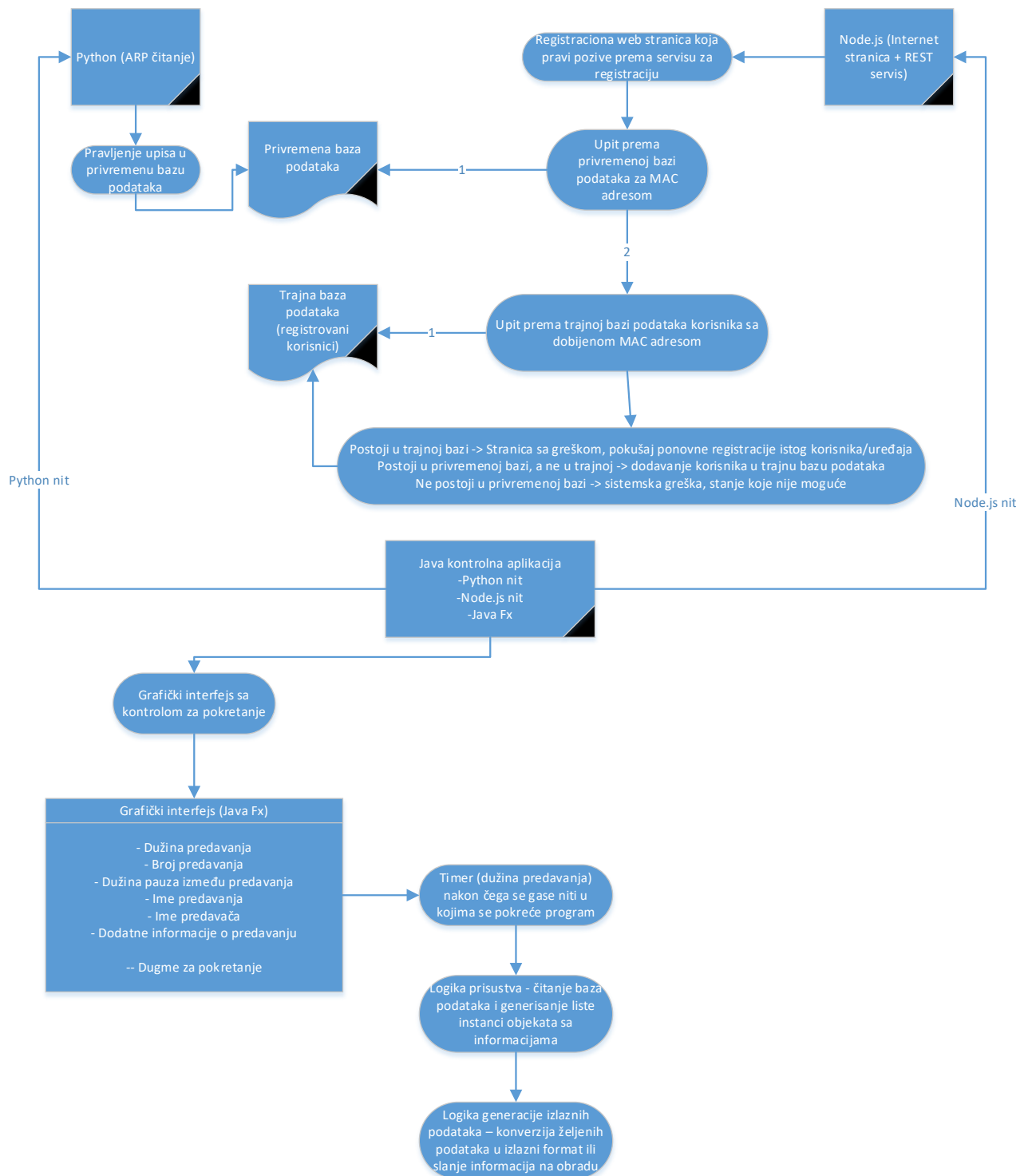
1. Linux platforma
 - a. Postojanje WiFi antene
 - b. Podrška drajvera u hostapd-u vezano za WiFi interfejs koji hardver posjeduje
2. Platforma koja nije bazirana na Linux operativnom sistemu
 - a. Postojanje WiFi antene
 - i. Podrška za čitanje MAC adresa
 - b. Postojanje podrške za hostovanje Web Servera
 - c. Postojanje podrške za kreiranje ljudski čitljivog formata kao što je xls ili csv fajl, ili postojanje drugog interfejsa za slanje podataka na obradu (ethernet, bluetooth)

Na ovaj način bi uz manje izmjene projekat bio primenjiv i u drugim okruženjima, koja se mogu pokazati kao ekonomski isplativija.

Softverska realizacija se zasniva na tome da platforma podržava Java i Python, ali je moguće korišćenje i nekog drugog jezika visokog nivoa. Sama šema projekta se nalazi u nastavku i predstavlja ideju realizacije u ovom slučaju, ali je moguće iskoristiti je i kao idejno rešenje pri korišćenju nekih drugih tehnologija. Sam projekat će biti usklađen na više uređaja putem Git distribuirane kontrole verzije. Git je djelo Linus Torvalda koji je kreator Linux kernela. Kao servis koji pruža git uslugu odabran je GitHub koji nudi privatne programske repozitorijume studentima u vidu Student Developer pakovanja.



Ilustracija 3: slika dijagrama



Ilustracija 4: sistemski grafik

3 Hardverska platforma

Raspberry Pi platforma je doživjela veliki uspjeh na trzistu i posjeduje dosta dokumentacije pa je iz tog razloga veoma lako zapoceti rad sa platformom. Sama platforma je SBC⁵ koja sadrži ploču na kojoj je mikroprocesor, memorija, I/O portovi. Sama platforma ima nekoliko verzija, s' obzirom da nije proširiva sem u pogledu memorijskog prostora putem SD Kartice ili USB memorijskog diska. Konkretno u ovom slučaju je korišćen Raspberry Pi 3, Model B.

3.1 Raspberry Pi 3

Hardverske specifikacije

| | |
|-------------------|-------------------------------|
| SoC | BCM2837 |
| CPU | Quad Core Cortex A53 – 1.2GHz |
| Instrukcijski Set | ARMv8-A |
| GPU | 400MHz VideoCore IV |
| RAM | 1GB SDRAM |
| Storage | Micro-SD |
| Ethernet | 10/100 Mbps |
| Wireless | 802.11n / Bluetooth 4.1 LE |
| Video Output | HDMI / Composite |
| Audio Output | HDMI / 3.5mm audio |
| GPIO | 40 |

3.1.1 Operativni sistemi

Ova platforma moze da pokrene razne operativne Sisteme, i to instalacijom live verzije na micro sd kartici. Tu su Windows 10 IoT Core, Android Things, razne linux distribucije, pored njih je i RISC OS koji je prvenstveno pisan za ARM platformu. Standardno se preporučuje Linux distribucija Raspbian, s' obzirom da ona sadrži sve potrebne drajvere i prilagođen je baš Raspberry Pi platformi. U ovoj distribuciji se nalazi mnoštvo softvera za sam početak razvoja na ovoj platformi, kao što su: Wolfram, Mathematica, Python, BlueJ, Node-Red, Scratch. U suštini se za kontrolu GPIO pinova

⁵ SBC: Single Board Computer

koristi Python, ali je moguće koristiti i druge programske jezike. Ako se koristi neka od linux distribucija, bitno je to da linux nije Real Time Operating System pa se ne može koristiti za razvoj projekata kod kojih je vrijeme izvršavanja najbitnija stvar. Kako se kod linux operativnih Sistema sva interakcija sa hardverom dešava na kernelskom nivou, dok se aplikacije pokreću u korisničkom prostoru i svaka je aplikacija u suštini proces koji dobija procesorsko vrijeme, to znači da procesor određuje kada će i koliko prioriteta dodijeliti određenim priključenim uređajima. Kod projekata koji zahtijevaju velike vremenske tačnosti moguće je napisati prekidne rutine koje u određenim momentima vrše drugu logiku koja u tom trenutku ima veći prioritet, a kod linux kernel to nije moguće jer kernel odlučuje o dodjeli procesorskog vremena. U praksi postoje operativni sistemi koji su usmjereni ka tome da linux bude RTOS, kao što su RTLinux i QNX. U slučaju projekta na kom će se raditi vrijeme nije krucijalan faktor, pa je odabrana platforma sasvim dovoljna za ovu primjenu. Kašnjenja koja su uzrokovana ovom vrstom nedostatka operativnog Sistema, nisu od velike važnosti s' obzirom da će samo ispitivanje korisnika da se vrši u vremenskom razmaku od deset i više sekundi.

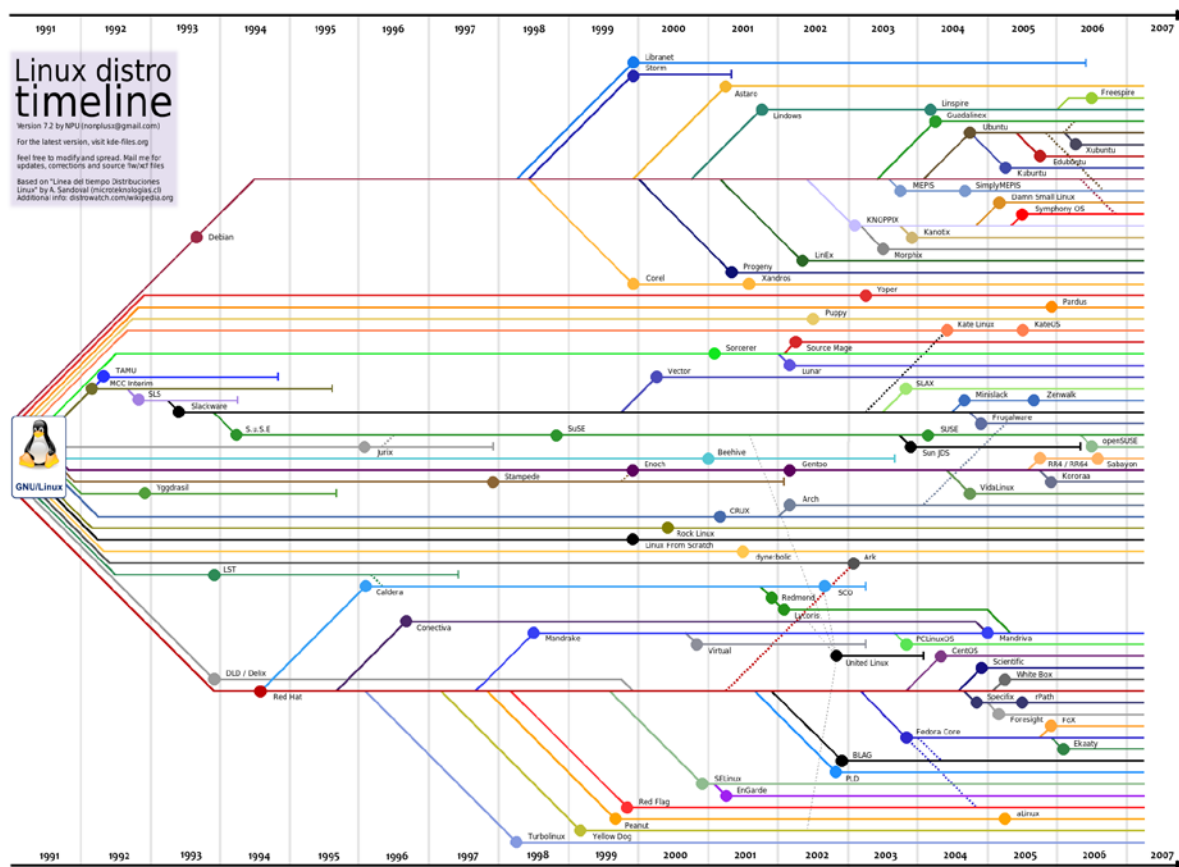
3.1.1.1 *Raspbian (Jessie)*

U trenutku pisanja aktuelna verzija operativnog sistema je Jessie. Raspbian je baziran na Debian operativnom sistemu, koji je najrasprostranjeniji u Linux svijetu. Jedna od odlika jeste da je dobro podržan i redovno ima ispravke i nadogradnje. Debian je volonterska organizacija koja se zasniva na: Debian Social Contract, Debian Free Software Guidelines i Debian Constitution, prema kojima developeri rade na softveru. Debian je podržan donacijama, dok je sama struktura odlučivanja u organizaciji poznata i lider projekta se bira jednom godišnje. Prilikom razvoja prate se standardi za kvalitet pisanja koda, a u cilju razvoja novih funkcionalnosti ili ispravki postojećih bug-ova. Nakon verifikacije i isporuke novog paketa, on se instalira u takozvani "pool" koji je kasnije distribuiran korisnicima širom svijeta. Po pitanju sigurnosti politika Debian-a je javno objavljivanje i rad na poboljšanjima i zakrpama. Raspbian kao package manager koristi DPKG⁶ i APT⁷ kao njegov front end. Grafičko okruženje u Raspbianu je

