WiFi Presence Logger

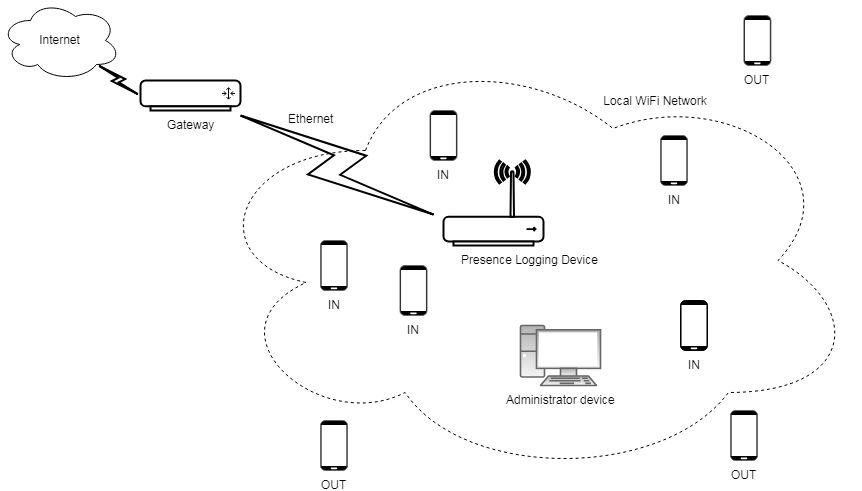
# Uvod

WiFi Presence Logger je servis za evidenciju prisutnosti WiFi uređaja u određenom prostoru uz pomoć WiFi tehnologije. Svaki uređaj (smartfon, laptop, ili neki drugi klijent sa podrškom za WiFi) koji se poveže na mrežu našeg Loggera (u daljem tekstu **WiFiPL**) biće evidentiran u realnom vremenu.

To podrazumeva tri glavne funkcionalnosti: evidenciju vremena povezivanja na mrežu (**ulaska**),vremena diskonektovanja sa mreže (**izlaska**) WiFI uređaja (u daljem tekstu **klijenata** ) u bilo kom trenutku. Na osnovu jedinstvene MAC adrese klijenta, moguće je jednoznačno utvrditi kretanje klijenata.

Druga funkcionalnost je registracija korisnika. Pod pretpostavkom da svaki korisnik ima jedan uređaj koji koristi (uglavnom smartfon), registracijom na WiFiPl uređaju, svakom jedinstenom klijentu dodeljujemo realno ime i prezime korisnika. Na taj način, imamo informaciju o aktivnostima korisnika u realnom vremenu u određenom prostoru.

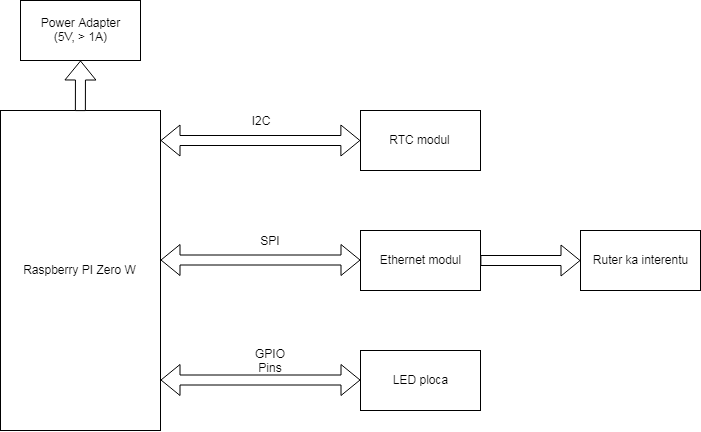
Treća funkcionalnost WiFiPL uređaja je pristup internetu korisnicima. WiFiPL uređaj se ponaša kao mrežni Access Point. Svim klijentima koji su povezani na njegovu WiFi mrežu omogućava prosleđivanje internet saobraćaja na Gateway, pod uslovom da je WiFIPL povezan ethernet kablom na ruter koji je Gateway.