⁶ DPKG: Debian Package Manager

⁷ APT: Advanced Package Manager

nazvano.PIXEL (Pi Improved Xwindow Environment, Lightweight) koji je napravljen da približi korišćenje linux operativnog sistema ljudima koji se sa njim nisu do sada susretali.

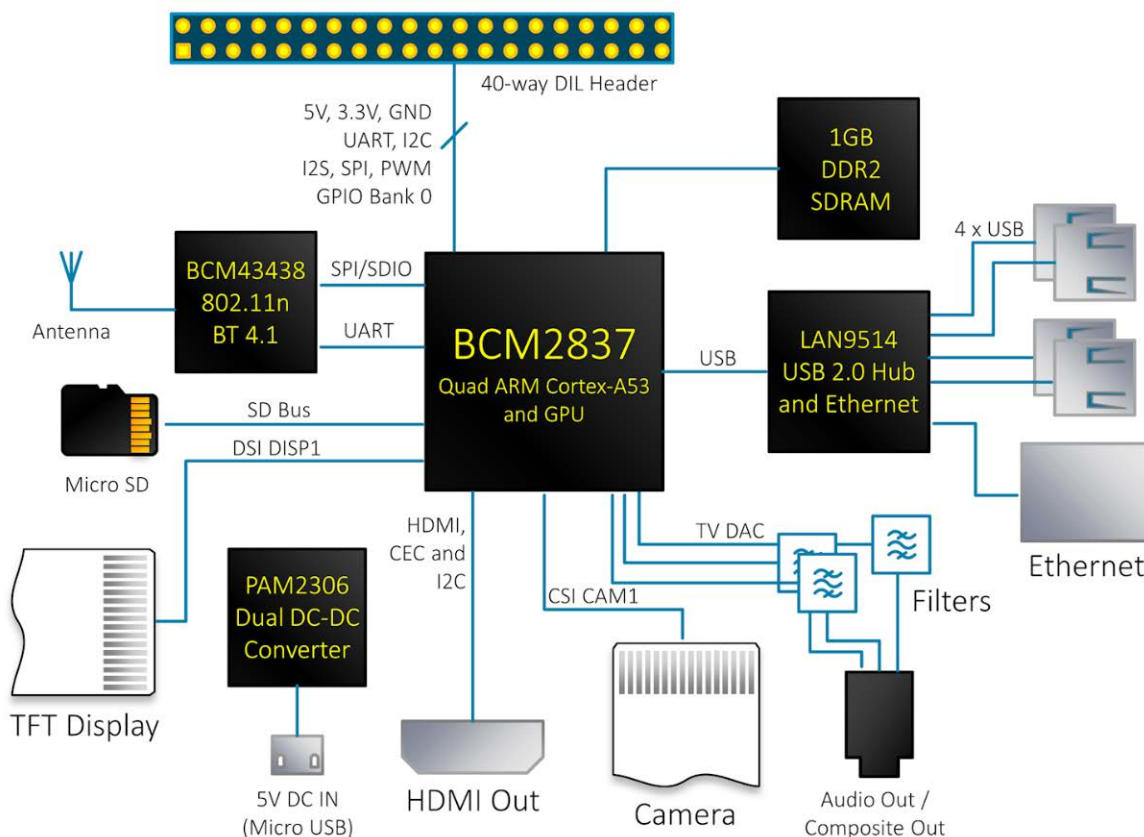


Ilustracija 5: Vremenski prikaz razvoja Linux distribucija

3.1.1.2 Način instalacije

Instalacija operativnog sistema je procedura koja podrazumijeva odabir operativnog sistema i preuzimanje, a zatim pravljenje live sd kartice putem nekog od

softverskih paketa kao što su Rufus ili Etcher, mada postoji i veliki broj drugih. Nakon toga odabir .img ili .iso u softveru kao i odabir kartice na koju je potrebno instalirati OS.



Ilustracija 6: Raspberry 3 blok dijagram

3.1.2 SoC

System on Chip je integrirano kolo koje može da sadrži procesorsku jedinicu, memoriju, grafičku jedinicu, WiFi, Bluetooth i slično. BCM2837 sadrži u sebi procesor sa 512 KB dijeljene L2 keš memorije, grafičku jedinicu (VideoCore IV).

3.1.2.1 Keš memorija

Po memorijskoj hijerarhiji keš memorija je prva nakon procesorskih registara po brzini memorije, ali takođe i najmanjeg kapaciteta. Njena efikasnost se zasniva na principu ne tako čestog prenošenja podataka u bržu mrežu i višestrukog pristupa istim podacima prije zamjene sa novim. Ovaj princip je moguć zbog lokalnosti referenci, koje mogu biti vremenske i prostorne. Prostorna lokalnost znači da je velika vjerovatnoća ako se pristupilo jednoj memorijskoj lokaciji, da će se pristupiti i nekoj okolnoj lokaciji.

Vremenska lokalnost znači da ako se pristupilo nekoj memorijskoj lokaciji, postoji vjerovatnoća da će se opet pristupiti toj lokaciji, što je slučaj kod petlji. Procesor pri zahtjevu za nekom memorijskom lokacijom generiše njegovu adresu i ona se može nalaziti u Keš memoriji. Ukoliko to nije slučaj, podatak može biti u operativnoj memoriji, a u krajnjem slučaju može se nalaziti na sekundarnoj memoriji (npr. Hard disk). Kao integrisano kolo ili kao dio procesora, jedinica za upravljanje memorijom ili MMU- Memory Management Unit bavi se mapiranjem adresa. Ono što MMU radi jeste da vodi evidenciju o tome koji su djelovi operativne memorije trenutno smješteni u keš memoriji, o tome koji blok je potrebno izbaciti iz pune keš memorije, kao i ažuriranje. sadržaja operativne memorije kada je bilo upisa na lokacijama u keš memoriji, s' obzirom da se u tom trenutku sadržaji na istoj memorijskoj lokaciji razlikuju. Tehnike preslikavanja:

- Direktno preslikavanje

Ovo je najjednostavnija tehnika preslikavanja koja smješta dolazeći blok iz operativne memorije na fiksnu lokaciju u keš memoriji. U ovom slučaju važi fiksna relacija $j = i * \text{mod}(NCB)$ gdje je:

i - broj dolazećeg bloka iz operativne memorije,

j - broj bloka u keš memoriji,

a NCB ukupan broj blokova.

Keš memorija se sastoji od tag memorije i data memorije. Ako operativna memorija sadrži 4k blokova, keš memorija 128 blokova i veličina bloka je 16 memorijskih riječi onda važi da MMU upisuje memoriju tako što je dijeli na tri polja: polje taga(T), polje keš bloka(CB) i polje riječi (W)

$$T = \log_2\left(\frac{NMB}{NCB}\right); CB = \log_2 NCB; W = \log_2 B; A = \log_2(B * NMB)$$

Pa tako dobijamo da je T=5 bitova, CB = 7 bitova I W=4 bita, dok je A=16 bitova

| | | |
|--|------|-----|
| T 5 | CB 7 | W 4 |
| A 16 bitna adresa u operativnoj memoriji | | |

Pomoću CB polja se određuje blok u keš memoriji, a nakon toga se poređenjem polja TM sa T poljem zaključuje da li je element u keš memoriji ili nije.

Ukoliko jeste traži se unutar bloka putem W polja, u suprotnom se prenosi iz operativne memorije u keš memoriju.

Tehnike preslikavanja:

- Asocijativno preslikavanje

Ovaj način preslikavanja je fleksibilniji jer se blok iz operativne memorije može smjestiti u bilo koji raspoloživi blok keš memorije. Adresa koja se generiše ima 2 polja za razliku od direktnog preslikavanja.

$$T = \log_2 NMB; W = \log_2 B; A = \log_2(B * NMB) = T + W$$

Ako koristimo iste podatke kao i za direktno preslikavanje onda imamo da je:

| | |
|--|-----|
| T 12 | W 4 |
| A 16 bitna adresa u operativnoj memoriji | |

Pristupanje elementa se vrši tako što se vrijednost polja T nađe među postojećim tagovima u TM, ako T postoji, element je u odgovarajućem bloku pa se element unutar bloka traži pomoću W, u suprotno prenosi se blok iz op. memorije.

- Set-Asocijativno preslikavanje

Kod ove vrste preslikavanja memorija je podijeljena u skupove pri čemu svaki skup ima određen broj blokova. Blok se iz operativne memorije mapira u jedan set keš memorije prema formuli: $s = i * \text{mod}(NS)$ gdje je:

i - broj blokova operativne memorije,

NS - broj setova u keš memoriji,

s – broj seta u keš memoriji koji se mapira. Adresa koju ova metoda generiše ima tri polja, a računaju se po sledećim formulama:

$$T = \log_2 \left(\frac{NMB * BS}{NCB} \right); S = \log_2 NS; W = \log_2 B; A = \log_2(B * NMB)$$

Gdje je BS broj blokova u setu. Koristeći podatke iz prethodnih primjera dobijamo:

| | | |
|--|-----|-----|
| T 7 | S 5 | W 4 |
| A 16 bitna adresa u operativnoj memoriji | | |

Na osnovu polja S se određuje set u koji se mapira i ako se T nalazi među postojećim tagovima u TM za dati set onda se pomoću W polja pronalazi blok, a u suprotnom se prenosi iz operativne memorije.

Tehnike zamjene:

- Random Selection

Koristi se generator slučajnih brojeva koji generiše brojeve od 0 do N-1 gdje je N broj blokova keš memorije, tako da se odabrani nasumični blok u trenutku promjene briše i dovodi blok iz operativne memorije.

- FiFo: First In First Out

Ovom tehnikom se iz keš memorije izbacuje blok koji je najviše vremena proveo u keš memoriji.

- LRU: Last Recently Used

Ova tehnika se zasniva na izbacivanju bloka koji je najranije poslednji put korišten, ali zahtijeva i upotrebu keš kontrolera koji čuva istoriju referenci na sve blokove.

Tehnike ažuriranja:

- Tehnika upisa ako blok postoji u keš memoriji

- Write trough

Svakom operacijom upisa se vrši upis u operativnu i keš memoriju

- Write back

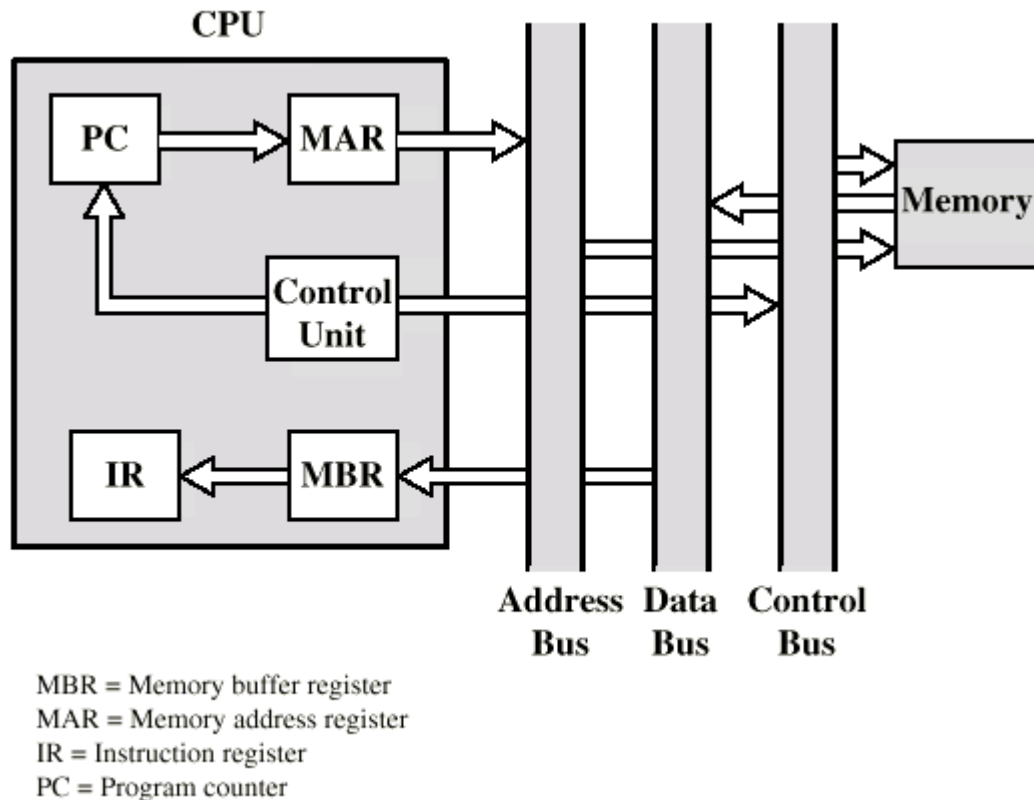
Upis se radi samo u keš memoriji, dok se upis u operativnu memoriju odlaže sve do trenutka kada blok treba da bude zamijenjen drugim blokom.

Svaki blok u keš memoriji ima dirty bit koji je indikator na to da li je došlo do promjena na bloku u keš memoriji.

- Tehnika upisa ako blok ne postoji u keš memoriji
 - Write allocate
Blok se prebacuje iz operativne u keš memoriju, a onda se postupa kao u prethodnoj tehnici
 - Write no allocate
Blok se ne prebacuje, već se direktno vrši upis u operativnu memoriju.
- Tehnika čitanja ako blok postoji u keš memoriji – pročitana se vrijednost
- Tehnika čitanja ako blok ne postoji u keš memoriji
 - Direktno prosleđivanje
Blok se prebacuje iz operativne u keš memoriju a odmah po pristizanju se prosleđuje procesoru
 - Prosleđivanje sa zadržkom
Blok se prebacuje iz operativne u keš memoriju i tek kad se u potpunosti prebaci podatak se prosleđuje procesoru.

3.1.2.2 Procesor

Kao centralni dio svakog računara, dio koji izvršava sve kalkulacije u kompjuteru je procesor. Struktura procesora sadrži centralne komponente i registarske komponente. Pod centralnim komponentama podrazumijeva se Aritmeticko-logicka jedinica i ona služi da izvršava instrukcije programa. Registarske komponente omogućavaju rad ALU i to su: programski brojač koji sadrži adresu sledeće instrukcije (Program Counter), adresni registar memorije (Memory Address Register) koji sadrži adresu memorijske lokacije kojoj treba pristupiti, privatni registar podataka (Memory Data Register, takođe i Memory Buffer Register) koji sadrži pročitani podatak sa memorijske lokacije ili podatak koji treba da bude upisan, privatni registar instrukcije (Instruction Register) koji sadrži instrukciju, privatni registar izvorišnih operandata koji su vezani na ulazne linije ALU, privatni registar rezultata koji je vezan na izlaz iz ALU.



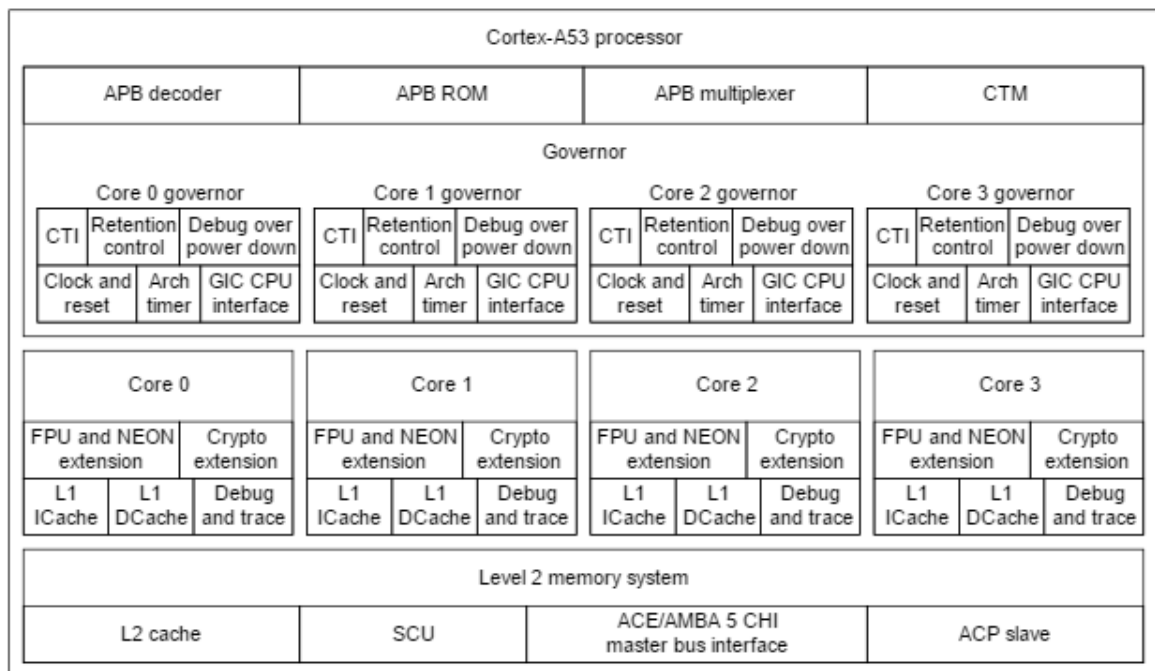
Ilustracija 7: Tok jednog ciklusa u procesoru, tokom preuzimanja podataka

Kako su procesorski registri mnogo brži od sledeće najbrže memorije, a to je Cache memorija koja može postojati u više nivoa, tj brzina i kapaciteta, procesor koristi algoritme za popunjavanje ove memorije. Ti algoritmi se baziraju na pogađanju instrukcija i podataka koji će biti potrebni procesoru u trenutku opsluživanja određenog toka. Prema nekim podacima današnji procesori pogađaju oko 80% potrebnih stvari, pa prema tome nema potrebe da se obraćaju sporijim memorijama u trenutku kada se izvršava neka lista instrukcija.

3.1.2.2.1 ARM Cortex-A53

U slučaju Raspberry Pi platforme u pitanju je Cortex A53 procesor koji ima četiri jezgra, koji ima cache memoriju podijeljenu u 2 nivoa. Na prvom nivou se nalazi 32kB memorije raspoređene na 4 jezgra kao instrukcijski keš i keš podataka. Na drugom nivou se nalazi 512kB koji je dijeljen između svih jezgara. Procesor je 64 bitni, što znači da je širina instrukcija i memorijskih lokacija sa kojima procesor može da radi 2^{64} pa time doprinosi većem broju instrukcija (kada se pređe na 64bitne instrukcije, s' obzirom da ARM i dalje koristi instrukcije dužine 32 bita),

a i samoj optimizaciji rada procesora, što se oslikava u bržem radu procesora, kao i adresiranju većeg broja RAM memorijskih lokacija.



Ilustracija 8: ARM Cortex-A53 dijagram

3.1.2.3 Instrukcijski setovi

Instrukcijski set je skup instrukcija kojim se specificiraju operacije koje procesor može da izvrši. Tipovi instrukcija su: aritmetičke, logičke, pomjeračke, instrukcije prenosa i instrukcije skoka. Instrukcijski setovi se mogu razlikovati i po broju operanada koji učestvuju u instrukcijama kao parametri. Pa tako na primer mašine koje rade sa 0 adresa (stek mašine), sa 1 adresom (akumulatorska mašina), 2 i 3 adrese (CISC i RISC mašine). Cortex A53 je Reduced Instruction Set Computer, iako je u poslednjoj verziji ARMv8-A arhitekture uveden veliki broj kriptografskih i atomskih funkcija za čitanje i pisanje. Primjer broja operanada kod instrukcija u instrukcijsom setu je:

3-adresni: prvi i drugi parametar su izvorišne adrese dok je treći odredišna adresa

2-adresni: prvi i drugi parametar su izvorišne adrese, a ujedno i jedan od njih je odredišna adresa

1-adresni: akumulator izvor i odredište

0-adresni: sa steka se čita i u stek se upisuje rezultat.

Kao aritmetičke operacije podrazumijevaju se sabiranje, množenje, dijeljenje, oduzimanje. Pod logičkim instrukcijama spadaju I, ILI, Ekskluzivno ILI, NE, kao i pomjeračke instrukcije koje pomjeraju binarne riječi za jedno mjesto ulijevo ili udesno. Instrukcije prenosa služe za prenos podataka sa jednog mjesta na drugo, i to: u memorijskim lokacijama, procesorskim registrima, u instrukciji kao neposredna veličina, u registrima kontrolera periferija, u akumulatoru, na stek. Takođe postoje instrukcije skoka, koji mogu biti uslovni i bezuslovni i omogućavaju uslovna izvršavanja instrukcija.

3.1.3 RAM

SoC na poledini sadrži ELPIDA B8132B4PB LPDDR2 SDRAM memoriju, ovakav način montiranja memorije na SoC se naziva i Package On Package.