Dijagram 1: prikaz funkcionisanja WiFiPL mreže

# Hardverski opis WiFiPL-a

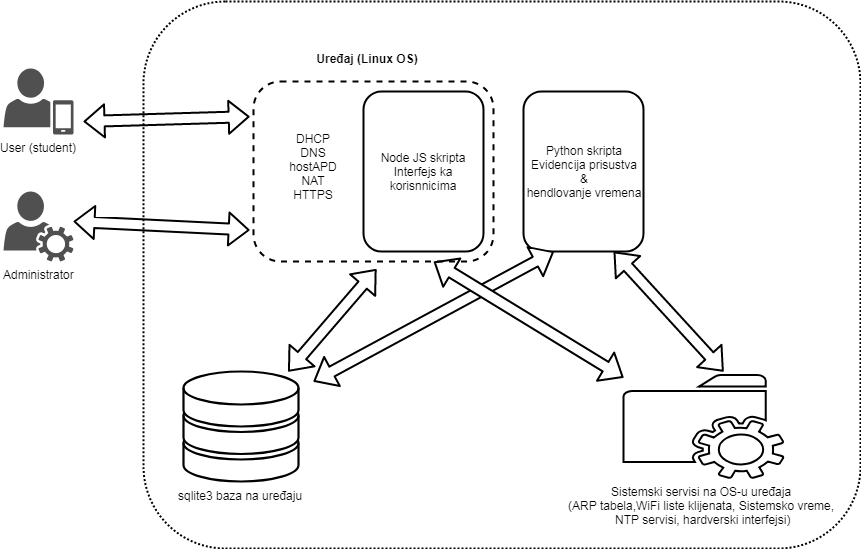
Glavni kontroler WiFiPL-a je Raspberry PI Zero W. Njegov operativni sistem je Raspbian Stretch Lite. Preko SPI interfejsa, kominicira sa Ethernet modulom je **ENC28J60.** Preko I2C modula, komunicira sa **DS3231** Real Time Clock-om. Sa pinovima GPIO 13,16,19,20,21,26 upravlja sa RGB diodama. O detaljima povezivanja, pogledati dokument „install upustvo“.



# Opis softvera WiFiPL-a

Dva glavna elementa softverskog resenja su

1. Evidencija prisutnosti klijenata koji su trenutno u mreži
2. Interfejs ka korisnicima radi registracije ili očitavanja podataka sa uređaja



Ilustracija 1: prikaz organizacije softvera

Evidencija prisustva se vrši periodično, tako da se u svakoj instanci uzimaju podaci iz sistemskih servisa na Raspbianu. Iz njih se saznaje koji su klijenti trenutno unutar mreže (koji su povezani) preko MAC adrese. Upoređujući prethodnu instancu, saznajemo koji su klijenti novi, a koji ne. Promene se beleže u bazu podataka.

Takođe, ovaj deo koda vrši i sinhronizaciju vremena na operativnom sistemu, vršeći podešavanja u zavisnosti od situacije i trenutnih podešavanja.

Ovaj deo je napisan u pythonu, i izvršava odmah nakon podizanja operativnog sistema.

**run.py** vrsi ovaj deo posla, uz restauraciju sistemskog vremena. Unutar run.py, periodicno se pokrece skripta **station\_check\_instance.py**  koja neposredno proverava ko je „usao“ a ko „izasao“ iz mreze.

Detaljni opis toga je dat u komentarima koda.

Interfejs ka korisnicima se vrši preko TCP/IP protokola. Na WiFiPL kao servis postoji i DHCP i DNS serveri, tako da svaki klijent koji je povezan na WiFiPL i nad kome se vrši evidencija, takođe ima dodeljenu ip adresu i može vršiti komnikaciju sa uređajem.

Zato, na WiFiPL-u postoje dve aplikacije preko kojih se ostvaruje interfejs prema korisnicima.