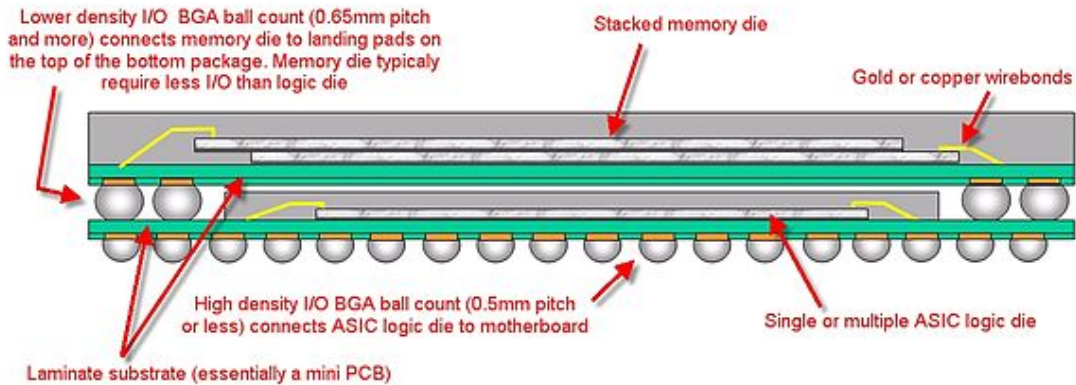
3.1.3.1 Tipovi RAM Memorije

- SRAM: Static Random Access
- DRAM: Dynamic Random Access
- FPM DRAM: Fast Page Mode DRAM
- EDO DRAM: Extended data-out DRAM
- SDRAM: Synchronous Dynamic Random Access
- DDR SDRAM: Double data rate synchronous dynamic RAM
- RDRAM: Rambus dynamic random access memory
- CMOS RAM: Complementary metal–oxide–semiconductor RAM
- VRAM: Video RAM ili MPDRAM - multiport dynamic random access memory

U ovom slučaju se radi o DDR SDRAM memoriji, tj. o Low Power verziji memorije koja radi na 400Mhz, što je jako bitno zbog potrošnje energije kao i zbog manjeg rasipanja energije u vidu toplote.

3.1.3.2 Package on package

Ovo je metoda pakovanja integrisanih kola da bi se kombinovala logička i memorijska kola. Koristi se matrica kontaktnih tačkica (prethodno pinovi), tako da se montiranjem dobija stalna veza između štampane ploče i kola. Postoje dvije konfiguracije i to: samo memorijsko vezivanje i memorijsko logičko vezivanje, tj. procesor i memorija, što je ovdje slučaj.



Ilustracija 9: Logičko-memorijsko slaganje kola

Ovim se vrši ušteda mjesta na štampanoj ploči, pa time je i izvodljivo da se dobije ovakav uređaj na tako malim dimenzijama.

3.1.4 I/O and LAN

Na ploči se takođe nalaze SMSC LAN9514 kontroler koji je zadužen za kontrolu Ethernet-a i USB Hub-a u jednom čipu.

3.1.4.1 Hub

Pod pojmom Hub-a se podrazumijeva funkcionisanje tako da sve što se dobije na jednom portu na hub-u emituje na svim ostalim portovima. Sam USB hub funkcionise tako da ima jedan “upstream” port i više “downstream” portova, pa se ono što je poslato sa upstream porta emituje svim uređajima, ali podaci sa downstream porta se šalju samo host-u. Na ovaj način su realizovana 4 USB porta na ploči.

3.1.4.2 NIC

Network Interface Controller je hardverska komponenta zadužena za povezivanje kompjutera na mrežu, u ovom slučaju putem Ethernet porta. Brzina porta je 10/100 Mbps, što je standard koji podržava konekciju do 100Mbps, ali je komplementaran i sa starijim uređajima koji mogu da rade samo do 10Mbps.

3.1.5 WiFi/Bluetooth

Na ploči se nalazi Broadcom BCM43438 modul za Wifi i Bluetooth.

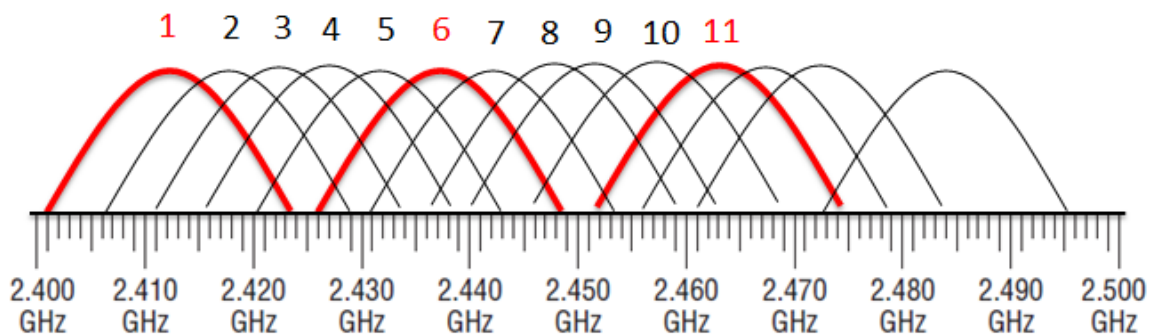
3.1.5.1 WiFi

Modul radi pod IEEE 802.11 standardom koji je set specifikacija vezanih za MAC i fizički nivo kod implementacije Wireless lokalnih mreža (WLAN) i omogućava komunikaciju na 900MHz, 2,4GHz, 3,6GHz, 5GHz i 60GHz. Konkretno u ovom slučaju

omogućena je komunikacija na 2,4GHz. Prednost manjih frekvencija jeste prenos podataka na veće fizičke udaljenosti, ali pri manjim brzinama u odnosu na veće frekvencije prenosa. Raspon frekvencija raspoređen je po standardu od 2.400GHz do 2.500GHz u 14 kanala širine 22Mhz, prema sledećoj tabeli:

| BROJ KANALA | DONJA FREKVENCIJSKA GRANICA | CENTRALNA FREKVENCIJA | GORNJA FREKVENCIJSKA GRANICA |
|-------------|-----------------------------|-----------------------|------------------------------|
| 1 | 2401 | 2412 | 2423 |
| 2 | 2406 | 2417 | 2428 |
| 3 | 2411 | 2422 | 2433 |
| 4 | 2416 | 2427 | 2438 |
| 5 | 2421 | 2432 | 2443 |
| 6 | 2426 | 2437 | 2448 |
| 7 | 2431 | 2442 | 2453 |
| 8 | 2436 | 2447 | 2458 |
| 9 | 2441 | 2452 | 2463 |
| 10 | 2446 | 2457 | 2468 |
| 11 | 2451 | 2462 | 2473 |
| 12 | 2456 | 2467 | 2478 |
| 13 | 2461 | 2472 | 2483 |
| 14 | 2473 | 2484 | 2495 |

Ovdje se može vidjeti da su samo kanali 1, 6 i 11 bez preklapanja i dobro je birati onaj kanal koji će imati najmanje preklapanja sa ostalim kanalima zbog performansi same veze. Ovo je slučaj u 802.11b i g standardu dok n standard nudi i širinu kanala od 40MHz, koji se u praksi nije pokazao kao najbolje rešenje zbog broja uređaja u našoj okolini ukoliko se ne koristi 5GHz mreža.



Ilustracija 10: Raspored kanala prema 802.11b/g standardu

3.1.5.2 Bluetooth

Bluetooth je tehnologija koja radi na frekvencijskom rasponu od 2.4 do 2.485 GHz. Kako je to jako čest frekvencijski opseg koji dijeli sa mnogim drugim uređajima, koristi se princip Frequency Hopping Spread Spectrum koji Bluetooth kanal od 1MHz mijenja 1600 puta u sekundi kroz neki od 79 kanala i to po predefinisanim obrascima kojih ima šest. Ukoliko postoji uređaj koji na jednoj određenoj frekvenciji emituje signal i time ometa signal na tom opsegu, to su vrlo mali gubitci, dok je takođe vrlo mala šansa da će se naći još jedan uređaj koji koristi istu sekvencu frekvencijskog skakanja da je sinhronizovana baš tako da se poklopi sa onom na uređaju.

Bluetooth 4.0 standard je donijeo novine kod ove tehnologije u smislu implementacije u uređajima koji koriste jako malo struje, pa je tako reklamiran pod Bluetooth Low Energy, dok 4.1 treba da donese povezivanje kod IoT-a, imajući u vidu broj uređaja koji će u budućnosti sve više da raste. Bluetooth je eliminisao neke probleme koje je imao prilikom rada uporedo sa 4g tehnologijom tako što je koordinisan sa tom tehnologijom i sada obje mogu da rade punom brzinom. Takođe je uklonjen fiksni timeout prilikom ponovnog povezivanja, tako da se prilikom osmišljavanja uređaja može bolje voditi računa o tome kako uređaj koristi energiju. Takođe podržana je konekcija ka više uređaja odjednom, od kojih samo jedan može aktivno komunicirati sa host uređajem.

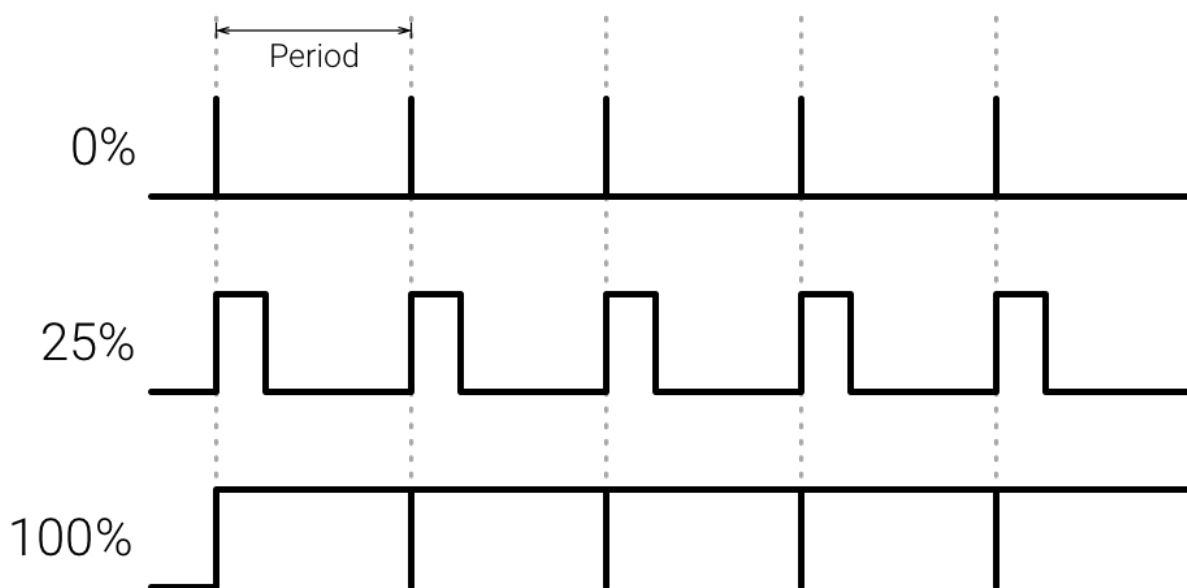
3.1.6 Napajanje

Na ploči se nalazi i PAM2306 Dual Step-Down konvertor, s' obzirom da mnoge komponente rade na 3.3v dok je ulaz na ploči 5v. Ovaj konvertor omogućava konstantnu

isporuku stabilnog napona zbog konstantne PWM kontrole, a takođe podržava i PSM mod koji smanjuje potrošnju prilikom lakog korišćenja uređaja.

3.1.6.1 PWM

Pulse Width Modulation je tehnika enkodiranja poruke u pulsni signal, ali iako se može koristiti za prenos informacije, glavna uloga je kontrola struje isporučene uređajima. Kao kontrolni signal on ima svoju frekvenciju, kao i procenat isporučenog pozitivnog signala po pulsu. Frekvencija određuje koliko će biti česta perioda uključenja signala, dok će količina uključenog signala u toj jednoj periodi dati isporuku samo dijela maksimalne struje.



Ilustracija 11: PWM Primjer

3.1.6.2 PSM

U napajanjima koja mijenjaju napon jako je bitno da bude što manje gubitaka, pa se u slučajevima kada potrošnja nije velika koriste razne metode za uštedu energije. Pulse skip modulation je jedna od njih i ona podrazumijeva da se kontrolni signal na određenim pulsevima uopšte ne šalje i tako uštedi energija, a ne uslovljava odstupanjima u izlaznom naponu kako potrošnja nije velika. Tako je efikasnost ovog DC-DC modula oko 96%.

3.1.7 GPIO

3.2 EKRAN

Raspberry pi podržava više načina povezivanja ekrana, i to: HDMI⁸ (koji se konvertorima može pretvoriti i u popularne Display Port ili VGA), DSI⁹, GPIO pinovima na Raspberry ploči i različitim vrstama interfejsa. S obzirom na to da HDMI ne bi mogao da podrži touch screen funkciju, poređenjem cijena ekrana, kao najisplativije rešenje po jedinici je bio touch ekran dijagonale 3.5 inča sa SPI interfejsom i XPT2046 kontrolerom ekrana na ploči. Rezolucija samog ekrana je 480x320 piksela. Brzina SPI interfejsa je na 32MHz, dok je povezivanje pinova moguće vidjeti na sledećoj tabeli:

| PIN NO. | SYMBOL | DESCRIPTION |
|--------------------------------|------------------|---|
| 1, 17 | 3.3V | Power positive (3.3V power input) |
| 2, 4 | 5V | Power positive (5V power input) |
| 3, 5, 7, 8, 10, 12, 13, 15, 16 | NC | NC |
| 6, 9, 14, 20, 25 | GND | Ground |
| 11 | TP_IRQ | Touch Panel interrupt, low level while the Touch Panel detects touching |
| 18 | LCD_RS | Instruction/Data Register selection |
| 19 | LCD_SI / TP_SI | SPI data input of LCD/Touch Panel |
| 21 | TP_SO | SPI data output of Touch Panel |
| 22 | RST | Reset |
| 23 | LCD_SCK / TP_SCK | SPI clock of LCD/Touch Panel |
| 24 | LCD_CS | LCD chip selection, low active |
| 26 | TP_CS | Touch Panel chip selection, low active |

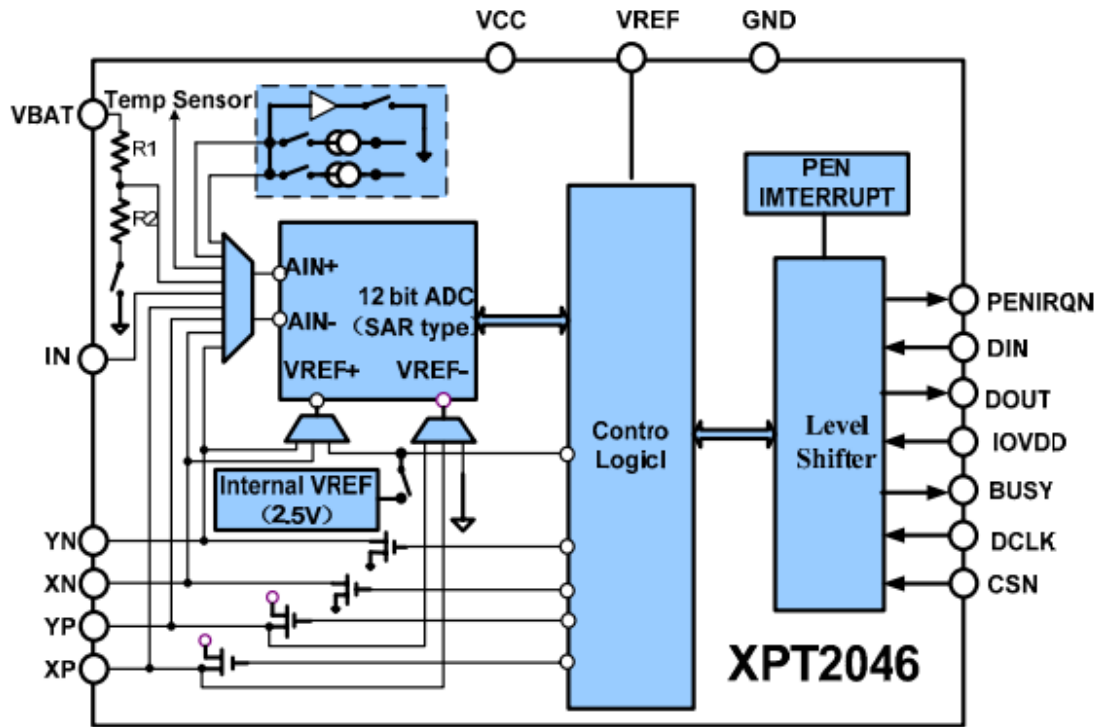
Ovaj način povezivanja nam ostavlja i neke pinove na ploči slobodne, ali za potrebe ovog projekta to neće biti potrebno.

3.2.1 XPT2046

Na samoj ploči displeja se nalazi kontrolna ploča sa pomenutim kontrolerom koji ima 12bitni ad konverter, sa njim je moguće mjeriti i jačinu pritiska, kao i temperaturu na čipu, što svakako zna biti zgodno kada bi se ovaj uređaj našao u masovnoj proizvodnji.

⁸ HDMI – High-Definition Multimedia Interface ustanovljen od strane HDMI Forum koji čini 7 vodećih kompanija na tržištu i služi za prenos video i audio signala.

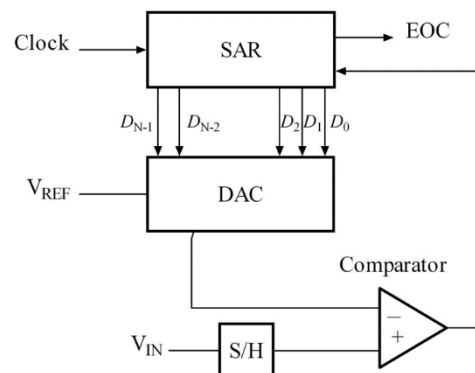
⁹ DSI – Digital Serial Interface ustanovljen od strane MIPI Alijanse u cilju smanjenja troškova mobilnih uređaja. Definiše serijski bus i protokol za prenos podataka.



Ilustracija 12: blok dijagram kontrolera

Kontroler radi na bazi SAR – Sukcesivno aproksimativnog registra koji vrši analognu digitalnu konverziju. Kod ovakve vrste konverzije nailazimo na 4 glavna dijela kola:

- Sample and Hold kolo
- Komparator napona
- Pomoćno kolo SAR-a koje daje predviđenu digitalnu vrijednost DAC uređaju
- DAC interno referencirani za komparaciju reference i izlaznog SAR signala



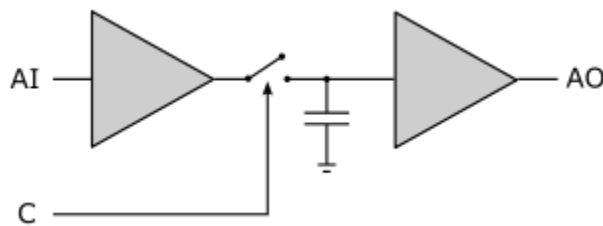
Ilustracija 13: Blok dijagram ADC-a sa sukcesivnom aproksimacijom

3.2.1.1 SAR

Kod ove konverzije dolazi do promjene kontinualnog analognog signala u diskretni digitalni signal, tj. njegovu predstavu binarnom pretragom kroz sve kvantizacione nivoe dok se ne poklopi sa analognim signalom, a da greška bude u rasponu manjem od rezolucije samog sistema.

3.2.1.2 Sample and hold

Ovo je kolo koje služi da bi kontinualni promenjivi analogni signal usrednjilo i zadržalo za neki period vremena, kako bi se samnjile promjene kod analogno digitalne konverzije i spriječilo uništavanje signala kod konverzije primjenom sukcesivne aproksimacije. Tako se signal tj. napon koji se nalazio na ulazu u ad konvertor prilikom početka konverzije zadržava i ostaje isti tokom periode konverzije. Nakon toga se kontrolnim signalom ponovo omogućava proticanje napona. Prilikom stanja kada je prekidač zatvoren i propušta napon se puni i kondenzator koji je na ulazu u pojačivač, a prilikom otvaranja prekidača napon u trenutku njegovog otvaranja ostaje na naponu na kondenzatoru.



Ilustracija 14: Blok dijagram Sample and Hold kola.

4 Softverska implementacija

4.1 Konkurentna logika izvršavanja

U programiranju izvršavanje programa može biti u jednoj ili više niti i ovo je samo jedna od podjela. Kod konkurentnog programiranja imamo više zadataka koji se preklapaju u izvršavanju i mogu a ne moraju biti paralelni. Oni pristupaju deljenim resursima, a takođe mogu i komunicirati između sebe, dok su sinhronizovani uslovno (kada postoji uslov koji treba biti zadovoljen da bi se neka nit izvršila) i ekskluzivno (kada je neophodno da se neki kritični dio niti izvrši prije nego što se pristupi izvršavanju drugih djelova).

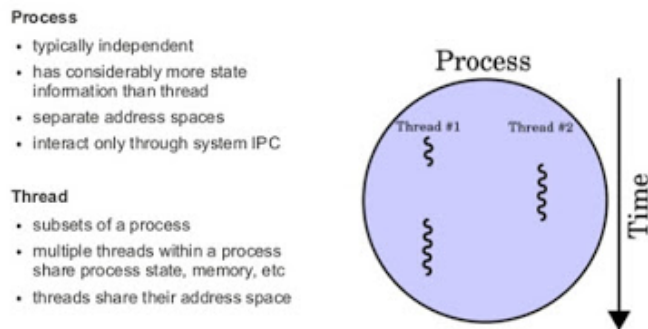
U našem slučaju bitno je da se program izvršava na ovaj način da bi se izbjegli problemi, jer se rešenje sastoji od nadgledanja ARP tabela – koja sadrži MAC adrese svih korisnika koje opslužuje Access Point, Registracije korisnika na web strani – što znači da može biti i više desetina zahtjeva za web stranicom ili registracijom u određenom trenutku, kao i upisa i procesiranja prikupljenih informacija u baze podataka.

4.1.1 Threads and processes in Java

U Java programskom jeziku moguće je izvršavati zadatke istovremeno na dva načina. Jedan je putem procesa dok je drugi putem niti (Thread). Glavne razlike su:

- Niti dijele memorijski prostor i prostor za adresiranje dok procesi imaju svoj zaseban prostor.
- Niti imaju direktan pristup segmentu podataka procesa koji ih je pokrenuo, dok procesi imaju kopiju segmenta podataka.
- Ukoliko postoji više niti u jednom procesu, te niti mogu komunicirati direktno. Kod procesa se koristi inter-procesna komunikacija između povezanih procesa.
- Komunikacija između niti kao i njihovo generisanje je relativno procesorski “jeftino”, dok je pokretanje jako memorijski i procesorski “skupo”.
- Niti imaju određenu mogućnost za određenom kontrolom drugih niti u istom procesu dok procesi mogu samo upravljati decom procesima, tj onima koji su proistekli iz njih samih. (novi proces se generiše duplikacijom postojećeg procesa)
- Promjene na glavnoj niti mogu uticati na ostale niti istog procesa, dok to ne važi za same procese.

Process v. Thread



Ilustracija 15: Poređenje procesa i niti

S' obzirom da u našem programu u svakom trenutku znamo koliko će nam niti biti potrebno da bi se program izvršavao koristi se

```
ExecutorService executor = Executors.newFixedThreadPool(2);
```

koji obezbeđuje tačan broj niti za korišćenje.