**Prva aplikacija** služi za registrovanje korisnika. Na taj način se povezuju Ime i Prezime korisnika sa njegovim jedinstvenim uređajem (njegovom MAC adresom).

Ip adresa WiFiPL-a je u stvari adresa default gateway-a, u ovom slučaju, to je 192.168.4.1 (mreža je 192.168.4.0/24), dok je aplikacija za registraciju pokrenuta na portu 80 (http server).

Svi klijenti mogu pristupiti toj aplikaciji kucajući u brauzer: <http://192.168.4.1>

Tu, pristupaju stranici za registraciju ili ako su već registrovanji, podacima koji su uneli prošli put. Postoji mogućnost i editovanja tih podataka.

Ovoj aplikaciji se može pristupiti i sa adresom <http://prijava.prisustva> , jer na WiFi presence loggeru radi i DNS servis, koji uparuje default gateway sa nazivom koji smo mi dali (prijava.prisustva)

**Druga aplikacija** je za interfejs ka administratorkom korisniku. Taj korisnik je privilegovan, jer može čitati, brisati, editovati podatke koje je uređaj do sad pohranio u bazama. Takođe može vršiti podešavanja samog uređaja. Port ove aplikacije je 3000, tako joj se pristupa sa <http://192.168.4.1:3000/imeMetode> .

Ova aplikacija ima definisane metode preko kojih se vrše sve definisane akcije administratora (imeMetode). Tip aplikacije je https.

Obe aplikacije imaju pristup sistemskim servisima kao što su arp (radi uparivanja MAC – IP adresa) i bazama radi čitanja/upisa/ispisa itd...

One su odrađene u NodeJS skripti.

Oba dela programa se pokreću kao dva odvojena procesa u linuxu, jer moraju da se paralelno izvršavaju. Oni se pokreću u vidu skripti, prvi u python skripti, a drugi u NodeJS skripti.

# Tok izvršavanja programa

# C:\Users\Korisnik\Documents\Untitled Diagram3.png

Ilustracija 2: tok celog softvera

Na slici je prikazan ceo tok programa. Plavom bojom je označena evidencija prisustva klijenata u WiFi mreži (izvršava se na pythonu). Crvenom bojom je označen interfejs uređaja (NodeJS). Zelenom bojom su označene sve shell skripte i komande linuxa koje im kao programer kontrolišemo, dok su sivom označene sve radnje koje se rade interno, automatski na linuxu i koje mi ne diramo.

1. Podizanje sistema se dešava prilikom priključenja uređaja na napajanje. Bootovanje traje 30-ak sekindi na Raspberry PI Zero-u. Radnja je automatska i mi ne utičemo na nju.
2. Pokretanje hostapd, dns, dhcp i ntp servisa se radi automatski na kraju podizanja sistema. Prilikom instalacije svih tih servisa, default podešavanja se ne diraju (vidi install upustvo).
3. Port forwarding je komanda linija koja preslikava lokalni mrežni domen koji se nalazi na wifi interfejsu (wlan0) na interfejs ethernet-a (eth0), tj saobraćaj koji nije adresiran na ip adrese unutar wlan0 mreže (mreža je 192.168.4.0/24) proseđuje se preko ethernet modula na spoljni saobraćaj (ako je WiFiPL povezan na internet preko rutera). Ona se pokreće unutar **rc.local** skripte koja se uvek pokreće na kraju bootovanja.
4. Unutar rc.local skripte, paralelno se pokreću dva procesa, za python i nodejs skriptu. Nakon toga, imamo dva procesa koji se izvršavaju sve dok radi uređaj. Naziv ovih skripti su **run.py** i **app.js** (unutar newNode foldera)
5. Python skripta  
   i) Uređaj uvek ima upisano poslednje vreme kada je radio validno,tako da u slučaju da je iskljucen neplanirano, prilikom prvog sledećeg pravilnog boota, učita poslenje validno vreme koje je zapisano u txt fajlu (**shutdown\_timetxt**).  
   ii) Znajući poslednje validno vreme, proverava se da li je neki rekord u bazama ostao nepopunjen, ako jeste, popunjava ga sa poslednjim validnim vremenom  
   iii) Učitava validno sistemsko vreme. To radi tako što proverava internet konekciju. Ako je ima, uzima globalno vreme sa ntp servera i koriguje hardverski RTC, a ako nema internet konekcije, u sistemsko vreme linuxa upisuje vreme sa RTC-a.  
   Nakon ovih inicijalnih radnji, koje se radi samo nakon podizanja sistema, skripta ulazi u loop, u kome ostaje dokle god radi uređaj. Unutar njega, na svakih 30s upisuje poslednje validno vreme kada je uređaj validno radio, pokreće instancu provere klijenata, što je pzthon skripta za sebe i pauzira na 30s.
6. NodeJS skripta. Prilikom pokretanja glavne JS skripte, podižu se dve aplikacije, **app** na portu 80, i **api\_app** na portu 3000. Nakon pokretanja, čekaju na tim portovima na zahteve od klijenata.

U zavsnosti od zahtevane metode, izvršava se određena radnja i vraća se rezultat. u **app** aplikaciji, vrši se registracija korisnika, i to prvo proverom da li je uređaj registrovan, provera arp tabele radi uzimanja MAC adrese uređaja i upis/adpejt tabele registrovanih. U api\_app aplikaciji, definisan je set metoda (dve verzije metoda, set i get) kojima pristupa administrator, na osnovu tokena. Radnje tih metoda su čitnje/pisanje/brisanje podataka iz baza, podešavanje vremena, WiFi podešavanja, itd..

# SQLite3 Baza Podataka

Čuvanje podataka se vrši korišćenjem SQLite3 baze podataka. Svi podaci Na uređaju su smešteni u dva .db fajla:

1. logBase.db
2. regBase.db

**logBase.db** služi za evidenciju prisutnosti klijenata. Za svaki dan u kome uređaj radi, kreira se nova tabela, sa nazivom u formatu: **TDD\_mm\_YY**

Format tabele je |Id (Primary key, unique, integer)|Host (Text)|Mac (Text)|Ip (Text)|Ulaz (Text)|Izlaz (Text)

**Id** je priamrni kljuc

**Host** je ime kljenta koje može da se dobije na osnovu mac adrese u arp tabeli (ne koristi se za sad)

**Mac** – MAC adresa, jedinstveni indetifikacioni podatak po kome jedinstveno određujemo uređaj

**Ip –** IP adresa u arp tabeli, u krajnjoj verziji je uvek na 1.1.1.1 jer se evidencija više ne vrši iz ARP tabele

**Ulaz:** Vreme kada je klijent pristupio mreži u formatu: **YYYY-MM-dd HH:mm:ss.dddddd**

**Izlaz:** Vreme kada je klijent pristupio mreži u formatu: **YYYY-MM-dd HH:mm:ss.dddddd**

**regBase.db** pamti podatke registrovanih klijenata.

Tabela regList povezuje Mac adresu i Ime korisnika, format tabele je: |RegId (Primary key, unique, integer) | Mac (Text) | Ime (Text) | Prezime (Text) | Id (Text)

**RegId** – Primary key

**Mac** – MAC adresa uredjaja

**Ime/Prezime/Id –** Podaci o korsniku koji poseduje uredjaj sa zadatom MAC adresom

Tabela typeTable daje podatke o tipu korisnika (u ovom slucaju Student i profesor) i preko stranog kljuca je jedinstveno preslikana u regList tabelu.

Format je |typeId (integer PRIMARY KEY NOT NULL UNIQUE)| Type(Text) | UserId FOREIGN KEY references regList(RegId))|

**typeId** – primarni kljuc

**Type** – tip korisnika

**UserId** – strani kljuc, referenciran na primarni kljuc tabele regList (**RegId**)