Problemi do kojih može doći prilikom odvijanja programa u nitima jesu vezani za pristup promenljivim koje ako nisu deklarisanе kao volatile, što može značiti da varijabla na kojoj je izvršena promjena od strane jedne niti ne mora biti vidljiva drugoj niti. Takođe pristup objektima može biti problem kod slučaja gdje želimo da program pristupa istoj instanci objekta (implementacija singleton design pattern-a) tako da u isto vrijeme pristupe objektu 2 niti i naprave 2 instance istog objekta, a ovo može dovesti do daljih komplikacija posebno kada se radi o objektima za pristup bazama podataka.

4.1.2 Primer razlike u implementaciji singleton obrasca

Pri pravljenju objekata možemo se susreti sa tehnikama poput “lazy instantiation” što se može vidjeti na primjeru lazy singletona, a to znači da se kreiranje objekta odlaže do onog trenutka kada nam je potreban, tj do trenutka kada pozovemo `getInstance()` nad `ClassicSingleton` klasom.

Standardna implementacija Singleton-a

```
public class ClassicSingleton {
    private static ClassicSingleton instance = new ClassicSingleton();
    protected ClassicSingleton() {
        // Hidden constructor.
    }
    public static ClassicSingleton getInstance() {
        return instance;
    }
}
```

Lazy Singleton

```
public class ClassicSingleton {
    private static ClassicSingleton instance = null;
    protected ClassicSingleton() {
        // Hidden constructor.
    }
    public static ClassicSingleton getInstance() {
        if(instance == null) {
            instance = new ClassicSingleton();
        }
        return instance;
    }
}
```

Kod ovih implementacija se može dogoditi da u isto vrijeme dođe do pristupa od strane niti i samim tim kreiranja dva različita objekta. Singleton pattern prvenstveno služi da bi se izbjegle takve stvari pa postoji još vrsta implementacije, koje pružaju sigurnost da se klasa koristi sa nitima.

Neke od tehnika jesu korišćenje sinhronizovane metode, dupla provjera zaključavanja klase i takozvani Bill Pugh Singleton.

Ono što je vrijedno pomenuti jeste da korišćenjem refleksije čak i kod ovih implementacija se dešava da se generišu različiti objekti pa u tom slučaju se može koristiti enum singleton koji je malo manje fleksibilan.

Sync method Singleton

```
public class ThreadSafeSingleton {

    private static ThreadSafeSingleton instance;

    private ThreadSafeSingleton(){}

    public static synchronized ThreadSafeSingleton getInstance(){
        if(instance == null){
            instance = new ThreadSafeSingleton();
        }
        return instance;
    }
}
```


Double lock check Singleton

```
public static ThreadSafeSingleton getInstanceUsingDoubleLocking(){
    if(instance == null){
        synchronized (ThreadSafeSingleton.class) {
            if(instance == null){
                instance = new ThreadSafeSingleton();
            }
        }
    }
    return instance;
}
```

Bill Pugh Singleton

```
public class BillPughSingleton {

    private BillPughSingleton(){}

    private static class SingletonHelper{
        private static final BillPughSingleton INSTANCE = new
BillPughSingleton();
    }

    public static BillPughSingleton getInstance(){
        return SingletonHelper.INSTANCE;
    }
}
```

Enum Singleton

```
public enum EnumSingleton {
    INSTANCE;
    public static void doSomething(){
        //do something
    }
}
```

Ovo su samo neki od primjera problema i rešenja vezanih za thread-safe implementaciju. U ovom projektu koristi se Double lock check implementacija za kreiranje objekta za komunikaciju sa bazama podataka.

4.2 Komunikacija izmedju programskih jezika

Za potrebe projekta korišćena su tri programska jezika i to:

- Java

Kao programski jezik koji je platformski nezavistan i open source, što znači da posjeduje mnoštvo dokumentacije i veliku korisničku bazu.

- JavaScript (node.js)

Node.js je jako popularan za kreiranje prototipa i koriste ga mnoge velike kompanije kojima je glavni oslonac neki drugi programski jezik. Facebook koristi PHP/Hack kao glavni programski jezik za back-end¹⁰ na svojim

¹⁰ Back-end: termin koji se koristi kod ljudi koji rade na razvoju softvera i opisuje dio serverski dio funkcionalnosti nekog sistema. Uglavnom su to server, aplikacija(programska logika) i baza podataka.

aplikacijama. U ovom slučaju jednostavan i relativno brz način da se dođe do web sajta sa registracijom, koji se može prebaciti u programski jezik Java ukoliko bi došlo do ozbiljnije i komercijalne upotrebe proizvoda.

- Python

Programski jezik koji postoji oko 26 godina na tržištu i posjeduje veliki broj biblioteka. Kako je jako brz jezik za procesiranje podataka, u ovom slučaju obavlja posao čitanja ARP tabela, tj. korisničkih MAC adresa.

4.2.1 ARP Tabela

ARP ili Address Resolution Protocol je protokol koji se bavi mapiranjem IP adresa sa fizičkim adresama (MAC) uređaja. Sama tabela sadrži dodijeljenu IP adresu uređaju koji je povezan, njegovu fizičku adresu i tip ip adrese. Na ovaj način svaki korisnik koji se registruje biće povezan sa fizičkim uređajem, tako da drugi korisnici neće moći da se prijave na drugo ime, ali će jedan korisnik moći da ima više uređaja kojim se prijavljuje.

4.3 Organizacija baza podataka

Uređaj će da generiše i koristi 2 baze podataka. U prototip verziji implementiraju se SQLite baze podataka. SQLite je dobro rešenje zato što ne zahtijeva server da bi se pokretala. Generisana baza se lako može kopirati i čuvati.

U ovom projektu se nalaze dvije baze podataka i imenovane su PDB (Permanent DataBase) i TDB (Temporary DataBase).

PDB ima 3 kolone i to: Id(Unique ID) – primarni ključ, Mac(String) – Mac adresa i Index(String) – koji predstavlja studentski broj indeksa.

TDB ima 4 kolone i to: Id(Unique ID) – primarni ključ, Mac(String) – MAC adresa, Ip(String) - IPadresa i DBDate(DateTime).

Prilikom rada, TDB će da dobija podatke od python skripte koja čita trenutno stanje mreže, dok će PDB sadržati sve registracije korisnika i čuvati ih. Nakon završetka rada procesiraće se podaci iz baza na osnovu unesenih parametara putem grafičkog interfejsa. Rezultat će biti generisana excel tabela sa podacima o prisustvu u toku rada uređaja.

4.4 Gui (Graphical User Interface)

Kod razvoja grafičkog prikaza pri razvoju nekog softverskog rešenja često je potrebno voditi računa o mnogim stvarima kao što su veličina ekrana, orijentacija uređaja,

skaliranje prozora unutar uređaja. Velika prednost razvoja za nepromjenjivi hardver jeste što su te specifikacije unaprijed poznate. Npr. razvoj igara za neku od konzola i razvoj igara za PC su veoma različiti procesi, jer se hardver jedne konzole ne mijenja u toku vremena tako da prilikom razvoja ne moraju se uzeti u obzir određeni aspekti kao što je to slučaj kod PC igara.

Dimenzije samog grafičkog interfejsa će pokrivati kompletan ekran (320x480 piksela) i neće imati mogućnost zatvaranja aplikacije jer će se na uređaju izvršavati samo java aplikacija. Kao grafička biblioteka se koristi JavaFx koja putem .fxml fajla, koji je zapravo fajl u .xml formatu koji određuje kako će biti raspoređen sadržaj unutar aplikacije kao i koji je tip elemenata koji će sadržati.

4.4.1 JavaFx

JavaFx je biblioteka koja je nastala kao zamjena za postojeću Java Swing biblioteku za pravljenje desktop aplikacija, ali ima i mogućnost da se koristi kao grafički interfejs unutar internet pretraživača. Sama popularnost biblioteke nije velika iz razloga što Java kao programski jezik sve više se koristi kao back-end jezik za programiranje, dok se za sam grafički interfejs i front-end koriste druga rešenja i biblioteke. Kako je u pitanju desktop aplikacija na linux operativnom sistemu u okviru projekta, postoji i mnoštvo drugih rešenja kao što su: SWT, SwingX, Qt, Apache Pivot. JavaFx je izabrano rešenje jer je sadržano u Java programskom jeziku i podržano od strane Oracle kompanije, što omogućava da aplikacija bude platformski nezavisna. Ovo je velika olakšica jer kod razvoja se može koristiti neki drugi operativni sistem.

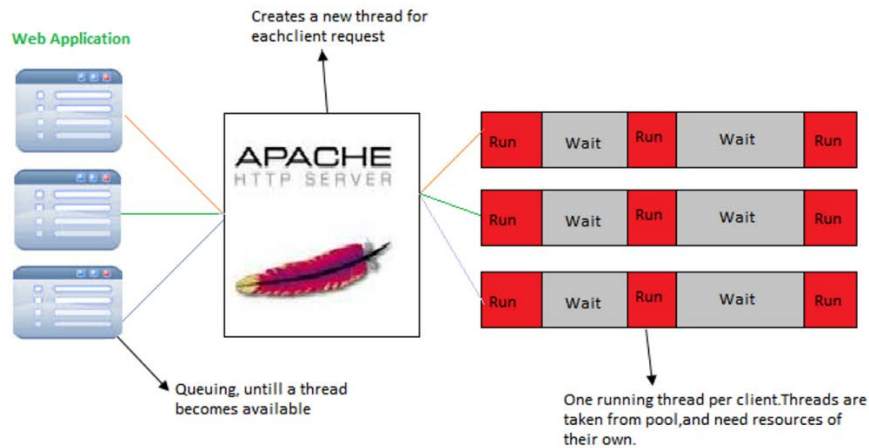
Sam grafički interfejs će sadržati unos za podatke o tome koliko časova treba da prati stanje na mreži, broj časova i dužinu pauza između časova, kao i podatke o predmetu i profesoru koji sprovodi predavanje. Naravno ovo je samo primjer korišćenja ovog sistema u svrhe detekcije prisustva studenata na predavanjima.

| | |
|-------------------|---|
| Run Logging | |
| Class length | |
| 45 | |
| minutes | ▼ |
| Number of classes | |
| 1 | |
| Break length | |
| 0 | |
| Class Name | |
| Testno predavanje | |
| Faculty | |
| Tehnicki Fakultet | |
| Year | |
| 4 | |
| Professor name | |
| Marko Tanasković | |

Ilustracija 16: izgled grafičkog interfejsa

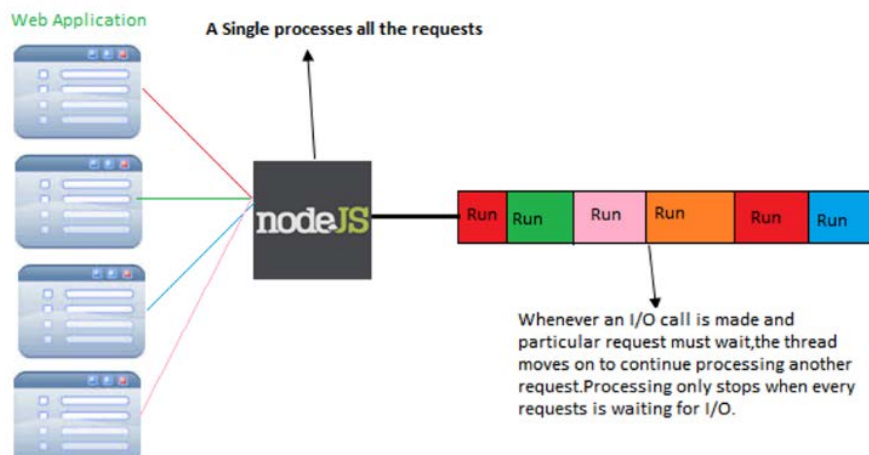
4.5 Node.js server

JavaScript kao jezik je prvobitno bio zamišljen i korišten kao skriptni jezik koji se izvršavao na klijentskim mašinama. On je uz web stanicu i standardni html kod isporučivan nakon čega bi se na korisničkom internet pretraživaču interpretirao. Node.js je open source tehnologija, koja se može pokretati na Windows, Linux i OS X operativnim sistemima. To je runtime sistem za izvršavanje JavaScript koda na strani servera. Node.js se bazira na programiranju zasnovanom na događajima, a funkcionalnosti se organizuju po modulima. Tradicionalno se na serverskoj strani koriste procesi i niti. Svaka napravljena nit je imala svoje resurse na server, dok je web stranica u redu čekanja dok ta nit ne postane dostupna da procesira zahtjev.



Ilustracija 17: Tradicionalan način funkcionisanja http servera (na primeru Apache servera)

Kod Node.js prilikom korišćenja HTTP modula koji omogućava da se izvršava samostalni web server, sve se odvija u jednoj niti. Nakon primanja zahtjeva, prepoznaje se događaj koji uslovljava izvršavanje nekog dijela programa na serveru. Takođe zahvaljujući Google v8 JavaScript Engine-u koji direktno kompajlira JavaScript kod u mašinski kod, omogućava da se izvršava brzo. Nakon dobijanja zahtjeva prolazi se kroz svaki zahtjev i oni koji ne zahtijevaju ništa će jednostavno biti preskočeni, a ostali događaji će se izvršavati.



Ilustracija 18: način funkcionisanja node.js http modula

Node.js aplikacija u ovom projektu ima 4 modula: http, express, body-parser i sqlite3. Http modul nam omogućava postavljanje web stranice na raspberry pi uređaj, dok putem express modula se mogu rutirati zahtjevi. Body parser je modul koji omogućava upravljanje podacima koji dolaze, tj otpakivanje podataka iz zahtjeva i tek kada se verifikuje da su to podaci koji su potrebni prosleđuje se dalje na procesiranje. SQLite3

modul omogućava komunikaciju i upravljanje bazom podataka, a u ovom slučaju će nam biti potrebno povezivanje i korišćenje podataka u dvije već pomenute baze.

4.6 Python skripte

Na pythonu se odvija mali dio programa koji čita arp tabele, za to koristi subprocess biblioteka i arp komanda. Putem ove komande moguće je dodavati, brisati kao i samo izlistavati podatke iz tabele. U ovom slučaju se radi sa wlan0 interfejsom koji je pristupna tačka na uređaju i izlistavaju se parovi ip i mac adrese. Postoji mogućnost da java nit kontroliše pozive putem python-a ili da se pokrene beskonačna petlja u python-u, dok java sluša na output i koristi dobijene podatke.

Kao optimalno vrijeme preuzimanja tabela je 5 sekundi, ali ovaj parametar može biti konfigurisan prilikom daljeg testiranja. Takođe, u našem slučaju nas zanimaju kako novi korisnici koji su se prikačili na mrežu, tako i korisnici koji više nisu na mreži, pa se svakih 60 sekundi briše arp tabela, direktnim pozivom *ip neigh flush* komande. Ova komanda zahtijeva administratorska prava na linux sistemima.

4.7 Migracije podataka

Način upravljanja podacima je jedan od aspekata ovog projekta koji je najviše proširiv. S' obzirom da postoji trajna baza registrovanih korisnika, ovim sirovim podacima je moguće upravljati vrlo lako, prebacivanjem u bilo koji drugi format, a korisnicima može biti zgodno interpretirati podatke u tabelarnom prikazu. Nakon svake sesije java program ima ugrađenu funkcionalnost koja će generisati tabelu sa kompletnim podacima vremena prisustva, kao i vremenima pristupa i odlaska sa mreže. Takođe u odvojenom prikazu će se nalaziti filtrirani prikaz koji će na osnovu unesenih parametara računati vrijeme prisustva kao i broj osvojenih poena, što je prvenstveno namijenjeno prisustvima na predavanjima. Ovo takođe može biti i parametar uspješnosti predavanja, koja ne moraju biti samo fakulteteska.

Kao dodatna funkcionalnost moguće je generisati i prenosive baze podataka sa istim podacima ukoliko je to od važnosti korisniku.

Kao najzanimljiviji aspekt jeste i dopremanje podataka na cloud ili servere ustanove koja koristi uređaj. Tako je moguće dobiti veliki broj podataka od značaja u kasnijoj statistici i organizaciji.

4.7.1 Povezivanje sa Cloud-om

Kako je IoT jedna od tema za koje raste interesovanje u poslednje vrijeme, tako postoji sve više kompanija koje nude servise za povezivanje uređaja, kontrolu uređaja, prikupljanje i čuvanje podataka. Neki od njih su Temboo, UbiDots, NearBus, Carriots, ThingSpeak. Trenutno su u razvoju i servisi Google-a i Microsofta. Povezivanje sa ovim servisima je uglavnom lako i jednostavno, a posjeduju dosta dokumentacije.

Kako uređaj podržava ip forwarding, tj povezivanje putem ethernet, moguće je pisanje aplikacije koja bi se povezivala i na neke od ličnih opcija skladištenja podataka korisnika, kao što su DropBox, Google Cloud, OneDrive i slične. Na ovaj način bi se korisniku dopremili podaci u vidu nekog od poznatih servisa i omogućilo lakše prilagođavanje samog korisnika.

3G moduli su takođe opcija kojom je projekat proširiv, ovo bi omogućilo konstantan protok podataka prema serveru i prikaz podataka u realnom vremenu, a koje ti moduli uglavnom posjeduju. Kako je java aplikacija osnova kojom se upravlja podacima na uređaju, postoji mnoštvo biblioteka za povezivanje sa bilo kojim od pomenutih servisa, a moguće je i pozivanje REST servisa putem Http nativnih java biblioteka, i generisanje samog poziva ka servisu koji bi čuvao podatke.

4.7.2 Razvoj servera pristupa

Ukoliko bi postojala organizacija koja ima veliki broj ovakvih uređaja, njihovo povezivanje bi bilo moguće putem servera koji je privatn i nudi najveći nivo promjena i proširenja u odnosu na druge opcije. U ovom slučaju postojao bi REST servis za opsluživanje poziva svih uređaja i na serveru bi se čuvali i procesirali podaci.

Kao trenutna opcija uređaj posjeduje mogućnost ethernet konekcije koja se može a ne mora proširiti na ostale uređaje koji su povezani na pristupnu tačku. Nije implementirano čuvanje generisanih tabela na neki od servisa, ali je u planu razvoj funkcionalnosti gdje bi se povezivanjem na ethernet automatski slali podaci putem emaila korisniku uređaja.

4.7.3 Generisanje Excel tabela

Excel tabele se generišu na kraju sesije putem java aplikacije. Kao opcije za biblioteke tu su Apache POI ili JXLS. Podaci koji generišu se prave pomoću upita ka bazi

i generisanja instance objekata, pomoću klasa modela koje se nalaze u projektu. Ostatak podataka se uzima od grafičkog interfejsa samog programa.

4.8 Access Point

Realizacija projekta se zasniva na funkciji gdje se ugrađena WiFi antena koristi kao predajnik signala. Ovo je takođe moguće i povezivanjem WiFi predajnika preko usb-a, ali stvara dodatne komplikacije zbog izbora drajvera sa kojim bi sve ispravno funkcionisalo. Iz ovog razloga je najbolji izbor treća vrezija ploče, koja ima ugrađenu antenu, koja može da funkcioniše i kao prijemnik i kao predajnik. O samim specifikacijama je već bilo riječi u poglavlju Hardverska platforma. Kao osnovan alat za ovu funkcionalnost odabran je *hostapd*.

Pri konfigurisanju potrebno je dodijeliti staticku ip adresu uređaju, a kako u poslednjim verzijama Raspbian distribucije linux operativnog sistema za kontrolu interfejsa je zadužen dhcpd (dhcp daemon), neophodno je da u konfiguracionom fajlu ovog alata dodamo komandu kojom će alat ignorisati naš wireless interfejs (wlan0). Nakon toga se u */etc/network/interfaces* dodaje konfiguracija koja dodjeljuje statičku IP adresu našem uređaju.

4.8.1 Hostapd

Nakon potrebnih podešavanja, potrebno je konfigurisati samu pristupnu tačku, tj. alat hostapd koji će biti zadužen za pravljenje naše virtuelne pristupne tačke. Ovaj alat se takođe može koristiti i za kreiranje više pristupnih tačaka na jednoj kartici (uglavnom zavisi od kartice). Takođe u jednoj instanci deamona, ukoliko postoje dva wireless mrežna interfejsa moguće ih je zajedno konfigurisati kao odvojene pristupne tačke.

U ovom slučaju podešena je jedna pristupna tačka za prijavu.

4.9 DHCP/DNS

4.9.1 DHCP

DHCP ili Dynamic Host Configuration Protocol je mrežni protokol koji se koristi u IP mrežama. Sam protokol je kontrolisan od strane DHCP servera koji je zadužen da distribuira parametre širom mreže. Ruteri su uglavnom konfigurisani da rade i kao DHCP serveri. Ukoliko jedna mreža ne bi imala DHCP server bilo bi potrebno da se svaki uređaj na mreži ručno konfiguriše i dodijeli mu se statička IP adresa na mreži. Kada korisnik uključi kompjuter ili se prikači na mrežu on djeluje kao DHCP klijent, koji nakon toga u

mreži šalje broadcast ¹¹ signal (DISCOVER signal) koji zapravo potražuje DHCP server (u tom trenutku klijent ne zna na kojoj je adresi DHCP server, kao ni da li postoji). Ruter tada usmjerava signal na DHCP server, nakon čega server određuje da li postoji slobodna adresa i ukoliko postoji rezerviše adresu za klijenta i njemu šalje odgovor (OFFER signal) koji sadrži informacije o adresi. Kada klijent dobije informacije on odgovara serveru sa REQUEST signalom da namjerava da koristi ponuđenu adresu. Server nakon toga klijentu daje ACK signal čime je dodjela adrese izvršena i sam klijent može da koristi adresu na određeno vrijeme koje je konfigurisano lease vremenom.

4.9.2 DNS

DNS ili Domain Name Server je sistem namijenjen pretvaranju domena u ip adrese. Ono što se dešava kada osoba pretražuje neku web stranicu jeste da se šalje DNS zahtjev za raspoznavanje IP adrese tog web sajta, ukoliko se web sajt ne nalazi na listi već predefinisanih ip adresa u hosts fajlovima koji se nalaze na samom uređaju. Prvi na listi su serveri pružaoca internet usluga i oni uglavnom keširaju na svojim DNS serverima potrebne informacije i odmah se dobija odgovor. Ukoliko nema rekorda koji bi dao odgovarajuću Ip adresu za traženi sajt, upit se prosleđuje root nameserveru koji dalje usmjerava i to čitajući s desna na lijevo i tako nas usmjeravaju na Top Level Domain nameserver koji dalje usmjerava upit na DNS server koji je odgovoran za adrese za taj specifični domen. Nakon toga rekurzivni DNS čuva taj rekord u svom kešu tako da na svaki sledeći upit za tim web sajtom već ima spreman odgovor, a ujedno i dogovara kompjuteru koji je zatražio ip adresu. Svi zapisi imaju TTL – time to live komponentu koja govori koliko taj zapis traje dok nije potrebno ponovo ga osvježiti i vidjeti da li taj web sajt i dalje postoji na istoj adresi.

4.9.3 DNSMASQ

Za ovaj projekat koristi se dnsmasq alat koji je zapravo kombinacija DHCP i DNS servera. S' obzirom da ovaj alat koristi relativno malo resursa, veoma je pogodan za ovakav projekat, a sama konfiguracija je jednostavna putem dnsmasq.conf fajla u kome su navedeni rasponi dhcp adresa koje će se dodjeljivati kao i njihov lease time, tj vrijeme

¹¹ Broadcast: u kontekstu mrežnih sistema označava metodu prenosa poruke svim korisnicima u mreži istovremeno.

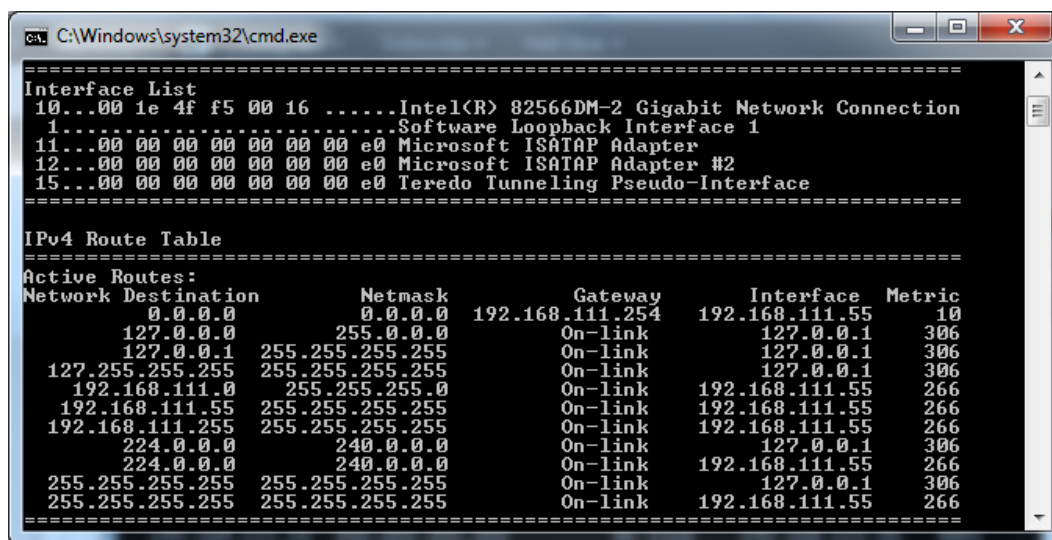
na koje se adrese dodjeljuju. Takođe konfigurisana je i adresa web servera koja se otvara kao predefinisana, da bi se izvršila registracija.

4.10 Ip forwarding

Rutiranje IP adresa je skup metodologija kojima se paketi kreću putem Internet protokola. Tu se podrazumijeva biranje pogodne putanje paketa od odredišta do destinacije, a zaduženje rutera jeste da bude mrežni prolaz koji će usmjeravati pakete podataka. Prosleđivanje IP adresa je algoritam koji uzima u obzir veličinu svakog paketa, tip servisa koji je naveden u zaglavlju paketa koji se prosleđuje, kao i karakteristike dostupnih putanja prema drugim ruterima u mreži. Tu se takođe nalaze i podaci o kapacitetu mrežne putanje, nivou iskorištenja i maksimalna veličina datagrama (paketa koji može biti poslat) koja je podržana. Većina softvera za rutiranje radi po principu najmanje putanje, ali postoje i drugi protokoli za rutiranje koji računaju i druge parametre za određivanje najbolje putanje. Kada se govori o rutiranju bitno je pomenuti tabele rutiranja.

4.10.1 Tabele rutiranja

One čine osnovu rada rutera i samog procesa rutiranja. U toj tabeli se nalaze informacije potrebne da bi paket stigao na odredišnu adresu.



```

C:\Windows\system32\cmd.exe

Interface List
10...00 1e 4f f5 00 16 .....Intel(R) 82566DM-2 Gigabit Network Connection
11...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
12...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
15...00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface

IPv4 Route Table
Active Routes:
Network Destination    Netmask          Gateway          Interface        Metric
0.0.0.0                0.0.0.0          192.168.111.254  192.168.111.55   10
127.0.0.0              255.0.0.0        On-link          127.0.0.1        306
127.0.0.1              255.255.255.255  On-link          127.0.0.1        306
127.255.255.255        255.255.255.255  On-link          127.0.0.1        306
192.168.111.0          255.255.255.0    On-link          192.168.111.55   266
192.168.111.55         255.255.255.255  On-link          192.168.111.55   266
192.168.111.255        255.255.255.255  On-link          192.168.111.55   266
224.0.0.0              240.0.0.0        On-link          127.0.0.1        306
255.255.255.255        255.255.255.255  On-link          127.0.0.1        306
255.255.255.255        255.255.255.255  On-link          192.168.111.55   266
  
```

Ilustracija 19: tabela rutiranja

Osnovne informacije koje se nalaze u tabeli su one od udaljene ili susjedne mreže kao i interfejs rutera ili adresa rutera preko kojeg se dolazi do te mreže. U tabeli se nalaze statičke i dinamičke rute. Statičke su one koje se unose ručno u tabele od strane

administratora, dok su dinamičke sve ostale koje se popunjavaju radom protokola za rutiranje.

4.10.2 Implementacija

U našem slučaju kako imamo dva mrežna interfejsa na ploči, i to: ethernet i wifi, gdje se wifi koristi kao pristupna tačka za našu implementaciju, dok se ethernet može koristiti za povezivanje na spoljnu mrežu sa pristupom internetu. U slučaju kada mi želimo tu internet konekciju da podijelimo sa korisnicima koji su prikačeni na našu pristupnu tačku potrebno je da podesimo prosleđivanje paketa. U linux operativnom sistemu to se može izvršiti komandom iptables i za to su potrebna administratorska prava.

Prvo je potrebno omogućiti da prosleđeni paketi budu prihvaćeni od strane firewall-a i da privatne ip adrese komuniciraju sa spoljnim adresama, a takođe i maskirati lokalnu ip adresu u spoljnu kada izađe iz naše mreže. Ovo znači da će korisnikova adresa dobiti onu adresu koju ethernet ima od strane druge mreže na koju je prikačen. Ovo se radi putem sledeće komande:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Druge dvije komande će se zapravo baviti prosleđivanjem tj praćenjem kod koga i od koga paket ide.

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

```
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

4.11 Automatizacija

Automatizacije cijelog procesa instalacije je izvršena pisanjem Shell skripte koja nakon pokretanja preuzima sve potrebne alate, podešava pristupnu tačku, kao i dhcp, dns server. Takođe je moguće dodati i podešavanja za prosleđivanje ip adresa, ali to je funkcija koja zavisi od namjene uređaja. Takođe java aplikaciju je moguće dodati u instalaciju putem skripte i konfigurisati sistema da prilikom pokretanja sistema na grafičkom interfejsu se prikazuje samo ova aplikacija, čime ovaj uređaj dobija izgled komercijalno upotrebljivog.

5 Verifikacija

Verifikacija rada sistema je izvršena u laboratorijskim uslovima, a planirano je i testiranje sistema u realnim uslovima. Testiranje je podrazumijevalo praćenje stanja baza podataka u realnom vremenu kao i toka samih niti programa, praćeno konekcijama uređaja na sistem i promjenjivim prisustvom na konekciji, kao i testiranjem višestruke registracije. Koraci koji su korišteni pri testiranju sistema su sledeći:

- Testiranje postojanja WiFi mreže
- Testiranje mogućnosti povezivanja sa WiFi mrežom
- Verifikacija prikačenog uređaja, dodijeljene IP i lične MAC adrese
- Mogućnost navigacije na web stranicu kada sistem (java program koji pokreće niti) nije aktivan
- Testiranje izlaza na internet, kada je omogućen routing i kada je isti isključen
- Testiranje pokretanja sistema putem grafičkog interfejsa u nedostatku potrebnih podataka
- Testiranje pokretanja sistema sa svim podacima upisanim
- Testiranje pristupa web stranici kada je sistem pokrenut
- Pregledanje podataka kada korisnik nije registrovan
- Registracija, kao i pokušaj višestruke registracije
- Pregledanje podataka kada je korisnik registrovan

6 Zaključak

Rad na projektu je donio dosta zanimljivih prepreka i mogućnosti za istraživanjem alternativa, ali se pokazao kao dobro rešenje u moru drugih koji zahtijevaju veći stepen interakcije korisnika. Jedini zahtjev kod ovog sistema jeste postojanje uređaja koji se vezuje na mrežu. Problemi su bili iz oblasti Internet Of Things-a koja zavrijeđuje sve veću pažnju trenutno na tržištu. Cilj samog projekta je bio funkcionalan prototip uređaja, kao i ispravno funkcionisanje svih softverskih sklopova. Kako se rad bazirao na korišćenju git kontrole verzije, problemi su se javljali u vidu rada na mrežnom dijelu projekta koji je često činio da sam uređaj bude bez internet konekcije, dok je postojala potreba da se kod koji se razvija na uređaju uskladi sa master granom. Sam problem je prevaziđen radom na uređaju i čestim lokalnim čuvanjima promjena a nakon toga usklađivanje nakon uspostavljanja prosleđivanja portova. Kao neisplativo rešenje pokazalo se korišćenje ekrana za uređaj, kako na postojanje web stranice koja bi mogla da podrži administratorski dio sa istim funkcionalnostima tako i kod grafičkog interfejsa na samom uređaju.

Prilikom testiranja čitanja ARP tabela pokazalo se da se ne može pouzdano znati samo čitanjem tabela da je korisnik napustio mrežu u nekom trenutku zato što se upis u tabeli ne briše nakon što klijent napusti mrežu. Rešenje se sastojalo u čišćenju arp tabele, tako da sledeće čitanje ima samo one korisnike koji su trenutno prikačeni na mrežu. Na taj način moguće je praćenje i dolaska i odlaska klijenata u okviru mreže.

7 Literatura

1. Veinović M. i Jevremović A. (2011) Računarske Mreže, Univerzitet Singidunum, Beograd
2. Veinović M, Šimić G., Jevremović. i Franc I. (2013) Baze Podataka, Univerzitet Singidunum, Beograd
3. Veinović M. i Jevremović A. (2013) Internet Tehnologije, Univerzitet Singidunum, Beograd
4. Horton I. (2004) Ivor Horton's Beginning Java 2, JDK 5 Edition, Wiley Publishing Inc., Indianapolis, Indiana
5. Bates B., Sierra K., Freeman E., Robson E. (2009) Head First Design Patterns A Brain-Friendly Guide, O'Reilly Media, Sebastopol, California
6. Stephens R. (2015) Beginning Software Engineering, Wiley Publishing Inc., Indianapolis, Indiana
7. Syed B. (2014) Beginning Node.js, Apress, New York City, New York
8. SourceMaking, Design Patterns Explained Simply, <https://sourcemaking.com/design-patterns-ebook>
9. JournalDev, Pankaj Kumar, Java Singleton Design Pattern Best Practices with Examples <http://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples>
10. Raspberry Pi Foundation, „Raspberry pi Documentation“ <https://www.raspberrypi.org/documentation/>
11. Raspberry Pi Foundation, „Configuring Raspberry Pi As Access Point“ <https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md>